Windafile

TOP

SE11 - DDIC.
SE21 - Package
SE38 - Reports

Part A

- R/3 Architecture (L1)
- Intro to ABAP Programming (L2)
- DDIC - I (L3)
- Selection Screen (L4)
- Command Control Statement (L5)
- Internal Tables - I (L6)
- DDIC - II (L7)
- Internal Tables - II (L8)
- String operations (L9)
- Reporting Events (L10)
- Modularization Techniques (L11)
  - → (Sub Routine)
  - → Function Module (L12)

Part B

- Module Pool Programming (L1)
- OO ABAP (L2)
- ABAP List View (L3)
- File Handling (L4)

Part C

- BDC [Batch Data Comm] (L1)
- Smartform (L2)
- Adobe Forms (L3)
- Enhancements (BADI) (L4)
- Performance Tuning (L5)
- Transports (L6)

Scanned with CamScanner

SAP HANA

Intro

ABAP -> HANA

Native SQL

Calculation View - Scripted

ADBC

CDS (Core Data Services

ALV with IDA

Open SQL Revisited

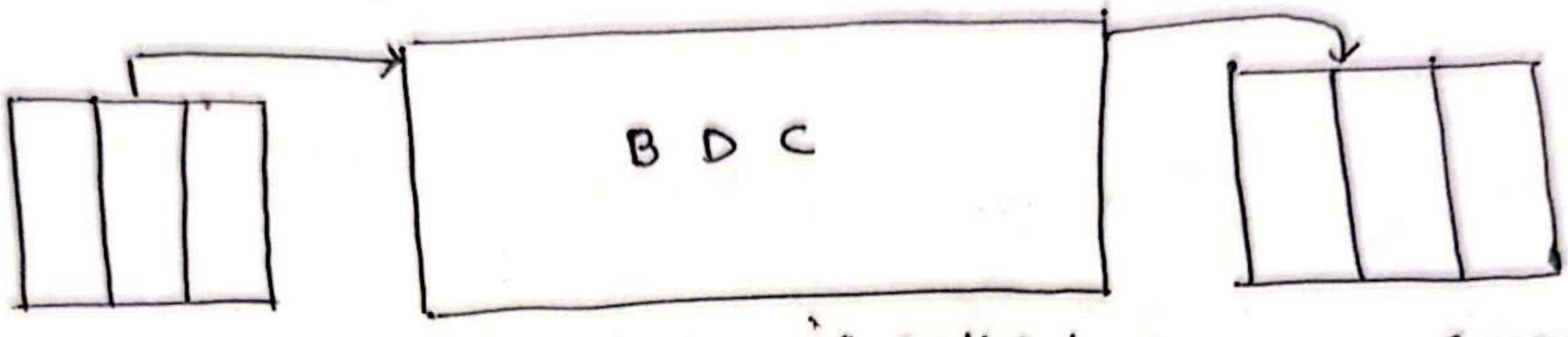
ABAP New Syntax

AMDP (Managed DB Procedure)

Scanned with CamScanner

SAP ABAP (Batch Data communications).

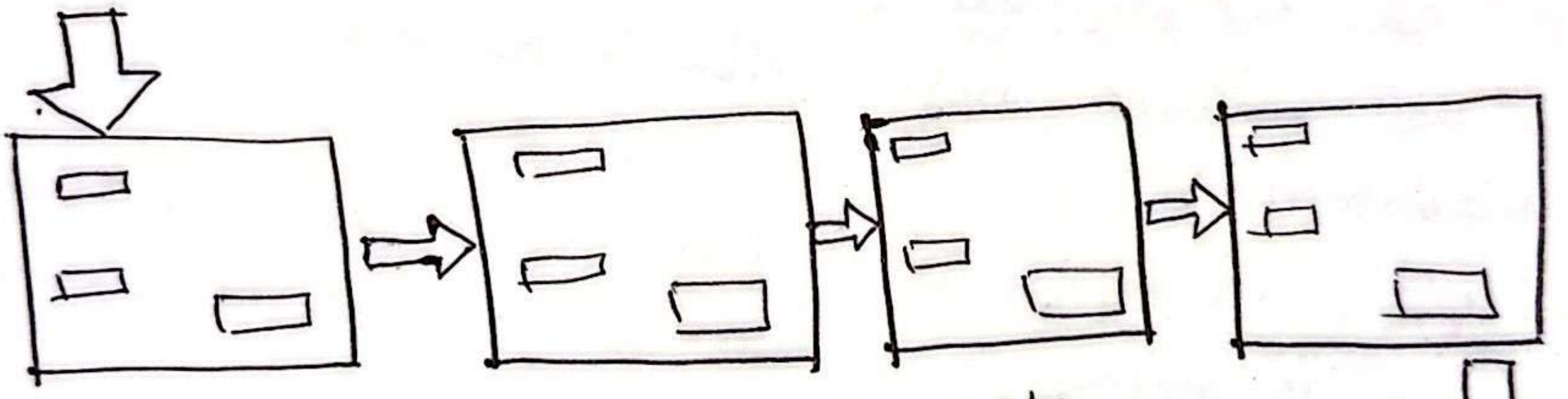
SAP should have a provision to transfer data from Non SAP system to SAP system.



Any Non SAP system is called Legacy System.

Data Transfer is used when ⇒

1) Transfer data from an external system into an R/3 system as it is installed.
2) Transfer data regularly from an external ~~program~~ system into an R/3 system.



User action in every screen gets recorded.
Instead of passing data with same process we save the process & pass the value on each run.

BDC

1) Programming technique for loading data into SAP.
2) Simulates the act of a user entering data into an SAP transaction.
3) System picks the data from DB using ABAP/4 program and feeds it to the SAP system using the corresponding transaction screen by screen.
4) The programmer can choose whether the ~~tran~~ transactions ~~sou~~ should run immediately or at later time.

Scanned with CamScanner

BDC - Data transfer can be two way.

1) Outbound

SAP → SAP
SAP → NonSAP

2) Inbound

SAP → SAP.
NonSAP → SAP

Three ways to transfer the data into SAP from any external system →

1) Sessions method
2) Call Transaction.

] — we talk about these two only. These are called BATCH INPUT SYSTEM.

3) Direct Input

Advantages of Batch Input. →

1) Processes large volumes of data in batch.
2) Can be planned & submitted in the background.
3) No manual interaction is required when data is transfered.
4) Data Integration is maintained as whatever data is ~~doing~~ transfered to the table is through transaction.

Steps Involved in writing Data Transfer Program

1) Analyzing Transaction.
2) Declaring Internal Tables
3) Transfer Data from flat files into internal Tables.
4) Population of BDC Data into Internal Table.
5) Looping internal ~~data~~ table to any of the data transfer methods.

Scanned with CamScanner

SE16N - General Table Display.

Steps involved in writing a Data Transfer Program

1) Analyzing Transaction.
2) Declaring Internal Tables.
3) Transfer Data from flat file into Internal Tables.
4) Population of BDC data into Internal Table.
5) Looping internal table to any of the data transfer methods

BDCDATA is a structure which is used to transfer data.

Analyzing Transaction

1) Name, Type & length of the field
2) Screen Number & Name of the module pool program.
3) Determine the mandatory fields.
4) Determine the fields which accept standard values
5) Determine if data conversion is required when converting to SAP format.

Declaring Internal Table

3 internal tables to be declared.

Internal table with structure similar to

1) The flat file.
2) DDIC structure BDCDATA.
3) DDIC structure BDCMSGCOLL.

Transfer Data from Flat file to Internal Table

1) Depends on the source of the file.
2) The flat file can be in
   - -> Presentation server
   - -> Application server.

Scanned with CamScanner

Depending on the server the programming is done in ABAP to upload Data from flat file to internal table.

If the flat file resides on the presentation server the function module GUI_UPLOAD is used.

Important parameters of GUI_UPLOAD are ⇒

1) Filename.

2) Filetype.

3) Has_field_separator.

4) Data_tab.

After uploading data to internal table, the BDC DATA internal table is filled with values required to process single record.

{SHDB ⇒ Transaction code R.for Recording Transaction}

→ Click on New Recording.

→ give Recording name → Zb1_mm01.

give Tcode → mm01

→ click continue.

LOOP.

CALL TRANSACTION ⇒

(CALL TRANSACTION <tcode> using IT_BDCDATA MESSAGES INTO IT_BDCMSG CALL)

SESSION TRANSACTION ⇒

Insert the IT_BDCDATA into a session by calling the Function Module → BDC_insert & is processed.

Scanned with CamScanner

Q) why its called session?

Because it first creates a session and then it creates a transaction.

SM35 => Tcode for processing transaction.

→ Create / Open a Session
→ Process the session
→ Close the session.

} So we need 3 Function Modules.

CALL TRANSACTION METHOD:

conversion prog uses the ABAP statement CALL TRANSACTION using to run an SAP transaction.

External data does not have to be deposited in a session for later processing.

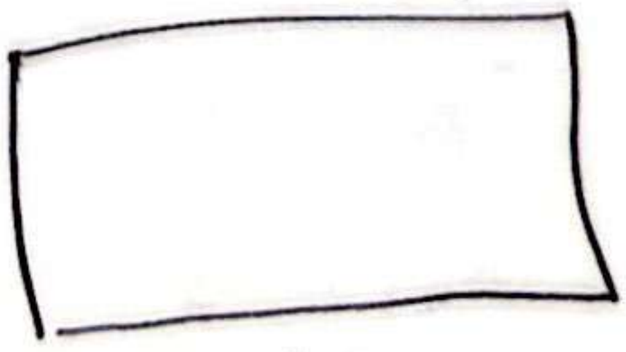
The entire batch i/p process takes place inline in the program.

CALL TRANSACTION - is faster b/w two data transfer methods as it is a 1 Step process.

Legacy data is processed inline in the data transfer program.

```
CALL TRANSACTION <tcode>
    USING <bdc_tab>
    MODE <mode>
    UPDATE <update>
MESSAGES INTO
    <BDCMSGCOLL_TAB>.
```

Display Modes
A, N, E



A → call screen
N → NO screen
E → Errors Only.

Scanned with CamScanner

<tcode> : Transaction Code.

<bdc-tab> : Internal table of structure BDC DATA.

<mode> : Display mode.

<update> : Update mode → Two values (A, S)
A → Asynchronous
S → Sync.

A → In Async, the system doesnt wait for (succes) status of any record.

<BDC MSG COLL. TAB> – collects msgs generated during process. Its an internal table of (BDCMSGCOLL)

BDC Table

Field	Type	Length	Desc
Program	char	40	
DynPro	Num	4	
DynBegin	char	1	
Fnam	"	132	
Fval.	"	132	

BDC → After table has been built, it has to be submitted for SAP processing.

Display Mode specifies whether the data transfer processing should be displayed as it happens. There are 3 modes :).

"A" →

"N" →

"E" →

Scanned with CamScanner

Update Mode - specifies how updates projected by a transaction should be processed. Three ways are available ⇒.

A → "Asynchronous updating" →

S → 'synchronous updating' →

L → 'Local Updating'. →

BDC Programming ⇒ (Using Recording)

1) select your recording
2) Click on program.
3) choose → Transfer from Recording.
4) give Title / Description.
5) click on Source code.
6) go to last → Copy these two (BDC Dynpro & BDC Field)
→ Remove all other FM's.
7) comment 1st include.
8) bdcdata → wa - bdcdata.

Scanned with CamScanner

DATA: it_mara TYPE STANDARD TABLE OF ty_mara,
wa_mara TYPE ty_mara.

DATA: it_bdcdata type table of bdcdata,
wa_bdcdata Type bdcdata.
lv_file Type IBIPPARM(*)_path,
Types: Begin ty_mara,

mbrsh Type mbrsh, MAKTX TypeMAKTX,
MTART Type MTART, meins Type meins,
End of ty_mara. Matkl type Matkl

Q) How to know which fields?

From the recording that I ~~fie~~ filled. All the things that we passed has to be declared. These things should come from the file.

(*) parameters: p_file type filename - fileintern

at selection- screen on value-Request for p-file.

call function 'F4_Filename'

Importing

file_name = lv_file.

p_file = lv_file.

1st pdf

Record, Process, Program, call Transaction, Session, BDC, (-⊖ Prog).

2nd Day & 3rd Day

BDC

4th Day

Enhancement & Exit & Smart forms.

1) Module Pool Programming.
2) OOP ABAP.
3) ABAP (ALV) List viewer
4) File Handling.

Scanned with CamScanner

SMART FORMS

Scripts - Outdated

T-Code : ~~Smartforms~~ SMARTFORMS (All caps).

Scenario to develop a Smartform for Tables

Airtel - 10 cr Customers.

Forms - Print program.
(Heirarchy of Creation) * *
Page -> Windows -> Page windows. (we will create).

Default Page : Page 1.

Default window : Main. / Header / Footer / Logo.

Text Elements.

Scenario (Security Dept).

warehouse -> Mars chocolates -> Truck -> Storage Locatio

whenever Truck is movingout -> we need to g
list of trucks

Security Dept -> Printout with list of material.

warehouse -> Mars chocolates -> Truck -> Storage Location -> Table -> MARD. (Std Table).

(*) we will print List of all data.

Header
To
Main
From
Footer.

All the things we gonna get from the Page.

These are all the windows we need to Create

-> Then we populate this window data

Scanned with CamScanner

we will have → Static Data - Dynamic Data.

[Structure, Table Type] → we have to create one in SE11
(Global)

[FOR HARD TABLE]

246117206LFB1.

Step 1 Create a structure using SE11.

Go to Datatypes give structure Name → select Structure → Save → Activate.

Global Structure & Global Datatype.

Step 2 Create a Table type using SE11.

Go to Datatype → give Table type name → select Table type [ZITAM_HARD]

In Line Type → give the name of structure created

[i.e ZTAM_HARD]

Save → Activate.

Step 3 Go to Smartforms

Tcode - SMARTFORMS.

we will design layouts out.

FORM → ZDXC - HARD - sf. → Create

To see/hide Layout → Click on Form Painter.

Go to Form Attributes → Output Option

Here we see size of the page.

DIN A4 → A4 size.

Scanned with CamScanner

a) go to FORM INTERFACE. → Tables.

Parameter Name	Parameter Type Types.	Associated Type.
IT_MARD		ZITAM_MARD (the Table type we created)

(*) Its Mandatory to give Associated Type.

b) Go to GLOBAL DEFINITIONS

Variable Name	Type Assignment Types	Associated Type.
WA_MARD		ZTAM_MARD (structure we created).

Save (4611 7206 P)

After saving we can see → 1) Default Page
% PAGE1 New Page
→ MAIN WINDOW

c) Click on New Page → select Create → Windows

LOOP 1
Window HEADER
Meaning HEADER. WINDOW

Save.

LOOP1
Only change.

