**SAP ABAP**

INTRODUCTION TO ABAP

ARCITECTURE

SERVICES OF SAP

CLIENT SERVER ARCITECTURE

1 TIER

2 TIER

3 TIER

WORK PROCESSES

DATA TYPES AND DATA OBJECTS

CONTROL STATEMENTS

STRING OPERATIONS

ABAP DEBBUGING

STRUCTURES

VIEWS

INTERNAL TABLES

1. Basic Program using DATA and TYPE.

DATA: LV\_A TYPE I VALUE 10,  
      LV\_B TYPE I VALUE 20,  
      LV\_C TYPE I.  
LV\_C = LV\_A + LV\_B.

write lv\_c.



1. Program using parameters

PARAMETERS: p\_a type i,  
p\_b type i.  
data lv\_c type i.  
lv\_c = p\_a + p\_b.  
write lv\_c.

Output: 

55

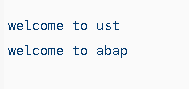
1. Program using constant keyword.

CONSTANTS c\_a type i value 10.  
\*c\_a = 20. " constants  cannot be changed  
write c\_a.

Output : 10.

1. Program using literals.

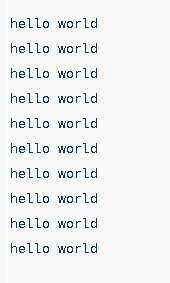
\*\*with literals  
write:/ 'welcome to ust',  
/ 'welcome to abap'.



Control statements.

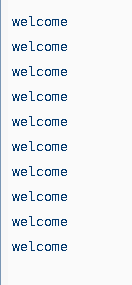
1. Using do program.

do 10 times.  
  write / 'hello world'.  
ENDDO.

Output: 

1. Using while program

data lv\_a type i .  
lv\_a = 0.  
while lv\_a < 10.  
  write / 'welcome'.  
  lv\_a = lv\_a + 1.  
ENDWHILE.



1. data lv\_a type i value 10.  
   if lv\_a eq 10.  
     write 'the given statement is true'.  
   else.  
     write 'the given condition is not true'.  
     ENDIF.

Output : 

1. data lv\_a type i value 10.  
   case lv\_a.  
     when  1.  
       write 'the given condition1 is true'.  
      when 5.  
        write 'the given condition2 is true'.  
     
      when 10.  
        write 'the given condition3 is true'.  
        when others.  
        write 'the given statment is not true'.  
   ENDCASE.

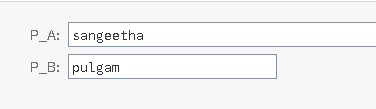
Output: 

1. PARAMETERS P\_A(30) TYPE C.  
   TRANSLATE P\_A TO upper CASE.  
   WRITE P\_A.

Output: 

SANGEETHA

1. PARAMETERS: p\_a(30) type c,  
   p\_b(20) type c.  
   data lv\_c(50) type c.  
   CONCATENATE p\_a p\_b into lv\_c  SEPARATED BY '&'.  
   write lv\_c.



