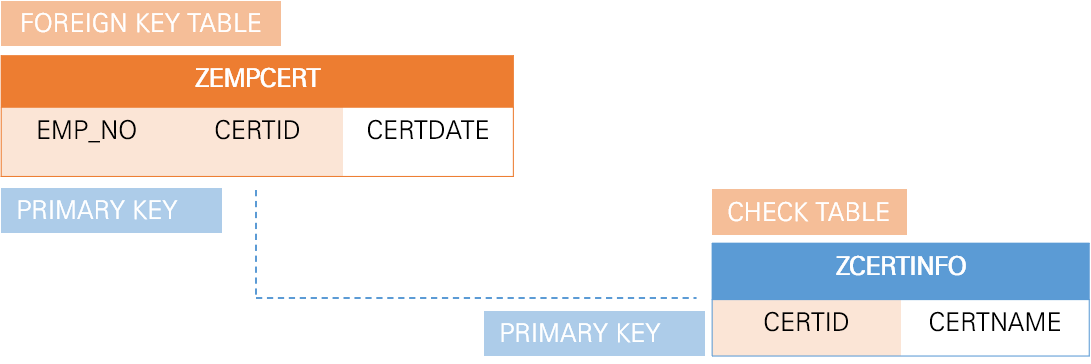
0701 | ABAP

FOREIGN KEY |

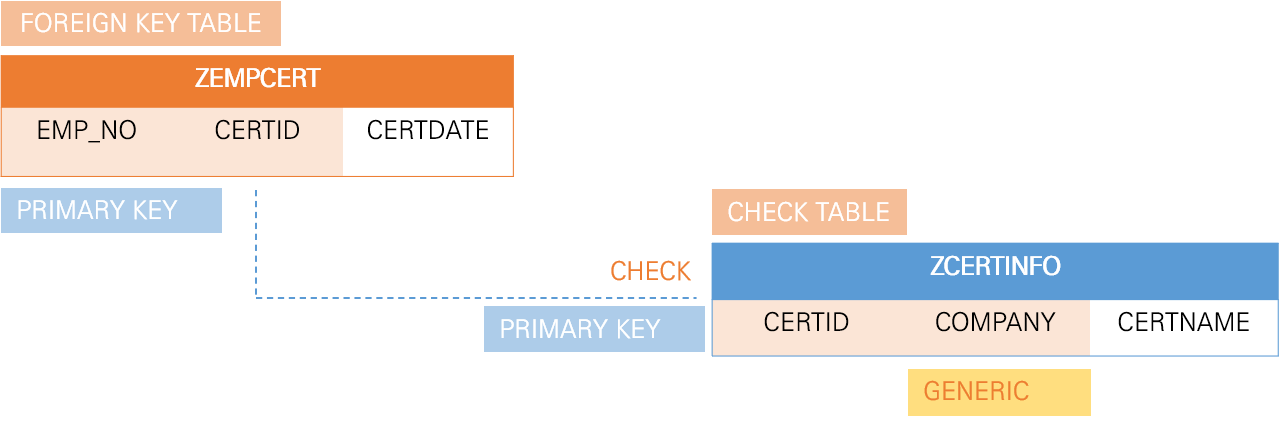


FOREIGN KEY는 관계형 데이터베이스에서 테이블 간의 연결을 정의하며, DATA INTEGRITY | 데이터 무결성 및 DATA CONSISTENCY | 일관성 을 보장하기 위해 사용된다. 이는 DATA DOMAIN에서 지정된 데이터의 유형을 벗어난 값의 입력을 배제하고 정확한 DATA를 입력할 수 있도록 도와주는 기능을 한다. FOREIGN KEY 는 ABAP DICTIONARY 에만 정의되어 있으며, 실제 데이터베이스에는 존재하지 않기 때문에 직접 테이블에 데이터를 추가하는 작업은 금지되어야 한다. FOREIGN KEY는 두 테이블 사이의 관계를 나타내는데 이때 FOREIGN KEY 테이블의 필드는 CHECK TABLE의 PRIMARY KEY 값을 참조한다. FOREIGN TABLE은 DEPENDENT TABLE이라 하며, CHECK TABLE은 REFERANCED TABLE이라 한다. FOREIGN KEY로 불리는 테이블의 필드는 CHECK TABLE의 PRIMARY KEY를 의미하며 FOREIGN KEY로 연결된 필드는 두 테이블 간 필드명이 달라도 같은 DATA TYPE과 길이를 가지며, 같은 DOMAIN을 사용한 필드도 가능하다.

FOREIGN KEY | CHECK FIELD와 VALUE CHECK

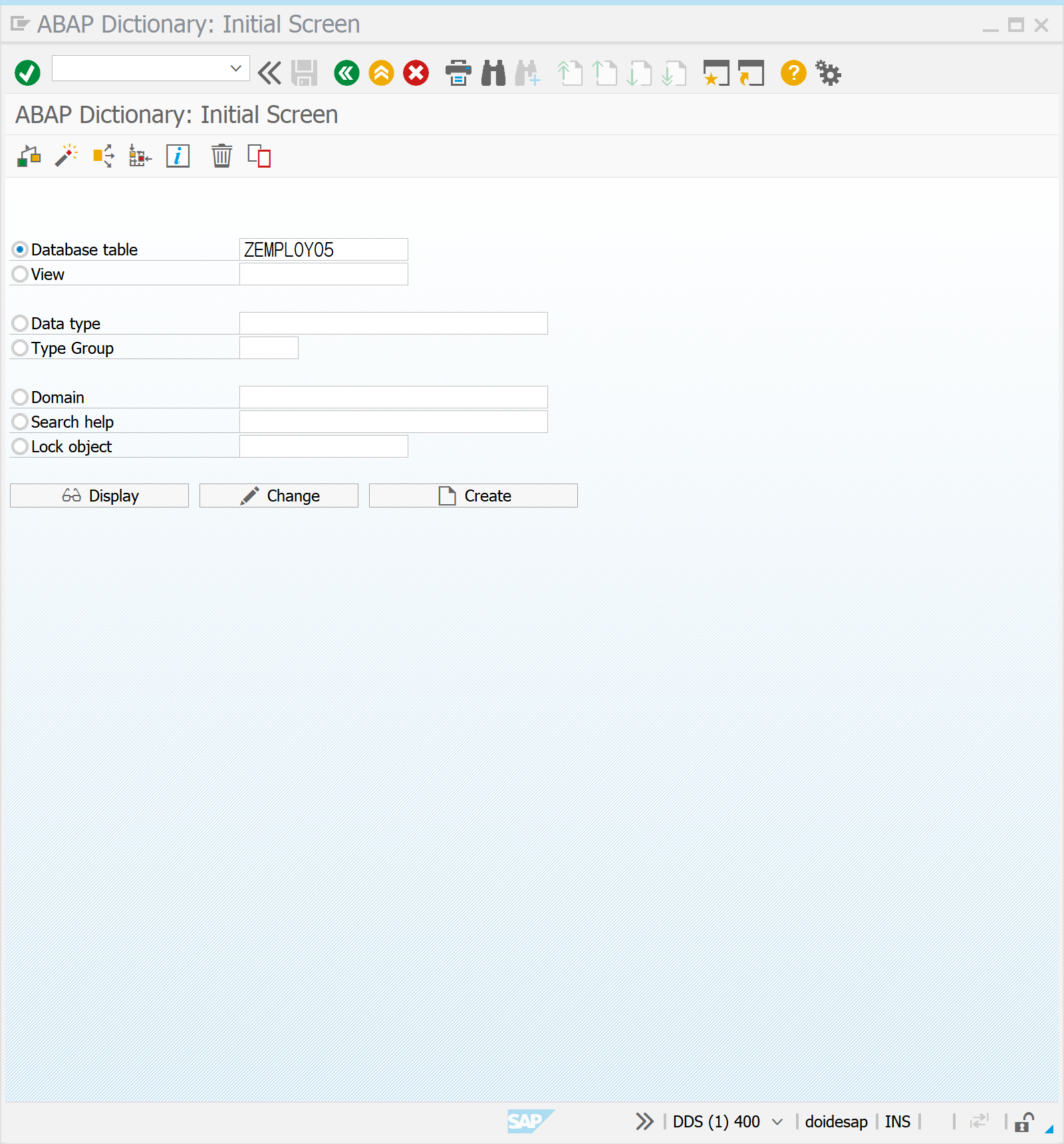
FOREGIN KEY가 CHECK FIELD와 연결되어 있다는 것은 FOREIGN KEY 필드에 입력되는 값은 CHECK TABLE에 값이 있는지 확인 과정을 거치게 되는 것을 의미한다. CHECK TABLE에 값이 없다면 FOREIGN KEY 필드에 값이 입력되지 않으며, 에러가 발생한다. FOREIGN KEY를 정의할 때 내부적으로 SELECT 문이 자동 생성되며, 조건문에 적합한 레코드가 CHECK TABLE에 있다면 해당 DATA는 유효한 DATA이다. FOREIGN KEY를 정의할 때 자동 생성되는 SELECT 문은 SELECT \* FROM CHECKTAB WHERE FOREIGTAB-ID = CHECKTAB-ID.

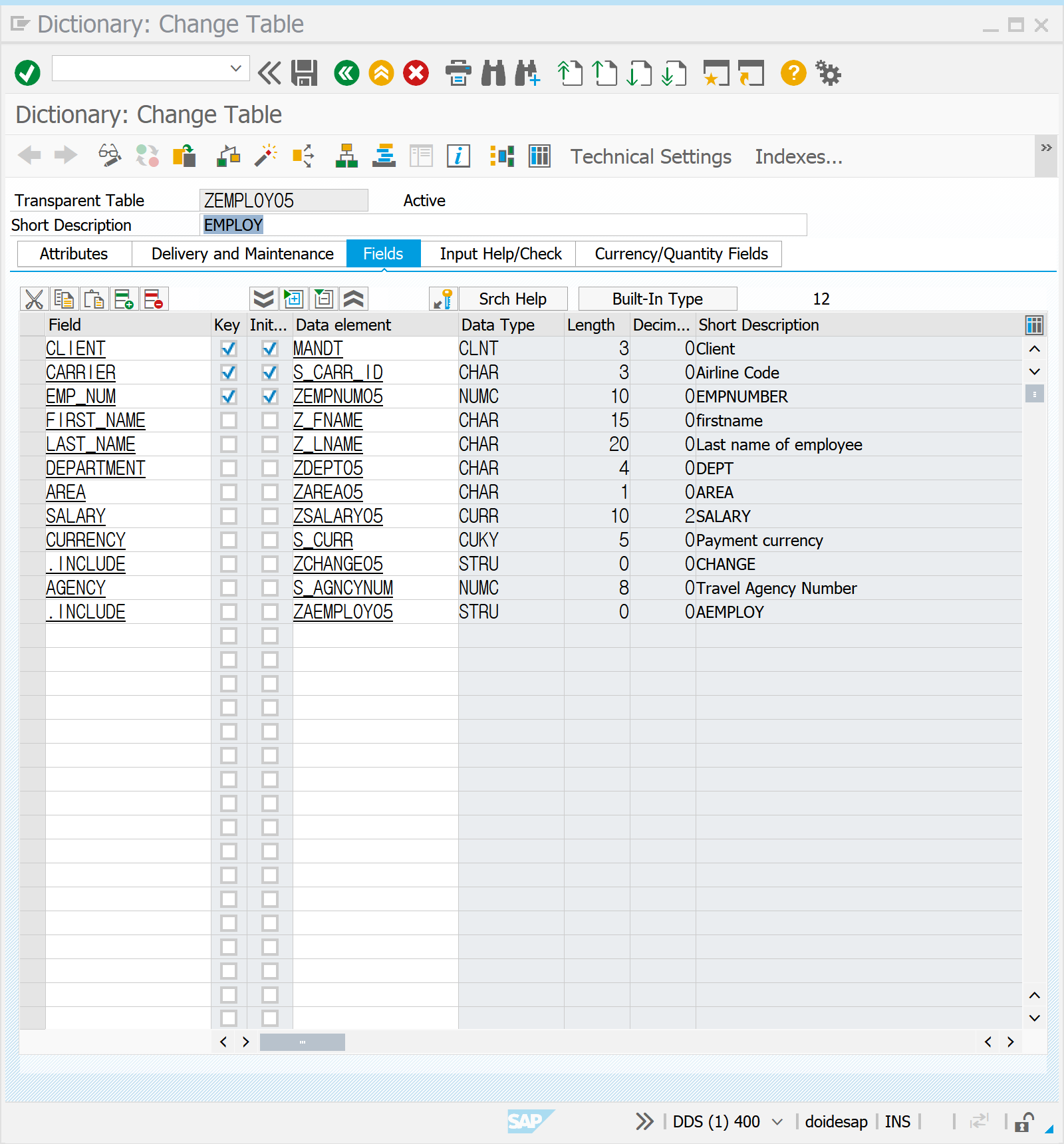
FOREIGN KEY | GENERIC 과 CONSTANT FOREIGN KEY



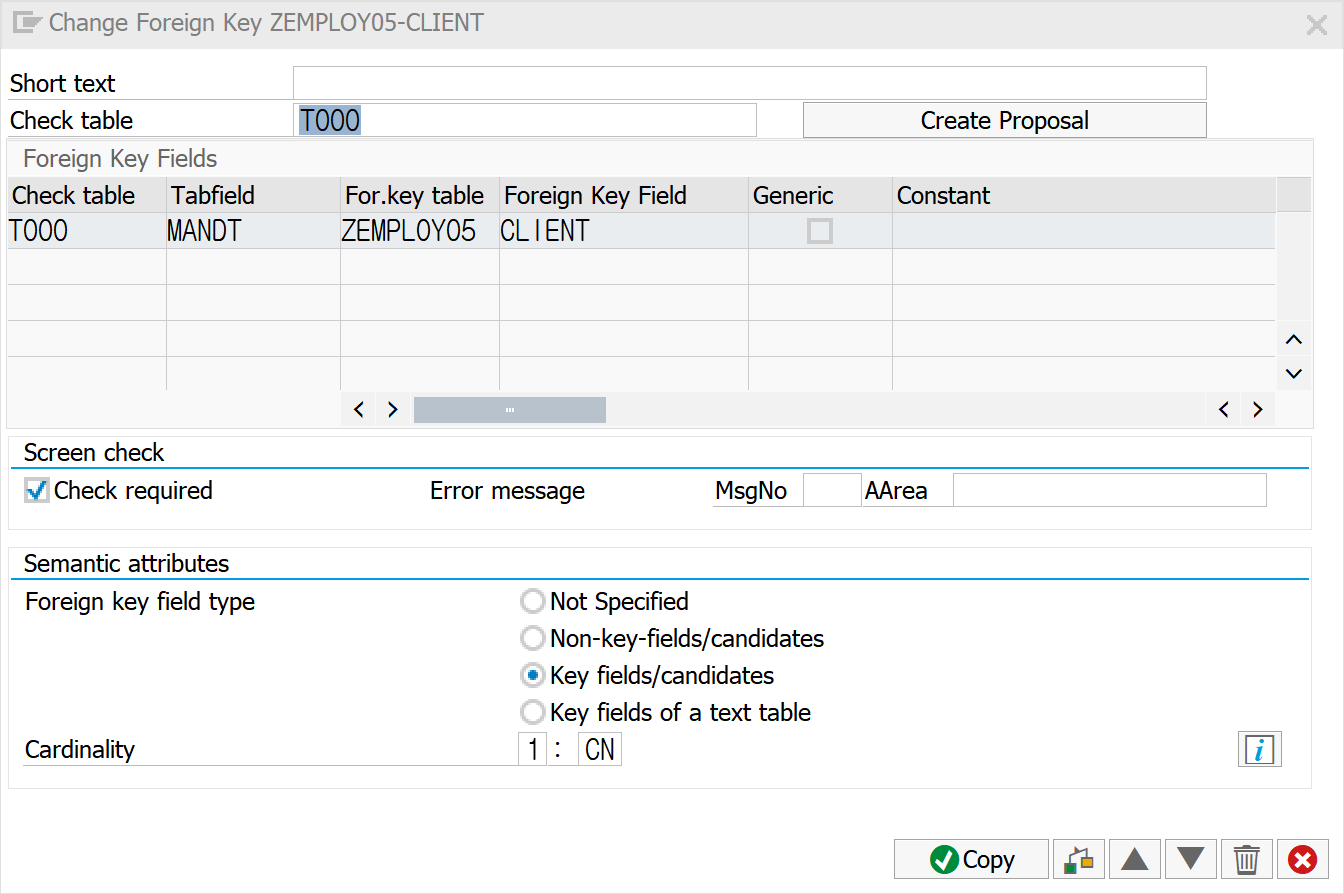
CHECK TABLE의 모든 KEY FIELD를 FOREGIN KEY로 CHECK 할 필요는 없다. 이 경우에는 FOREGIN KEY로 연결할 필요 없이 GENERIC FOREGIN KEY 또는 CONSTANT FOREGIN KEY로 정의하게 된다. CHECK TABLE에 여러 개의 KEY 필드가 있을 경우 그중 CHECK 할 필요가 없는 필드에 GENERIC KEY를 설정하면 CHECK TABLE에서 필드의 값을 무시하고 체크하며 이는 SELECT 문이 수행될 때 조건절에서 GENERIC 필드의 조건을 삭제한다. CONSTANT FOREIGN KEY는 FOREIGN KEY를 위해 상수 값을 지정하여 사용할 수 있다. 해당 필드의 값이 상수 값과 일치하면 FOREIGN KEY에 값이 입력되며, 일치하지 않으면 입력되지 않는다.

SE11 | ZEMPLOY05 | FOREIGN KEY 지정하기

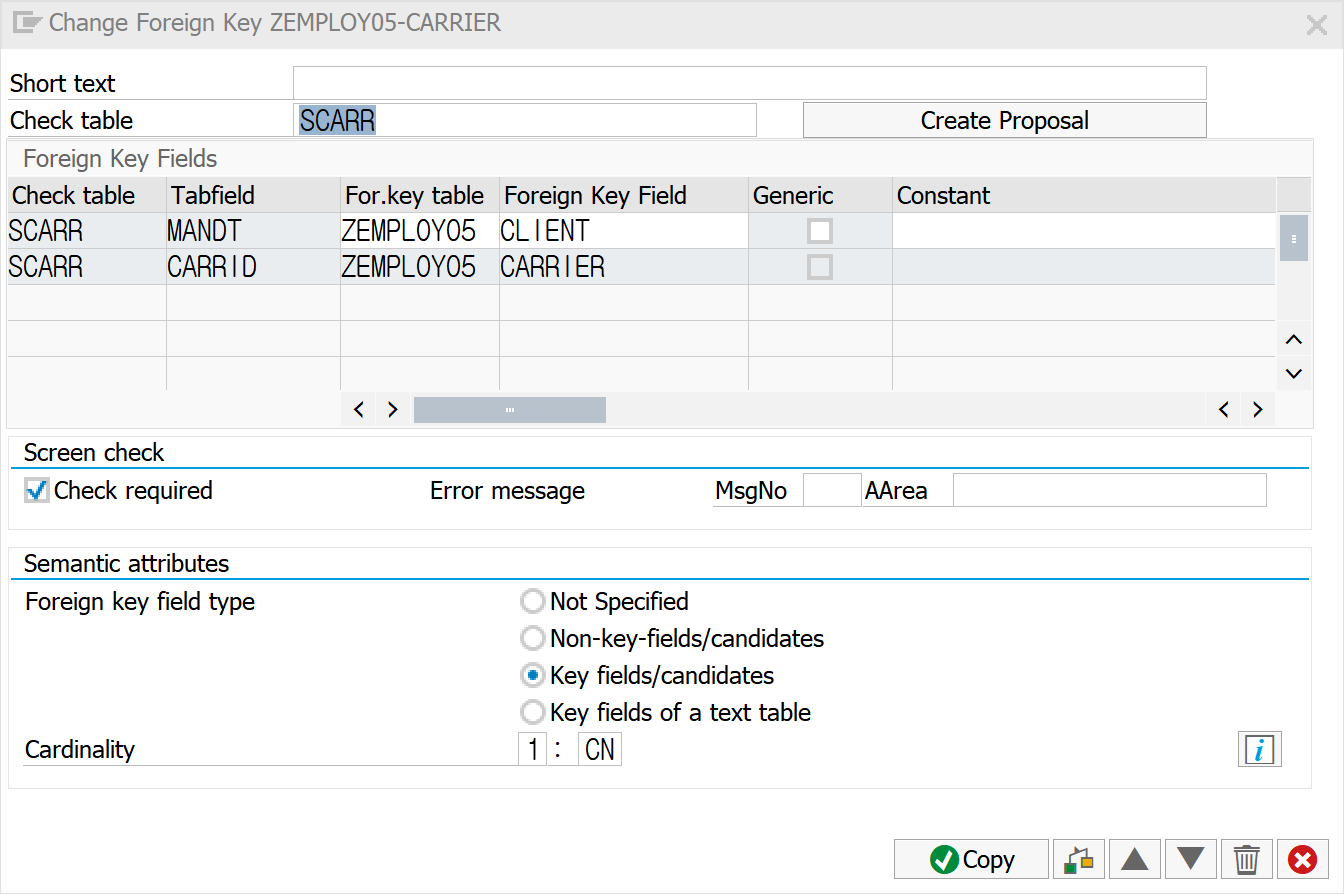




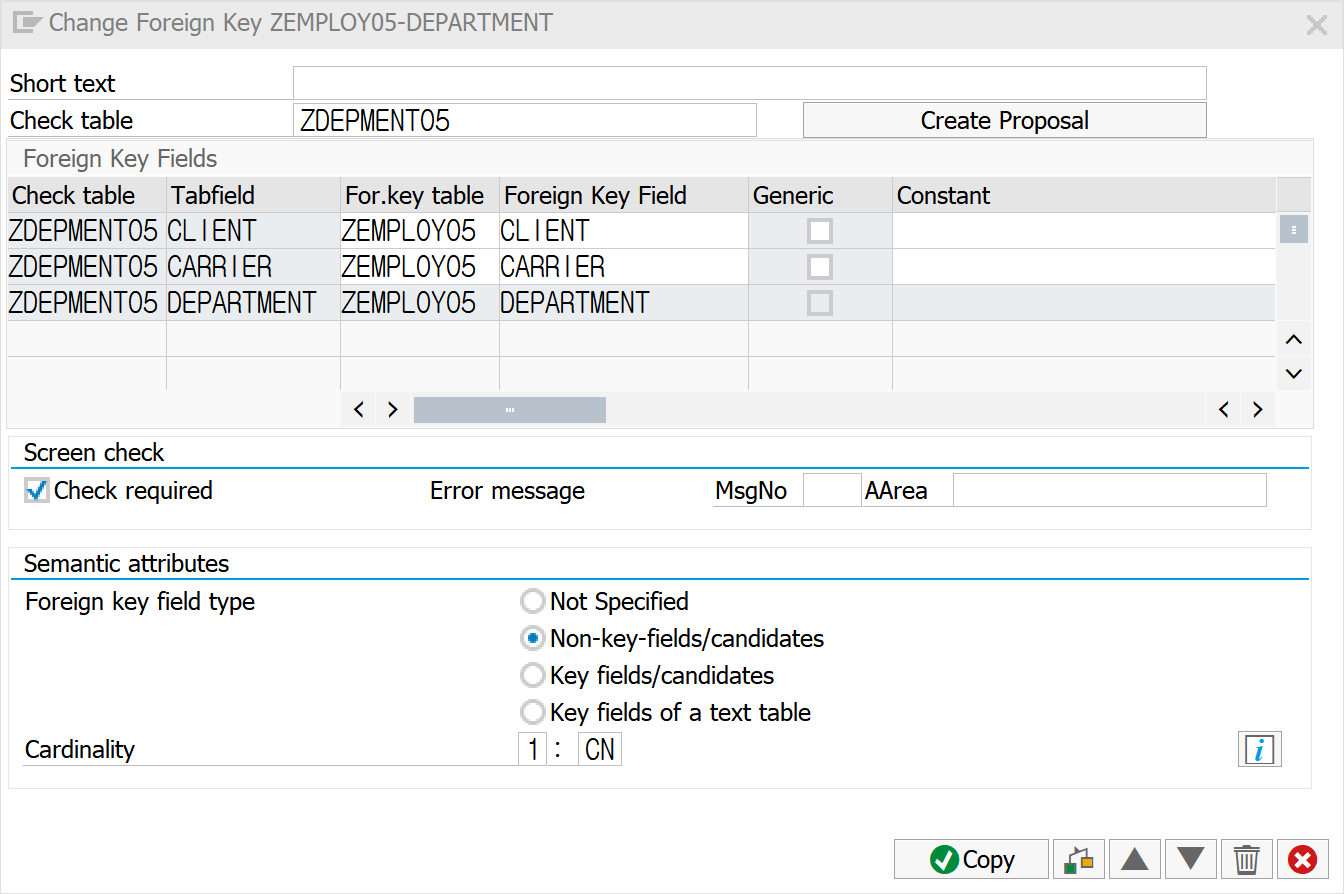
SE11 | ZEMPLOY05 | FOREIGN KEY 지정하기 | CLIENT



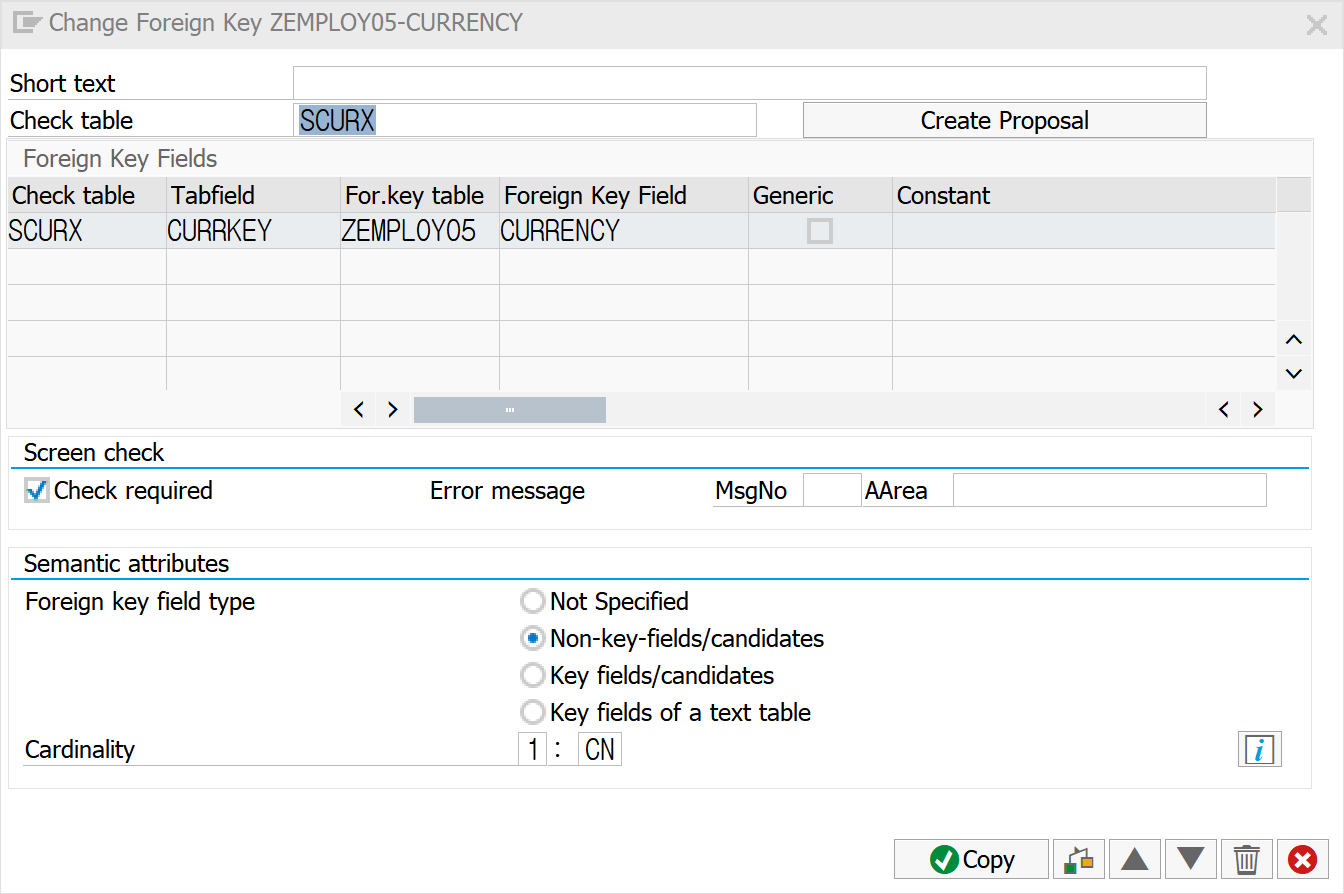
SE11 | ZEMPLOY05 | FOREIGN KEY 지정하기 | CARRIER



