INTRODUCCION A LA PROGRAMACION SAP…

ABAP PASO A PASO

Tabla de contenido

INTRODUCCION- NETWEAVER

Primeras palabras

El propósito de este Paso a Paso de Abap es introducirlos en el mundo SAP y la programación Abap. No pretende, ni mucho menos, ser un completo manual de referencia, sino, darles las primeras herramientas básicas, para aprender, de forma totalmente practica, a programar en SAP. Destacamos que las introducciones teóricas son breves, priorizando los paso a paso prácticos, con las imágenes que acompañan cada proceso. Se puede ahondar con tutoriales y manuales que encontramos en internet, manuales oficiales de SAP, etc. Y posiblemente, en un futuro cercano, saquemos una segunda parte más avanzada de este curso.

Introducción

Generalidades de SAP

SAP pertenece a una empresa alemana, y su significado en castellano es “Sistemas, Aplicaciones y Productos en tratamiento de Datos”.

El sistema SAP consta de varios módulos. Algunos de ellos son:

FI: Finanzas

SD: Ventas y Distribución

MM: Gestión de Materiales

PP: Gestión de Producción

WF: WorkFlow

HR: Planificación y Recursos Humanos.

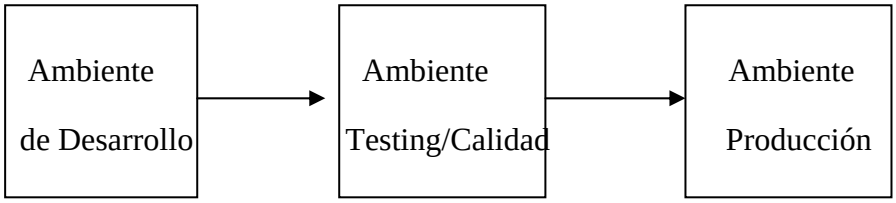
Consultores SAP

Básicamente en SAP encontramos tres tipos de consultores:

- Basis: encargados de la instalación y administración del sistema
- Funcionales: encargados de realizar los relevamientos, blueprints, parametrizaciones y especificaciones funcionales para que los programadores realicen los desarrollos correspondientes.
- Programadores: Realizan la programación abap.

Estructura SAP

En gral, los sistemas SAP R/3 tienen la siguiente estructura:



Ambiente de Desarrollo: Aquí se realizan los desarrollos y parametrizaciones del sistema. Al realizar un nuevo desarrollo, se genera una orden de transporte. Mediante la misma, el desarrollo pasará a los demás ambientes.

Ambiente de Testing/Calidad: Los objetos generados en Desarrollo pasan a este ambiente por medio de la orden de transporte. En este ambiente, se realizan las pruebas integrales para verificar el correcto funcionamiento de los programas y parametrizaciones.

Ambiente Productivo: Aquí están los datos reales y es el ambiente con el que opera la compañía que posee el sistema SAP.

Adicionalmente, puede usarse un ambiente Sandbox para testeo de configuraciones y desarrollos.

Abap: el lenguaje de SAP

El lenguaje que se utiliza para programar en SAP es el ABAP.El Abap es un lenguaje gobernado por eventos, o sea, que existen eventos que condicionan la secuencia de ejecución de los programas. Abap es propietario de SAP, y no es un lenguaje en el que, como en otros (Visual Basic, C, etc.), podemos realizar un ejecutable para correr en cualquier equipo.Los programas son ejecutables dentro del ambiente SAP.

Netweaver: nuestro ambiente de desarrollo

Para programar, necesitamos conectarnos a un servidor SAP. Afortunadamente, en la actualidad SAP nos provee un entorno para poder instalar en una pc y así tener una suerte de servidor ‘minisap’. Si bien en el mismo no encontraremos los diversos módulos que se encuentran en una instalación real (SD,MM,FI, etc), este entorno nos provee lo necesario para aprender a programar y dar los primeros pasos en ABAP.

Netweaver: instalación

Para ello, instalaremos el Netweaver trial, que acompaña este DVD y es el que puede descargarse gratuitamente del sitio de SAP.

Al ejecutar el archivo 'start', se abre una página web con un completo instructivo y los requerimientos de hard y soft para realizar la instalación.

Para acceder al sistema, debemos loguearnos en el mismo, mediante una interfase de usuario llamada Sap Graphical User Interfase (SAPGUI).

Entonces, en primer lugar realizaremos la instalación del netweaver.

Nota: En WinXP funciona correctamente, no tengo referencia si lo hace en otros sistemas operativos. Si se tiene en la pc el Windows Vista, se recomienda realizar la instalación del netweaver sobre una virtual machine. Por cualquier duda o consulta, pueden escribirme a om.abap@gmail.com.

Una vez realizada la instalación y configurado el saplogon (siguiendo el instructivo del instalador), iniciaremos sesión en el sistema.En instalaciones reales, iniciando sesión desde nuestra pc, ingresaremos a un sistema SAP que se encuentra en un servidor, pero en este caso, el cliente y el servidor quedan montados sobre la misma pc.

Inicio de sesión

Datos de login:

Mandante: es un área de trabajo dentro de un ambiente de SAP. Dentro del ambiente de desarrollo, podemos tener distintos mandantes: uno para desarrollo, otro para pruebas, otro para customizing, etc. En el caso de nuestra instalación netweaver, dejamos el valor que viene por defecto.

Usuario: en nuestro caso, ingresamos 'bcuser'.

Contraseña: en nuestro caso, ingresamos 'minisap'.

Idioma: SAP provee diversos idiomas para ingresar al sistema. En nuestro caso, lo dejamos en blanco, y nos logueamos por defecto en ingles.

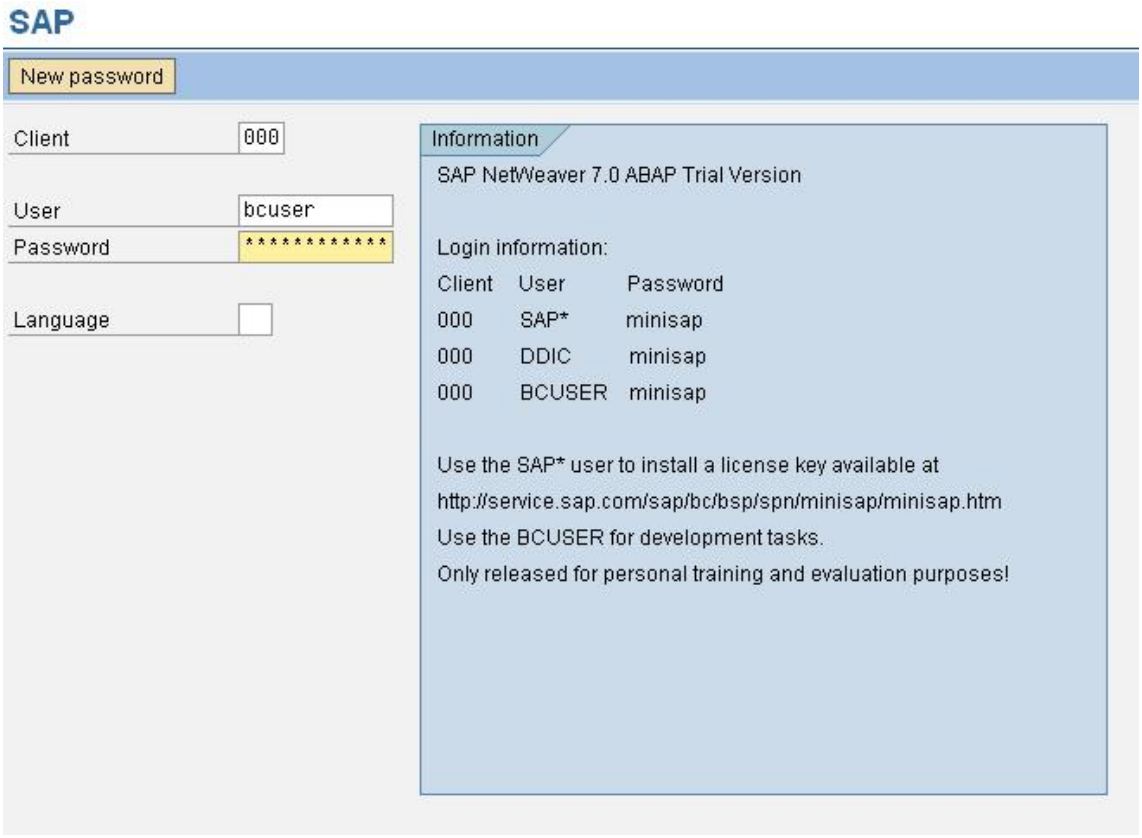


FIG. 1- PANTALLA DE LOGIN

Transporte

Cuando hablábamos de los distintos ambientes, decíamos que para pasar un desarrollo, parametrizacion, corrección, etc, de uno a otro (ej., del ambiente de desarrollo al de testing), lo hacíamos por medio de una orden de transporte. En las instalaciones reales, es siempre así, pero en nuestro caso, tenemos un solo ambiente, y no generaremos nunca orden de transporte. Volveremos a este tema cuando generemos nuestro primer programa.

Inicio

Una vez que ingresamos al sistema, aparecerá una pantalla similar a esta:

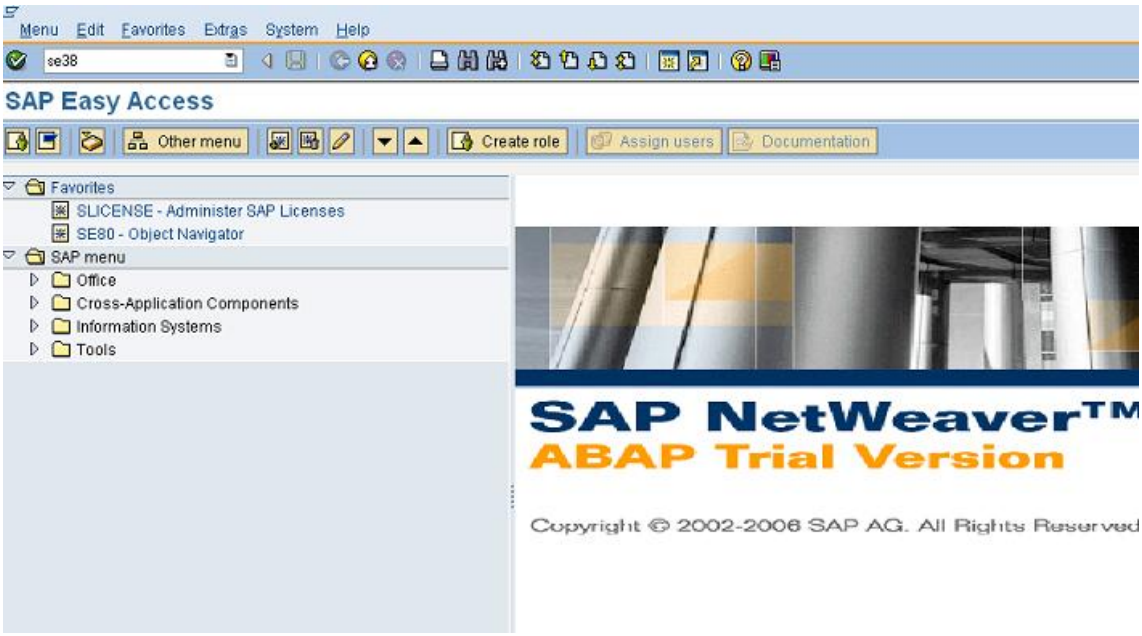


FIG.2- PANTALLA INICIAL

En SAP se opera por medio de **transacciones**. Existen muchas y diversas transacciones propias de SAP, y veremos más adelante, que podemos crearlas también nosotros. Existen transacciones para ingresar facturas, para ingreso y salida de mercaderías, para pedidos, etc. Nosotros veremos las que nos interesan para programar, y comenzaremos con una de las más importantes (junto con la del editor Abap), que es la del Diccionario de Datos: **SE11**.

Entonces, en el textbox que aparece a la derecha del botón  (en donde dice se38 en la fig. 2) escribiremos **se11** y presionaremos enter (o click en el botón de la izquierda).

DICCIONARIO DE DATOS

Transacción SE11-Diccionario de datos

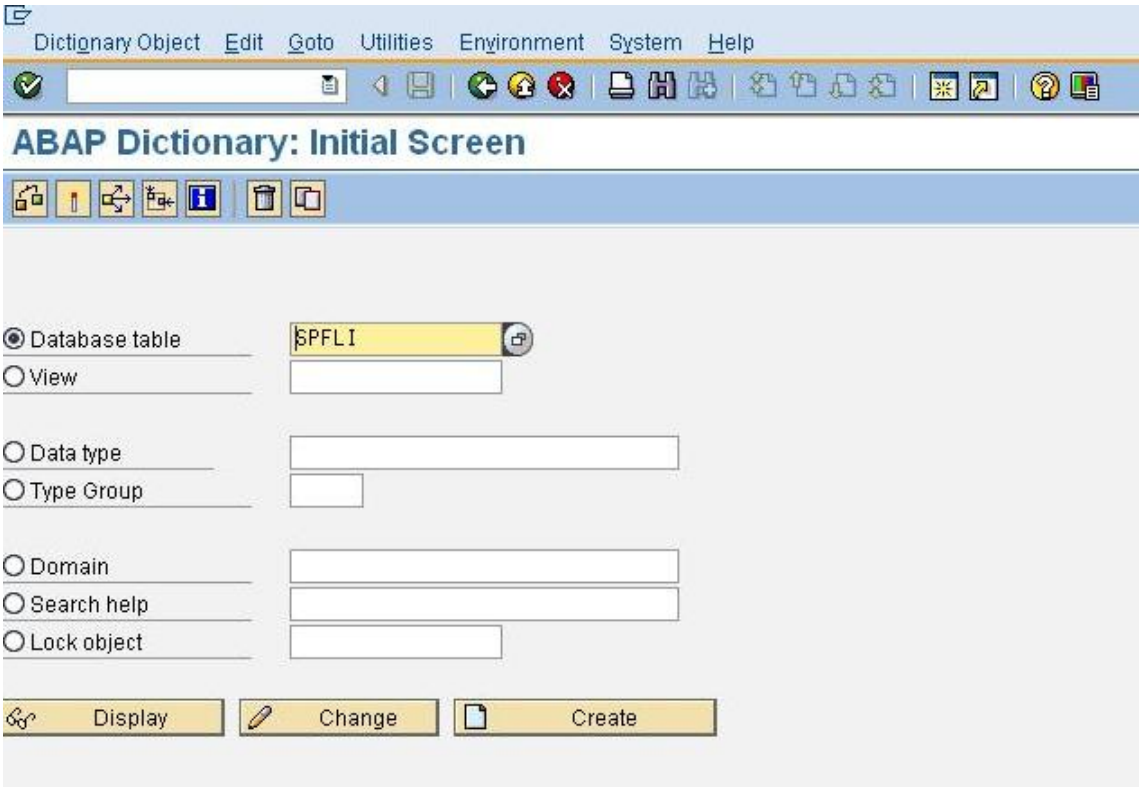


FIG. 3- DICCIONARIO DE DATOS

En esta transacción, tal como vemos en la figura 3, podemos gestionar tablas, vistas, tipos de datos, etc.

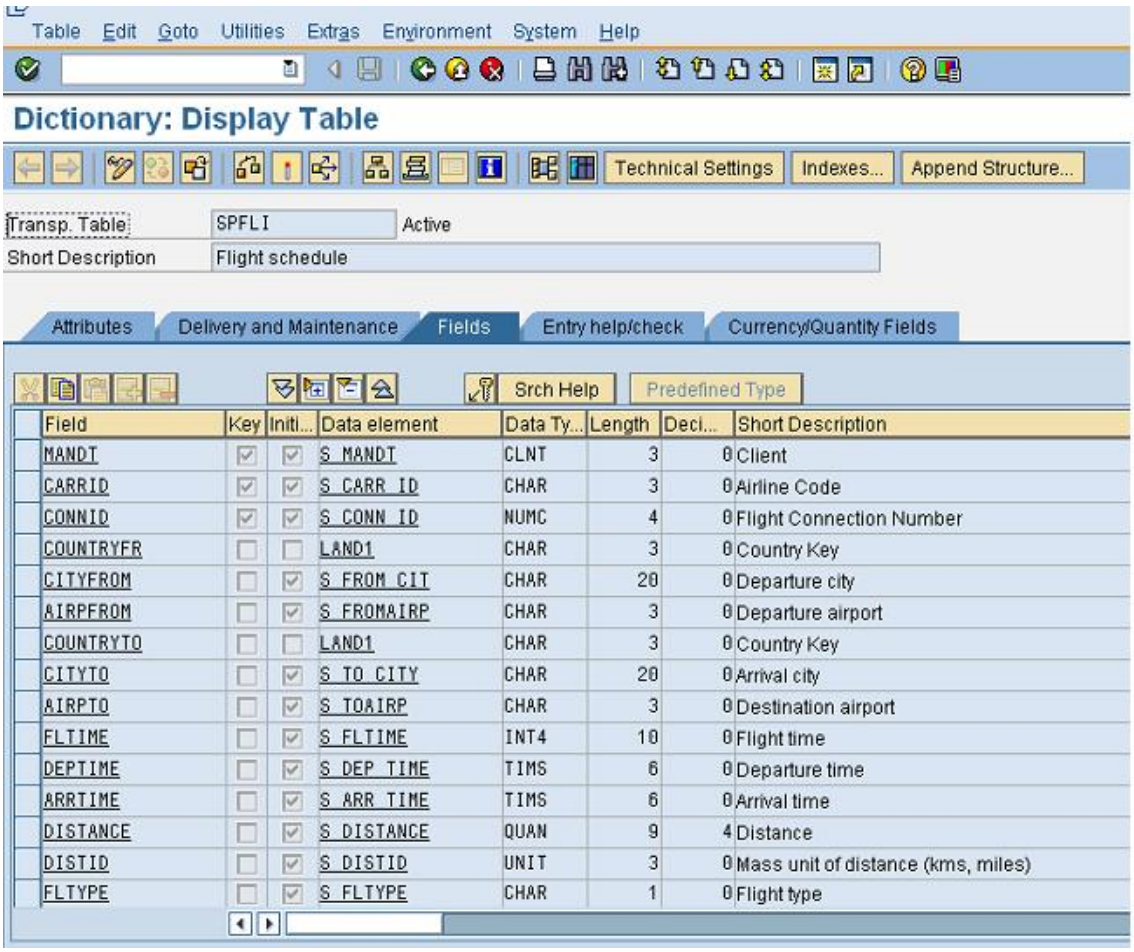
Comenzaremos viendo las tablas.

Aclaración importante: en SAP, cuando creamos objetos propios (tablas, estructuras, programas, transacciones, etc.) el nombre de los mismos SOLO puede comenzar con Y o Z (ej. Creamos el programa ZPROGRAM). Si probamos crear un programa con el nombre PROGRAM, el sistema emitirá un mensaje de error.

Modelo de vuelos

Con fines didácticos, SAP provee unas tablas de un sistema de vuelos que ya vienen creadas y con datos en nuestro netweaver. Comencemos entonces a visualizar dichas tablas.

Como muestra la fig.3, seleccionaremos con el radiobutton ‘Database table’, y en el nombre, SPFLI. Hacemos click en el botón display, y el sistema nos llevará a una nueva pantalla, en donde podremos visualizar la tabla de conexiones de vuelos.



Field	Key	Initi...	Data element	Data Ty...	Length	Deci...	Short Description
MANDT	☑	☑	S_MANDT	CLNT	3	0	Client
CARRID	☑	☑	S_CARR_ID	CHAR	3	0	Airline Code
CONNID	☑	☑	S_CONN_ID	NUMC	4	0	Flight Connection Number
COUNTRYFR	☐	☐	LAND1	CHAR	3	0	Country Key
CITYFROM	☐	☑	S_FROM_CIT	CHAR	20	0	Departure city
AIRPFROM	☐	☑	S_FROMAIRP	CHAR	3	0	Departure airport
COUNTRYTO	☐	☐	LAND1	CHAR	3	0	Country Key
CITYTO	☐	☑	S_TO_CITY	CHAR	20	0	Arrival city
AIRPTO	☐	☑	S_TOAIRP	CHAR	3	0	Destination airport
FLTIME	☐	☑	S_FLTIME	INT4	10	0	Flight time
DEPTIME	☐	☑	S_DEP_TIME	TIMS	6	0	Departure time
ARRTIME	☐	☑	S_ARR_TIME	TIMS	6	0	Arrival time
DISTANCE	☐	☑	S_DISTANCE	QUAN	9	4	Distance
DISTID	☐	☑	S_DISTID	UNIT	3	0	Mass unit of distance (kms, miles)
FLTYPE	☐	☑	S_FLTYPE	CHAR	1	0	Flight type

FIG.4- TABLA SPFLI

En esta pantalla podemos observar, entre otros datos, el nombre de los campos, tipo de datos, longitud, descripción breve.

Para observar el contenido de la tabla, hacemos click en el botón  , que se encuentra a la izquierda del botón Technical Settings. Esto nos lleva a la pantalla que muestra la figura 5.

