

PROGRAMACIÓN EN DIALOGO

INDICE

1. OBJETIVO.....	5
2. EXPECTATIVAS	6
3. TÉCNICAS BÁSICAS DE PROGRAMACIÓN EN DIÁLOGO	7
3.1. Componentes principales	7
3.2. Screen Painter.....	9
3.2.1. <i>Module Pool</i>	9
3.2.2. <i>Creación de Programas</i>	10
3.2.3. <i>Definición de Pantallas</i>	11
3.2.4. <i>Diseño de Pantallas</i>	12
3.2.5. <i>Creando objetos en la pantalla.</i>	13
3.2.6. <i>Lista de vistas de campos</i>	14
3.2.7. <i>Creando objetos desde el diccionario de datos.</i>	15
3.2.8. <i>Definiendo los atributos individuales de cada campo.</i>	16
3.2.9. <i>Definiendo campos en el Module Pool.</i>	18
3.3. Definición de la Flujo de Control	19
3.3.1. <i>Introducción a la Lógica de Proceso.</i>	19
3.3.2. <i>Creación de Módulos ABAP/4</i>	20
3.3.2.1. <i>Process Before Output (PBO).</i>	21
3.3.2.2. <i>Process After Input (PAI).</i>	21
3.3.3. <i>Definiendo la llamada (Códigos de Transacción).</i>	21
3.3.4. <i>Leyendo códigos de función en programas.</i>	22
4. TÉCNICAS ESPECIALES "SCREEN & MENU PAINTER"	23
4.1. La validación de los datos de entrada.....	23
4.1.1. <i>Verificación automática</i>	23
4.1.1.1. <i>Verificación de formato</i>	23
4.1.1.2. <i>Verificación de campos obligatorios</i>	24
4.1.1.3. <i>Verificación de llaves foráneas</i>	24
4.1.1.4. <i>Verificación de valores fijos</i>	24
4.1.2. <i>Verificación manual en Module Pool.</i>	25
4.1.3. <i>Mensajes en pantalla</i>	27
3.4.3.1. <i>Mensaje de Error</i>	27
3.4.3.2. <i>Mensaje de Advertencia (Warning)</i>	28
3.4.3.3. <i>Mensaje de Información</i>	28
3.4.3.4. <i>Mensaje de Buen resultado</i>	29
3.4.3.5. <i>Mensaje de Interrupción (Abend)</i>	29
4.2. Secuencia dinámica de pantallas	30
4.2.1. <i>Introducción.</i>	30
4.2.2. <i>Configuración dinámica de la siguiente pantalla</i>	30
4.2.3. <i>Inserción de una o más pantallas</i>	31
4.3. Ejecución condicionada de módulos	33
5. MENÚ PAINTER.....	35

5.1.	DISEÑO DE MENÚS (Menú Painter).....	35
5.1.1.	<i>Introducción.....</i>	35
5.1.2.	<i>Taclas de Función.....</i>	36
5.1.3.	<i>Los 'Pushbuttons'.....</i>	37
5.1.4.	<i>La Barra de Menús.....</i>	38
5.1.5.	<i>Otras utilidades del Menú Painter.....</i>	39
5.1.5.1.	<i>Activación de Funciones.....</i>	39
5.1.5.2.	<i>'FastPaths'.....</i>	39
5.1.6.	<i>Títulos de Menu.....</i>	39
5.1.7.	<i>Prueba, Chequeo y Generación de Status.....</i>	39
5.1.8.	<i>Menús de ámbito o de área.....</i>	40
6.	ACTUALIZACIÓN ASÍNCRONA.....	41
6.1.	Concepto de transacción.....	41
6.1.1.	<i>Transacción de Base de Datos (DB LUW).....</i>	41
6.1.2.	<i>Transacción SAP.....</i>	41
6.1.3.	<i>Transacción SAP y Transacción DB.....</i>	42
6.1.4.	<i>Transacción SAP y Actualizaciones Asíncronas.....</i>	42
6.2.	Concepto de Actualización Asíncrona.....	44
6.2.1.	<i>Programa de dialogo y módulo de función para actualización.....</i>	44
6.2.2.	<i>Modulo de Función de Actualización.....</i>	45
6.2.3.	<i>Programa de diálogo y tabla de registro.....</i>	45
6.2.4.	<i>Rollback en los programas de Dialogo: Borrando marcas de Actualización.....</i>	46
6.2.5.	<i>Rollback en el Programa de Actualización.....</i>	46
6.2.6.	<i>PERFORM <subrutina> ON COMMIT.....</i>	47
7.	CONCEPTO DE BLOQUEO DE SAP.....	49
7.1.	Utilización de bloqueos.....	49
7.1.1.	<i>Bloqueo de Base de Datos.....</i>	49
7.1.2.	<i>Introducción al Concepto de Bloqueo de SAP.....</i>	50
7.1.3.	<i>Objetos de Bloqueo SAP.....</i>	51
7.1.4.	<i>Modulo de Función Enqueue / Dequeue.....</i>	51
7.1.5.	<i>Llamando Módulos de Bloqueo.....</i>	52
7.1.6.	<i>Tabla de Bloqueos.....</i>	52
7.1.7.	<i>Características Especiales con ENQUEUE.....</i>	53
8.	MODIFICACIÓN DINÁMICA DE PANTALLAS.....	54
8.1.	Modificación dinámica.....	54
8.1.1.	<i>Introducción.....</i>	54
8.1.2.	<i>Atributos de campos Modificables.....</i>	54
8.1.3.	<i>Atributos: Modificación de grupos.....</i>	55
8.1.4.	<i>Programa.....</i>	55
9.	TABLAS DE CONTROL.....	57
9.1.	Características del Control de Tabla.....	57
9.2.	Principios para el control de Tabla.....	58

9.2.1.	Llenado.....	58
9.2.2.	Creación en modo Gráfico.....	59
9.2.3.	Creando Tablas de control (Fullscreen Alfanumérico).....	59
9.2.4.	Definición de una tabla de control en "Module Pool".....	60
9.2.5.	Flujo lógico de la tabla de control.....	61
9.2.6.	Modificación.....	62
9.2.7.	Control de páginas.....	63
9.2.8.	Posición del cursor.....	65
10.	VINCULACIÓN CON LOS COMPONENTES DE PROGRAMAS.....	66
10.1.	Diálogos (pantallas) y procedimientos de listas.....	66
10.1.1.	Ramificación de una lista a un procesamiento de diálogo.....	66
10.1.2.	Ramificación de un diálogo a un procesamiento de listas.....	67
10.1.3.	Lista en una caja de dialogo modal.....	68
10.1.4.	Llamado a un reporte.....	68
10.1.5.	Declaración "SUBMIT".....	69
10.1.6.	Llamado a una transacción.....	69
10.1.7.	Transferencia de datos entre programas (ABAP/4 Memory).....	70
11.	FUNCIONES AUTOMÁTICAS DE AYUDA PROGRAMADAS.....	72
	Funcionalidad F1 y F4.....	72
11.1.1.	Funciones de Ayuda Automática.....	72
11.1.2.	Programadas.....	73
11.1.3.	Extensión de la funcionalidad "F4" "HELP VIEW".....	73
11.1.4.	Creación "HELP VIEW".....	74
11.1.5.	Búsqueda "MATCHCODE".....	75
11.1.5.1.	Definición "MATCHCODE OBJECT".....	76
11.1.5.2.	Selección de campos.....	76
11.1.5.3.	Definición "MATCH CODE ID".....	77
11.1.5.4.	Selección de campos.....	78
11.1.6.	Programación de la funcionalidad de "F1" y "F4".....	79
11.1.7.	Flujo de los eventos "POH" y "POV".....	80
11.1.8.	Modulos de funciones.....	81
11.1.9.	Transportación de campos desde y hacia la pantalla.....	81
11.1.10.	Determinación del contenido de los campos de pantalla (POV, POH).....	82
11.1.11.	Transportación de valores a los campos de pantalla (POV).....	82
11.1.12.	Visualización de los valores de entrada para el campo de la tabla (POV).....	82
ANEXO 1	ABAP EDITOR.....	84
	<u>COMMANDOS DE CABECERA.....</u>	<u>85</u>
	<u>COMANDOS DE LINEA:.....</u>	<u>87</u>

1. Objetivo

Los objetivos principales del presente curso, son los siguientes:

- Proporcionar los conocimientos básicos de la programación en diálogo
- Técnicas para mantenimiento de pantallas dinámicas
- Técnicas para mantenimiento de funciones de ayuda automática
- Técnicas para mantenimiento de menús dinámicos

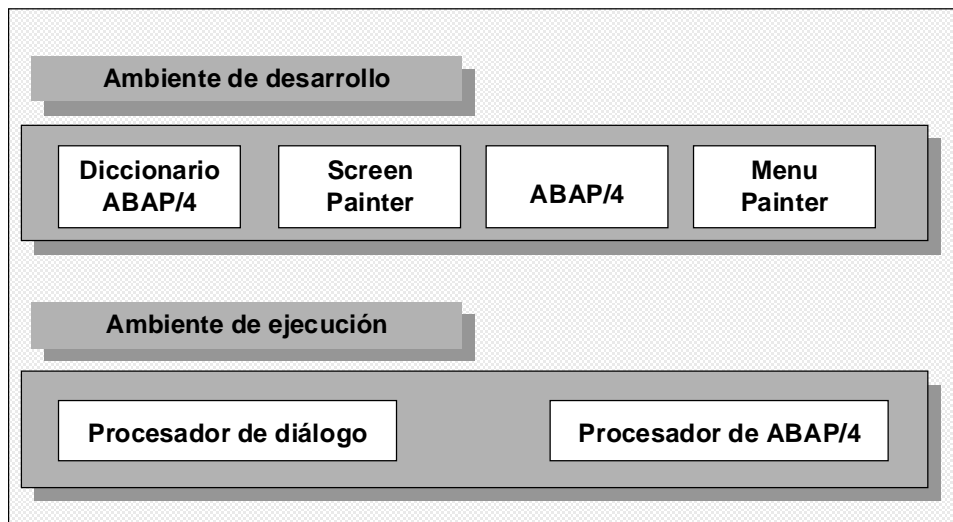
2. Expectativas

Al finalizar el presente curso, se espera que el participante podrá describir:

- Los conceptos básicos para la programación en diálogo
- Las técnicas para definición de pantallas
- Las técnicas para definición de menús

3. Técnicas Básicas de programación en diálogo

3.1. Componentes principales



El Screen Painter y el Menu Painter se utilizan para crear y diseñar plantillas de pantalla y programas de pantalla.

El procesamiento lógico de las pantallas se define en un programa ABAP/4 (conocido como module pool).

Para crear una transacción, será necesario la generación de una serie de objetos de desarrollo. Cada transacción puede dividirse en varias pantallas, cada una de las cuales puede utilizar distintos menús y todo ello controlado por un programa en ABAP/4 denominado **Module Pool**, que controla el flujo de la transacción y realiza las acciones necesarias para cumplir la funcionalidad de la transacción.

Por lo tanto los pasos a seguir para el desarrollo de transacciones será:

- 1º Crear el programa ABAP/4 (Module Pool).
- 2º Definir las pantallas que intervienen en la transacción con el Screen Painter.
- 3º Especificando que datos aparecen en pantalla y de que forma, además de una lógica de proceso de cada pantalla.
- 4º Definir los menús con el Menú Painter.
- 5º Especificando el contenido de los menús Pop-up, las teclas de función y los botones de comandos que se pueden utilizar.
- 6º Definir el Flujo de pantallas en el Module Pool.

- 7º Programar, en el Module Pool, los **módulos** de cada pantalla, es decir lo que debe hacer cada pantalla. Programando las acciones ha realizar en tiempo de **PBO ('Process Before Output')**, antes de que aparezcan los datos de la pantalla y en tiempo de **PAI ('Process After Input')**, después de que se hayan introducido los datos en los campos de entrada.
- 8º Crear el **código de transacción**.

Herramientas -> Case -> Desarrollo -> Transacciones.

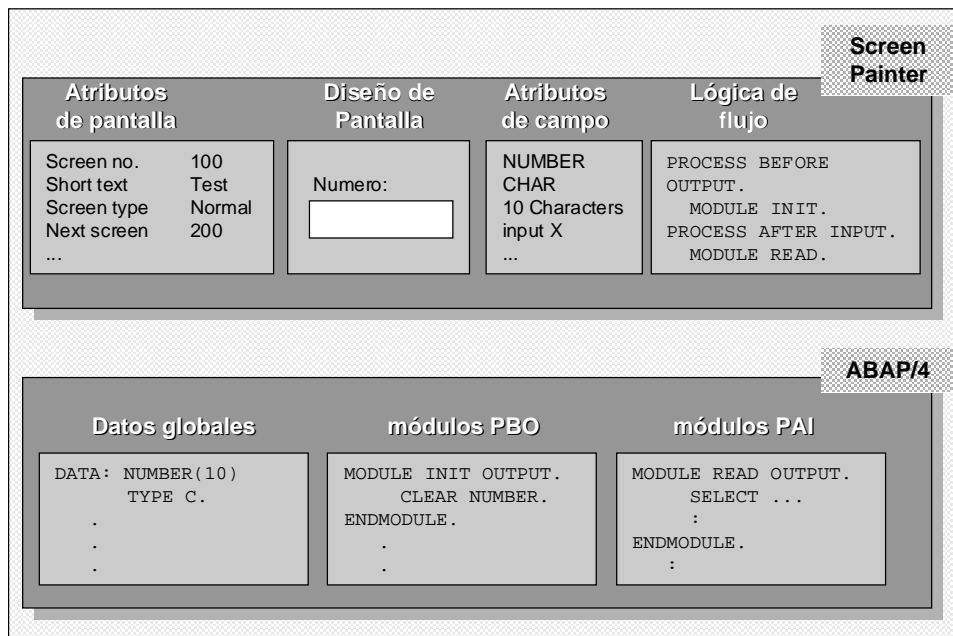
Indicándole: el tipo de transacción, la descripción de la transacción, el nombre del programa ABAP/4 (Module Pool), el número de la primera pantalla, y opcionalmente un objeto de verificación para ejecutar la transacción.

Las estructuras de datos estan definidas en el Diccionario ABAP/4. Desde el programa ABAP/4 se pueden acceder estas estructuras y utilizarlas para definir los campos de pantallas.

El procesador de diálogo controla el flujo de un programa de diálogo.

La programación de diálogo necesita técnicas especiales de codificación en ABAP/4, además de herramientas específicas, como son un editor de pantallas (**Screen Painter**) y un editor de superficies (**Menú Painter**).

3.2. Screen Painter



Para crear una pantalla se deben seguir los siguientes pasos:

- Definir las características básicas de una pantalla (atributos de pantalla)
- Diseñar el formato de pantalla (en el editor fullscreen)
- Definir los atributos de campos (lista de campos)
- Escribir el flujo de pantalla (lógica de flujo)

Los componentes más importantes del programa ABAP/4 se encuentran en los siguientes objetos:

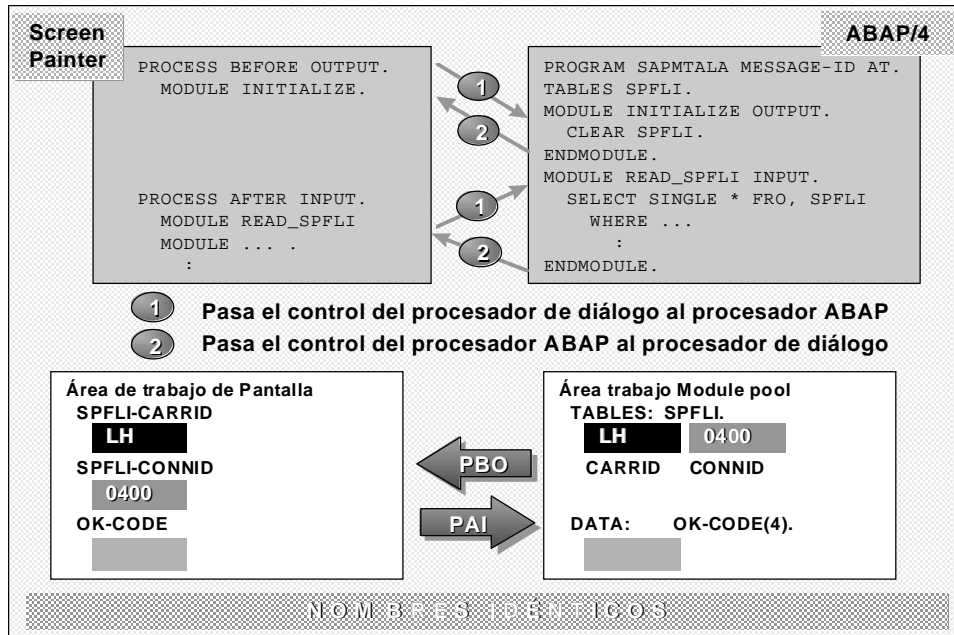
- Datos globales o estructuras de Diccionario en el programa "Top Include" (declaraciones de datos)
- Módulo PBO (Proceso antes de presentar pantalla)
- Módulo PAI (Proceso después de entrada de datos)
- Subrutinas (solo si son requeridas).

3.2.1. Module Pool

El flujo lógico de pantalla está dividido en dos eventos para cada pantalla:

- El evento "Process Before Output" (PBO) es ejecutado antes de que la pantalla sea desplegada
- El evento Process After Input (PAI) es ejecutado después de que el usuario ha presionado la tecla Enter.

El sistema procesa los módulos de un evento en forma secuencial.

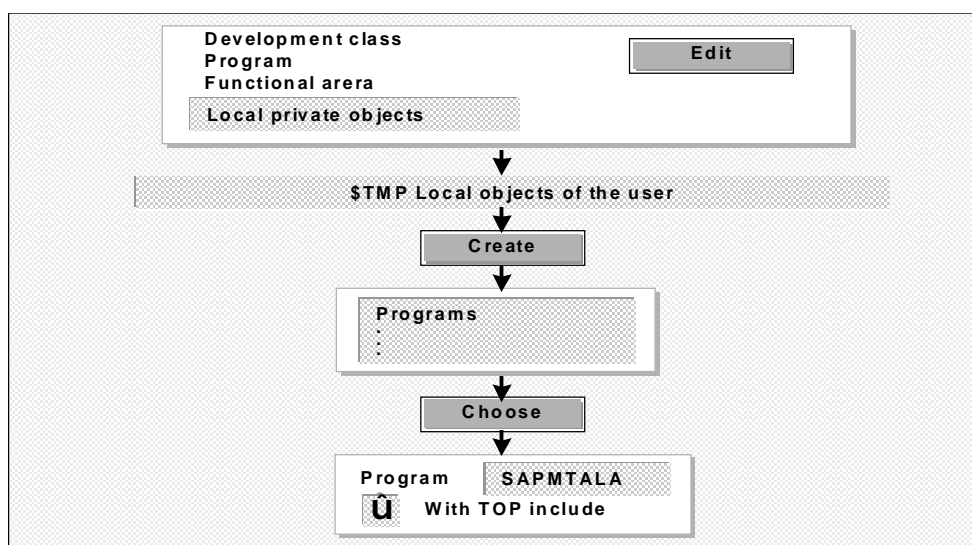


En cada módulo el control pasa del procesador de diálogo al procesador ABAP/4. Después del procesamiento el control es regresado al procesador de diálogo.

Cuando todos los módulos de PBO han sido procesados, el contenido de los campos del área de trabajo ABAP/4 es copiado a un campo con nombre idéntico en el área de trabajo de pantalla.

Antes de que el módulo PAI sea procesado, el contenido de los campos del área de trabajo de pantalla es copiado a los campos nombrados idénticamente en el área de trabajo de ABAP/4.