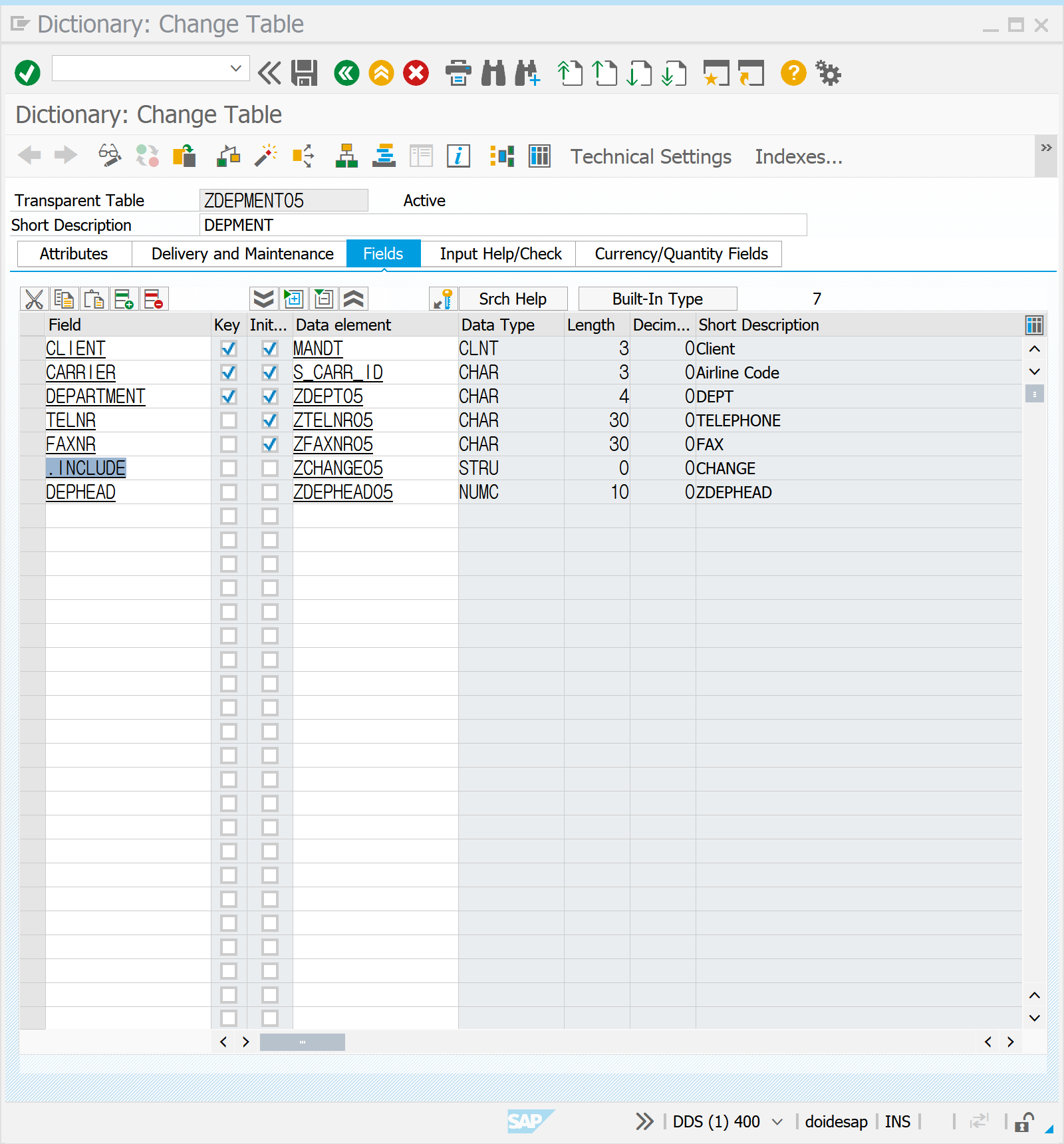
SE11 | ZEMPLOY05 | FOREIGN KEY 지정하기 | DEPARTMENT



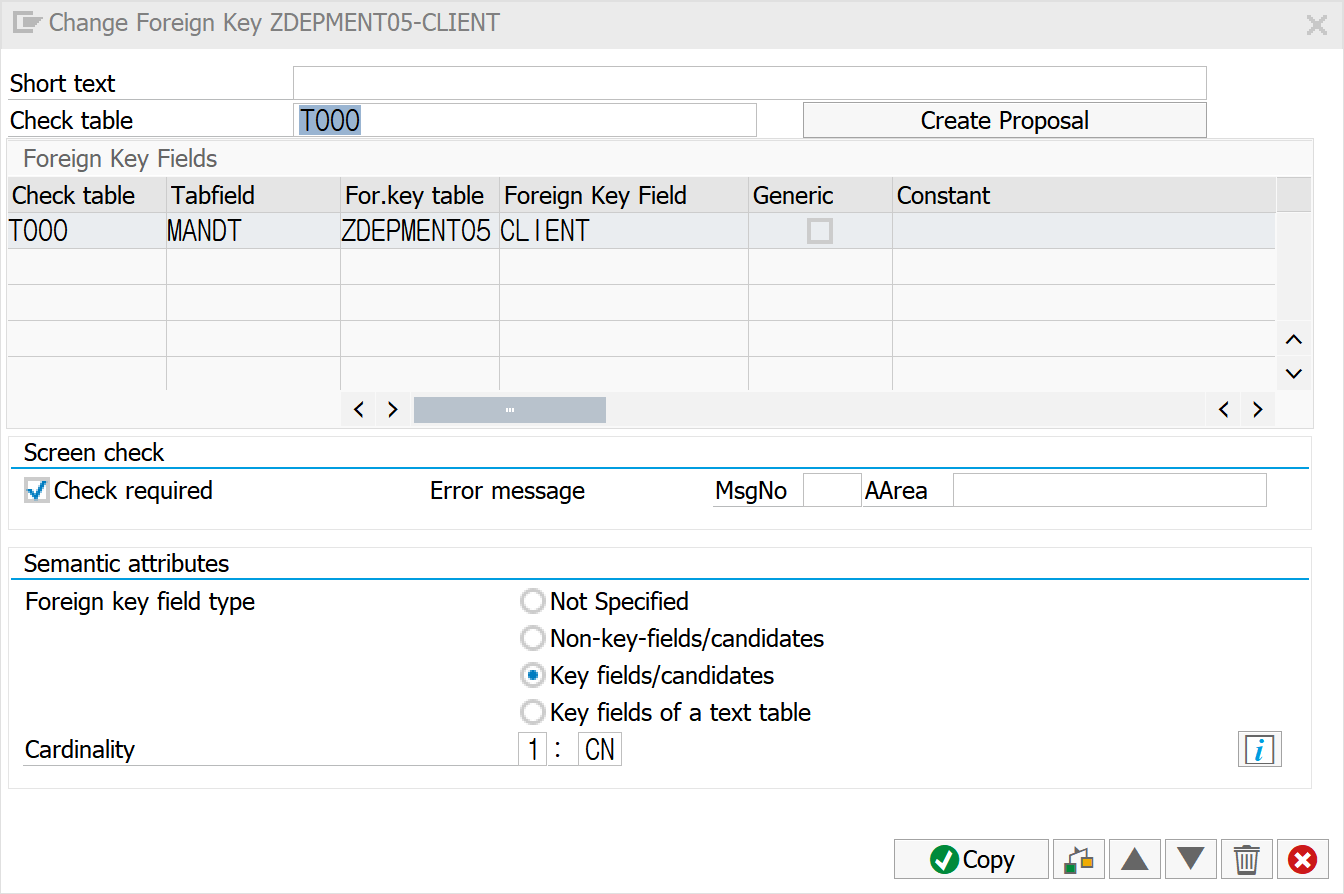
SE11 | ZEMPLOY05 | FOREIGN KEY 지정하기 | CURRENCY



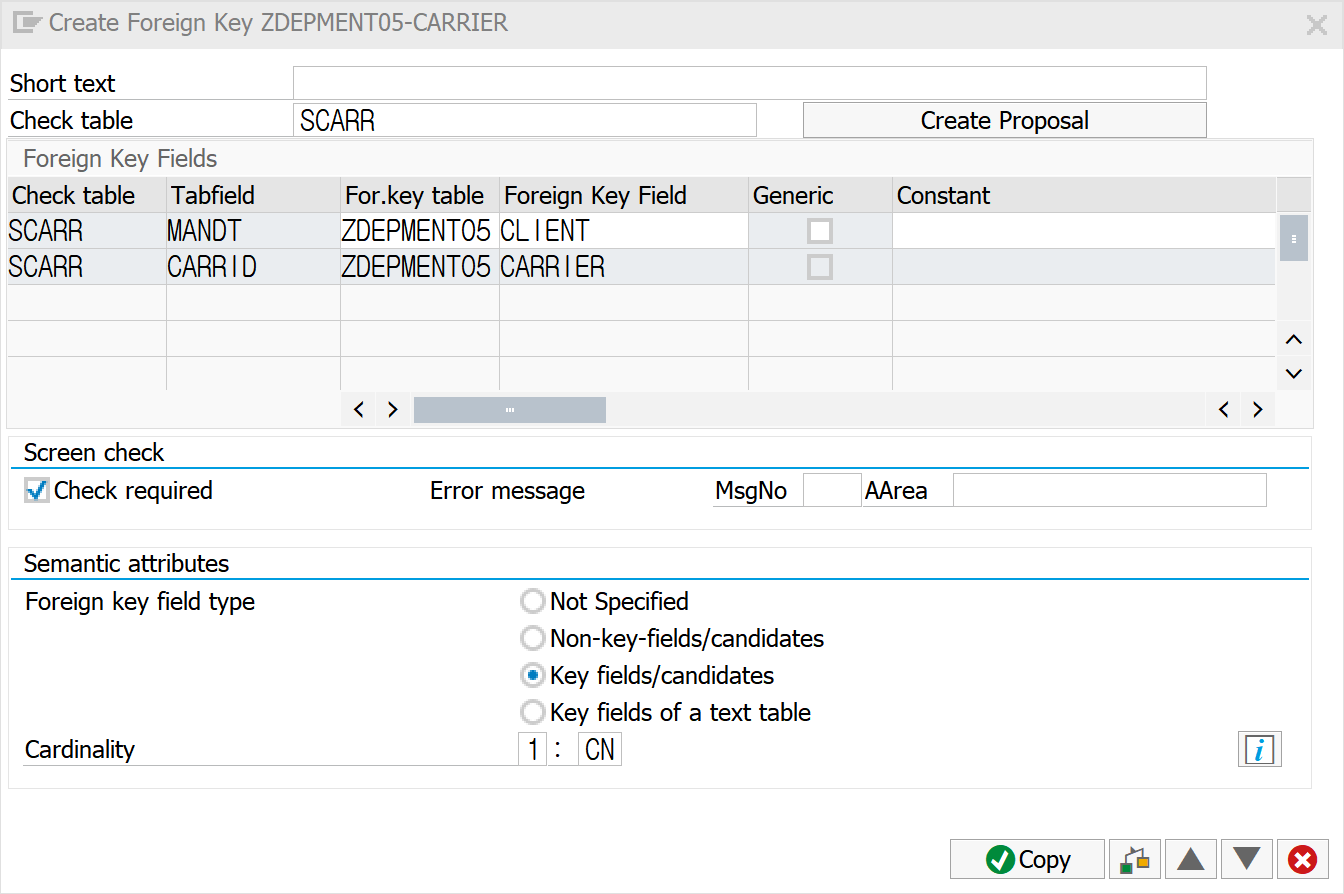
SE11 | ZDEPMENT05 | FOREIGN KEY 지정하기 |



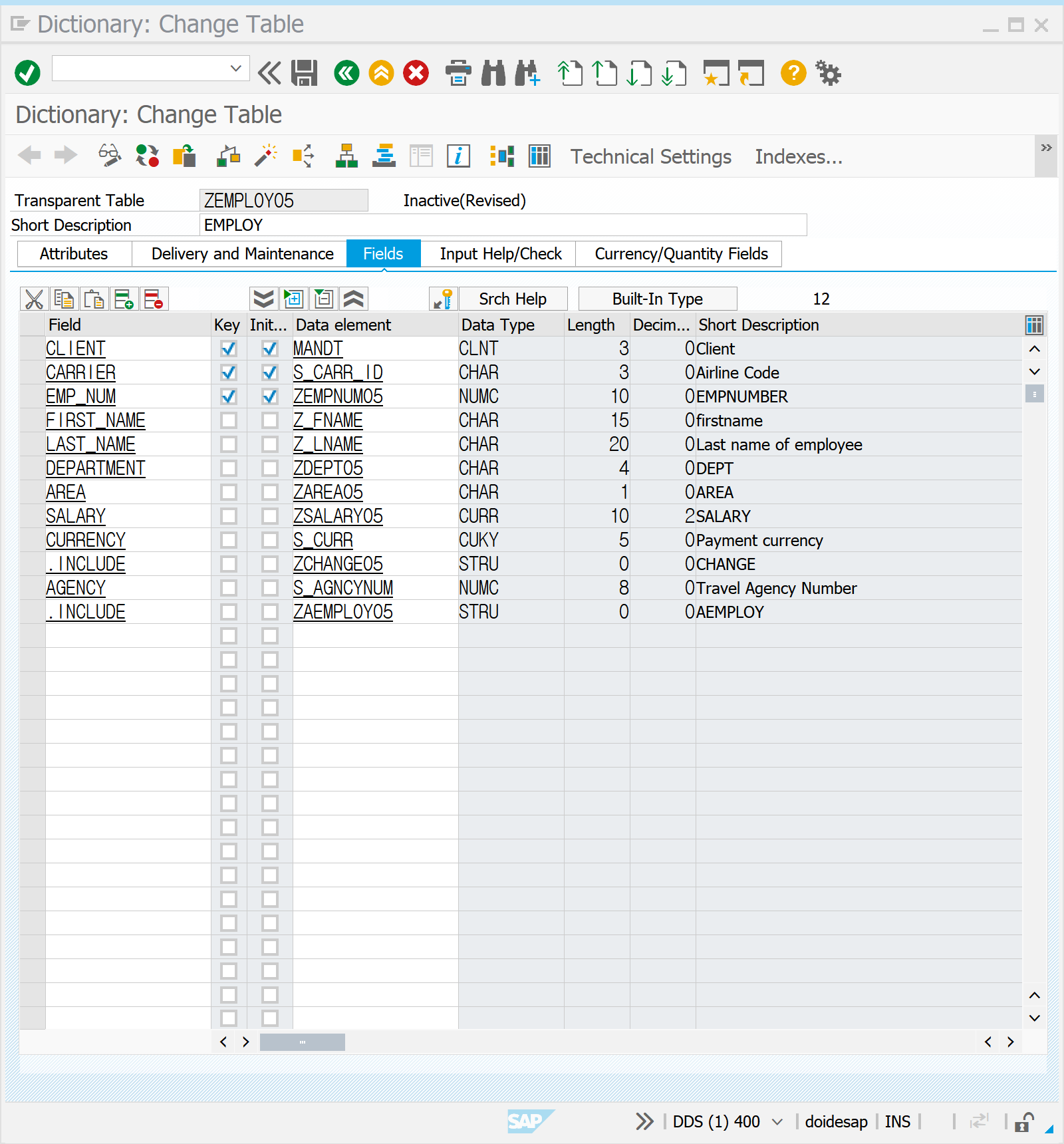
SE11 | ZDEPMENT05 | FOREIGN KEY 지정하기 | CLIENT



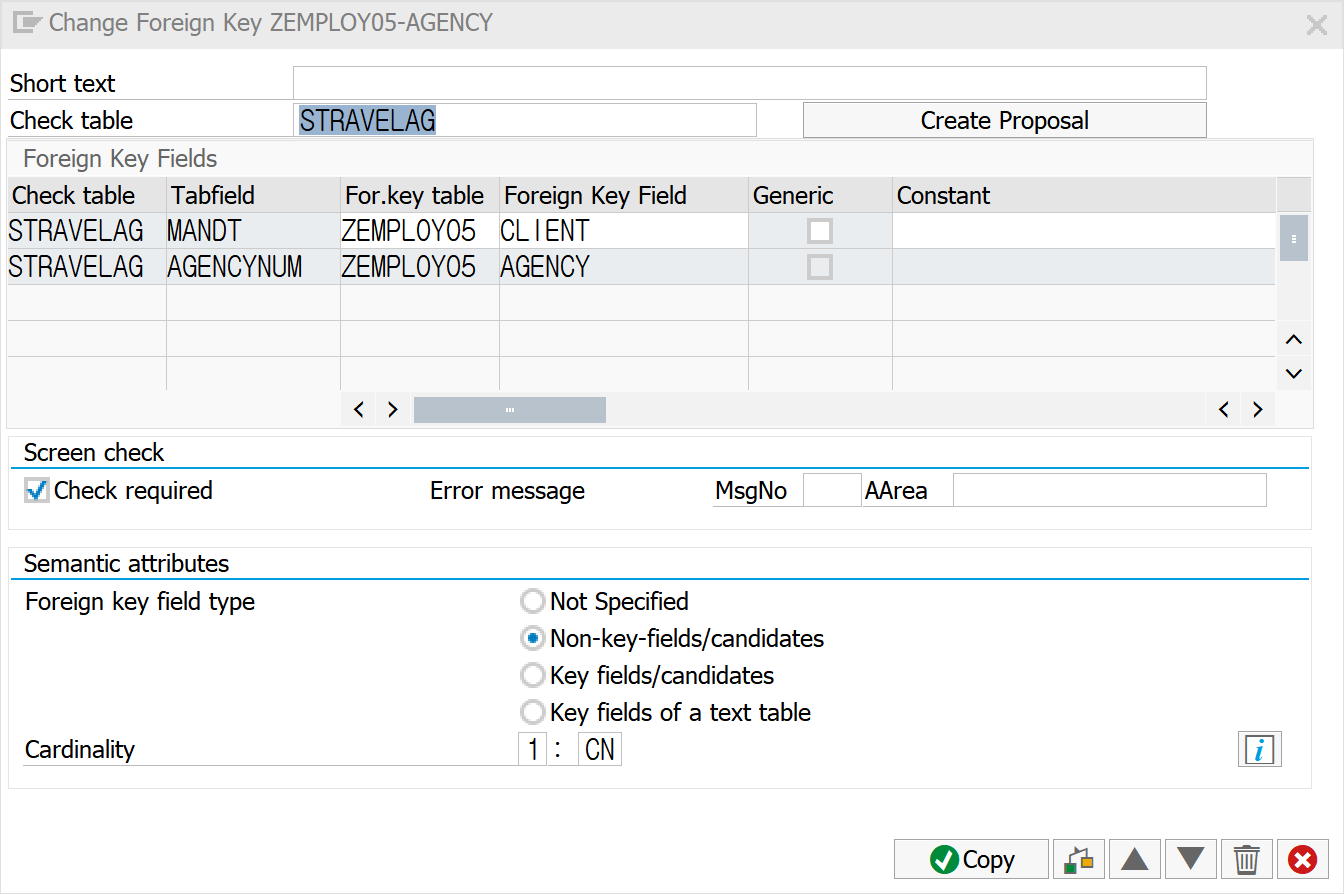
SE11 | ZDEPMENT05 | FOREIGN KEY 지정하기 | CARRIER



SE11 | ZEMPLOY05 | FOREIGN KEY 지정하기 | AGENCY



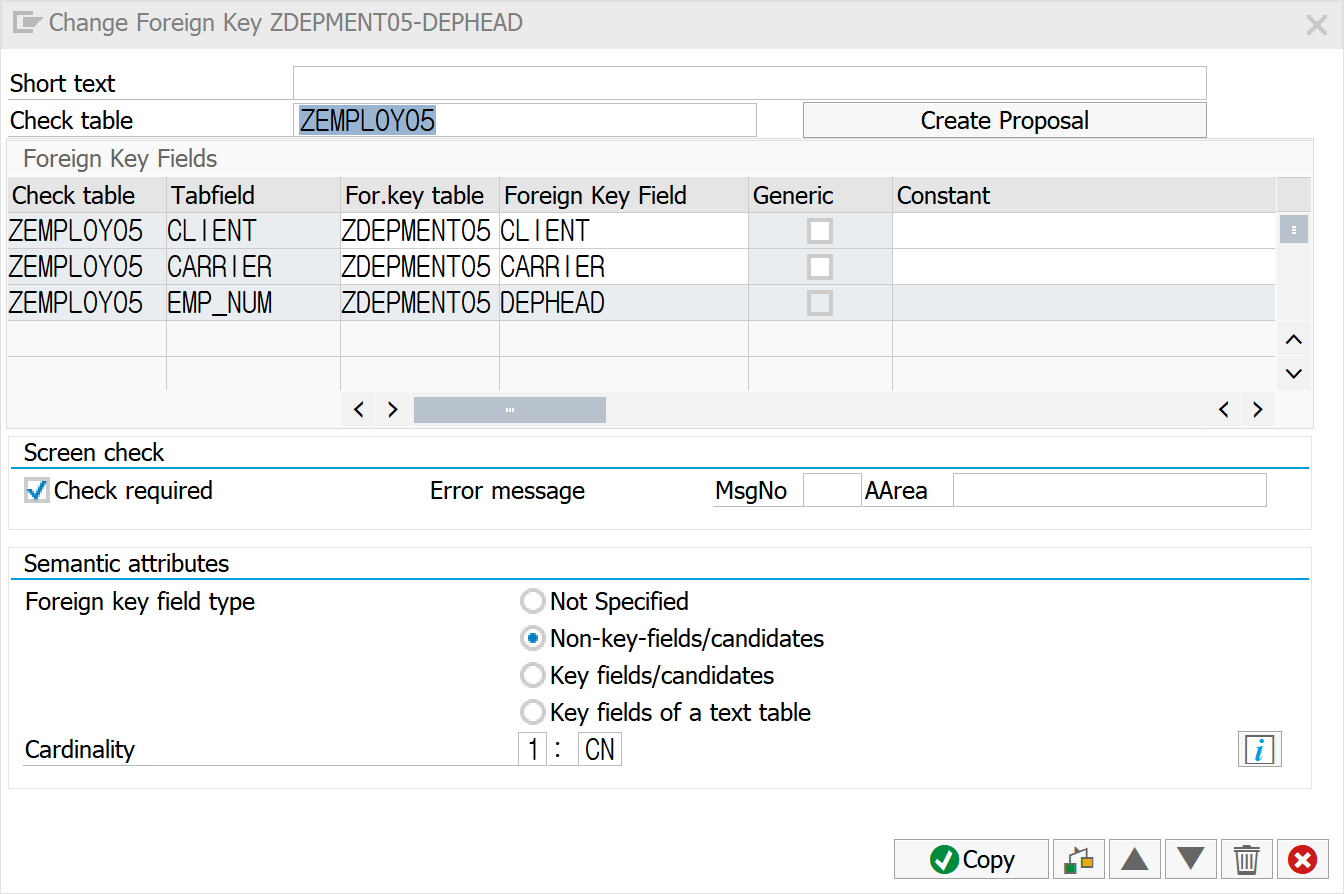
SE11 | ZEMPLOY05 | FOREIGN KEY 지정하기 | AGENCY



SE11 | ZDEPMENT05 | FOREIGN KEY 지정하기 | DEPHEAD



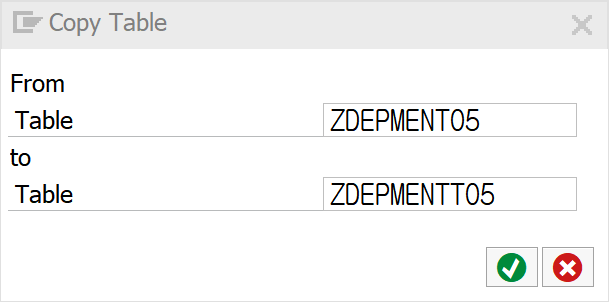
SE11 | ZDEPMENT05 | FOREIGN KEY 지정하기 | DEPHEAD



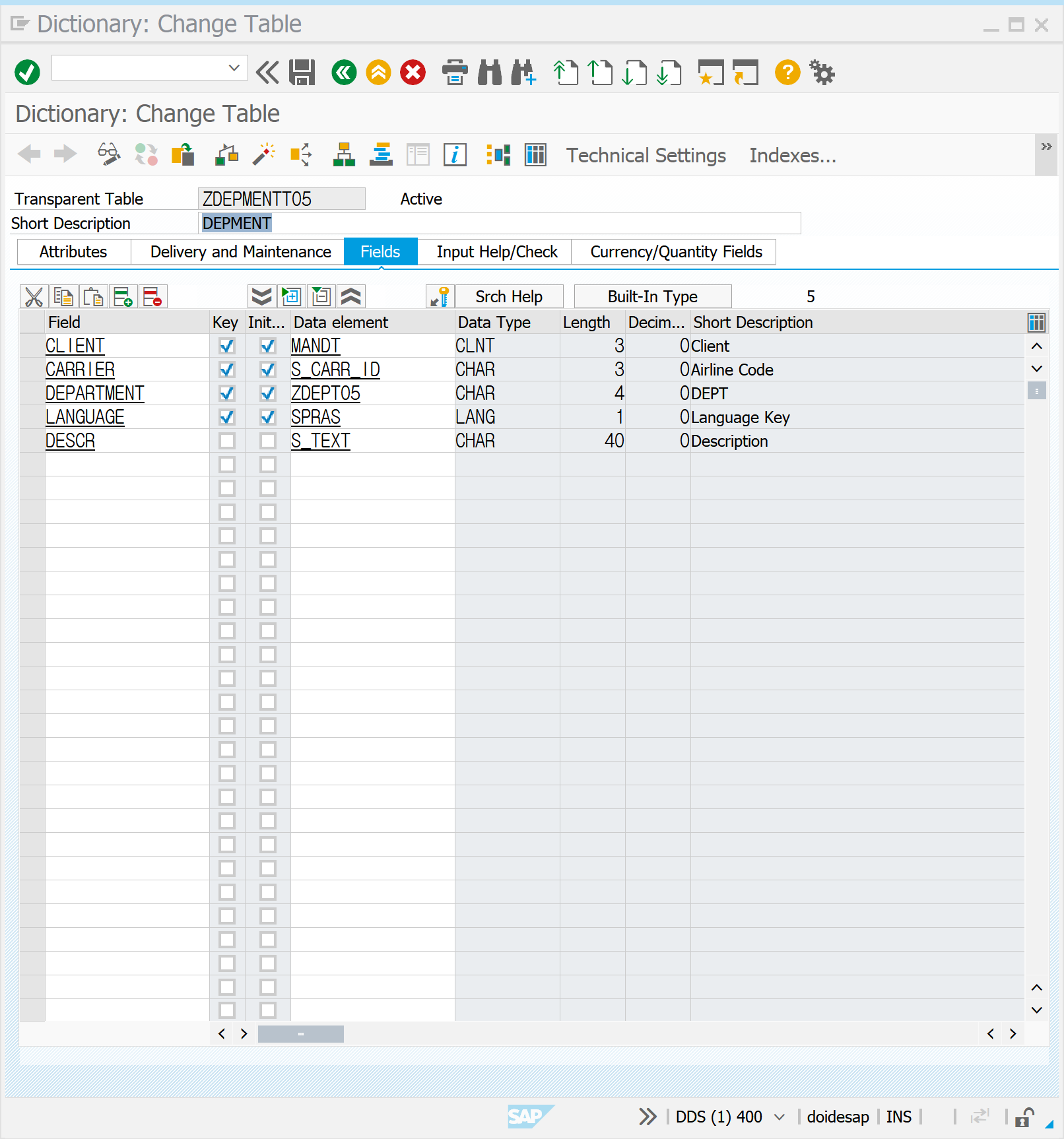
TEXT TABLE |

TABLE은 다른 연관된 테이블로부터 각각의 언어키를 가지고 해당 언어로 SUBSCROPTION을 작성할 수 있다. 이는 동일한 필드가 다른 언어로 작성될 수 있음을 의미한다.