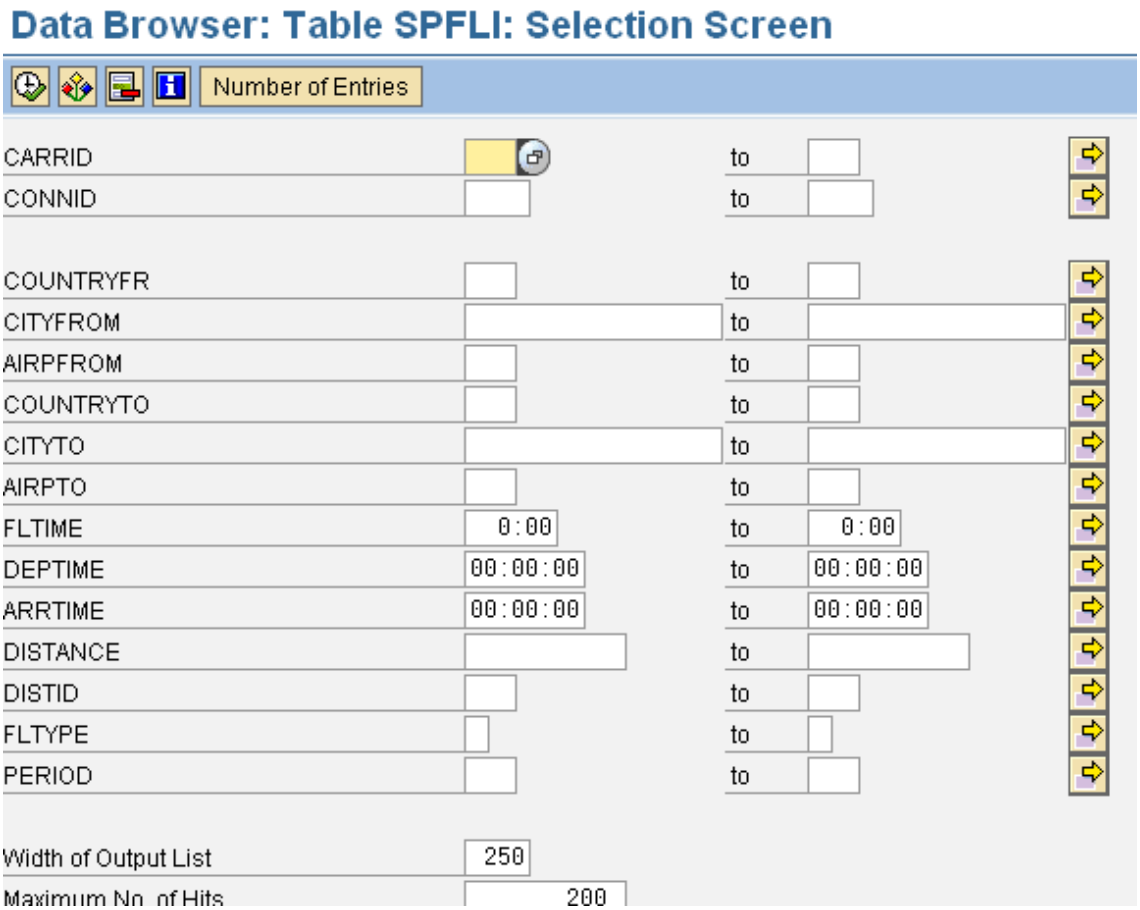


FIG. 5- PANTALLA DE SELECCIÓN DE LA TABLA SPFLI

Vemos una pantalla de selección, donde podemos llenar algunos campos para filtrar la selección, o dejarlos todos en blanco, y así visualizaremos todos los datos de la tabla.

Por ejemplo, para ingresar algún valor en el campo CARRID, podemos tipearlo directamente, o hacer click en el círculo que aparece a la derecha, pegado al cuadro que vemos en la fig 5 pintado de amarillo. Se desplegará una pantalla como la que podemos observar en la fig. 6

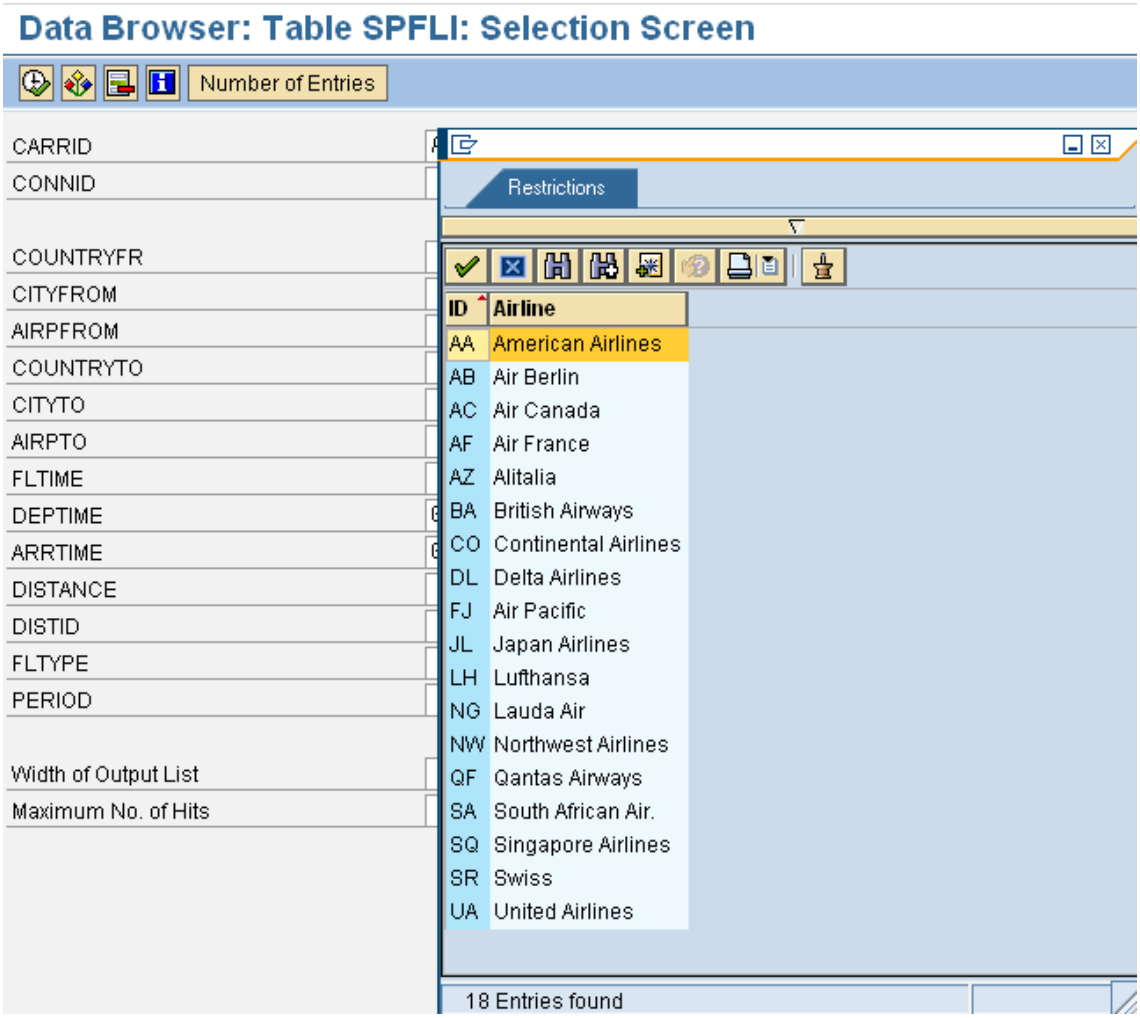


FIG.6- SELECCIÓN DE CARRID

Hacemos doble click en el valor deseado.

Luego presionamos el botón  y visualizaremos el contenido de la tabla.

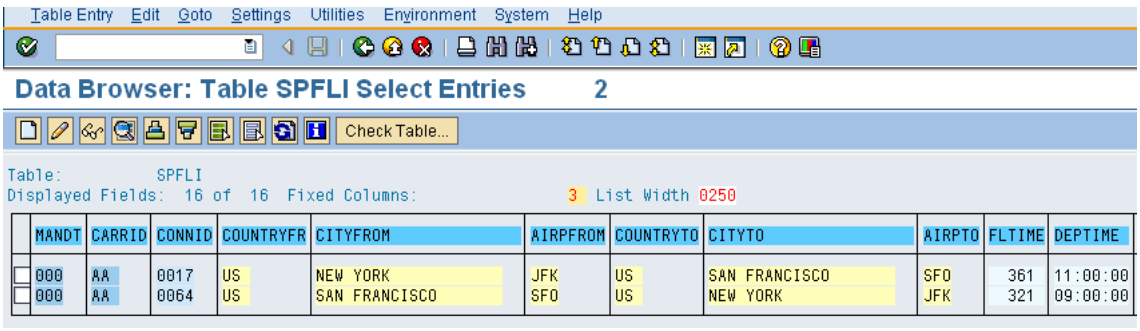
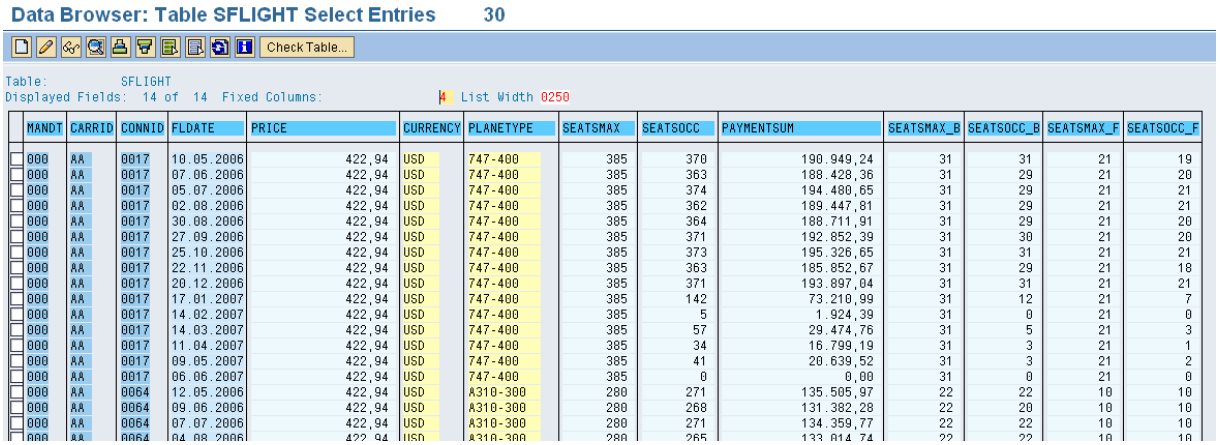


FIG.7- CONTENIDO DE LA TABLA SPFLI

En cualquier transacción, siempre que presionamos el botón  , navegamos a una pantalla anterior. Presionemos tres veces dicho botón, y visualicemos la tabla de vuelos SFLIGHT. Con esto, vamos familiarizándonos con las tablas que utilizaremos en los siguientes ejercicios.



Data Browser: Table SFLIGHT Select Entries 30

Check Table...

Table: SFLIGHT
Displayed Fields: 14 of 14 Fixed Columns: 4 List Width 0250

MANDT	CARRID	CONNID	FLDATE	PRICE	CURRENCY	PLANETYPE	SEATSMAX	SEATSOCC	PAYMENTSUM	SEATSMAX_B	SEATSOCC_B	SEATSMAX_F	SEATSOCC_F
000	AA	0017	10.05.2006	422,94	USD	747-400	385	370	190.949,24	31	31	21	19
000	AA	0017	07.06.2006	422,94	USD	747-400	385	363	188.428,36	31	29	21	20
000	AA	0017	05.07.2006	422,94	USD	747-400	385	374	194.480,65	31	29	21	21
000	AA	0017	02.08.2006	422,94	USD	747-400	385	362	189.447,81	31	29	21	21
000	AA	0017	30.08.2006	422,94	USD	747-400	385	364	188.711,91	31	29	21	20
000	AA	0017	27.09.2006	422,94	USD	747-400	385	371	192.852,39	31	30	21	20
000	AA	0017	25.10.2006	422,94	USD	747-400	385	373	195.326,65	31	31	21	21
000	AA	0017	22.11.2006	422,94	USD	747-400	385	363	185.852,67	31	29	21	18
000	AA	0017	20.12.2006	422,94	USD	747-400	385	371	193.897,04	31	31	21	21
000	AA	0017	17.01.2007	422,94	USD	747-400	385	142	73.210,99	31	12	21	7
000	AA	0017	14.02.2007	422,94	USD	747-400	385	5	1.924,39	31	0	21	0
000	AA	0017	14.03.2007	422,94	USD	747-400	385	57	29.474,76	31	5	21	3
000	AA	0017	11.04.2007	422,94	USD	747-400	385	34	16.799,19	31	3	21	1
000	AA	0017	09.05.2007	422,94	USD	747-400	385	41	20.639,52	31	3	21	2
000	AA	0017	06.06.2007	422,94	USD	747-400	385	0	0,00	31	0	21	0
000	AA	0064	12.05.2006	422,94	USD	A310-300	280	271	135.505,97	22	22	10	10
000	AA	0064	09.06.2006	422,94	USD	A310-300	280	268	131.382,28	22	20	10	10
000	AA	0064	07.07.2006	422,94	USD	A310-300	280	271	134.359,77	22	22	10	10
000	AA	0064	04.08.2006	422,94	USD	A310-300	280	265	133.014,74	22	22	10	10

FIG 8- CONTENIDO DE LA TABLA SFLIGHT

Veamos ahora el concepto de **modos.**Si estamos, por ejemplo, en la transacción SE11, visualizando una tabla, y queremos entrar al editor de Abap, podríamos navegar hacia atrás, hasta volver a la pantalla inicial, y colocar allí el nombre de la transacción en cuestión (SE38). Otra forma es, desde la pantalla actual, ingresando en el textbox correspondiente () **/n** seguido por el nombre de la transacción a la cual queremos ingresar. En nuestro caso, ingresaremos /nse38.Pero si no queremos abandonar la transacción actual, podemos abrir otra 'instancia', o sea, otro modo.O sea, tendremos abiertas las dos transacciones a la vez. Es similar a cuando abrimos dos instancias del navegador web para tener dos páginas abiertas a la vez. Esto podemos hacerlo de dos formas. Una es presionando el botón  . La otra forma es ingresando en la pantalla actual **/o** seguido del nombre de la transacción que deseamos abrir. En nuestro caso, /ose38.

PRIMER PROGRAMA

SE38-Editor Abap

COMENCEMOS A PROGRAMAR… EDITOR ABAP.

En primer lugar, abriremos el editor abap, con la transacción SE38.

Primer programa

Aparece la pantalla inicial del editor, como muestra la fig.9. Comenzaremos haciendo un programa abap, por lo que dejaremos seleccionado Source code, y en nombre de programa colocaremos Z_FIRST_PROG. Recordemos que los nombres de programa SIEMPRE deben comenzar con Y o Z.

Creación de programa

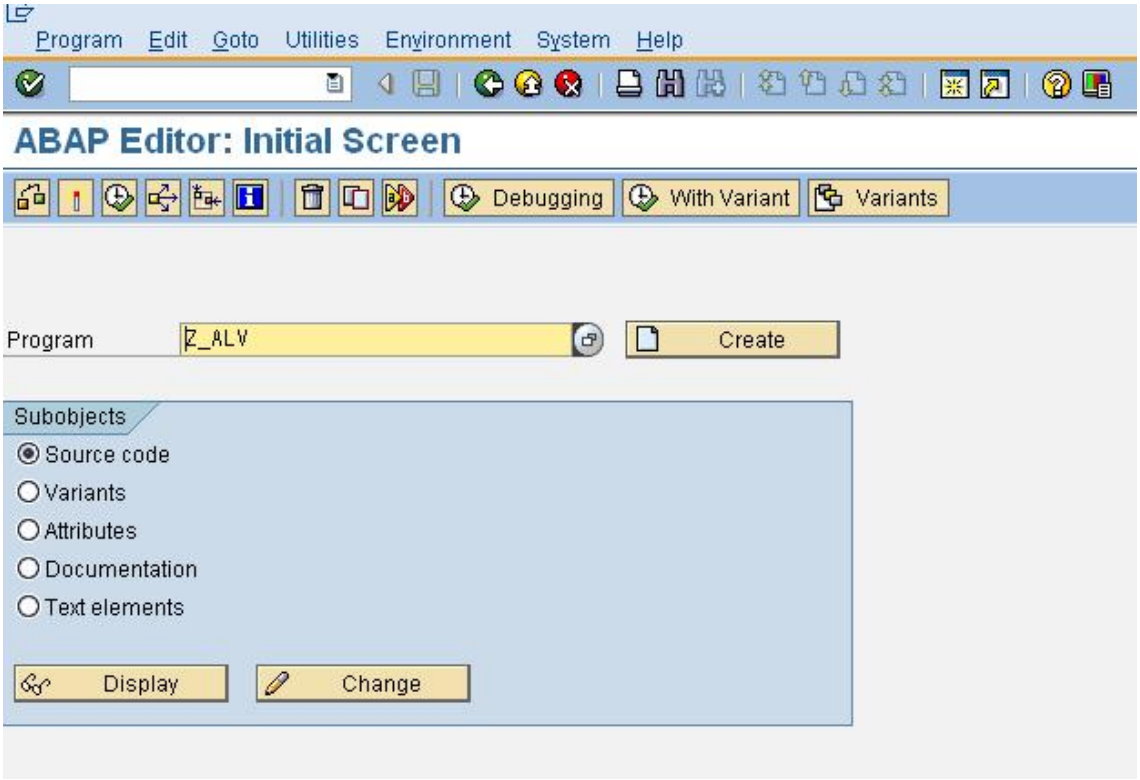


FIG. 9- EDITOR ABAP. PANTALLA INICIAL.

Una vez introducido el nombre, presionamos el botón Create. Aparecerá una ventana como se muestra en la fig. 10, en la cual deberemos ingresar los atributos del programa.

En título, podemos colocar una descripción cualquiera, ingresemos: 'Primer programa'.

Tal como se ve en la figura, en type debemos seleccionar ‘Executable program’.

Finalmente hacemos click en Save.

Ahora aparecerá una nueva ventana como se podemos observar en la fig. 11. Mencionábamos anteriormente, que en instalaciones reales, cuando realizamos un programa, creamos una orden de transporte, para poder pasar dicho programa posteriormente a los demás ambientes. Como nosotros vamos a trabajar acá con un solo ambiente, (y un solo mandante), presionaremos el botón ‘Local Object’. Esto significa que los objetos creados se grabarán en forma local (en nuestra pc) y nunca serán transportados a otros ambientes.

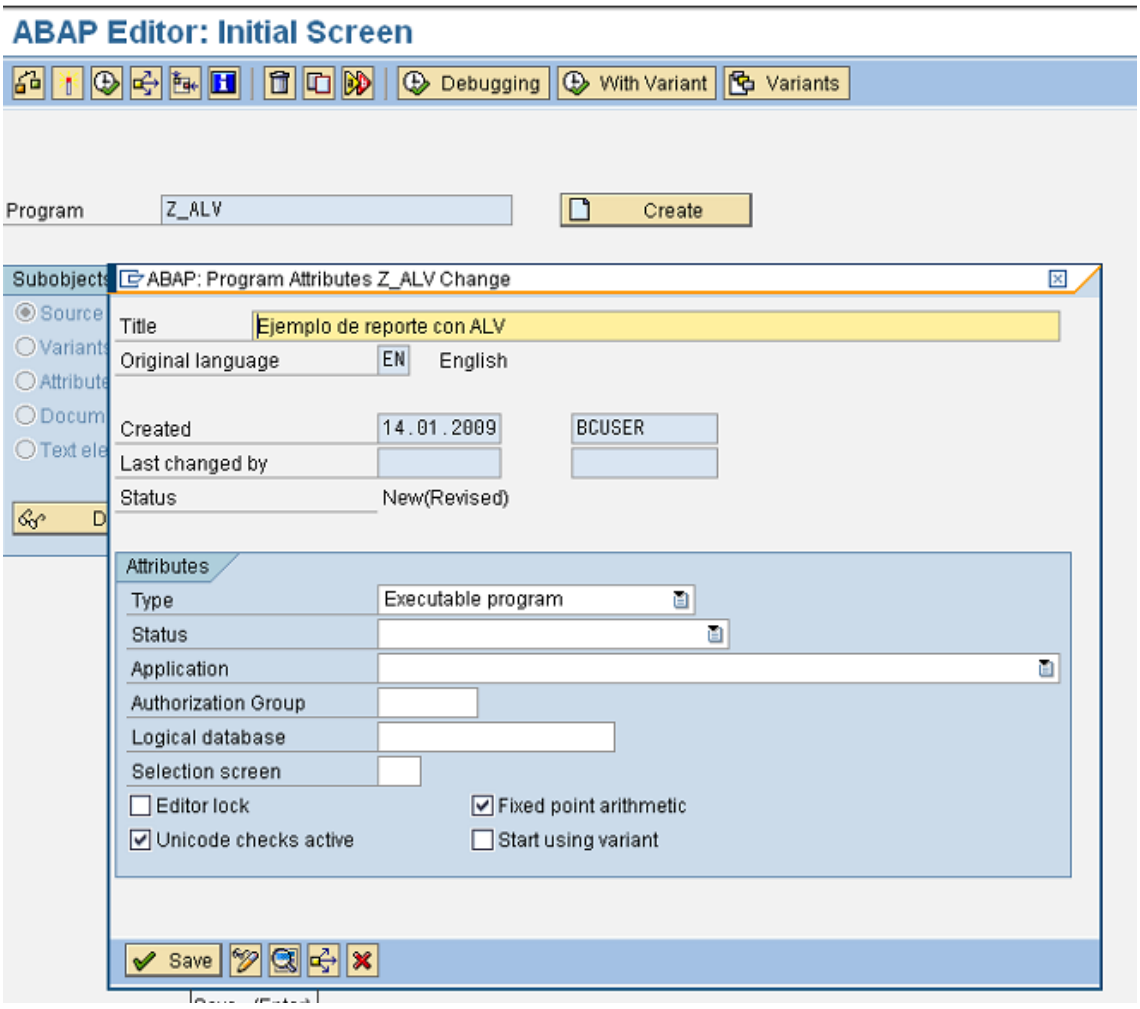


FIG. 10- ATRIBUTOS DEL PROGRAMA

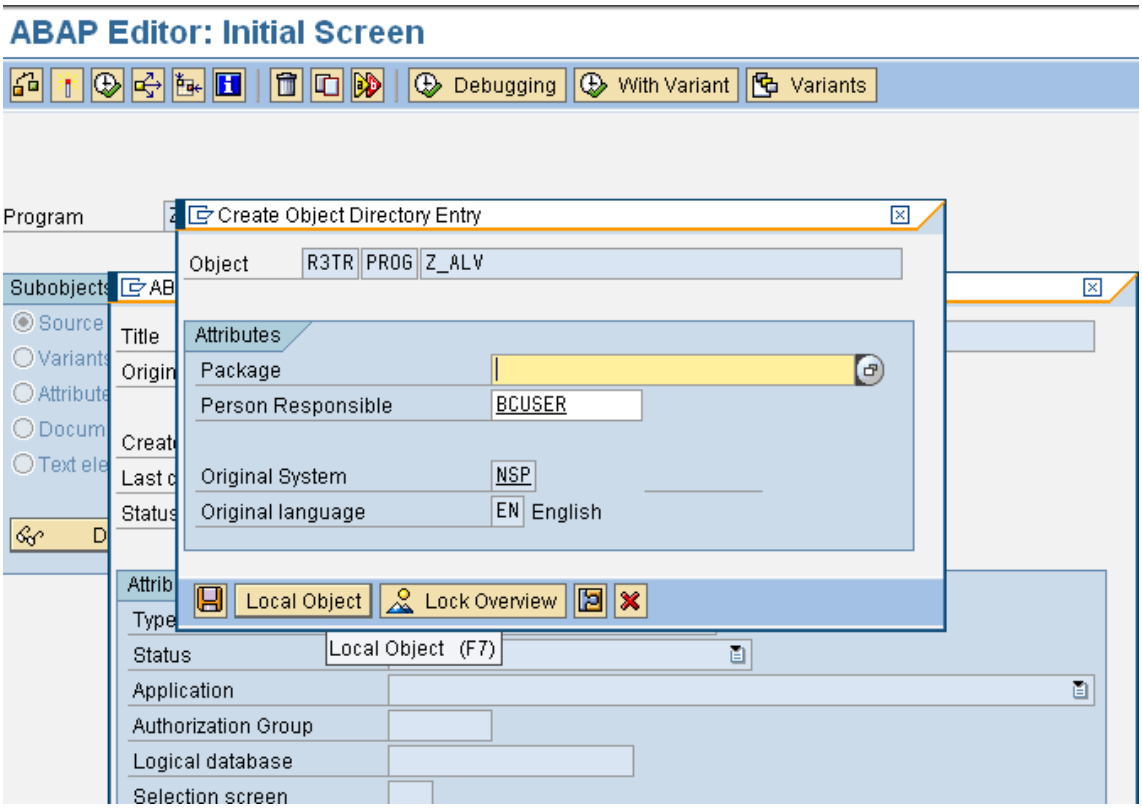


FIG.11- ATRIBUTOS DEL OBJETO

Hecho todo lo anterior, nos aparecerá el editor abap.