3.2.2. Creación de Programas



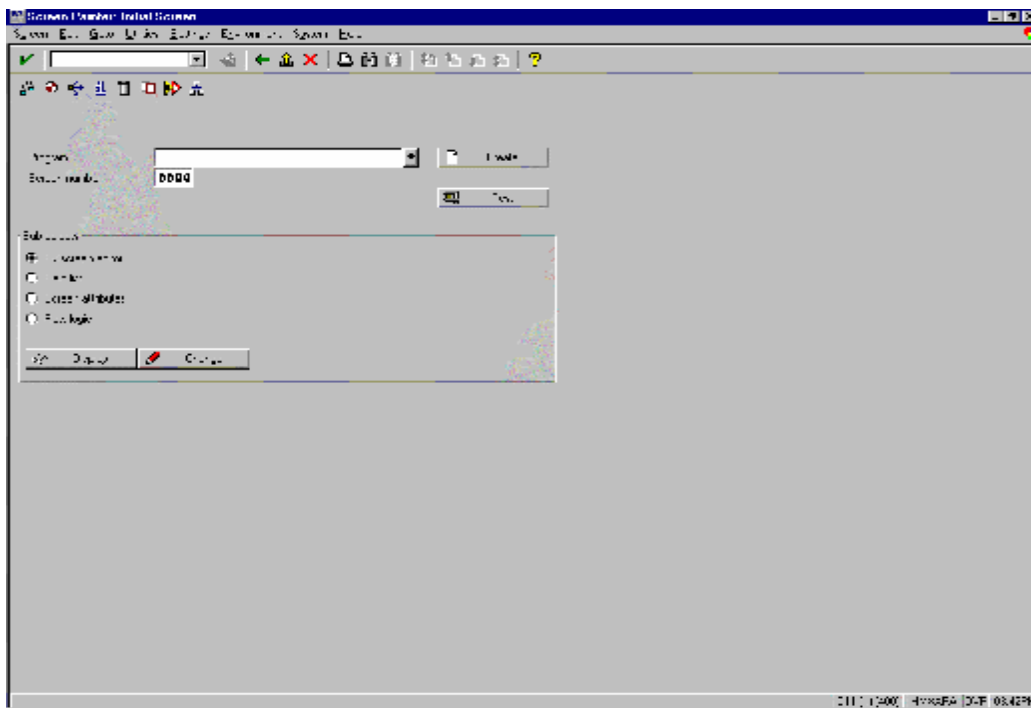
Para crear un programa (ABAP/4 module pool) se usa el ABAP/4 Development Workbench. Un programa de diálogo del cliente debe tener el formato SAPMZXXX ó SAPMYXXX.

Inicialmente, el programa incluido TOP contiene solo la instrucción PROGRAM. Se puede entonces agregar todas las declaraciones de datos e instrucciones de tablas (datos globales) necesarias.

Si se usan programas incluidos, el sistema propone los nombres de acuerdo a la siguiente regla:

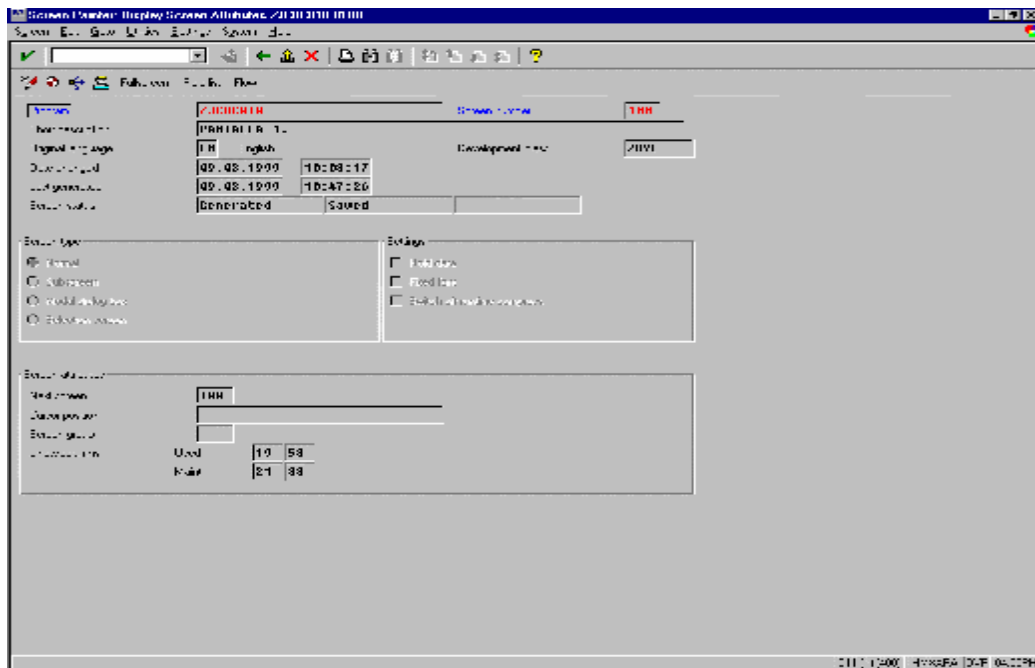
Los primeros 5 caracteres son los últimos 5 caracteres del nombre del programa. El sexto caracter identifica el contenido del incluido-por ejem. O para módulos PBO, F para rutinas FORM (subrutinas) -, mientras que el séptimo y el octavo caracter son iguales a 01, excepto el programa Top Include. Ejemplos: El programa Top Include para el module pool SAPMTALA sería MTALATOP; El programa incluido para los módulos PAI del mismo programa debería ser MTALAI01, y para los módulos PBO sería MTALAO01:

3.2.3. Definición de Pantallas



Así tras acceder al Screen Painter desde el ABAP/4 Workbench, tendremos que introducir el programa y el número de pantalla que deseamos mantener. Una vez hecho esto aparecerá el editor de pantallas '**Fullscreen Editor**'.

Si estamos creando el dynpro por primera vez, nos pedirá los atributos de pantalla.



Se introduce un texto corto que describa a la pantalla, se elige el tipo de pantalla y por último se especifica el número de la pantalla siguiente.

3.2.4. Diseño de Pantallas

Generalmente se definen los campos de una pantalla utilizando los atributos de campos que ya existen en el Diccionario de ABAP/4. Así mismo también puedes utilizar los atributos de campos que ya fueron definidos en el module pool.

Existen dos modos de funcionamiento de editor de pantallas: el **modo gráfico** y el **modo alfanumérico**, dependiendo de la interfase gráfico sobre el que funcione SAP.

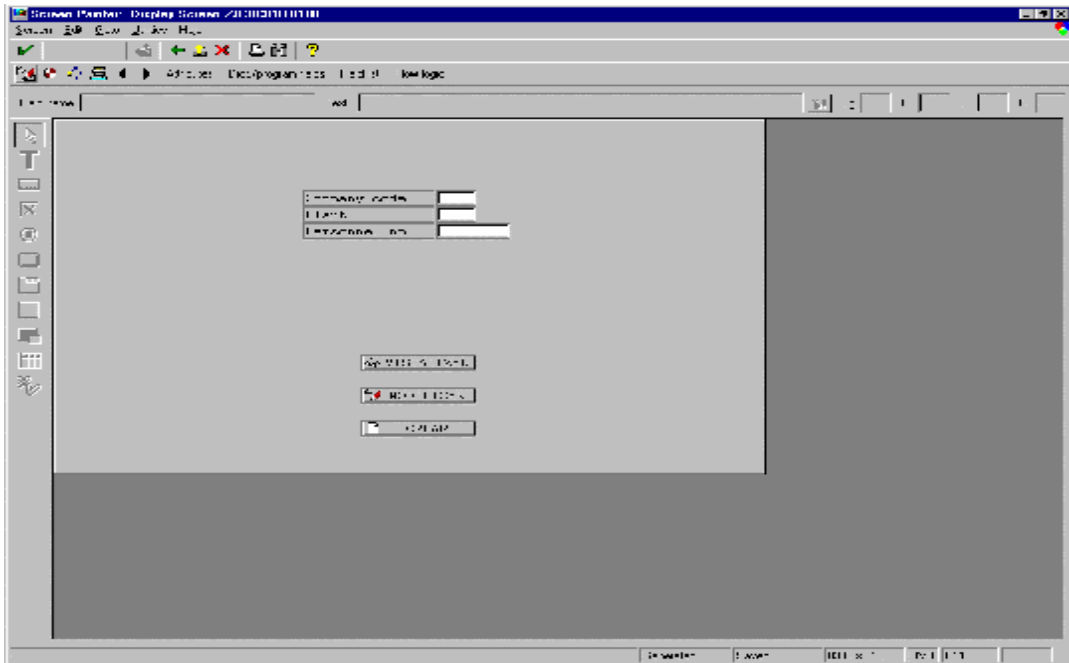
En este capítulo se describe el uso del Screen Painter en modo gráfico (versión 3.0), ya que se le considera como el más cómodo y avanzado, siendo además soportado por los interfaces gráficos más extendidos (MS WINDOWS y X11/MOTIF UNIX). De cualquier modo la funcionalidad de ambos modos del editor es la misma.

Para activar o desactivar el modo gráfico de pantallas se deben seguir la siguiente ruta:

Settings -> Graphical Fullscreen

El Screen Painter gráfico contiene funciones fáciles de usar para definir los elementos de las pantallas (por ejemplo, Campos de entrada/salida, textos de campo, cajas, etc.). Se elige cada elemento de pantalla y se posiciona en pantalla utilizando el mouse.

Para borrar elementos de pantalla, se selecciona el elemento con el mouse y entonces se elige el botón *Delete*.



Para mover elementos de pantalla, se usa el mouse arrastrando el elemento a la posición requerida.

En el editor de pantallas podemos observar tres áreas diferenciadas.

- § La **cabecera**: con datos sobre la pantalla y el campo que se esta manteniendo en ese preciso instante.
- § La **barra de objetos** (columna izquierda): Lista de los objetos que se pueden crear en la pantalla: textos, entrada de datos, 'checkboxes', 'frames', 'screens'...
- § El **área de dibujo**: Es el área donde se dibuja la pantalla que estamos diseñando.

3.2.5. Creando objetos en la pantalla.

Para dibujar un objeto en la pantalla tendremos que colocarlo en el área de trabajo y posteriormente definir sus características (atributos). Para ello tendremos que pulsar el botón correspondiente en la barra de objetos y marcar el área donde vamos a situar el objeto. Si queremos cancelar la creación de un objeto pulsaremos el botón **Reset** de la misma barra de objetos.

Objetos disponibles:

- **Textos**: Textos, literales, ... que son fijos en pantalla y no pueden ser manipulados por el usuario. Para considerar varias palabras como un mismo texto, tendremos que colocar un símbolo '_' entre ellas, ya que de otro modo las considerará como objetos de texto completamente distintos.
- **Objetos de entrada/salida ('Templates')**: Son campos para introducir o visualizar datos. Pueden hacerse opcionales u obligatorios. Los caracteres de entrada especifican con el símbolo '_', pudiendo utilizar otros caracteres para dar formato a la salida. Por ejemplo una hora podemos definirla como: _:__:_.

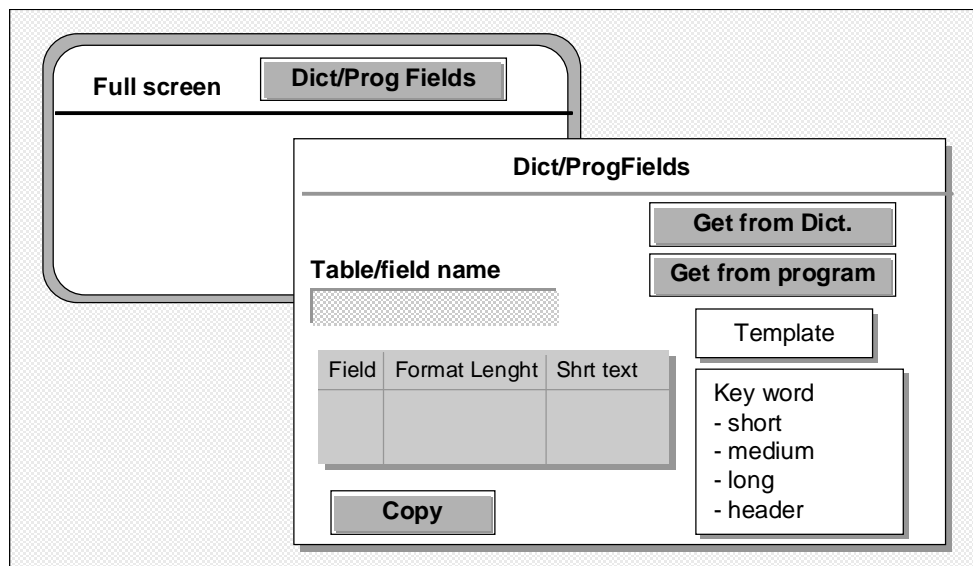
posible también efectuar el mantenimiento de atributos de un campo en el editor de pantallas (editor fullscreen).

En el editor de pantallas, se usan tipos de datos externos. Los tipos de datos externos de los campos seleccionados en el Diccionario ABAP/4 son desplegados en la columna 'Format'. En el caso de campos (plantillas) que no tienen ninguna referencia al Diccionario ABAP/4, el usuario puede definir un tipo de dato externo.

Se puede encontrar una correspondencia de tipos de datos externos con tipos de datos internos (por ejem. Tipos de datos ABAP/4), al consultar la documentación referente a la palabra reservada "TABLES". Ejemplos, es esto son:

Tipo de datos en Diccionario ABAP/4	Tipo de datos en programas ABAP/4
CHAR	C
NUMC	N

3.2.7. Creando objetos desde el diccionario de datos.

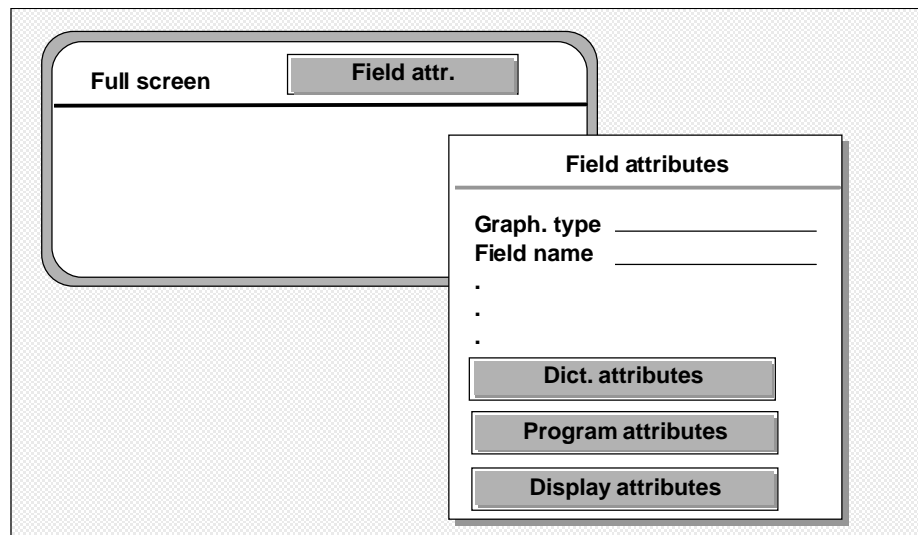


En la pantalla que estamos diseñando, podemos utilizar campos que están guardados en el Diccionario de Datos o declarados en el Module Pool. Para ello tendremos que seleccionar: **Dict/Program fields.**

Aparecerá una pantalla de selección de datos en la que indicaremos el campo o la tabla de la cual queremos obtener datos. Además se deberá seleccionar, si queremos ver la descripción de cada campo (indicando la longitud) y si queremos realizar una entrada de datos ('templates') de dicho campo por pantalla. Finalmente pulsaremos el botón correspondiente a crear desde el diccionario de datos o desde un programa.

Marcaremos el campo que queremos incorporar a nuestra pantalla y los copiaremos sobre el área de trabajo, situándolos en la posición que creamos más conveniente.

3.2.8. Definiendo los atributos individuales de cada campo.



Los atributos de los objetos definen las características de estos.

Podemos mantener los atributos desde el mantenimiento de atributos de campo o desde listas de campos.

Podemos distinguir entre atributos generales, de diccionario, de programa y de visualización.

§ Atributos Generales.

- **MatchCode** : Permite especificar un MatchCode para la entrada de un campo.
- **References** : Especificamos la clave de la moneda en caso de que el campo sea de tipo cantidad (CURR o QUAN).
- **Field Type** : Tipo de Campo.
- **Field Name** : Nombre del Campo. Con este nombre se identificarán desde el programa.
- **Field Text** : Texto del Campo. Si queremos utilizar un icono en vez de texto dejaremos este valor en blanco.
- **With Icon** : Si queremos utilizar iconos en la entrada de datos ('templates').
- **Icon name** : Identifica el nombre de un icono para un campo de pantalla.
- **Rolling (Scrolling)**: Convierte un campo en desplegable, cuando su longitud real es mayor que su longitud de visualización.
- **Quick Info** : Es el texto explicativo que aparece cuando pasamos por encima de un icono con el ratón.
- **Line** : Especifica la línea donde el elemento aparecerá. El sistema completa este valor automáticamente.
- **Cl** : Especifica la columna donde el elemento aparecerá. El sistema completa este valor automáticamente.
- **Ht** : Altura en líneas. El sistema completa este valor automáticamente.
- **DLg** : Longitud del campo.
- **VLg** : Longitud de visualización.
- **FctCode** : Código de función (código de 4 dígitos). Atributo sólo para pushbuttons.
- **FctType** : Especifica el tipo de evento en el cual el campo será tratado.

- **Ltype :** Tipo de Step Loop (fijo o variable). El tipo variable significa que el tamaño del step loop se ajusta según el tamaño de la pantalla, mientras que fijo no ajusta el step loop.
- **Lcnt :** Número de líneas de un step loop.
- **Groups :** Identifica grupos de modificación para poder modificar varios campos simultáneamente. Podemos asignar un campo a varios (4) grupos de modificación.

§ Atributos de Diccionario:

- **Format :** Identifica el tipo del campo. Determina el chequeo que realiza el sistema en la entrada de los datos.
- **Frm DICT. :** El sistema rellena este atributo en el caso de que el campo lo hayamos creado a partir de un campo del diccionario de datos.
- **Modific. :** El sistema rellena este campo si detecta alguna diferencia entre la definición del campo en el diccionario de datos y su utilización en pantalla.
- **Conv. Exit :** Si queremos utilizar una rutina de conversión de datos no estándar, especificamos aquí el código de esta.
- **Param. ID :** Código del parámetro SET / GET. (ver siguiente atributo).
- **SET Param GET Param:** Los parámetros **SPA** (Set Parameter) y **GPA** (Get Parameter), nos permiten visualizar valores por defecto en campos. Si marcamos el atributo **SET param**, el sistema guardará en un parámetro ID lo que entremos en este campo. Si marcamos el atributo **GET param**, el sistema inicializa el campo, con el valor del parámetro ID que tenga asignado en el atributo anterior.
- **Up./Lower :** El sistema no convierte la entrada a mayúsculas.
- **W/o template :** Marcamos este atributo si queremos que los caracteres especiales se traten como textos literales.
- **Foreign Key :** Si queremos que sobre el campo el sistema realicen chequeo de llave externa. (hay que definir previamente las claves externas en el diccionario de datos).

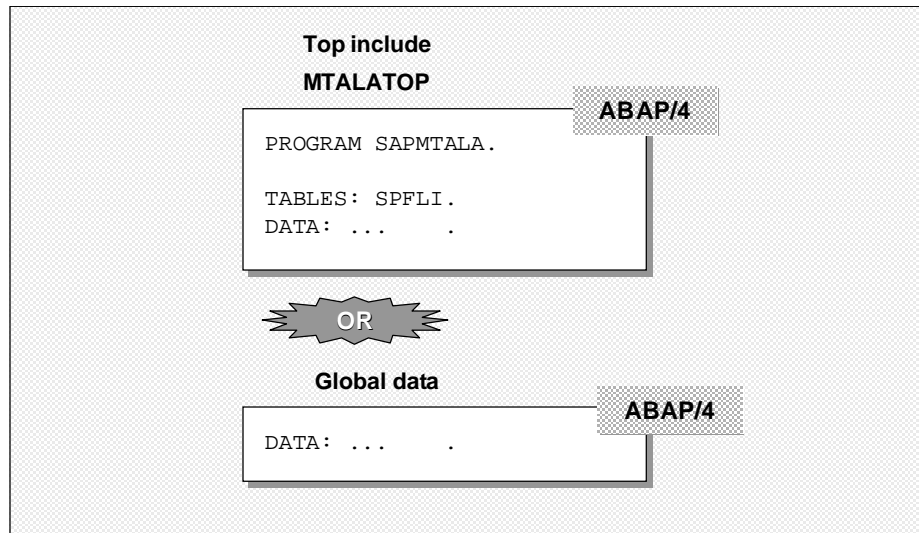
§ Atributos de Programa:

- **Input field :** Campo de entrada.
- **Output field :** Permite visualización. Se puede utilizar en combinación con el anterior.
- **Output only :** Sólo Visualización
- **Required field:** Atributo para campos obligatorios. Se distinguen con un ζ .
- **Poss. Entry :** El sistema marca este atributo si hay un conjunto de valores para el campo. No es posible modificar el contenido del atributo.
- **Poss. Entries :** Indica como podemos ver la flecha de entradas posibles en un campo.
- **Right-Justif :** Justifica cualquier salida del campo a la derecha.
- **Leading Zero :** Rellena con ceros por la izquierda en el caso de salidas numéricas.
- ***-Entry :** Permite la entrada de un asterisco en la primera posición de un campo. Si se introduce un * se podrá hacer un tratamiento en un módulo: FIELD MODULE ON *-INPUT.
- **No Reset :** Cuando activamos este atributo, la entrada de datos no podrá ser cancelada mediante el carácter !.