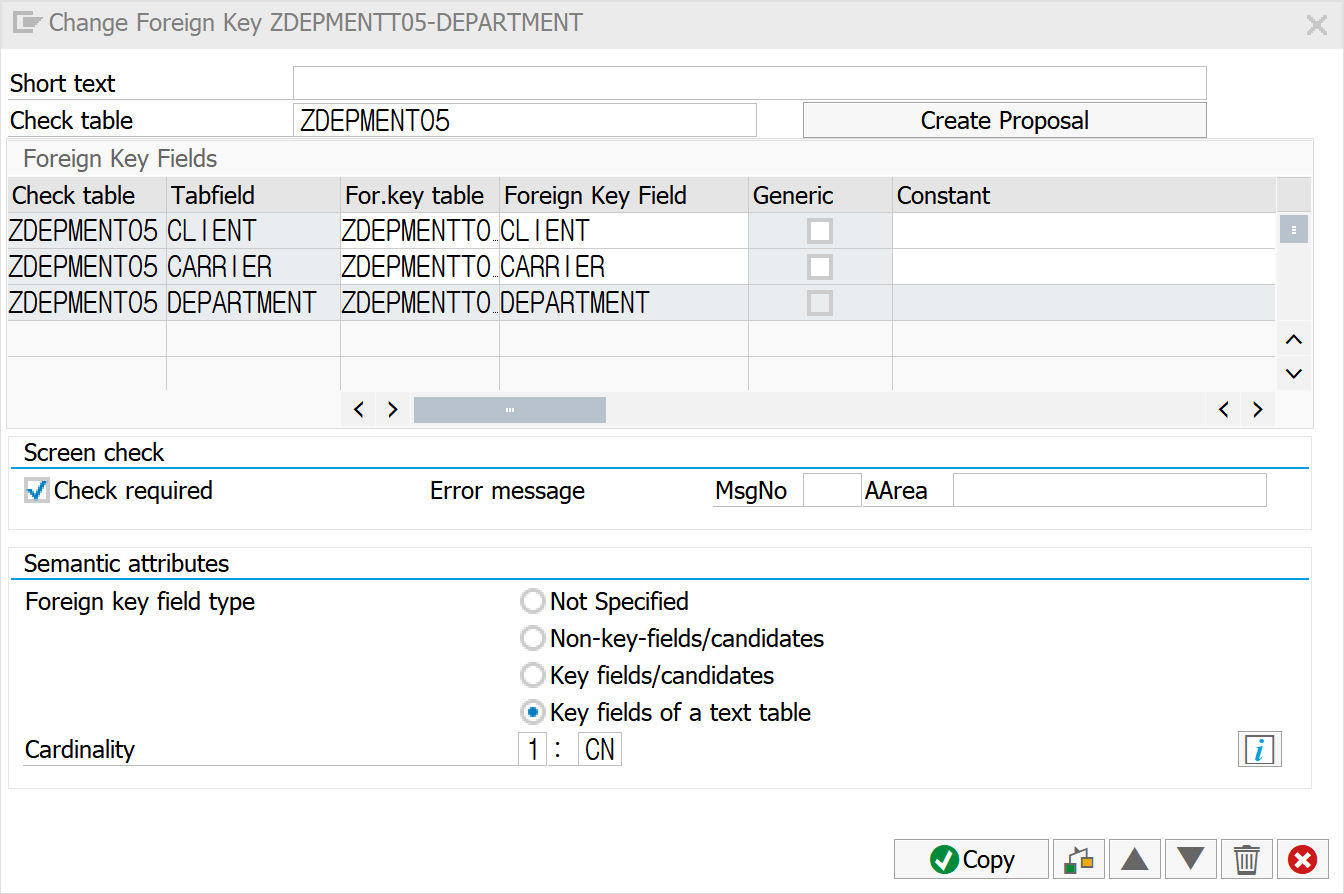
SE11 | ZDEPMENTT05 | COPY ZDEPMENT05



SE11 | ZDEPMENTT05 | LANGUAGE & DESCR



SE11 | ZDEPMENTT05 | FOREIGN KEY 지정하기 | DEPARTMENT



SE11 | ZDEPMENTT05 | FOREIGN KEY 지정하기 | LANGUAGE

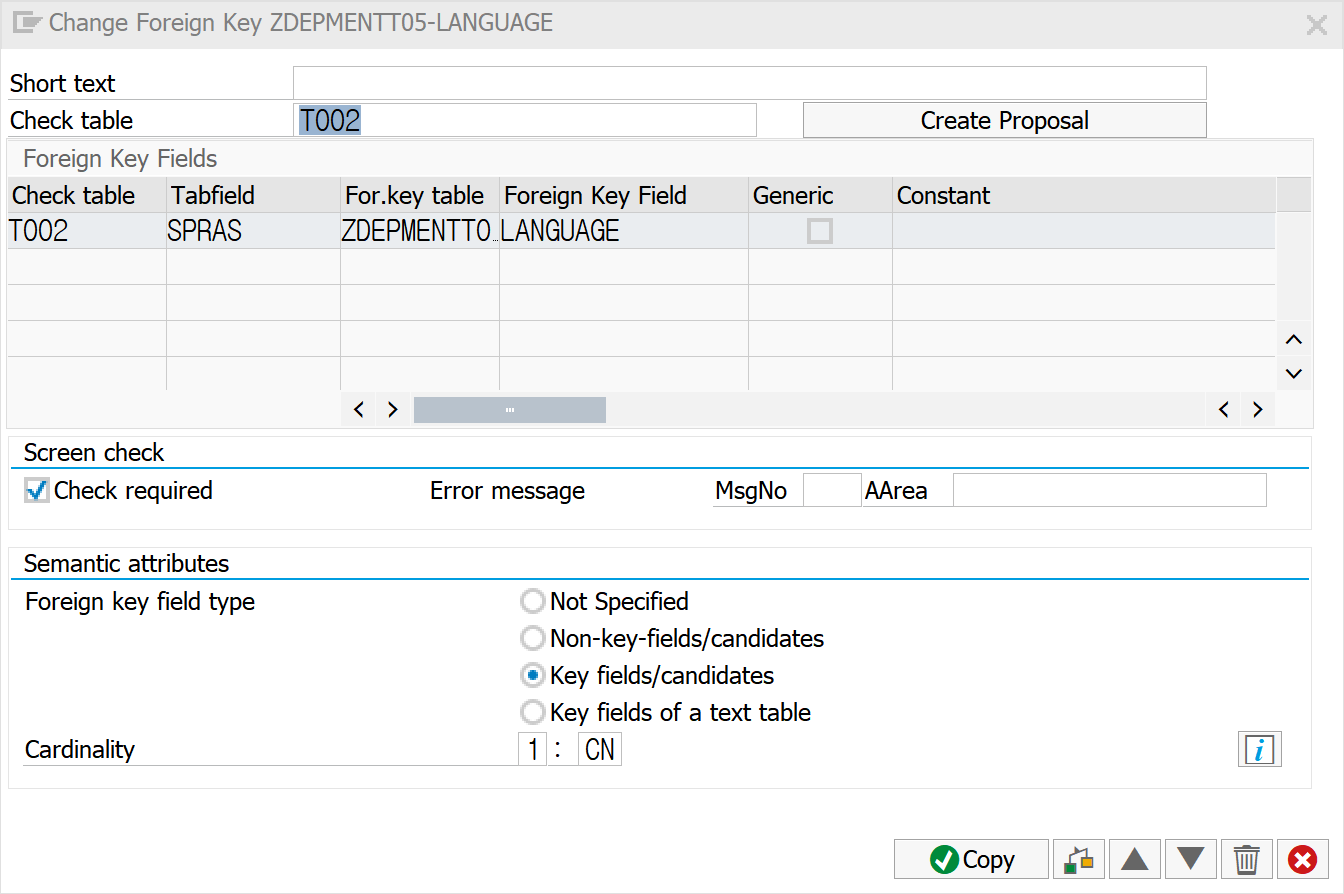


TABLE ENHANCEMENT | INCLUDE STRUCTURE | EDIT | INCLUDE

STRUCTURE 전체를 TABLE 또는 STRUCTURE에 추가하거나 개별 필드와 INCLUDE를 조합하여 TABLE을 생성할 수 있다. 여러 개의 TABLE에 같은 구조를 가지는 필드들이 사용될 경우 개별 테이블마다 필드를 생성 | 변경하는 것은 시간이 많이 들어가기 때문에 공통으로 사용되는 필드들을 STRUCTURE로 생성하고 TABLE에서 INCLUE 하여 사용한다. INCLUDE는 9단계 DEPT 까지 사용할 수 있으며 TABLE은 구조체 및 TABLE을 포함할 수 있고, STRUCTURE 필드에는 STRUCTURE 및 TABLE을 입력할 수 있다.

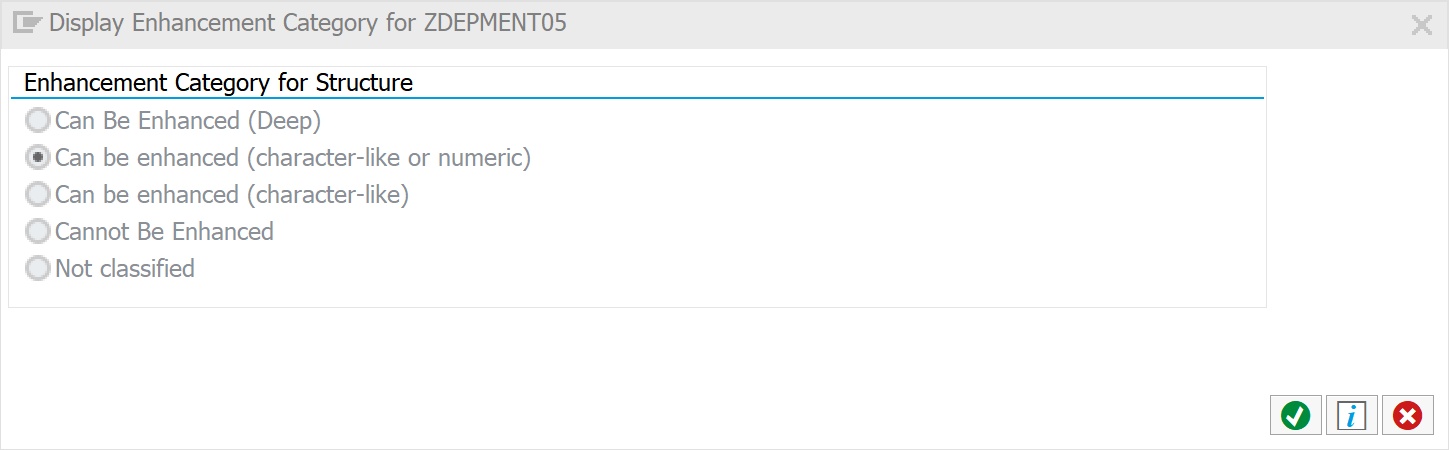
TABLE ENHANCEMENT | APPEND STRUCTURE | GOTO | APPEND STR-

APPEND STRUCTURE 기능은 하나의 테이블 또는 STRUCTURE 에만 할당할 수 있는 STRUCTURE 로 테이블 자체를 수정하지 않고 필드를 추가하게 한다. 이는 SAP 입장에서 CUSTOMER FIELD 라고 한다. 하나의 APPEND STRUCTURE 은 하나의 테이블에서만 사용될 수 있고 개별 TABLE은 여러 개의 APPEND STRUCTURE을 사용할 수 있다. POOLED TABLE과 CLUSTER TABLE은 APPEND STRUCTURE를 사용할 수 없으며 LCHR 나 LRAW과 같이 LONG TYPE의 필드가 포함된 테이블에도 사용할 수 없다. APPEND STRUCTURE을 테이블에 추가하여 활성화하면, DATABASE에 APPEND STRUCTURE의 모든 필드를 포함하는 TABLE이 생성된다.

|  |
| --- |
| APPEND STRUCTURE의 수행 |

APPEND STRUCTURE은 다음과 같은 역할을 수행한다. STANDARD 또는 CBO TABLE에 신규 FIELD를 추가한다. 이미 존재하는 TABLE에 FOREIGN KEY를 추가 및 정의하며 필드의 탐색 도움말을 추가한다.

TABLE ENHANCEMENT CATEGORY | EXTRAS | ENHANCEMENT CATE-



SE11 에서 TABLE 생성 후 활성화하면 경고창이 조회된다. 경고창은 ENHANCEMENT CATEGORY FOR TABLE MISSING 내역을 조회할 수 있으며, 이는 TABLE APPEND, INCLUDE 기능을 사용하기 위한 유형을 지정하라는 것을 의미한다. ENHANCEMENT CATEGORY의 유형은 다음과 같다.

|  |  |
| --- | --- |
| CANNOT BE ENHANCED | APPEND INCLUDE 기능 사용 X |
| CAN BE ENHANCED  CHARATER-TYPE | APPEND INCLUDE 기능 사용 하나  CHARACTER 타입 필드로만 확장 |
| CAN BE ENHANCED  CHARACTER-TYPE OR NUMERIC | APPEND INCLUDE 기능 사용 하나  CHARACTER, 숫자 타입 사용 O |
| CAN BE ENHANCED  DEEP | ANY TYPE | APPEND INCLUDE 기능 사용  모든 DATA TYPE 사용 O |

VIEW |

관계형 DATABASE 에서는 여러 개의 테이블이 존재하며, 테이블에 데이터들이 분산된 경우가 있다. 이러한 데이터를 한 번에 조회하는 경우 VIEW를 사용한다. VIEW는 하나 또는 여러 테이블에 존재하는 데이터를 통합하여 조회할 수 있도록 지원한다. ABAP DICTIONARY에서 VIEW를 활성화하면 DATABASE 에 생성되고, DATABASE VIEW에 접근하기 위해 DATABASE INTERFACE를 통해 ACESS 한다. 필요한 TABLE과 FIELD만 조합하기 때문에 불필요한 필드를 조회하지 않아 INTERFACE를 최소화시키며 VIEW는 ABAP PROGRAM 에서 테이블과 같이 SELECT 구문에 사용할 수 있다.

VIEW | DATABASE VIEW |

DATABASE VIEW는 ABAP DICTIONARY에 정의되면 활성화 시 자동으로 DATABASE에 생성된다. ABAP DICTIONARY에서 DATABASE VIEW의 구조가 변경되면 변경 사항은 바로 DATABASE VIEW에 영향을 주지 않는다. VIEW는 데이터를 가진 것이 아니므로 기존의 VIEW를 삭제하고 ABAP DICTIONARY에 정의된 새로운 VIEW를 생성시켜야 한다. VIEW를 이용하여 데이터베이스의 자료를 액세스 하기 위해서는 OPEN SQL이나 NQTIVE SQL을 이용하는 두 가지 방법 모두를 사용할 수 있으며, OPEN SQL은 DATABASE INTERFACE에 의해 DATABASE에 의존적인 NATIVE SQL로 번역된다.

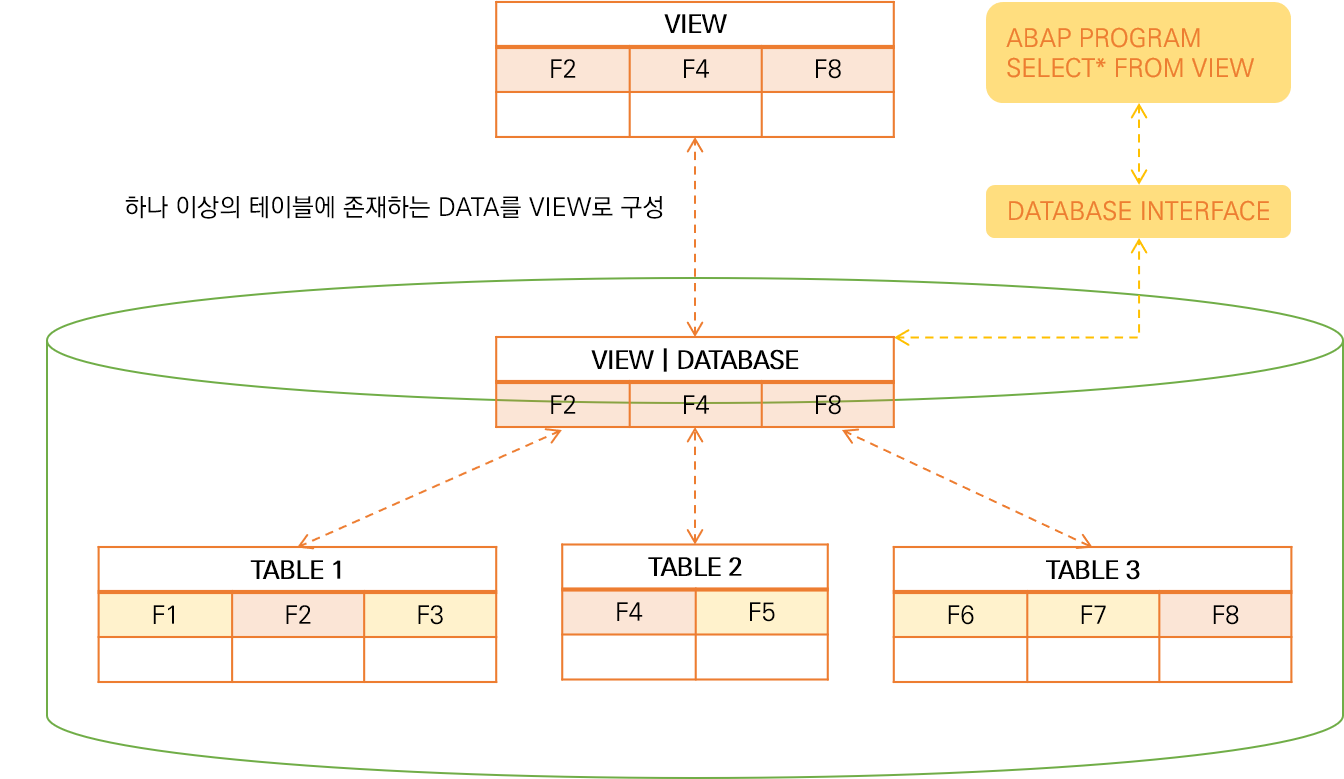
VIEW | MAINTANANCE VIEW |

MAINTANACE VIEW는 VIEW를 통해 읽기 속성만 부여할 것인지 또는 쓰기도 가능하게 할 것인지를 정의한다. 만약 DATABASE VIEW가 두 개 이상의 테이블로 구성되어 있다면 VIEW를 통해서는 읽기 작업만 할 수 있다.

VIEW 종류 |

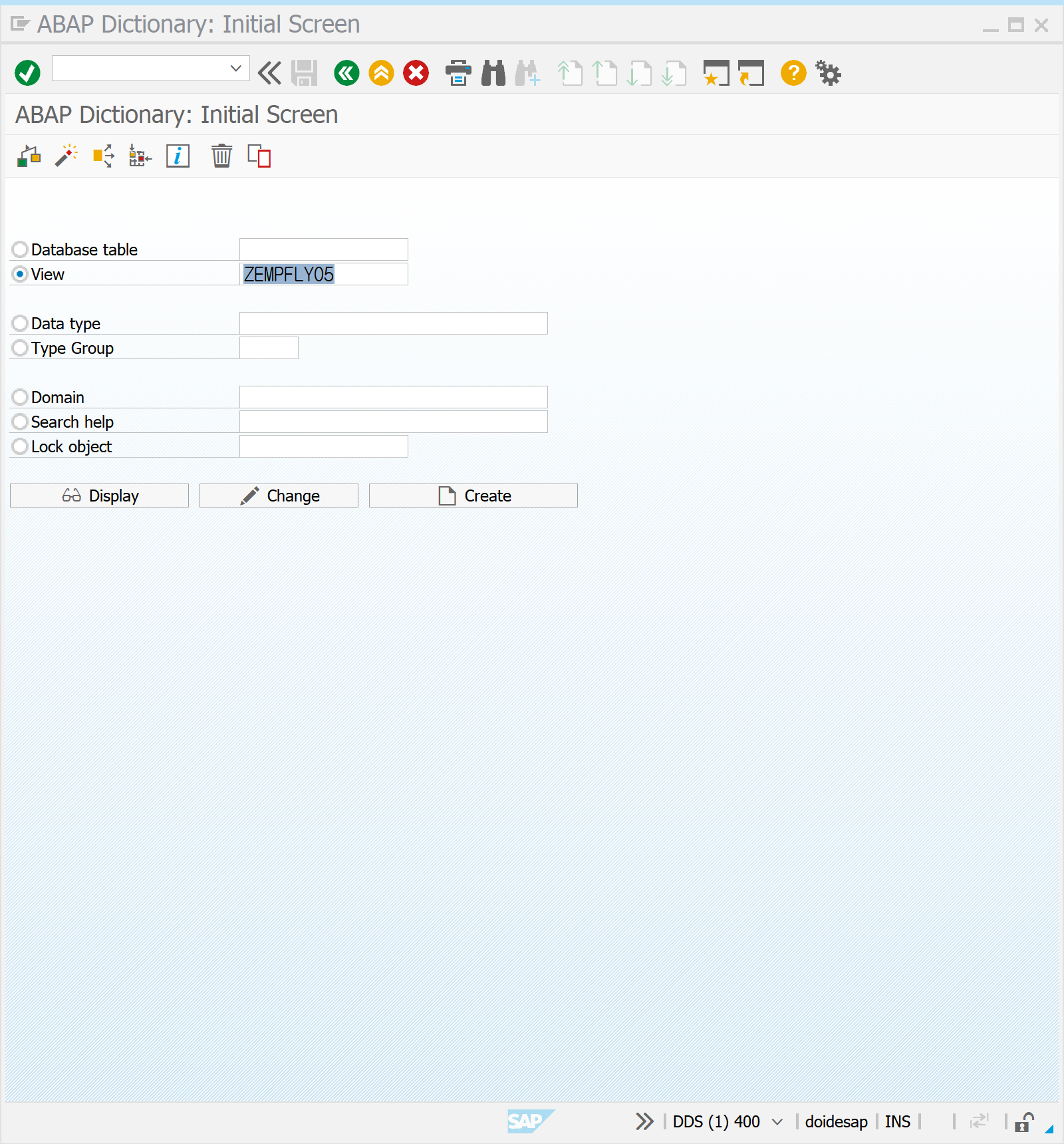
|  |
| --- |
| DATABASE VIEW |
| PROJECTION VIEW |
| HELP VIEW |
| MAINTANNCE VIEW |

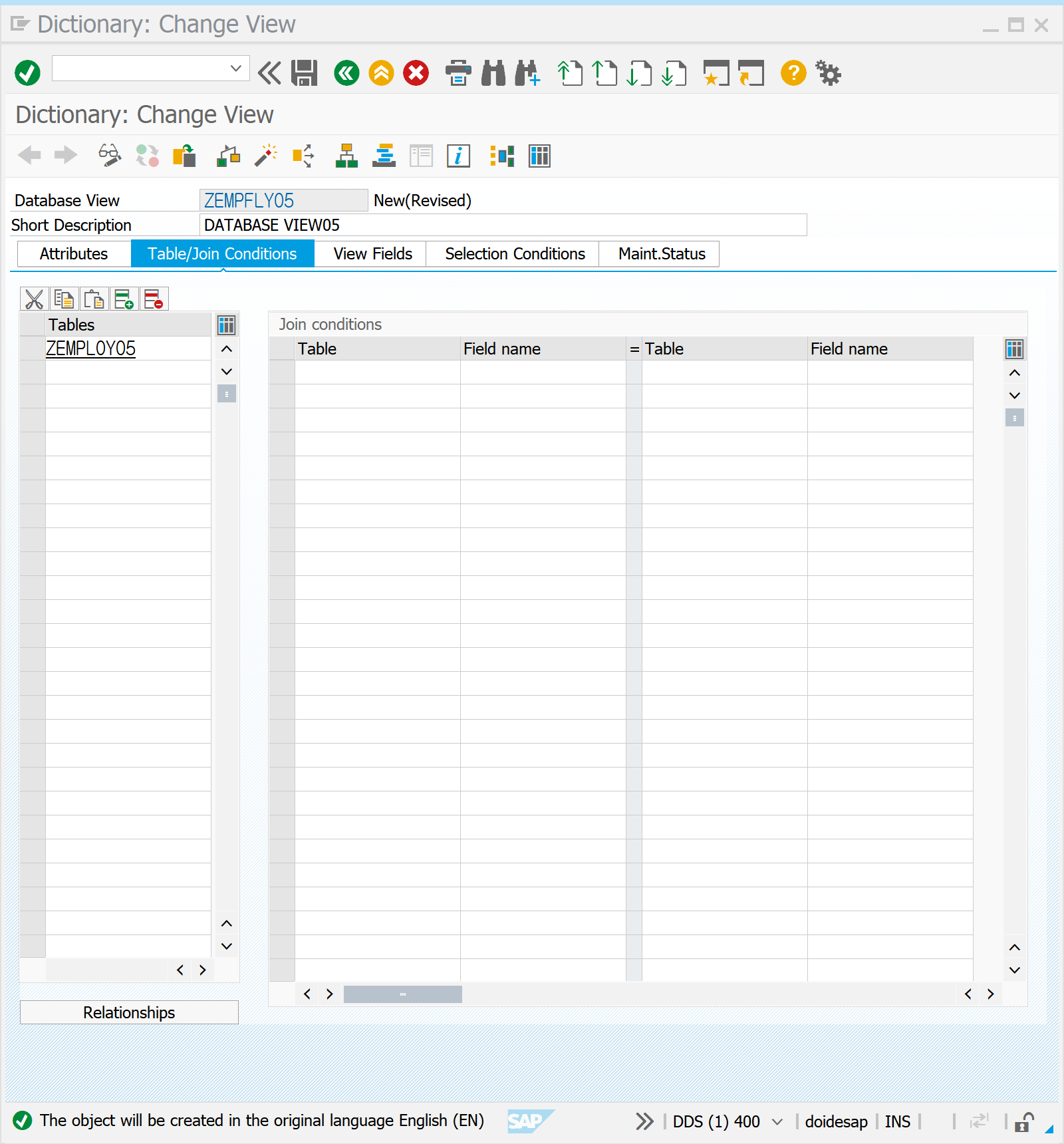
DATABASE VIEW |

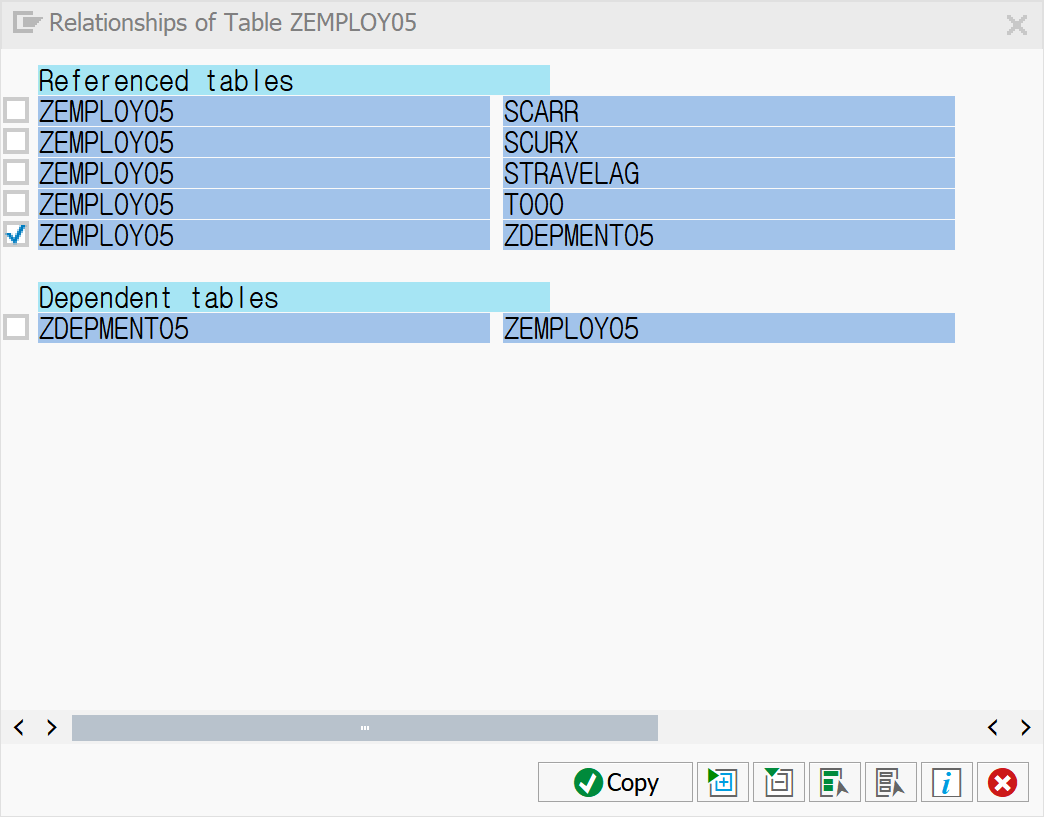


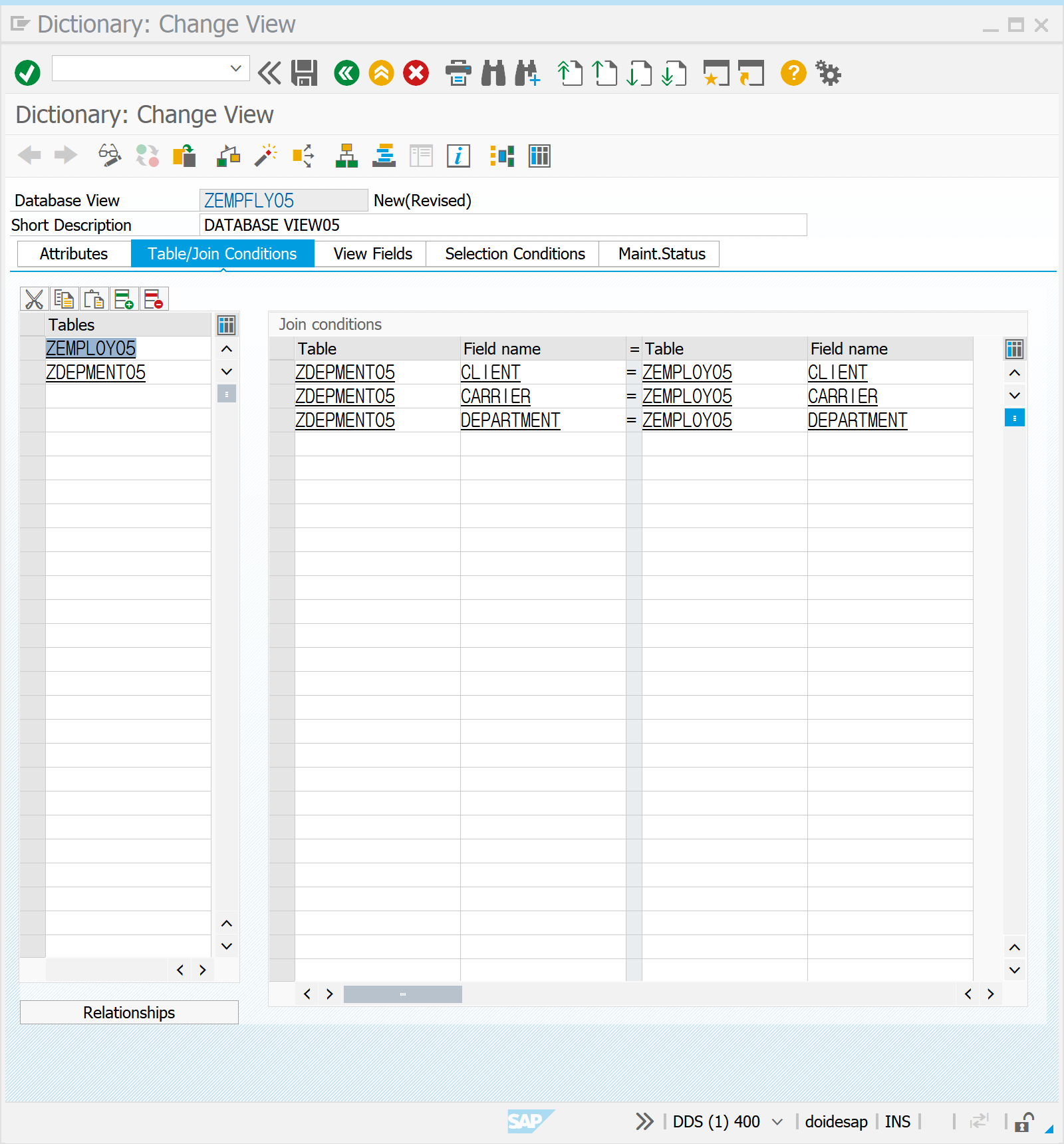
DATABASE VIEW는 여러 개의 테이블에서 필요한 데이터들을 추출한 VIEW를 의미한다. VIEW는 활성화되면 ABAP DICTIONARY에 의해 사용될 수 있다. ABAP DICTIONARY에 의해 정의되고 사용되기 때문에 일반 ABAP DICTIONARY 테이블과 같이 OPEN SQL, NATIVE SQL로 프로그램에서 사용하여 데이터를 가져올 수 있다. 만약 하나의 테이블만 사용하여 VIEW를 정의하면, MAINTANANCE STATUS를 이용하여 일기/쓰기를 정할 수도 있다. DATABASE VIEW는 TRANSPARENT TABLE만 사용할 수 있다.