Upload de código

Podemos observar que ya aparece una cabecera y la palabra REPORT seguida del nombre que le hemos dado al programa en el paso anterior. Copiemos ahora el código de nuestro primer programa, para empezar a estudiar el mismo. Para ello haremos como indica la fig. 12. Hacemos click en utilities->more utilities->upload/download->upload.

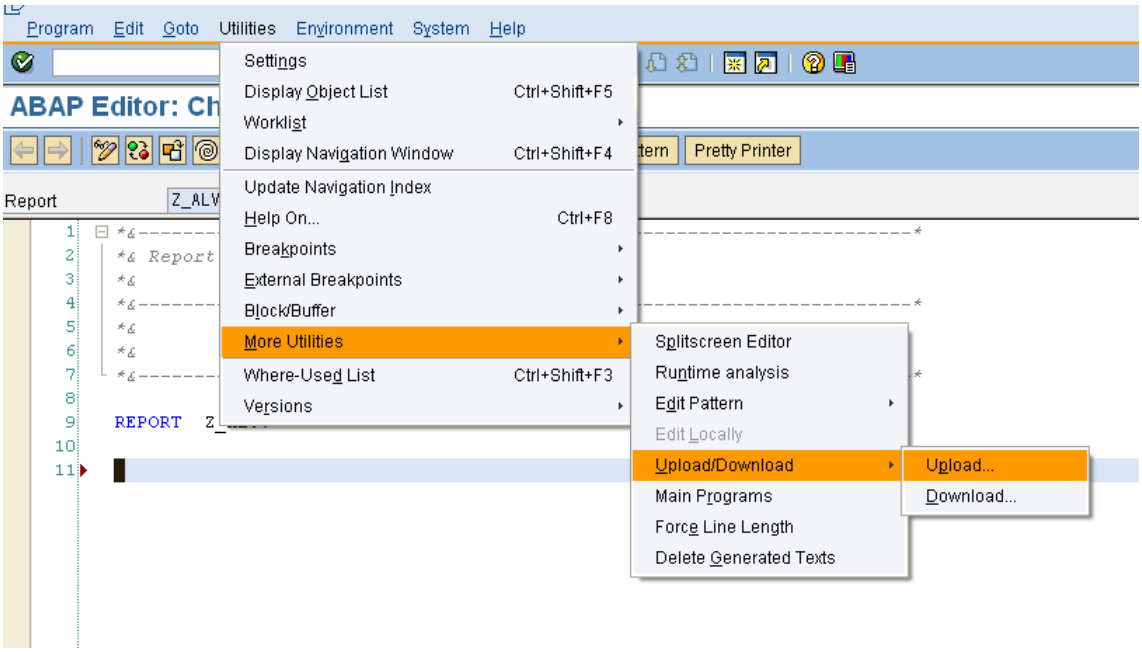


FIG.12- EDITOR ABAP

Seleccionamos el archivo Z_FIRST_PROG.TXT, de la carpeta EJEMPLOS del DVD, y presionamos abrir.

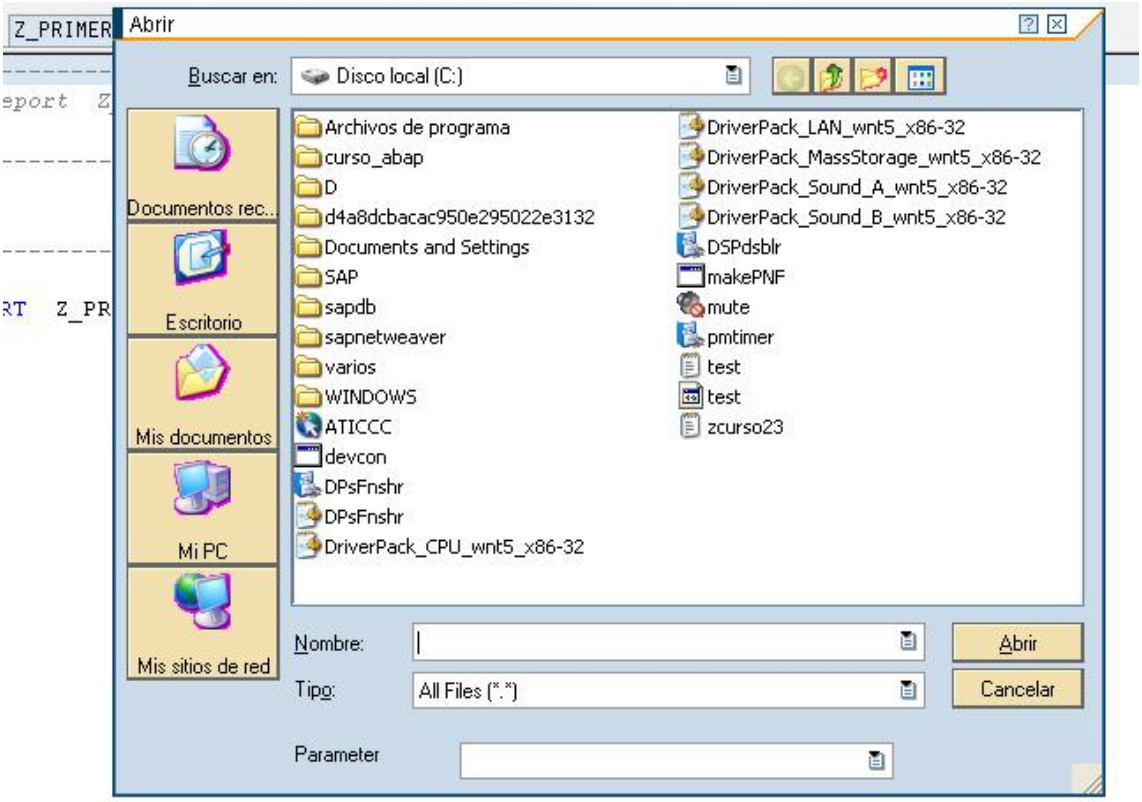


FIG.13- SELECCIÓN DEL ARCHIVO PARA UPLOAD

Veremos que ahora nos aparece el código completo del programa en pantalla.

Activación del programa

Para poder ejecutar el report, necesitamos **activarlo.** Solo el código activo puede ejecutarse.El estado podemos observarlo a la derecha del nombre, como se ve en la figura (en este caso como recién lo creamos, su estado es inactivo).

Para activarlo, presionaremos el botón **de activación** que aparece en el menú.

Aparecerá una pantalla como la de la fig. 14

Haciendo click en el cuadrado de la izquierda de cada línea, la seleccionamos (vemos que queda resaltada, en este caso marcamos ambas). Finalmente, haciendo click en el botón  de la parte inferior, el programa quedará activado (y veremos que a la derecha del nombre del report, el estado paso de inactive a active).

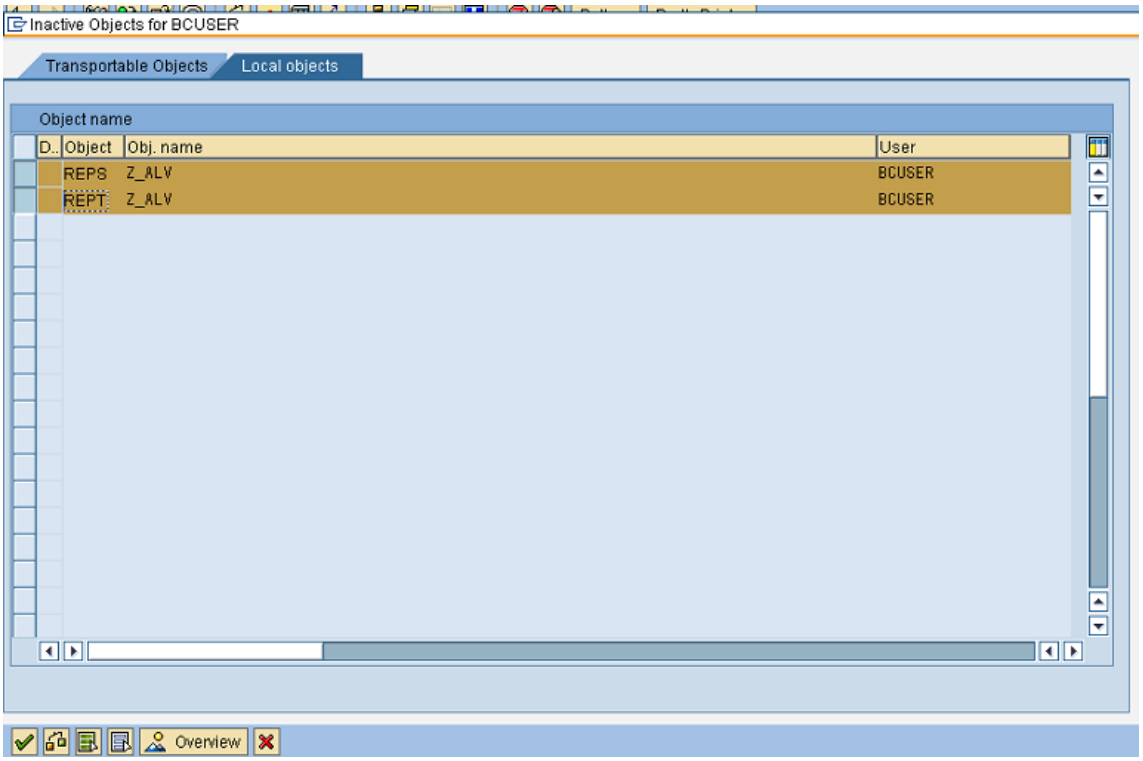


FIG. 14- ACTIVACION DE REPORT

Ejecución de nuestro primer programa

Ejecutémoslo para ver que hace, y luego explicaremos el código. Presionando la tecla **F8**, ejecutamos el programa. Al hacerlo, nos aparecerá una pantalla similar a esta:

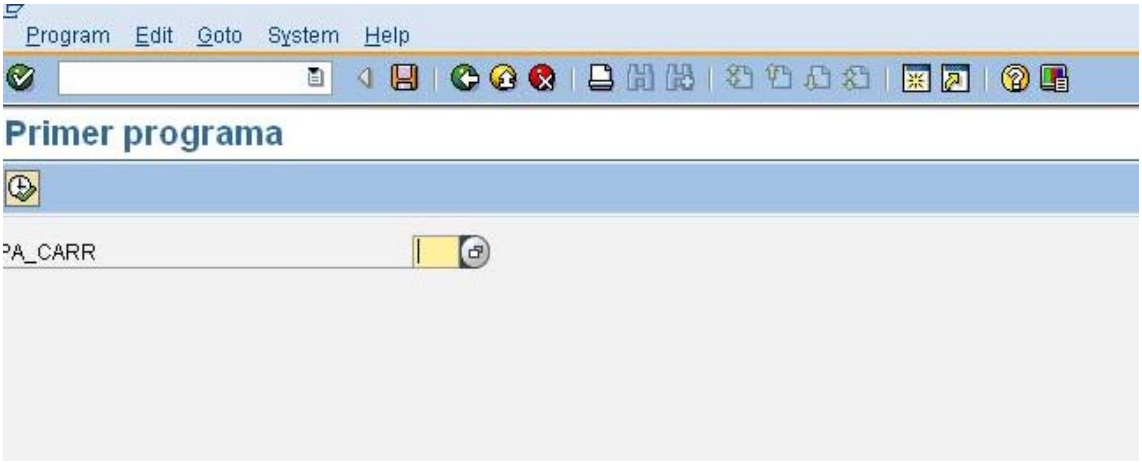
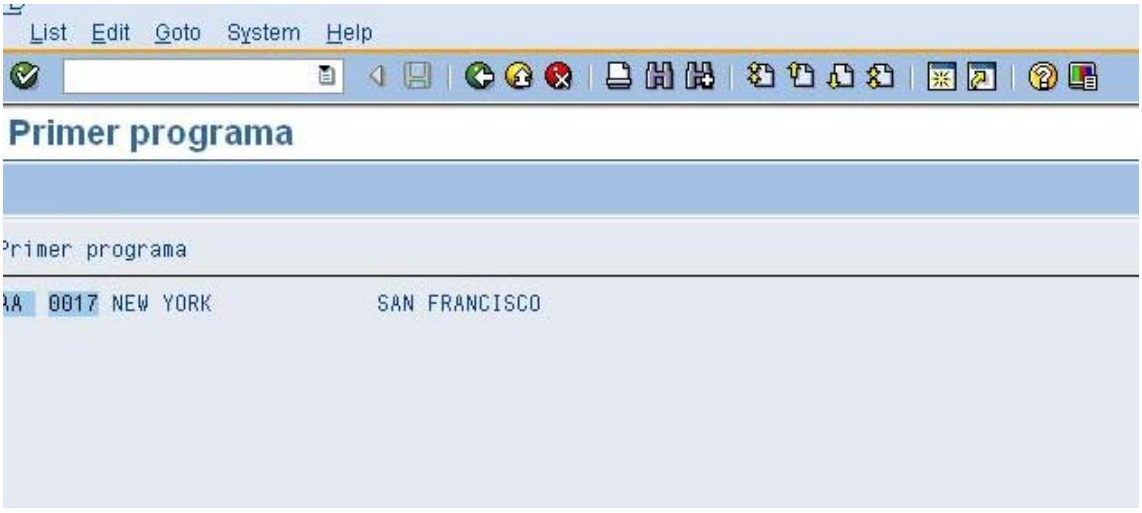


FIG 15- EJECUCION DE PRIMER PROGRAMA

Escribimos 'AA' en el cuadro (valor de PA_CARR), y presionamos nuevamente **F8 (o click en el botón**  **).** El report se ejecuta y aparecerá una pantalla similar a esta:



F16- RESULTADO EJECUCION DEL PRIMER PROGRAMA

Elementos de texto

Naveguemos ahora nuevamente hacia el editor de código (presionando dos veces el botón ).

En la pantalla de selección (Fig.15), vemos que la etiqueta del textbox para ingresar valores es PA_CARR. Este es el nombre del parámetro, pero no da información del valor que hay que ingresar (es nro. de vuelo? Código de aerolínea?). Entonces, podemos modificar este valor, de la siguiente manera.

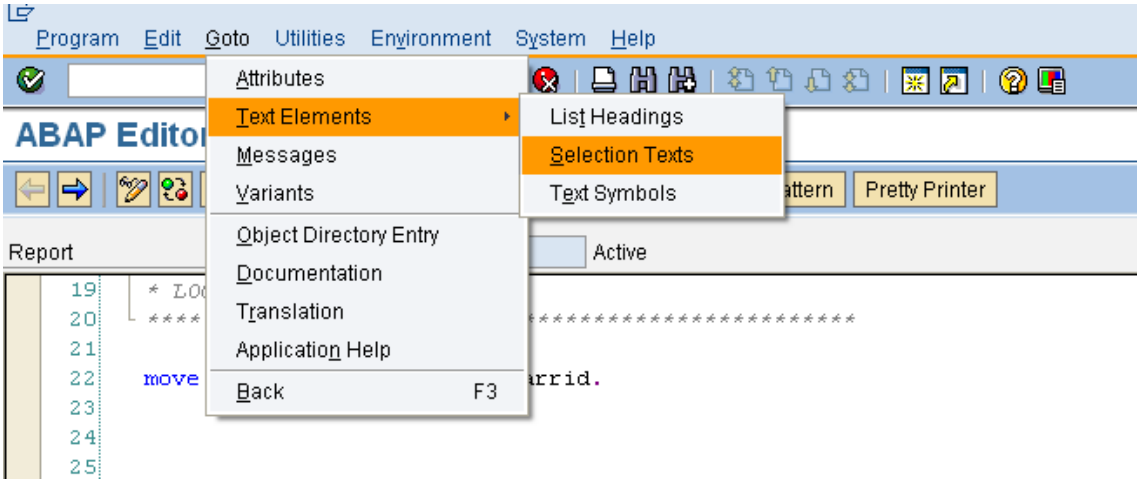


FIG 17- ELEMENTOS DE TEXTO

En el editor Abap, vamos a Text Elements->Selection Texts, como indica la figura 17.

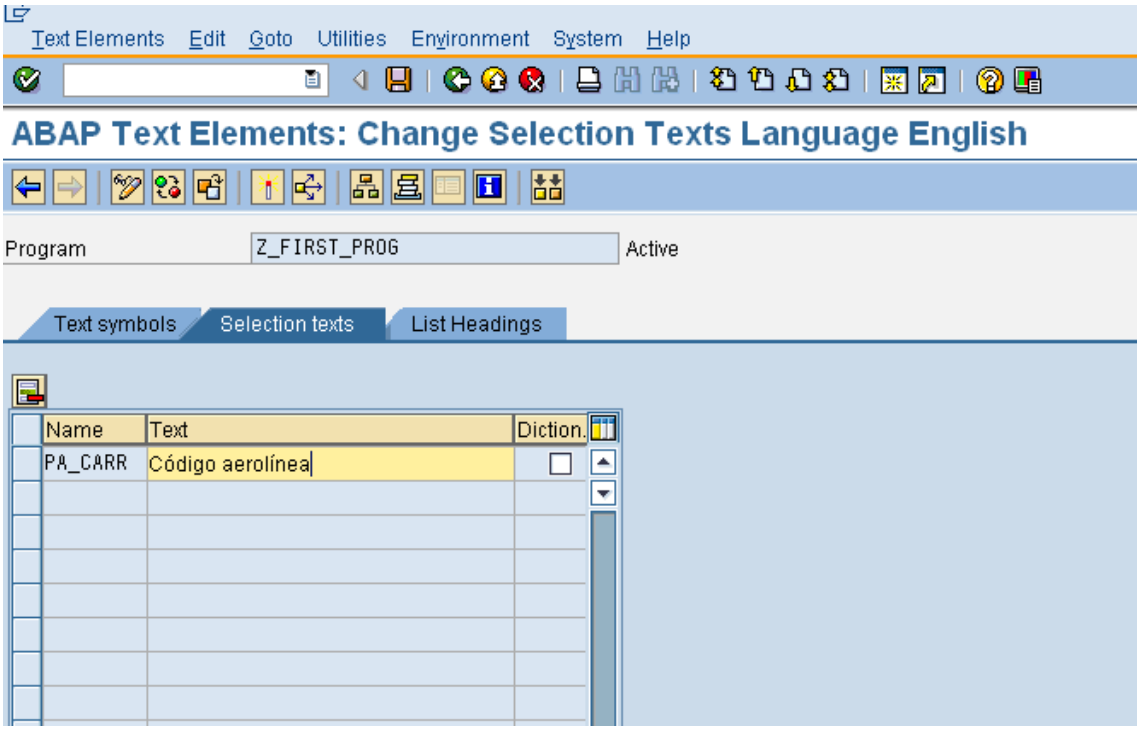


FIG 17- ELEMENTOS DE TEXTO

Aparecerá una pantalla como la de la fig.17. En la segunda solapa, Selection texts, introducimos el texto que tendrá el parámetro PA_CARR. Luego grabamos y activamos.