§ Atributos de Visualización:

- **Fixed Font :** Visualiza un campo de salida en un tamaño fijo (no proporcional). Sólo se puede utilizar en campos Output only.
- **Bright :** Visualiza un campo en color intenso.
- **Invisible :** Oculta un campo.
- **2-Dimens :** Visualiza un campo en dos dimensiones en vez de en tres.

3.2.9. Definiendo campos en el Module Pool.



En el procesamiento en diálogo, los datos son transferidos entre pantallas y programas ABAP/4. El sistema ejecuta esta comunicación automáticamente, pero es necesario usar nombres idénticos en las pantallas y en em module pool.

Se definen los campos relevantes como datos globales en el programa Top Include. Al hacer esto, se procesa el programa Top Include, ó las secciones concernientes a las estructuras de datos globales, ó las estructuras de Diccionario.

3.3. Definición de la Flujo de Control

3.3.1. Introducción a la Lógica de Proceso.

<pre>PROCESS BEFORE OUTPUT. MODULE INITIALIZE. PROCESS AFTER INPUT. MODULE READ_SPFLI.</pre>	Screen Painter
---	---------------------------

Una vez hemos definido gráficamente las pantallas, será preciso escribir una **lógica de proceso** para cada una de ellas, pasándose a denominar Dynpros.

Para introducir la lógica de proceso de las pantallas, utilizaremos una versión especial del editor de ABAP/4. **Goto -> Flow Logic**.

La lógica de proceso de las pantallas tienen una estructura determinada, y utilizan comandos y eventos propios de manejo de pantallas, similares a los utilizados en ABAP/4.

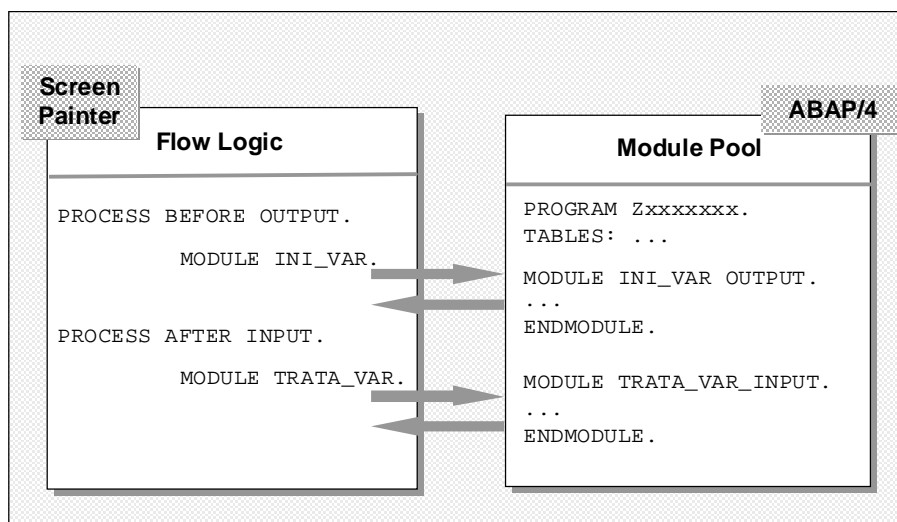
Consistirá en dos eventos fundamentales:

- PROCESS BEFORE OUTPUT (PBO).**
- PROCESS AFTER INPUT (PAI).**

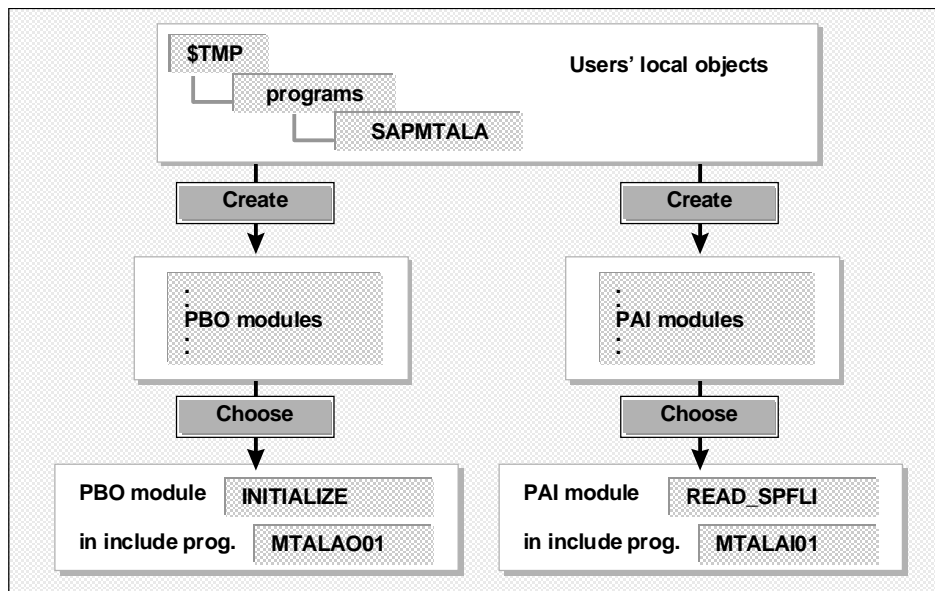
El **PBO** será el evento que se ejecutará previamente a la visualización de la pantalla, mientras que el **PAI**, se ejecutará después de la entrada de datos del usuario en la pantalla.

Además de estos dos eventos obligatorios existen eventos para controlar las ayudas, **PROCESS ON HELP-REQUEST (POH)**, y para controlar los valores posibles de un campo **PROCESS ON VALUE REQUEST (POV)**.

Desde la lógica de proceso de las pantallas no se actualizan datos, únicamente se llaman a los módulos del Module Pool que se encarga de esta tarea.



3.3.2. Creación de Módulos ABAP/4



Para llamar a un módulo utilizaremos la sentencia **MODULE**.

MODULE <nombre_módulo>.

Si se selecciona un módulo con doble click, el sistema crea las instrucciones **MODULE ...**
END MODULE en el programa incluido correspondiente.

Si el include no existe, el sistema creará uno automáticamente si así se desea. Esto también inserta una instrucción **INCLUDE** en el programa principal, para hacer referencia a este programa incluido

En el Module Pool el código del módulo empezará con la sentencia:

MODULE <nombre_módulo> OUTPUT.

En el caso de ser un módulo llamado desde el PAI será:

MODULE <nombre_módulo> INPUT.

```

MODULE INITIALIZE OUTPUT.
  CLEAR SPFLI.
ENDMODULE.

MODULE READ_SPFLI INPUT.
  SELECT SINGLE * FROM SPFLI
  WHERE CARRID = SPFLI-CARRID
  AND CONIID = SPFLI-CONNID.
  .
  .
ENDMODULE.
    
```

3.3.2.1. Process Before Output (PBO).

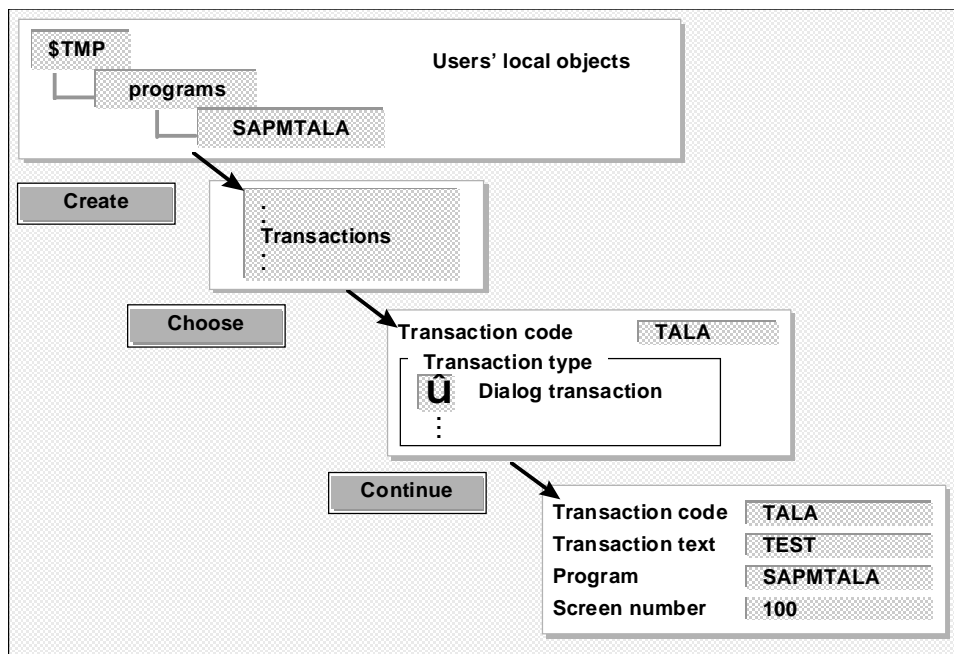
En el módulo del PBO los inicializaremos al valor que queramos. Si no inicializamos explícitamente, el sistema le asignará su valor inicial por defecto, al no ser que lo hayamos definido como parámetro SPA/GPA.

En el módulo PBO, indicaremos todos los pasos que queremos realizar antes de que la pantalla sea visualizada, como por ejemplo inicializar los campos de salida. Esta inicialización se realizará en un módulo independiente dentro del Module Pool.

3.3.2.2. Process After Input (PAI).

El PROCESS AFTER INPUT se activa cuando el usuario selecciona algún punto de menú, pulsa alguna tecla de función o pulsa ENTER. Si alguno de estos eventos ocurre, el PAI de la pantalla necesitará responder apropiadamente a la función seleccionada por el usuario.

3.3.3. Definiendo la llamada (Códigos de Transacción)

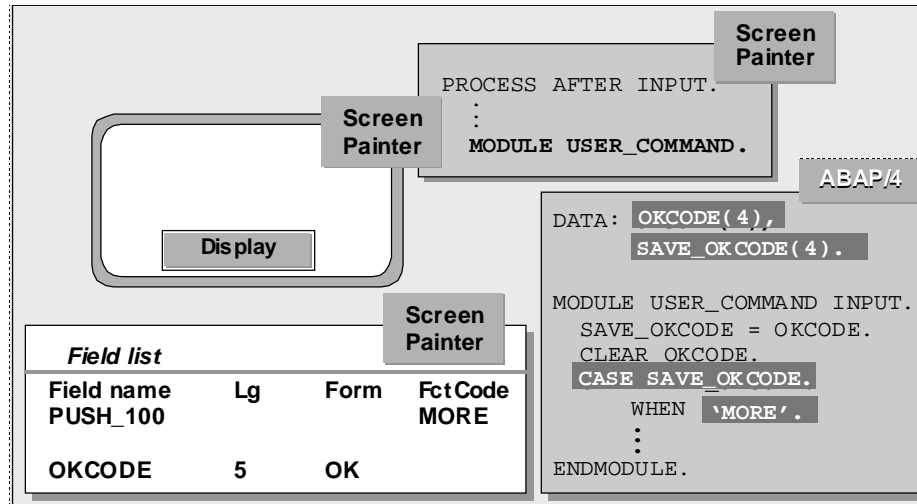


Se puede iniciar un programa de diálogo ABAP/4 ("transacción"), especificando un código de transacción.

Las transacciones de cliente deben iniciar con "Z" ó "Y".

El sistema almacena las especificaciones en la tabla TSTC. (En vez de crear una transacción desde la lista de objetos, se puede modificar la tabla directamente).

3.3.4. Leyendo códigos de función en programas.



Cuando el usuario de una transacción, pulsa una tecla de función, un punto de menú, un pushbutton, un icono o simplemente la tecla ENTER, los datos introducidos en la pantalla se pasan a los módulos del PAI para ser procesados junto a un código de función que indicará que función a solicitado el usuario.

En el Screen Painter, será necesario crear un campo de tipo código de función, **OK**, (de longitud 4), que normalmente aparece al final de la lista de campos de cada pantalla. Tradicionalmente a este campo se le denomina **OK_CODE**, y será declarado en nuestro module Pool como cadena de caracteres de 4 posiciones:

En la lógica de proceso de cada pantall, tendremos que realizar al tratamiento del OK_CODE. Para ello utilizaremos un modulo que deberá ser el últimodel evento PAI, es decir que se ejecutará una vez que todos los datos de entrada han sido validados correctamente.

Una vez procesado el módulo de función, borraremos el contenido del OK_CODE, inicializándolo para la próxima pantalla. Podemos guardar el contenido del OK_CODE en una variable intermedia e inicializarlo inmediatamente

4. Técnicas especiales "Screen & Menu painter"

4.1. La validación de los datos de entrada.

Una de las funciones más importantes de Process After Input, es la de **validar los datos de entrada** de la pantalla antes de ser usados. Existen dos tipos de validación de los datos de entrada: Un **chequeo automático** realizado por el sistema y un **chequeo manual** programado con el comando **FIELD** de la lógica de proceso de Dynpros.

4.1.1. Verificación automática

El sistema realiza automáticamente una serie de chequeos de los datos de entrada, antes de procesar el evento PAI.

4.1.1.1. Verificación de formato

The diagram shows a 'Screen Painter' window containing a 'Field list' table and two input fields. The 'Field list' table is as follows:

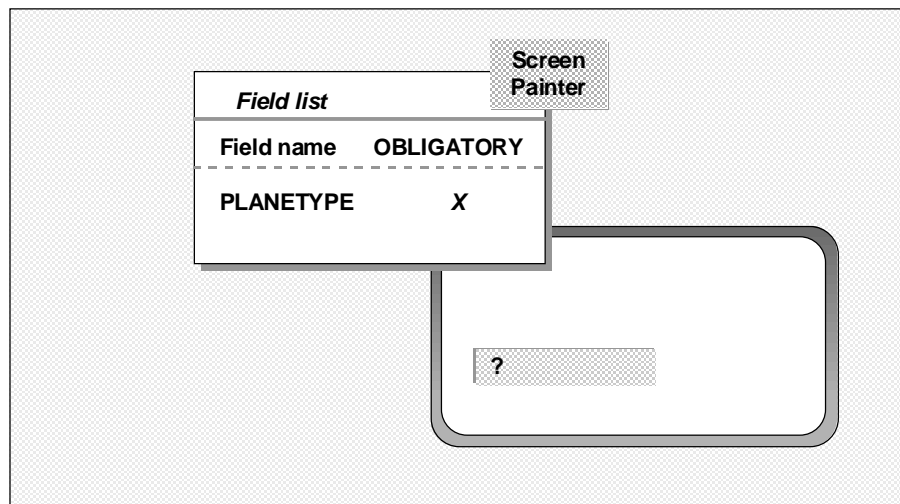
Field name	Format
DATE	DATS
.	.
Amount	DEC

Below the table, two input fields are shown:

- Field 1: Date (30.02.1996), Amount (). Error message: E: Invalid date.
- Field 2: Date (), Amount (12A3). Error message: E: Please enter numeric value.

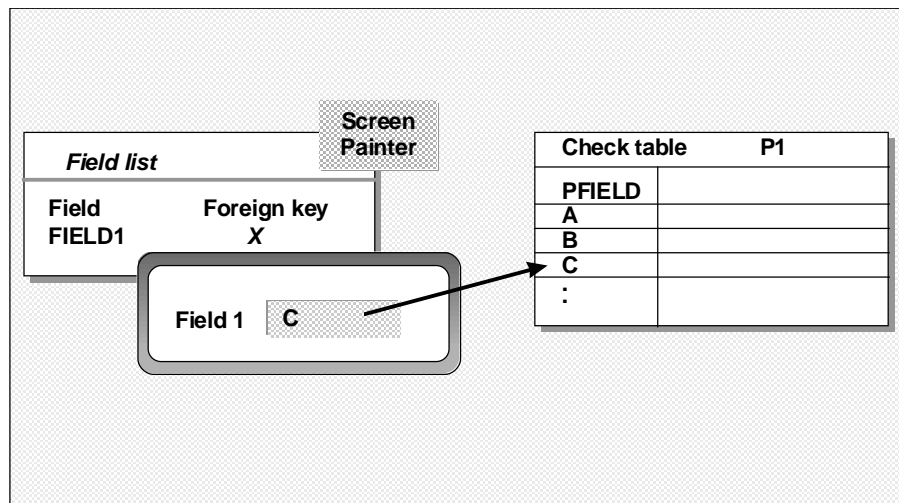
El procesador de diálogo valida las entradas de acuerdo a los atributos de cada campo. Si el sistema detecta un valor incorrecto, despliega un mensaje de error y vuelve a mostrar los campos para su nueva entrada.

4.1.1.2. Verificación de campos obligatorios



Al momento de que algún campos de la pantalla se le asigna el atributo de que es obligatorio, el procesador de diálogo no continua con el proceso, almenos que todos los campos obligatorios tengan algún valor.

4.1.1.3. Verificación de llaves foráneas



Una verificación de clave foránea es procesada solo si un campo de pantalla se refiere a un campo del Diccionario para el cual se ha definido una tabla de verificación.

Al mismo tiempo, la funcionalidad de la tecla F4 es activada. Esto significa que las posibles entradas para un campo son desplegadas.

4.1.1.4. Verificación de valores fijos

En el Diccionario ABAP/4, se pueden definir los valores fijos para los dominios. Si se define un campo de pantalla con referencia a un dominio con valores fijos, ocurre lo siguiente:

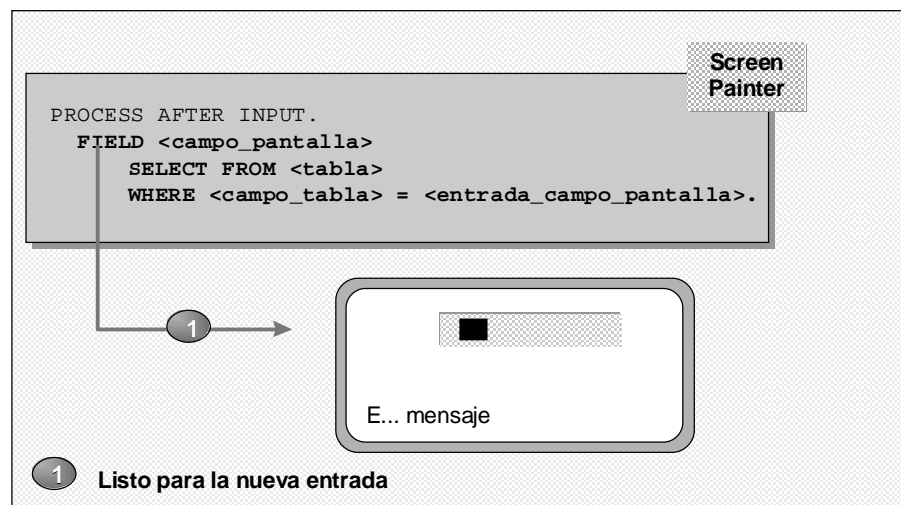
- Los valores fijos son desplegados si el usuario presiona la tecla F4 para ver los posibles valores para el campo de entrada.
- El procesador de diálogo verifica los valores introducidos en el campo contra el conjunto de valores fijos del Dominio correspondiente.

4.1.2. Verificación manual en Module Pool.

Además del chequeo automático es posible realizar una validación más extensa de los valores de entrada a los campos valores de entrada a los campos con las instrucciones **FIELD** y **CHAIN** de la lógica de proceso del Screen Painter.

Con **FIELD** podemos validar individualmente cada campo de forma que en caso de error, la siguiente entrada de datos sólo permitirá introducir el campo erróneo sobre el que estamos utilizando la instrucción FIELD.

Dependiendo del tipo de sentencia FIELD que utilicemos, el mecanismo de chequeo se realizará en la lógica de proceso del Screen Painter o en un módulo ABAP/4.



Es posible realizar distintas validaciones de un campo de entrada dependiendo de la fuente con la que contrastamos los valores posibles. Así podemos checar el contenido de un campo comparandolo con una tabla de la base de datos, con una lista de valores, o realizando la validación en un módulo del Module Pool.