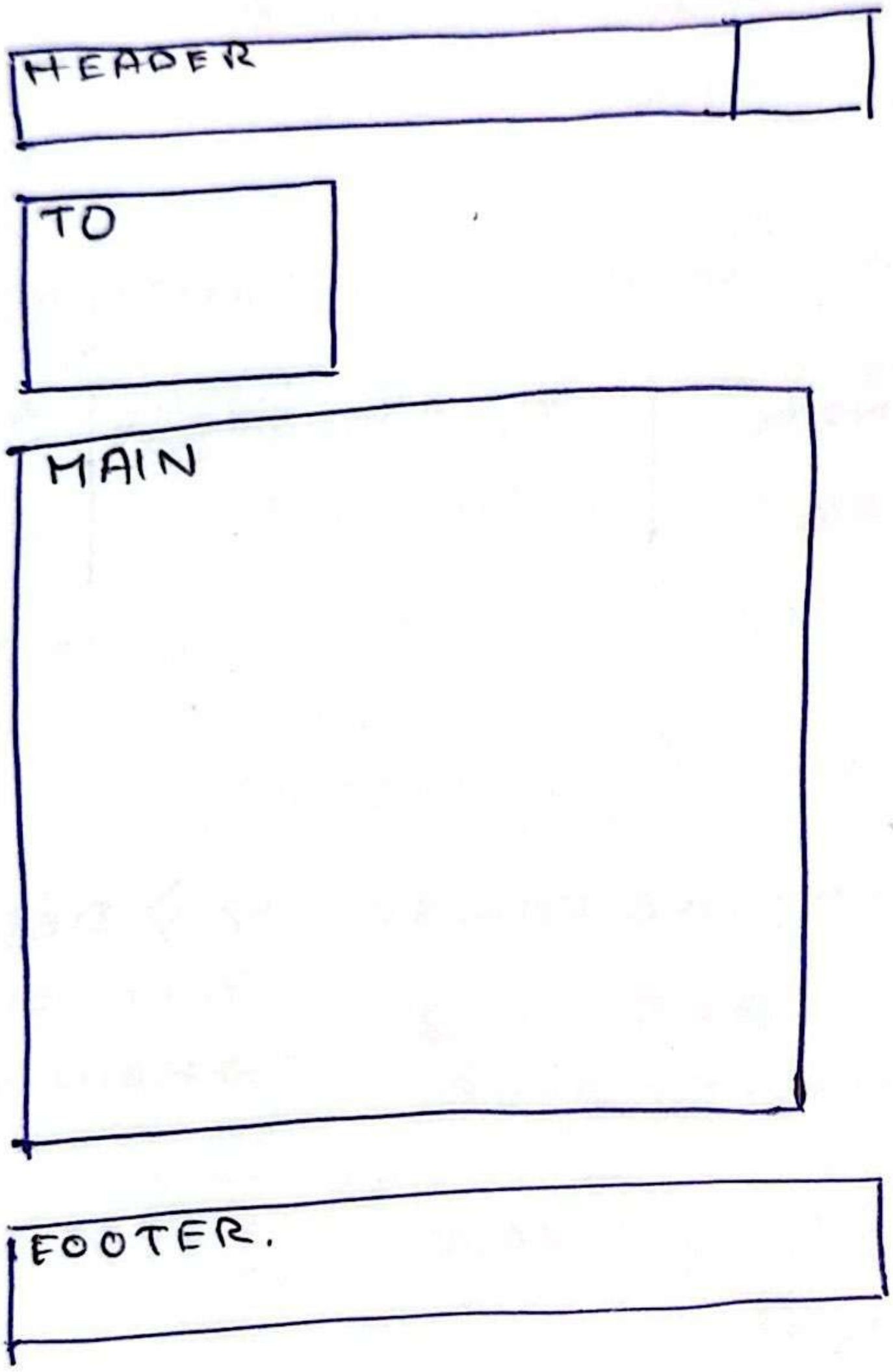
Window FOOTER
Meaning FOOTER WINDOW.

LOOP1
window TO
Meaning TO Address.

Scanned with CamScanner

Click Output Options →
Here we can see default choices.

d) Click on FORM PAINTER. Change Dimension manually as required.



→ Save

e) Now we need to create Text Element for each window.

i) Click on "Header window" → Click on Create →

Click on Text.

If there is a Popup → choose Allow.

In General Attributes → Click on Editor Btn (On left side).

Scanned with CamScanner

⇒ Type (AIRTEL INDIA PVT LTD) Start from 2nd Ln. + Press Enter.

Go Back.

ii) Click on "TO ADDRESS" → Click on Create → Click on Text.

In General Attributes → Click on Editor.

⇒ Type TO
SECURITY
EAST WING,
WAREHOUSE 3,
MARS.

Go Back.

iii) Click on "FOOTER WINDOW" → Click on Create. → Click on Text.

In General Attributes → Click on Editor

Type TERMS & CONDITIONS APPLY.

Go Back.

** In MAIN WINDOW we put all the details.

Header
1 ROW

Main window
Multiple Row.

Footer
1 ROW

Scanned with CamScanner

Now we need to create a table in Main window.

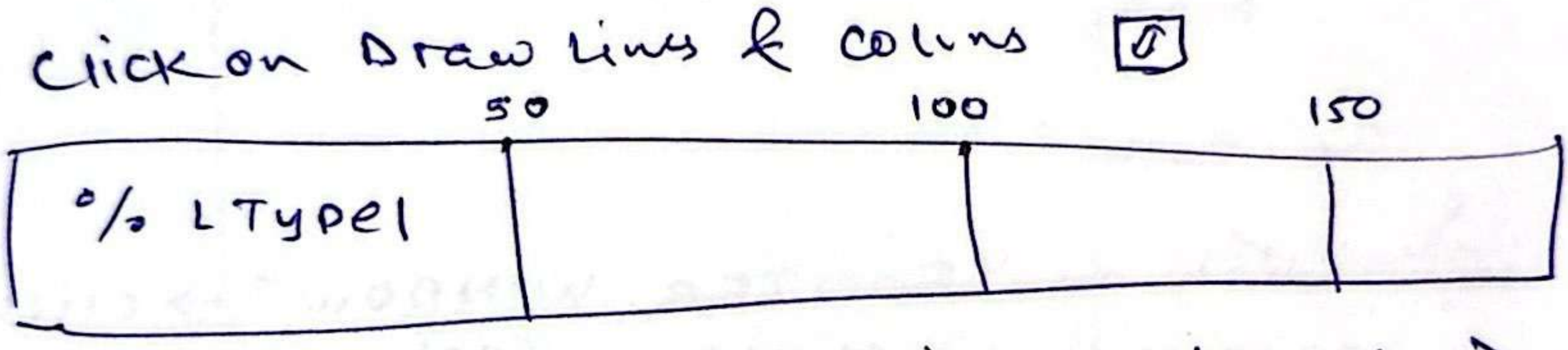
iv) Click on "Main Window" → Click on Create → Click on Table.

(Here we will have Line Types)

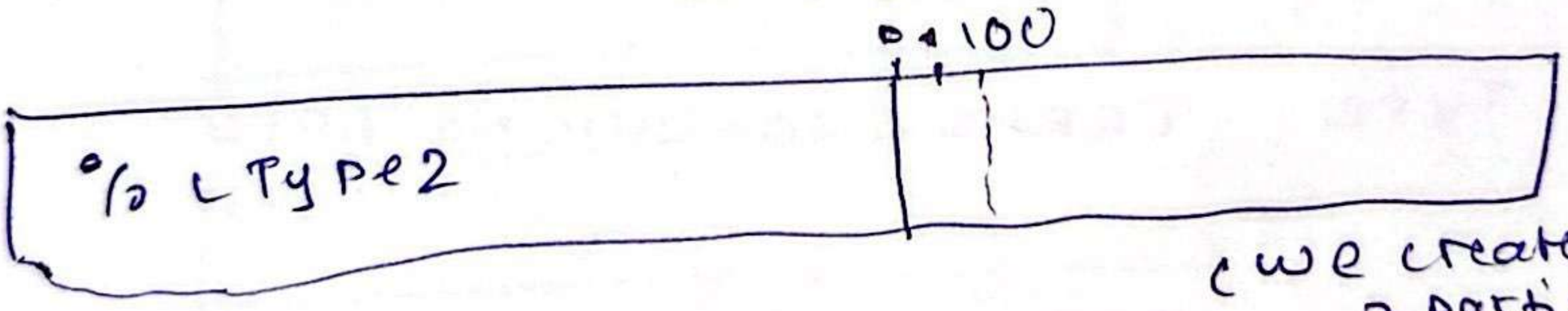
Line Type 1 → 4 columns. (Meaning).
Line Type 2 → 2 columns.

In Table → Make Partitions.

Click on Draw Lines & Colins



Right click on LType1 → Click on Insert → Empty Line Underneath.



(we created 2 partitions)

⇒ Save (so two Line Types are generated)

Now,

Right click on HEADER (under Table) → Click on Create → Click on Table Line.

Line Type → choose % LTYPE1. (4)
Automatically 4 colums will come.

Scanned with CamScanner

Right Click on "Main Area" → Click on Table line → choose Line Type 1.

Same for Footer → (choose Line Type 2)

2 columns will be automatically selected.

Save

NOW Header (HEADING CREATION)

% ROW 1

% CELL 1 (← Right click on this).

% TEXT4 → create Text.

Click on Editor. → Type MATERIAL NUMBER

% CELL 2 → (create Text)

% TEXT5

Click on Editor → Type PLANT.

% CELL 3 → (Create Text)

% TEXT6

Click on Editor → Type STORAGE LOC

% CELL 4 → (create Text)

% TEXT7

Click on Editor → Type MAINTENANCE STATUS.

After all this → Double Click on Table Name (New Table 1) → Go to Data

Scanned with CamScanner

In Data → Fill in the Internal table Name.

☑ Internal Table [IT_MARD] into [WA_MARD]

↳ Creating a loop.

In Sort Criteria (Below)

Field Name – (give Table name)

	Event (Begin)	Event (End)
MATNR	☑	☐
MATNR	☐	☑

System expects us to align the data.

Now. (Under Main Area).

Create Text in %CELLS under %ROW 2

click on Editor →

Here we are creating FIELD NAMES.

%6 CELL5 NC5

&WA_MARD – MATNR %TEXT8 NT8

↳ Here add their colum num (3rd)

%CELL6 NC6

&WA_MARD-WERKS %TEXT9 NT9

%CELL7 NC7

&WA_MARD-PSTAT %TEXT10 NT10

%CELL8 NC8

%TEXT11 NT11

&WA_MARD-LVORM

↳ Here add 4th column name.

Scanned with CamScanner

Now (Under Footer)

SF-SY-PAGE.

% ROW 3 NL 3.

Creat text in both PAGE NUMBERS → % CELL 9 NC 9

→ % TEXT 12 NT 12

% CELL 10 NC 10

Here we give page number → % TEXT 13 NT 13

&SFSY-PAGE&

Now we need to write the print program (SE38) ⊛

Whenever we create a smartform, by defaut a system defined FUNCTION MODULE is generated

To check → Open the smartform → More → Enviornment → Function Module Name

/1BCDWB/SF00000738. (copy)

Now we need to call this from a print program.

⊛ ZDXC_MARD_SFL

Click on Pattern. → Paste Function Module Name.

→ Enter.

(LP01 - Defaut Printer)

Scanned with CamScanner

IT_MARD = IT_MARD

ADD THIS CODE TO TOP →

```
DATA: IT_MARD TYPE ZITAM_MARD
      WA_MARD TYPE ZTAM_MARD

SELECT-OPTIONS: S_MATNR FOR WA_
                MARD-
                MATNR.

SELECT MATNR WERKS PSTAT LGORT
       FROM MARD
       INTO TABLE IT_MARD
       WHERE MATNR IN S_MATNR.
```

Save → Execute →

S_MATNR 1000 to 10000

Then click on Print Preview.

PO CREATE BOX →
click on Table Name
→ choose Pattern →
select Framed Pattern →
select a Pattern
Save & Activate.

Scanned with CamScanner

To create our own style → Goto Smartforms → Choose style → ZDXC_STYLE1

Click on Paragraph Format → Click on Create Node. → Name P1 →

Goto (Font) → select COURIER → → size 20 → Choose style → color →

Save → Goto HEADER DATA

Give Paragraph Name (i.e P1)

Save → Activate.

Now style needs to be assigned to our smart form.

Goto Form → Give Name → Press Change.

Goto Header → Click on Text Elements

Goto O/P option → Now

Enter Style name

(ZDXC_STYLE1)

Scanned with CamScanner

Text Module

If we have length information & we want to copy it & use it in multiple forms then we use Text Module.

Goto SMARTFORMS → Click Text Module → ZDXC-TM1 → Create.

Click on Editor → Add the data → Save.

Goto Form → Click change.

So we are changing Footer → From Text Element to Text Module.

Click Footer → Click Change Text Mode → Select Text Module.

Give Text module Name (ZDXC-TM1)

Give Language

Save → Activate.

Scanned with CamScanner

How to upload LOGO ??

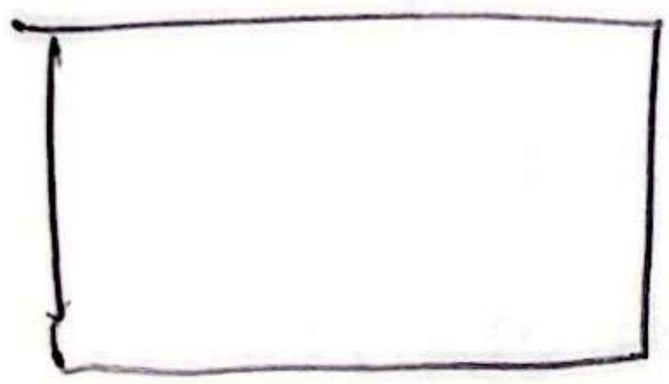
Go to SE 78 (T-Code)

Here we upload our Logo image.

Before this

Go to Form → Give Name of Form → open Page
% PAGE1 New Page

Click on Form Painter →
Beside To BOX



Right Click →
Create → Graphics

(Make Logo in PAINT
save as. bmp)

* Go to GRAPHICS
→ choose Bitmap Image.
→ ~~Give Image~~ Name. choose
color
→ click Import (on top) FS
(I) Logo.

(SAPLOGO. BMP)
→ Choose location of Image

After IMPORTING → Go BACK TO
FORMS → GRAPHICS
Created
[give Image Name in GA]

Object → GRAPHICS

ID → BMAP.

If Image too Big → Give More
(Resolution)
eg- 200)

Scanned with CamScanner

SAP Smartforms

Smartforms : It is a GUI tool which is used to design the business documents of SAP such as :

→ PURCHASE ORDER
→ GRN - Goods Receipt Note.
→ Sales Order Booking Pur.
→ Delivery Challan.
→ Invoice Copy.
→ Barcode Printing.
→ Cheque Printing.

The O/P can be over a printer, fax, email, pdf file. or via internet in HTML/XML O/P.

Smartforms are introduced by SAP in 1998 to overcome the limitations in SAP Scripts.

→ Very Easy to Develop, maintain & Transport.

Features & Advantages →

1) Easy to develop without much programming knowledge with the help of G.U.I.
2) Different Text fonts & colors are possible in SF.
3) We can create multiple page SF.
4) To make any change we can use Drag & Drop → Cut & Paste.

Scanned with CamScanner

5) We can keep the backup very easily, complete sf or of individual mode.

6) We can insert static / Dynamic Text Elements.

7) Address window to print the customer/vendor Address with the help of ADRNR number.