Si ejecutamos nuevamente el programa, veremos que ahora, en la pantalla de selección, aparece el texto que introdujimos en el paso anterior.

Y ahora, veamos un poco el código del programa.

Pantalla de selección

Para ello, primero veamos algunos aspectos de abap. El lenguaje abap es un lenguaje guiado por eventos. Estos son los que controlan la ejecución del programa. Veremos que existen varios eventos, que tienen un orden de ejecución, y que no es necesario que se escriban en un orden determinado.

Si la declaramos, la pantalla de selección (fig.15) será la primera en ejecutarse, y en ella ingresaremos los datos y pulsadores necesarios para el programa.

En abap, al igual que en otros lenguajes, podemos tener variables y constantes.

Variables

Para crear las mismas, usamos la instrucción **DATA.**

Podemos crear las variables de diversas maneras:

1- En función de un tipo base:
   DATA ZVARIABLE(3) TYPE C.
   Crea una variable tipo carácter, de longitud 3.
   Nota: si no se declara el tipo, se asume por defecto tipo char, y si no se indica la longitud, por defecto será 1.

2- En referencia a un objeto del diccionario de datos:
   Se usa **LIKE ,** y de esta forma, la variable creada tendrá el mismo tipo y longitud que el objeto del diccionario de datos al cual referencia.
   DATA ZVARIABLE LIKE SPFLI-CONNID
   Crea la variable VARIABLE que tendrá la misma longitud y tipo de datos que el campo CONNID de la tabla SPFLI.
   DATA ZVARIABLE TYPE S_CONN_ID
   Crea la variable ZVARIABLE del mismo tipo que el elemento o de datos S_CONN_ID.
   El efecto de las dos sentencias es idéntico, en ambos casos ZVARIABLE será numérico de longitud 4.

   Las variables pueden ser GLOBALES (accesibles desde cualquier parte del programa) o LOCALES (solo accesibles en la porción de código en donde son declaradas).

Tablas internas

Para gestionar datos en los programas, se usan las tablas internas. Las mismas solo existirán en tiempo de ejecución. Podemos tener dos tipos de tablas internas

con cabecera de línea
sin cabecera de línea

Si declaramos la tabla sin cabecera, deberemos declarar un área de trabajo, que tendrá la misma estructura (definición de campos) que la tabla interna. Si la tabla tiene cabecera, no es necesario declarar área de trabajo. Entenderemos mejor esto con los ejemplos que vayamos viendo más adelante.

Ej:

Declaración de tabla con cabecera

```
DATA: BEGIN OF TABLA OCCURS 0,

      COLUMNA1 TYPE C,

      COLUMNA2 TYPE I,

      END OF TABLA.
```

Al colocar el OCCURS 0, la tabla tendrá cabecera, y no será necesario declarar área de trabajo.

```
DATA: BEGIN OF TYE_TABLA,
        COLUMNA1 TYPE C,
        COLUMNA2 TYPE I,
        END OF TYE_TABLA.
```

Aquí estamos declarando una estructura, que es un tipo tabla. A continuación, declaramos la tabla en referencia al tipo creado:

```
DATA TABLA TYPE TYE_TABLA WITH HEADER LINE.
```

En este caso estamos declarando la tabla con cabecera (WITH HEADER LINE).

```
DATA: TABLA TYPE STANDARD TABLE OF TYE_TABLA.

DATA: WA_TABLA TYPE TYE_TABLA.
```

Aquí estamos declarando en primer lugar, la tabla sin cabecera, y luego el work área (área de trabajo WA_TABLA).

Procesamiento de tablas internas

Podemos realizar las siguientes operaciones sobre tablas internas:

- Insertar registros
- Leer registros
- Modificar registros
- Ordenar los registros
- Determinar sus atributos

1- Insertar registros
   Append: añade registros al final de la tabla interna.
   Insert: inserta nuevos registros antes de una línea existente en la tabla interna
   Collect: añade o modifica registros de la tabla interna en función de la clave estándar de la tabla. Es utilizado para calcular totales de campos numéricos.
2- Leer registros
   Read: lee un registro de la tabla interna usando un índice o clave.
   Loop-Endloop: se recorre la tabla interna, y se realiza la lectura, registro a registro.
   SUM: suma los campos numéricos de los registros de una tabla interna.
3- Modificar registros
   MODIFY: modifica el contenido de un registro

WRITE ..TO: modifica el contenido de un registro.
DELETE: borra registros de la tabla.

4- Ordenar los registros
SORT: Ordena los registros de la tabla, según criterios indicados
5- Determinar sus atributos
Recupera información de una tabla interna: numero de registros, etc.

Entonces, para recorrer la tabla interna, usamos la instrucción

LOOP AT <intab> (WHERE <cond>).

...

ENDLOOP.

En cada iteración coloca la línea de la tabla que se está procesando en la línea de cabecera . Podemos usar la condición **WHERE** para restringir el proceso de una tabla.

Si la tabla no tiene cabecera, debemos usar el area de trabajo.

LOOP AT <intab> INTO <work_area> (WHERE <cond>).

...

ENDLOOP.

Tratamiento de cortes de control

En el tratamiento de un LOOP podemos utilizar cortes de control.

AT FIRST

....

ENDAT.

Realiza las instrucciones que hay a continuación del AT FIRST para la primera entrada de la tabla.

AT LAST

....

ENDAT.

Realiza las instrucciones que hay a continuación del AT LAST para la última entrada de la tabla.

AT NEW <campo>

....

ENDAT.

Realiza las instrucciones que hay a continuación del AT NEW para cada inicio de nivel de ruptura.

AT END OF <campo>.

...

ENDAT.

Realiza las instrucciones que hay a continuación del AT END para cada final de nivel de ruptura.

Utilizando la instrucción **SUM** dentro de un AT ... ENDAT se sumaran todos los campos P,I,F de ese nivel de ruptura (para el cálculo de subtotales).

El resultado lo encontraremos en el área de trabajo de la tabla.

SORT <intab> BY <c1> <c2>.

LOOP AT <itab>.

AT FIRST ... (SUM) ... ENDAT.

AT NEW <c1>.

... (SUM) ...

ENDAT.

AT NEW <c2>.

... (SUM) ...

ENDAT.

..... “Proceso Normal de la tabla

AT END OF <c2>.

... (SUM) ...

ENDAT.

AT END OF <c1>.

... (SUM) ...

ENDAT.

AT LAST ... (SUM) ... ENDAT.

ENDLOOP.

Iremos viendo más características del lenguaje, y ahondando en temas, a medida que avancemos en los ejemplos.

Primer programa: explicación del código

Veamos entonces el código de nuestro primer programa.

```
*&---------------------------------------------------------------------*
*& Report  Z_FIRST_PROG
*&
*&---------------------------------------------------------------------*
```

```
*&
*&
*&---------------------------------------------------------------------
```

En primer lugar, observamos unas líneas que comienzan con '*'. Estos son comentarios, o sea que no se toman como código ejecutable.

```
report  z_first_prog.
```

El report siempre debe comenzar con la palabra report y el nombre del programa en cuestion.Cada sentencia debe finalizar con un punto.

```
*********************************************************
* DECLARACION DE DATOS
*********************************************************

data wa_spfli like spfli.
parameters pa_carr like spfli-carrid.
```

Declaración de datos

Declaramos en primer lugar un work área que tendrá los mismos campos que la tabla spfli. El work área vendría a ser como la estructura de una tabla, pero que almacenará un solo registro de la misma; y con 'like' estamos diciendo que tendrá la misma estructura que la tabla SPFLI del diccionario de datos.

Con parameters, declaramos parámetros de la pantalla de selección. O sea, es un campo que tendrá un valor único. Aquí colocaremos el valor que tendrá el código de línea aérea.

```
*********************************************************
* LOGICA DEL PROGRAMA
*********************************************************

* Selection de datos
select single carrid
            connid
            cityfrom
```

```
            cityto
      from   spfli
     into corresponding fields of wa_spfli
     where carrid = pa_carr.
```

Lo que hacemos acá es seleccionar los valores de los campos carrid, connid, cityfrom y cityto, de la table spfli. Estos valores los almacenamos en los correspondientes campos del work área wa_spfli declarado anteriormente. Esta selección la hacemos para aquellos registros que cumplan con la condición de carrid = pa_carr (el valor que introducimos en la pantalla de selección).(para entender mejor el programa, podemos apoyarnos viendo la tabla spfli con la transacción SE11).Al colocar la palabra **single** luego del select, estamos haciendo la selección de un solo registro. Si hubieran varios que cumplan con la condición dada, se recupera en wa_spfli solo el primero.

```
if sy-subrc ne 0.
  message i208(00) with text-001.
  leave to current transaction.
endif.
```

SY-SUBRC hace referencia al campo subrc de la estructura SYST de abap (podemos visualizarla en el diccionario de datos). En esta variable obtendremos un código de retorno.En este caso, de la operación select. Si se obtuvieron datos, en sy-subrc se almacenará el valor cero (operación exitosa). En caso contrario, un valor distinto de cero.

Entonces, lo que hacemos es consultar el valor de sy-subrc. Si es distinto de cero, hacemos que se despliegue por pantalla un mensaje, que tendrá el texto text-001. A continuación, volverá a ejecutar la pantalla de selección, para poder cambiar el criterio de selección de datos.

Creación de elementos de texto

Podemos crear el texto text-001 de dos formas diferentes.

1- Vamos a Text Elements->Selection Texts, como indica la figura 17, y allí introducimos en la primera solapa el valor del texto en cuestión.
2- Hacemos doble click sobre ‘text-001’. Como el texto aun no existe aparecerá el mensaje que podemos ver en la figura 18.Clickeamos ok, y luego seguimos igual que la primera forma.

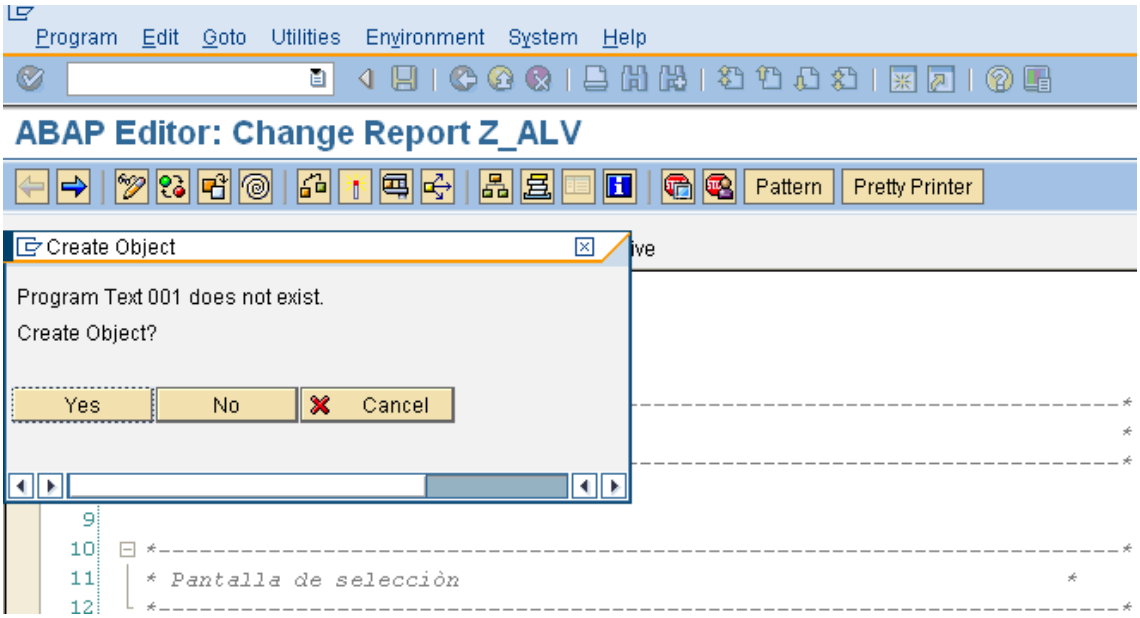


FIG 18- CREACION ELEMENTO DE TEXTO

```
* salida de datos
write:/ wa_spfli-carrid color col_key,
        wa_spfli-connid color col_key,
        wa_spfli-cityfrom,
        wa_spfli-cityto.
```

Aquí vemos como con la simple sentencia write, damos salida por pantalla al resultado de la selección, almacenado en wa_spfli.

Visto todo esto, vayamos avanzando con otros ejemplos.

SEGUNDO PROGRAMA

Ingresamos a la transacción SE38, y escribimos Z_SEC_PROG. Click en create, y una vez dentro, repetimos lo visto en el primer programa, para realizar un upload del correspondiente report (que se encuentra en la carpeta EJEMPLOS del DVD.

Al igual que en el primer ejemplo, creamos el texto para el mensaje de error, introduciendo texto que podría ser 'No existen datos para la selección'.Idem con el texto de selección. Activamos el report, lo ejecutamos, con Código de Aerolínea (PA_CARR) 'AA'.Observamos en este caso que no obtenemos un solo registro, sino varios.

Según vemos en el código, lo que cambió en este report con respecto al anterior es:

1- Hacemos un select sin colocar single, con lo cual se seleccionan todos los registros de la tabla sflight que cumplen con la condición.
2- Como vamos a seleccionar múltiples registros, vemos que el select se cierra con un endselect. El efecto de hacer esto es que, van seleccionándose los registros de la tabla sflight, y por cada uno, se realiza la impresión del mismo (es un ciclo).

```
*&---------------------------------------------------------------------*
*& Report  Z_SEC_PROG
*&
*&---------------------------------------------------------------------*
*&
*&
*&---------------------------------------------------------------------*

report  z_sec_prog.
*****************************************************
* DECLARACION DE DATOS
*****************************************************

data wa_flights like sflights.
parameters pa_carr like sflight-carrid.

*****************************************************
* LOGICA DEL PROGRAMA
*****************************************************

* Selección de datos
```

```
select  carrid
        connid
        fldate
        seatsmax
        seatsocc
   from   sflight
   into corresponding fields of wa_flights
   where carrid = pa_carr.


* salida de datos
  write:/ wa_flights-carrid,
          wa_flights-connid,
          wa_flights-fldate,
          wa_flights-seatsmax,
          wa_flights-seatsocc.

endselect.
```

TERCER PROGRAMA

Realizamos el mismo proceso que en los dos programas anteriores; en el editor abap, creamos el programa Z_THI_PROG. Una vez dentro del mismo, realizamos el upload del archivo del mismo nombre, en la carpeta EJEMPLOS del DVD.Repetimos el proceso para crear textos de selección y elemento de texto del mensaje de error.

Si lo ejecutamos (F8), vemos que el resultado es el mismo que en el ejemplo anterior. ¿Qué cambió en este report?

Veamos por partes:

```
report  z_thi_prog.


**********************************************************
* DECLARACION DE DATOS
**********************************************************

data: begin of ti_sflight occurs 0,
      carrid   like sflight-carrid,
      connid   like sflight-connid,
      fldate   like sflight-fldate,
      seatsmax like sflight-seatsmax,
      seatsocc like sflight-seatsocc,
      end of ti_sflight.

parameters pa_carr like sflight-carrid.
```

En primer lugar vemos que hemos declarado una table interna con cabecera (occurs 0). Hemos declarado solo algunos campos que pertenecen a la tabla del diccionario de datos, SFLIGHT, y, según vimos anteriormente, declaramos estos datos en referencia a los campos de dicha tabla (like…). El parameter no cambia.

```
**********************************************************
* LOGICA DEL PROGRAMA
**********************************************************

* Selección de datos
```

```
select  carrid
        connid
        fldate
        seatsmax
        seatsocc
   from   sflight
   into table ti_sflight
   where carrid = pa_carr.

if sy-subrc ne 0.
  message i208(00) with text-001.
endif.
```

En este caso, realizamos el select de los campos que nos interesan (los que declaramos en la tabla interna), y lo hacemos 'into tabla' ti_sflight. Es importante que en el select figuren los mismos campos que en la tabla interna, y en el mismo orden, para no obtener resultados erróneos (salvo que usemos into corresponding fields of ti_sflight).

```
* salida de datos
loop at ti_sflight.
  write:/ ti_sflight-carrid,
          ti_sflight-connid,
          ti_sflight-fldate,
          ti_sflight-seatsmax,
          ti_sflight-seatsocc.

endloop.
```

Aquí la otra diferencia importante, es que vemos que no hacemos un ciclo con select..endselect. En vez de ello, hacemos un select de los registros en la tabla interna ti_sflight, y luego un loop. Esto es, una vez que tenemos los datos almacenados en la tabla interna ti_sflight, recorremos la tabla interna con loop, y por cada registro, realizamos el write correspondiente.

CUARTO PROGRAMA

Repetimos el procedimiento visto en los programas anteriores, para la creación del reporte.

Este programa es similar al anterior (podemos verificarlo ejecutándolo y viendo la salida por pantalla)pero con algunas diferencias en la codificación.

Para declarar la tabla interna, en este caso (tal como vimos anteriormente cuando explicábamos tablas internas), declaramos primero un tipo de datos.Esto lo hacemos con el types. Luego, con

data: ti_sflight type standard table of ty_sflight with header line.

lo que hacemos, es declarar ya la table, pero del tipo ty_sflight creado anteriormente. Con el header line, hacemos que tenga cabecera. Con esto logramos hacer el loopeo de la tabla, sin tener que declarar un area de trabajo.Aclaremos en este punto: cuando realizamos el loopeo de la tabla interna, si tiene cabecera, durante el recorrido, vamos obteniendo cada registro en la cabecera de la tabla. Este será el valor que leamos, borremos, modifiquemos, etc. Si la tabla no tiene cabecera, debemos crear un área de trabajo, y el loopeo se realiza sobre la misma (loop at tabla into workarea).Y con los datos de esta work area se realizaran las operaciones correspondientes.

```
report  z_four_prog.

**********************************************************
* DECLARACION DE DATOS
**********************************************************

types: begin of ty_sflight,
     carrid   like sflight-carrid,
     connid   like sflight-connid,
     fldate   like sflight-fldate,
     seatsmax like sflight-seatsmax,
     seatsocc like sflight-seatsocc,
     end of ty_sflight.

data: ti_sflight type standard table of ty_sflight with header line.

parameters pa_carr like sflight-carrid.

**********************************************************
* LOGICA DEL PROGRAMA
**********************************************************
```

```
* Selección de datos
select  carrid
        connid
        fldate
        seatsmax
        seatsocc
    from   sflight
    into table ti_sflight
    where carrid = pa_carr.

if sy-subrc ne 0.
  message i208(00) with text-001.
endif.

* salida de datos
loop at ti_sflight.
  write:/ ti_sflight-carrid,
          ti_sflight-connid,
          ti_sflight-fldate,
          ti_sflight-seatsmax,
          ti_sflight-seatsocc.

endloop.
```

Todo el código restante es similar al anterior.

QUINTO PROGRAMA

Vamos a crear ahora el programa Z_FIF_PROG.Luego de realizar el upload correspondiente, nos quedará el código que podemos ver más abajo.

Ejecutando el report, podemos ver que es similar al anterior.La diferencia importante que encontramos está en lo que mencionábamos más arriba.

En este caso, también declaramos un tipo ty_sflight. Pero ahora, la tabla interna la declaramos sin cabecera (no tenemos el with header line).En su lugar, declaramos un área de trabajo (wa_sflight)del mismo tipo ty_sflight.

```
report  z_fif_prog.
**********************************************************
* DECLARACION DE DATOS
**********************************************************

types: begin of ty_sflight,
     carrid   like sflight-carrid,
     connid   like sflight-connid,
     fldate   like sflight-fldate,
     seatsmax like sflight-seatsmax,
     seatsocc like sflight-seatsocc,
     end of ty_sflight.

data: ti_sflight type standard table of ty_sflight.
data: wa_sflight type ty_sflight.

parameters pa_carr like sflight-carrid.


El select es igual al realizado en los ejemplos anteriores.

**********************************************************
* LOGICA DEL PROGRAMA
**********************************************************

* Selección de datos
select  carrid
       connid
       fldate
```

```
      seatsmax
      seatsocc
   from   sflight
   into table ti_sflight
   where carrid = pa_carr.

if sy-subrc ne 0.
  message i208(00) with text-001.
endif.
```

En el loopeo vemos lo que señalábamos anteriormente. Como no tenemos tabla con cabecera, el loop lo realizamos en el work area. Esto quiere decir, que se irán recorriendo uno a uno los registros, cada uno de ellos se almacenara temporalmente en wa_sflight, y desde allí lo imprimiremos por pantalla.

```
* salida de datos
loop at ti_sflight into wa_sflight.
  write:/ wa_sflight-carrid,
          wa_sflight-connid,
          wa_sflight-fldate,
          wa_sflight-seatsmax,
          wa_sflight-seatsocc.

endloop.
```

SEXTO PROGRAMA: LISTAS DOBLES

Veamos ahora un programa un poco más complicado, en donde crearemos listas dobles interactivas.

Crearemos ahora el programa Z_LISTAS_DOBLES (y haremos el upload del archivo correspondiente).

Al ejecutarlo vemos una salida por pantalla, al igual que los reportes anteriores. La gran diferencia, es que, si nos paramos sobre cualquier línea, y hacemos doble click sobre la misma, iremos a una segunda pantalla, con detalles de dicho registro. Navegando hacia atrás, volveremos a la primera pantalla, y podemos repetir este proceso.