Para **chechar un campo contra la base de datos** utilizamos:

```
FIELD <campo_pantalla> SELECT FROM <tabla>
      WHERE <campo_tabla> = <entrada_campo_pantalla>.
```

Si no se encuentran registros en el Diccionario de Datos el sistema emite un mensaje de error estándar. Existe una versión ampliada de la instrucción anterior que permite enviar mensajes o warnings en caso de que encuentre o no registros:

```
FIELD <campo_pantalla> SELECT * FROM <tabla>
WHERE <condición>
WHENEVER (NOT) FOUND SEND
ERRORMESSAGE / WARNING <número>
WITH <campo-texto>.
```

Para **chechar un campo respecto a una lista de valores** utilizamos:

```
FIELD <campo_pantalla> VALUES (<lista_valores>).
```

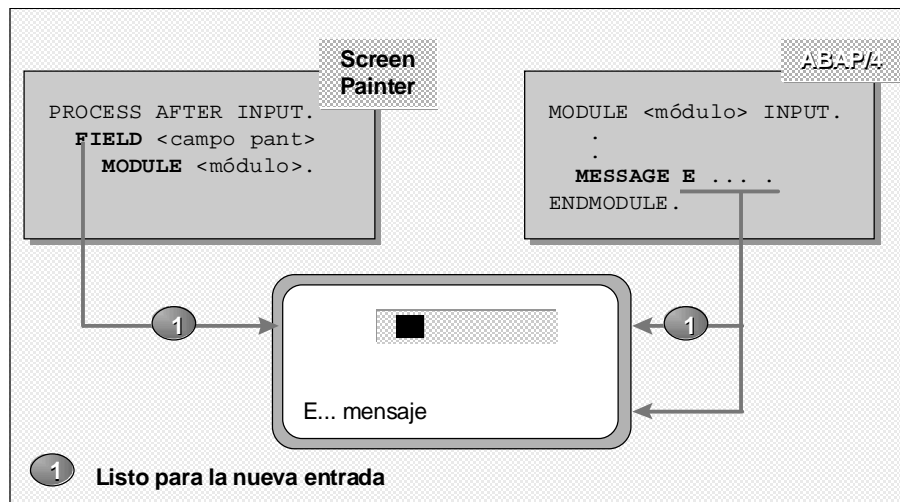
Donde <lista_valores> puede ser:

```
('<valor>')
(not'<valor>')
('<valor 1>','<valor 2>',...NOT'<valor n>')
(BETWEEN '<valor 1>' AND '<valor 2>')
(NOT BETWEEN '<valor 1>' AND '<valor 2>')
```

Si el valor entrado por el usuario no corresponde a ningun valor de la lista el sistema emite un mensaje de error.

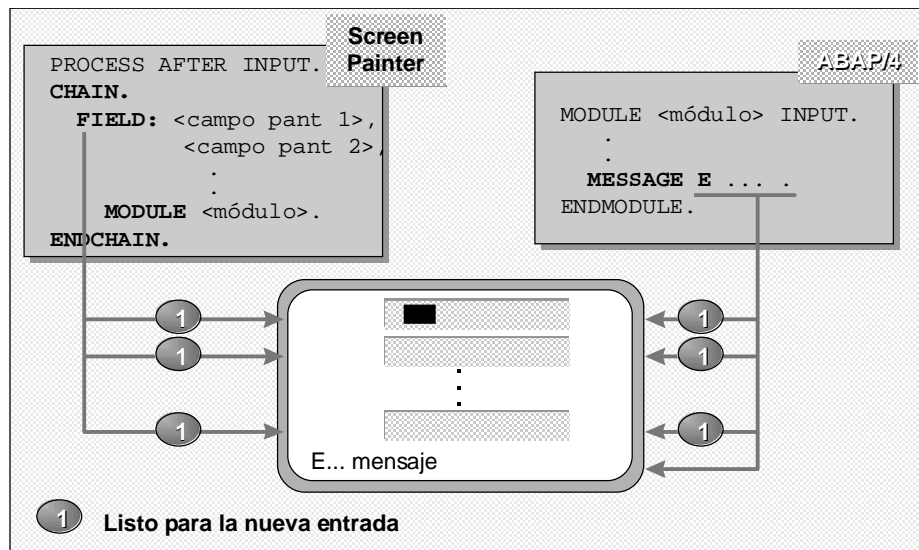
Para **chechar un campo en un módulo de ABAP/4** utilizamos:

```
FIELD <campo_pantalla> MODULE <módulo_ABAP/4>.
```



Si el módulo resulta con un error (E) o un mensaje de advertencia (W), la pantalla es desplegada nuevamente pero sin procesar los módulos **PBO**. El texto del mensaje es mostrado, y solo el campo que ocasionó el error estará disponible para introducir datos nuevamente.

La instrucción **CHAIN...ENDCHAIN** encierra un conjunto de instrucciones **FIELD**, en caso de error en la entrada de alguno de ellos, todos los campos del CHAIN se podrán modificar, mientras que los que no pertenezcan al CHAIN estarán bloqueados para la entrada de datos.



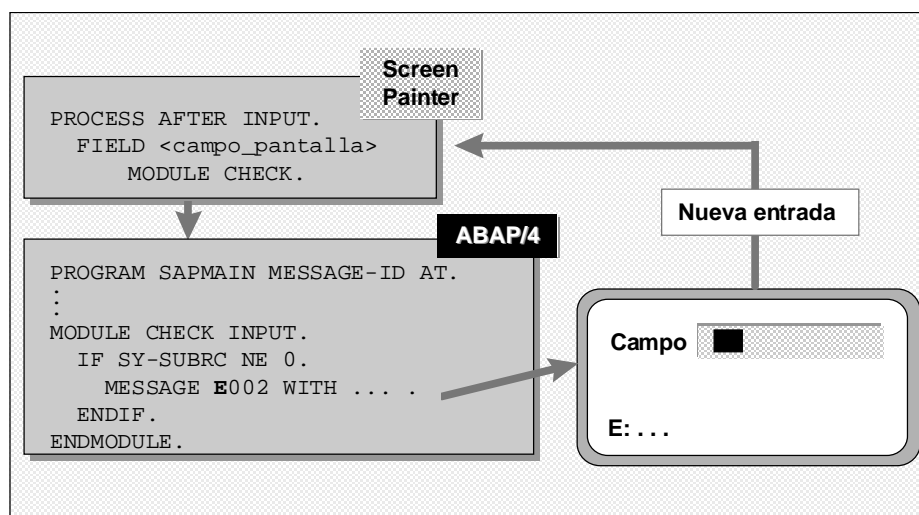
```

CHAIN.
  FIELD <campo 1>, <campo 2>, <campo 3>.
  MODULE <mod1>.
  MODULE <mod2>.
ENDCHAIN.

CHAIN.
  FIELD <campo 1>,<campo 2>.
  MODULE <mod1>.
  FIELD <campo 3> MODULE <mod2> ON CHAIN INPUT.
ENDCHAIN.
    
```

4.1.3. Mensajes en pantalla

3.4.3.1. Mensaje de Error

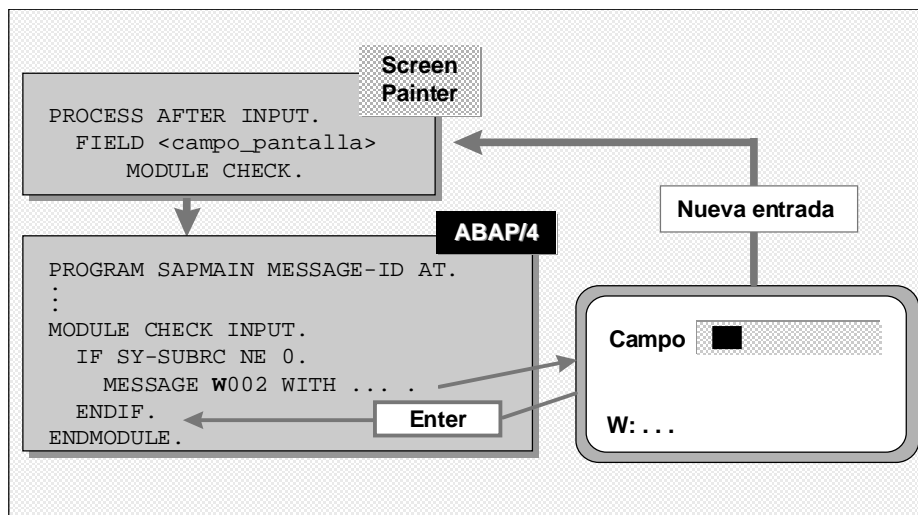


El texto de un mensaje de error ('E') es desplegado en la pantalla actual.

Todos los campos de pantalla asignados al módulo correspondiente (instrucción FIELD) se vuelven disponibles para introducir información de nuevo.

El sistema obliga al usuario a reintroducir datos.

3.4.3.2. Mensaje de Advertencia (Warning)

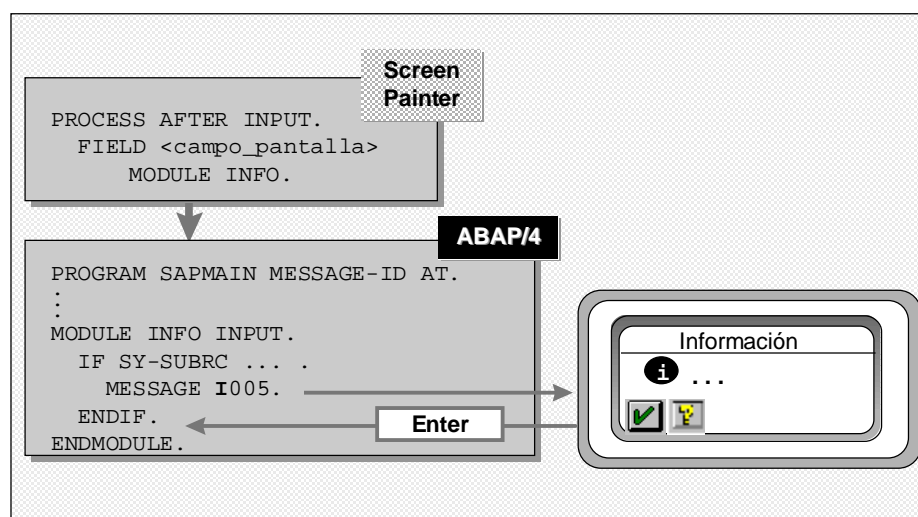


El texto del mensaje de advertencia ('W') es desplegado en la pantalla actual.

Todos los campos de pantalla asignados al módulo correspondiente (instrucción FIELD) se vuelven disponibles para introducir información de nuevo.

El usuario puede reintroducir los datos o ignorar el mensaje de advertencia presionando la tecla **ENTER**.

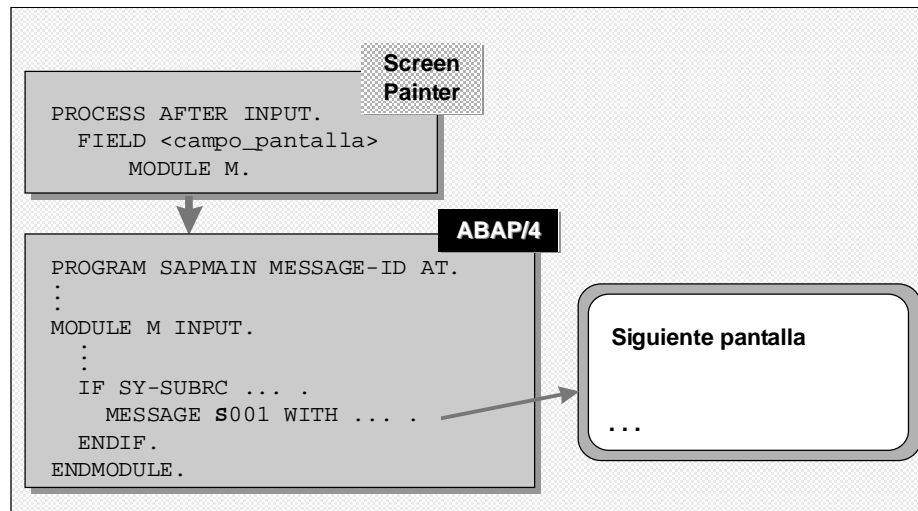
3.4.3.3. Mensaje de Información



El texto de un mensaje de información ('I') es desplegado en la pantalla actual.

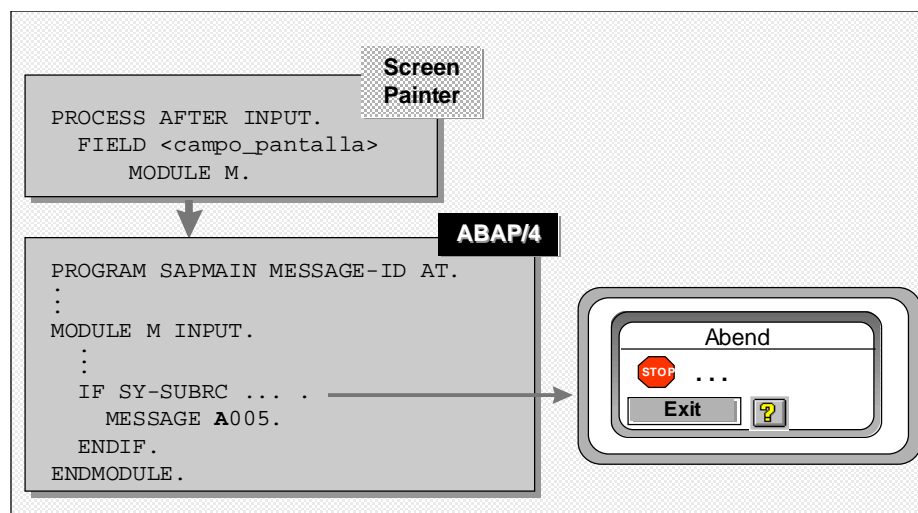
El proceso de la pantalla actual es suspendido. Después de que el usuario presione la tecla **ENTER**, el programa continua con su ejecución normal desde el punto donde fue suspendido.

3.4.3.4. Mensaje de Buen resultado



Un mensaje de texto de buen resultado ('S') es desplegado en la pantalla siguiente a la actual.

3.4.3.5. Mensaje de Interrupción (Abend)

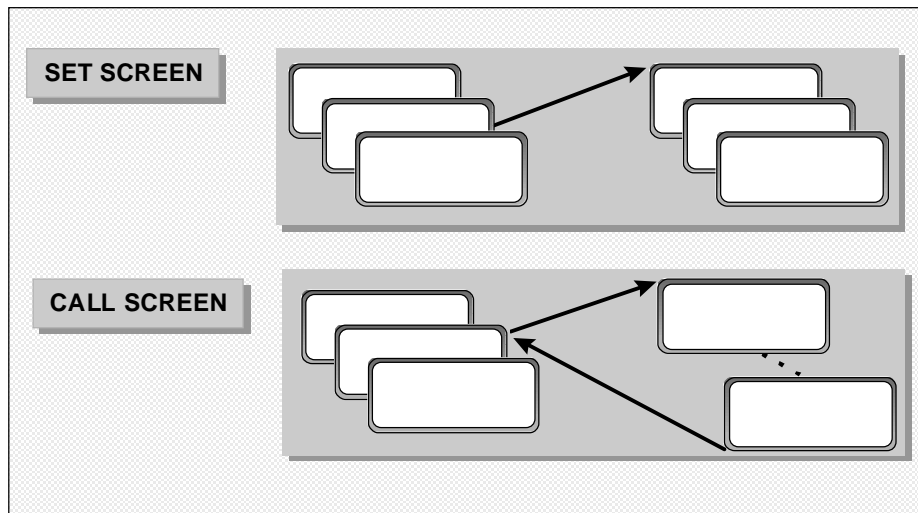


El texto de un mensaje de Interrupción ('A') es desplegado en la pantalla actual.

Después de que el usuario presione la tecla **ENTER**, el proceso actual es terminado y el proceso regresa a la pantalla inicial.

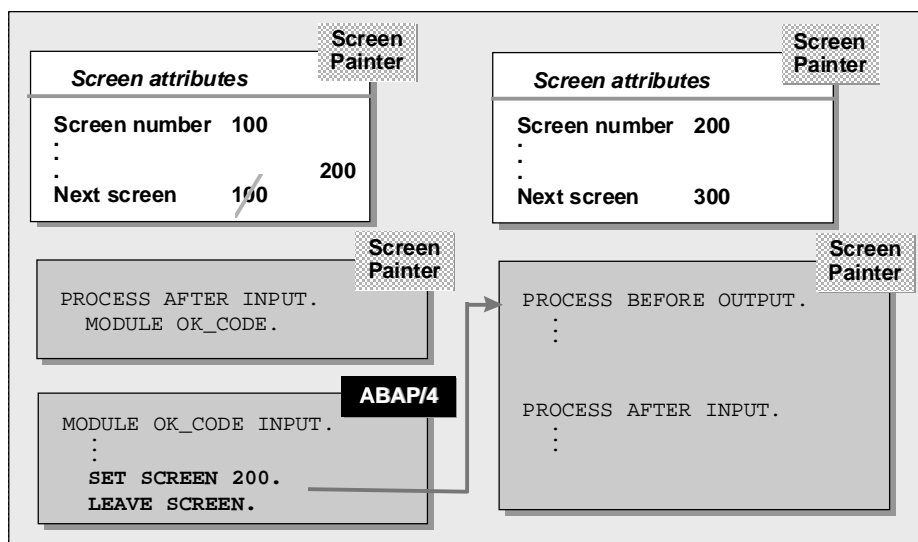
4.2. Secuencia dinámica de pantallas

4.2.1. Introducción



Desde una transacción podemos ir controlando el flujo de pantallas de la misma, llamar a otras transacciones o reportes.

4.2.2. Configuración dinámica de la siguiente pantalla



Por defecto, cuando acaben los módulos del evento PAI, el sistema saltará a la pantalla que indique el atributo **Next Screen** de la pantalla en ejecución. Es posible modificar el atributo de la próxima pantalla con la instrucción **SET**.

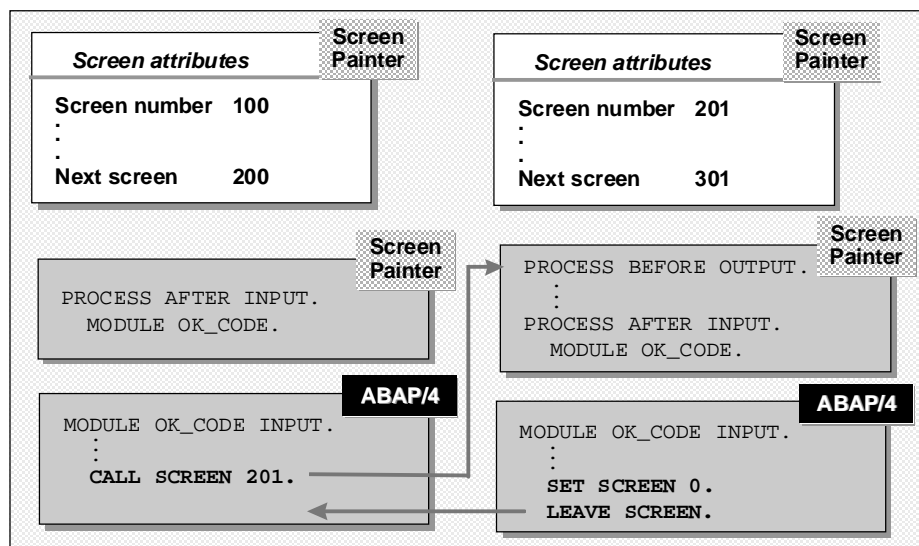
SET SCREEN <no._pantalla>.

La instrucción **SET SCREEN nnnn** reescribe temporalmente la siguiente pantalla a procesar. La pantalla nnnn debe ser una pantalla del mismo "module pool".

La pantalla siguiente es procesada después de procesar la pantalla actual, o al menos que se termina la ejecución de la pantalla actual con la instrucción **LEAVE SCREEN**. Al encontrar esta instrucción, se ejecuta la pantalla siguiente en forma inmediata.

Si se desea terminar el procesamiento de la pantalla actual e ir directamente a la pantalla siguiente en una sola instrucción, se puede usar el estatuto **LEAVE TO SCREEN nnn**.

4.2.3. Inserción de una o más pantallas



La instrucción **CALL SCREEN nnnn** interrumpe el procesamiento de la pantalla actual para procesar la pantalla nnnn y las pantallas subsiguientes.

La pantalla llamada con esta instrucción deberá ser una pantalla del mismo "module pool".

Cualquiera de las instrucciones: **SET SCREEN 0**, **LEAVE SCREEN**, **LEAVE TO SCREEN 0**, regresa el control a la locación donde fue ejecutada la instrucción **CALL SCREEN nnnn**.

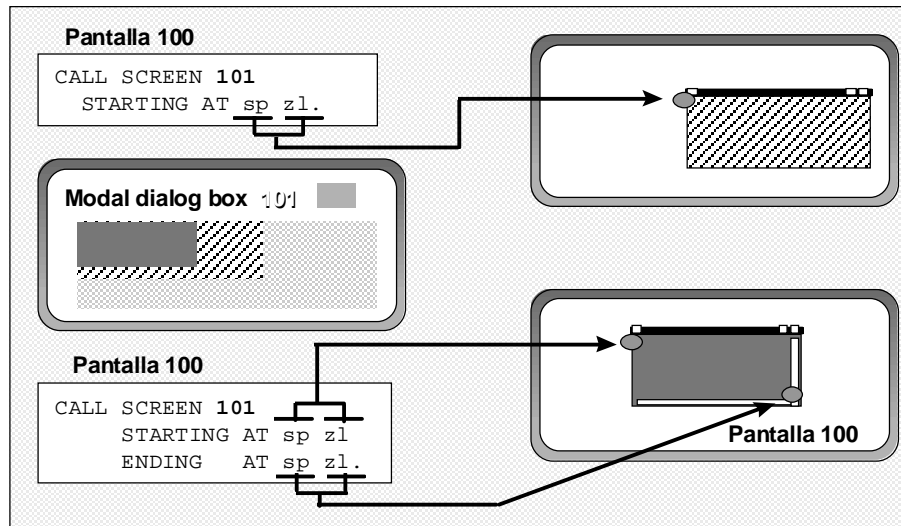
Si se usa cualquiera de estas instrucciones cuando no se está en el modo de llamada, el programa termina. Por ejemplo el regresar al lugar donde este programa fue llamado (ver también la instrucción **LEAVE PROGRAM**).