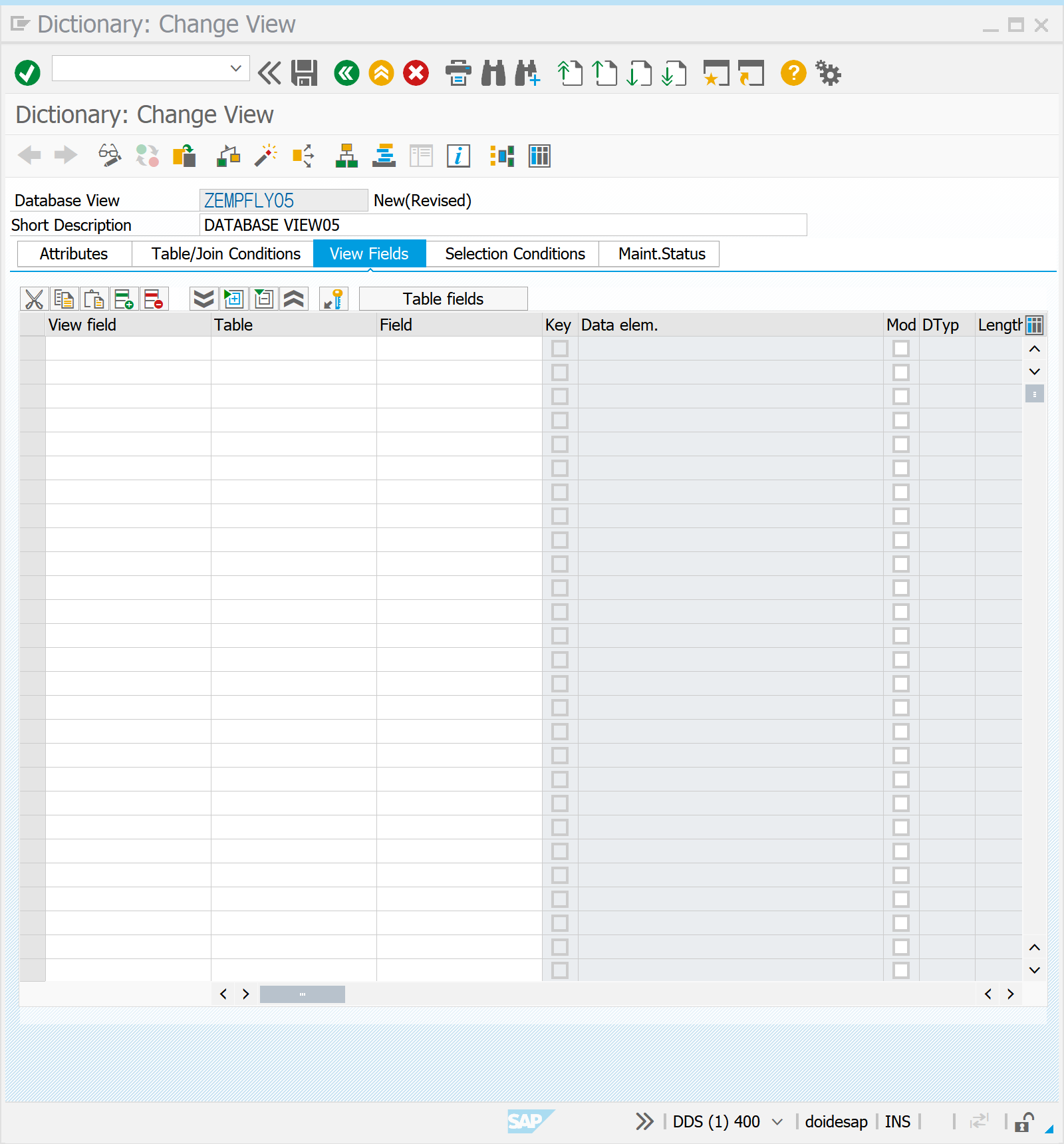
SE11 | DATABASE VIEW

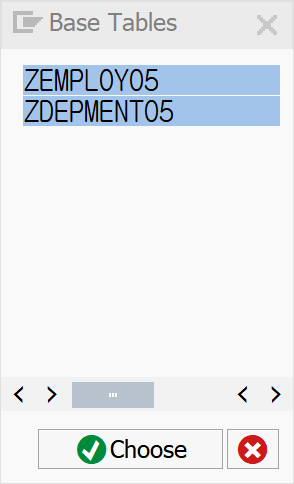


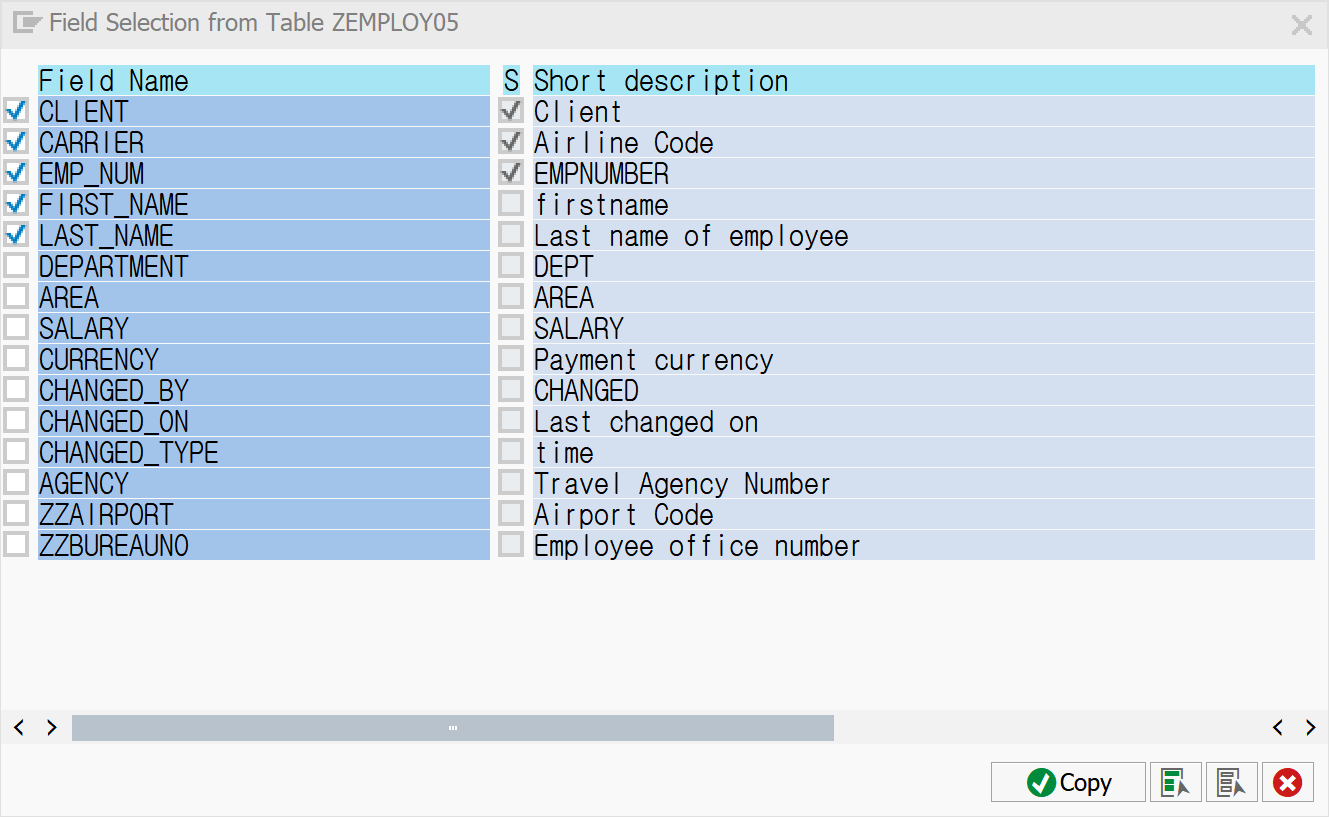


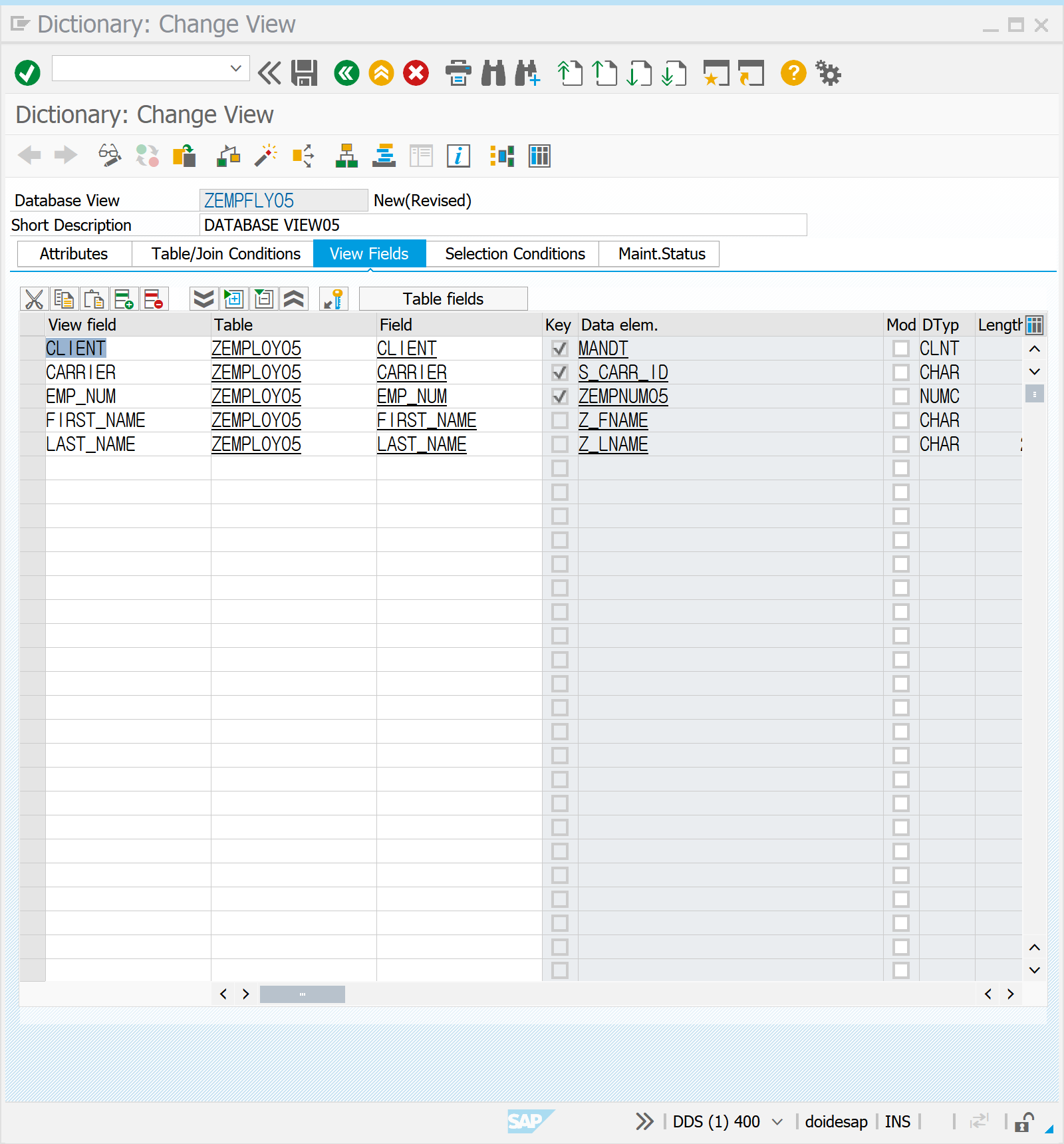


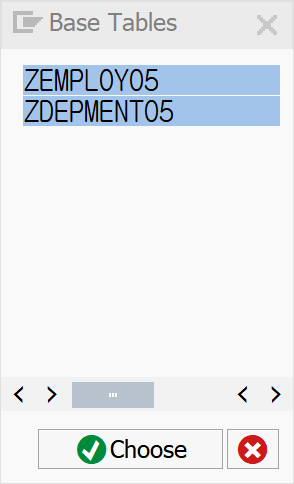


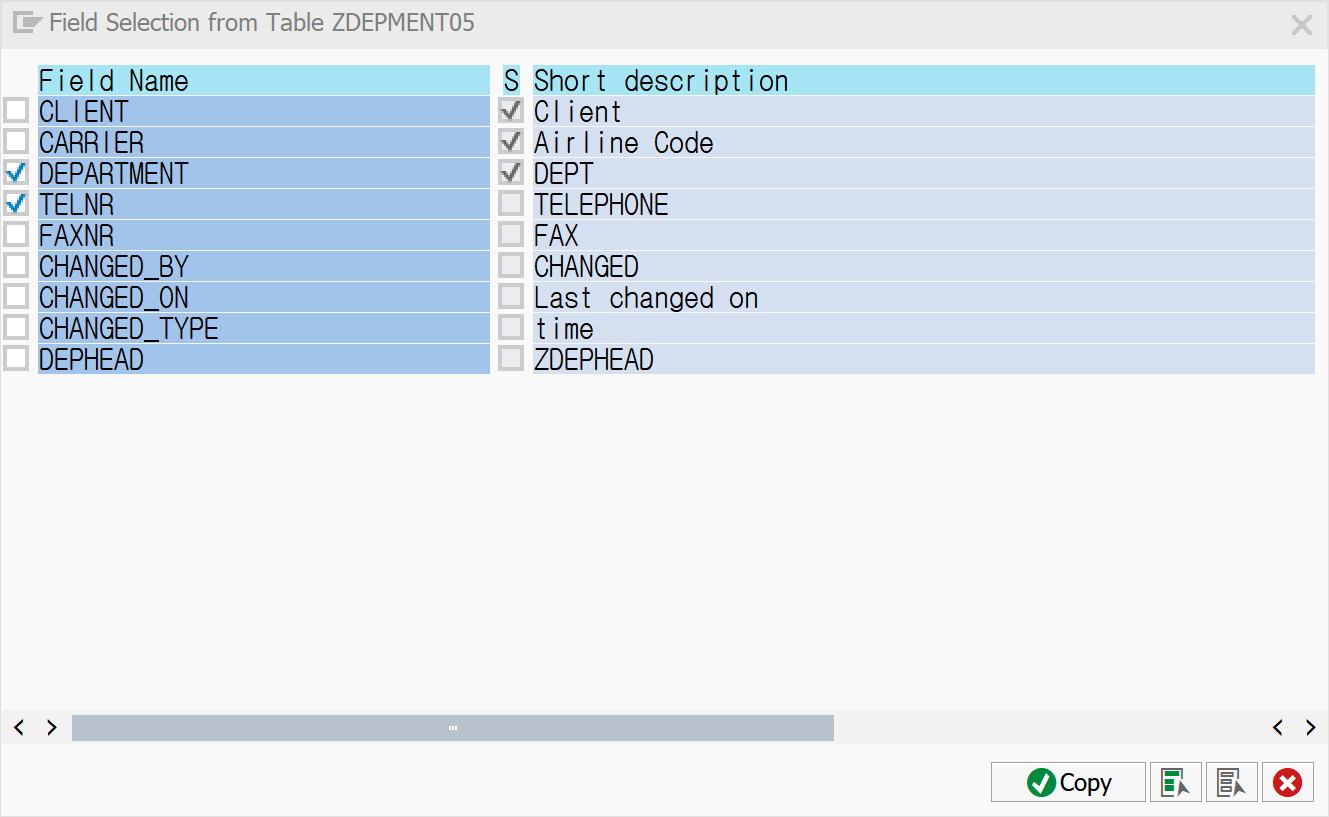


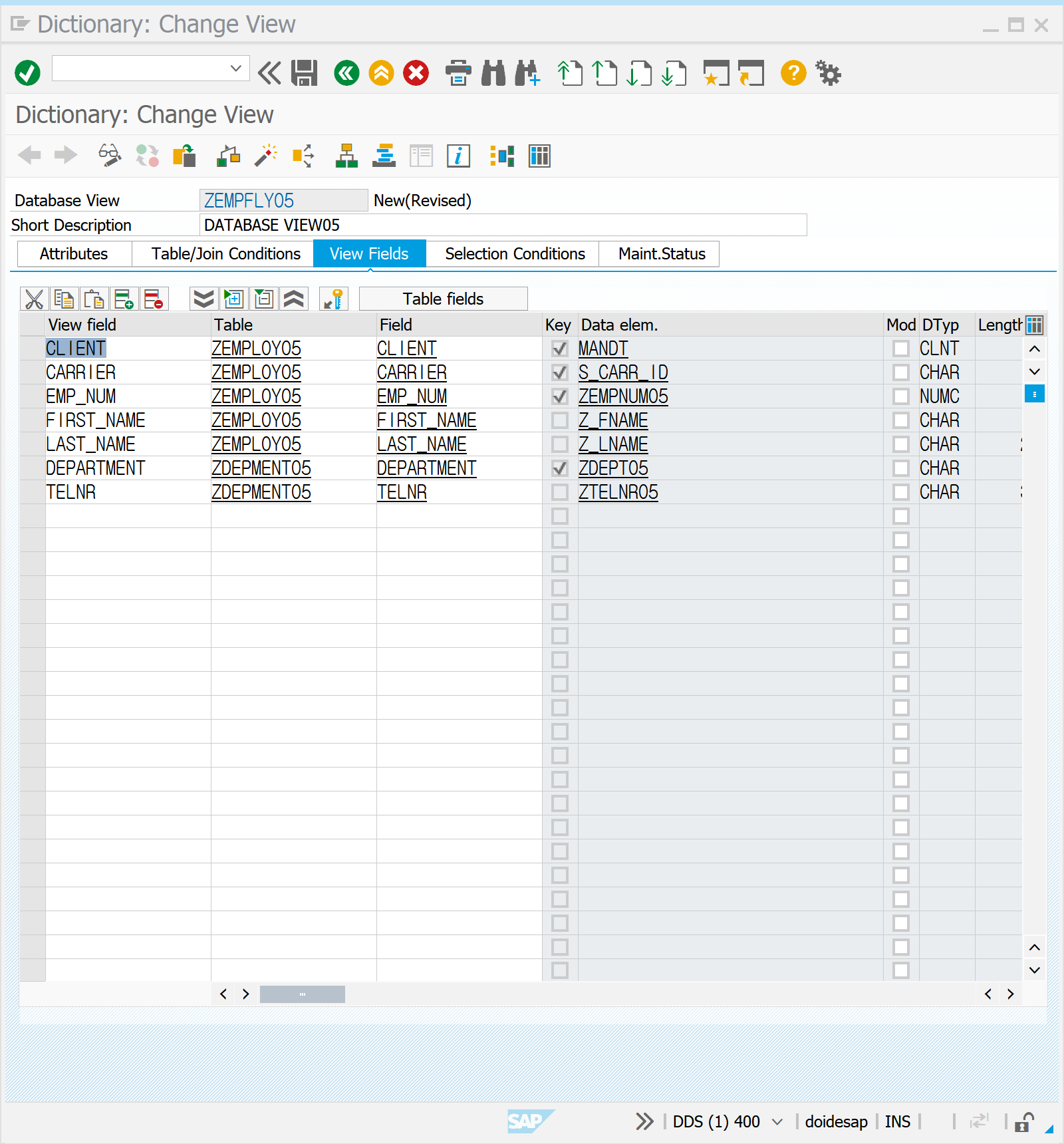


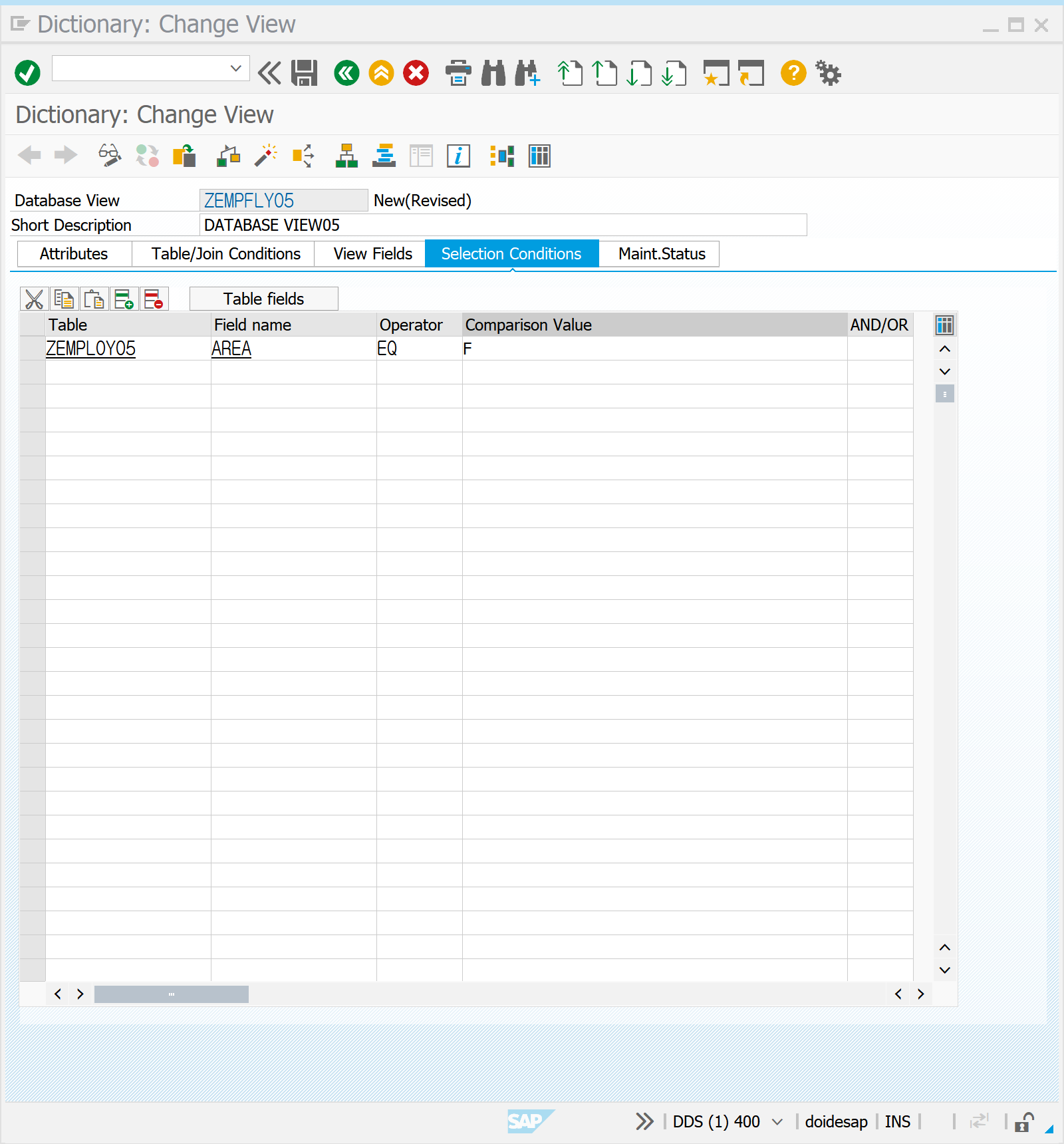




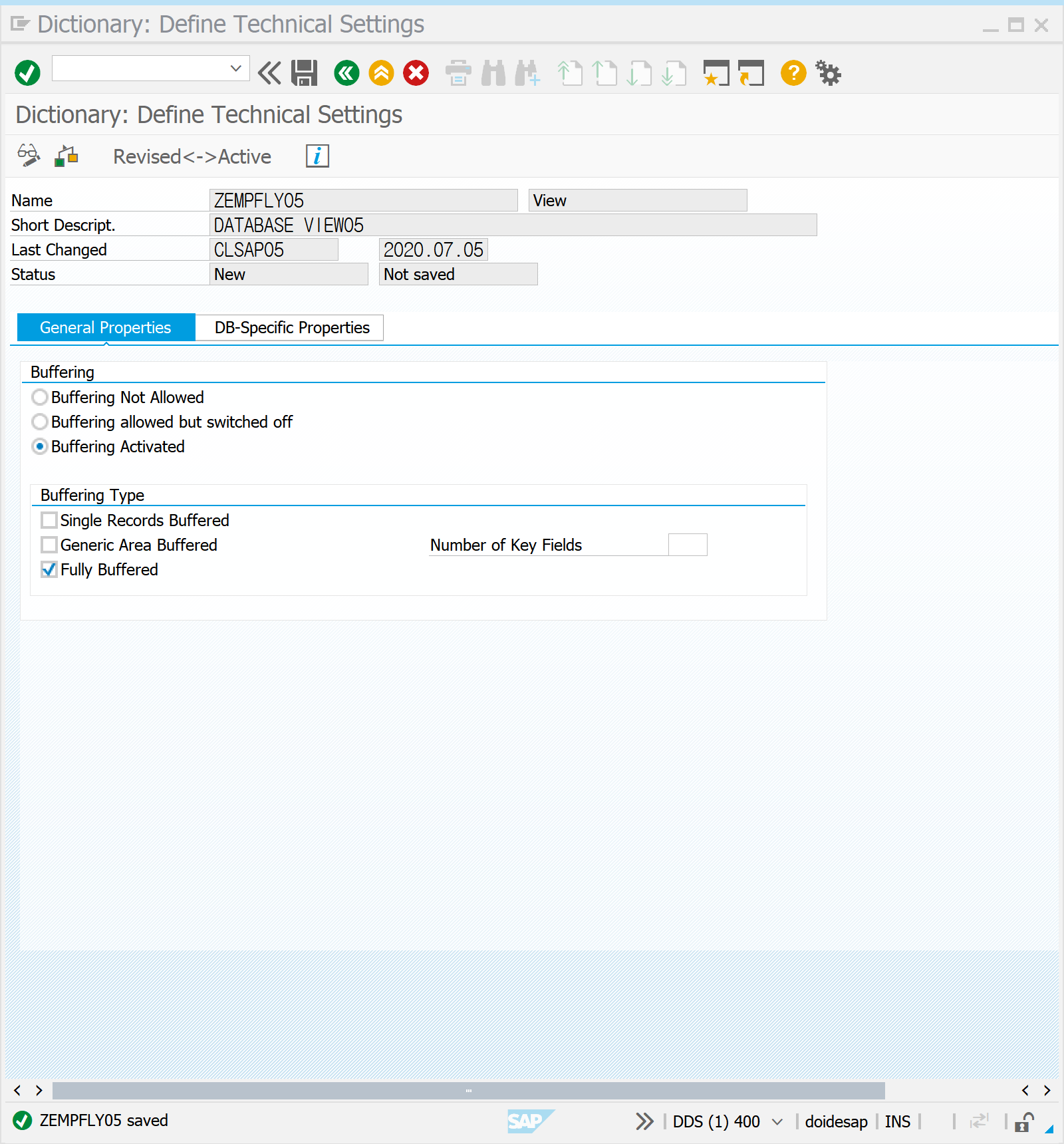








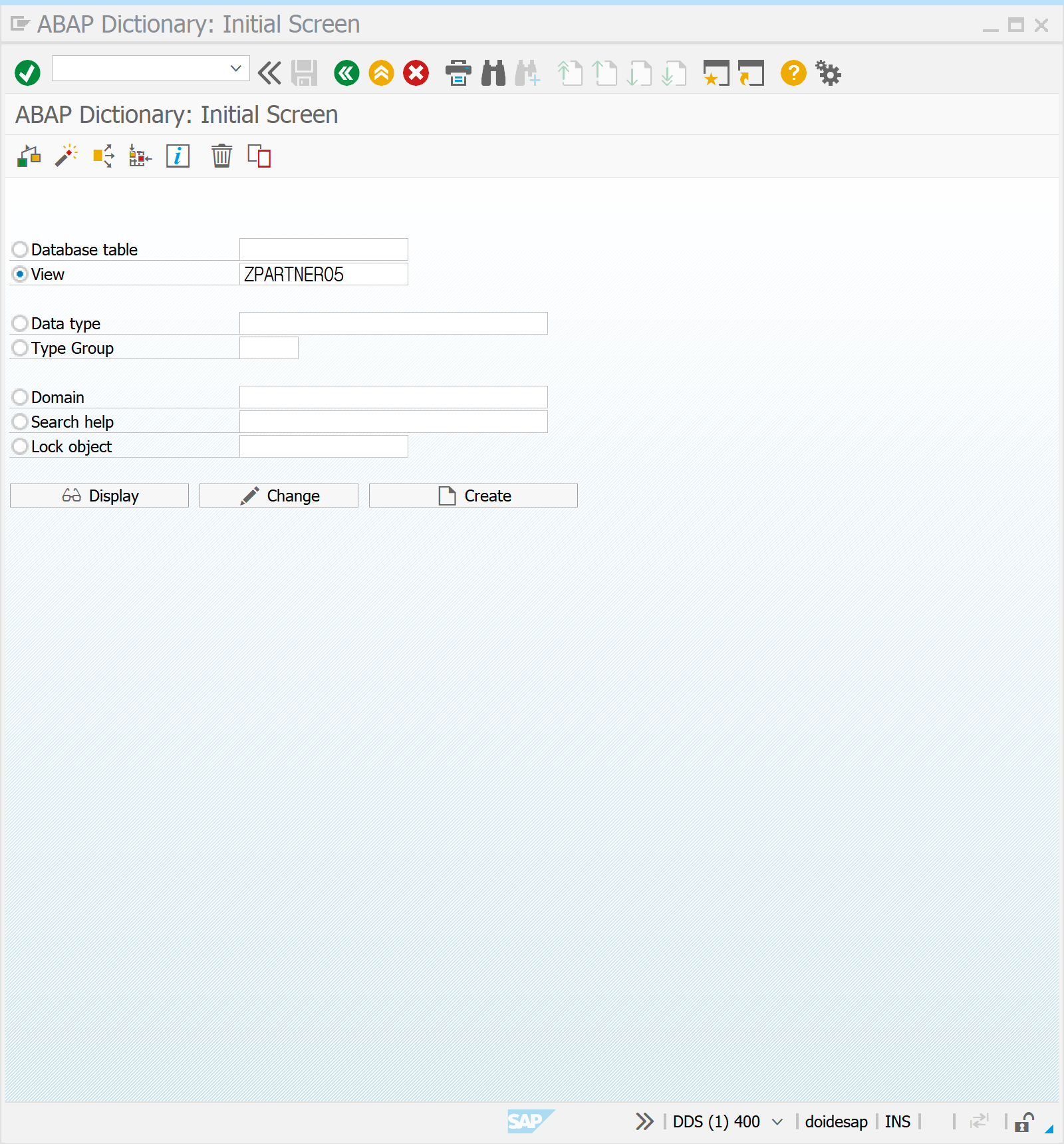
SE11 | DATABASE VIEW | DICTIONARY TECHNICAL SETTINGS



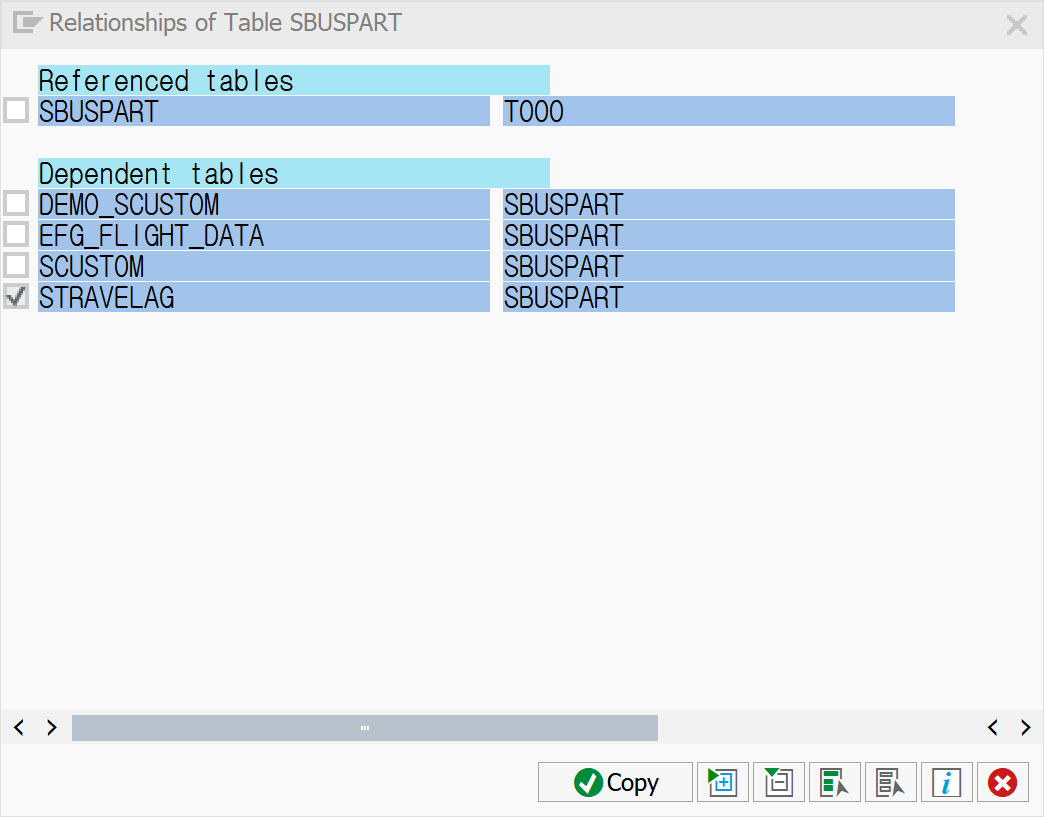
MAINTANANCE VIEW |

MAINTANANCE VIEW는 여러 개의 테이블을 동시에 유지보수 할 수 있는 VIEW를 의미한다. 이때 테이블들은 반드시 FOREIGN KEY로 연결되어 있어야 한다. MAINTANANCE VIEW를 이용해 한 번에 데이터를 조회하고, 변경하고, 생성할 수 있다. 즉, FOREIGN KEY로 연결된 테이블들의 원하는 필드들을 하나로 모아 MAINTANACE VIEW로 생성하고, VIEW에서 데이터를 입력, 삭제, 변경하면 실제 테이블의 데이터도 수정된다. ABAP 프로그램 내 SELECT문과 같은 곳에서는 사용할 수 없다.