Analicemos el código paso a paso

REPORT Z_LISTAS_DOBLES.

Declaramos el tipo de estructura

```
TYPES: BEGIN OF struc,
id LIKE scustom-id,
name LIKE scustom-name,
END OF struc.
```

Declaramos la table interna del tipo struct con clave id.

```
DATA: it_scustom TYPE STANDARD TABLE OF struc
WITH KEY id.
```

Declaramos el área de trabajo del tipo struct
```
DATA: wa_scustom TYPE struc.
```

Declaramos áreas de trabajo para las tablas sflight y sbook (consultar para más claridad estas tablas en el diccionario de datos , transacción SE11).
```
DATA: wa_sflight LIKE sflight,
wa_sbook LIKE sbook.
```

Parámetro de la pantalla de selección
```
PARAMETERS: pa_carr LIKE sflight-carrid.
```

Recordemos que mencionábamos, cuando explicábamos características del lenguaje abap, que el mismo es gobernado por eventos. Uno de ellos es el START-OF-SELECTION, y es el que se ejecuta luego de procesar la pantalla de selección.

Como wa_sflight lo declaramos igual que la tabla sflight, y seleccionamos solo algunos campos de dicha tabla, realizamos el **INTO CORRESPONDING FIELDS OF**.

```
START-OF-SELECTION.
* Leer datos correspondientes a los criterios de selección
  SELECT carrid connid fldate planetype seatsmax seatsocc price
  currency
  INTO CORRESPONDING FIELDS OF wa_sflight
  FROM sflight
  WHERE carrid = pa_carr.
    WRITE: / wa_sflight-carrid COLOR COL_KEY,
    wa_sflight-connid COLOR COL_KEY,
    wa_sflight-fldate COLOR COL_KEY,
    wa_sflight-planetype COLOR COL_NORMAL,
    wa_sflight-seatsmax COLOR COL_NORMAL,
    wa_sflight-seatsocc COLOR COL_NORMAL,
    wa_sflight-price CURRENCY wa_sflight-currency
    COLOR COL_NORMAL,
    wa_sflight-currency COLOR COL_NORMAL.
```

Utilizamos HIDE, lo que nos permitirá tener en memoria secundaria los campos indicados (carrid, connid y fldate), para poder visualizarlos en el momento de seleccionar una linea.

```
    HIDE: wa_sflight-carrid, wa_sflight-connid, wa_sflight-fldate.
  ENDSELECT.


*Grabar en memoria intermedia los datos del cliente
  SELECT id name INTO TABLE it_scustom FROM scustom.
*Inicialización
  CLEAR wa_sflight.
*Final del evento Inicio de la selección
* Evento interactivo, lista secundaria
```

Aquí vemos otro evento, que es el de selección de una línea.Cuando hacemos doble click sobre la línea, se ejecuta el código a continuación del evento.

```
AT LINE-SELECTION.
  WRITE: text-001 COLOR COL_NORMAL,
  wa_sflight-carrid COLOR COL_TOTAL,
  wa_sflight-connid COLOR COL_TOTAL,
```

```
  wa_sflight-fldate COLOR COL_TOTAL,
  text-002 COLOR COL_NORMAL.

  Skip avanza una linea
  SKIP.


*Cabecera de página para lista secundaria
  WRITE: text-003 COLOR COL_HEADING.
  SKIP.


* Seleccionar datos para lista secundaria
  SELECT bookid customid custtype class order_date smoker
  INTO CORRESPONDING FIELDS OF wa_sbook
  FROM sbook
  WHERE carrid = wa_sflight-carrid
  AND connid = wa_sflight-connid
  AND fldate = wa_sflight-fldate.
    WRITE: / wa_sbook-bookid COLOR COL_KEY.
* Leer nombre del cliente

    Se limpia el valor de wa_scustom
    CLEAR wa_scustom.

    Ahora, el valor del campo id de wa_scustom tomará el valor de wa_sbook-customid.
    MOVE wa_sbook-customid TO wa_scustom-id.

    Aquí leemos el registro de la tabla interna it_scustom, en el área de trabajo
wa_scustom, cuando se cumpla que el campo id sea igual a wa_scustom-id.
    READ TABLE it_scustom INTO wa_scustom
    FROM wa_scustom.

    Hacemos el write que corresponde al detalle de la segunda lista.
    WRITE: wa_scustom-name COLOR COL_NORMAL,
    37 wa_sbook-custtype COLOR COL_NORMAL,
    42 wa_sbook-class COLOR COL_NORMAL,
    47 wa_sbook-order_date COLOR COL_NORMAL,
    wa_sbook-smoker COLOR COL_NORMAL.
  ENDSELECT.
  CLEAR wa_sflight.
```

SEPTIMO PROGRAMA

Veremos ahora un reporte con una pantalla de selección mas elaborada, e introduciremos el concepto de rutinas.

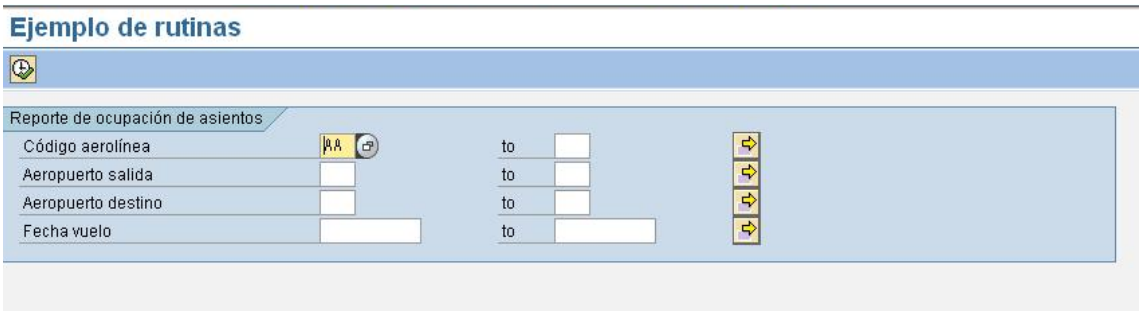
Creemos el programa Z_FORM (y hagamos el upload del archivo correspondiente).

Este es un reporte que genera una lista con los porcentajes de asientos libres para cada vuelo, ordenado por ciudad de partida, ciudad de destino, aerolínea y fecha de vuelo. Para cada par de origen-destino, imprime por pantalla el promedio de porcentaje de asientos libres.

El template del listado por pantalla es de la forma:

```
Departure city
                Arrival city
                Airline carrier
                Flight date      Occupancy
--------------------------------------------------------------------------------
BERLIN
                FRANKFURT
                Promedio de ocupación de todos los vuelos
                desde BERLIN            a FRANKFURT      :  52  %
                LH
                09/07/1998       31  %
                10/10/1998        8  %
                11/29/1998       98  %
                12/02/1998       70  %
FRANKFURT
                BERLIN
                Promedio de ocupación de todos los vuelos
                desde FRANKFURT     a BERLIN             :  81  %
                LH
                08/28/1998       78  %
                09/30/1998       82  %
                11/19/1998       96  %
                11/22/1998      100  %
                11/29/1998       49  %
                12/19/1998       74  %
                12/21/1998       91  %
                NEW YORK
                Promedio de ocupación de todos los vuelos
                desde FRANKFURT     a NEW YORK       :  82  %
                LH
                08/28/1998       99  %
                09/07/1998       97  %
                09/30/1998       39  %
                    :                :
```

Al ejecutarlo, aparecerá la siguiente pantalla de selección.



Veamos el código:

```
report  z_form.


tables: spfli,
        sflight.
```

Aqui vemos que declaramos la pantalla de selección con selection-screen. Con begin of block creamos un bloque lógico de elementos. Con with frame el bloque se engloba dentro de un cuadro. Con title aparecerá el titulo en la primera línea del cuadro. Este texto lo creamos según lo vimos anteriormente.

```
selection-screen begin of block b1 with frame title text-010.
```

Veíamos anteriormente que con la declaración de un parameter, podíamos introducir un valor único. Con SELECT-OPTIONS podemos, al momento de introducir un dato en la pantalla de selección, seleccionar un valor único, un rango de valores, o varios valores individuales. También podemos excluir valores individuales o rangos de valores (ej. podríamos indicar que se seleccionen todas las aerolíneas salvo la AA).Se declaran con la sentencia SELECT-OPTIONS seguida del nombre del mismo, y con FOR indicamos que tendrá el tipo que sigue al mismo y que con la ayuda de selección, obtendremos los datos de dicha tabla.O sea, presionando la tecla F4 o click en el circulo a la derecha del textbox, se desplegará una pantallita con los valores que posee el campo de la tabla (ej. SPFLI-CARRID).

```
select-options: s_carrid for spfli-carrid,
                s_afrom  for spfli-airpfrom,
                s_ato    for spfli-airpto,
                s_fldate for sflight-fldate.
```

```
selection-screen end of block b1.
```

Aquí declaramos tres tipos, como lo vimos anteriormente.

```
types:  begin of struct_spfli,
                carrid      like spfli-carrid,
                connid      like spfli-connid,
                cityfrom    like spfli-cityfrom,
                cityto      like spfli-cityto,

              end of struct_spfli.

types:  begin of struct_flight,
                carrid      like sflight-carrid,
                connid      like sflight-connid,
                fldate      like sflight-fldate,
                seatsmax    like sflight-seatsmax,
                seatsocc    like sflight-seatsocc,
       end of struct_flight.

types: begin of struct_list,
                carrid      like spfli-carrid,
                connid      like spfli-connid,
                cityfrom    like spfli-cityfrom,
                cityto      like spfli-cityto,
                fldate      like sflight-fldate,
                seatsmax    like sflight-seatsmax,
                seatsocc    like sflight-seatsocc,
                capacity    type i,
                number      type i,
      end of struct_list.
```

Aqui declaramos variables y valores.

```
data:  pos1 type i value 18, len1 type i, len2 type i, len3 type i,capacity type i.
```

Declaración de las tablas con cabecera.

```
data:  itab_flight type standard table of struct_flight with header line.
data : itab_spfli type standard table of struct_spfli with header line.
data : itab_list type standard table of struct_list with header line.
```

Comienza el código principal.

```
start-of-selection.
```

Aquí asignamos valores a las variables.

Nota: si queremos ver el valor o significado de algún elemento del diccionario, haciendo doble click,vamos a la transacción correspondiente y nos posicionamos en el elemento correspondiente. Por ejemplo, si no sabemos que es sy-linsz, al hacer doble click sobre el mismo, navegamos automáticamente a la transacción se11, dentro de la misma visualizamos la estructura SYST y posicionados LINSZ. Aquí podremos ver que corresponde al ancho de la línea de la lista.

```
  len1 = sy-linsz.
  len2 = len1 - pos1 + 1.
  len3 = len2 - 23.
```

Aquí se hace el select igual que en los ejemplos anteriores

```
  select carrid
         connid
         cityfrom
         cityto
    from spfli
    into table itab_spfli
    where carrid in s_carrid
    and   airpfrom in s_afrom
    and   airpto   in s_ato.

  if sy-subrc ne 0.
    message e208(00) with text-003.
  endif.
```

Aquí veremos una variación en el select.Hacemos un select de la table SFLIGHT pero como criterio de selección, vemos que hacemos ‘FOR ALL ENTRIES IN ITAB_SPFLI’. Esto quiere decir que tomara todos los valores de la tabla interna ITAB_SPFLI, y hará la selección de datos de la tabla sflight para aquellos registros que tengan los campos CARRID Y CONNID con los valores recuperados de ITAB_SPFLI.Tambien vemos que debe cumplirse ‘fldate in s_fldate’. En el select-options, en vez de colocar, como en el parameter, el criterio =, colocamos ‘in’ (no ponemos fldate = s_fldate, sino fldate in s_fldate). Esto es porque, como indicamos anteriormente, en el select-options podemos tener rangos de valores, y nos interesa

considerar todos.

```
  select carrid
         connid
         fldate
         seatsmax
         seatsocc
    from sflight
    into table itab_flight
    for all entries in itab_spfli
    where carrid = itab_spfli-carrid
    and   connid = itab_spfli-connid
    and   fldate in s_fldate.

  if sy-subrc ne 0.
    message e208(00) with text-007.
  endif.

  loop at  itab_flight.
    if itab_flight-seatsmax < itab_flight-seatsocc.
      message i208(00) with text-009.
    endif.
    if itab_flight-seatsmax eq 0.
      message e208(00) with text-008.
    endif.
    read table itab_spfli
      with key carrid = itab_flight-carrid
               connid = itab_flight-connid.
```

Ahora veamos la rutina. El form es una porción de código, que va entre las sentencias FORM…ENDFORM, y se invoca con la sentencia PERFORM.Puede o no tener parámetros. Con **using,** pasamos los parámetros de entrada, con **changing** los parámetros que serán modificados por la rutina(aquí en gral. se devuelven los valores del resultado del procesamiento). Y si usáramos tables, le pasamos tablas internas, que pueden ser leídas y modificadas.

En nuestro ejemplo, invocamos a la rutina f_calcula_porcentaje, a la que le pasamos como parámetros de entrada la cantidad de asientos (itab_flight_seatsmax)y la cantidad de asientos ocupados (itab_flight-seatsocc). Luego de realizar el cálculo correspondiente, el resultado es devuelto en itab_list_capacity.

Creación de subrutina

Si en el código, hacemos doble click sobre ‘f_calcula_porcentaje’, navegaremos directamente al código de la rutina. Si estuviéramos escribiendo nosotros el código desde cero, al tipear:

perform f_rutina , y hacer doble click sobre el nombre del form (f_rutina), como todavía no declaramos el form, aparecerá un mensaje similar al de la figura 19.

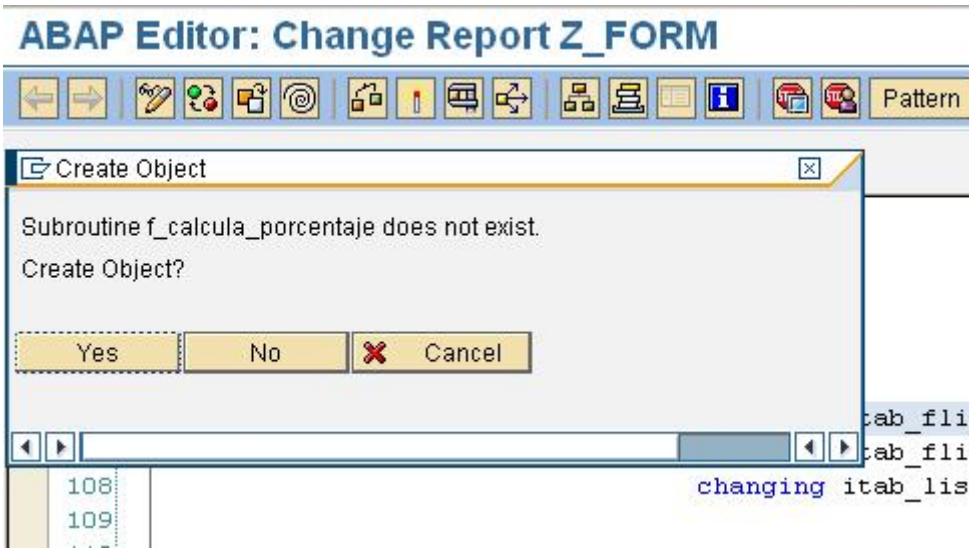


FIG. 19-CREACION DE FORM

Hacemos click en Yes y aparece:

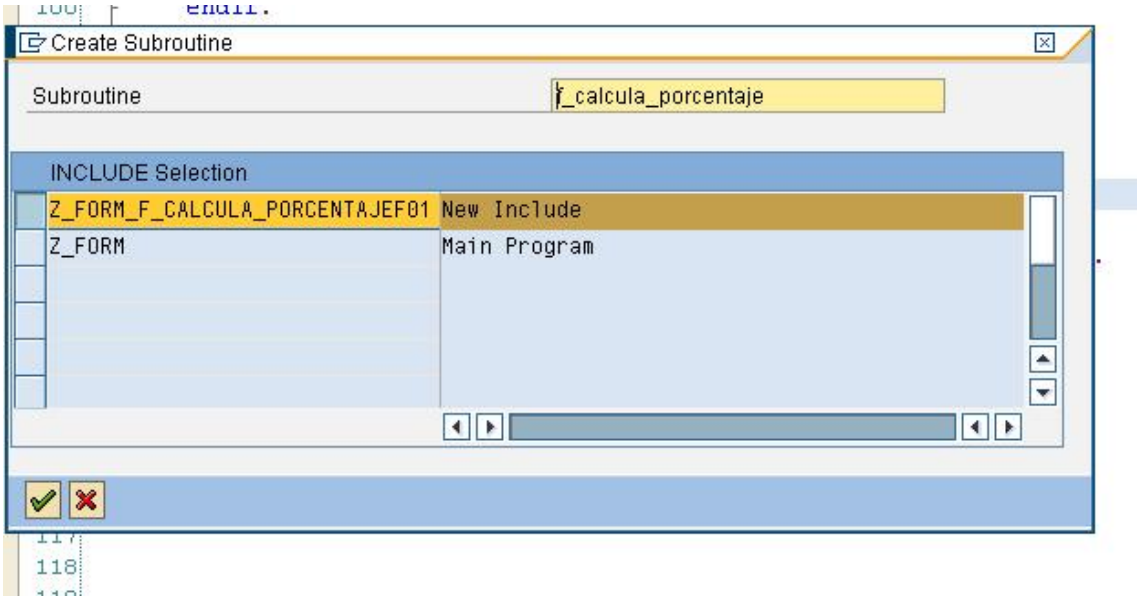


FIG.20-CREACION DE FORM

Esto quiere decir que la subrutina podemos crearla en un nuevo include o en el programa principal.

Nota: Un include es como un nuevo reporte, tendrá un nombre distinto al reporte principal, pero no es ejecutable como el reporte. Es invocado por el programa principal, y se usa normalmente para modularizar los programas, y colocar en los mismos, declaración de variables, rutinas, etc.

En nuestro caso, seleccionamos Z_FORM, hacemos enter, y veremos que la subrutina se crea en el programa principal.

```
*&---------------------------------------------------------------------*
*&      Form  f_calcula_porcentaje
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*      -->P_ITAB_FLIGHT_SEATSMAX  text
*      -->P_ITAB_FLIGHT_SEATSOCC  text
*      <--P_ITAB_LIST_CAPACITY  text
*----------------------------------------------------------------------*
form f_calcula_porcentaje  using    p_itab_flight_seatsmax
                                    p_itab_flight_seatsocc
                           changing p_itab_list_capacity.

endform.                    " f_calcula_porcentaje
```

FIG.21- CREACION DE SUBRUTINAS

```
perform f_calcula_porcentaje using    itab_flight-seatsmax
                                      itab_flight-seatsocc
                           changing itab_list-capacity .
```

Entonces, se almacenan los valores de las tablas internas itab_spfli e itab_flight en la tabla interna itab_list. Y, como podemos observar, queda almacenado en itab_list_capacity el valor devuelto por la rutina f_calcula_porcentaje.

Con el append itab_list, hacemos que el valor de la cabecera de itab_list quede efectivamente almacenado en la tabla itab_list.

```
itab_list-number = 1.
move-corresponding: itab_spfli  to itab_list,
                    itab_flight to itab_list.
append itab_list.
endloop.
```

Aqui ordenamos la table itab_list por los campos indicados (cityfrom, cityto, carrid, fldate).

```
* Sort internal table
 sort itab_list by cityfrom cityto carrid fldate.
```

```
* Control level processing
 loop at itab_list.
```

Cuando encuentre un nuevo itab_list-cityfrom, lo imprime por pantalla.

```
*  AT NEW cityfrom
   at new cityfrom.
     format color col_heading intensified on.
     write at: /(len1) itab_list-cityfrom.
   endat.
```

Cuando encuentre un nuevo itab_list-cityto, lo imprime por pantalla.Suma todos los porcentajes de asientos libres hallados, calcula el promedio y lo imprime.

```
*  AT NEW cityto
   at new cityto.
     format color col_heading intensified off.
     write at: /pos1(len2) itab_list-cityto.
*       SUM
     sum.
     format color col_total intensified on.
     capacity = itab_list-capacity / itab_list-number.
     write:  at /pos1(len2) 'Average occupancy percentage according to flight'(001),
                  at /pos1 'von'(002), (15) itab_list-cityfrom, 'to'(003),
                  (15) itab_list-cityto, ':', (4) capacity, '% '.
   endat.
```

Imprime el carrid cuando encuentra uno Nuevo, durante el loopeo.

```
*   AT NEW carrid.
   at new carrid.
     format color col_key intensified off.
     write at: /pos1(len2) itab_list-carrid.
   endat.
```

Procesamiento normal de la lista

```
*   Single data processing
    format color col_normal intensified off.
    write: at /pos1 itab_list-fldate, itab_list-capacity,
                  at (len3) '% '.

  endloop.
*&---------------------------------------------------------------------*
*&      Form  f_calcula_porcentaje
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
form f_calcula_porcentaje
                      using    pt_seatsmax like sflight-seatsmax
                               pt_seatsocc like sflight-seatsocc
                      changing pt_capacity type i.

  pt_capacity =  ( pt_seatsmax - pt_seatsocc )  *  100
                               /  pt_seatsmax.

endform.                    " f_calcula_porcentaje
```

OCTAVO PROGRAMA

Veremos ahora el concepto de funciones. Repetiremos el ejemplo anterior, pero en vez de usar una rutina para el cálculo de asientos libres, usaremos una función.

Una función es una porción de código como la rutina, pero se genera de distinto modo, es independiente del reporte, y puede ser invocada por cualquier reporte. Tiene también parámetros de entrada y de salida. Veámosla en forma práctica.