Usando las adiciones **STARTING AT** y **ENDING AT** en la instrucción **CALL SCREEN**, se puede especificar la posición y el tamaño de la pantalla a llamar. En estos casos, los estándares ergonómicos de SAP establecen que la pantalla debe estar definida como de diálogo tipo modal.

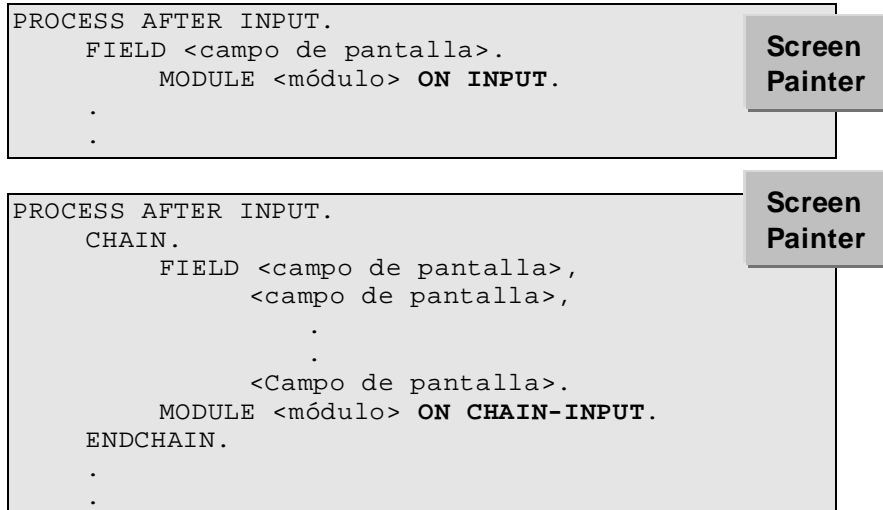
Se puede usar la adición **STARTING AT** sin la adición **ENDING AT**. Aquí, el sistema determina el tamaño de la pantalla de diálogo según el tamaño definido en el atributo de pantalla conocido como "Used size". El punto de comienzo de esta pantalla será la esquina superior izquierda. Si

también se usa ENDING AT, el sistema incluye lo más posible de la pantalla de diálogo dentro del área definida por las coordenadas, iniciando en la esquina superior izquierda.

Si la pantalla aparece incompleta, se incluye en la misma una barra de desplazamiento.



4.3. Ejecución condicionada de módulos



Si se especifica la adición **ON INPUT** después de MODULE en una instrucción FIELD, el módulo es ejecutado solamente si el campo relevante contiene un valor diferente al valor inicial.

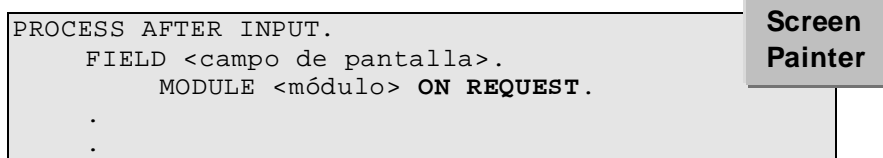
En un estatuto **CHAIN** se debe usar la instrucción **ON CHAIN-INPUT**. Entonces, el módulo concerniente es procesado solamente si al menos uno de los campos de pantalla del estatuto CHAIN contiene un valor diferente al valor inicial.

Se puede usar la adición ON INPUT solamente si la instrucción MODULE es especificada dentro de una instrucción FIELD.

Si se especifica la adición **ON REQUEST** después de MODULE en una instrucción FIELD, el módulo es ejecutado únicamente si el campo relevante tiene una nueva entrada.

En un estatuto **CHAIN**, se debe usar la instrucción **ON CHAIN-REQUEST**. Entonces, el módulo concerniente es procesado solamente si al menos uno de los campos de pantalla del estatuto CHAIN tiene una nueva entrada.

Se puede usar la adición ON REQUEST solamente si la instrucción MODULE es especificada dentro de una instrucción FIELD.

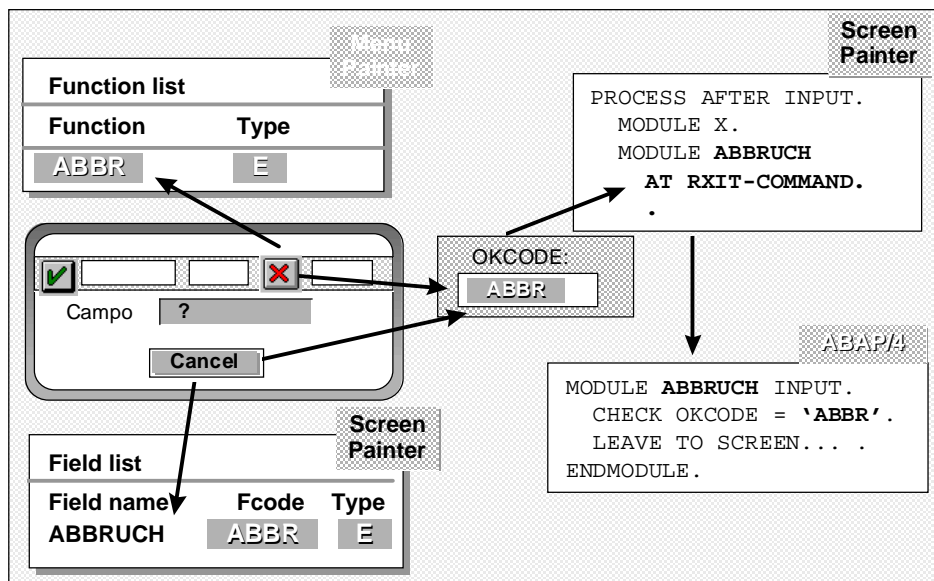


```

PROCESS AFTER INPUT.
CHAIN.
  FIELD <campo de pantalla>,
    <campo de pantalla>,
    .
    <Campo de pantalla>.
MODULE <módulo> ON CHAIN-REQUEST.
ENDCHAIN.
    
```

Es posible que en alguna ocasión el usuario quiera salir de la pantalla sin necesidad de pasar las validaciones automáticas (Por ejemplo utilizando las funciones estándares BACK, EXIT o CANCEL). En este caso utilizaremos la cláusula **AT EXIT COMMAND** de la instrucción MODULE.

MODULE <módulo_ABAP> AT EXIT-COMMAND.



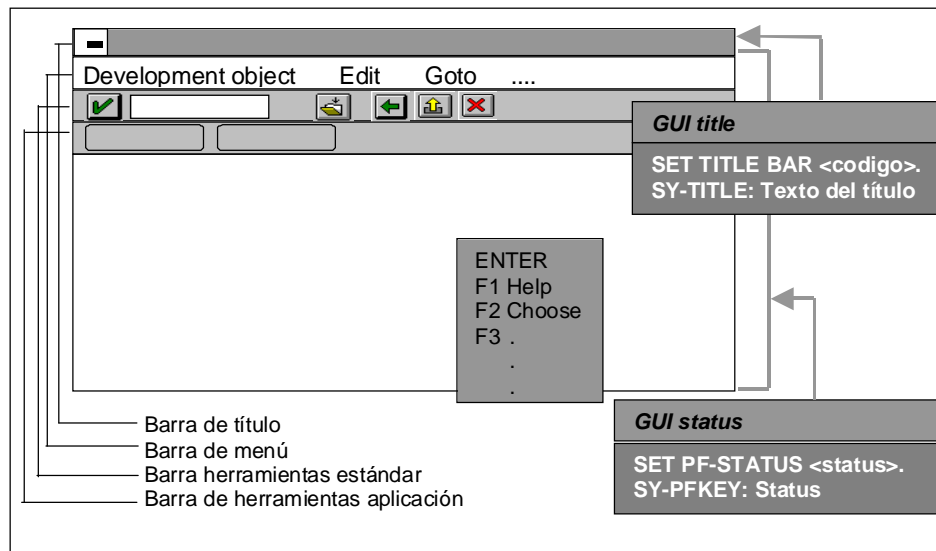
Para poder utilizar un AT EXIT COMMAND en un botón de campo, será necesario asignar el valor **E** en el atributo de campo **FctType** del editor de pantallas o en las funciones del Menú Painter.

En el módulo que llamamos incluiremos las instrucciones necesarias para poder salir de la transacción o de la pantalla en proceso. Por ejemplo: **LEAVE TO SCREEN 0**.

5. Menú Painter

5.1. DISEÑO DE MENÚS (Menú Painter).

5.1.1. Introducción



Con el **Menu Painter** diseñaremos las superficies **GUI**, (Grafical User Interface), sobre las que correrán las transacciones SAP.

Una GUI contiene todos los menús, teclas de función, pushbuttons, etc... disponibles para el usuario, durante la ejecución de una transacción.

Podremos indicar el status que utilizaremos en una pantalla o el título en un módulo PBO de la pantalla con las instrucciones:

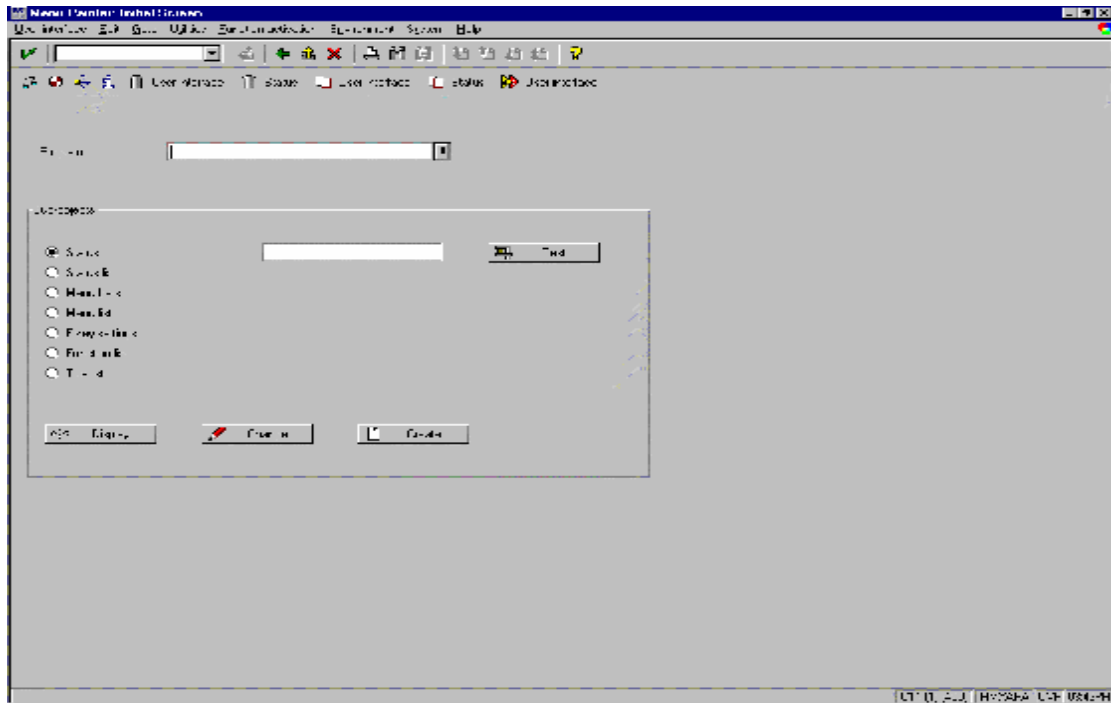
```
SET PF-STATUS <cod_status>.
SET TITLEBAR <cod_título>.
```

Indicaremos las diferentes interfaces GUI de una transacción mediante los **status**. Una transacción tendrá muchos status diferentes. No será necesario redefinir todos los objetos de los status, ya que muchos objetos definidos en un status podrán ser utilizados en otro. Por ejemplo es posible crear una barra de menús igual para toda una transacción.

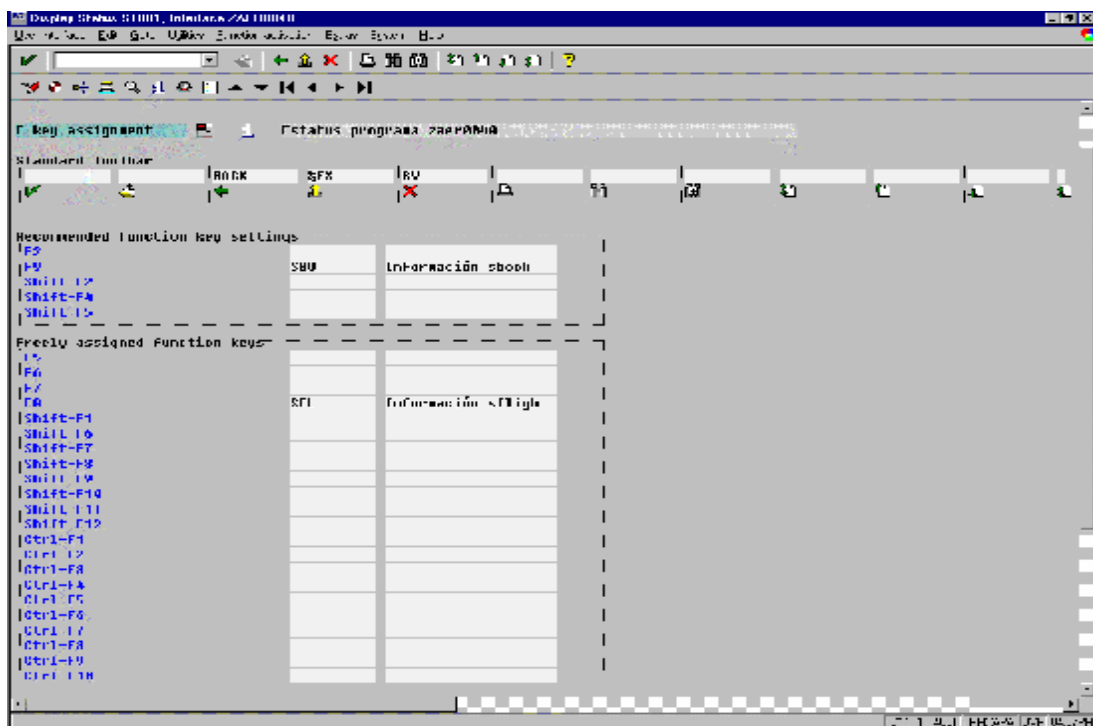
Para iniciar el Menú Painter, puedes usar cualquiera de los siguientes caminos:

- Seleccionando: **Tools -> ABAP/4 Workbench -> Desarrollo -> Menú Painter.**
- Dando el código de transacción **SE41**.
- Dentro del editor de ABAP/4, posicionar el cursor en el nombre del status en el estatutto SET PFSTATUS <status> dando doble clic y/o presionar F2.

Es posible mantener tanto un estatus de un determinado programa, como los diferentes objetos de un GUI que forman parte de los status (Barras de Menús, teclas de función, títulos de menú...).



5.1.2. Teclas de Función.

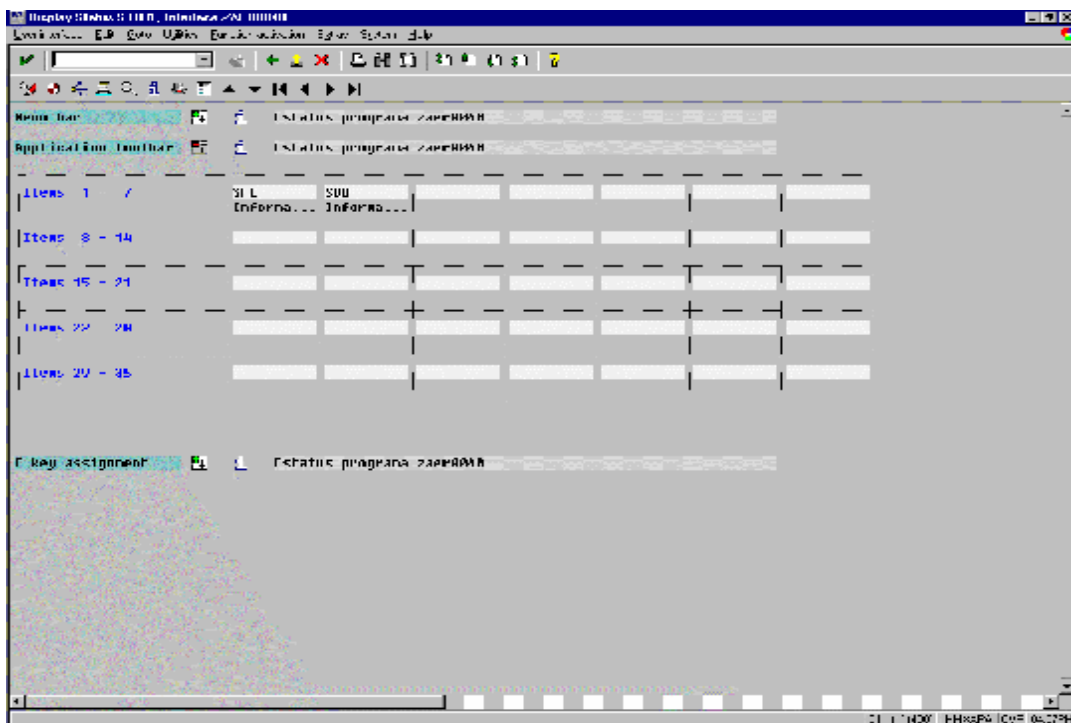


Como es recomendable que todas las teclas de función que se definan, estén incluidas en la barra de menús y si se desea en la barra de aplicación, comenzaremos por definir estas teclas de función primeramente.

Para definir las teclas de función utilizamos el espacio destinado para ello. Indicando el código de la función en la línea correspondiente a la tecla que deseamos utilizar. El texto de la tecla de función aparecerá automáticamente, pero podrá ser modificada en caso de desearlo.

SAP no recomienda definir nuevas teclas de función en el espacio reservado para teclas de función estándar.

5.1.3. Los 'Pushbuttons'.



Los pushbuttons son botones tridimensionales que aparecen debajo de la barra de herramientas estándar (barra de aplicación).

Previamente a definir un botón será necesario definir la función deseada como una tecla de función.

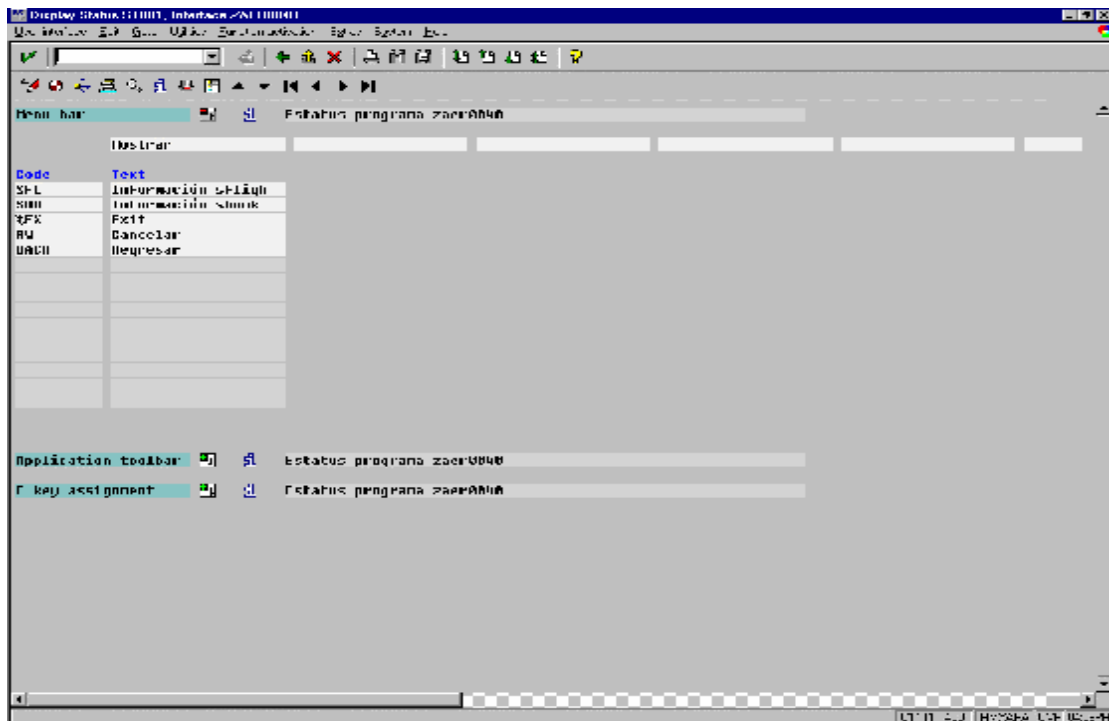
Para ver que funciones se pueden utilizar nos situaremos sobre '**Application Toolbar**' y pulsaremos **F4**.