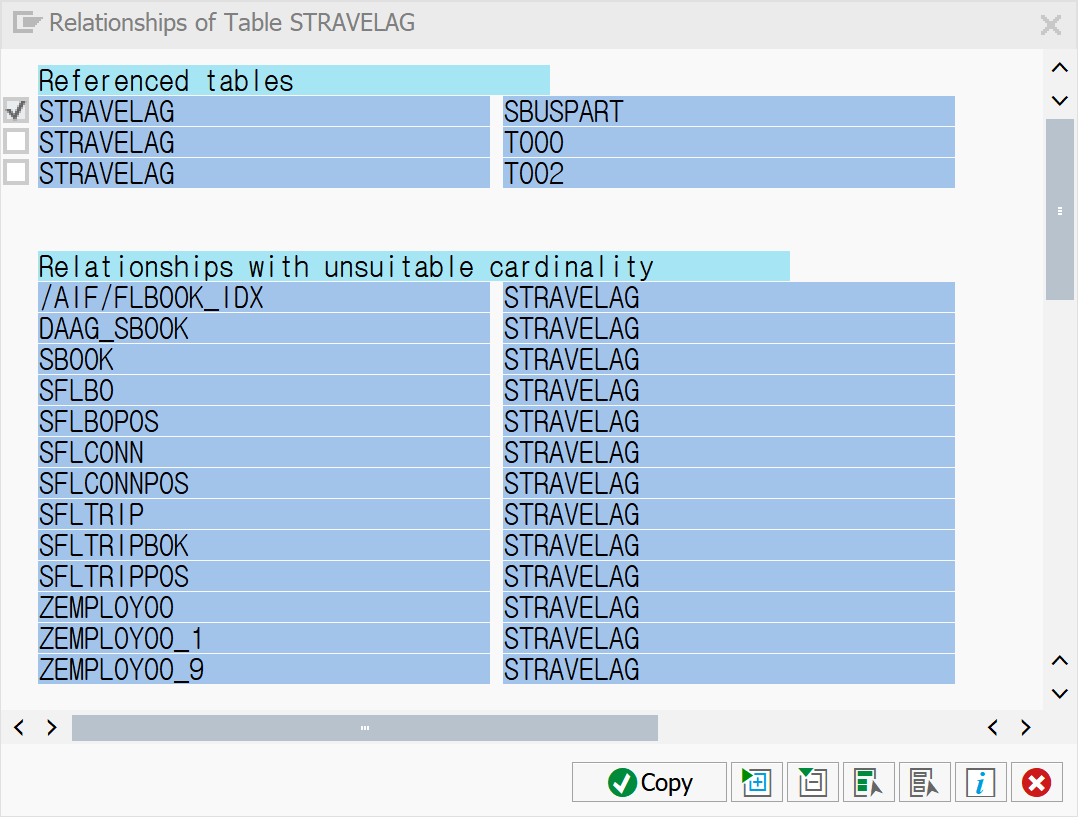
SE11 | MAINTANACE VIEW |

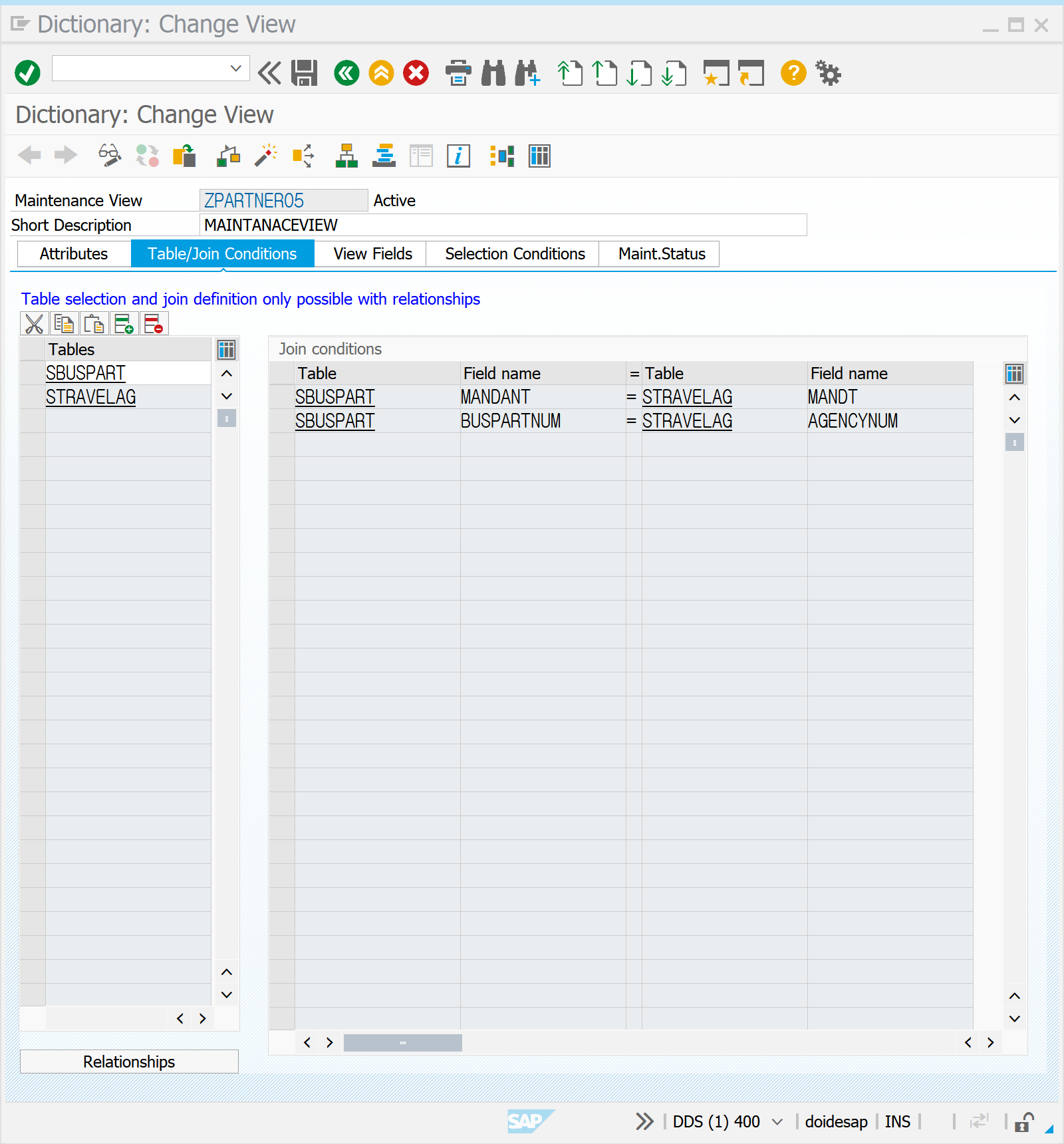


SE11 | MAINTANANCE VIEW | SBUSPART | RELATIONSHIP



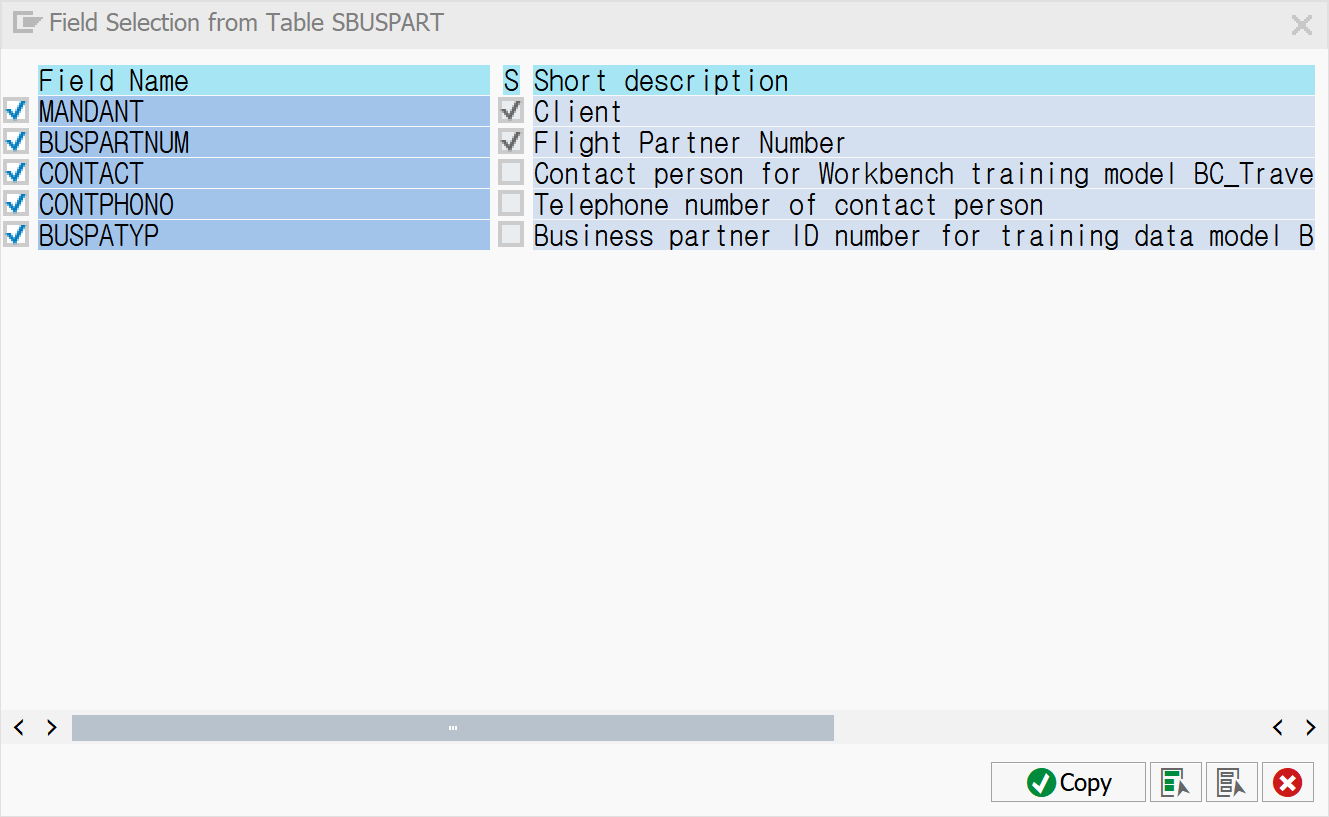
SE11 | MAINTANANCE VIEW | STRAVELAG | RELATIONSHIP



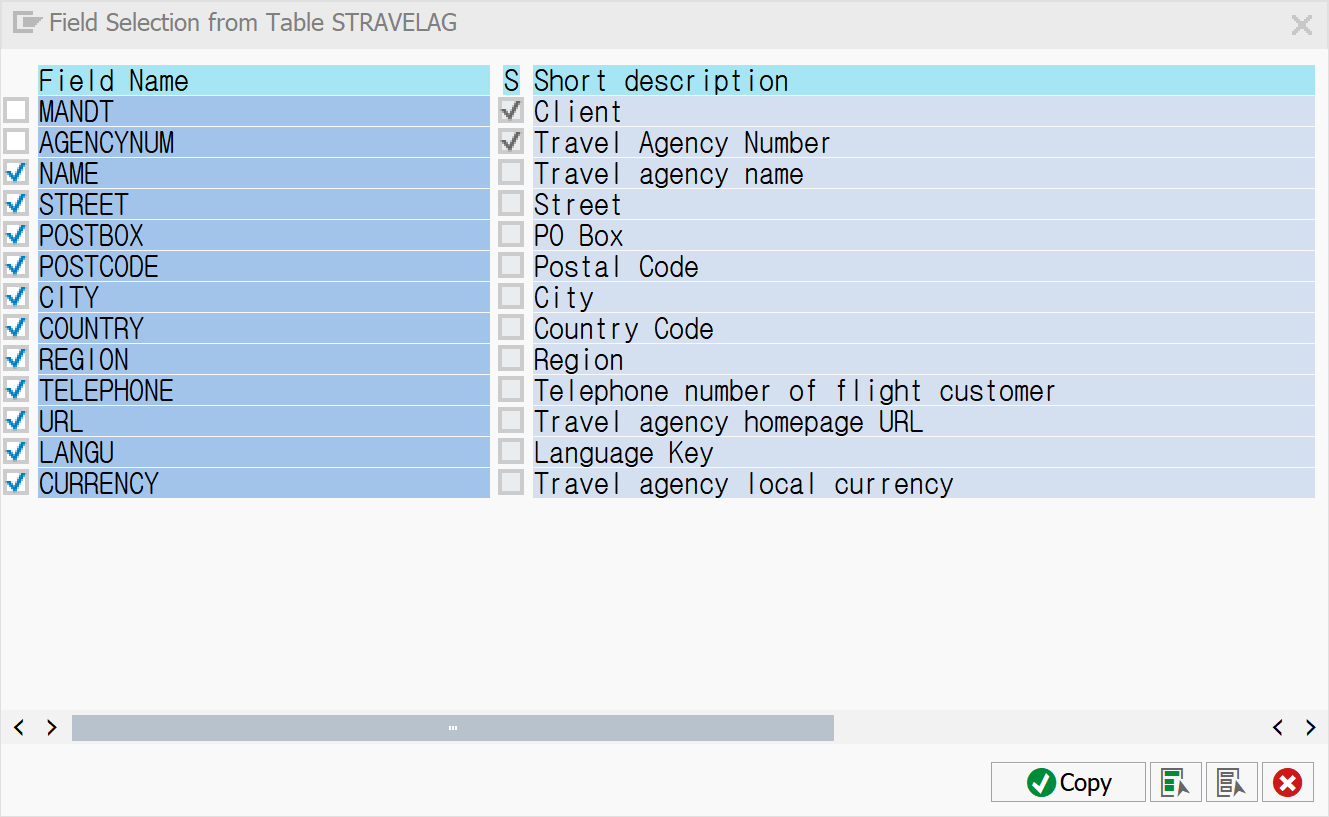


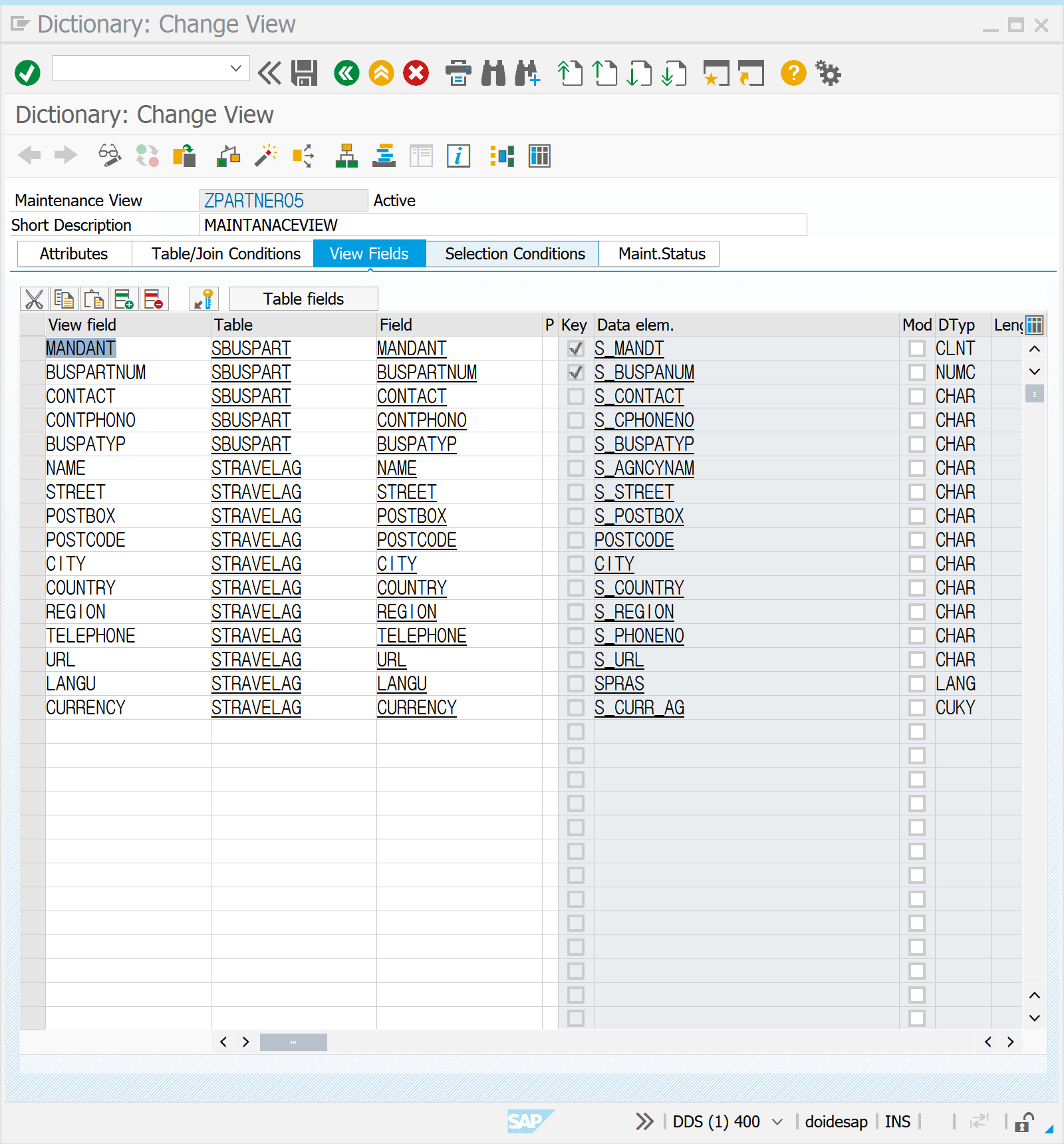
SE11 | MAINTANANCE VIEW | VIEW FIELDS











VIEW FIELD OPTION |

|  |  |
| --- | --- |
| R | VIEW FIELD 는 오직 읽기만 가능 |
| S | VIEW FIELD 들을 SUBSET 으로 설정 |
| H | VIEW FIELD 를 숨김 속성으로 지정, 화면에 조회 X |

ACCESS OPTION |

|  |  |
| --- | --- |
| READ ONLY | VIEW DATA 는 읽기만 가능 |
| READ CHANGE  DELETE & INSERT | VIEW DATA 는 읽기 변경 삭제 추가 가능 |
| READ & CHANGE | VIEW DATA 는 변경할 수 있으나 삭제나 추가 불가 |
| READ & CHANGE  TIME-SPECIFIC VIEWS | NON-TIME-DEPENDENT KEY 가 존재하는  VIEW의 경우 같은 값이 존재하는  DATA만 추가될 수 있도록 설정 |

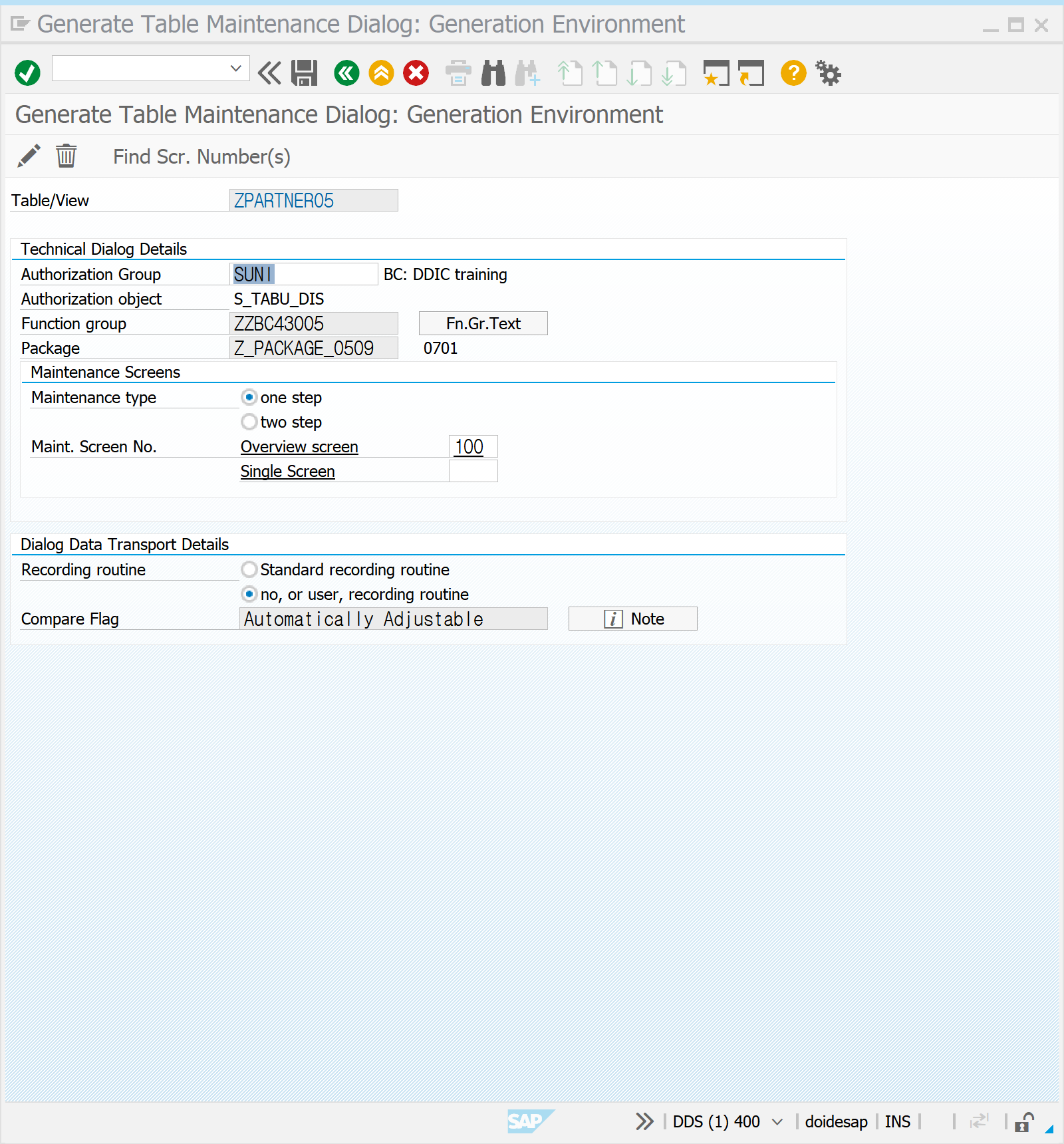
TABLE MAINTANANCE DIALOG |

MAIINTANANCE DIALOG 는 MAINTANACE VIEW와 TRANSPARENT TABLE의 데이터를 관리하는 화면을 제공하며 TABLE과 VIEW에서 생성법은 같으므로 MAINTANACE VIEW의 생성 과정을 테이블에도 똑같이 적용할 수 있다.