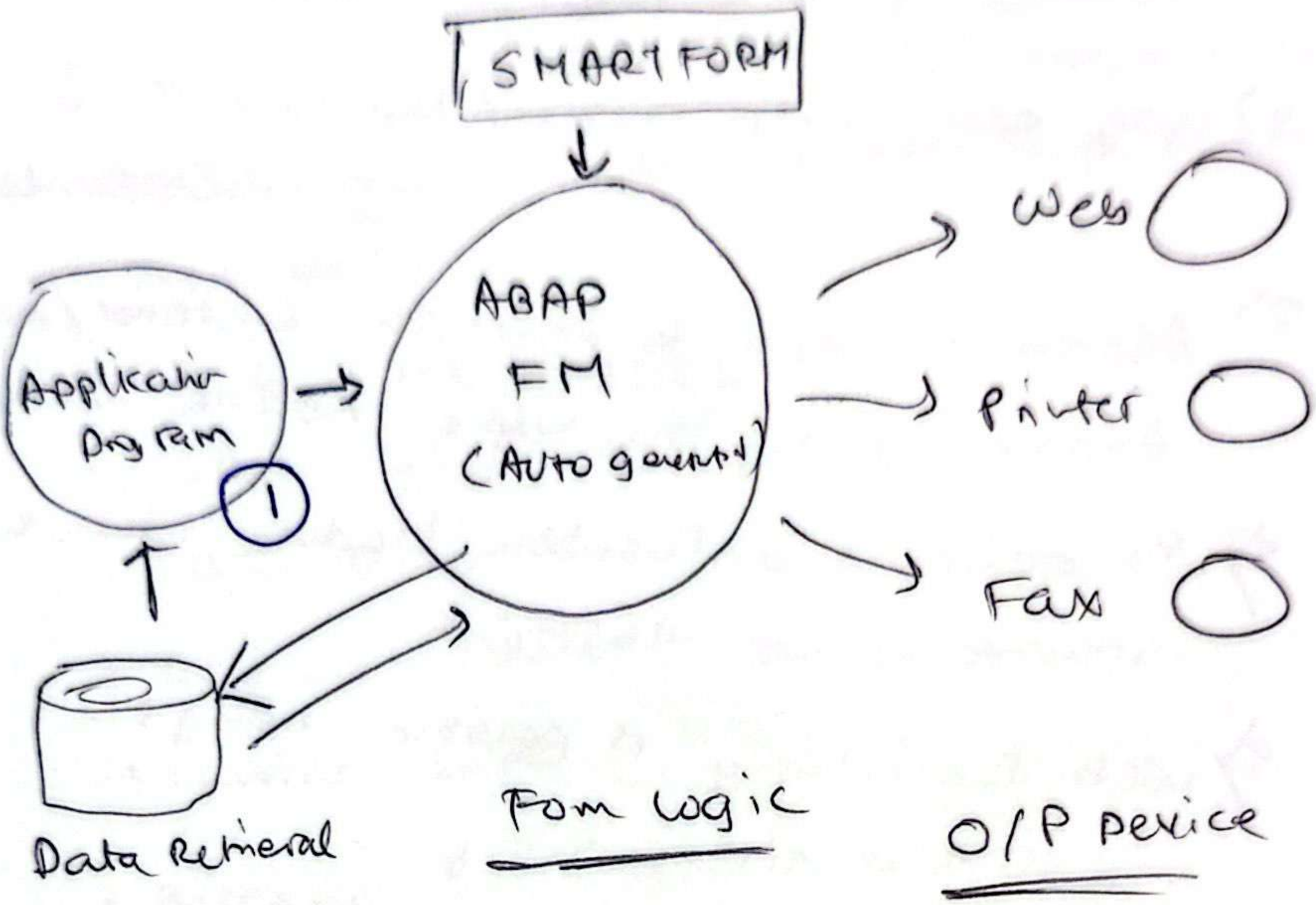
8) It generates a Function Module when the smartforms are activated.

9) Web publishing is possible using the generated XML output.

SMARTFORMS	SCRIPTS
1) Smartforms are client independent.	1) SAP scripts are client dependant.
2) Text supports various colors	2) Only black & white colour possible.
3) Main window - not compulsory	3) Main window - compulsory
4) Multiple page formats possible	4) Multiple page formats not possible.
5) FM generated automatic	5) No FM generated
6) web publishing possible (use XML)	6) web publishing - Not possible.
7) We can write routine not possible	7) Routines not possible
8) Mass Printing of forms → possible	8) Mass Printing of forms → not possible.

Scanned with CamScanner

Architecture



1. we are passing data in form of variable/structure/smartforms.

T-Code ⇒ SMARTFORMS.

Components ⇒ (SMARTFORM)

There are 2 main components of SF.

1) Global settings. → Form Attribute. → Form Interface → global setting.
2) Pages & windows.

i) FORM Attribute – specifies the system information like who created, Date & time, package, smart style – page format. (DIN A4/DIN A3)

Scanned with CamScanner

2) Form Interface → used as a mediator for providing export of a driver prog as an input to the smartforms that can be as a variable, structure or table, also we can export the parameters from smartforms back to the driver program.

3) Global Definition → use for data declaration / variables / types / field / symbols / form Routines / code Logic / Comments / Queries Fields.

PAGES & WINDOWS

1) Page is used to define layout of a smartform

2) Windows is use to display text or information at a particular place.

3) Graphics is use to display the logo or images.

4) Address → use to display address of vendor, customer, or using address number field.

Scanned with CamScanner

Windows ⇒

Inside windows we can create text, table, template, flow logic, folder.

1) Texts are use to display the static/dynamic information in a window.

2) Table is use to display the information in the form of a table.

3) Template is use when we have fix number of rows & columns, its like table only but with fix rows & columns

4) Program lines are use to write the abap logic.

Types of Window ⇒.

1) Main window – use to cont o/p, the content of the main can be display of multiple pages

2) secondary Window – use to when the max number of rows are fixed that all can be displayed on single page, its not continues.

3) Final windows process at the end of form processing, no matter if we are displaying the final window at the start of first page, first ~~page~~ page system will

Scanned with CamScanner

complete the entire form process as the print data inside of final window.

4) Copies Window → use to mark the pages as original, duplicate copy etc.

SAP ABAP → HANA

HANA → Developing
Modelling
Admin.

High Performance Analytic Appliance.
(Hasso's New Architecture).

SAP co-founder → Hasso.

Two User ID & pass for Database & Application.

[Open Perspective has all ABAP Plugins]

1st Step

connect Application Server from SAP ~~ABAP~~ Software
↳ This same system needs to be added from Eclipse

(Dont click on Finish)

Delete Default Project → Type own name.

After connecting → Go to Modeler.

Row Store - good for data storage & work.

Column Store - good for arithmatic calc

~~[illegible]~~ https://tools.hana.ondemand.com/mon.

Help → Install New Software
→ Add (Top Right).

Scanned with CamScanner

None adt.

Tools - https://tools.ondemand.com/neon.

Ok → ○ Unique box for adt & hanatools.

○ Unique box for one ranging.

→ Next → Next → Finish.

ADT - Abap Developement Kit.

SAP Hana studio is an Eclipse based tool.

↳ central dev. environment & main distribution tool for system.

Eclipse - Open IDE integrates different tools same)

Open perspective → ABAP. (Pass & User ID will be

Add Plugins → https://tools.ondemand.com/neon.

Catalog represents SAP HANA's data dictionary, i.e all data structures, tables and data which can be used

All the physical tables & views can be found under

User ID - trainee05.
Pass - Welcome08.

We need a package - SE21 / SE80.

Scanned with CamScanner

Then goto SE38 → Do demo program.

Open Perspective → ABAP → Click on left top → Click on ABAP Project → Select proper SAP server. → Next → Next → Give Password (ABAP) → Welcome8

Next → Give new name of Project.

(To Run → ▶ → ABAP Application) or F8.

ctrl + shift >/< → comment/uncomment

Right Click on Editor → Quick fix.

(For Quick Access of any Package → Add it to Favorite Package by right clicking on it)

[No response in 1 to 2 days? → Give Reminder]

HR Connect

1) VDI issues to be written to VDI team.
<pptvdi.global@capgemini.com> Always mention Batch Name & Emp ID

Keep cc - Narendekar S ; Sabhya Pathak.
↑ (only if extremely necessary).

2) Mobile Pass Issue - write to training co-ordinator (Keep Sabhya in cc)

3) Server Issue - write to transversal team.
<sap transversal basis.in@capgemini.com>

4) Exam Link - Unique to every person.
↳ So dont ask friends.
↳ write to Sabhya Pathak Mam / Poorva Dharap Maam
<poorva.dharap@capgemini.com>.

Scanned with CamScanner

5) Issues during exam → write to.
support@doselect.com 2267557744
(keep Saubhya & Poorva in cc)

6) Mention your complete batch name & emp id when writing mails to anyone.

7) software installation or system dispatch
Hiranmai Ramakrishnan.
<Hiranmai.Ramakrishnan@capgemini.com>

8) VDI availability - Lohakare, Ravindra.
<ravindra.Lohakare@capgemini.com>.

9) Explain usage of VDI & parallel system allocation & software installation.

Excel sheet - Emp ID, Name, Email ID.
12-15 people.
(keep Saubhya in cc)
(send me to Naranekar, S.)

10) Regularly fill in details (Google Drive).

11) Coursera / Nat Platform — Training coordinators.

12) Project Code issue - Nakkeeran, Sony
<Sony.p-k@capg.com>

Scanned with CamScanner

How to open SE11 in Eclipse →.

Select Project → select SAP ui Launcher
→ Do same steps to run the table.

PACKAGE - {246117206_TAM}

when HANA & ABAP are connected using same server details, every change made in any program in Hana will be reflective in ABAP.

[ctrl + shift + <] → comment line.
+ > → uncomment.

ctrl + shift + F1 / Right Click on Editor → Quick Fix.
(For any suggestion)

Right click on Editor → Source Code → (Format / Add/Remove comment)

Here Pretty Printer → Quick Fix.

[HANA has no PATTERN]

1) How to open any standard table on Eclipse?

Place cursor on your project → select SAPUI btn → Next → Next.

Go to SE11 → search with std table name → Source code of Table will open → Run directly.

We can filter data however we like.

Scanned with CamScanner

Max Rows / Filters / select columns / No of Entries.

For knowing data elements → use SE11.
(Dont ~~consow~~ consult to predefined data elements)

→ {CONNECTED MODELER TO ABAP.}

MODELER.

LS4@LS4 (TRAINEE10) LS4
→ Catalog (Hana Dictionary).
→ Content
→ Provisioning
→ Security.

All of this is stored in SAPHANADB

we can access this using Catalog → v → select SAPHANADB.

To find a Table in this SAPHANADB →
Expand → RC on Table → select Find Table.

1) physical DB is being stored in ABAP, whenever we are opening it using SE11 in Eclipse, the particular DB is being uploaded into the ~~DOB~~.
Application server.

Scanned with CamScanner

NEW HANA SYNTAX

1) Inline Data Declaration
2) Explicit Type Declaration
3) Std Internal Table Declaration.
4) Sorted " " "
5) Internal Table with more components.
6) How to work with Deep structure.
7) MOVE - CORRESPONDING for Internal Tables.
8) Table Expressions.
9) GROUP BY for Internal Tables.
10) FILTER expressions.
11) INNER JOIN
12) NEW keyword for creating objects.
13) CONVERSION - EXIT - ALPHA - INPUT/OUTPUT.
14) Using Switch statement.
15) New Open SQL.

Inline data declaration.

```
Data (v_Data) = 'ABC 199 XYZ'.
Write: 'Output:', v_Data.
```

O/P → ABAP on HANA

Output: ABC 199 XYZ.

Declaration of work Areas

```
LOOP AT itab INTO DATA(wa).
...
END LOOP.
```

Scanned with CamScanner

Declaration of Actual parameter →

OLD

```
Data a1 Type ---
Data a2 Type ---
oref -> meth (IMPORTING p1 = a1
              IMPORTING p2 = a2
              --- ).
```

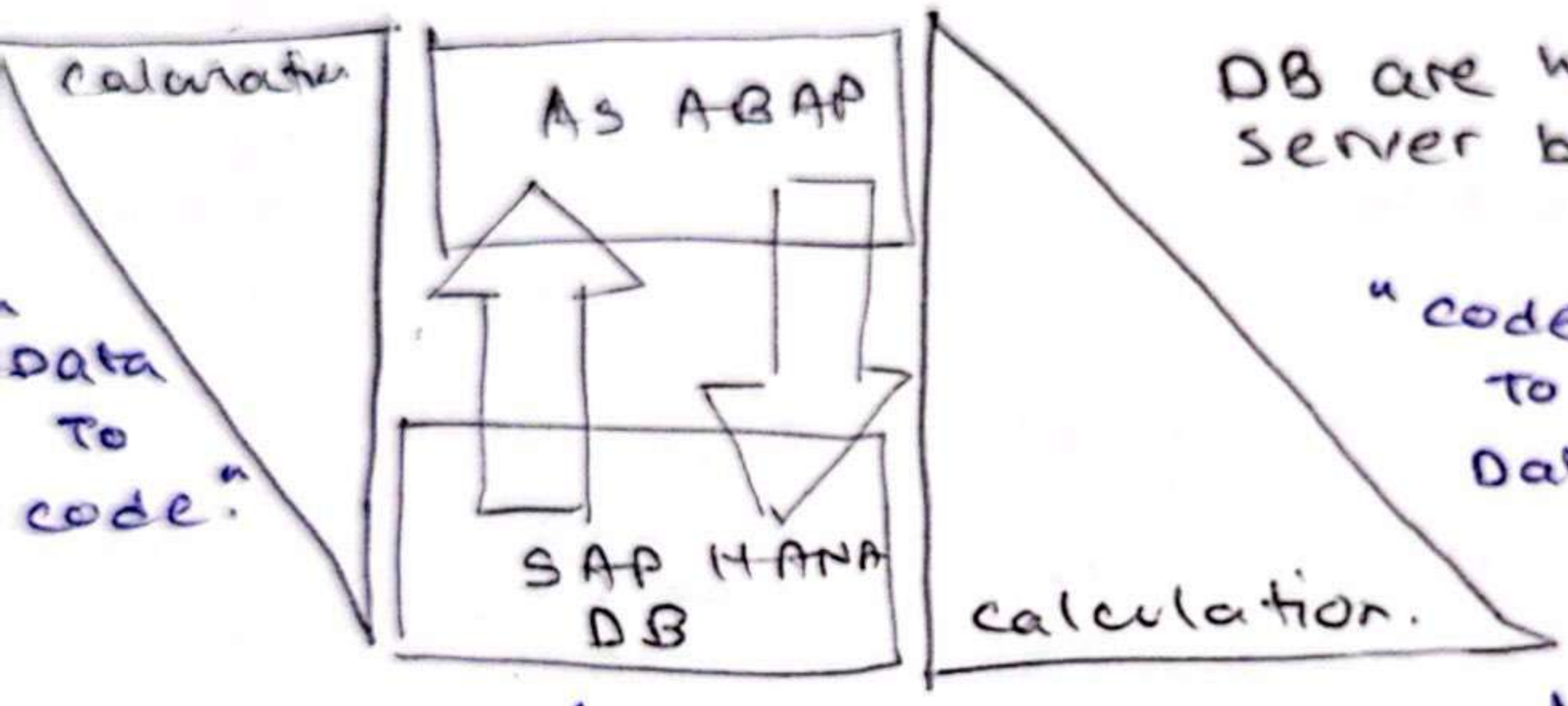
NEW

```
oref -> meth (IMPORTING
              p1 = data(a1)
              IMPORTING p2 = data
              (a2)
              .... )
```

CODE PUSHDOWN.

All codes are in ZDEMO_PROG2_INNER

DB are hold in Application server but not stored.



"code To Data" Query is going down.

Normal

```
START-OF-SELECTION
  SELECT
    VBELN
    POSNR
    MATNR
    PMATWA
    :
```

Inline

```
SELECT
VBELN,
POSNR,
MATNR,
MATWA,
:
```

```
INTO TABLE IT_TAB
FROM VBAP
WHERE VBELN IN S_VBELN
```

NO INTO TABLE @ IT_TAB FROM VBAP.
WHERE VBELN IN @S_VBELN

(Here we are using @ to understand Escape) as Query is moving down to DB layer (all calculation) the reason part is sent back ↑ → This results in

Scanned with CamScanner

INTO TABLE @data(it_tab) 1
FROM VBAP
WHERE VBELN IN @S_VBELN.

LOOP at IT_TAB INTO data(wa_tab). 2.

Q) In 1 we are using @(ESCAPE) but in 2 we are not, why?

(code to Data)

Because Loop in going down & Query hits the DB. (There is no interaction b/w internal table)

Next

In order to display table we use object.
CL_DEMO_OUTPUT (T-Code - SE24).

Then go to program → Select Pattern → ABAP objects → select → Class/Interface is
↳ CL_DEMO_OUTPUT.

And METHOD as DISPLAY_DATA
choose

LATEST
~~Call method~~ CL_DEMO_OUTPUT => DISPLAY_DATA

EXPORTING.
VALUE = IT_TAB (My internal table)
NAME = 'SALES ORDER INFO 1'. (Table Heading)

Q) If Pattern is suggesting old method why we use CL_DEMO_OUTPUT

=> class component selector → object component selector.