Indicaremos el código de función que deseamos que aparezca en la barra de herramientas de aplicación. Podemos especificar si queremos que aparezca un texto corto o únicamente un icono que identifique la función.

No será necesario definir las funciones de la barra de herramientas estándar, '**Standard Toolbar**'.

Para definir iconos para visualizarlos en la barra de herramientas de aplicación será necesario:
 Seleccionar: **Edit -> Insert -> Function with icon.**
 Entrar el código de función.
 Introducir el nombre del icono y el texto del menú.

5.1.4. La Barra de Menús.



Para definir un menú, se le pone el nombre en espacio disponible. Se pueden incluir hasta 6 menús en la barra de menús. Además de los menús del usuario, el sistema añadirá automáticamente **System y Help**.

Cada menú puede tener hasta 15 entradas. Cada una de las cuales puede ser otro menú en cascada o una función.

Para abrir un menú o submenú, hacer un Doble-Click sobre el nombre. Cada entrada estará compuesta de un código de función y un texto de función o un texto de menú. Con **F4** podemos ver una lista de las funciones que podemos utilizar.

Se pueden anidar hasta 4 niveles de submenús. En el caso de las funciones bastará con indicar el código de la función para que el texto de esta aparezca automáticamente, si esta ya existe previamente. Podemos definir los atributos de una función nueva con Doble-Click sobre la nueva función definida.

En el caso de un menú en cascada, no será necesario indicar el código, y con Doble-Click podemos desarrollar las opciones del submenú.

5.1.5. Otras utilidades del Menú Painter.

5.1.5.1. Activación de Funciones.

Podemos hacer que las funciones de la barra de menús estén en modo **activo** o **inactivo**. En caso de estar inactivas, se visualizarán en la barra de menús en baja intensidad y su selección no implicará efecto alguno, en cambio las funciones activas serán completamente ejecutables.

Para activar o desactivar funciones seleccionar '**Function Activation**'.

5.1.5.2. 'FastPaths'.

Un 'FastPaths' (Camino rápido), es una manera rápida de escoger funciones en un menú, asignando a cada función una tecla.

Podemos mantener 'FastPaths' seleccionando:

Goto -> Further Options -> FastPath.

Indicaremos para las funciones deseadas una letra, normalmente la primera, teniendo cuidado de no especificar la misma letra para distintas funciones.

5.1.6. Títulos de Menu.

Es posible mantener distintos títulos para un menú.

Goto -> Title List.

Cada título se identificará con un código de título de 3 dígitos.

Introduciremos el texto del título, pudiendo utilizar variables de la misma forma que lo hacíamos con los mensajes en ABAP/4, es decir utilizando el símbolo **&**. Posteriormente será en el programa ABAP/4 donde le indiquemos que título vamos a utilizar con la instrucción:

SET TITLEBAR <cod_título> WITH <var1> <var2>...

En tiempo de ejecución el título del menú se guardará en la variable del sistema SY-TITLE.

5.1.7. Prueba, Chequeo y Generación de Status.

Podemos probar el status simulando la ejecución de la interface con:

User Interface -> Test Status, y introduciendo los datos: Número de pantalla, y Código del título.

Antes de usar la interfase podemos comprobar que la hemos definido correctamente, realizando un proceso de chequeo con: **User Interface -> Check Syntax**. Posteriormente realizaremos un proceso de generación de la interface que incluye el chequeo y la grabación de la misma.

5.1.8. Menús de ámbito o de área.

Un menú de ámbito es una agrupación de transacciones en forma de menú. Es una manera de agrupar las transacciones más frecuentemente utilizadas por un usuario bajo un mismo menú. A diferencia de una transacción de diálogo, el menú de ámbito sólo llama a otras transacciones, no pudiendo incorporar otro tipo de funciones propias.

Podemos crear los menús de ámbito con una version simplificada del Menú Painter.

Tools ... Development -> Other Tools -> Area Menus.

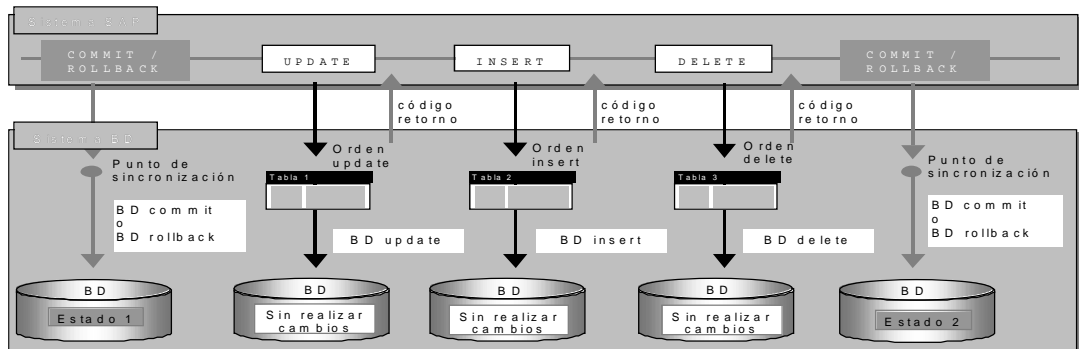
Unicamente será necesario introducir los códigos de transacción (tabla **TSTC**) y el texto del menú.

Para más información sobre las posibilidades del Menú Painter, ver el capítulo **Menú Painter** del manual '**BC ABAP/4 Worckbench Tools**' .

6. Actualización Asíncrona

6.1. Concepto de transacción.

6.1.1. Transacción de Base de Datos (DB LUW)



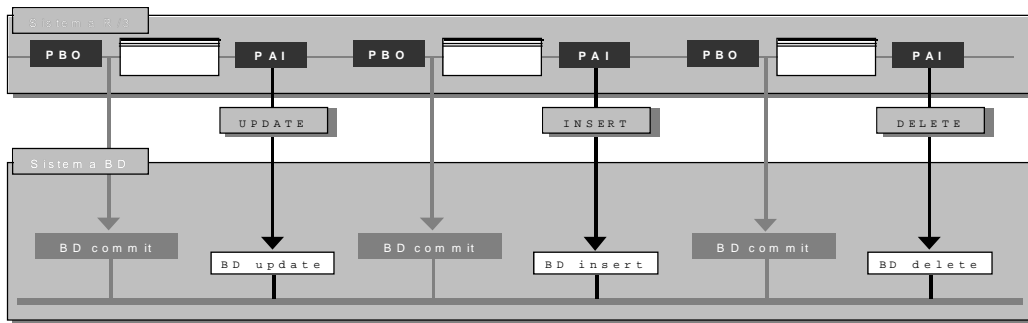
- Una transacción de base de datos es introducida por un punto de sincronización el cual es puesto por la aplicación de la base de datos (en el caso del sistema R/3).
- En el curso de una transacción de base de datos, el sistema realiza actualizaciones de la tabla, en requerimientos hechos por el sistema R/3. Las entradas a la tabla modificada permanecen bloqueadas hasta que la transacción haya terminado.
- Después de cada actualización de la base de datos, el sistema de base de datos informa al sistema R/3 que la actualización fue exitosa o no exitosa, en forma de un código de retorno.
- Una transacción de base de datos es terminada por otro punto de sincronización puesto por el sistema R/3 el cual envía un COMMIT para confirmar las actualizaciones de las tablas, o un ROLLBACK al sistema de base de datos. En respuesta, el sistema de base de datos realiza un commit para confirmar las actualizaciones de la tabla, o realiza un rollback el cual cancela las actualizaciones realizadas por la transacción de base de datos. En este caso el estado 2 es idéntico al estado 1.
- En ambos casos, los bloqueos hechos por la base de datos son liberados.

6.1.2. Transacción SAP.

- Una transacción SAP consiste en procesos de dialogo. Un proceso de dialogo comienza cuando el usuario presiona Enter, cuando activa una función presionando alguna tecla función, hace un doble clic o escoge una función de un menú. Esto termina cuando la siguiente pantalla es desplegada.

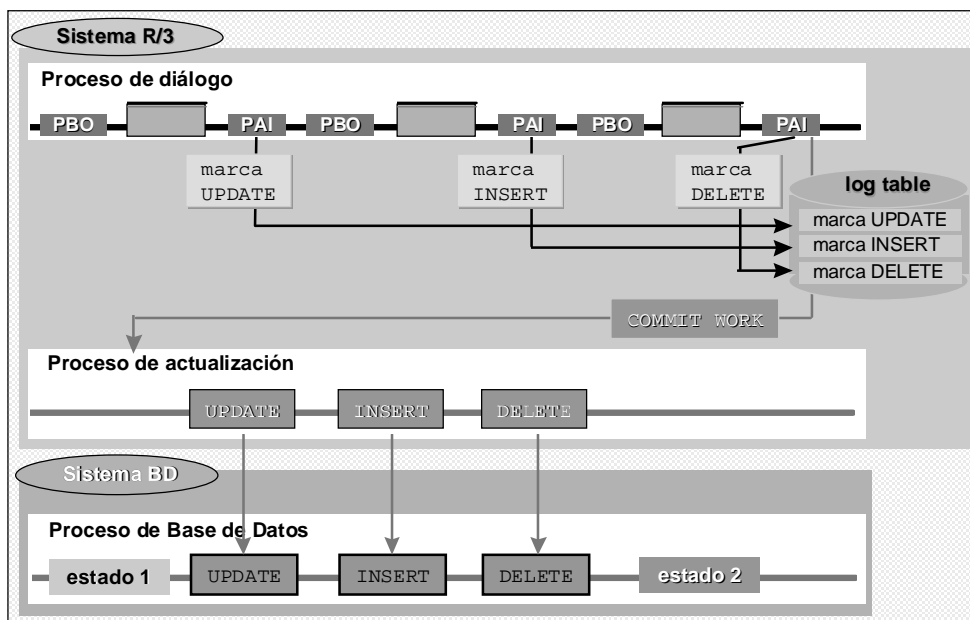
- En el curso de un proceso de dialogo, los módulos PAI pertenecen a la pantalla en ejecución y los módulos PBO pertenecen a la siguiente pantalla que es ejecutada.
- Cada proceso de dialogo puede contener requerimientos de actualización (UPDATE, INSERT, DELETE).

6.1.3. Transacción SAP y Transacción DB.



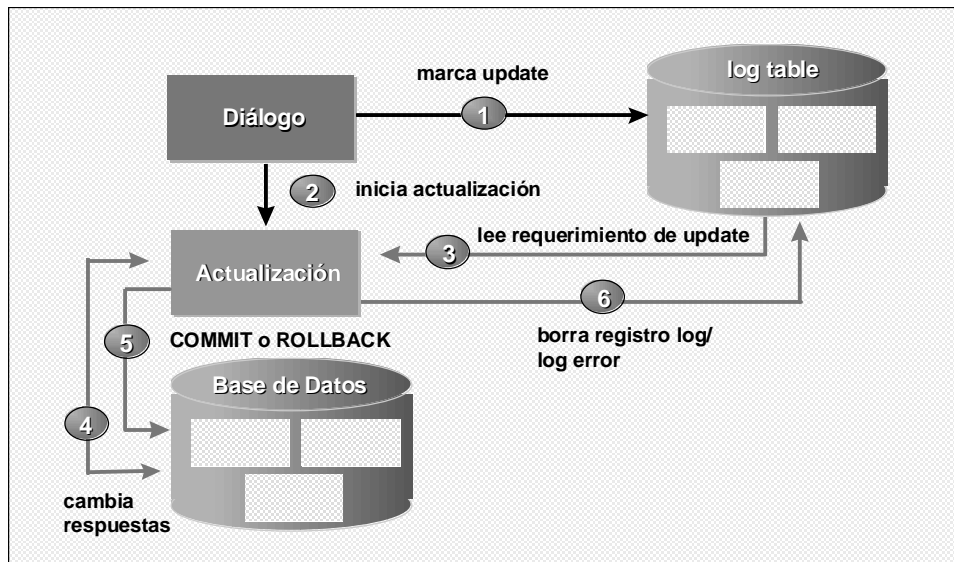
- Después de cada proceso de dialogo, el sistema R/3 automáticamente pasa un commit de base de datos al sistema de base de datos. El sistema de base de datos distribuye los requerimientos de actualización de un proceso de dialogo individual pasando por varias transacciones.
- Un rollback en un proceso de dialogo no tiene efecto en actualizaciones previamente realizadas a la base de datos en procesos de dialogo previos.

6.1.4. Transacción SAP y Actualizaciones Asíncronas.



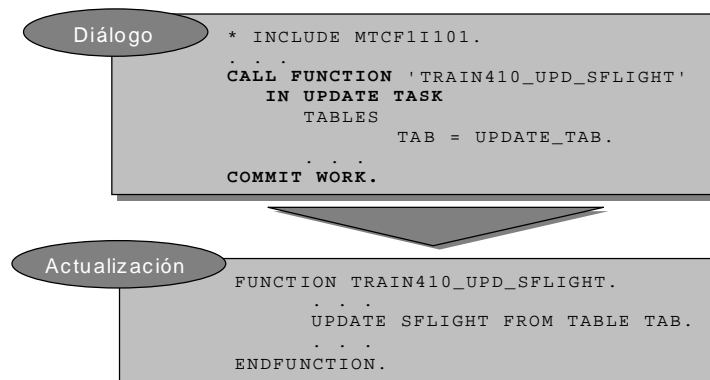
- Una actualización asíncrona permite combinar un total de actualizaciones realizadas por procesos de diálogo consecutivos de una transacción SAP, en una unidad conocida como Logical Unit of Work (unidad lógica de Trabajo) o SAP LUW.
- En el LUW, todas las actualizaciones son realizadas en una sola. Aquí los requerimientos de actualización no son pasados directamente a la base de datos, pero son almacenados en una tabla de registro (log table) para actualizarse.
- El comando de ABAP/4 COMMIT WORK concluye el LUW. El sistema R/3 comienza un proceso de actualización especial el cual usa la tabla de registro (log table) para realizar la actualización en la base de datos dentro del contexto de una transacción de base de datos.
- Si ocurre un error, el LUW es terminado por el comando ABAP/4 ROLLBACK WORK. Las entradas a la tabla de registro son descartadas y no comienza el proceso de actualización.
- El COMMIT WORK termina la tarea del diálogo y continua en la tarea de actualización.

6.2. Concepto de Actualización Asíncrona.



- En SAP, la actualización asíncrona para el manejo de requerimientos de bases de datos es dividida entre un programa de dialogo interactivo y un programa de actualización el cual corre en background.
- En SAP, la actualización asíncrona divide el proceso en actualizaciones de tiempos críticos (V1) y actualizaciones de tiempos menos críticos (V2).

6.2.1. Programa de dialogo y módulo de función para actualización.



- Para la implementación del concepto de actualización, se requiere de un programa para el proceso de dialogo y uno o más módulos de función para el proceso de actualización.
- Si en el comando **CALL FUNCTION** en el programa de dialogo tiene además **IN UPDATE TASK**, la llamada no es ejecutada inmediatamente, pero es agregada en la tabla de registro.

6.2.2. Modulo de Función de Actualización.

- Se pueden asignar módulos de función usados en una transacción SAP a diferentes grupos de funciones.
- En la sección de administración, se especifica el tipo de función: V1 (ejecución inmediata) o V2 (ejecución posteriormente).
- En un módulo de función para actualización, solo los parámetros de entrada (import) y las tablas son tomadas en cuenta. Se especifica los campos de referencia o la estructura según corresponda.
- Los parámetros de salida (export parameters) y las excepciones (exceptions), son ignorados en una función de actualización.
- Si se desea reglamentar la opción para realizar después una actualización después de un error, con la transacción SM13 seleccione *Immediate start, no restart*.

6.2.3. Programa de diálogo y tabla de registro.

- En el momento que cada **CALL FUNCTION ... IN UPDATE TASK** es ejecutada en un programa en diálogo, se agrega una entrada en la tabla registro (log table) con el nombre de la función y sus parámetros.
- Todos los requerimientos de actualización en un SAP LUW tiene la misma llave (conocida como update key).

- Un registro de encabezado (log header) para asociar los registros se genera solo cuando un COMMIT WORK es ejecutado.

6.2.4. Rollback en los programas de Dialogo: Borrando marcas de Actualización.

Programa de dialogo

```

PROGRAM . . .
MODULE <read ok-code>.
. . .
CASE <ok-code>.
  WHEN 'UPDA' .
    .
    COMMIT WORK.
  WHEN 'BACK' .
    .
    ROLLBACK WORK.
  .
ENDCASE .
. . .
ENDMODULE .
    
```

- En el curso de un dialogo que involucra varios pasos, se puede juntar una serie de actualizaciones y ejecutar el requerimiento asociado con un **COMMIT WORK** explícito.
- Sin embargo, se pueden tener para borrar todas las actualizaciones del SAP LUW en curso con **ROLLBACK WORK**. El comando ROLLBACK regresa todas las actualizaciones hechas por el LUW en ejecución.

6.2.5. Rollback en el Programa de Actualización.

Programa de Actualizació

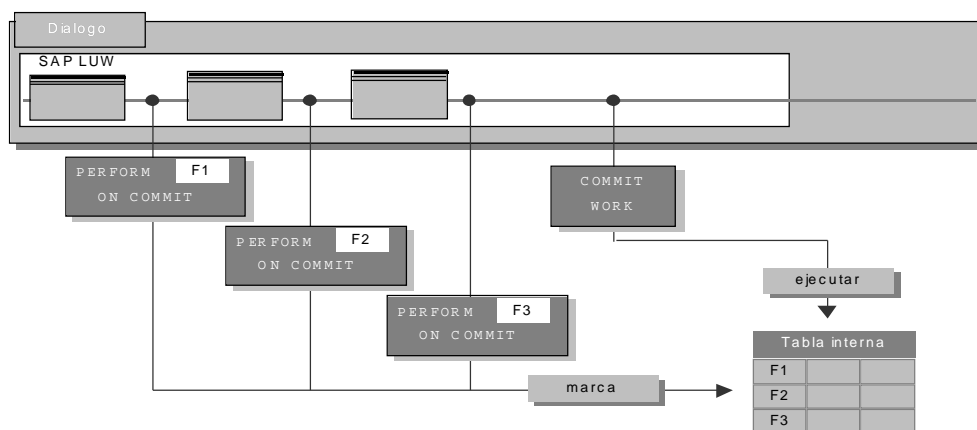
```

FUNCTION-POOL . . .
FUNCTION . . .
. . .
UPDATE . . .
IF SY-SUBRC NE 0.
  MESSAGE Annn . . .
ENDIF.

INSERT . . .
IF SY-SUBRC NE 0.
  MESSAGE Annn . . .
ENDIF.
. . .
ENDFUNCTION.
    
```

- La función de actualización pasa el requerimiento de actualización a la base de datos y analiza el código de retorno. Si la base de datos no ejecuta correctamente la actualización, en el módulo de función se decide la forma de proceder en estos casos.
- Si se desea realizar un Rollback de base de datos en el programa de actualización, es necesario desplegar un mensaje tipo abend.
- Los comandos ABAP/4 COMMIT WORK Y ROLLBACK WORK no son permitidos en módulos de función de actualización. Solo pueden ser usados en programas de diálogo.