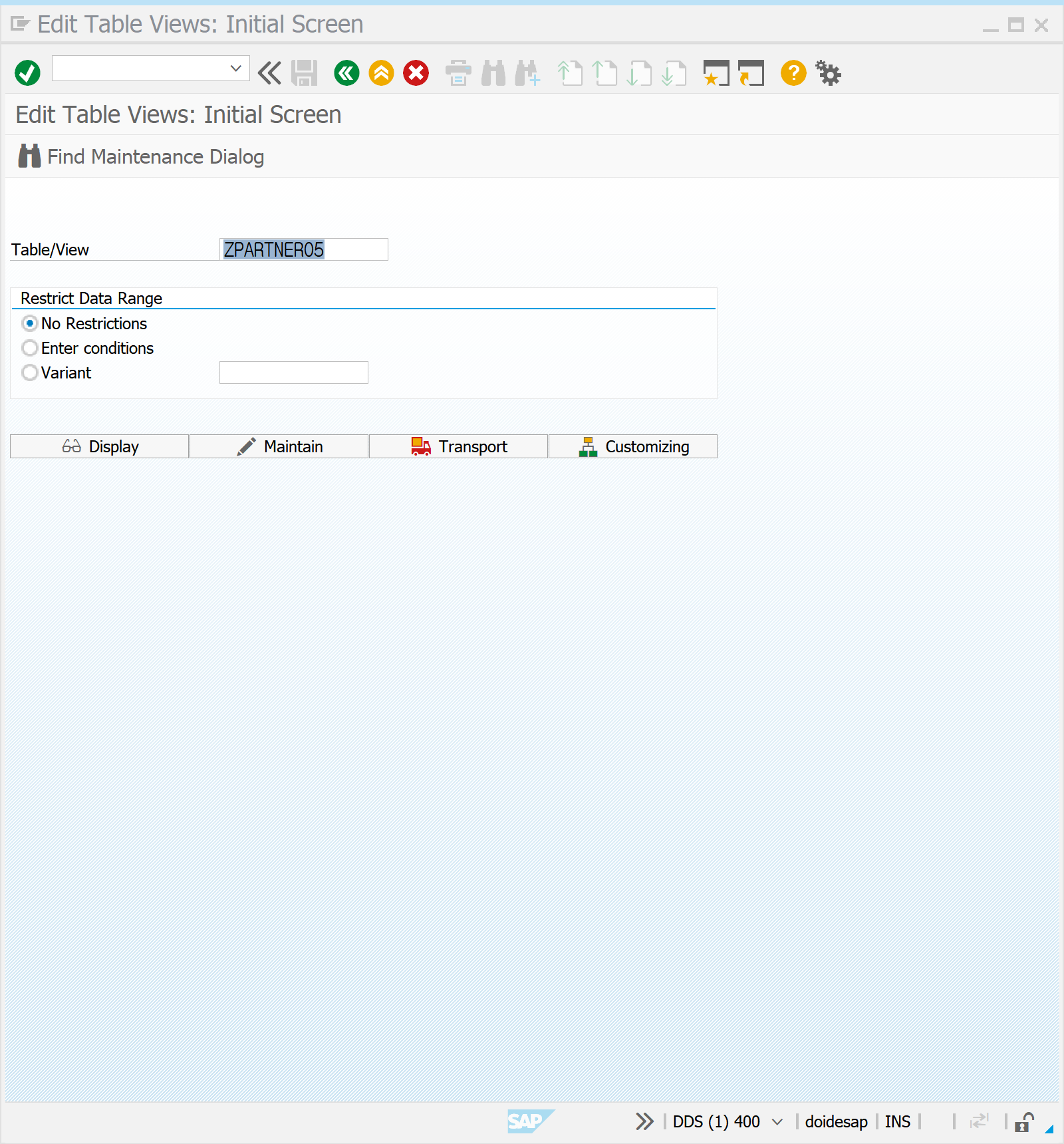
TABLE MAINTANANCE DIALOG OPTION |

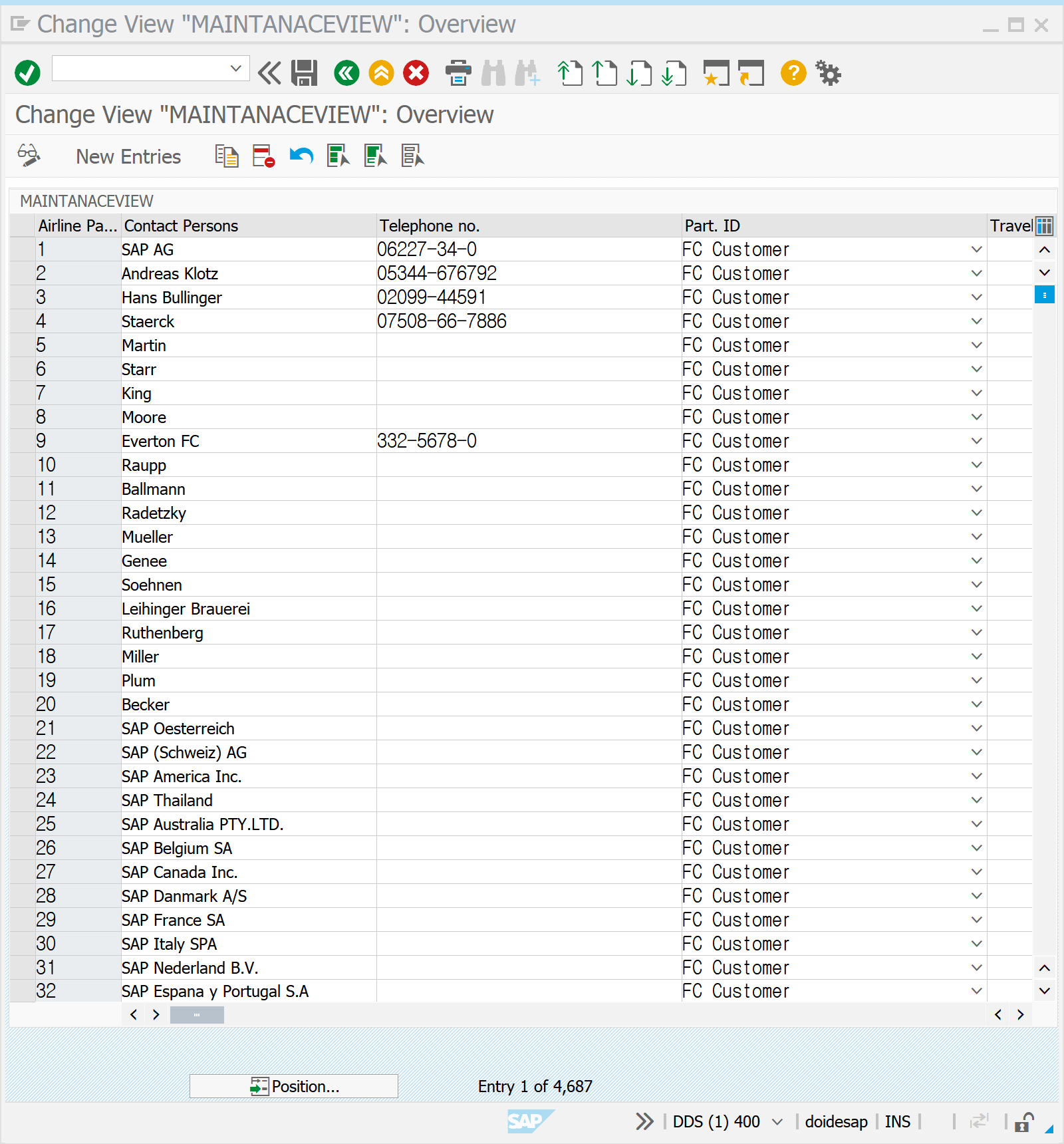
|  |  |
| --- | --- |
| AUTHORIZATION GROUP | GROUP; TABLE/VIEW 데이터를 관리할 수 있는  권한 그룹으로 체크 필요 없을 때는 ‘&NC&’ 을 선택 |
| FUNCTION GROUP | 소속되는 FUNCTION GROUP 선택 |
| MAINTANANCE TYPE | 1 또는 2 단계 화면 선택  1단계는 모든 필드를 포함하는 OVERVIEW 스크린  2단계는 KEY FIELD와 CHAR 타입의 필드가 OVERVIEW에 조회, 모든 필드는 DETAIL 화면제공 |
| MAINTANANCE SCREENS | MAINTANANCE 화면의 내부 번호로 일반적으로는 FIND SCR. 버튼을 이용해 시스템 제안 번호 선택 |
| RECORDING ROUTINE | MAINTANACE VIEW에서 변경된 DATA를 REQUEST에 포함해 전송할 것인지 선택 |

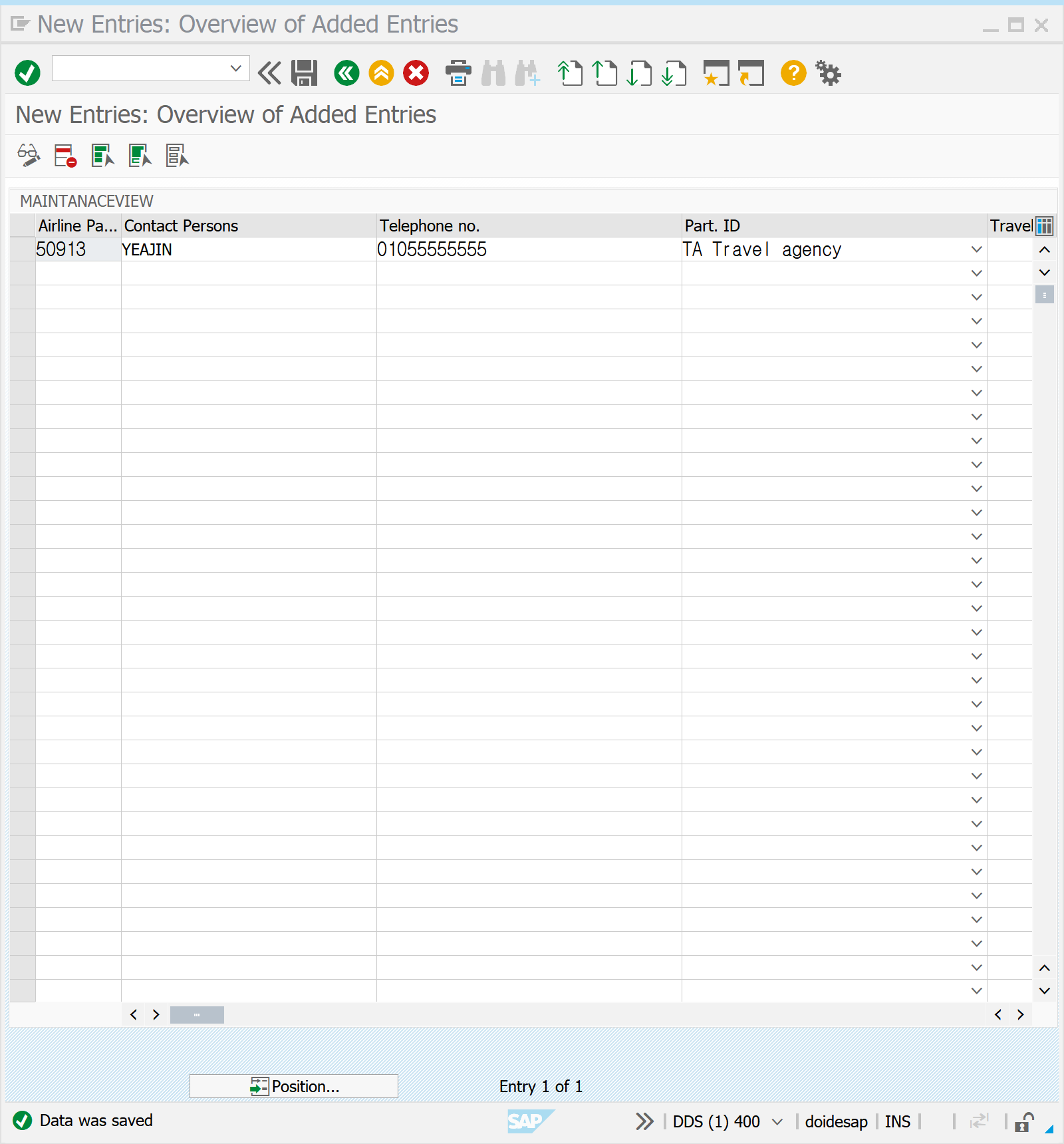
SE11 | 이어서 | UTILITIES | TABLE MAINTANANCE GENERATOR



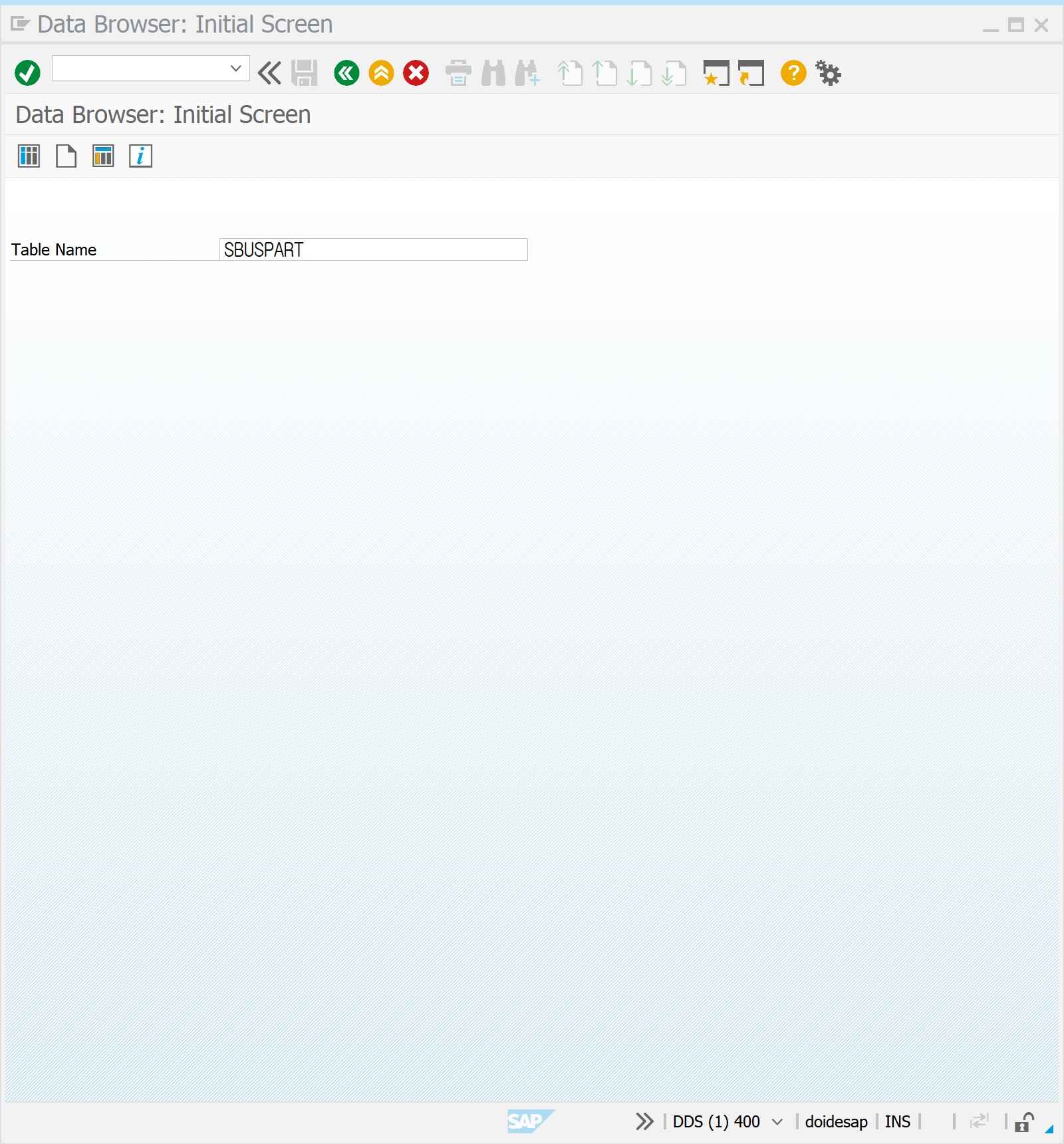
SE11 | 이어서 | SYSTEM | SERVICES | TABLE MAINTENANCE | EXTENDED TABLE MAINTENANCE | ENTER VIEW NAME | MAINTAIN

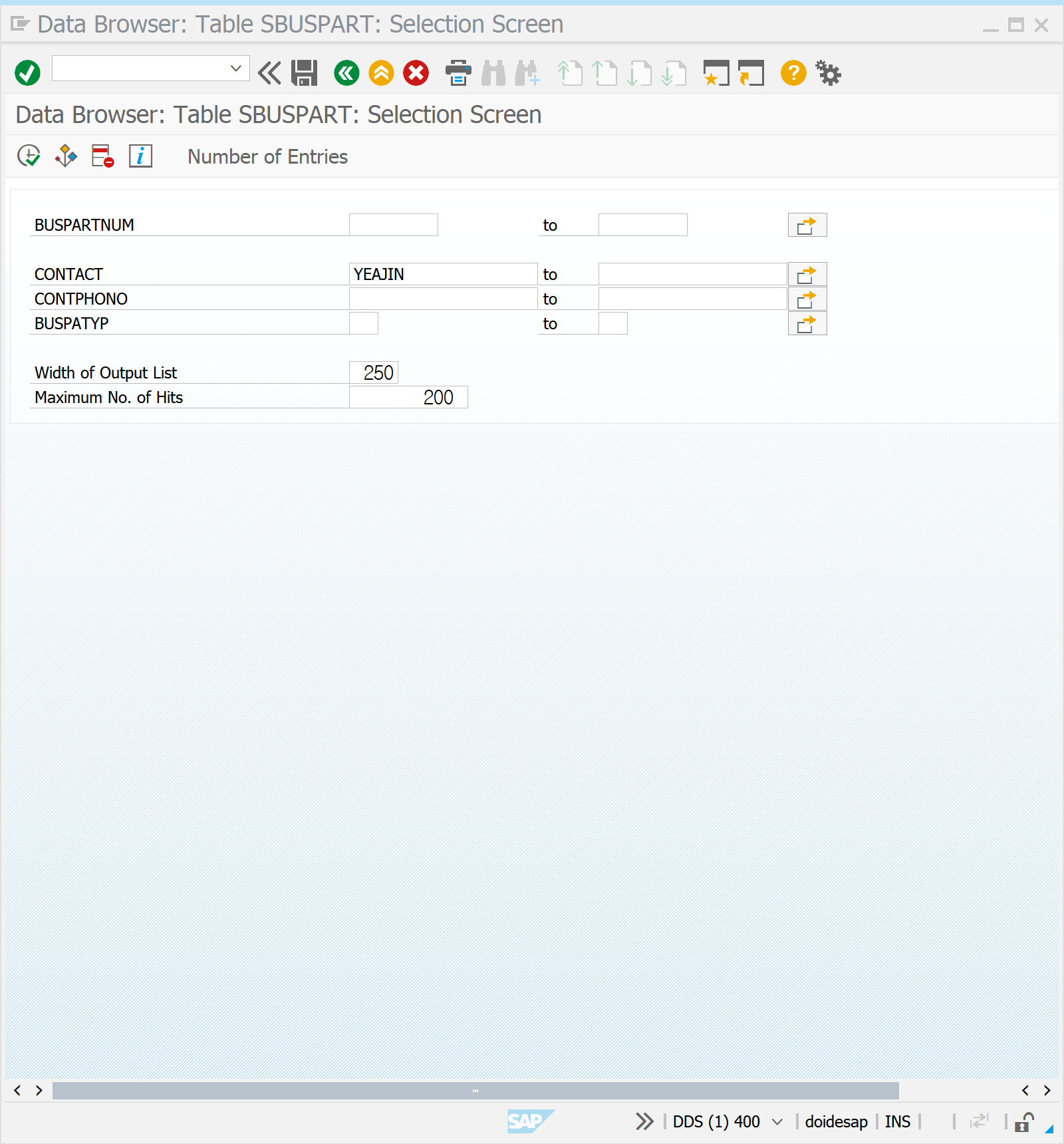


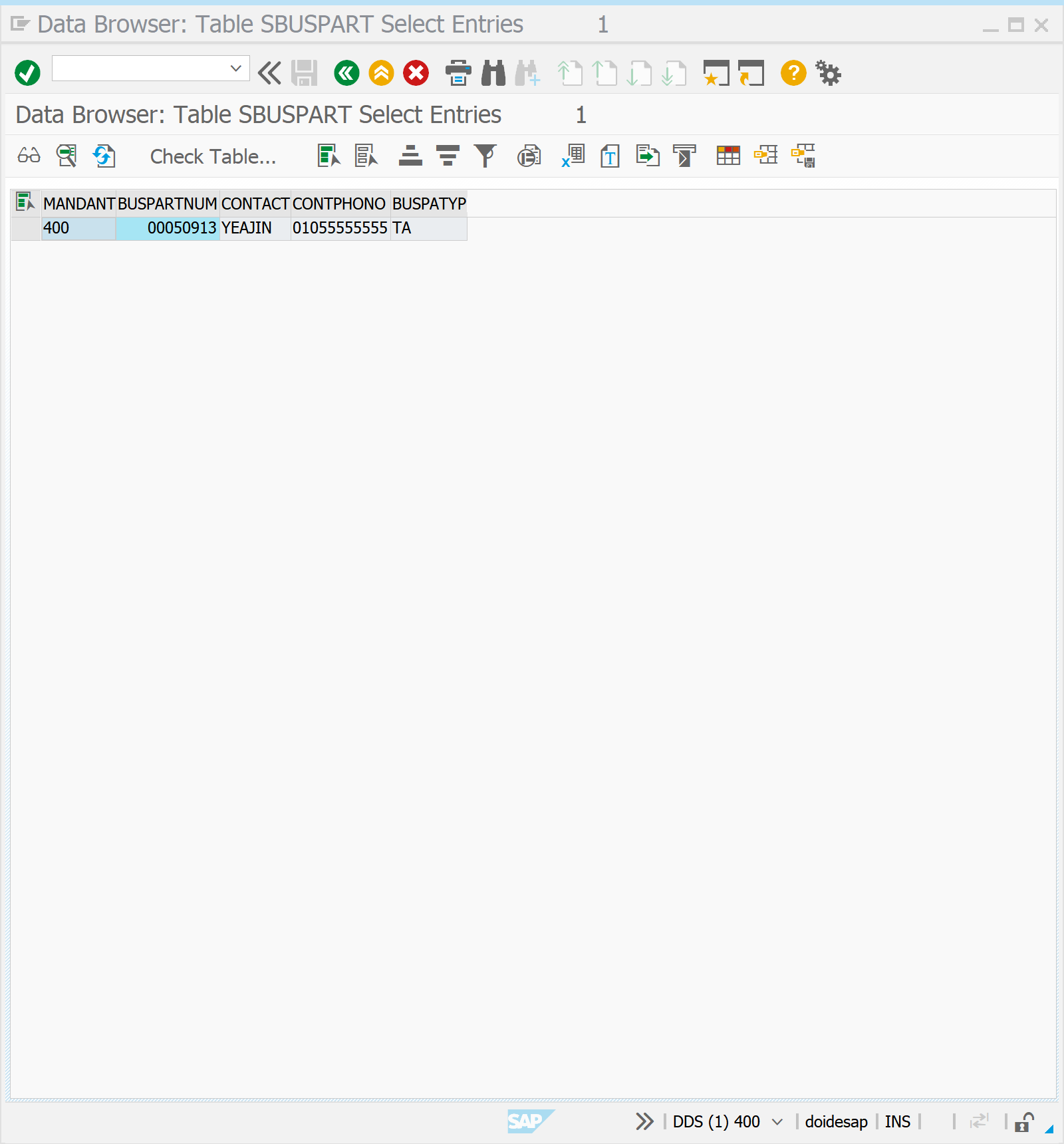




SE16 | 확인해보기







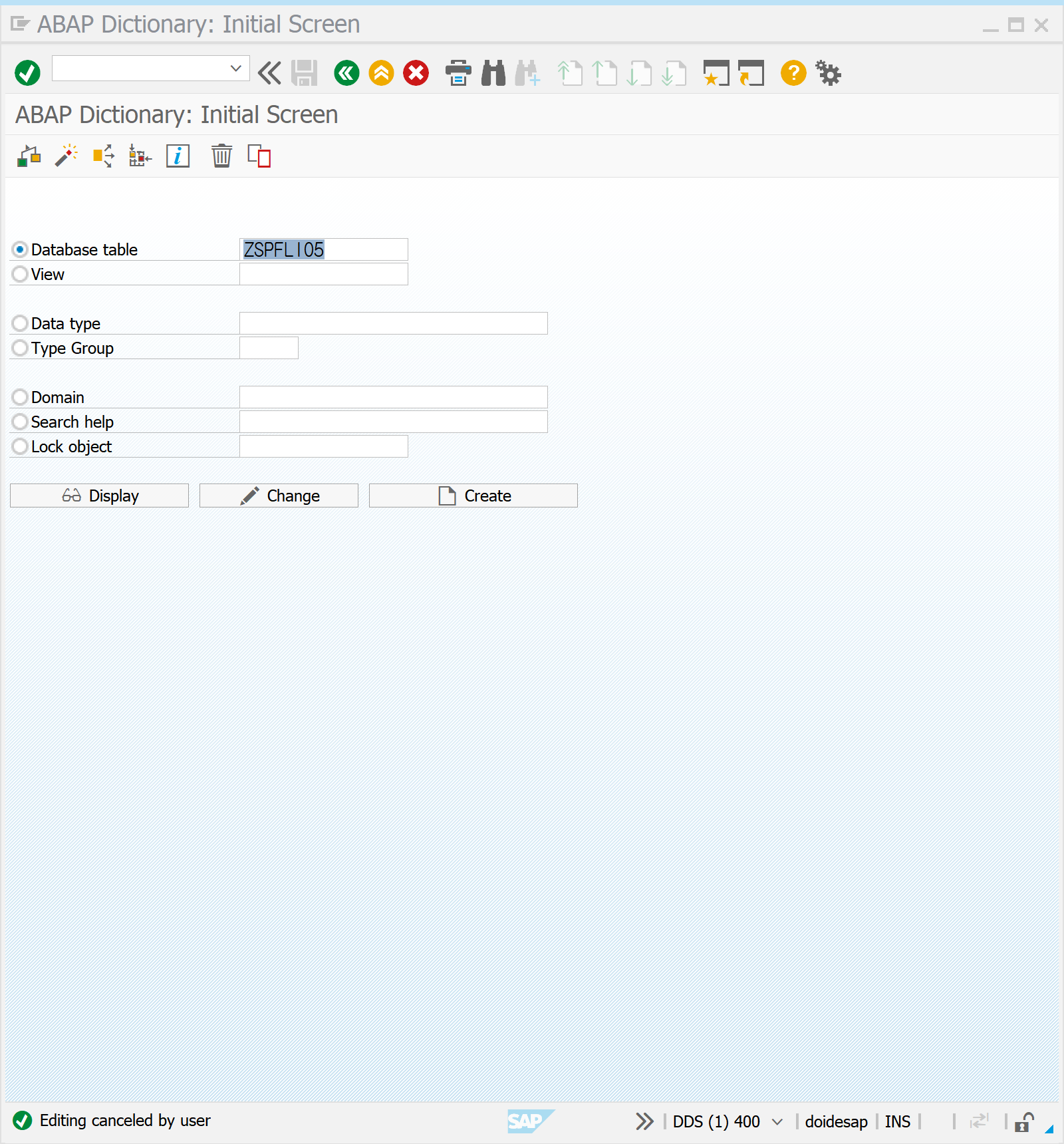
VIEW CLUSTER |

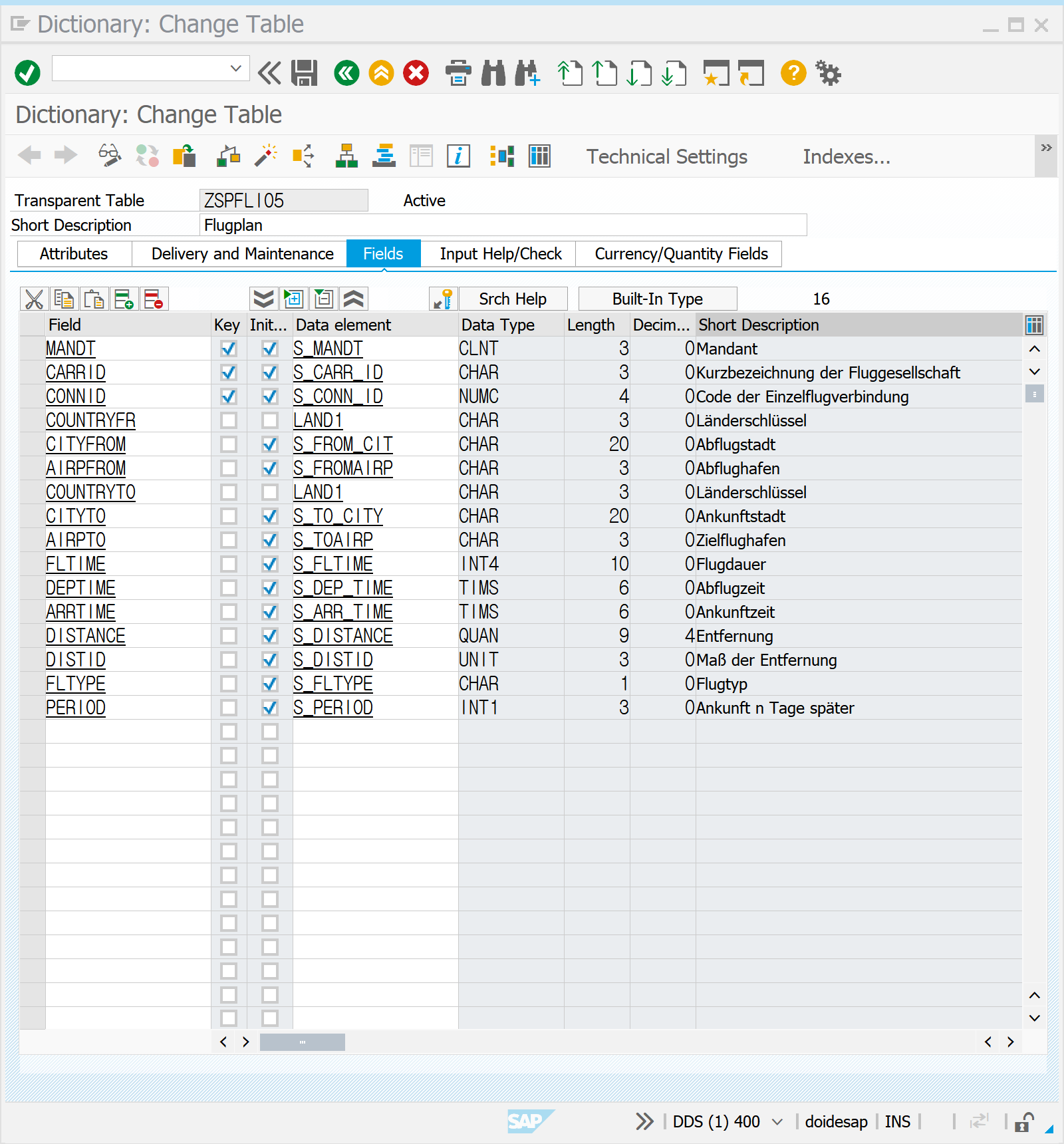
단위 업무와 관련된 마스터 데이터 여러가지가 연계된 경우 VIEW CLUSTER을 이용해 사용자가 일목요연하게 데이터를 관리할 수 있도록 지원한다. 이는 SAP 환경 설정을 하는 IMG | IMPLEMENTATION GUIDE 에서 데이터를 유지보수 할 때 주로 사용되는 방법이다. VIEW CLUSTER 는 종속적인 관계와 독립적인 관계 2 가지 유형으로 생성할 수 있다.