CL_DEMO_OUTPUT => (Click ctrl + space) use up+down ↑↓) Then After selecting DISPLAY_DATA) → PRESS Ctrl + shift + Enter.

Scanned with CamScanner

5) SELECT
VBELN AS SALES_ORNO,
POSNR AS SALES_ITEM,
MATNR AS MAT_NO,
.
.
.

This is called using Aliases.

LOOP AT IT_TAB INTO DATA (WA_TAB).
WRITE:/ WA_TAB-SALES_ORNO

ENDLOOP.

Instead of using real column name we are using an alias to print.

6) SELECT (1 + 1) AS TWO, → Arithmetic Op → o/p gets stored in row

{(Time - 18:31) / Paris / 28 Sept.}

24) Using switch statement.

Parameters P_DAY TYPE I.
DATA 'A(20) TYPE C.
DATA (LV_DAY) = SWITCH #(P_DAY
when 1 Then ('Monday ---')
when 2 Then ('Tuesday')
when 3 Then ('Wednesday')
when 4 Then ('Thursday')
when 5 Then ('Friday')
when 6 Then ('Saturday')
when 7 Then ('Sunday')
Else 'invalid').
WRITE: / P_DAY, LV_DAY COLOR3.

Scanned with CamScanner

concatenation ⇒

CONCATENATE

DATA(LV_TEXT) = 'ABAP' && '' && 'ON' &&

'HANA MB INLINE CODE'. "Nahm

WRITE / LV_TEXT.

Old Method ↑

New Method ↓

Data (LV_text) = 'ABAP AND HANA MB INLINE CODE'.

write / LV_text.

INNER JOIN Improvement.

we can use wildcard like SELECT * in new innerjoin.

(~ - Til operator)

```
SELECT a~*, b~posnr, b~matnr FROM
        VBAK AS a INNER JOIN
            VBAP AS b.

    ON a~VBELN = b~VBELN
    WHERE a~auart = 'ZIEN'
    INTO
```

Scanned with CamScanner

In Breakpoint (Debugging).

F5 → Line to line o/p.
F6 ↑ & F8 → Direct o/p (Final).

In order to close Debug Perspective → Right click on ☆ → Close.

→ Internal Table. → Integer size

7) Types ty_TAB type $\boxed{\text{sorted}}$ table of $\boxed{\text{I}}$ with unique key table-line.

DATA (it_MATNR) = value ty_TAB((234) ( )
(2349))

cl_DEMO_OUTPUT → display (it_MATNR)

• sorted table → can sort automatically
standard table → require extra sort functionality

o/p →

IT_MATNR
234
0
2349.

Scanned with CamScanner

8) New keyword for creating Objects.

we can use 'NEW' instances an object instead of use CREATE OBJECT

New syn →

lo_myclass = NEW ZCL_Myclass(myname = 'India').

↑ object name ↑ class

Old syn →

Data: lo_myclass TYPE REF TO ZCL_MYCLASS.

CREATE OBJECT lo_myclass EXPORTING myname = 'India'.

1) Memory space will be created for that object.

2) Constructor is triggered automatically when memory space is allocated.

CL_DEMO_OUTPUT => display (IT_MATNR)

↑ This is a reference object & it cannot be printed directly.

Types ty_TAB type standard table of I with default key.

(2) DATA (IT_MATNR) = new ty_TAB((234) (1) (234a)).

↳ instead of value, we are creating a instance of the class but we cannot get values directly.

CL_DEMO_OUTPUT => display (IT_MATNR)

(

) Put a Breakpoint here → Run → Go to Desktop 3 choose Ln 2 → single step → choose Ref pin.

Scanned with CamScanner

a) types TY_BATH TYPE

DATA (TT_TABA) = NEW TY_DATA(

(SIGN = 'I' OPTION = 'EQ' LOW = '0030000012')

(" " " " HIGH = '0030000020')

).

cl_DEMO_OUTPUT => display_data (IT_TABA)

ROW	SIGN	OPTION	LOW	HIGH
1	I	EQ	0030000012	
2	I	EQ	0030000020.	

12) select vbeln,
posnr,
matnr,

SUBQUERY

From vbap / vbak

into TABLE @data(it_vbap).

where vbeln = any (select vbeln from vbap

where vbeln BETWEEN '0000000001' and '0000000002').

Press F on any

8) new.

Data (dref) = NEW i(iii).

cl_demo_output => write(/ dref -> * & dref -> *).

clear dref.

Assign dref -> * To Field symbol (<fs>).

cl_demo_output => write(/ sy-subrc & sy-subrc).

cl_demo_output => display ().

Scanned with CamScanner

dref = 111 → integer.

Data (dref) = New (i)(111) → assigning value.

→ comment

cl_demo_output => write (|dref ->* the value is:

{ dref ->* }|).

↳ actual value.

clear dref. → clearing memory

14) select vbeln,
erdat,
erzet,

FROM vbak.

into TABLE @data (it_vbak)

WHERE EXISTS (SELECT matnr from vbap

WHERE MATNR = '1FG1_CP').

(sy-subrc mandatory)

IF SY-SUBRC EQ 0

cl_DEMO_OUTPUT => display_data

(VALUE = it_vbak).

else.

WRITE 'no data'.

endif.

|| → pipeline symbol → concatenating two exam data.

16) Types pack_dec Type p Decimals 3.

Data: A_1 Type P Decimals 3 value '2.349',

B_1 Type P Decimals 5 value '2.34944'.

Scanned with CamScanner

START-OF-SELECTION

```
Data (value-1) = CONV PACK_DEC ('2.349'
                                '2.39944');

cl_Demo_output => write_Data(
    Exporting
      Value = value_1
      Name = 'FIRST STMT'
    ).
```

Hard.

```
Data (value_2) = CONV PACK_DEC (A-1 + B-1).

cl_Demo_output => write_Data (
    Exporting
      value = value_2
      Name = 'SECOND STMT2'
    ).
```

Easier way

```
Data value_3 Type PACK_DEC

value_3 = A-1 + B1

cl_Demo_output => write Data (
    Exporting
      Value = Value-3
      Name = 'THIRD STMT3'
    ).
```

Easiest way

```
cl_Demo_output => Display( )
```

↳ if we dont use this we will o/p in 3 pages.

(All three gives same o/p but we use different ways)

Scanned with CamScanner

FIELD SYMBOLS → Pointer concept in (ABAP). ABAP is called FIELD SYMBOL

Data : F1(20) Type C.
Field-symbols <FS>.
F1 = 'ABAP TECH'.
ASSIGN F1 TO <FS>. (eg &A)
WRITE : <FS>.

TABLE [ ] replaces Read Table.

20) FILTER (cmd) - used for filtering source b/w two tables.

HANA Modelling

key ~~op~~ concepts. →

1) Measure
2) Attribute
3) Dimension
4) Star Schema
5) Hierarchy.
6) Semantics.

- **Measure** - a numeric value such as a price/quan/volume on which you can process arithmetic/statistics operations. such as sum/avg/top values & calculation. eg: Number of Products Sold.

Attribute - an element that is used to describe the measure. eg: Product ID / Product Name

Scanned with CamScanner

Dimension + analyzing measures are easier when you group attributes together by Dimension.

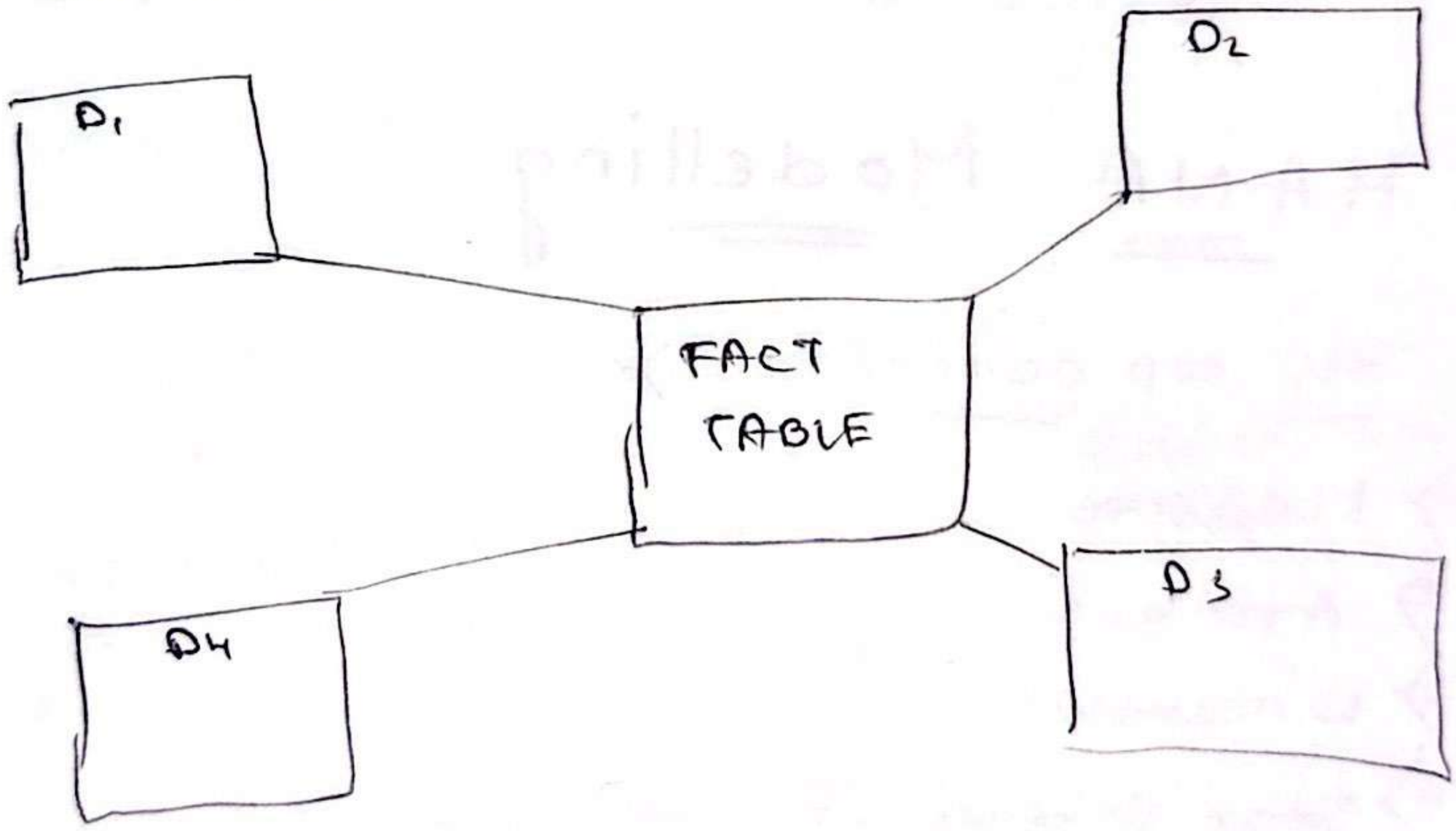
eg: Dimension 1

Product

- key
- Name
- category
- supplier

Star Schema + consists of one fact table that references one or several dimension tables.

The fact table contains FACTS / MEASURES as well as key used to identify for each dimension which dimension member corresponds to each fact.



Hierarchy - structured representation of an organization a list of products the time dimension & so on by levels.

Semantics - describes what a data means or relate to

Why Modelling?

Traditional

1) Role of DB is to provide data.
2) App sends down select statement to individual database tables.
3) Row data sent from DB to application.
4) App then begins to process the data by combining it, aggregate & perform calculation

Here Application does all the Hard work.

Results

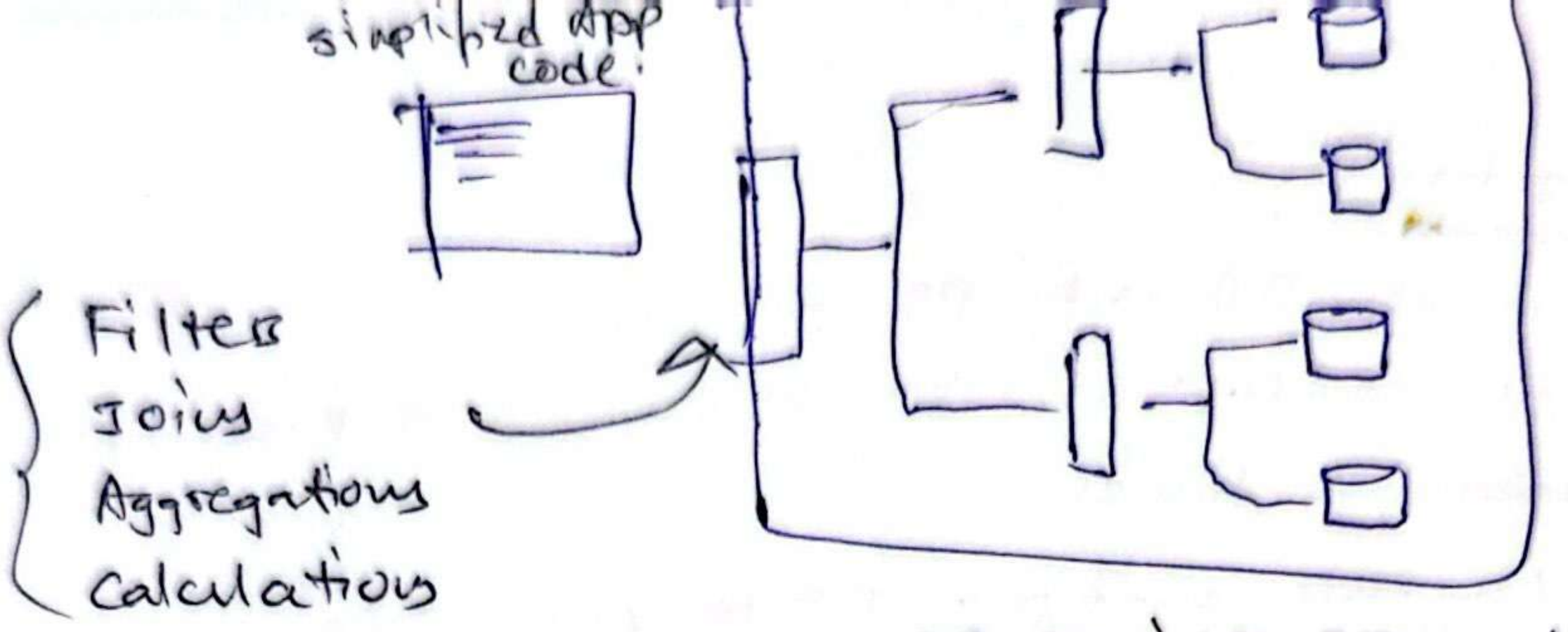
Moving a lot of raw data b/w the DB & application
Application code becomes complex.
Performance degradation.

HANA Approach ⇒

1) One can build a modelling layer on top of the DB tables, to provide data to the application in a ready to go processed form.
2) With HANA one can build Information Views that combines data from multiple tables & apply filters, conditions, calculations & aggregation
3) Info views are developed using easy to use modelling tools, & are stored in SAP HANA alongside the DB tables in the DB catalog.

Here Application is not the ~~[illegible]~~ Hard worker as it simply calls Info View & the processing is pushed down

Scanned with CamScanner



DB provides processed data.

Information View → are used in SAP HANA to create a virtual DB Model, based on the data that resides in the SAP HANA DB schemas.

Info View → Attribute View
→ Analytical View
→ Calculation View.

46.42 (3rd vid, 29th).

CORE DATA SERVICES

CDS stands for core Data services.

1) A view is an entity that is not persistant, it is defined as the projection of other entities.
2) CDS view is reusable data models on the DB.
3) It is a data model that represents framework of what relationships are in a DB.
4) With CDS, data models are defined & consumed on the DB rather than on the application server.