6.2.6. PERFORM <subrutina> ON COMMIT.



- Si se ejecuta una subrutina con **PERFORM <subrutina> ON COMMIT**, no es ejecutado hasta el siguiente **COMMIT WORK**.
- Con PERFORM ... ON COMMIT no se pueden pasar parámetros.
- Con el ROLLBACK WORK, los elementos de la tabla interna son borrados. Las subrutinas que son ejecutadas con PERFORM ... ON COMMIT pueden ellas mismas contener llamadas a módulos de funciones de actualización.

```

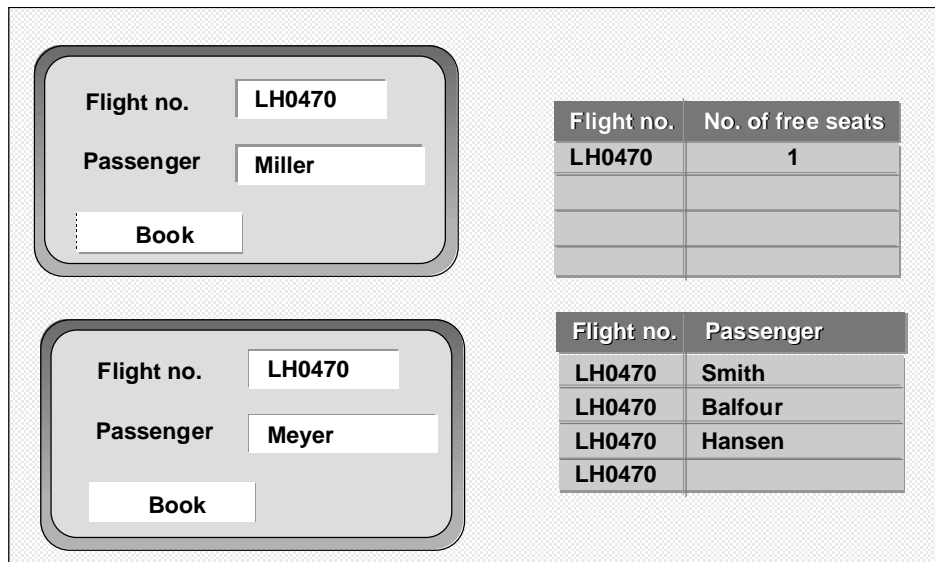
MODULE PAI_100.
    . . .
    PERFORM F1 ON COMMIT.
    . . .
ENDMODULE.
MODULE PAI_200.
    . . .
    PERFORM F2 ON COMMIT.
    . . .
    COMMIT WORK.
ENDMODULE.
. . .

FORM F1.
    CALL FUNCTION 'UPDATE_LFA1'.
    IN UPDATE TASK
    
```

```
EXPORTING . . . .  
    . . .  
ENDFORM.  
FORM F2.  
    CALL FUNCTION 'UPDATE_LFB1'.  
        IN UPDATE TASK  
        EXPORTING . . . .  
    . . .  
ENDFORM.
```

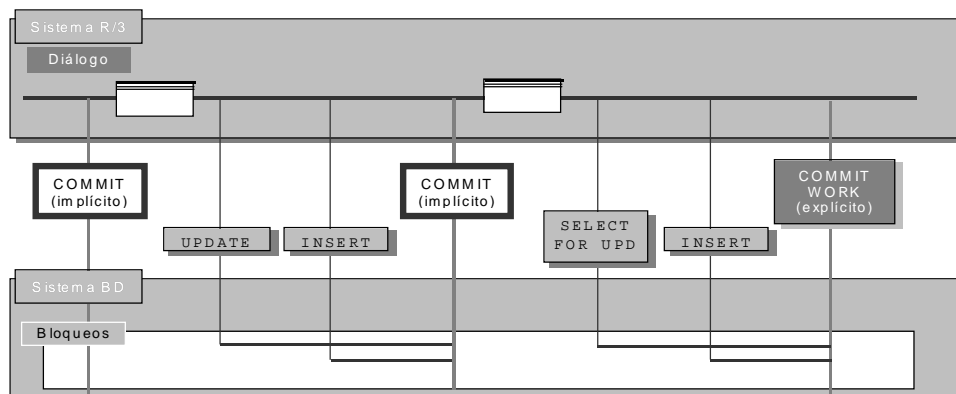
7. Concepto de Bloqueo de SAP.

7.1. Utilización de bloqueos



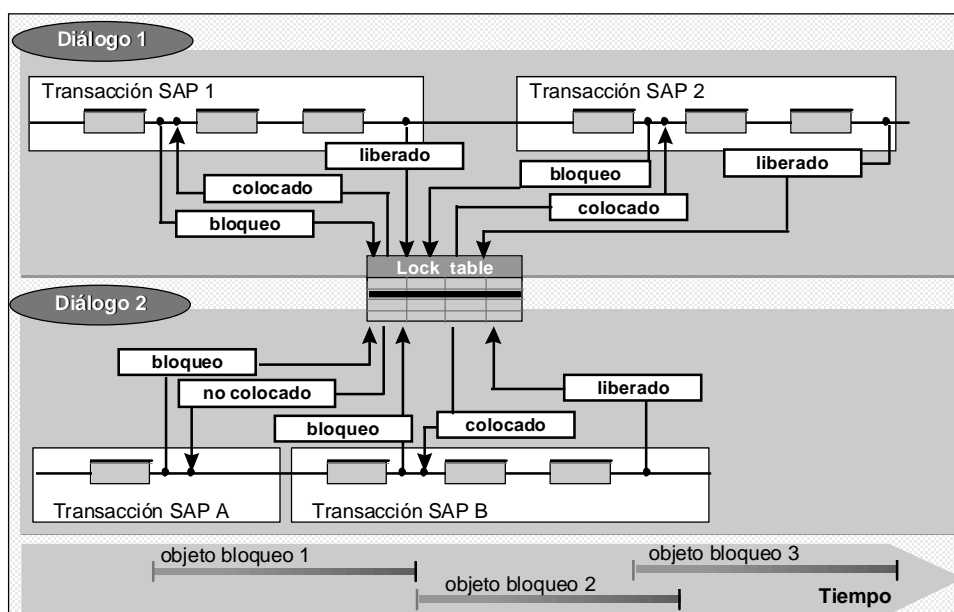
- Si varios usuarios quieren tener acceso a un mismo recurso, éstos deben estar sincronizados para garantizar la consistencia de los datos.
- Los bloqueos, constituyen un conveniente método para coordinar los accesos de cada usuario a los recursos. Cada usuario requiere de un bloqueo antes de tener acceso a datos críticos.

7.1.1. Bloqueo de Base de Datos.



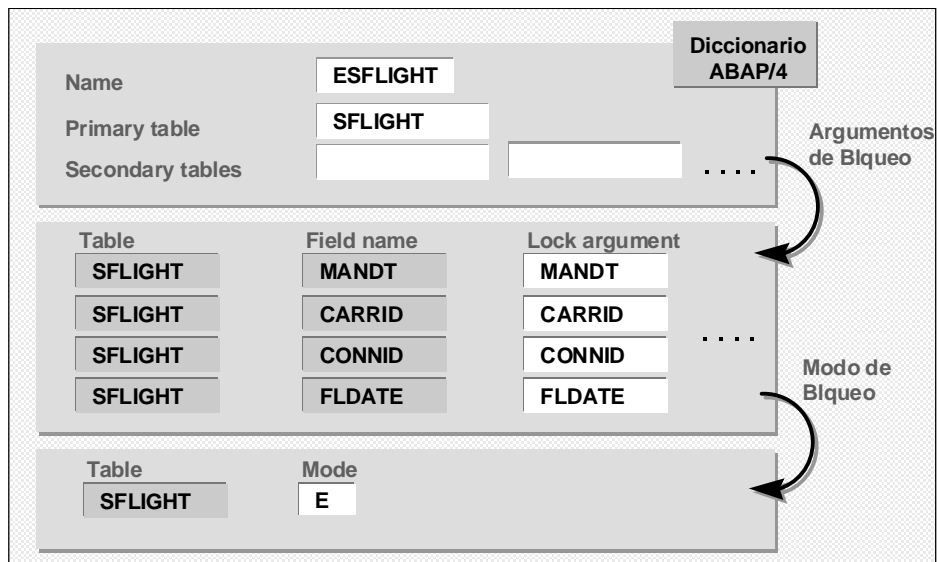
- Si un programa de diálogo contiene estatutos de actualización, el sistema de base de datos pone los bloqueos apropiados.
- Al final de una transacción de base de datos, el sistema de base de datos libera todos los bloqueos puestos durante la transacción.
- No obstante, el sistema R/3 realiza un commit implícito a la base de datos en cada cambio de pantalla, el bloqueo de base de datos puesto durante un paso de diálogo, solo es retenido mientras no termine este paso e inicie el siguiente.

7.1.2. Introducción al Concepto de Bloqueo de SAP.



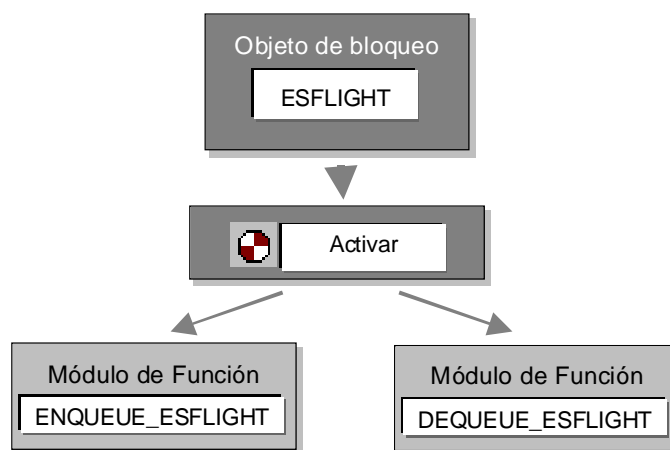
- Los bloqueos de base de datos son insuficientes si se desea que el bloqueo tenga una duración de varios cambios de pantalla.
- En el contexto de bloqueo de base de datos de SAP, hay una transacción SAP la cual coloca los bloqueos en una tabla (lock table) para que los registros de la tabla sean procesados.
- La transacción SAP obtiene información sobre el suceso del bloqueo para retornar un código de retorno.
- Al final del proceso, los bloqueos deben ser liberados **explícitamente** por el programa de diálogo.
- Si el usuario termina el programa de diálogo (tecleando en la línea de comandos /n, o dentro del programa se ejecuta un estatuto LEAVE PROGRAM o LEAVE TO TRANSACTION; o si es desplegado un mensaje tipo A), los bloqueos son liberados automáticamente.

7.1.3. Objetos de Bloqueo SAP.



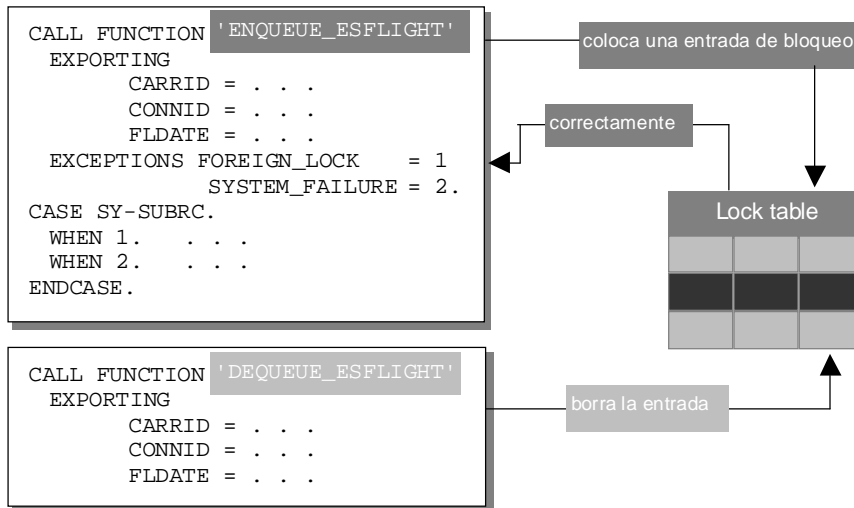
- Para realizar un bloqueo, primeramente se define un objeto de bloqueo en diccionario de ABAP/4. Estos objetos cubren las tablas que van a ser bloqueadas.
- Consiste de una tabla primaria, pero también se pueden agregar otras tablas secundarias para usar llaves foráneas dependientes.
- El nombre de un objeto para clientes debe iniciar con EY o EZ.
- Los argumentos del bloqueo son los campos que forman la llave de las tablas.
- Por cada tabla se puede definir el modo de bloqueo: Modo E para exclusivo, modo S para compartido.

7.1.4. Modulo de Función Enqueue / Dequeue.



- Cuando un objeto bloqueado ha sido activado exitosamente, el sistema genera unos módulos de función ENQUEUE y DEQUEUE.

7.1.5. Llamando Módulos de Bloqueo.



- Cuando se llama un módulo de función ENQUEUE, el programa de diálogo coloca un bloqueo.
- Los parámetros de exportación que se refieren al bloqueo de argumentos, identifican las entradas de la tabla o las entradas a ser bloqueadas.
- Si uno de estos parámetros no contienen un valor, el sistema lo trata como una especificación genérica.
- Si se desea borrar todos los bloqueos de la tabla, los cuales han sido puestos en tu programa se puede usar el módulo de función DEQUEUE_ALL.

7.1.6. Tabla de Bloqueos

- Los argumentos de los bloqueos, son almacenados en la lock table para cada tabla bloqueada.

Client	User	Time	Shared	Table	Lock argument
007	Meyer	11:00		SFLIGHT	LH470
007	Miller	11:01		SFLIGHT	UA250
007	Miller	11:01		LFB1	0074712 0815
007	Miller	11:01		LFC1	0074712 08151994
007	Smith	11:02	X	YLFA	LIEF1
007	Schultz	11:03	X	YLFA	LIEF1
007	Balfour	11:04		YLFB1	00764715

- Para desplegar la tabla de bloqueos, elija *Tools -> Administration -> Lock entries* (Transacción SM12).

7.1.7. Características Especiales con ENQUEUE.

- El parámetro `MODE_<table>` maneja el modo de bloqueo definido por el bloqueo del objeto ('S' = shared / compartido, 'E' = Exclusivo).
- El parámetro `_SCOPE` define la duración del bloqueo y los libera cuando no se necesitan más.

`_SCOPE = 1` : El Bloqueo permanece en el programa de dialogo.

`_SCOPE = 2` (default) : El bloque es retenido por el programa de actualización.

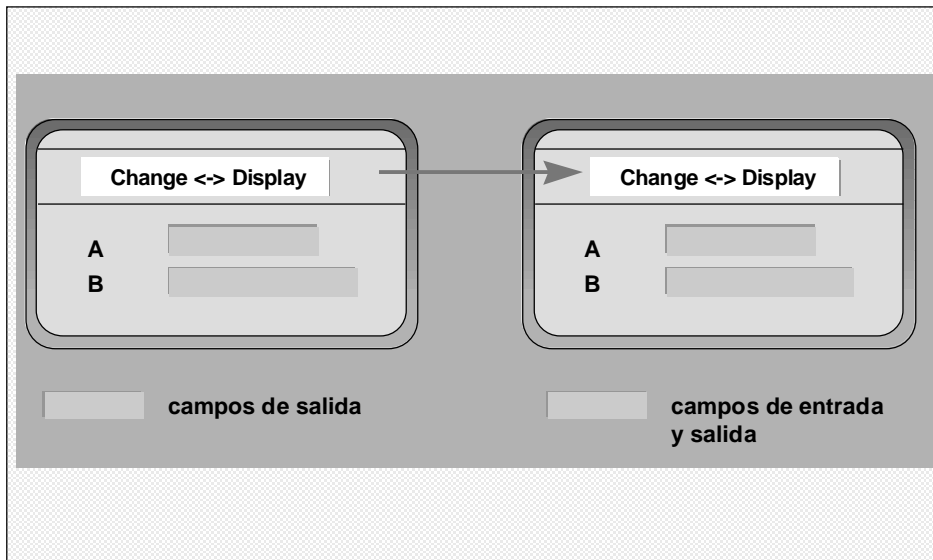
`_SCOPE = 3` : El programa de dialogo y actualización son dueños de los bloqueos y hay dos entradas por objeto.

- El parámetro `_WAIT` define donde un bloqueo deberá repetirse después de un error. Se pueden especificar la duración de las repeticiones con el parámetro `ENQUEUE / DELAY`.
- Si un argumento de bloqueo no contiene valor o tiene un espacio, se establece un bloqueo genérico. Si se desea bloquear el valor inicial, se debe marcar el parámetro `X_<lock argument>`.

8. Modificación dinámica de pantallas

8.1. Modificación dinámica

8.1.1. Introducción



Se pueden cambiar temporalmente ciertos atributos de campos, por ejemplo cambiar los campos de solo-lectura en campos de entrada/salida.

También se puede usar la modificación dinámica de pantallas para facilitar el ocultar ciertos campos y así evitar secuencias dinámicas de pantallas.

8.1.2. Atributos de campos Modificables

Los campos de pantalla y sus atributos modificables son automáticamente almacenados en la tabla interna SCREEN.

La tabla SCREEN es inicializada con los campos definidos en el Screen Painter y con sus atributos cada vez que el módulo PBO es ejecutado.

Para determinar los campos para los cuales se puede cambiar uno ó más atributos, se lee el campo SCREEN-NAME y del campo SCREEN-GROUP1 al campo SCREEN-GROUP4 en la tabla SCREEN.

El campo SCREEN-REQUEST está reservado para uso interno del sistema.

Tabla SCREEN / Atributos modificables de campos

SCREEN-NAME	<i>Field name</i>
SCREEN-GROUP1	<i>Modification group1</i>
SCREEN-GROUP2	<i>Modification group2</i>
SCREEN-GROUP3	<i>Modification group3</i>
SCREEN-GROUP4	<i>Modification group4</i>
SCREEN-REQUIRED	<i>Required field</i>
SCREEN-INPUT	<i>Input field</i>
SCREEN-OUTPUT	<i>Output field</i>
SCREEN-INTENSIFIED	<i>Highlighted field</i>
SCREEN-INVISIBLE	<i>Invisible field</i>
SCREEN-LENGTH	<i>Field length</i>
SCREEN-ACTIVE	<i>Active field</i>
SCREEN-DISPLAY_3D	<i>3-dimensional field</i>
SCREEN-VALUE_HELP	<i>Field with value help</i>
SCREEN-REQUEST	<i>Input exist (PAI only)</i>

8.1.3. Atributos: Modificación de grupos

Lista de campos: Modificación de grupos					
Field name	Gr1	Gr2	Gr3	Gr4	...
SPFLI_ITAB-CONNID	SEL				
SPFLI_ITAB-CITYFROM	SEL				
SPFLI_ITAB-CITYTO	SEL				
...					

Screen Painter

Se puede asignar un campo a cuatro grupos diferentes. Los nombres de grupos son de tres caracteres de longitud y pueden ser definidos libremente.

8.1.4. Programa

Para programar la modificación de la pantalla, es necesario un módulo que se ejecuta durante el módulo PROCESS BEFORE OUTPUT.

Se pueden cambiar los atributos de un campo y/o un grupo de campos en la tabla SCREEN. (LOOP AT SCREEN WHERE ... y READ TABLE SCREEN son soportados.)

Se activan o desactivan los atributos asignando valores 1 ó 0. Para almacenar los cambios realizados, se utiliza la instrucción MODIFY SCREEN.