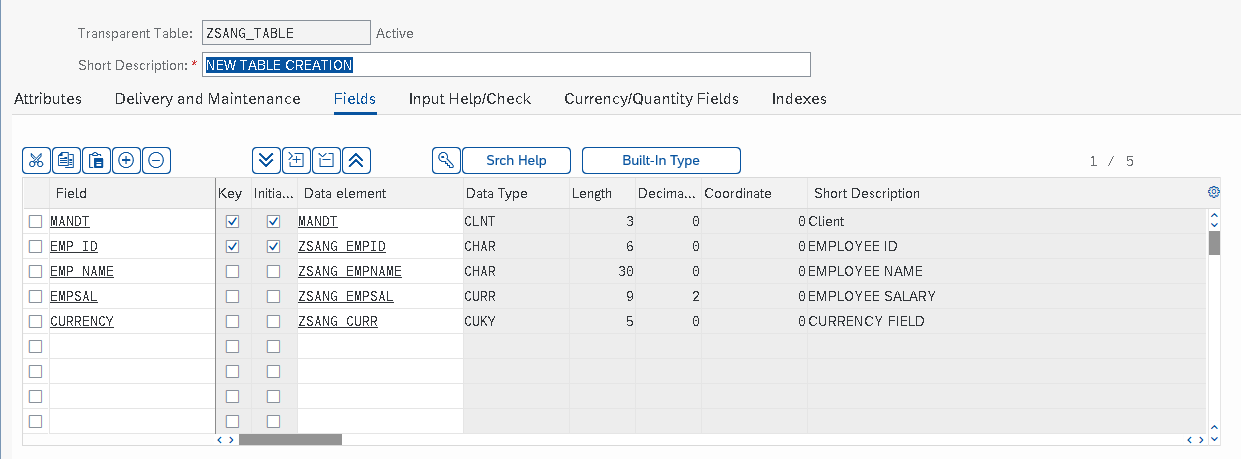


1. PARAMETERS P\_A(20) TYPE C.  
   DATA LV\_A TYPE I.  
   LV\_A = STRLEN( P\_A ).  
   WRITE LV\_A.

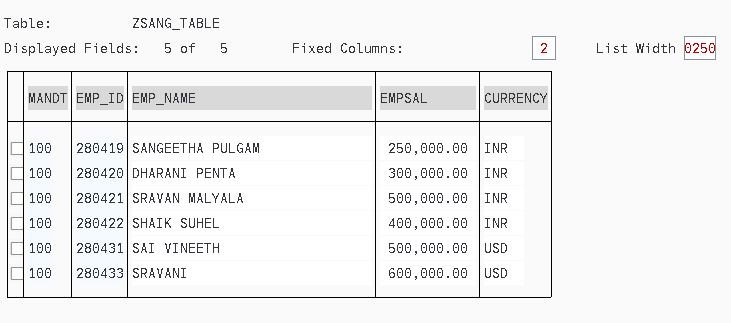




TABLES CREATION

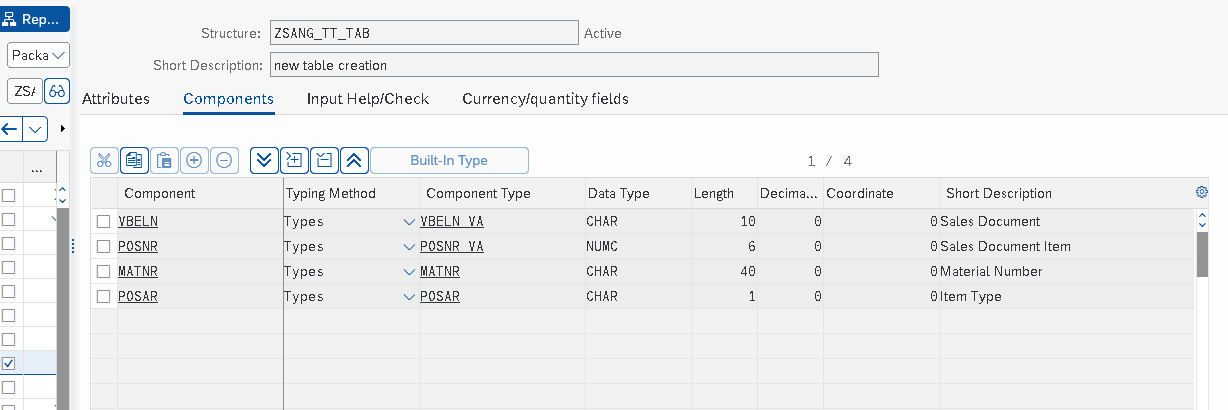


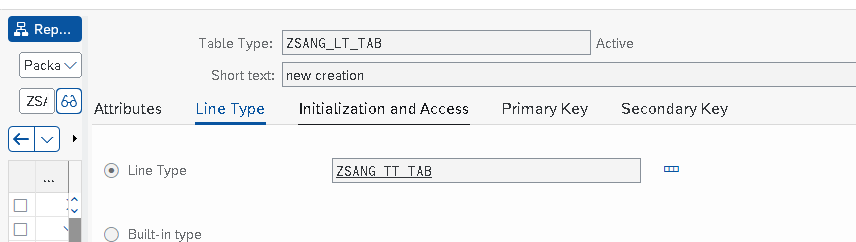
Output:

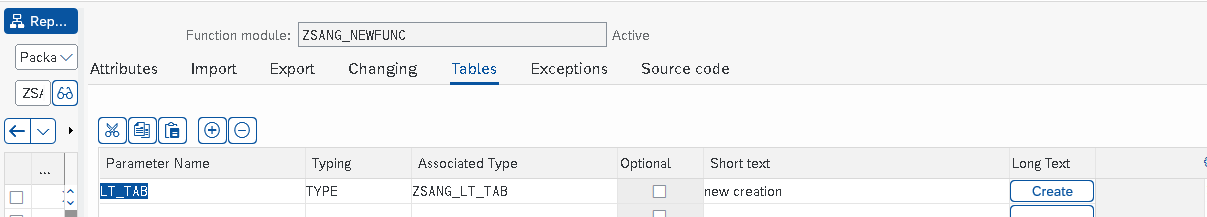


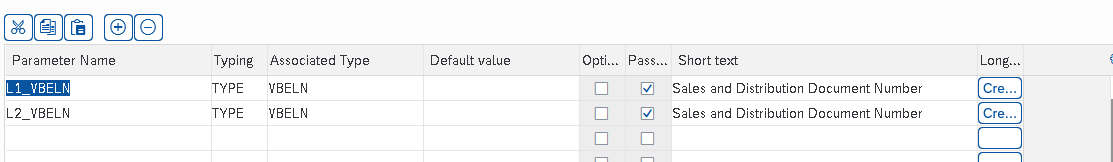
STRUCTURES

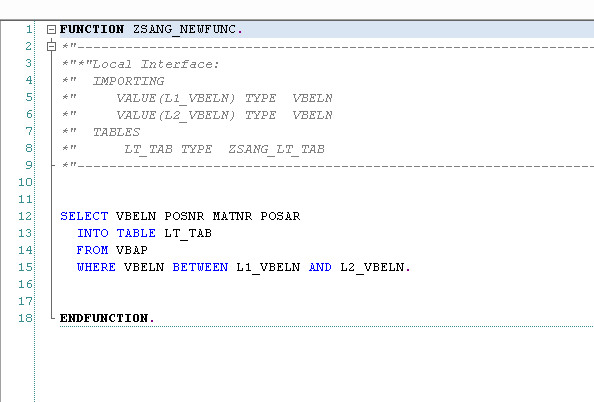
DATA : LS\_TAB TYPE ZSANG\_LT\_TAB,  
      WA\_TAB TYPE ZSANG\_TT\_TAB.  
SELECT-OPTIONS S\_VBELN FOR WA\_TAB-VBELN.  
  
  
  
CALL FUNCTION 'ZSANG\_NEWFUNC'  
  EXPORTING  
    l1\_vbeln       = S\_VBELN-LOW  
    l2\_vbeln       = S\_VBELN-HIGH  
  TABLES  
    lt\_tab         = LS\_TAB  
          .  
LOOP AT LS\_TAB INTO WA\_TAB.  
  WRITE: / WA\_TAB-VBELN,  
  WA\_TAB-POSNR,  
  WA\_TAB-MATNR,  
  WA\_TAB-POSAR.  
  ENDLOOP.