Scanned with CamScanner

Rule of Thumb ⇒
Do as much as you can in the database to get the best performance.

Features ⇒

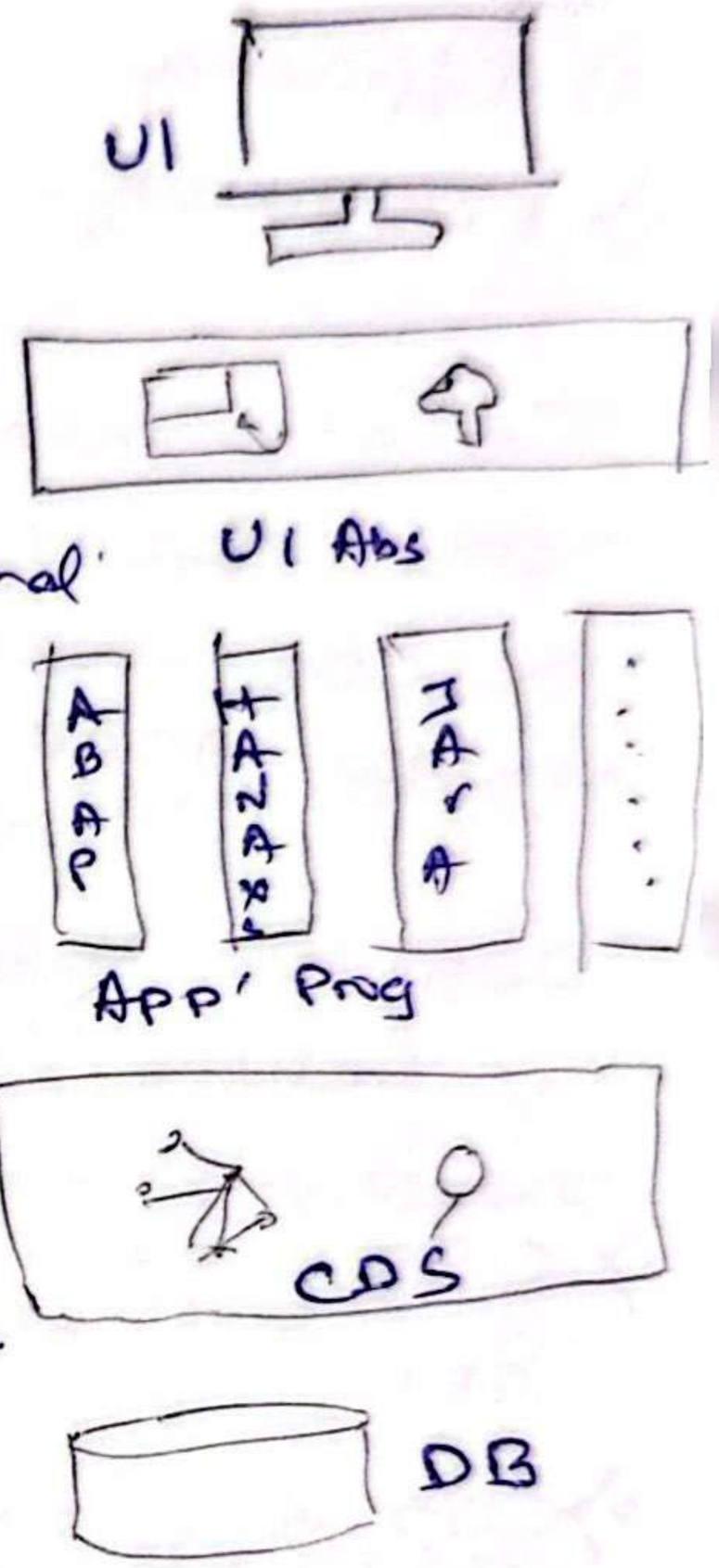
Core Data Services.

→ Next gen of data definition & access for data centric applications.

→ optimized application programming model for all domains. (Transactional Analytical).

→ Technically an extension to SQL:
  → Expressions.
  → Domain specific Metadata.
  → Associations.

→ CDS includes
  DDL – Data Definition Language (Mainly)
  QL – Query Language
  DML – Data Manipulation Language.
  DCL – Data Control Language



CDS view - shows data from DB & needs ODATA as interface & SAP UI5 FIORI as Front End.

→ Code to Data Paradigm
  supported through extended view functionality.

→ Definition of semantically rich data model in the ABAP dictionary.

Scanned with CamScanner

Advantages =>

1) semantically rich data models.
2) compatibility across any db platforms.
3) Efficiency.
4) Extensibility.

The statement DEFINE VIEW is used to create the CDS DDL in ABAP. This is done in the CDS source code of a CDS data definition in the ABAP Developement Tools (ADT).

Definition is only possible with ABAP Dev. Tools in Eclipse / HANA. studio.

1) CDS view cannot be created via transaction SE11.
2) CDS view can be developed and maintained in SAP HANA studio and in ABAP in Eclipse.

CDS view definition =>.

1) View name.
2) semantic Information (key field).
3) Projection List.
4) Aliases.

Projection List =>

1) Client Dependency.
2) semantic Information (key)
3) Aliases.
4) Aggregation
5) Literals.
6) Arithmetic Expressions.
7) conditional Expressions.

Scanned with CamScanner

In CDS, @ → means Annotation.

@ Abap Catalog. sqlViewName : 'zcdsVieo'
, compiler. compareFilter : true.

CDS

1) □ - Choose Other (search ddl.)
2) Then choose Data Definition
3) Give Name & Details.
4) Next → choose type of CDS → Next.
5) Give SQL View Name different from program name.

MANDT
~~KUNNR~~ LIFNR
LAND1
NAME1
ORT01
~~DATER~~.

@ Abap Catalog. sqlViewName : 'ZDEMO_7206_CDS'
@ Abap Catalog. compiler. comparefilter : true.
@ Abap Catalog. preservekey : true.
@ Access control. authorization check : #CHECK.
@ EndUserText. label : 'CDS SO ORD TABLE'

define view ZDEMO_46117206_CDS.

as selected from 246117206_lfal as SO_H

↳ use aliases always.

{
key SO_H.mandt, → Mention key for key field.
SO_H. lifnr,
SO_H. land1,
SO_H. name1,
SO_H. ort01

{ SO_H ↳ Header }

};

Scanned with CamScanner

Normal views are only graphical methods, and doesnt support any code.

where Lifnr = '1002';

↳ Parameters from one row column.

2) From Templates, choose Define view Parameter.

Sql view Name '27206 - RNG - CDS'

define view Z7206_RNG

with parameters p_lifnr : Z7206_lifnr

parameter name

as select from 246117206 - lfal.

as so-i

→ this data element can also be written like abap.char(10)

↳ Not good method.

{

key so_i.lifnr,

key so_i.mandt,

so_i.name1,

so_i.ort01

} where so_i.lifnr = $parameter.p_lifnr;

SO.I → line Items.

This is to ~~make~~ take a query as user input.

Scanned with CamScanner

Range

```
sql view Z7206_RNG2_CDS
define view Z7206_RNG2
with parameters { p_lifnr1 : Z7206_lifnr,
                  p_lifnr2 : Z7206_lifnr
```

This is a user range input.

```
as select from 246117206_lfa1 as so_i
{
  key so_i.lifnr,
  key so_i.mandt,
  so_i.name1,
  so_i.ort01,
} where so_i.lifnr between :p_lifnr1 and
                           :p_lifnr2;
```

User can give range to see values for a particular field.

~~Join~~ Join.

Template → Data View with Join. (2 tables)

```
define view Z7206_JOIN
with parameters p_lifnr1 : Z7206_lifnr,
                p_lifnr2 : Z7206_lifnr,

as select from 246117206_lfb1 as SO_H
  inner join 246117206_lfa1 as SO_I.
  on SO_H.Lifnr = SO_I.lifnr
```

Scanned with CamScanner

{
key so-H. Lifnr,
key so-I. mandt, } key fields must be together.

so-H. bukrs,
so-H. zterm,
so-H. webtr, } table 2

so-I. name1,
so I. ernam

} where so-H. lifnr between :p-lifnr1
2nd table
and :p-lifnr2

CURRENCY

define view Z7206_CDS-CURR
with parameters/to'currency:abap. cuky(5), → target currency.
exc_date:abap. dats → exchange date

as select from snwd_so as so

{

so.so_id,
so.gross_amount as Befor_Gross_Convt,

currency_conversion(amount => so.gross_amount,
source_currency => so. currency_code,
round => 'X',
target_currency => :to_currency,
exchange_rate_date => :exc_date,
error_handling => 'SET_TO_NULL') as

3. : to_currency as currencies // to know which currency.

write currency-conversion (then press shift + space)

SO - Sales Order

BPA - Business Partner Association

Other

as select from snwd_bpa

{ lpad(bp_id, 7, '0') as bp_role_id,
email_address → truncates length to 7 digits, with 0.

}; MEP

Low

300, 400, 500, 600, 900, 700, 800

LFB1-WEBTR CASE a. webtr

when is blw 1 & 300 the LIFNR
1 low BUKNR

Vendor Order Address

SAP LOGO

es c NA.
cus as grade

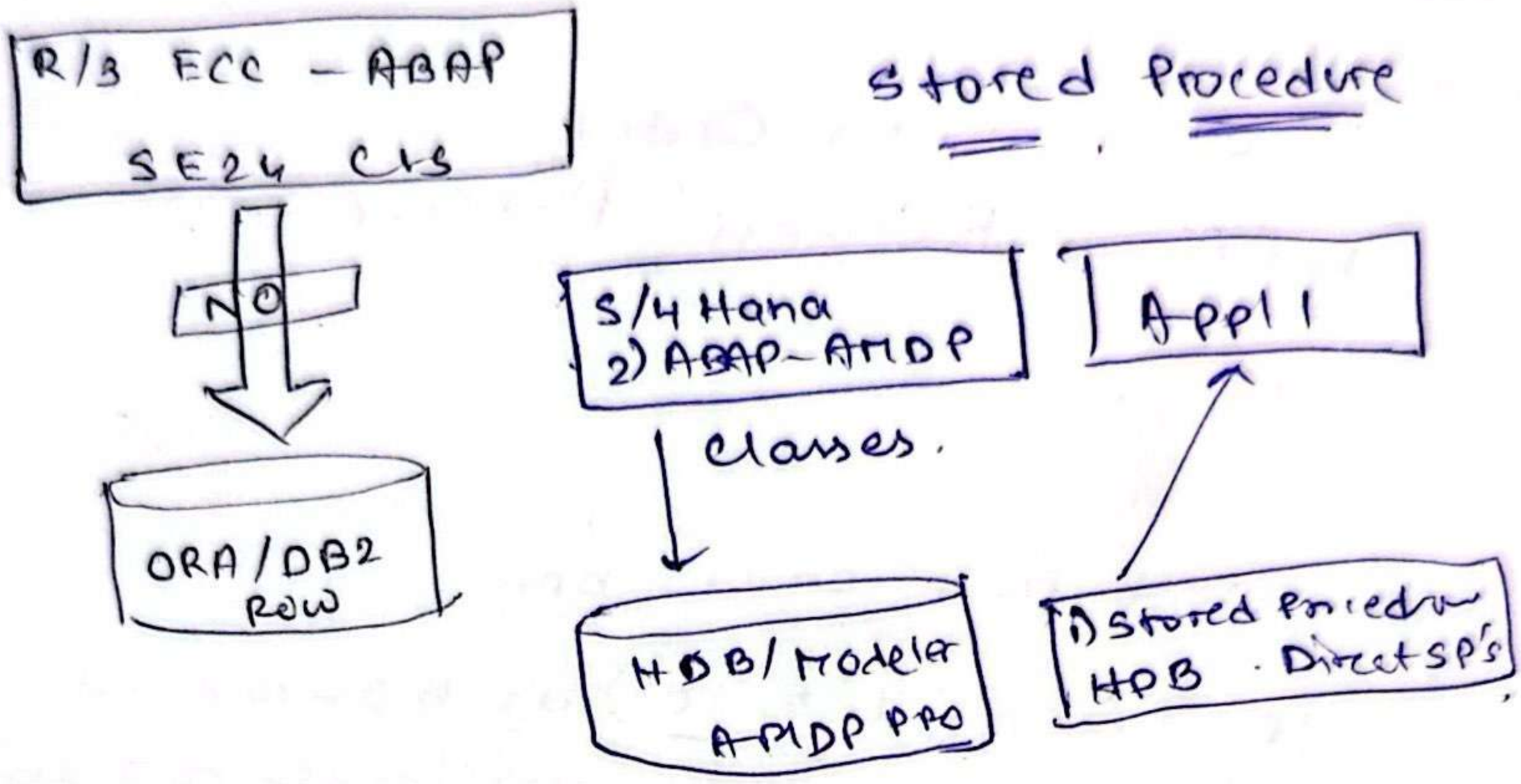
LFB1 (VENDOR MASTER (comp code))

LFMIN - Vendor master

Page 1 of 10

Terms & conditions
goods must be return within 10 days of purchase.

AMDP (Abap Managed Database Procedure).



A procedure allows us to group SQL statements into a single block. Stored Procedures are used to achieve certain results across application,

1) Stored procedures are used to achieve certain results across applications.

2) The set of SQL statements & the logic that is used to perform some specific task are stored in SQL Stored Procedures.

3) SP are also used for performance optimization as it contains series of SQL statements & results from one set of statement determines next set of statements to be executed.

SP prevents users to see the complexity & details of tables in a db.

Scanned with CamScanner

Procedure

Procedures define reusable data processing functions.
which means we can avoid ~~or~~ developing overly complex calculation ~~are~~ again & again.
we simply define a procedure & call it whenever required.

eg ⇒ You could create a procedure that calculates the tax for each item sold. We would ~~se~~ simply define the input parameter as a tax percentage & then define the output parameter as ~~out~~ total output tax.

Calling Stored Procedure

Stored procedures are db specific, They can be called using:

1) Using Native SQL.
2) ADBC
3) AMDP.

Native (outdated).