Crearemos para ello el reporte Z_FUNCT. Realicemos el upload del archivo correspondiente. Si inspeccionamos el código, veremos que es similar al anterior. La diferencia que encontraremos, es que ahora no usamos una subrutina, sino que invocamos una función para el cálculo del porcentaje de asientos libres.

La parte del código que varia es:

```
call function 'Z_VUELOS'
     exporting
       seatsmax         = itab_flight-seatsmax
       seatsocc         = itab_flight-seatsocc
     importing
       percentage       = itab_list-capacity
     exceptions
       division_by_zero = 1
       occ_gt_max       = 2.


   case sy-subrc.
     when 1.
       write:  text-zer, itab_flight-carrid, itab_flight-connid.
     when 2.
       write:  text-max, itab_flight-carrid, itab_flight-connid.
     when 0.

       itab_list-number = 1.
       move-corresponding: itab_spfli  to itab_list,
                           itab_flight to itab_list.

   endcase.
```

Aquí vemos que invocamos a una función llamada Z_VUELOS. Le pasamos como parámetros de entrada itab_flight-seatsmax e itab_flight-seatsocc. Recibiremos el resultado (percentage) en itab_list-capacity.Vemos también que hay un manejo de excepciones, que entenderemos enseguida cuando veamos el código de la función. Por

ahora diremos que si el valor del sy-subrc es 1, significa que la cantidad de asientos total es cero, por lo que imprimiremos el registo para lo cual se produce esto, y no lo tendremos en cuenta en el cálculo total. Si el sy-subrc es 2, significa que la cantidad de asientos total es menor que la de asientos ocupados, lo cual también es un error, por lo que imprimimos el registro para el cual se da este caso, y tampoco lo tenemos en cuenta para el cálculo total.

Creemos la función para ir comprendiendo mejor los conceptos.

La transacción para gestionar funciones es la **SE37**.

Cuando creemos nuestro modulo de funciones, deberemos hacerlo pertenecer a un grupo de funciones, por lo cual, previamente crearemos el mismo. Al pertenecer a un grupo de funciones, los módulos de funciones compartirán ciertas características. Por ejemplo, al declarar variables globales en un modulo de funciones, las mismas se compartirán con los demás módulos de función que integren el grupo de funciones.

Creación de grupo de funciones-Transacción SE80

Para crear el grupo de funciones, usaremos la transacción **SE80**, que es el Object Navigator.

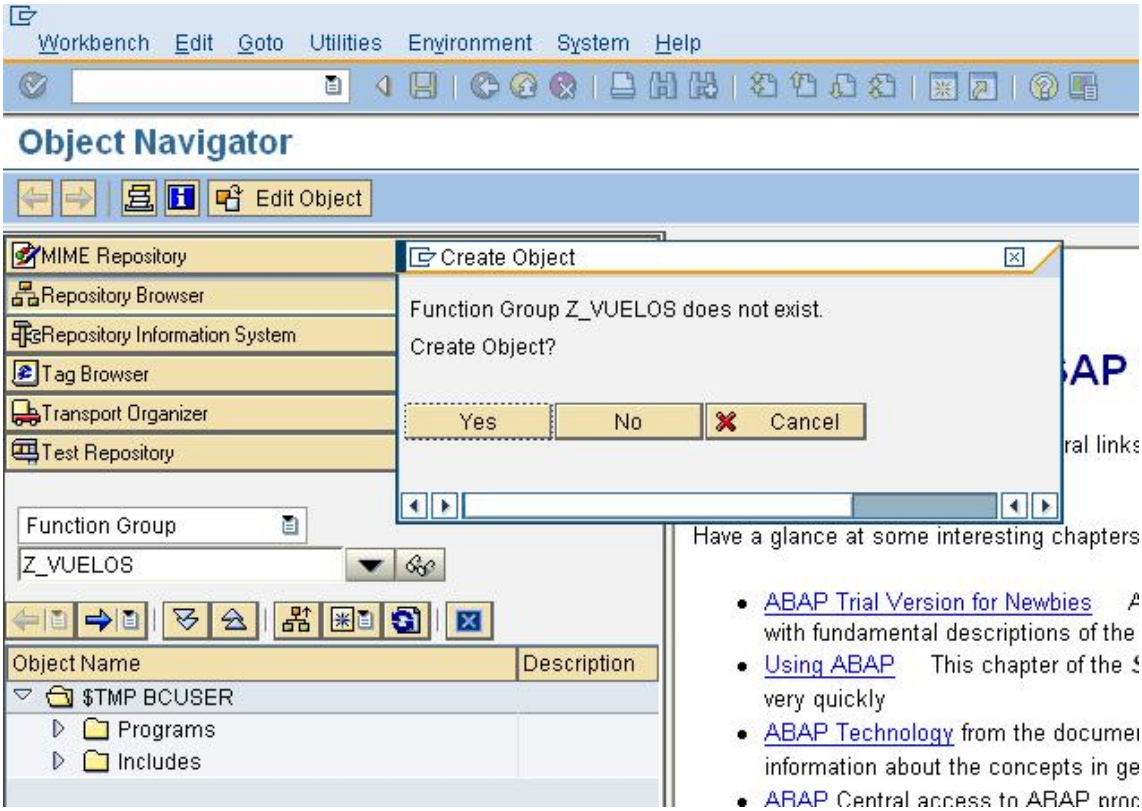


FIG.21- GRUPO DE FUNCIONES

Tal como vemos en la FIGURA 21, seleccionaremos como objeto para crear FUNCTION GROUP y debajo, ingresaremos el nombre que tendrá nuestro grupo de funciones, Z_VUELOS.Cuando presionemos enter, aparecerá el mensaje que podemos observar en la figura, indicando que el grupo de funciones no existe. Presionamos Yes para crearlo.Aparecerá una nueva ventana como podemos observar en la figura 22.Colocamos una descripción en short text y presionamos Save.Como podemos ver en la figura 23, se ha creado el grupo de funciones, con dos includes.Y a la derecha, en dicha figura, vemos el editor del primer include, en donde podremos ingresar código.Podemos observar también en esta parte de la figura, que el include está inactivo, por lo cual procederemos a activarlo (igual que como hacíamos en la transacción SE38).Como podemos ver en la figura 24, aparecerán dos objetos para activar.Seleccionamos los dos y los activamos.

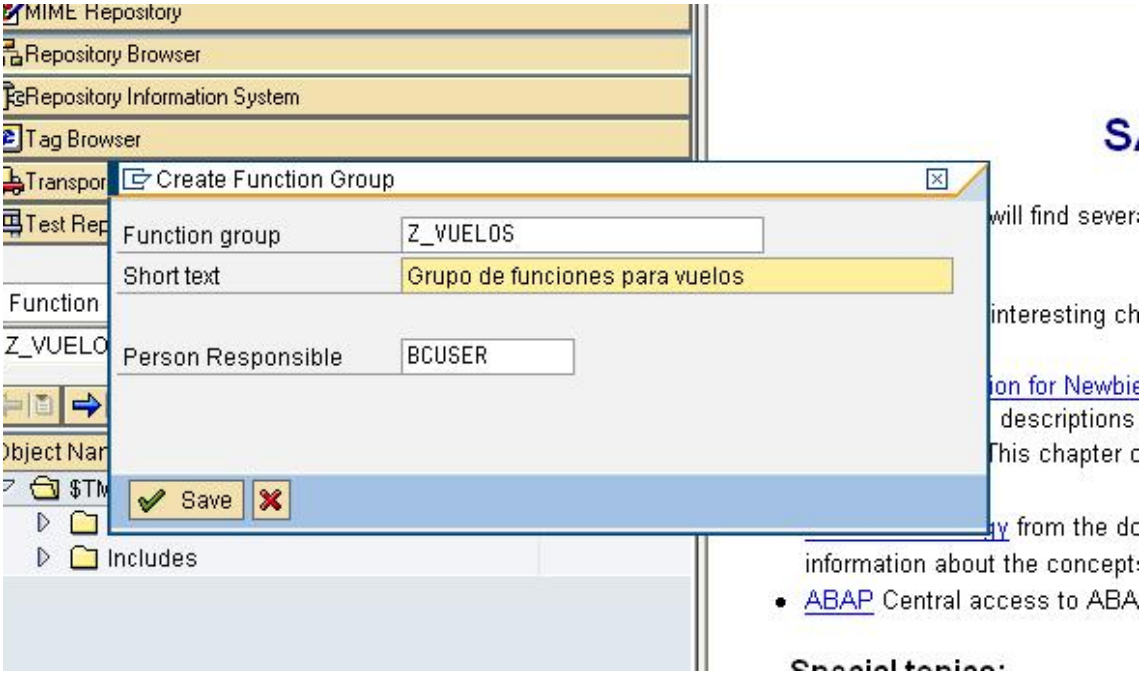


FIG.22- GRUPO DE FUNCIONES.

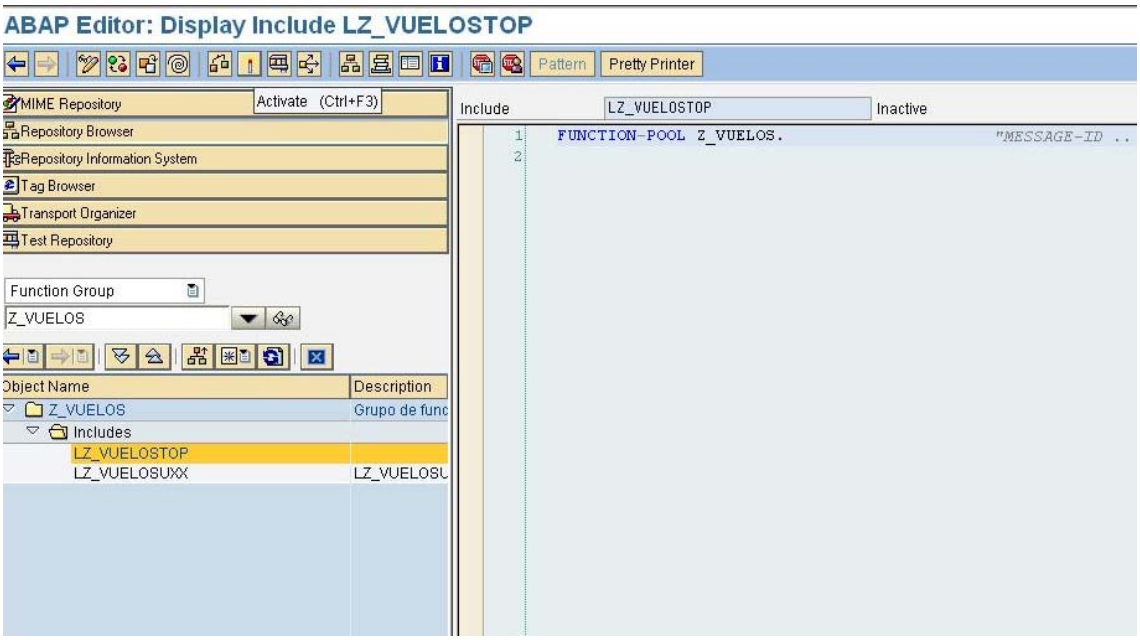


FIG. 23- GRUPO DE FUNCIONES

FIG.24- ACTIVACION DEL GRUPO DE FUNCIONES

Ahora estamos en condiciones de crear nuestro modulo de funciones.

Creación de módulo de funciones

Ejecutamos la transacción SE37.

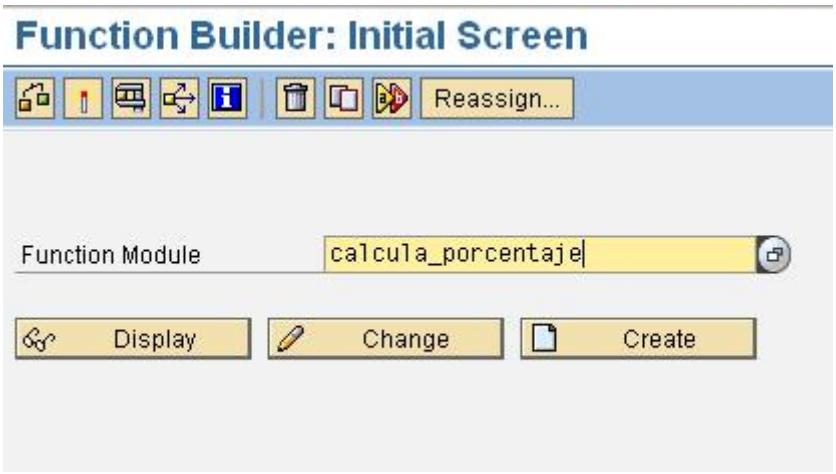


FIG.25- CREACION DE MODULO DE FUNCIONES

En nombre colocamos Z_VUELOS y presionamos Create.

Nos aparecerá una nueva ventana como la de la figura 26. En Function group colocamos el nombre del grupo de funciones que creamos en el paso anterior, y debajo una breve descripción.Como podemos observar en la figura 28, se ha creado un nuevo modulo de funciones, y ahora debemos configurar los parámetros de entrada, salida, excepciones y código.Comenzamos con los parámetros de entrada. Colocamos los datos que observamos en la figura 27.

Para el parámetro de salida, vamos a la solapa Export, e ingresamos los datos tal como indica la figura 28.Idem, para las excepciones, vamos a la solapa correspondiente y repetimos el proceso, como se ve en la fig.29.

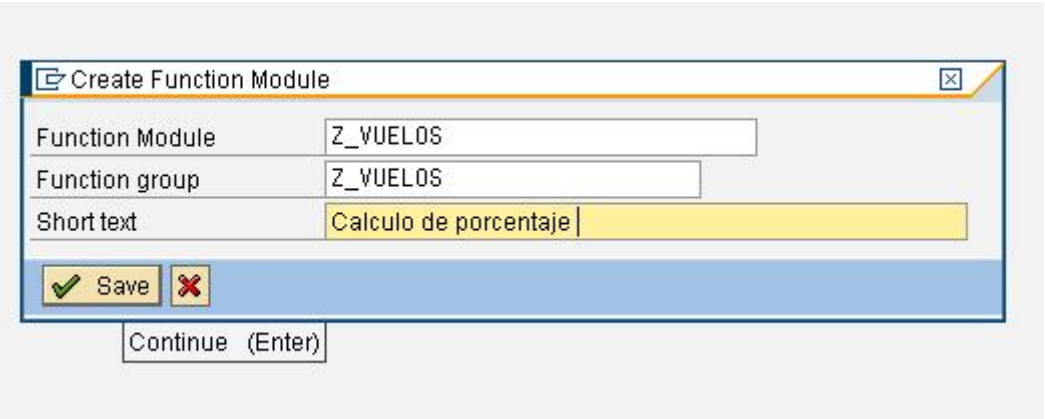


FIG.26- MODULO DE FUNCIONES

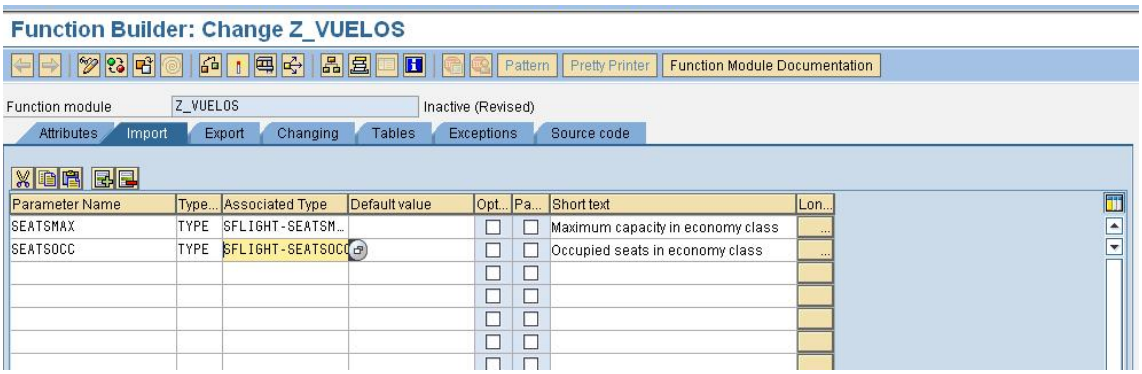


FIG.27- MODULO DE FUNCIONES-IMPORT

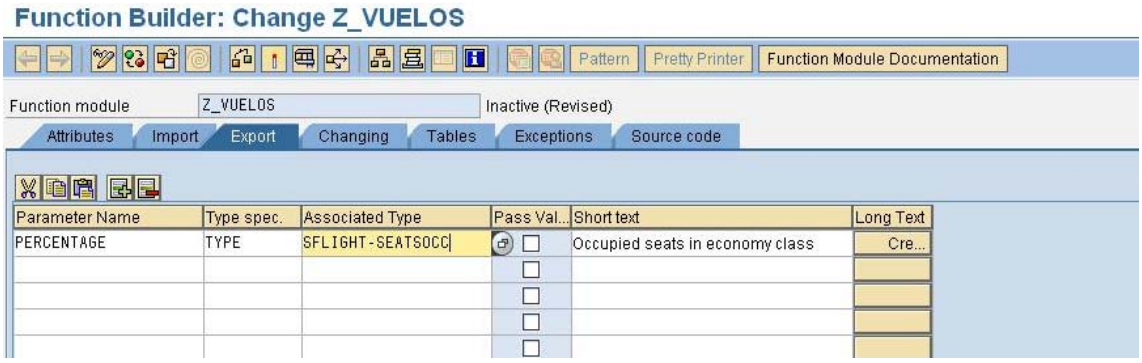


FIG.28- MODULO DE FUNCIONES-EXPORT

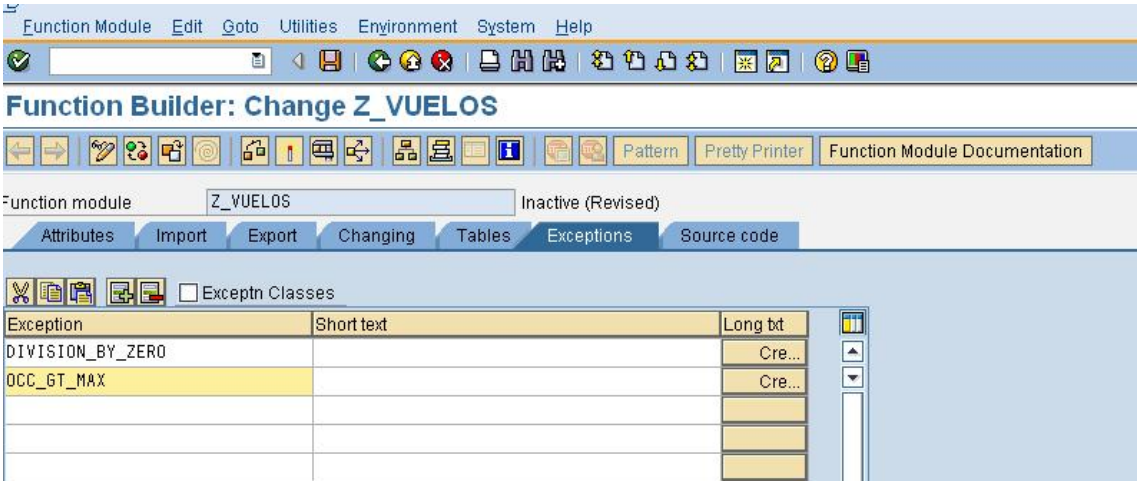


FIG.29- MODULO DE FUNCIONES-EXCEPTIONS

Finalmente, iremos a la solapa de código, e ingresaremos el mismo (Fig. 30).

```
IF SEATSMAX = 0.
  RAISE DIVISION_BY_ZERO.
ELSEIF SEATSOCC > SEATSMAX.
  RAISE OCC_GT_MAX.
ENDIF.

PERCENTAGE = ( SEATSMAX - SEATSOCC ) * 100 / SEATSMAX.
```

El código es el mismo que utilizamos cuando realizamos la rutina en el ejemplo anterior.

Una vez escrito el código, activamos.

Ahora, si en el reporte Z_FUNCT nos posicionamos en '**call function 'Z_VUELOS**' y hacemos doble click sobre el nombre del modulo de funciones (Z_VUELOS), automáticamente navegaremos a la transacción que nos mostrara el modulo de funciones que acabamos de crear.

Si ejecutamos el reporte, veremos que el resultado es el mismo que obtuvimos con el ejemplo anterior, al usar rutinas.