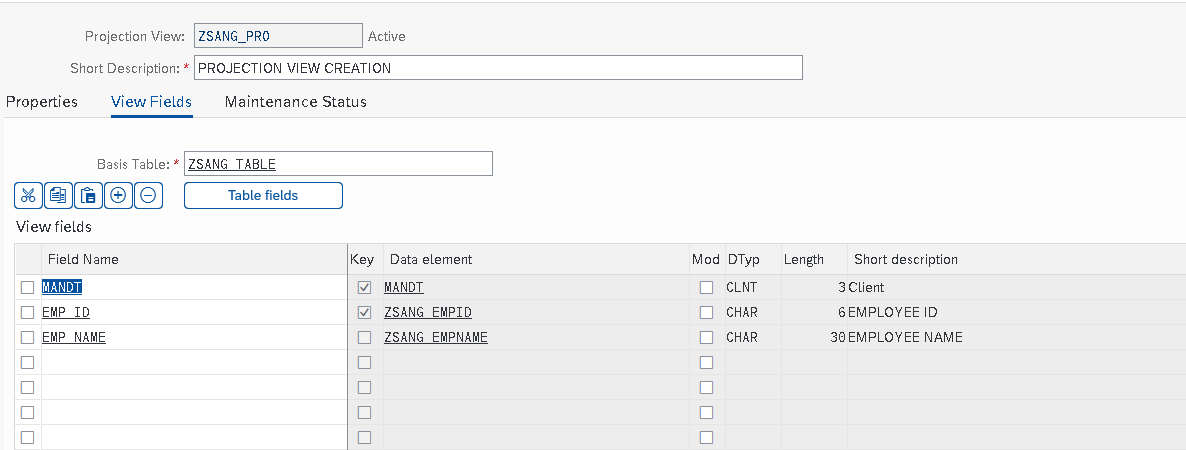


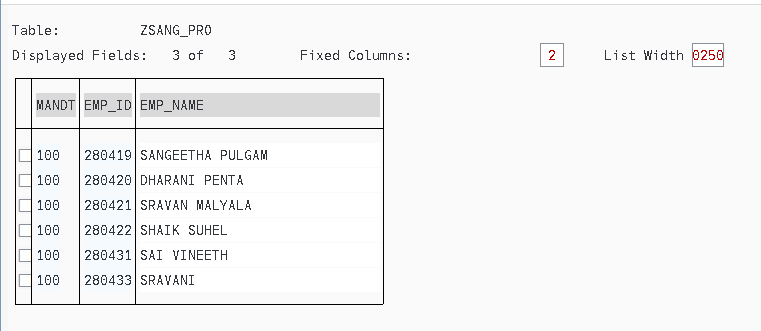




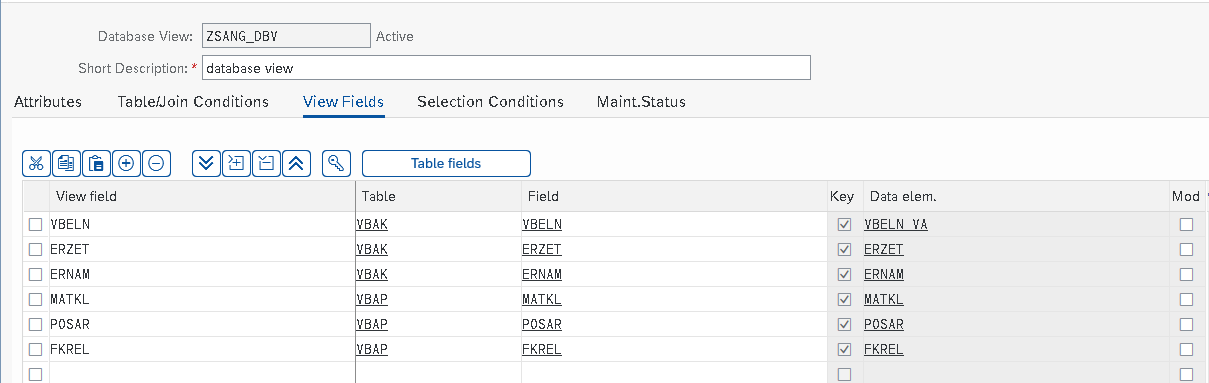
VIEWS:

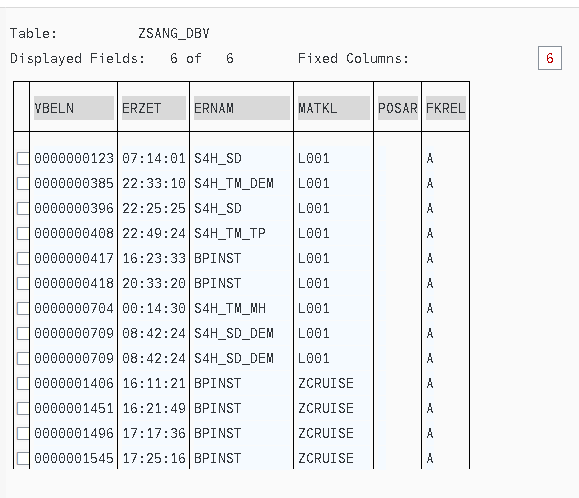
Projection view





DB VIEW





types: begin of ls\_tab,  
  VBELN type vbeln\_va,  
  erzet type erzet,  
  ernam type ernam,  
  matkl type matkl,  
  posar type posar,  
  fkrel type fkrel,  
  end of ls\_tab.  
data : lt\_tab type STANDARD TABLE OF ls\_tab,  
      wa\_tab type ls\_tab.  
select vbeln erzet ernam matkl posar fkrel  
  into table lt\_tab  
  from zsang\_dbv.  
loop at lt\_tab into wa\_tab.  
  write: / wa\_tab-vbeln,  
  wa\_tab-erzet,  
  wa\_tab-ernam,  
  wa\_tab-matkl,  
  wa\_tab-posar,  
  wa\_tab-fkrel.  
  ENDLOOP.