EXEC SQL

ENDEXEC

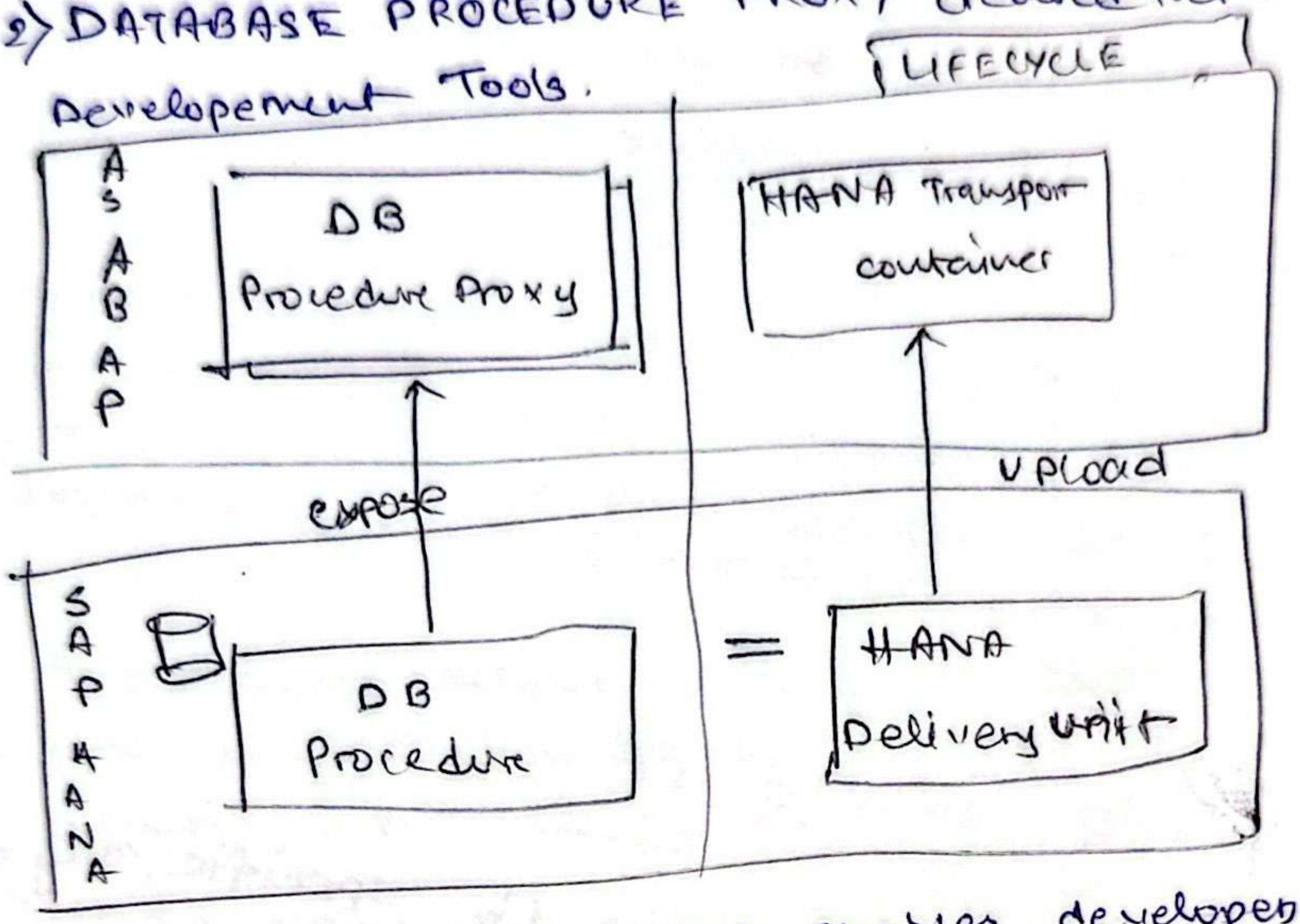
Classical Approach for accessing Procedure ⇒
current way for using HANA Procedures in ABAP.
"Bottom Up" Approach with DB procedure Proxies.

Scanned with CamScanner

ABAP & HANA servers involved for developing managing & calling HANA procedures in ABAP.

1) HANA PROCEDURE developed in the HANA studi

2) DATABASE PROCEDURE PROXY created in the ABAP Developement Tools.



The Top-Down approach enables developers to create a procedure in ABAP DE.

~~AMDP is implemented~~

ABAP serve as sole Master for developing managing & calling DB procedures.

AMDP provided as methods of global classes marked with tag interfaces.

(*) HANA procedure created at the first call of the AMDP method.

(*) Transport only required for the AMDP class.

Scanned with CamScanner

AMDP - Abap Managed DB Procedure.

- New feature in Abap allowing developers to write DB procedure directly in ABAP.

DP is a function stored & executed in the DB.

In SAP HANA its called SQL script.

Using AMDP allows developers to create & execute those DB procedures in the ABAP environment using ABAP methods & ABAP datatypes.

AMDP is specifically for HANA DB.

- To allow native consumption of HANA features from within the ABAP Layer, the HANA DB procedure language SQL Script has been integrated into the ABAP stack.

- AMDP is implemented in ABAP class methods (so-called AMDP methods) that serve as a container for SQLscript code.

- This approach enables the shipment of AMDP as same as any another method.

(CREATE PROCEDURE → CONSUME IT WITH SE38 Prog)

1) Classes with AMDP must use Interface IF_AMDB_MARKER_HDB

2) All AMDP methods parameters must be passed by value (like RFC).

3) All AMDP parameters must be tables with elementary components of scalar types.

4) AMDP support (secondary) DB connection to the primary DB via input parameter CONNECTION

Scanned with CamScanner

(type DBCON_NAME).

```
class zcl_amdp_simple_00 DEFINITION

  PUBLIC
  FINAL
  CREATE PUBLIC.

  PUBLIC SECTION
  INTERFACES : IF_AMDP_MARKER_HDB

  METHODS get_customer_infos
    IMPORTING
      VALUE(<input>) TYPE <input type>
    EXPORTING
      VALUE(<output>) TYPE <output_type>
    RAISING <amdp_exception>

  ENDCLASS.
```

```
METHOD get_customer_infos
  BYDATABASE PROCEDURE
  FOR HDB
  LANGUAGE SQLSCRIPT
  OPTIONS READ-ONLY
  using <dictionary-artifacts>

  get_customer_info =
  SELECT ---- FROM ----;

ENDMETHOD.
```

1) Indicates method contains DB specific code → BYDATABASE PROCEDURE

2) DB platform → FOR HDB

3) DB Procedure Language → LANGUAGE SQLSCRIPT

4) Used ABAP Dictionary Tables → using <dictionary-artifacts> → meaningful SQLScript coding

5) Native SAP HANA SQL SCRIPT SOURCE CODE

Scanned with CamScanner

AMDP consumption like any other ABAP method call.

AMDP Runtime

At first call of an AMDP, several SAP HANA artifacts are created in the SAP <SID> schema such as the SAP HANA DB Procedure.

```
Report zr_amdp_01_call

Data(lo_amdp) =NEW zcl_amdp_
                simple_00( )

lo_amdp -> get_customer_info(
    IMPORTING
      et_customer_info = DATA
        (lt_result).
```

SE24 To create class / structure in SAP ABAP.

ZCL -> start with this to directly create class

TT -> TN.

ADBC (Abap Database Connectivity).

1) connect to any DB using DB specific queries. (Native SQL queries).

2) ADBC is DB independent. (not specific to HANA DB).

3) used for below SAP NW 7.4. (if above then CDS or View Proxy)

(App Prog Interface)

4) ADBC is an API → API is a set of methods & classes which has some logic. when we expose it using a framework it can be consumed

Scanned with CamScanner

* CL_SQL_STATEMENT → Clears to establish connection to DB.

* CREATE STATEMENT () → Passes ~~SELEQ~~ SELECT Query which will be executed in DB directly

* SET_PARAM_TAB() → Points to the internal table where the Data would come & reside.

* NEXT_PACKAGE () → Triggers the query to DB, Fetches data & assigns to Internal Table.

8 Important Steps in ADBC Programming.

1) Set the connection to the DB (HANA/NON HANA).
2) Instantiate SQL Statement.
3) Prepare Native SQL statements.
4) Issue Native SQL call.
5) Assign Target variable for Result set.
6) Retrieve Result set.
7) Close the query, release source.
8) Close the DB connection.

Scanned with CamScanner

lr-connection -> create_statement

ls-statement = (select M.M ATNR, M.MTART etc

lr-sql-statement -> execute_query(ls-statement)

lr-result -> set-param!table(
exporting-itab-ref = litab-ref).

lr-result -> next_package()

lr-result -> close().

IDA-ALV Integrated Data Access / Abap List Viewer.

(SALV-IDA)

Conventional/Procedural.

Reuse_Alv_List_Display -> Reuse_Alv_Grid_Display.

→ Using CL_GUI_CUSTOM_CONTAINER ?

OOPS ALV — CL_SALV_TABLE

ALV-IDA. ↵

(why used?) → For code pushdown.

Existing / Traditional ALV

CLS
INT
DB
ALV → reuse-alv-list disp-sel
OOALV → SE24 CL_GU IF

HANA.

SALV-IDA.

Scanned with CamScanner

1). Existing ALV has more functionality on the application layer.

2) This makes it very slow as full data is being selected & sent to the ALV framework, which translates that into display on the GUI container.

3) Since the entire data is being selected beforehand, the framework has to parse the data as required.

SALV - IDA {Intro}

1) Stands for SAP ABAP List Viewer - Integrated Data Access.

2) with HANA DB (aka) HDB (aka) In memory DB, many of the ~~operating~~ operation which can be executed on the front end can be send to the DB - the code pushdown.

3) The new SALV-IDA works more on code push down concept.

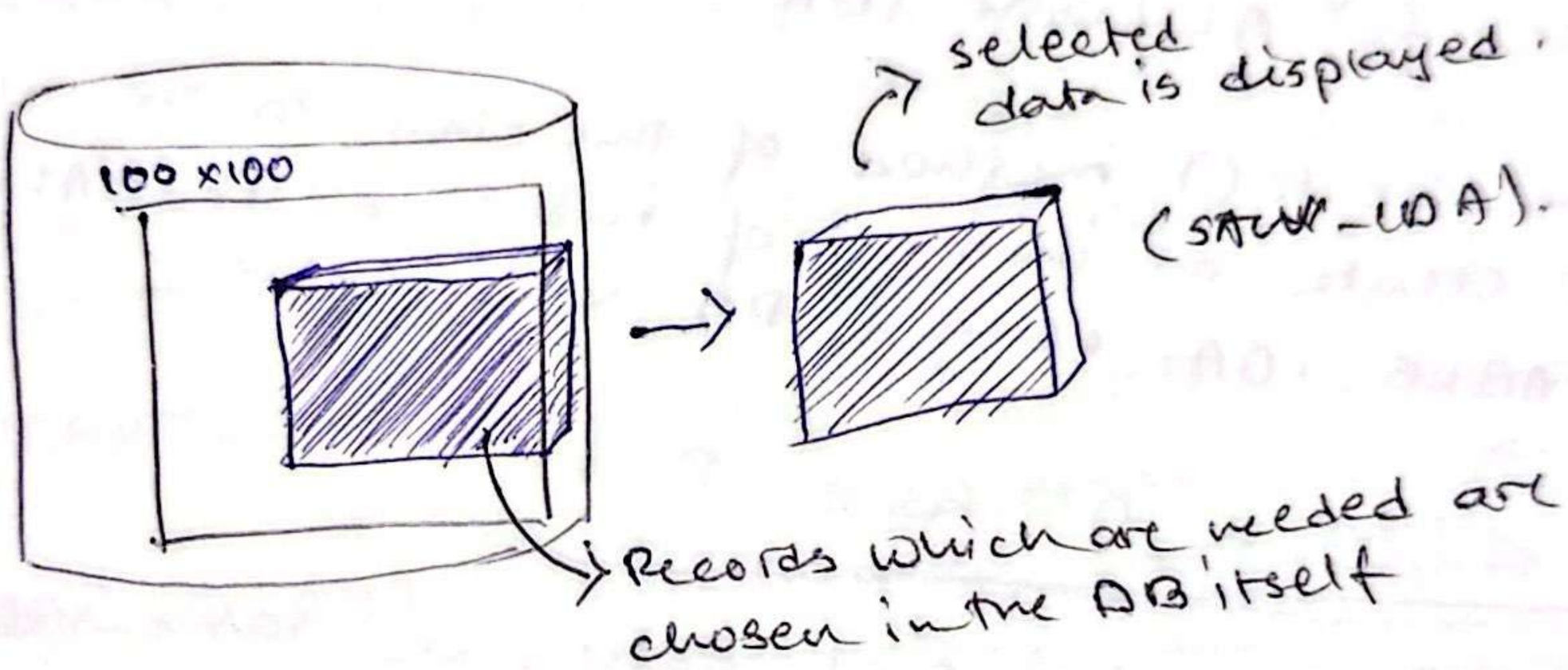
4) Means we dont select the data & send that to the ALV, instead we generate the ALV for the DB table, DB view or a CDS views.

↳ The IDA framework then analyze the required columns, analyze the filters to get the required where condition & then executes the select query.

Scanned with CamScanner

SALV IDA for SAPGUI is an advanced list control to display in tabular format from variety of data sources represented in ABAP Data dictionary, for ex:- SAP HANA artifacts (calc view) represent as external views.

In memory DB like SAP HANA can lead to significant performance improvements when processing large quantities of data.



* Users should benefit from the required data being displayed more quickly and the ability to manipulate their view of that data in different ways.

* In order to make this possible in the ALV environment SAP, has designed a special version of the List Viewer, the SALV IDA

* This variation of the ALV is optimized for use with SAP HANA.

* Only VISIBLE / REQUIRED rows are copied from DB.

Features of SALV-IDA

1) Real time.

2) Only visible rows are copied.

3) Data volume transferred from DB to App server is drastically reduced.

Scanned with CamScanner

ii) Code to Data Paradigm (alen Code Pushdown)
Data intensive operations moved to DB.

Tools : CDS view, HANA view (using SQL script).

ALV with IDA

A factory class CL_SALV_GUI_TABLE_IDA is used for ALV with IDA.

The create() method of this class can be used to create an instance of interface IF_SALV_GUI_TABLE_IDA using a DB view/table.

Eg: Using a DB table BSEG.

DATA (lo_alv) = cl_salv_gui_table_ida =>
create (iv_table_name =
'BSEG'/ your table)

lo_alv -> fullscreen () -> display ().

Eg: Using a view ZDCS_VIEW

→ same except create ('ZDCS_VIEW').

SALV-IDA Methods

i) Field catalog ()

set_field_header_text()
get_available_fields()
set_available_fields().
disable_aggregation() / sort() / filter()

Scanned with CamScanner

2) Display options()
enable_alternating_row_pattern()
set_title().

default_layout()
standard_functions()
Toolbar()
Handle_hot_spot().

SALV-IDA with tables (FULLSCREEN ↳ Available in Interface)

START-OF-SELECTION. (05E24) → class Ctrl+(-) ↳ choose

DATA(I_OBJ) = CL_SALV_GUI_TABLE_IDA => CREATE
IV_TABLE_NAME = 'LFA'
IO_GUI_CONTAINER =
IO_CALC_FIELD_HANDLER=
).

CATCH CX_SALV_DB_CONNECTION..
" CX_SALV_DB_TABLE_NOT_SUPPORTED.
" CX_SALV_IDA_CONTRACT_VIOLATION.

Ctrl+(-) ↳ choose

I_OBJ → FULLSCREEN( ) → DISPLAY

CATCH CX_SALV_IDA_CONTRACT_VIOLATION

(Fullscreen → is an interface, which is an instance method).
in SE24 → IF_SALV_GUI_TABLE_IDA.

Scanned with CamScanner

LFA1 (Supplier Master).

mandt lifnr landl name1 ort01 pstlz sort1 ~~adrnr~~
erdat ernam spras

X

LFB1 . (vendor Master)

mandt lifnr . bukrs zterm webtr.

X

(Q1) Two table . → (Q2) DB view → . (Q3) create a lock object for write LOCK. → (Q4) Create a elementary search Help → (Q5) Create a Table type using a structure .

MANDT

LIFNR	Char 10	Acc No
LANDI	Char 3	Country key
NAME1	Char 35.	Name 1
ORTO1	Char 35	city nam
PSTLZ	Char 10	Postal code
SORT1	Char 10	Sort field.
ERDAT	Date 8	Date on which record for.
ERNAM	Char 12	Name of person who created
SPRAS.	Char 35	House No & Street.

LIFNR	Char 10	Acc No.
BUKRS	Char 4	comp code.
ZTERM	Char 4	Terms of payment key.
WEBTR.	~~Char 13~~ CUR 13.	Bill Exce Limit.

Scanned with CamScanner

ALV Reports → ABAP List Viewer ⇒.

1) Classical Reports.
2) Interactive Reports.
3) ALV reports.

Procedure to develop ALV Report :-

1) Structure declaration with required fields.
2) Internal Table & Work Area declaration.
3) Selection Screen designing.
4) Fetch the data from DB table to internal table
5) Field catalog designing. Addition from classical
6) Display the data.

ALV Report based on selection screen.

Q1) MATNR ERSDA ERNAM.

↑
Enter Material No,

MATNR ERSDA ERNAM (DE).

FIELD CATALOG 1) Z9206_ALV REPORT.

SLIS_T_FIELDCAT_ALV → Internal Table
SLIS_FIELDCAT_ALV → Work Area } 3 structures of Type SLIS.
SLIS_LAYOUT_ALV → Work Area.