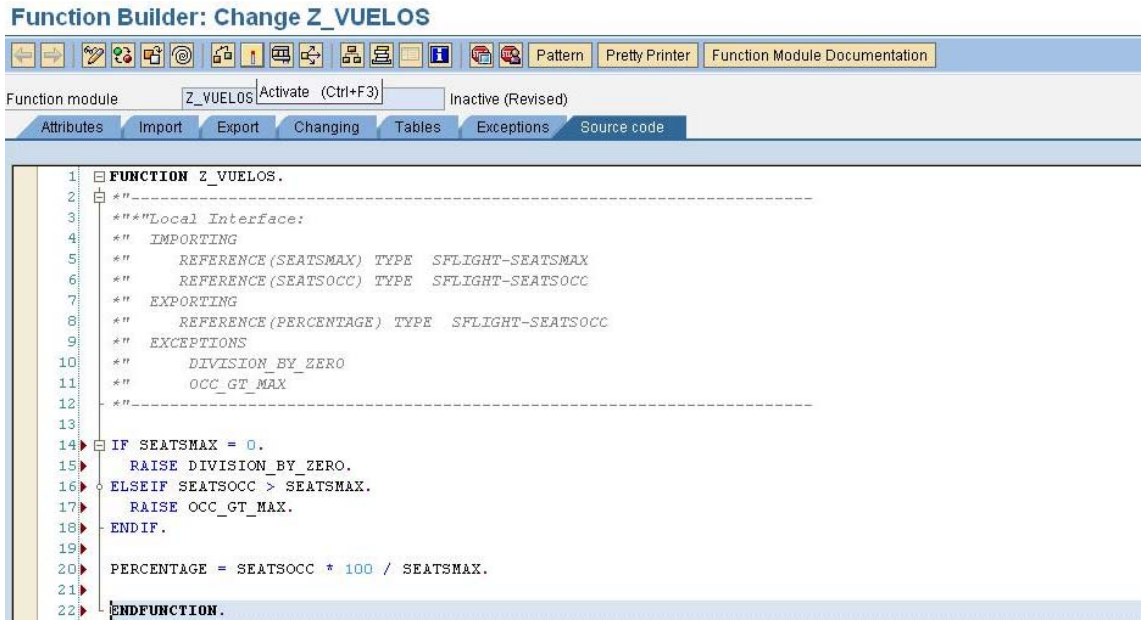


FIG.30- MODULO DE FUNCIONES-CODIGO

Si tuviéramos que escribir nuestro código desde cero, para incorporar la invocación a una función, podemos hacerlo asi:

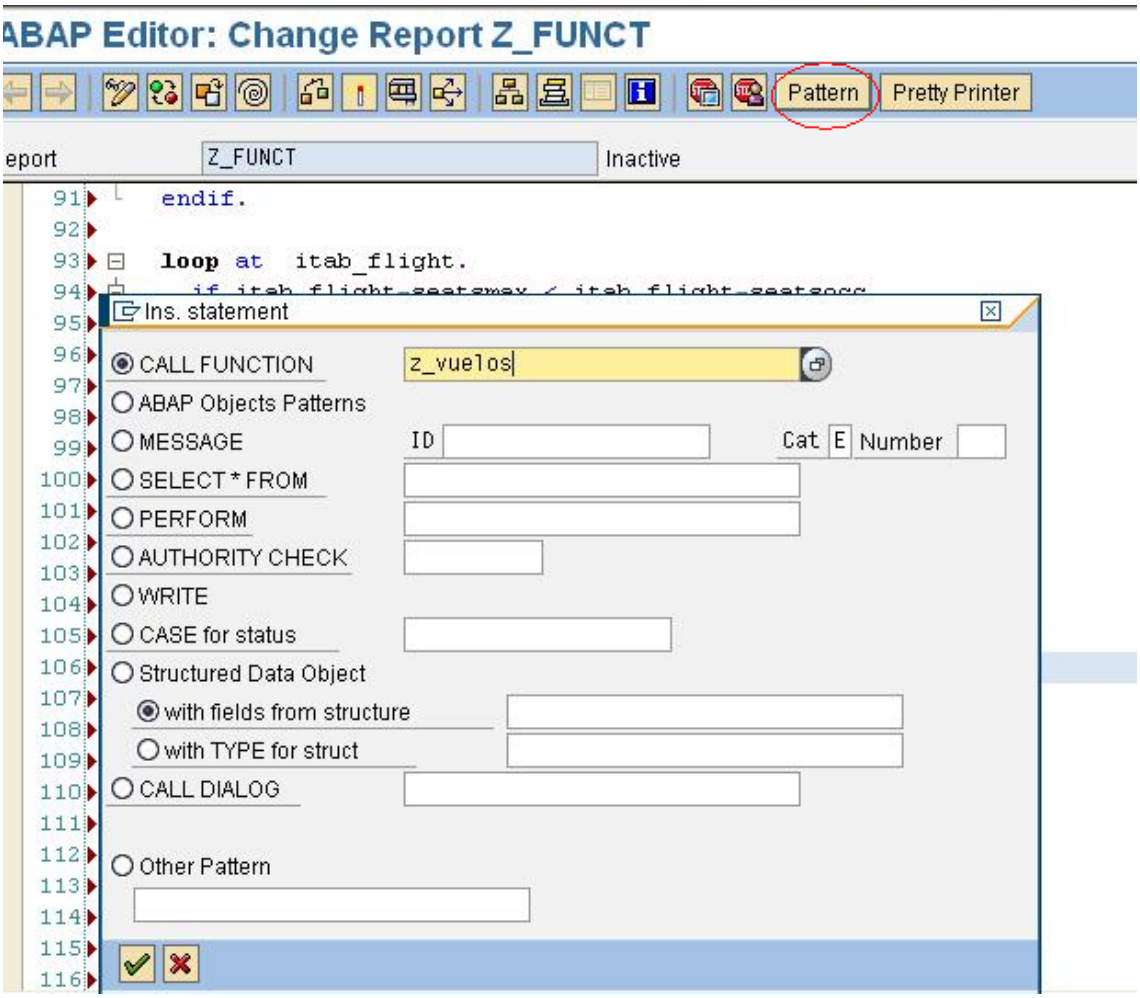


FIG. 31- CALL FUNCTION

Tal como muestra la figura 31, presionamos el botón PATTERN, y aparecerá una ventana para insertar diversos tipo de objetos. Seleccionamos CALL FUNCTION, ingresamos el nombre (o lo buscamos con F4)y presionamos la tecla de ok. Veremos que, en el editor aparece la invocación a la función, con la ventaja ya figuran todos los parámetros, para evitar los errores que podrían aparecer si la escribiéramos a mano.

Pruebas de módulos de funciones

Veamos ahora un poco las pruebas individuales que podemos realizar sobre los módulos de funciones. Si queremos probar el funcionamiento de un modulo de funciones realizado por nosotros, o necesitamos saber si alguno de los estándar de SAP sirve para lo que necesitamos, no es necesario escribir un report e invocarlo desde aquí. Podemos ejecutar los módulos de funciones en forma individual. Veamos como.

Ingresamos a la transacción SE37, y escribimos la función Z_VUELOS.Si presionamos Display, ingresaremos a visualizar el modulo.Ahora presionemos F8, para ejecutar. Aparecerá una pantalla como la de la figura 32.

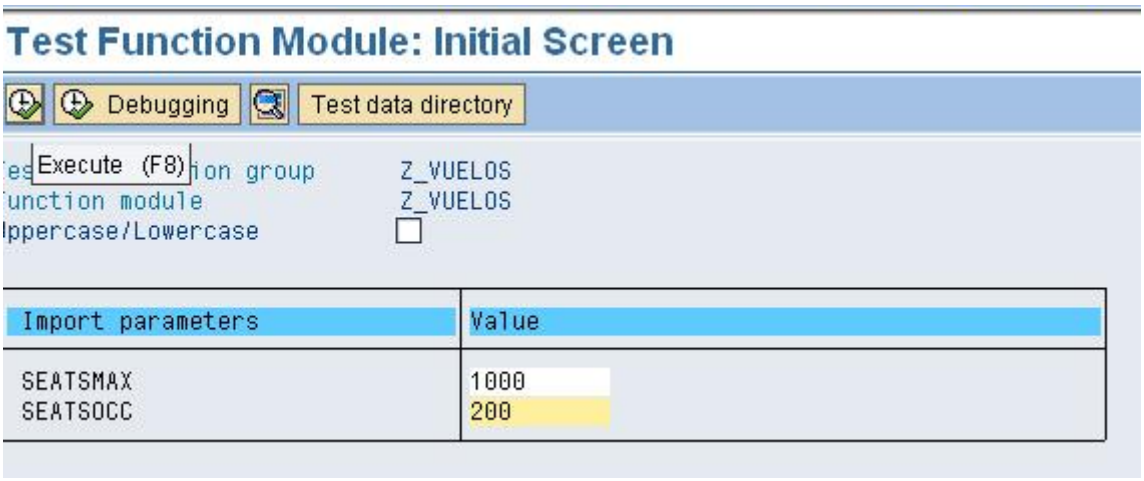


FIG. 32- TEST MODULO DE FUNCIONES

Ingresamos los valores de la figura 32 para los parámetros de entrada, y ejecutamos (F8 o el primer botón, a la izquierda del de Debugging).

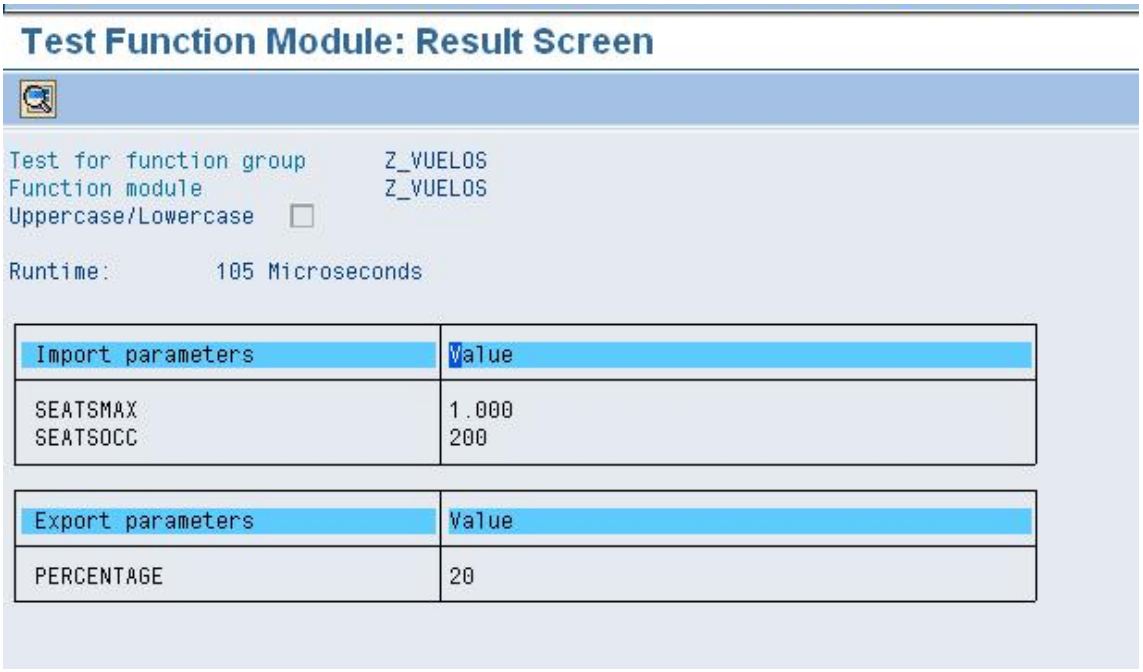


FIG. 33- TEST MODULO DE FUNCIONES

Como vemos en la figura 33, aparecerá la pantalla de resultado. En la parte superior vemos los valores que ingresamos para los parámetros de entrada, y en la parte inferior, PERCENTAGE, el valor que devuelve la función.

Si tuviéramos una función con gran cantidad de parámetros de entrada, o tablas, y quisiéramos dejar grabada la variante de ejecución, hacemos lo siguiente:

Cuando ingresamos los parámetros de entrada, antes de ejecutar, presionamos el botón de guardar

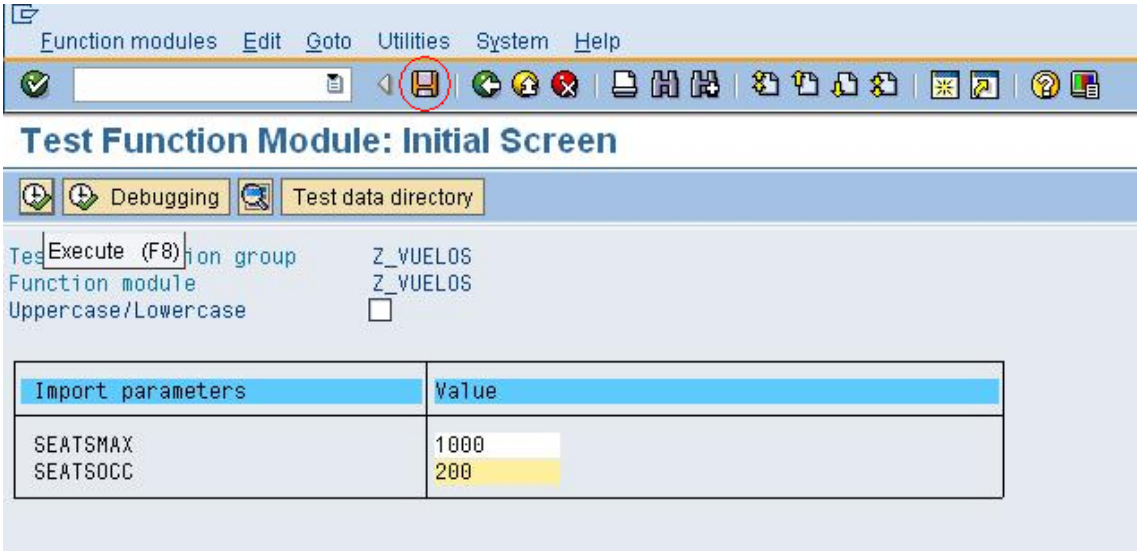


FIG. 34- GENERACION JUEGO DE DATOS

Aparecerá una ventana como se ve en la figura 35

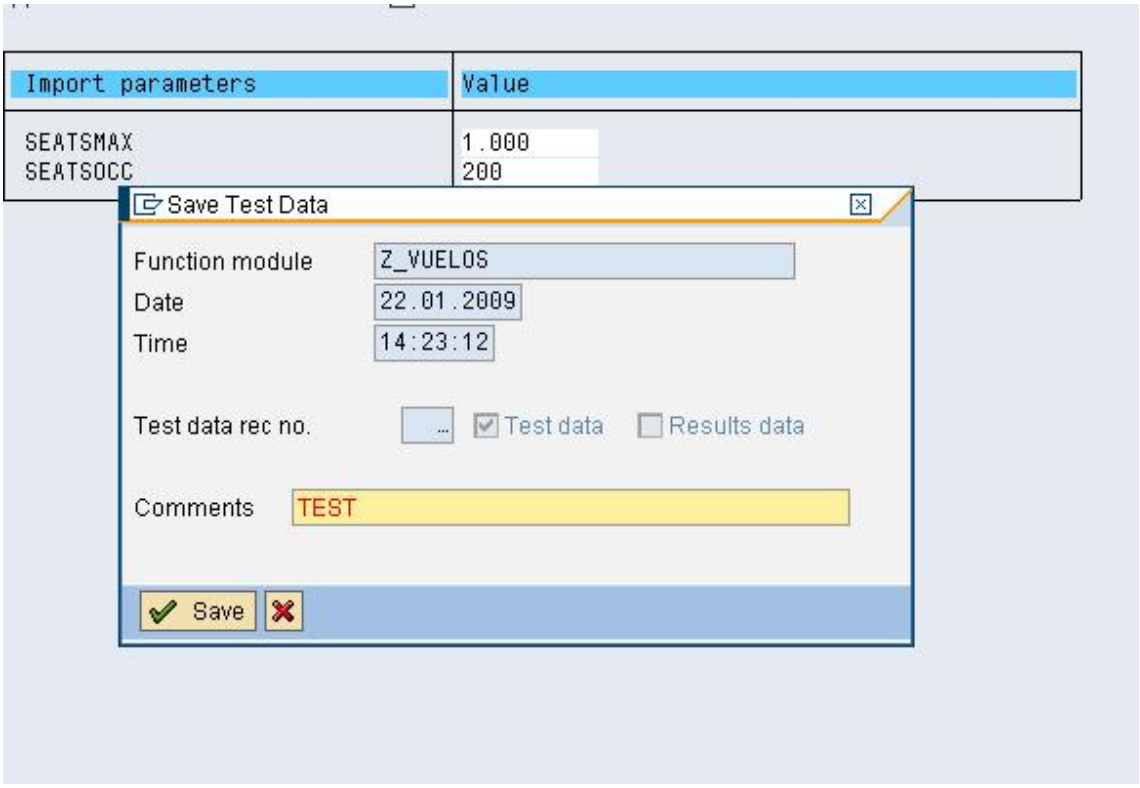


FIG. 35- GENERACION JUEGO DE DATOS

En Comments ingresamos un texto cualquiera, con el que identificaremos el nombre del juego de datos.

Entonces, al momento de ingresar los valores de los parámetros de entrada, si presionamos el botón Test data directory, iremos a una pantalla, que nos mostrara un listado con todos los juegos de datos almacenados. Haciendo doble click sobre cualquiera de ellos (en nuestro caso tendremos solo uno), aparecerán automáticamente los valores de los parámetros de entrada cargados.

FIG. 36- RECUPERACION DE JUEGO DE DATOS

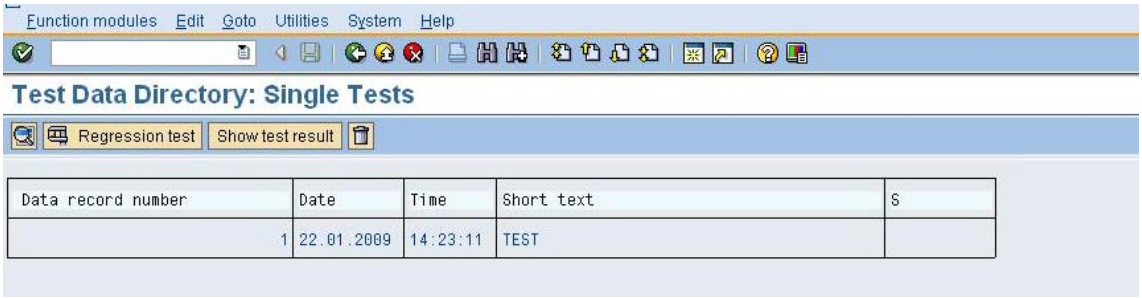


FIG. 37- RECUPERACION DE JUEGO DE DATOS

NOCIONES DE DEBUG

Veamos ahora un tema fundamental: el debugeo. Por medio del mismo, podremos ir revisando nuestros desarrollos, comprobar funcionamiento y detectar fallas.

Hay varias formas de ingresar al modo debugeo.Una de ellas, es indicándole al editor que coloque un break en la parte del código en el cual deseamos que el programa detenga la ejecución y comience el debugeo. Coloquemos un break posicionándonos en la línea 62 del código y presionando el botón correspondiente, como indica la figura 38, y veremos como aparece el indicador de break en dicha línea. Ahora, cuando presionemos la tecla F8 para ejecutar el programa,en primer lugar aparecerá la pantalla de selección.Luego de ejecutar la misma, no aparecerá como hasta ahora la pantalla con el listado, sin que veremos que aparece una pantalla como la de la figura 39.Vemos a la izquierda el código del report, comenzando por el punto de break, y a la derecha, una pantalla con ITAB-SPFLI, de la cual nos ocuparemos en un instante.

Nota: para que aparezcan las variables o tablas que queremos inspeccionar (en este caso ITAB-SPFLI) debemos hacer doble click sobre la misma en el código, o ingresar el nombre manualmente.

Primero veamos, en la parte izquierda, sobre el código, encerrado en la elipse roja, un conjunto de cuatro pulsadores.

Estos son: Single step (F5), Execute(F6),Return (F7) y Continue (F8).

Step: Presionando el mismo, o F5, iremos ejecutando las instrucciones paso a paso.

Execute: Presionando el mismo, o F6, iremos ejecutando las instrucciones paso a paso, pero las rutinas y funciones se ejecutaran como una instrucción, sin ingresar al código de las mismas.

Return (F7): Si ingresamos a una rutina o función, presionando F7, finalizaremos la ejecución de la misma, y volveremos al código principal.

Continue (F8): Con el mismo, saltaremos hasta el próximo break establecido, o , si no hay más breaks, finalizaremos la ejecución del report.

A la derecha del pulsador de Continue, vemos un botón con un círculo y la leyenda Stop. Con el mismo, podemos establecer nuevos breaks en tiempo de ejecución.

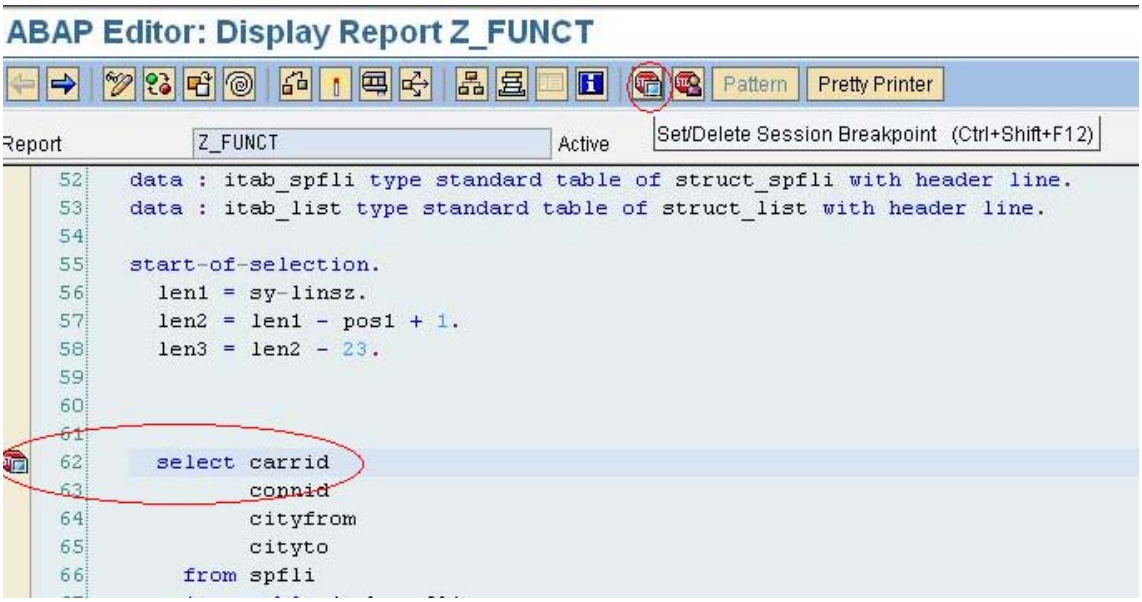


FIG. 38-ACTIVACION DE DEBUG

FIG. 39- PANTALLA DEBUG

Ahora, para visualizar el contenido de la tabla interna ITAB-SPFLI.