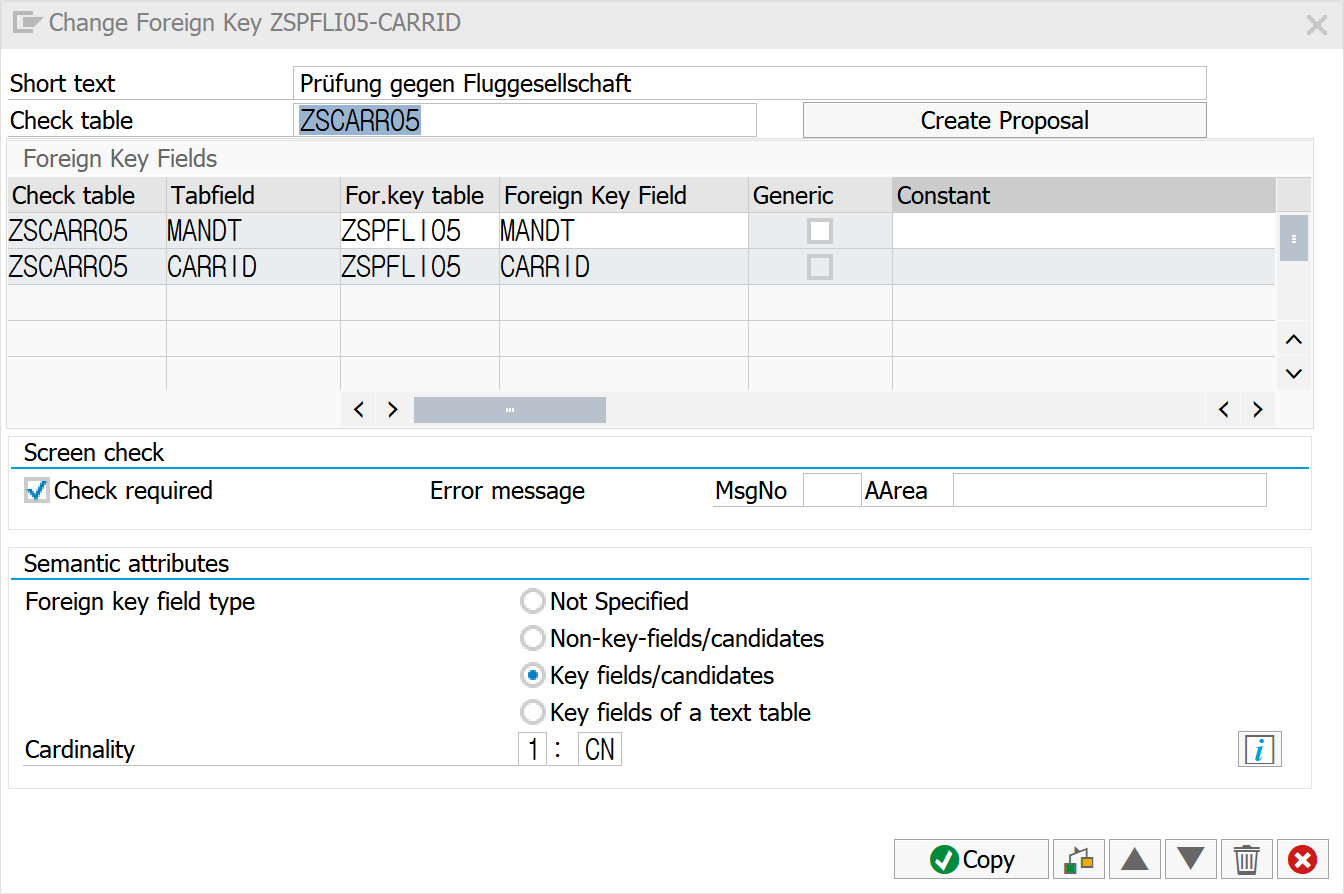
SE11 | VIEW CLUSTER | COPY

|  |  |
| --- | --- |
| ZSCARR05 | CSARR |
| ZSPFLI05 | SPFLI |
| ZSFLIGHT09 | SFLIGHT |

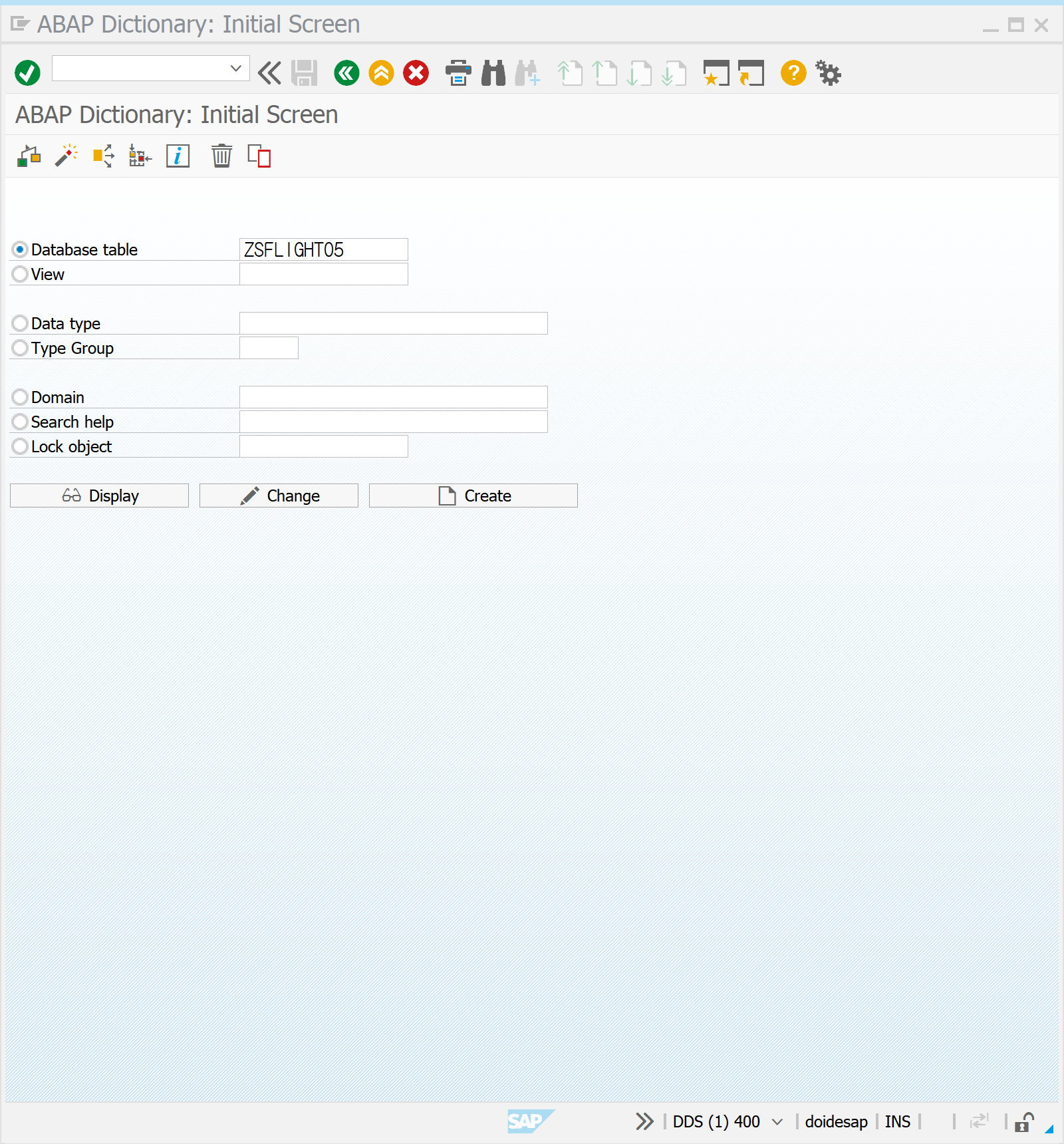
SE11 | VIEW CLUSTER | FOREIGN KEY | ZSPFLI05

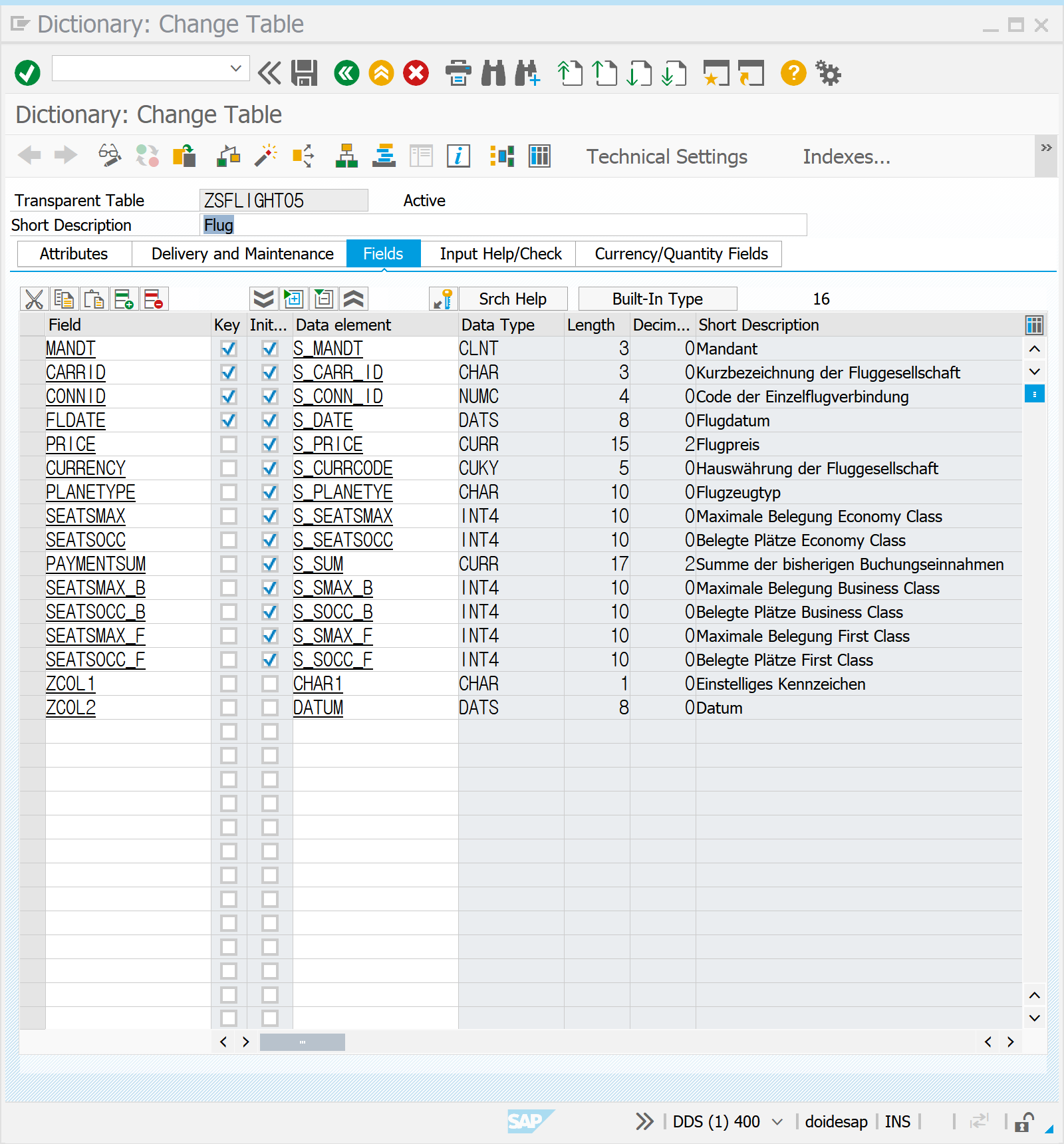


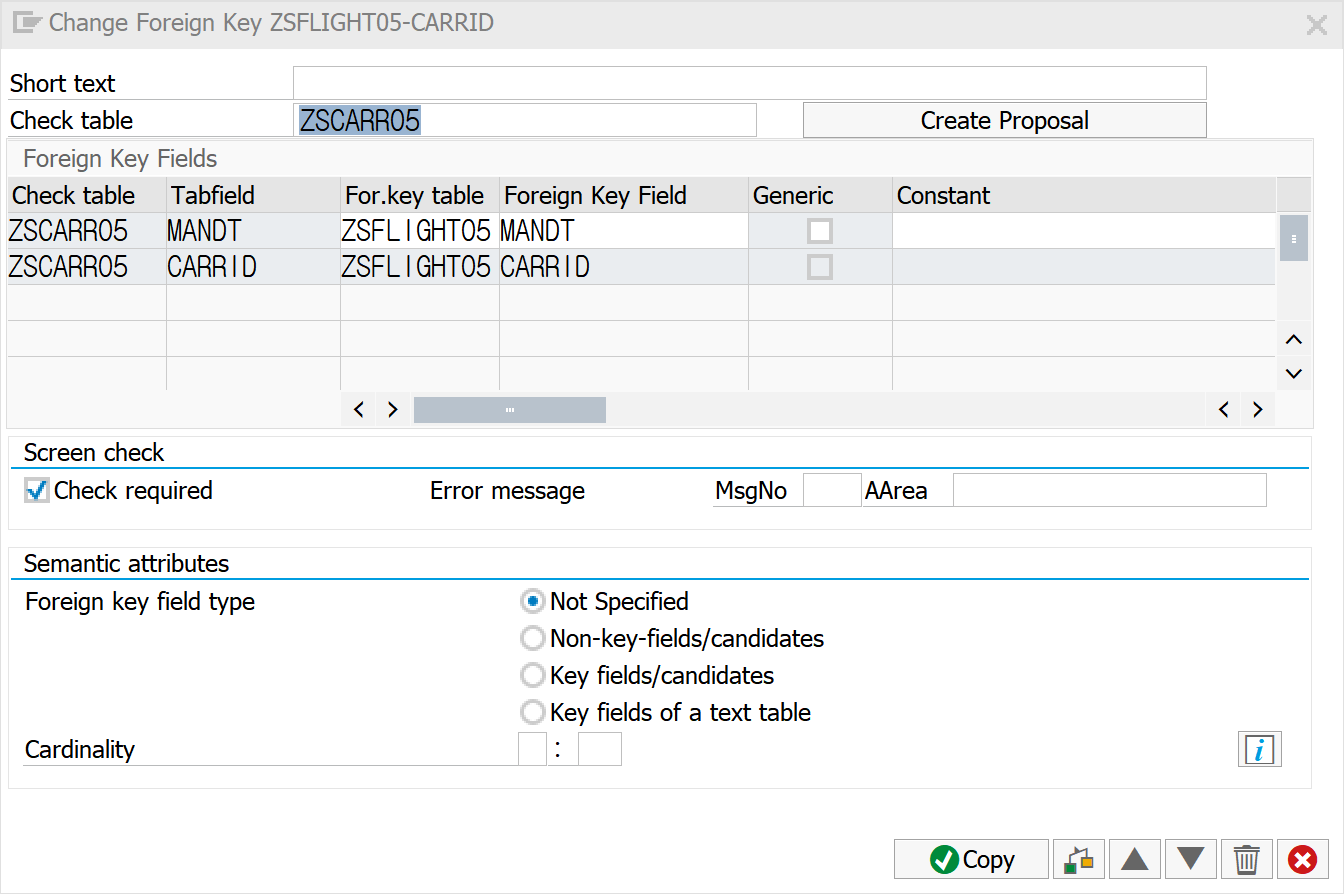


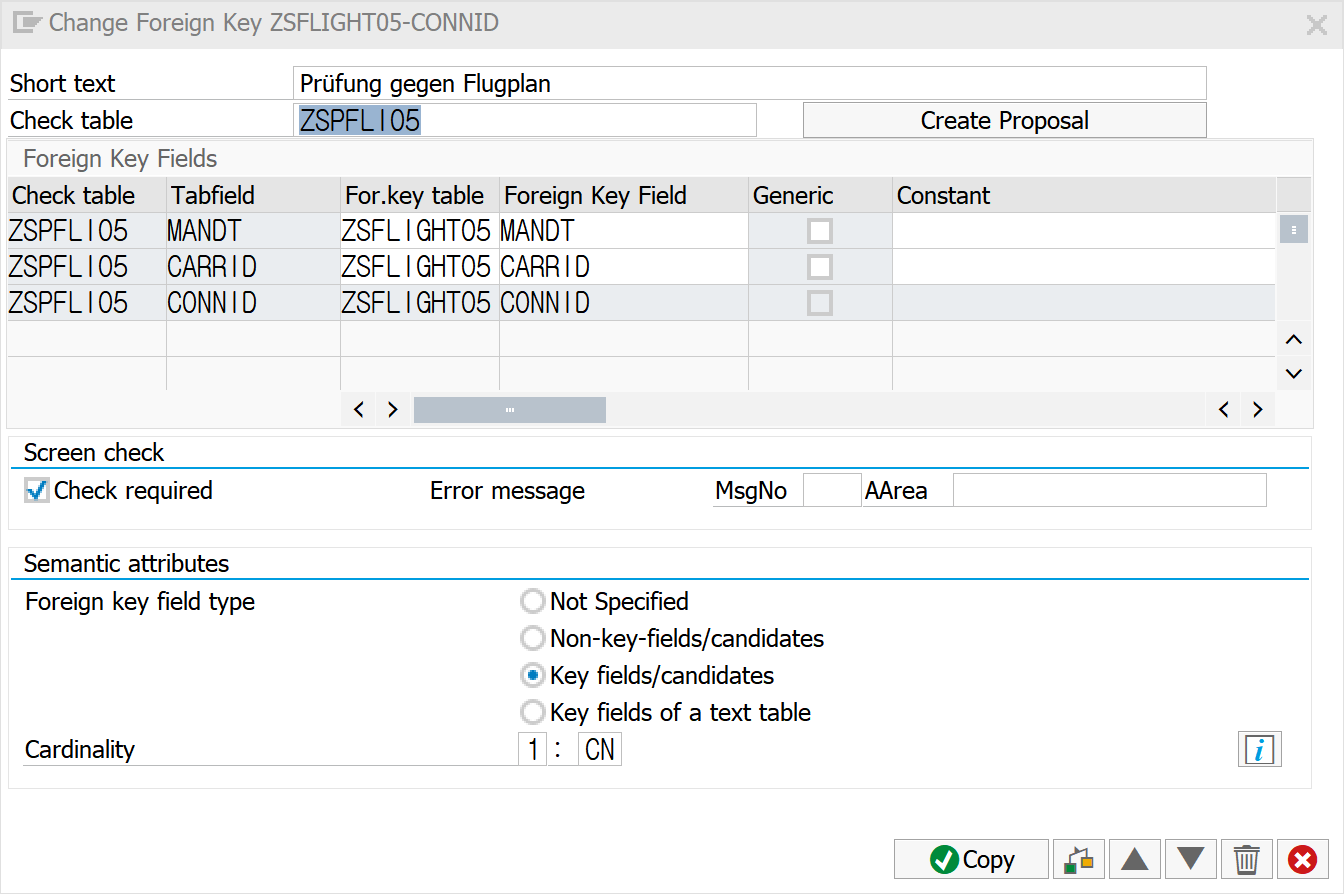


SE11 | VIEW CLUSTER | FOREIGN KEY | ZSFLIGHT05



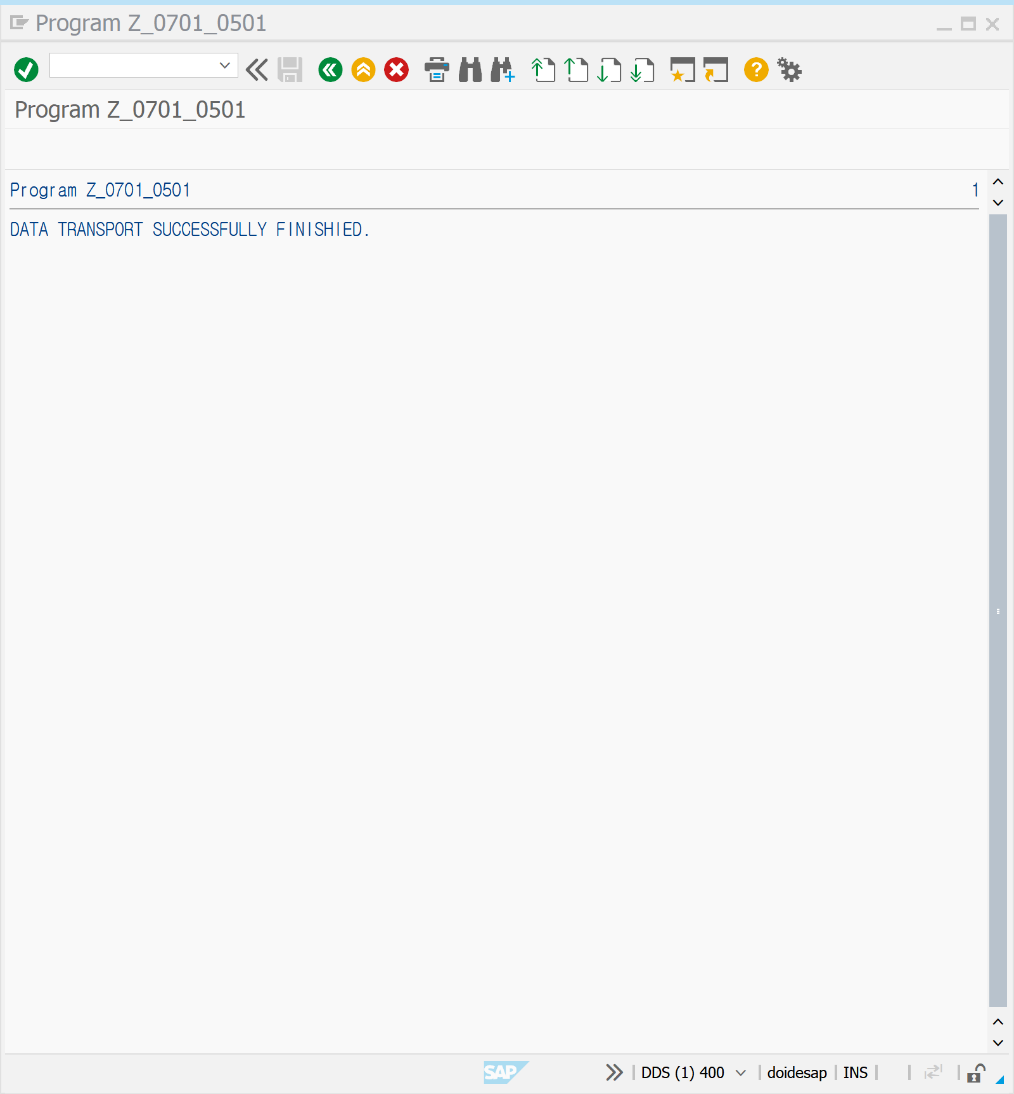




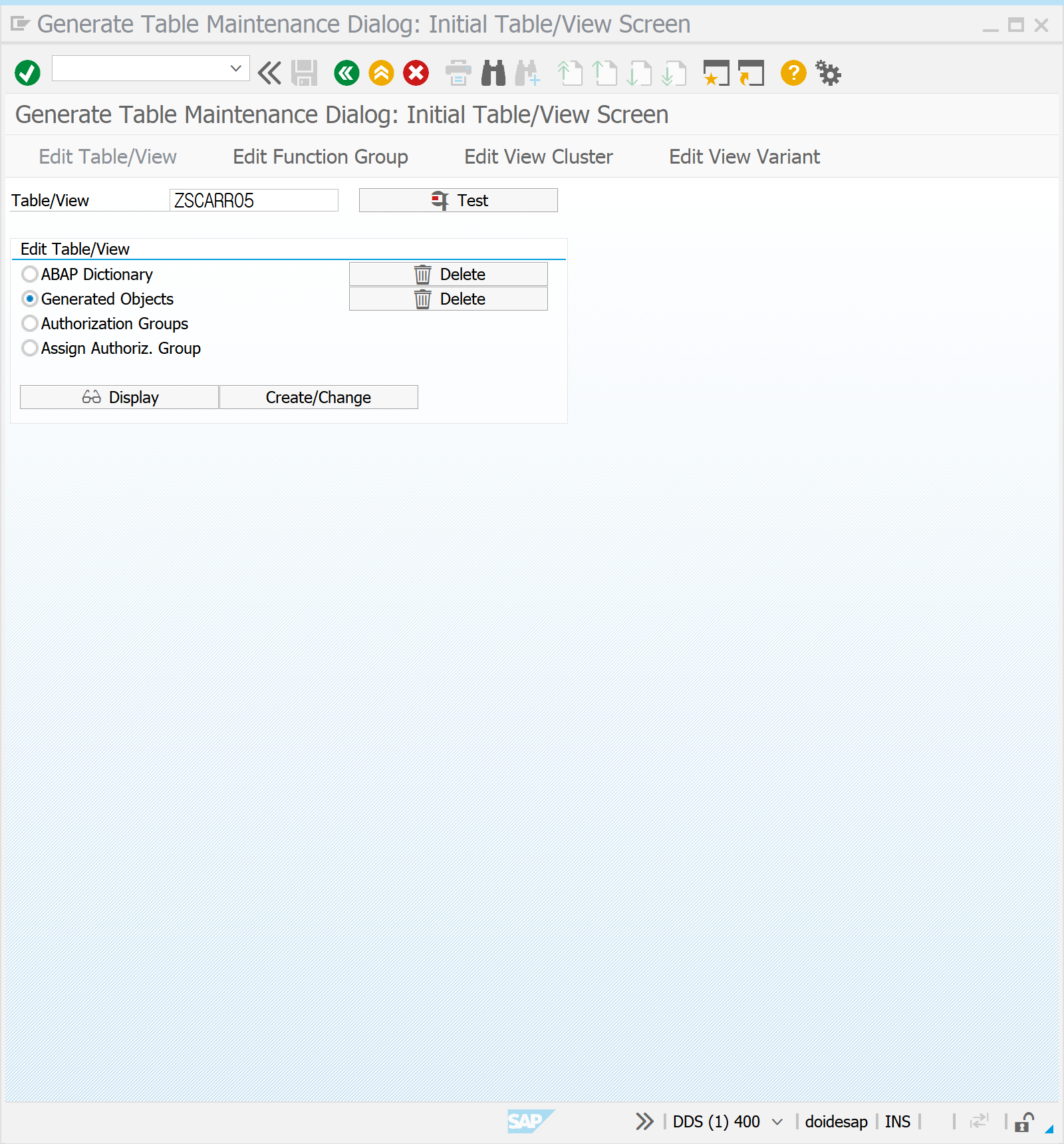


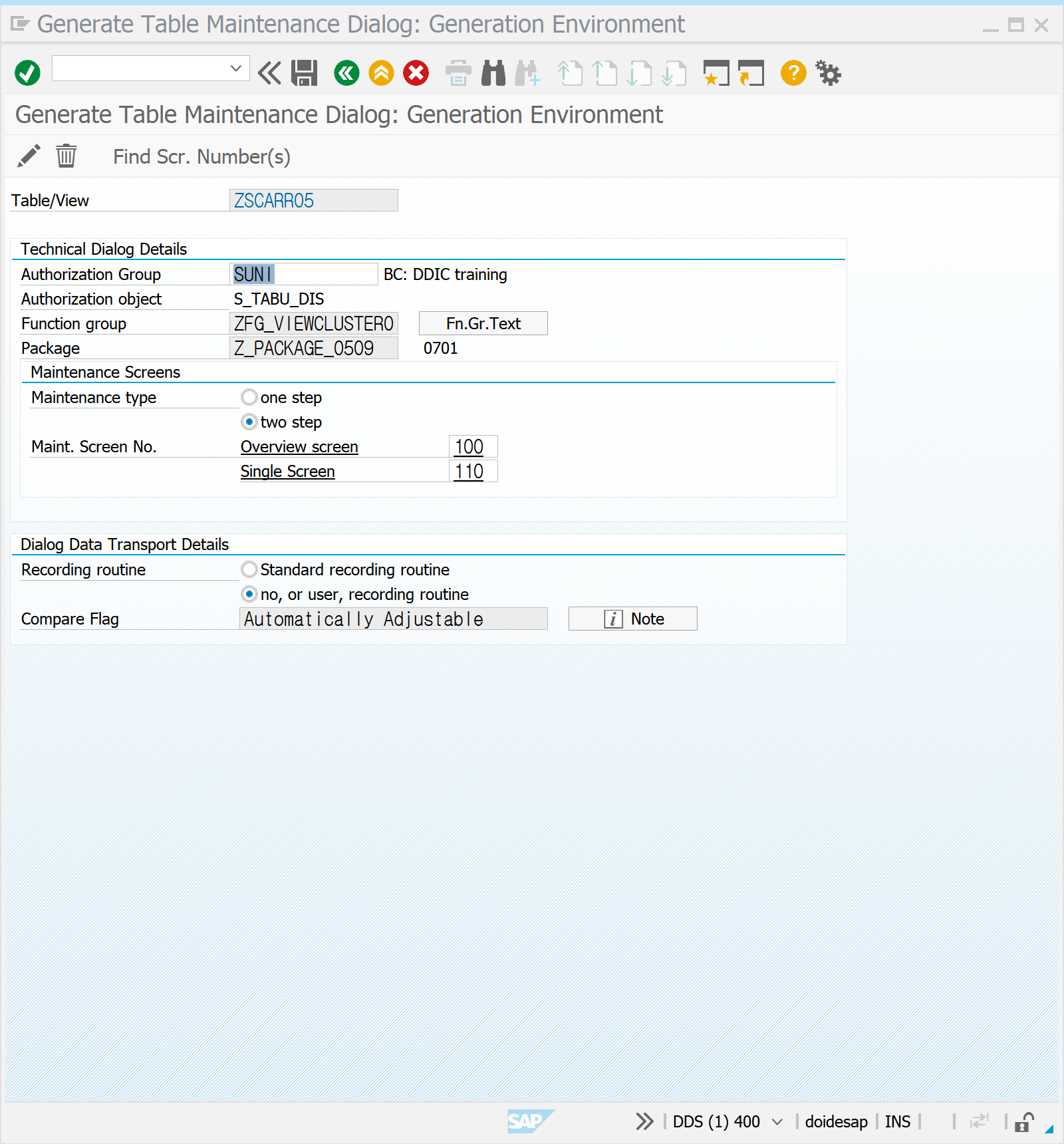
SE80 | Z\_0701\_0501

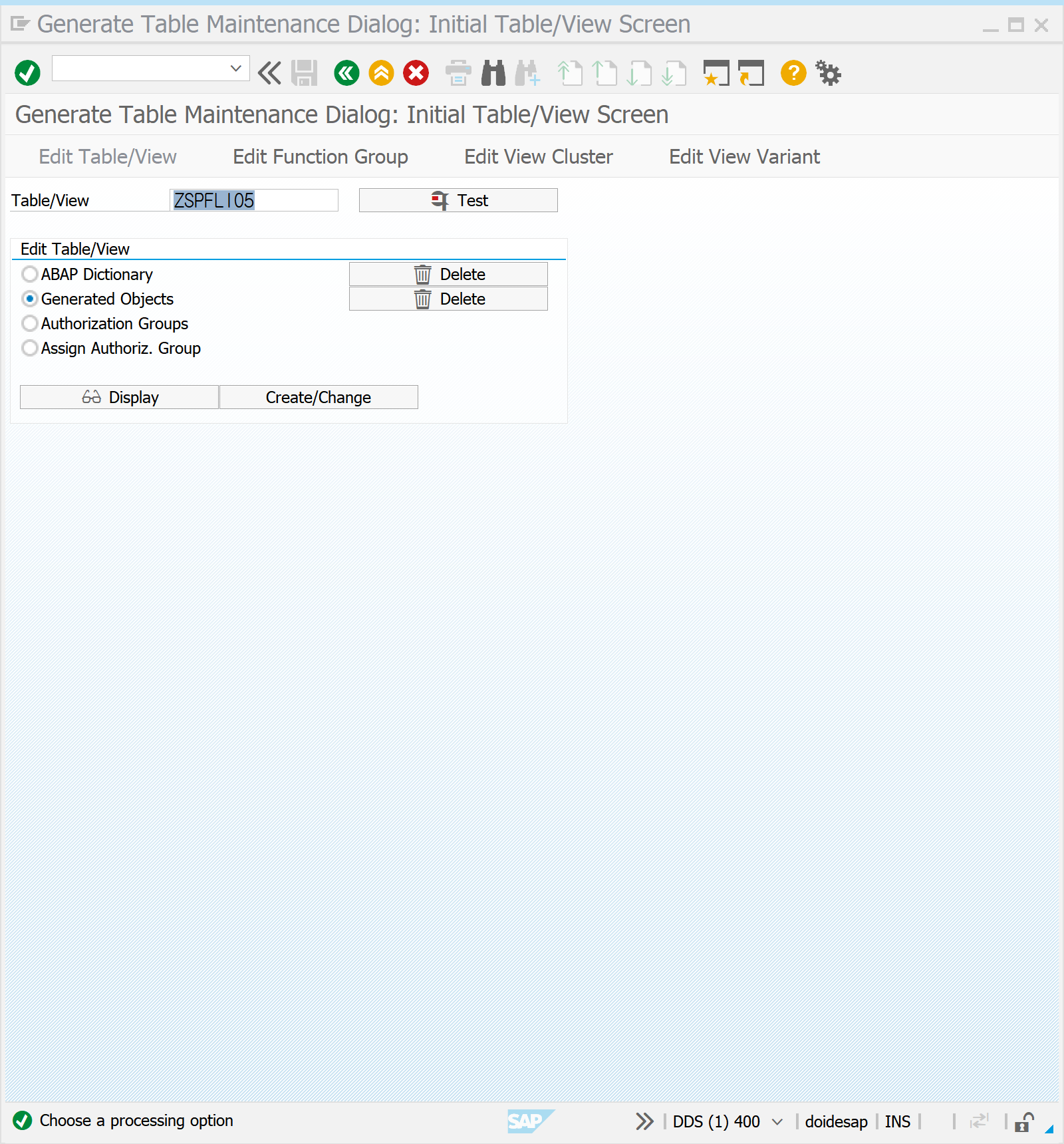
DATA WA\_SCARR TYPE SCARR.  
DATA WA\_SPFLI TYPE SPFLI.  
DATA WA\_FLIGHT TYPE SFLIGHT.  
  