REUSE_ALV_GRID_DISPLAY
is the function module for display data in ALV output

Scanned with CamScanner

ALV report to fetch data / records from 2 standard tables.

table1 - MARA.
table2 - MARC.

Can 3 from MARA & 2 fields from MARC.

MARA

MATNR

ERSDA

ERNAM..

MATNR.
BSTAT -> BSTAT_D

ERSDA.

WERKS -> WERKS_D

2 STRUCTURES

2) 27206 - ALVDUO.

3) Intraction Report. (CFB1)

{Vendor No (LIFNR)

POSTAL CODE (PSTLZ)
REGION.

Vendor Name (NAME1)
(LAND 1)

CITY (ORT01)

CIFNR
BUKRS company

Interactive Report :

Events

Initialization

At selection screen.

start of selection.

Top of page.

End of page

End of selection.

27206_INTRPT.

IR Events

At line selection

At user command.

set pf status,

LFA1 (Supplier data)

LFB1 (Vendor)

LIFNR

NAME1

LAND1

ORT01

PSTLZ

LIFNR

BUKRS.

PERNR.

LFC1

BUKRS

GJAHR

FDDAT.

Scanned with CamScanner

SAP - Systems Applications & Products.

SAP is a ERP (Enterprise Resource Planning) software.

ABAP (Advance Business Application Programming). - is a coding language for SAP to develop RICEFW objects.

Reports -
Interfaces
Extensions
Forms
workflows.

ABAP 4 data dictionary is used to describe logical structure of the objects.
Also used to show underlying relational DB

Transparent Table - It has one to one relation with table in DB.

Pooled Tables. It has many to one relation with tables in DB.

Scanned with CamScanner

BDC (Batch Data Communication)

↳ automatic procedure to transfer large /extend data into SAP systems.

Queue files is the central component of the transfer, which receiver the data through batch input programs & g

Data classes ⇒

Master Data : The data in this class seldom change.

Transaction Data : The data can be changed often in this class.

Org Data.

System Data - ~~us~~ data used by R/3 system itself

Data Element - describe an elementary type by defining data type, length & possibly decimal places.

Data Element	Domain
Gives the field labels & documentation for table fields.	Describes technical attributes → data type, length of a table field.

(RAISE, TRY - CATCH, CLEANUP)

Scanned with CamScanner

DML	DDL
Select, Insert, Update, Delete.	Create, Drop, Alter

MODULE POOL PROGRAMMING.

A user dialog is any form of interaction b/w the user & the program & could be any of the following

1) Entering a data
2) Choosing a menu item
3) Clicking a button
4) Clicking or double clicking an entry.

1) Dialog program is also used when we need to navigate back and forth b/w screens.

2) Dialog programs are created with type as 'M'. Module Pool. They cannot be executed independently & must be attached to atleast one transaction code in which you specify an initial screen

3) A dialog program allows us to ~~think~~ work interactively with the system & to change the contents of the DB tables. Each dialog program has a certain sequence of screens that are processed by the system one after the other.

Scanned with CamScanner

All objects are linked hierarchically to the main program and are executed in a sequence dictated by the Dialog main program.

COMPONENTS USED ARE ⇒.

1) Transaction Code ⇒

a) Transaction Code starts a screen sequence.

b) we create T-Code in the Repository Browser in the ABAP workbench or using Transaction (SEQ 3)

c) A transaction code is linked to an ABAP program & an initial screen.

d) you can start a screen sequence from any ABAP program using the CALL SCREEN statement.

RICEFW object ⇒ Report, Interface, Conversion, Enhancement, Form, workflow.

GAP (if dev req) | FIT (if dev req) | ?? → Feasibility Analysis (Y/N) (Time needed)

Functional Design Document (FDT)

{ Functional requirement in detail. }

ABAP (OOPS TRAINING).

1) what is object orientation?

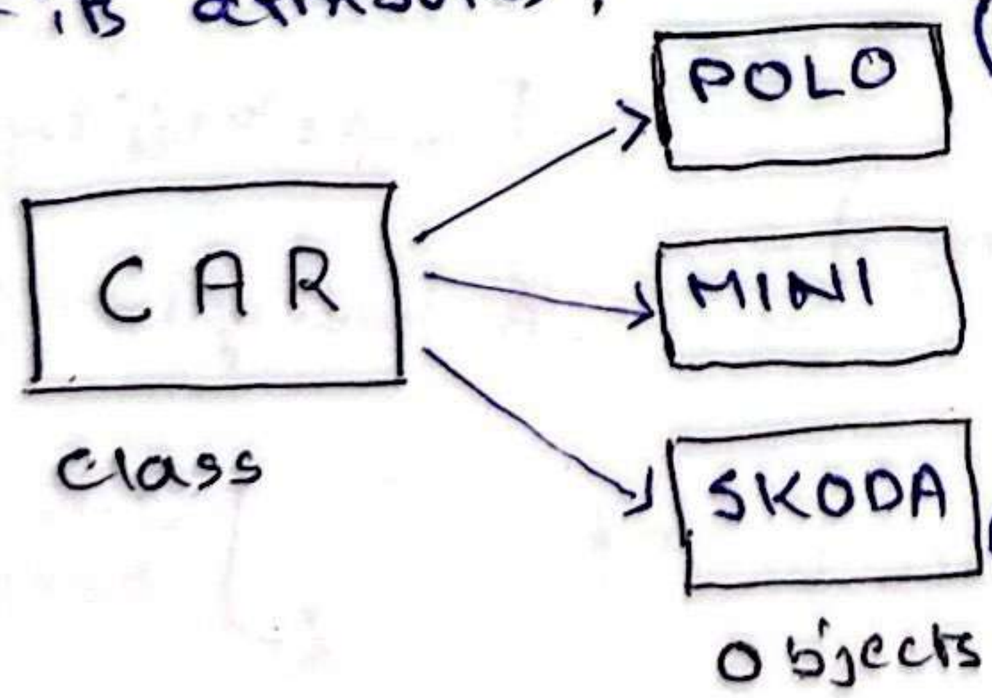
Object-oriented programming, is a problem-solving method in which the software solution reflects objects in the real world.

Key Features =>

a) Instantiation
b) Encapsulation & Abstraction
c) Inheritance.
d) Polymorphism
e) Interfaces
f) Reduction in code redundancy.
g) stress on data security & access.

Objects → an object is a section of source code that contains data (attributes) and provides services (via methods). They help programmers to map a real problem and its proposed software solution on one to one basis.

Classes → classes describes objects i.e objects are run time instances of classes. Each instance (object) of a class has a unique identity & its own set of values for its attributes.



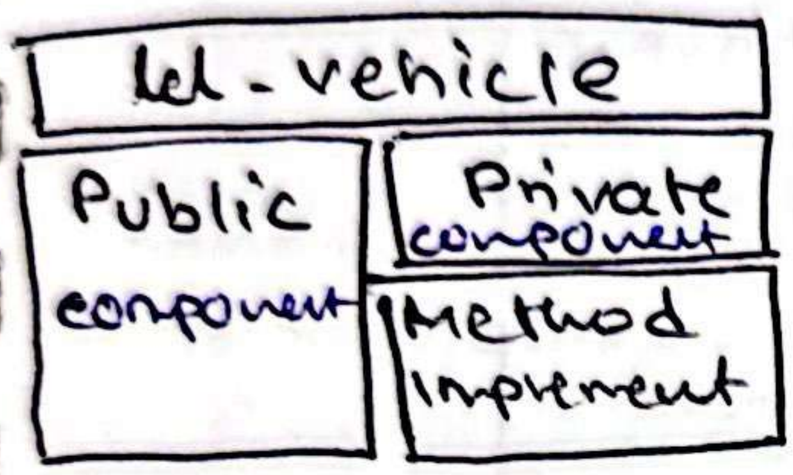
* A class is a set of objects that have the same structure and same behavior.
* A class is therefore like a blueprint, in accordance with which all objects in that class are created.

Scanned with CamScanner

ABAP CODE

```
CLASS lcl_vehicle DEFINITION
...
ENDCLASS
```

```
CLASS lcl_vehicle IMPLEMENTATION
...
ENDCLASS.
```



→ The components of the class are defined in the definition part.

→ components can be ⇒
1) Attributes 2) Methods 3) Events
4) Constants 5) Types 6) Implemented interfaces.

→ Only methods are implemented in the implementation part.

"CLASS" statements cannot be Nested as we cant define a class inside another class.

COMPONENTS OF CLASSES ⇒

There are two main components of classes :

1) Instance and static components.

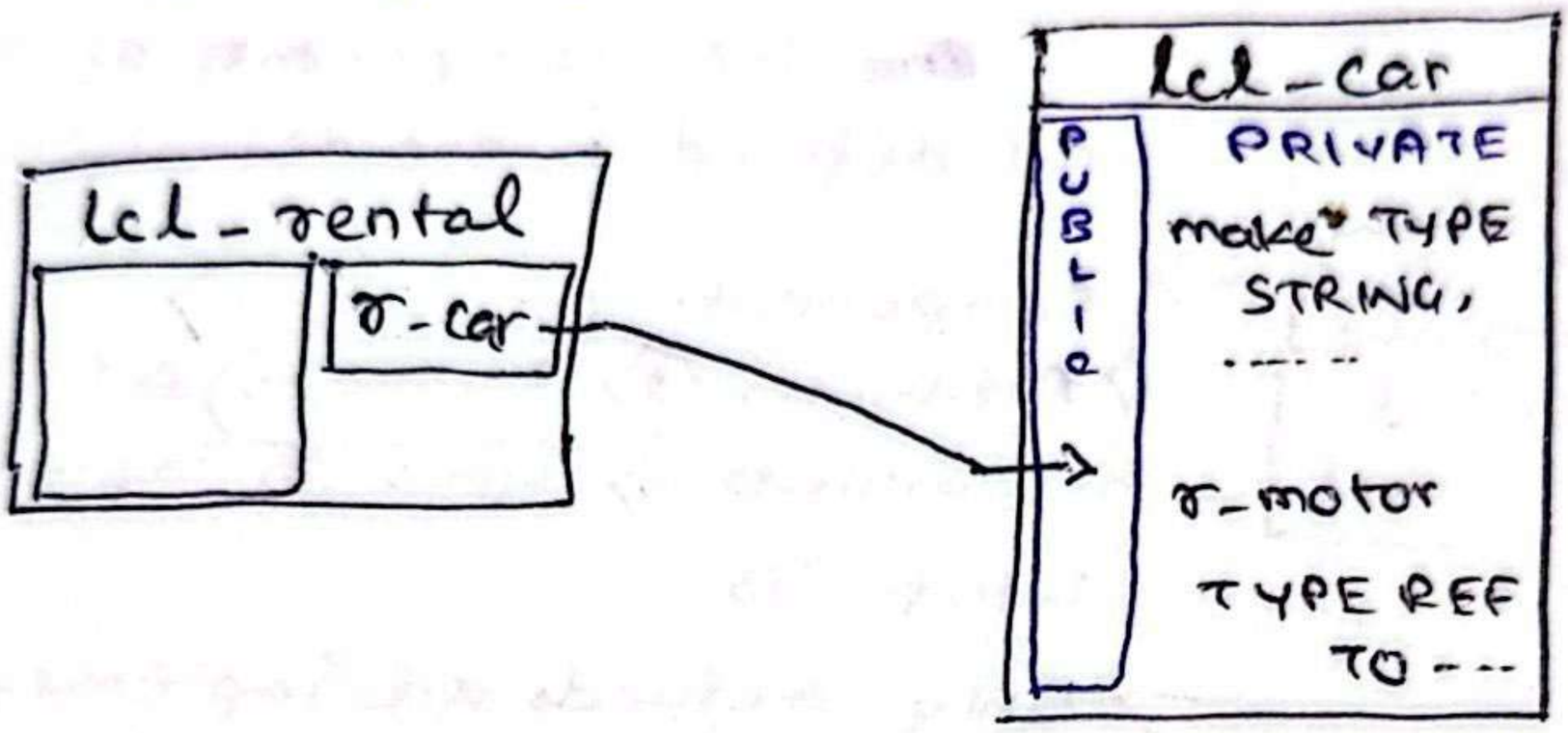
Instance Components	Static Components.
Data : For instance attributes	Class - Data : For static attribute
Methods : For instance method	Class - Methods : For static methods
Events : For instance ~~methods~~ events	Class - Events : For static events.
Instance components exist separately in each instance (object) of the class.	Static components only exists one per class & are valid for all instance of the class.
To access instance components instance component selector (→) is used.	To access static components static component selector (=>) is used.

Scanned with CamScanner

Attributes → described the data that can be stored in the objects of a class.

Attributes can have any kind of data type:

→ C, N, I, P, ..., STRING
→ Dictionary Types.
→ User defined Types.
→ TYPE REF TO - defines a reference to an object, in this case "r_car"



CLASS <class name> DEFINITION
...

TYPES : <normale Type definition>.
CONSTANTS : constant TYPE <type> VALUE <value>.

DATA : variable 1 TYPE <type>,
variable 2 TYPE <ddic-type>,
variable 3 LIKE variable1,
variable 4 TYPE <type> VALUE <value>,
variable 5 TYPE <type> READ-ONLY.
variable 6 TYPE REF TO <classname>,
variable 7 TYPE REF TO <interface>.

CLASS-DATA: only-once TYPE.....

ENDCLASS.

Scanned with CamScanner

- In classes, ~~you~~ we can only use the TYPE addition to refer to data types.
- The READ-ONLY addition means that a public attribute declared with DATA can be read from outside, but can only be changed by methods of the class.
- Currently ~~you~~ we can use the READ-ONLY addition only in public visibility section (PUBLIC SECTION) of a class declaration or in a interface definition.

Attribute & Visibility ⇒

Public Attributes

↳ can only be viewed & changed by all users & in all methods.

↳ direct access.

Private Attributes

↳ can only be viewed & changed from within the class

↳ No direct access from outside the class.

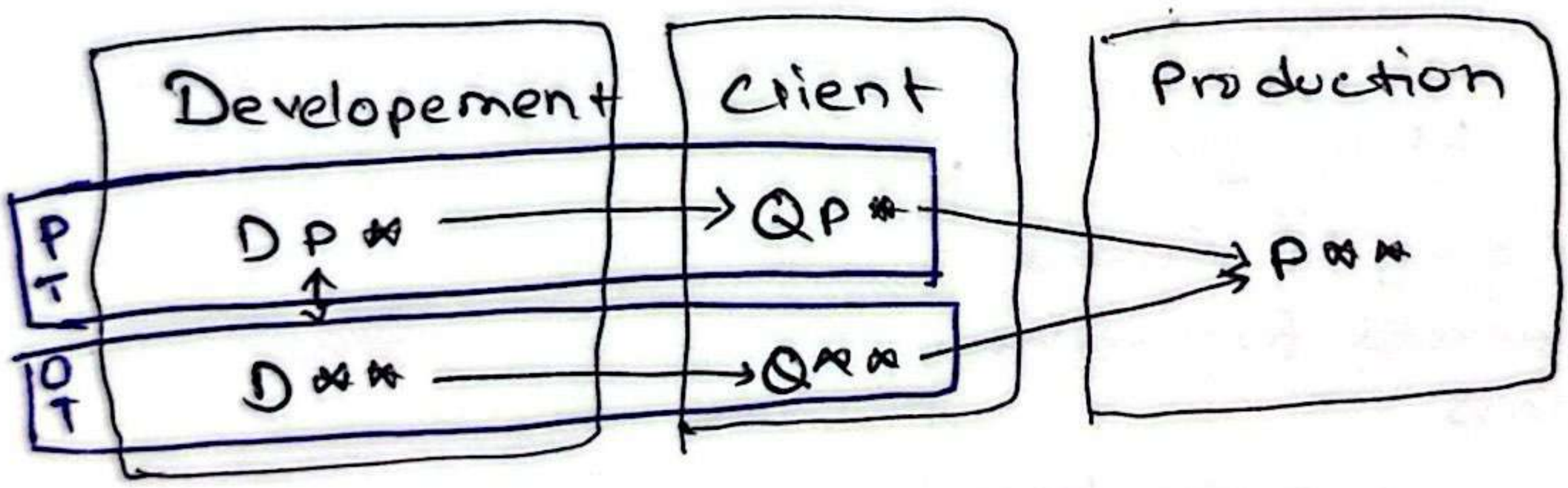
Scanned with CamScanner

SAP Developer & Customizer (IFX)

→ Basis Knowledge.
→ Developement Guidelines.
→ Design & Governance Processes.
→ Release & Change Management.
→ Projects & P2OPs
→ Roles & Authorizations.