Si realizamos la visualización en esta instancia del debugeo, la misma aparecerá vacía, ya que todavía no recuperamos ningún dato. Presionemos F5.Ahora hagamos doble click en el botón de la izquierda de ITAB_SPFLI

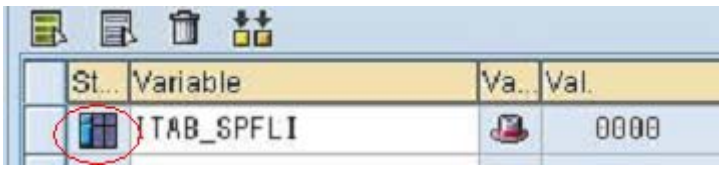


FIG. 40- PANTALLA DEBUG

Iremos a una pantalla, tal como muestra la figura 40, en donde visualizamos el contenido de la tabla interna. Vemos los campos de la tabla, y el contenido de los dos registros.

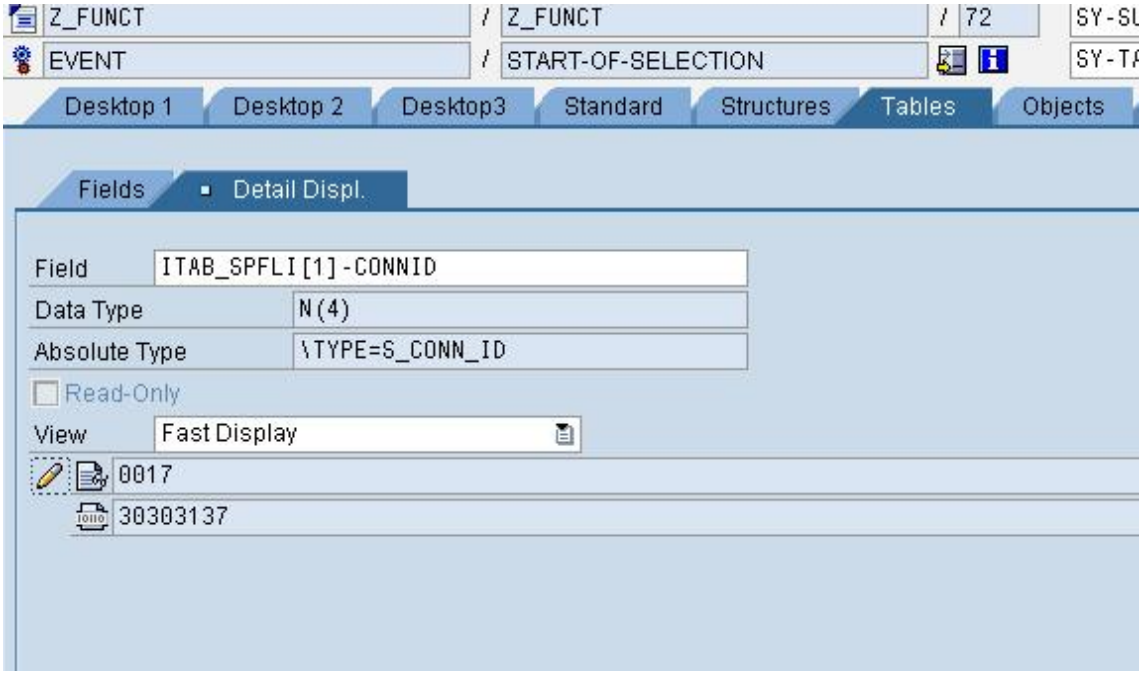


FIG. 41-DEBUG_AMBIO DE VALORES EN TIEMPO DE EJECUCION

Para diversos fines de pruebas en el programa, podemos modificar los valores de variables en tiempo de ejecución. Veamos cómo. Hagamos doble click sobre un valor del primer registro, por ej., sobre el 0017 (connid). Navegaremos hacia una pantalla como la de la figura 41. Vemos que a la izquierda del 0017 hay un lapicito, hacemos doble click sobre el mismo, y pasamos a modo de edición. Veremos que podemos ahora modificar el valor. Ahora, en lugar de un lápiz tendremos unos anteojos, lo que nos indica que presionando sobre los mismos, pasamos nuevamente a modo solo visualización.Navegando hacia atrás, podemos seguir con la ejecución del reporte, con el nuevo valor ingresado.

NOVENO PROGRAMA: ALV

Veremos ahora un programa introduciendo el concepto de ALV. Estas son las siglas de Abap List Viewer. SAP nos provee la misma para poder sacar listados mucho más elegantes y con más funcionalidades que las que tienen los listados vistos hasta ahora con el WRITE. Veremos que en este listado, tendremos pulsadores para poder realizar ordenamientos,filtros,exportación a Excel, podremos variar el layout, etc.

Veremos un reporte muy simple, para comprender los conceptos.

Creemos en el editor Abap el report Z_ALV, y realicemos el upload del archivo con el código correspondiente.

Luego de activarlo, al ejecutar el mismo, veremos que la pantalla de selección es similar a las vistas hasta ahora. Pero al ejecutar (F8), veremos que el listado aparece distinto.

FIG. 42- ALV

Podemos observar que los datos aparecen en un grid, y en la parte superior una serie de pulsadores. Apoyando el mouse en cada uno de ellos, veremos que funcionalidad tienen. Y todos ellos actúan sin que tengamos que escribir una sola línea de código. O sea, no tenemos que codificar para ordenar la lista, exportar a Excel, enviar por mail, etc.

Veamos el código. Comentaremos solo las partes nuevas, referentes al ALV.Aclaremos en primer lugar, que existen diversos tipos de ALV: list, grid, jerárquicos, etc. Y en segundo lugar, que hay también diversas formas de programar ALV: se pueden realizar como se indica en el ejemplo, los catálogos pueden armarse automáticamente, puede utilizarse programación con objetos, etc. Esto excede el alcance de esta introducción, en la web podemos encontrar tutoriales y ejemplos bien explicados de todo esto. Dicho esto, vayamos a nuestro código:

```
report  z_alv.
```

```
*---------------------------------------------------------------------*
* Declaración de las tablas                                           *
*---------------------------------------------------------------------*
```

Es necesario declarar la tabla para el select-options (pantalla de selección)

```
tables: spfli.
```

```
*---------------------------------------------------------------------*
* Types                                                               *
*---------------------------------------------------------------------
```

Declaración necesaria para el ALV

```
type-pools: slis.
```

```
*---------------------------------------------------------------------*
*                    Declaración de tablas internas                   *
*---------------------------------------------------------------------*
data: begin of it_spfli occurs 0,
      carrid like spfli-carrid,
      countryfr like spfli-countryfr,
      cityfrom  like spfli-cityfrom,
      countryto like spfli-countryto,
      cityto    like spfli-cityto,
      end of it_spfli.
```

```
*---------------------------------------------------------------------*
*                    Datos ALV                                        *
*---------------------------------------------------------------------
```

Declaraciones necesarias para el ALV.

```
data: i_fieldcat  type slis_t_fieldcat_alv,
      i_layout    type slis_layout_alv,
      v_repid     like sy-repid.


*---------------------------------------------------------------------*
* Pantalla de selecciòn                                               *
*---------------------------------------------------------------------*

selection-screen begin of block block with frame title text-001.
select-options: s_carrid for spfli-carrid obligatory.
selection-screen end of block block.


start-of-selection.
```

```
  perform f_obtiene_datos.
  perform f_generar_alv.



*&---------------------------------------------------------------------*
*&      Form  f_obtiene_datos                                          *
*&---------------------------------------------------------------------*
*       Busqueda de datos                                             *
*----------------------------------------------------------------------*
form f_obtiene_datos .

  clear it_spfli.
  refresh it_spfli.

  select carrid
         countryfr
         cityfrom
         countryto
         cityto
    from spfli
    into table it_spfli
    where carrid in s_carrid.


  if sy-subrc ne 0.
    message e208(00) with text-002.
  endif.



endform.                    " f_obtiene_datos


*&---------------------------------------------------------------------*
*&      Form  f_generar_alv                                            *
*&---------------------------------------------------------------------*
*     Genera el alv para la selección de interior-interior influencia  *
*----------------------------------------------------------------------*
form f_generar_alv.

* Preparamos parametros de funcion ALV
  perform f_preparar_parametros.
```

```
* Genera la alv
  perform f_mostrar_alv.

endform.                    " f_generar_alv

*
*&---------------------------------------------------------------------*
*&      Form  f_mostrar_alv
*&---------------------------------------------------------------------*
*       Genera alv
*----------------------------------------------------------------------*
form f_mostrar_alv.

  constants:
            lc_status     type slis_formname value 'F_PF_STATUS_SET',
            c_ucomm   type slis_formname value 'F_USER_COMMAND'.

  v_repid = sy-repid.
```

Función que saca por pantalla el listado (ALV Grid).

Vemos que al modulo de funciones le pasamos una tabla interna que tiene el layout(i_layout) del ALV, otra tabla interna con el catalogo (i_fieldcat), y la tabla interna con los datos a visualizar (it_spfli).Más abajo veremos que es cada cosa.

```
  call function 'REUSE_ALV_GRID_DISPLAY'
    exporting
      i_callback_program = v_repid
      is_layout          = i_layout
      it_fieldcat        = i_fieldcat[]
    tables
      t_outtab           = it_spfli
    exceptions
      program_error      = 1
      others             = 2.
  if sy-subrc <> 0.
    message id sy-msgid type sy-msgty number sy-msgno
            with sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  endif.

endform.                    " F_MOSTRAR_DOCUMENTOS
```

```
*&---------------------------------------------------------------------*
*&      Form  F_PREPARAR_PARAMETROS                                    *
*&---------------------------------------------------------------------*
*       Prepara catalogo y opciones de visualización del ALV          *
*----------------------------------------------------------------------*
*      -->P_L_I_FIELDCAT  CAtalogo de campos a mostrar                 *
*      -->P_L_I_LAYOUT    Opciones de visualizacion                    *
*----------------------------------------------------------------------*
form f_preparar_parametros.

* Prepara columnas a mostrar
  perform f_preparar_catalogo.

* Set report layout
  perform f_preparar_layout.

endform.                    " F_PREPARAR_PARAMETROS
```

Aquí preparamos el catalogo. En el mismo indicamos que campos mostraremos, que leyenda aparecerá en la cabecera de cada columna, etc.

```
*&---------------------------------------------------------------------*
*&      Form  f_preparar_catalogo
*&---------------------------------------------------------------------*
*       Prepara columnas a mostrar en ALV                             *
*----------------------------------------------------------------------*
*      -->i_fieldcat  Catalogo ALV                                    *
*----------------------------------------------------------------------*
form f_preparar_catalogo.

  v_repid = sy-repid.

  data: w_i_fieldcat type slis_fieldcat_alv.

* Load columns details to report
  refresh i_fieldcat.
```

En i_fieldcat vamos guardando los nombres de los campos que se visualizarán (tabla interna w_i_fieldcat-tabname y campo w_i_fieldcat-fieldname). Si hacemos doble click sobre i_fieldcat, navegaremos hasta la parte del report que tiene la declaración de la misma

data: i_fieldcat type slis_t_fieldcat_alv,

y si hacemos click sobre slis_t_fielcat_alv, veremos la composición de dicho tipo.

```
* Pais origen
  clear w_i_fieldcat.
  w_i_fieldcat-fieldname     = 'COUNTRYFR'.
  w_i_fieldcat-tabname       = 'IT_SPFLI'.
  w_i_fieldcat-seltext_l     = text-003.
  w_i_fieldcat-seltext_m     = text-003.
  w_i_fieldcat-seltext_s     = text-003.
  append w_i_fieldcat to i_fieldcat.


* Ciudad origen
  clear w_i_fieldcat.
  w_i_fieldcat-fieldname     = 'CITYFROM'.
  w_i_fieldcat-tabname       = 'IT_SPFLI'.
  w_i_fieldcat-seltext_l     = text-004.
  w_i_fieldcat-seltext_m     = text-004.
  w_i_fieldcat-seltext_s     = text-004.
  append w_i_fieldcat to i_fieldcat.

* Pais destino
  clear w_i_fieldcat.
  w_i_fieldcat-fieldname     = 'COUNTRYTO'.
  w_i_fieldcat-tabname       = 'IT_SPFLI'.
  w_i_fieldcat-seltext_l     = text-005.
  w_i_fieldcat-seltext_m     = text-005.
  w_i_fieldcat-seltext_s     = text-005.
  append w_i_fieldcat to i_fieldcat.

* Ciudad destino
  clear w_i_fieldcat.
  w_i_fieldcat-fieldname     = 'CITYTO'.
  w_i_fieldcat-tabname       = 'IT_SPFLI'.
  w_i_fieldcat-seltext_l     = text-006.
  w_i_fieldcat-seltext_m     = text-006.
  w_i_fieldcat-seltext_s     = text-006.
  append w_i_fieldcat to i_fieldcat.



endform.                    "f_preparar_catalogo
```

```
*&---------------------------------------------------------------------*
*&      Form  f_preparar_layout                                        *
*&---------------------------------------------------------------------*
*       Prepara Detalles de visualizacion del ALV                      *
*----------------------------------------------------------------------*
*      <--P_I_LAYOUT  Estructura con detalles visuales de ALV          *
*----------------------------------------------------------------------
```

Aquí configuramos el layout. Igual que antes, podemos investigar la composición haciendo doble click sobre i_layout y navegando.

```
form f_preparar_layout.
  constants: c_x value 'X'.
  constants: c_sel(3) type c value 'SEL'.

  i_layout-reprep          = c_x.
  i_layout-zebra           = c_x.
  i_layout-box_tabname     = 'I_SPFLI'.

endform.                    " f_preparar_layout
```

O sea, en principio podemos realizar un simple ALV usando la estructura de este programa. Recapitulando, lo primero que hacemos es la selección de datos, luego completar catalogo, layout, y la invocación del modulo de funciones que desplegará el ALV.

TRANSACCIONES

Para finalizar, veremos como crear una transacción.Hasta ahora, los programas que creamos los ejecutamos desde el editor Abap. Pero no es lo habitual que los usuarios finales tengan acceso a esta transacción. Entonces, lo que hacemos es crear una transacción para nuestro reporte. Y para ejecutarla, en vez de ingresar a la transacción SE38 y desde allí colocar el nombre del programa, directamente ingresaremos el nombre que le demos a nuestra transacción. Veámoslo en forma práctica.

Creando una transacción-Transacción SE93

Ingresemos a la transacción SE93, coloquemos en el Transaction code Z_ALV y presionemos el botón create. Aparecerá una ventana como la de la figura 43. Introducimos una descripción en Short text. Podemos crear distintos tipos de transacciones. En nuestro caso elegiremos la segunda opción, que es la de reportes (programa y pantalla de selección) y presionamos el botón con el tilde verde de ok. Iremos a una pantalla como la de la figura 44. En Program debemos introducir el nombre del programa, en nuestro caso es el Z_ALV creado anteriormente. Tildamos las opciones del GUI support y grabamos.

Con esto, hemos creado la transacción. Ahora, si vamos a la pantalla inicial, y donde ingresamos los códigos de transacción, tipeamos (en vez de SE38) Z_ALV, veremos que se ejecuta directamente el programa.

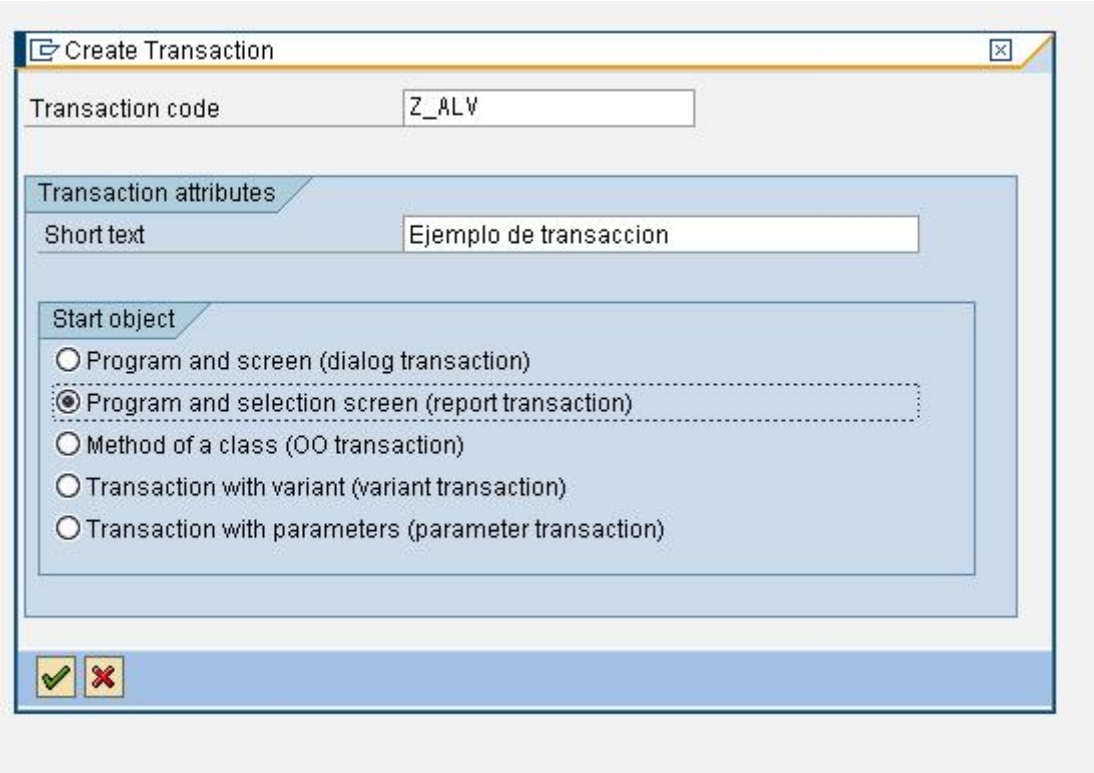


FIG. 43- CREACION DE TRANSACCION

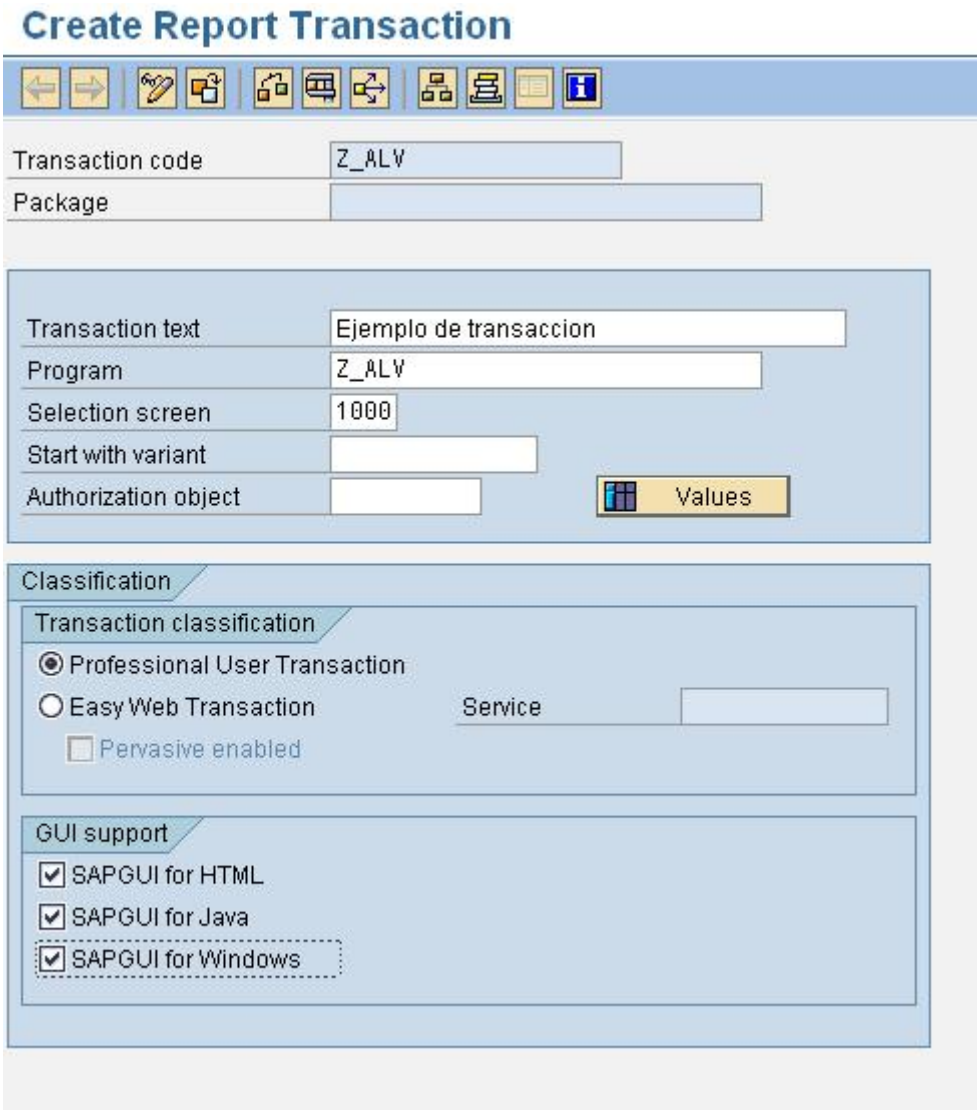


FIG. 44- CREACION DE TRANSACCION