DATA MY\_ERROR TYPE I VALUE 0.  
  
START-OF-SELECTION.  
  
  DELETE FROM ZSCARR05.  
  DELETE FROM ZSPFLI05.  
  DELETE FROM ZSFLIGHT05.  
  
  SELECT \*  
    FROM SCARR  
    INTO WA\_SCARR.  
    INSERT INTO ZSCARR05 VALUES WA\_SCARR.  
  ENDSELECT.  
  
  IF SY-SUBRC = 0.  
    SELECT \*  
      FROM SPFLI  
      INTO WA\_SPFLI.  
      INSERT INTO ZSPFLI05 VALUES WA\_SPFLI.  
    ENDSELECT.  
  
    IF SY-SUBRC = 0.  
      SELECT \*  
        FROM SFLIGHT  
        INTO WA\_FLIGHT.  
        INSERT INTO ZSFLIGHT05 VALUES WA\_FLIGHT.  
        ENDSELECT.  
  
        IF SY-SUBRC <> 0.  
          MY\_ERROR = 1.  
        ENDIF.  
      ELSE.  
        MY\_ERROR = 2.  
      ENDIF.  
    ELSE.  
      MY\_ERROR = 3.  
    ENDIF.  
  
    IF MY\_ERROR = 0.  
      WRITE:/ 'DATA TRANSPORT SUCCESSFULLY FINISHIED.'.  
    ELSE.  
      WRITE:/ 'ERROR:', MY\_ERROR.  
    ENDIF.

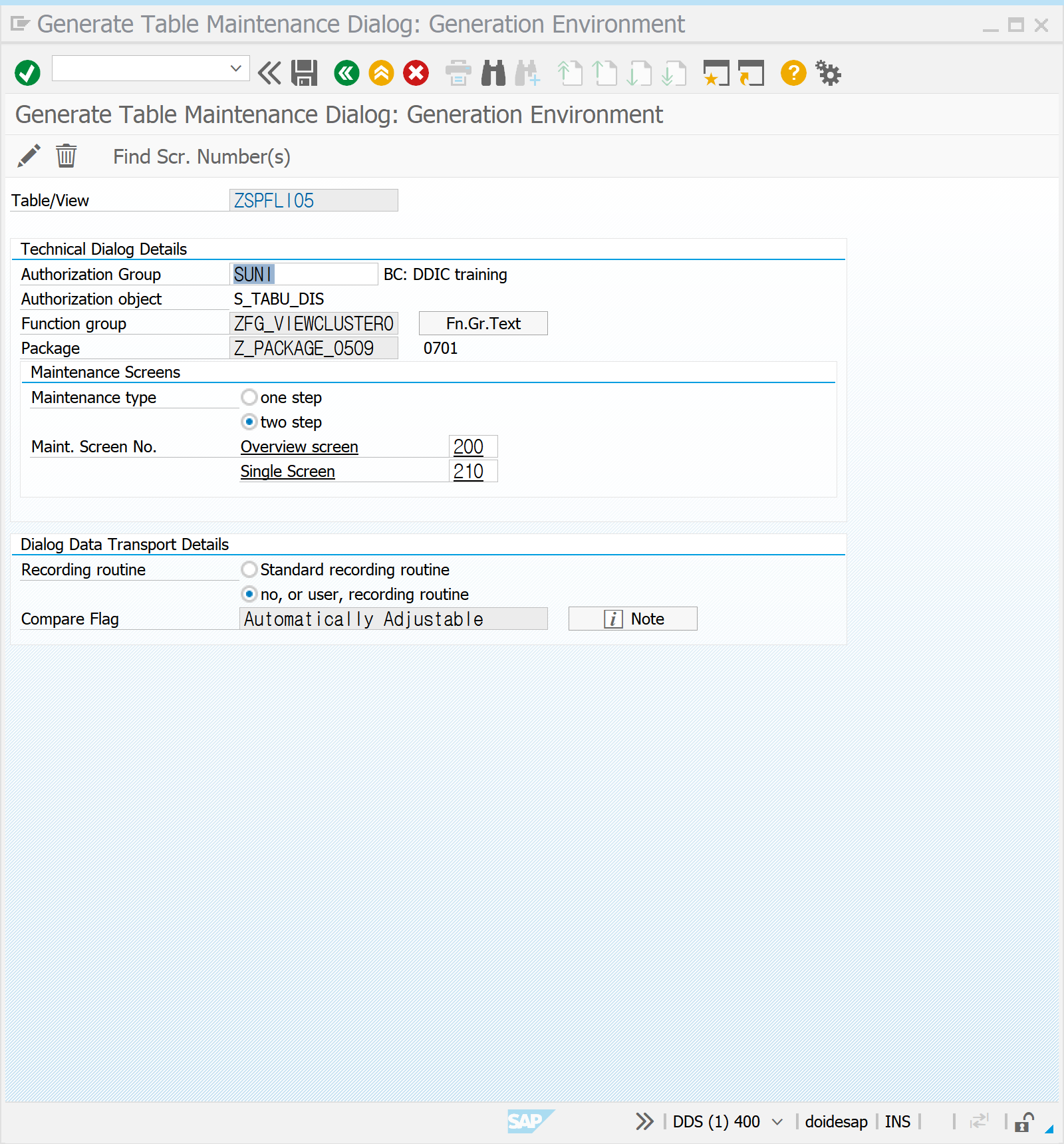


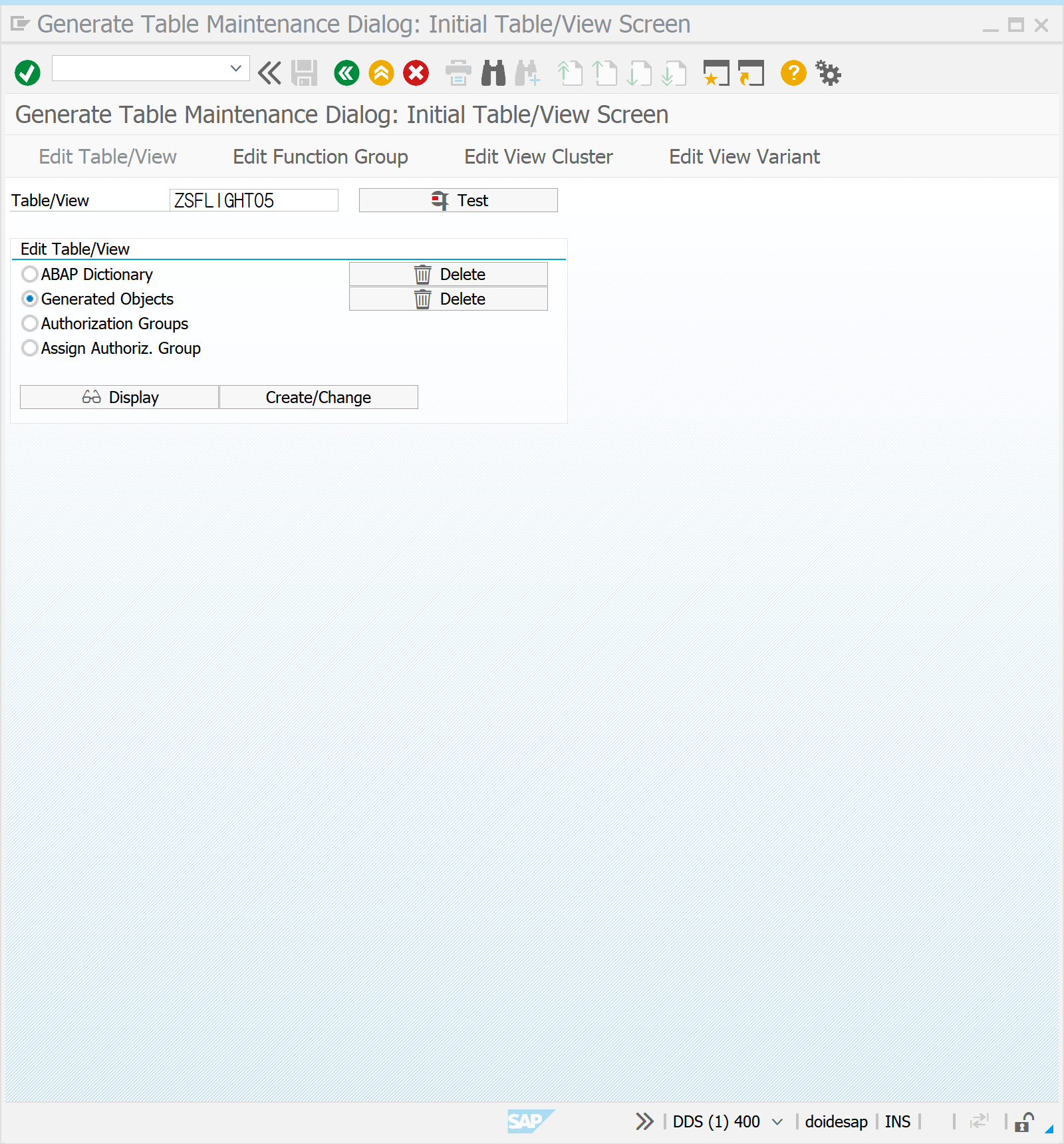
SE54 | MAINTANANCE DIALOG

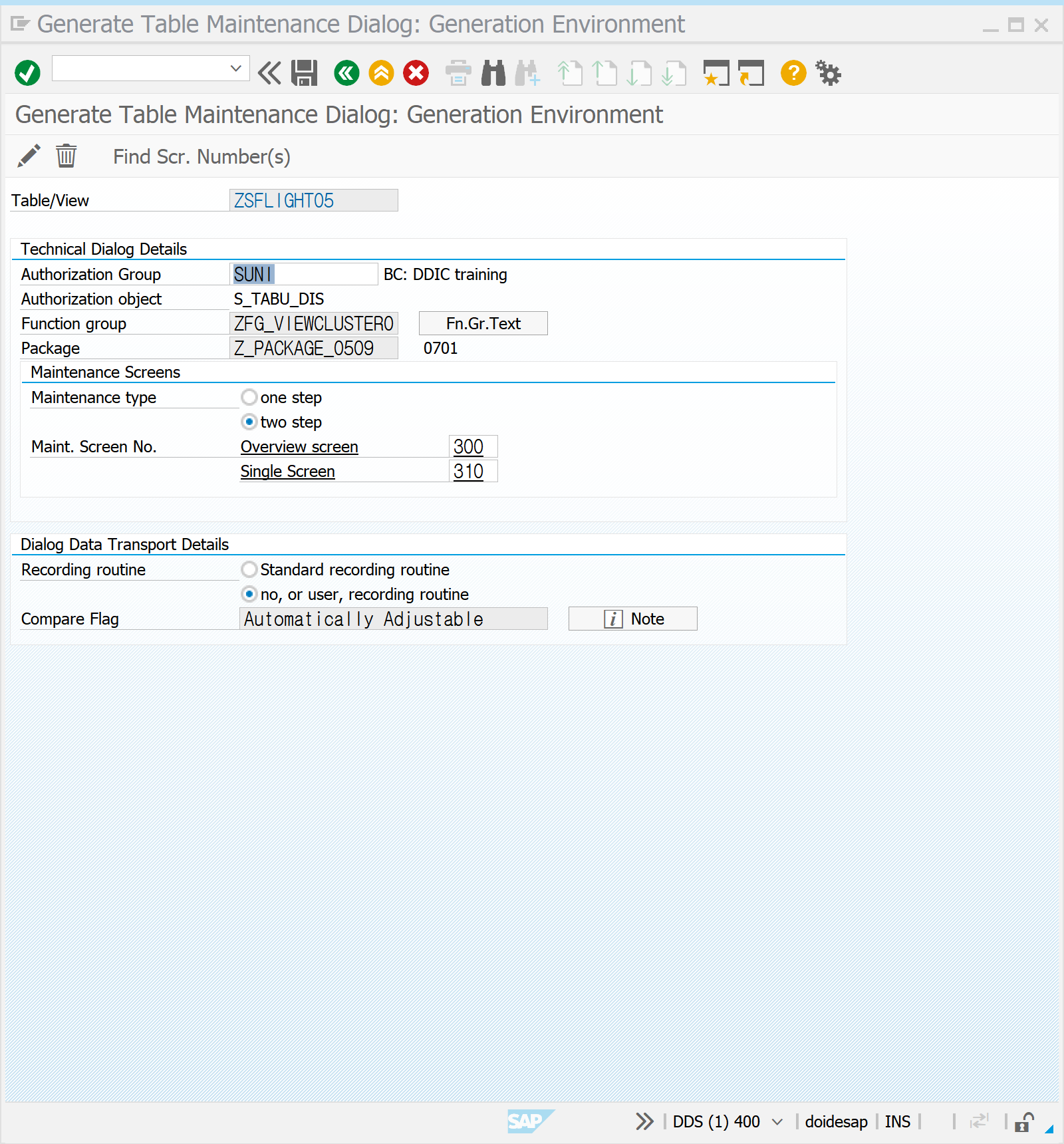




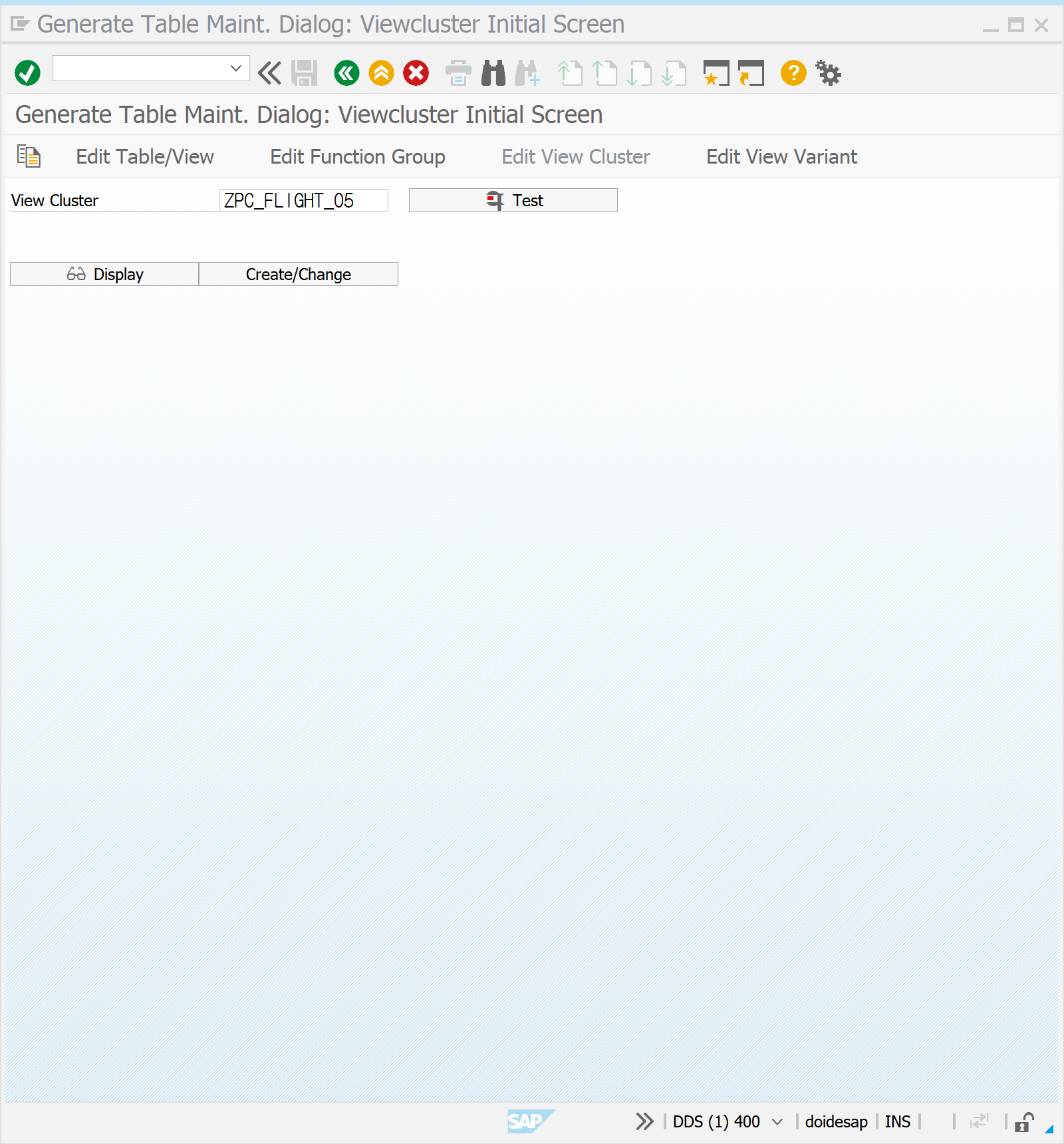


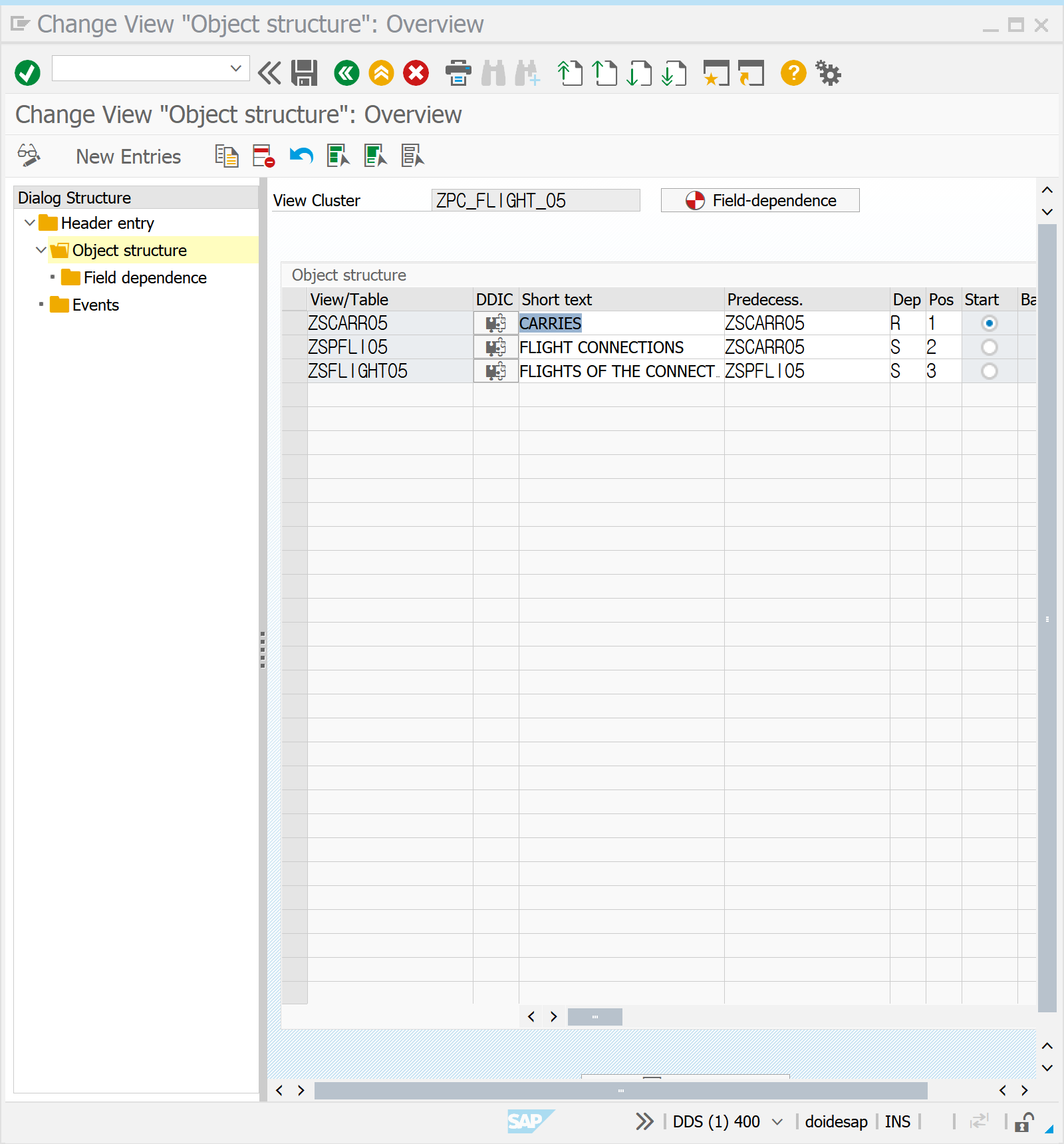


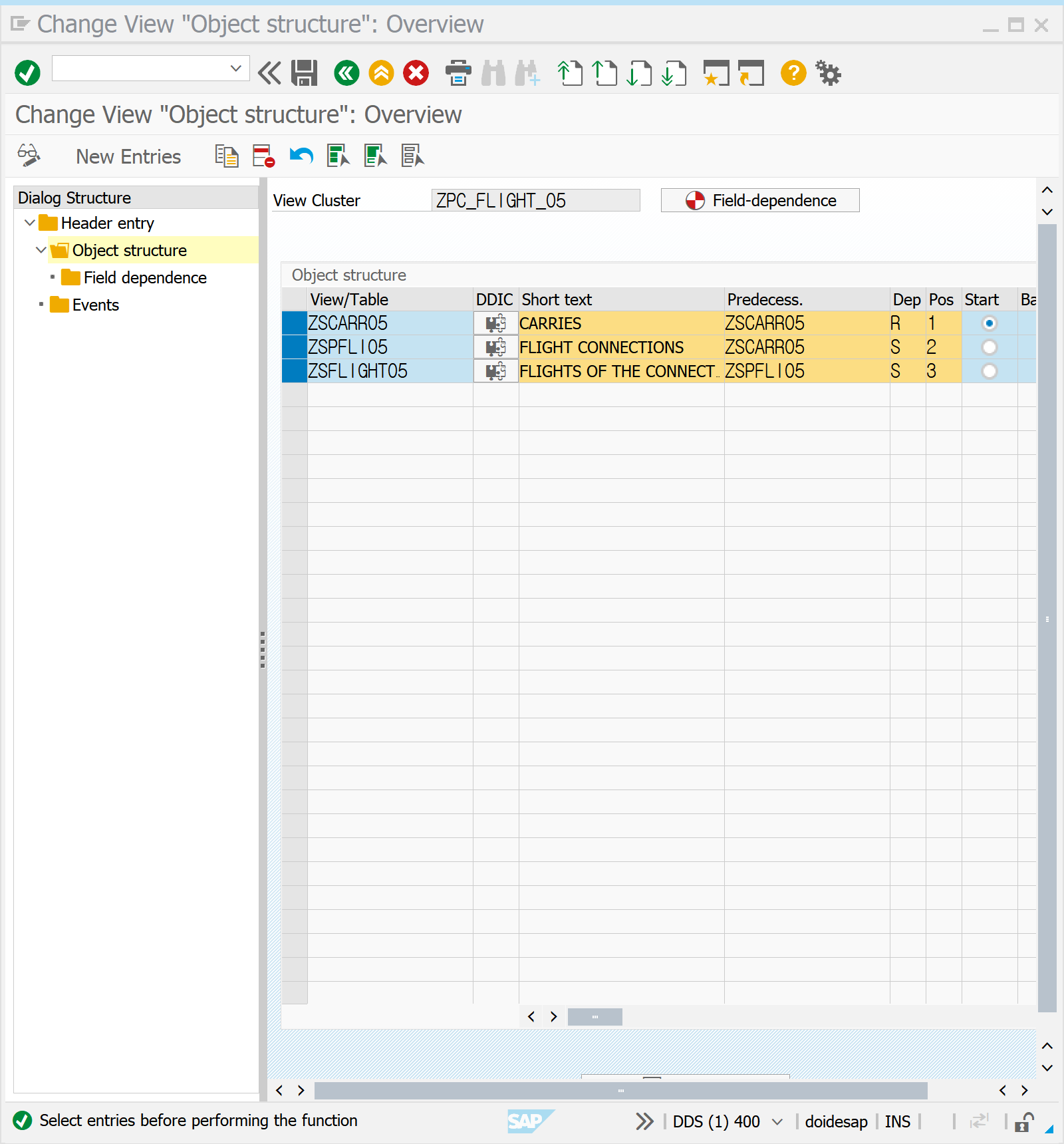


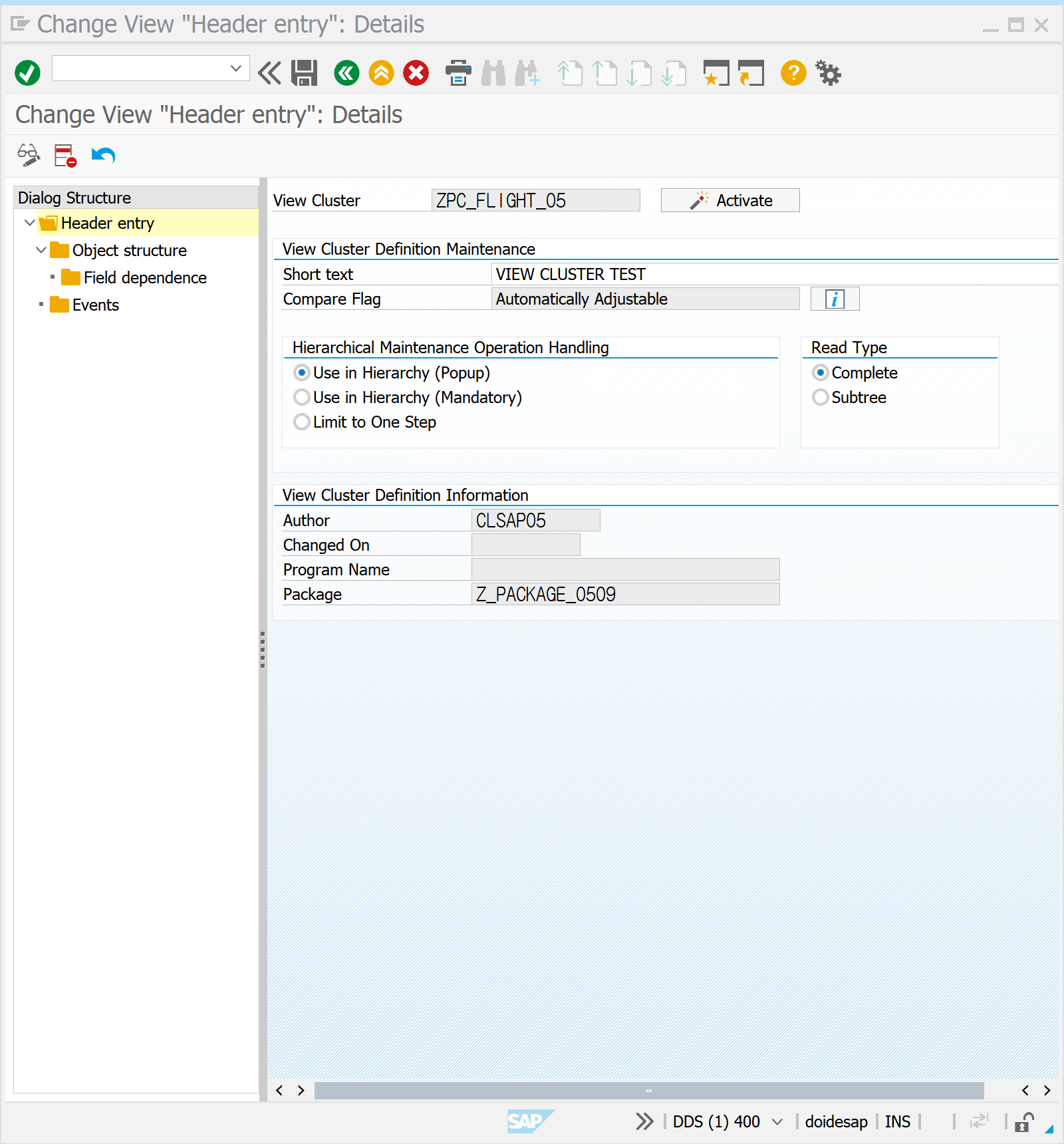


SE54 | CREATE VIEW CLUSTER









SM34 | VIEW CLUSTER