BASIS.

Overview of all SAP systems can be found in ⇒
↳ Goto/sops & page System Administration.



Infineon has both 3 & 5 System Landscapes

> Project Track (PT) → Projects & changes
> Operations Track (OT) → Emergencies & Bug Fixes.

General Client → 100.

(Pg 8/9)

(Pg 25)

DEVELOPEMENT GUIDELINES (4/1/22)

Infineon is very careful on touching standard code or any action that might have impact on future upgrade activities. ∴ (IFX)

Scanned with CamScanner

Following are not allowed ⇒

> Modifications
> Implicit Enhancements.
> Cops of SAP standard.
> Pre / Post / Over write Exit methods.

↳ Can be allowed with good justification & special approval.

Double checking that IFX guidelines are not violated ⇒

> The design we plan to implement ⇒ what & how to do it.

DESIGN REVIEW → mandatory for projects.

> MANUAL CODE REVIEW → spot checks
↳ guideline follow
↳ frame works
↳ utility class

> The code we implemented by ABAP Test Cockpit (ATC) → automatically checking by SAP standard tool (mandatory for everything as with priority 1 & 2 findings no transport can be released).

Topics to be kept in mind for High Quality Programming ⇒

1) Security
↳ Authorizations and secure programming are mandatory.

Scanned with CamScanner

2) Performance → i) Keep performance rules in mind make selection screen parameters mandatory
ii) Try to push down code to DB on S/4 HANA.

3) Documentation → i) comment why we are doing something, not what we do.
ii) we need to write things that might help in Operations & for maintainability.

4) Readability & Transparency →
~~Abo~~ i) Avoid complex coding
ii) Use pretty printer
iii) Never use ABAP Macros.

5) Internationalization → Main language is English.

6) State of the Art Developement →
i) Avoid obsolete statements
ii) New Abap language needed
iii) Remember change b/w R/3 & S/4 (eg tables)
iv) Never use pseudo comments ("#EC")

7) Robustness →
i) Do proper error handling.
ii) Implement stable code.

NAMING CONVENTIONS ⇒

OBJECT	NAMING.
1) Fields in Append Structure	ZZ_XXXXXXX.....
2) Class	ZCL_mmmm_XXXXX
3) Interface	ZIF_mmmm_XXXX

Scanned with CamScanner

4) Table	ZTmm - x x x x x .....
5) Function Groups (FG)	Zmm[mm] - x x x x
6) Tcn for TMG	Z <table-name without - T>

eg => Table Name ZTMM_MATS
Then FUGR = ZMM_MATS

General Rule:

- ZMM [MM]_ xxxxx·xxxxx
- MM (Module)
- [MM] (Optional submodule)
- xxxxxxxx (Meaningful Description)

Which kind of Hardcoding should be used?

> Distinguish b/w hard coding due to coding reason or hard coding something that has functional impact and can easily change in future.

Which options exists to avoid hard coding?

> Text Element (for shorter texts)

> Message classes (for feedback like success or Error)

> SO 10 texts (for longer texts like footers or mail bodies)

> Selection screen Parameters (for various topics depending on situations).

> constants of classes. (for repeated technical hardcoding where it makes sense)

> SAP Distribution Lists (for mail receivers).

Scanned with CamScanner

> Logical connections (for RFC destination)

> Global variables table (for conditions / mappings)

> Own variables table (If one solution has many records).

Q Why use Global variable Table? (Ons SAP/PIO)

Main idea of Global variable table →

1) To avoid hardcoding & have small, customizing options for custom developements.

{with help of this table it is possible to only change the table data if needed without change the code.}

Q What to consider when going for secure programming

> Take care on injections possibilities (OS/SQL/ABAP)

> Dont trust on user input

> Always validate & do not directly reuse it for generic calling something.

> check on appropriate authorization explicitly

> Avoid coding for special situations/users. [No backdoors]

> Only upload doc type that do not support coding like Macros.

Allowed Types ⇒ CSV, TXT, JPG, PDF.

Scanned with CamScanner

There are two types of AUTHORIZATION GROUPS.

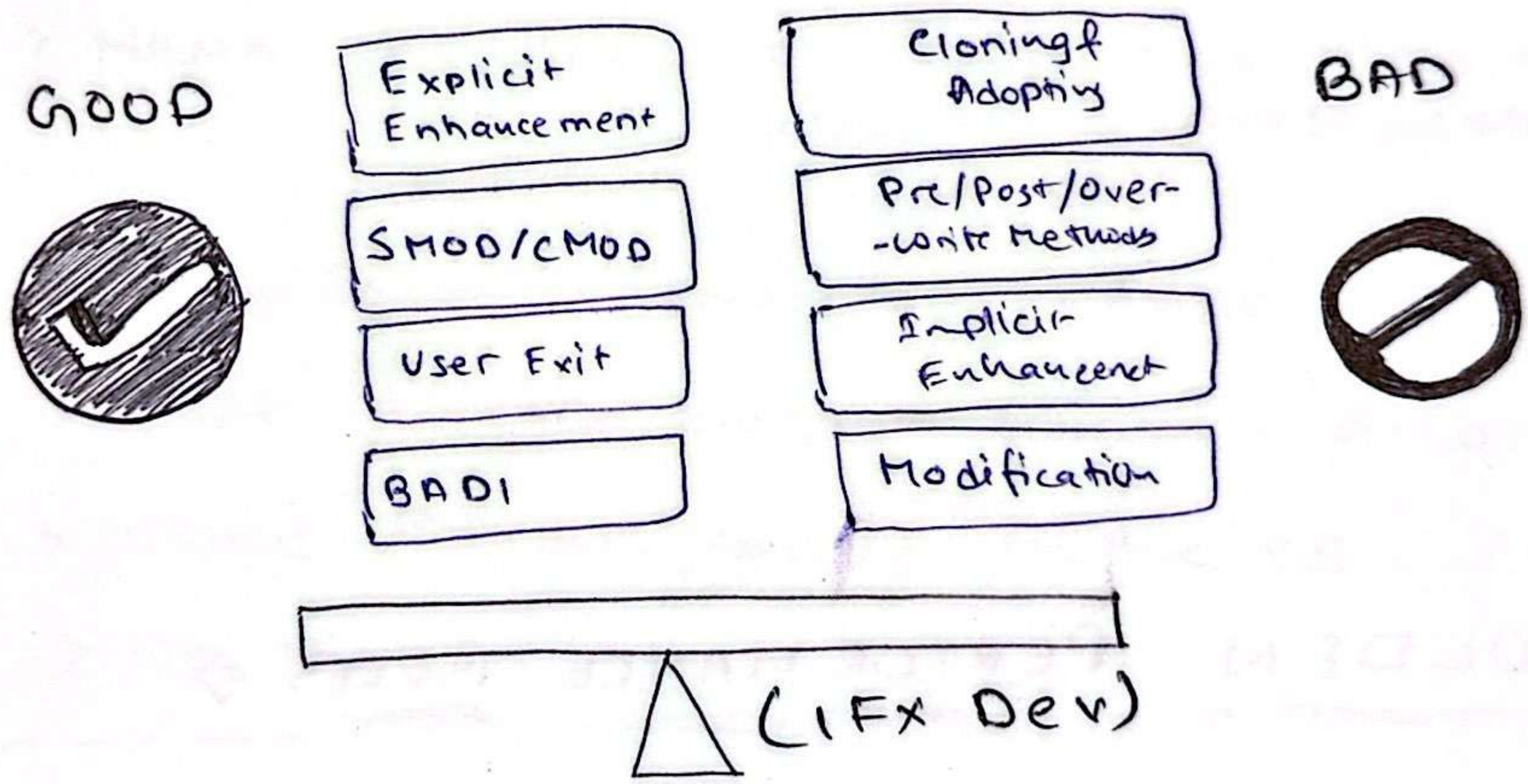
→ For Tables

→ For Reports.

Creation of AUTH GRP & assigning objects to it depends on Developers.

→ TABLE : SE54 Creation & SE11 with utilities & Assign group Authorization.

→ REPORT : SM30 & VTPGP for creation SE38 to Properties / Attributes to assign.



What to do if we have to use BAD IFX Dev

↳ Get in contact with

SAP-IFX-Design@infineon.com

↳ prepare a description of the use case & reason for the situation.

↳ Additional approval of IT ES Line Head.

Scanned with CamScanner

Enhancement concepts on oneSAP systems?
(PIO/PRC/PES)

1) On the newer system we use BADI LAYERING CONCEPT.

2) whole concept is using standard by creating custom BADI's, its easy to activate & deactivate solutions. (Pg 48)

GOLDEN PERFORMANCE RULES ⇒

1) Keep the result set small
2) Minimize amount of transfered data.
3) Minimize the search overhead.
4) In Non HANA : Keep load away from DB
HANA : Code Pushdown on DB as much as possible

BEST PRACTICES WITH INTERNAL TABLES :-

Scanned with CamScanner

→ No Appends, but inserts into sorted IT.
→ No sort statement within loops.
→ No nested loops to process.
→ Deleting adjacent duplicates only after sorting.
→ Check if table is Empty before using F for all entries.
→ Clear internal Tables - once data not needed in memory anymore.

Which DB related Implementations are not allowed at IFX?

> Native SQL
> HANA CDS views (only ABAP CDS views are allowed - no direct access to HANA DB)
> ABAP Database Connectivity - ADBC
> ABAP Managed Database Procedure - AMDP
> Stored Procedure created on the DB
> DB Hints.

What to consider on ABAP CDS views on S/4 HANA systems?

> Its highly encouraged to use new options like ABAP CDS views to pushdown to the DB.

> If there exists a CDS view ~~name~~, a SQL view will be created automatically, take care to read in your program always from the CDS view

Scanned with CamScanner

> Additionally there is a distinction b/w socalled interface views & consumption views.

ERROR HANDLING =>

1) Think of wrong input or error possibilities during implementation - what to do in case of them.

2) check on SY-SUBRC

3) Check on class exceptions with [TRY-CATCH] Block & give propriate handling.

When to use Application Log? (SLG1) (For Feedback)

1) For reports that should continue even after an error (Mass Update) we can use the Application Log for logging.

2) Utility Classes that offers methods for it → use them
ZCL_CA_COMM_UTILITIES (PIO)

3) Application Log is only for technical monitoring & not business related logging

DO NOT USE WAIT STATEMENTS -

i) Its not allowed to use WAIT statements with fix amount of time. (eg: WAIT UP TO 1 SECOND)

ii) Use it maximum as the _wait parameters for BAPI's or research for other options like ENQUEUE_ xx

Scanned with CamScanner

How to document & what?

> Major steps should be documented
→ why something has been done & Not what.

> Complex DB accesses should be briefly described.
→ Especially on S/4 HANA the code pushdown to DB is highly recommended, selects can get more complex with various joins & calculations.

> If exceptional IMPORT / EXPORT to Memory is allowed, it always needs inline doc where the other part is used.

> If SAP Notes have manual implementation in Z-namespace → always give header with comment with note #

How to maintain technical documentation & attributes/properties?

> All coding objects (like program, class, function module, etc) & tables / views should need a technical documentation := F1 Help Doc.

> Find & Create it using Goto → Documentation when being in the corresponding object.

> Additionally the attributes/properties of such objects need to be maintained to see which type of program it is. (Normally a customer program)

Along with this a meaningful short text description & the application component it belongs to. (MM, SD, BC etc)

Scanned with CamScanner

V.V.I

How to maintain change log & change history?

> On creation and change, the change log needs to be maintained at top of the object.

PATTERN → OTHER PATTERN →

Z_IFX_HEADER

Z_IFX_HISTORY

> Remove not needed, code instead of commenting & mark all the changes with new/changed tag.

How does Documentation look like then?

> On creation, (TOP), use Z_IFX_HEADER pattern & fill it out with all detailed information.

```
** ------------------------------------------------------
** & Description              : Program Desc
** &
** &
** &
** &
** & Requeste By              : Developer Name
** & Author                   : current Dev
** & Date created             :
** & Preconditions            : N/A.
** &
** & Reference                : BR3334
** & SAP Version              : SAP S/4 HANA 1709
** & Request #                : DPOK904800.
** ------------------------------------------------------
```

Scanned with CamScanner

On each change, add Change Log History by using pattern Z_IFX_HISTORY.

FIORI

CDS Data.

(Fiori Elements) (Types of App)

What is Entity Type ? Work Area.
What is Entity set ? Internal Table.

SAP Fiori element is a framework that comprises the most commonly used floorplan template dis designed to:

i) Speed up dev by reducing the amount of frontend code needed to build SAP Fiori apps.

ii) Drive UX consistency & compliance with the latest SAP Fiori guidelines.

Refer -> SAP FIORI DESIGN GUIDELINES.

Q) What is (SAP) Fiori Elements App ?

An SAP Fiori elements app is an application that leverages SAP UI5, SAP UI5 ~~concept~~ control & SAP UI5 model view controller (MVC) concepts. In a plain SAP UI5 or a freestyle SAP UI5 app, all the views & controllers are part of your project.

We will do coding using Annotations.

Scanned with CamScanner

SAP Fiori elements are libraries which are built on top of SAP UI5 & provide designs for UI patterns & predefined templates.

Fiori Application development involves Metadata driven approach which means creating application based on OData service & annotations & requires no or less Javascript UI coding.

Annotations for OData service can be created by means of any one of the following ways.

1) CDS view Annotation – Create metadata Extension for CDS view & extension will have all UI ~~Appli~~ Annotation for Fiori elements.

2) Local Annotation – In WebIDE full stack we can create Local Annotation file in UI5 project folder.

4 types of Fiori Elements :-

1) List Report & Object Page.

2) Worklist.

3) Overview Page.

4) Analytical List Page.

** Always execute fiori element file via Sandbox System. (as index.html doesn't run first, component.js does).

Scanned with CamScanner

npm i → install all usuable files.
npm install

1) List Report Pages – (In this blog, you will find more details about it).

2) Object Pages 3) Overview Pages (OVP)

4) Analytic List Pages. (ALP).

1) List Report (LR)

Users can view & work with a large set of items. This floorplan offers powerful features for finding & acting on relevant items. This often used an entry point for navigating to the item details, which are usually shown on object page.

LR Floor Plan

1) Header

2) Content Area. (Table Area).

3) Object. (Go to Open Guided Development)

Local Annotations.

To create a selection field :-

UI.SelectionField. → creates additional selection field.

(Add a new filter field on to the filter bar)

2) Object Pages:-

In selection fields, we are working with

1) Entity Type → as we are working one row at a time.

Scanned with CamScanner

Select Filter field → Click "Insert Snippet"

webapp → annotations.xml. (here we will see our added code snippet).

(* *) we have to write our code inside <Schema> tags

Q) Why Manifest.json is important?

→ Flow of the code/project is handled here.

3) Overview Page — is a data driven SAP Fiori app type & floorplan that provides all that information a user needs in a single page based on the user's specific domain or role.

It allows the user to focus on the most imp tasks, views, filters & react to information quickly.

Types of Analytical cards:

1) Analytical card.
2) List card.
3) Bar chart list card.
4) Link list card.
5) Table card
6) Stack card | Quick View card.
7) Custom card.

* UI5 (Creation) Fiori

Custom App — Standard App

Scanned with CamScanner