No es permitido el usar la instrucción SCREEN-ACTIVE = 1 como una operación dinámica para activar un campo que no se ha definido como oculto en el Screen Painter. Sin embargo, si se puede usar esta instrucción para inactivar un campo definido como visible en el Screen Painter. SCREEN-ACTIVE = 0 tiene el mismo efecto que las tres instrucciones SCREEN-INVISIBLE =1, SCREEN-INPUT = 0 y SCREEN-OUTPUT = 0.

```
PROCESS BEFORE OUTPUT.  
.  
.  
MODULE MODIFY_SCREEN.  
.  
.
```

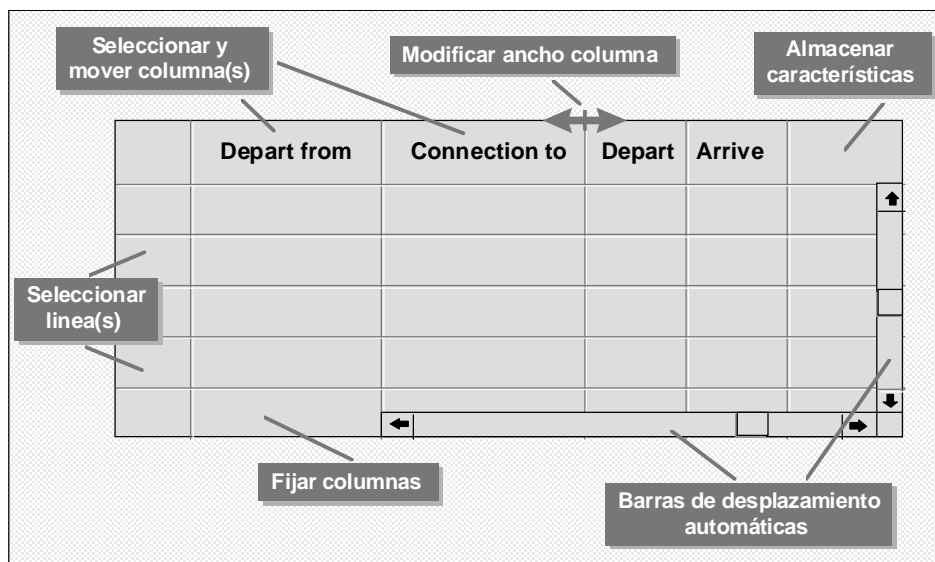
**Screen
Painter**

```
MODULE MODIFY_SCREEN OUTPUT.  
LOOP AT SCREEN.  
IF SCREEN-GROUP1 = 'SEL'.  
SCREEN-INPUT = 1.  
ENDIF.  
IF SCREEN-NAME = 'SFLIGHT-CARRID'.  
SCREEN-ACTIVE = 0.  
ENDIF.  
MODIFY SCREEN.  
ENDLOOP.  
ENDMODULE.
```

ABAP/4

9. Tablas de Control

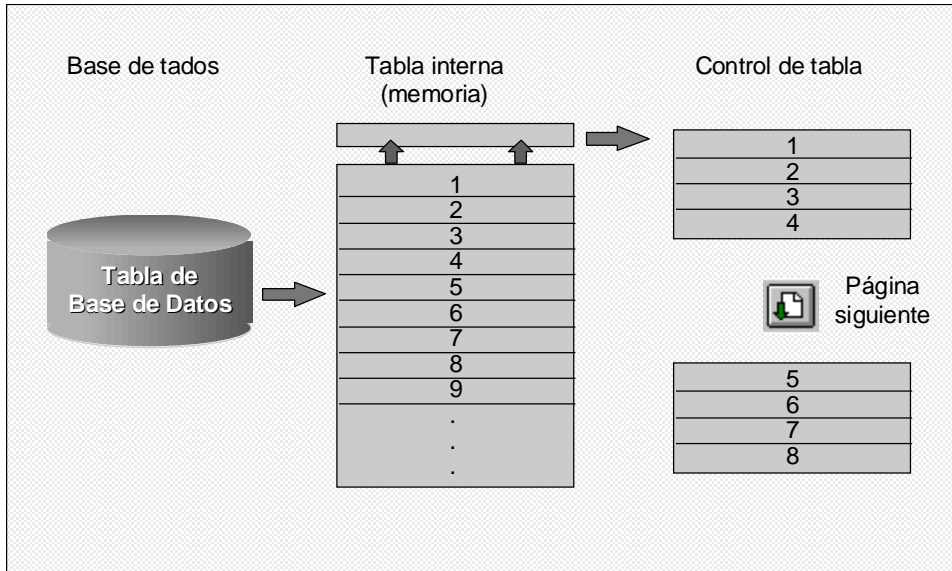
9.1. Características del Control de Tabla



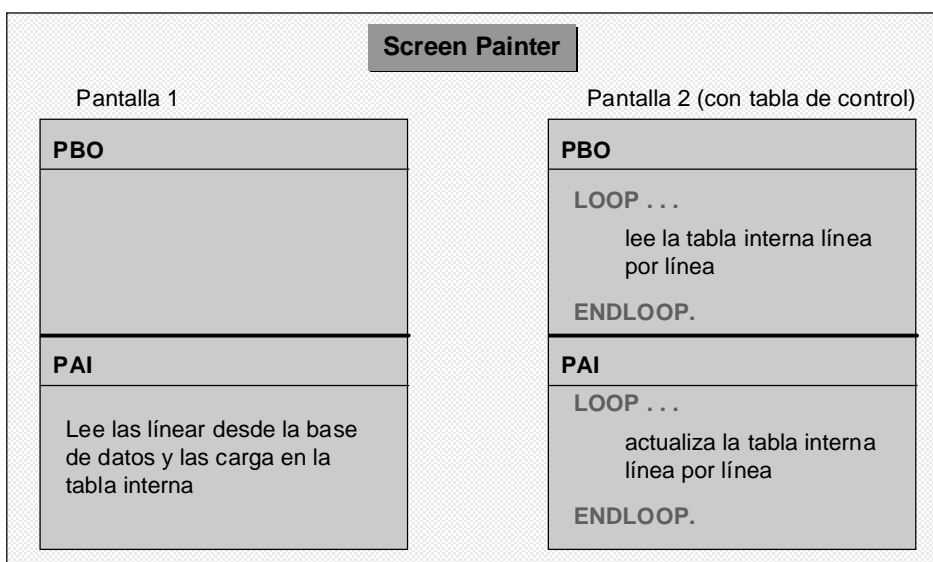
- Con el control de tabla, puedes mostrar o introducir líneas y/o datos en forma tabular.
- Alcance de Función:
 - Cambiar de tamaño la tabla para desplegar y editar dato.
 - El ancho y posición de columna puede ser modificado por el usuario.
 - La selección de columna y/o línea selección con color-intenso.
 - Selección de línea(s), múltiple, total y de-seleccionar.
 - Encabezados de columna son mostrados como botones para selección de columna.
 - Desplazamiento horizontal y vertical con barras de desplazamiento.
 - Compuesto de algún número de columnas clave (columnas fixed lead).
 - Los atributos de celda modificables.
- El usuario puede almacenar diferente características variables y colocar alguno de estos o el las características básicas como el actual.

9.2. Principios para el control de Tabla.

9.2.1. Llenado.

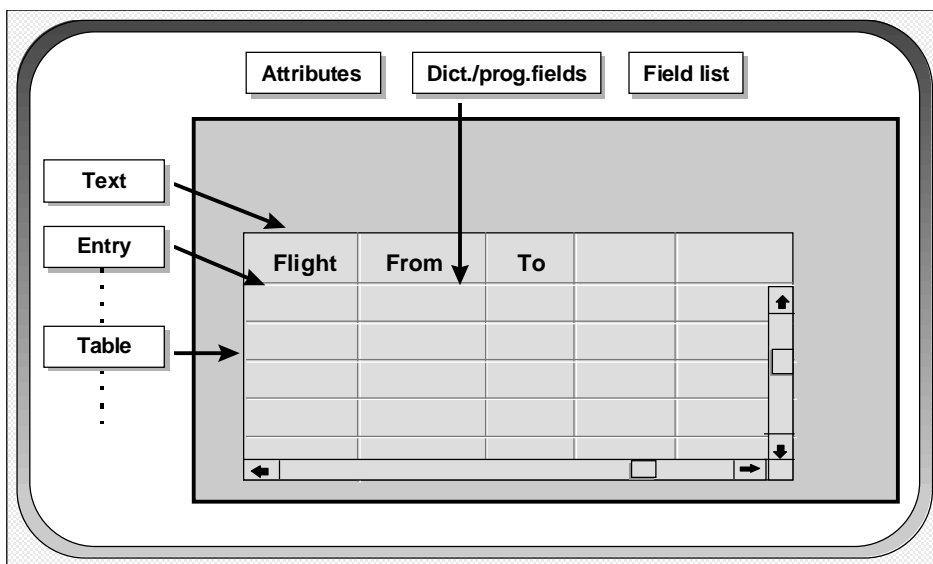


- Por razones de optimización, lees los datos para la tabla de control una vez y los almacenas en una tabla interna. Las líneas de la tabla de control son entonces tomadas de esta tabla interna.
- Hay solo un área de trabajo para editar líneas dentro de la tabla de control. Por esta razón, necesitas una instrucción LOOP ... ENDLOOP para cada tabla de control en los módulos PBO y PAI del flujo lógico.



- En el módulo PBO, una línea de la tabla de control debe estar llena con una línea de la tabla interna cada vez que el ciclo es procesado.
- Similarmente, el módulo PAI debe copiar los cambios en una línea de tabla de control dentro de la línea de tabla interna correspondiente.
- Cuando manejas funciones, debes diferenciar entre las que se aplican solo en una línea en una tabla de control y las relacionadas con la pantalla completa.

9.2.2. Creación en modo Gráfico

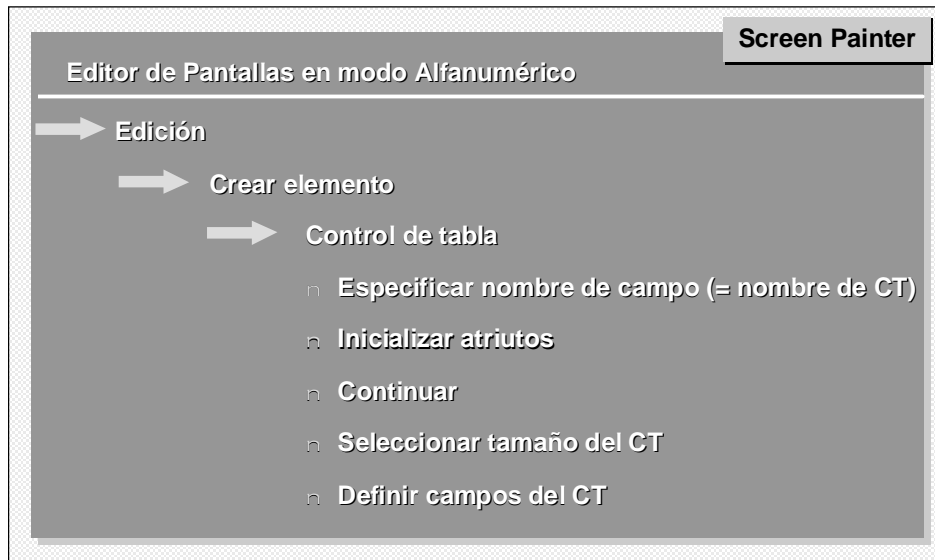


- En el editor *fullscreen gráfico*, elige *Table* para crear una tabla de control y usa el botón izquierdo del ratón para posicionarlo en la pantalla.
- Entonces, define los campos en la tabla de control, por ejem: usando estos en ABAP/4 Dictionary.

9.2.3. Creando Tablas de control (Fullscreen Alfanumérico)

- Puedes definir una tabla de control en el editor alfanumérico fullscreen.
- En el menú *Edit*, elige *Create element* y entonces *Table control*.
- Obtienes un cuadro de diálogo donde introduces el nombre de la tabla de control e inicializas los atributos.
- Vas al modo de selección y determinas el tamaño de una tabla de control posicionando el control. Entonces, defines los campos de la tabla de control.

- Puedes usar también campos de ABAP/4 Dictionary o crear nuevos en el programa.



9.2.4. Definición de una tabla de control en "Module Pool"

CONTROLS ctrl TYPE TABLEVIEW USING SCREEN scr.

El tipo **TABLEVIEW** corresponde a la estructura **CXTAB_CONTROL** con los siguientes campos:

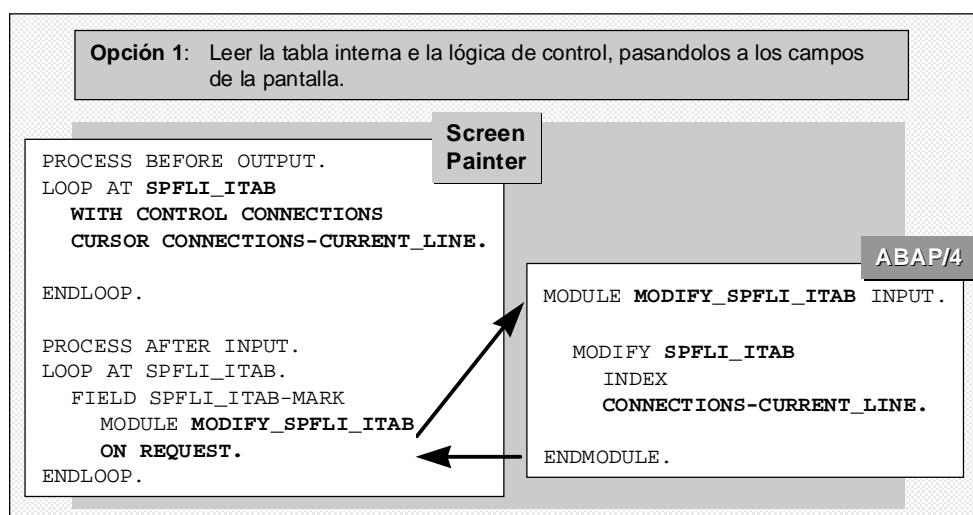
FIXED_COLS	TYPE I	Número de columnas fijas
LINES	TYPE I	Número de líneas para el desplazamiento vertical
TOP_LINE	TYPE I	Primera línea en el siguiente PBO
CURRENT_LINE	TYPE I	Línea actual (en un LOOP ... ENDLOOP)
LEFT_COL	TYPE I	Primera columna desplegada y movable
LINE_SEL_MODE	TYPE I	Selección de línea (0=ninguna, 1=simple, 2=múltiple)
COL_SEL_MODE	TYPE I	Selección de columna (0=ninguna, 1=simple, 2=múltiple)
LINE_SELECTOR		Indicador de línea seleccionada
V_SCROLL		Indicador de barra de desplazamiento vertical
H_GRID		Indicador de línea de grid horizontal
V_GRID		Indicador de línea de grid vertical
COLS	TYPE CXTAB_COLUMN	OCCURS 10

El tipo **CXTAB_COLUMN** consta de los siguientes campos:

SCREEN	LIKE SCREEN	Atributos de la estructura SCREEN
INDEX	TYPE I	Posición de columna (secuencia de despliegue)
SELECTED		Indicador de columna seleccionada
VISLENGTH	LIKE ICON-OLENG	Ancho visible de columna
INVISIBLE		Indicador de columna invisible

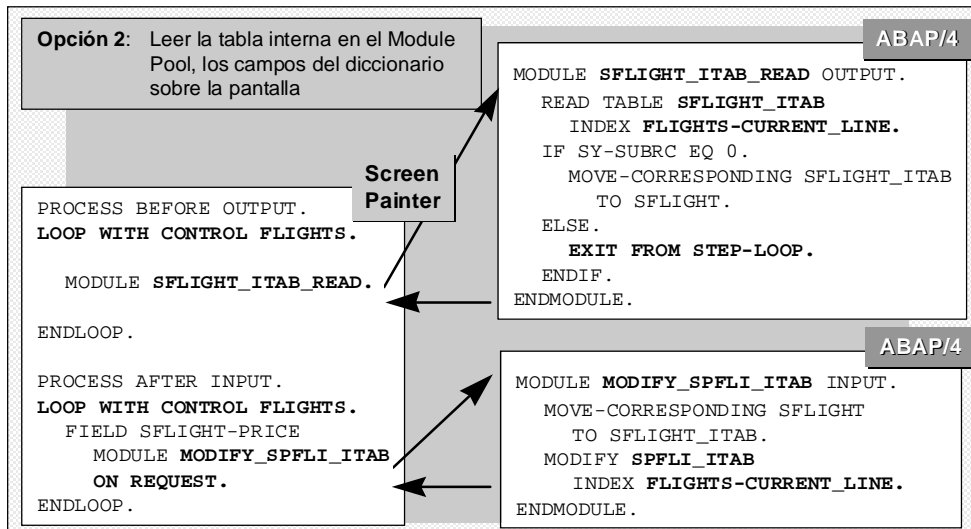
- La instrucción **CONTROLS** define un objeto de dato complejo de el tipo **TABLEVIEW** el cual corresponde a el tipo **CXTAB_CONTROL** definido en el ABAP/4 Dictionary (ver grupo tipo **CXTAB**).
- Mantienes los valores iniciales en el Screen Painter. Con la adición **USING SCREEN**, determinas la pantalla de la cual quieres obtener los valores iniciales para la tabla de control.
- Puedes usar la instrucción **REFRESH CONTROL ctrl FROM SCREEN scr** alguna vez para inicializar una pantalla con valores iniciales de la pantalla screen.

9.2.5. Flujo lógico de la tabla de control.



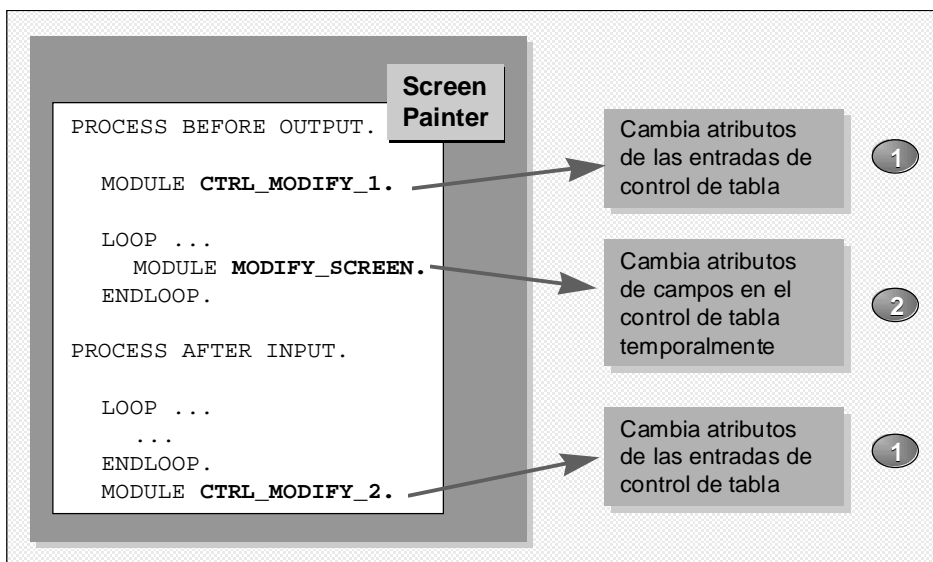
- En el flujo lógico, puedes leer una tabla interna usando la instrucción **LOOP**. Defines la referencia a la tabla de control especificando **WITH CONTROL <ctrl>**.
- Determinas cual entrada de tabla es leída especificando **CURSOR<ctrl>-CURRENT_LINE**. El sistema calcula $\langle ctrl \rangle\text{-CURRENT_LINE}$ de $\langle ctrl \rangle\text{-TOP_LINE} + SY\text{-STEPL}$, el cual es el ciclo especial indexado por **LOOPS** en el flujo lógico.
- El sistema calcula $\langle ctrl \rangle\text{-TOP_LINE}$ cuando el usuario desplaza **scrolls** con la scroll bar, pero no hace lo mismo para desplazar una página. Tú tienes que programarlo.
- Después de leer la operación, el contenido del campo es colocado en la línea de encabezado de la tabla interna. Si los campos en la tabla de control tienen los mismos nombres que los campos de la tabla interna, enseguida son llenados.
- Debes reflejar algunos cambios que el usuario hace a los campos de una tabla de control en la tabla interna. De otra manera, no aparecerán cuando la pantalla se vuelve a mostrar después de que **PBO** es ejecutado otra vez, por ejem. Cuando el usuario presiona **ENTER** o **scrolls**.

- Sin embargo, este procesamiento se debe ejecutar solo si los cambios han sido hechos a la línea de la tabla de control.

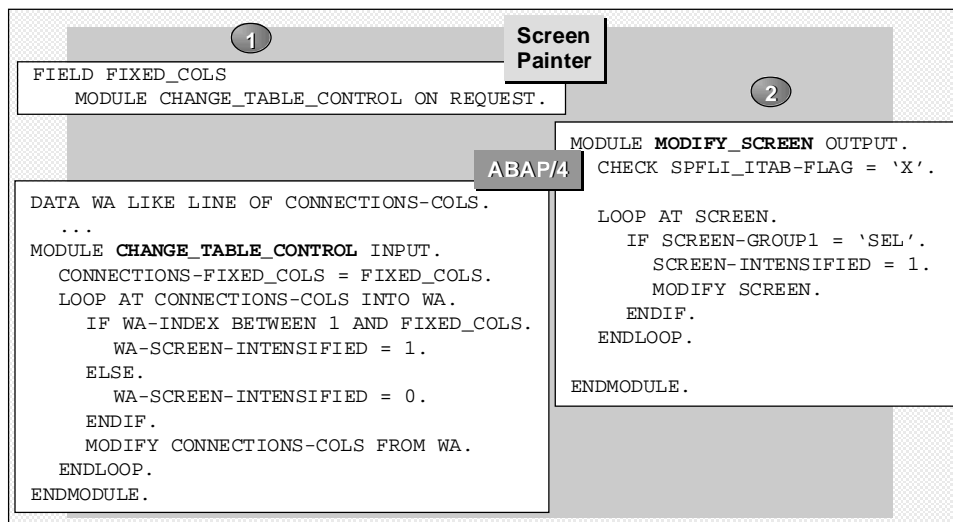


- Si usas una instrucción LOOP sin una tabla interna en el flujo lógico, debes leer el dato en un módulo PBO el cual es llamado cada vez que el ciclo es procesado.
- Ya que, el sistema no puede determinar el número de entradas de tabla interna, se debe usar la instrucción EXIT FROM STEP-LOOP para asegurar que no hay líneas vacías blank desplegadas en la tabla de control si no hay más entradas de tablas interna correspondientes.
- También debes determinar el número de líneas para desplazamiento vertical como punto conveniente. <ctrl>-LINES = número de entradas.

9.2.6. Modificación



- Puedes cambiar los atributos de la tabla de control por los contenidos de campos.
- Para modificar los atributos de celdas individuales temporalmente (!), cambias la tabla SCREEN en un módulo PBO que es procesado entre LOOP y ENDLOOP en el flujo lógico. (Use LOOP AT SCREEN, MODIFY SCREEN).



- Ejemplo 1 muestra como puedes reaccionar al requerir el componente de una columna. Si el usuario requiere el componente de una columna, la tabla de control es por consiguiente cambiada (CONNECTIONS-FIXED_COLS = FIXED_COLS). Los componentes de columnas son remarcados en color. Este es almacenado cambiando la tabla de control.
- Ejemplo 2 muestra una posible reacción de una línea seleccionada. Durante PAI, el campo ayuda itab-FLAG es llenado con 'X'. En PBO, el procesamiento reacciona a esto remarcando el color todos los campos en una línea donde el itab-FLAG es colocado a 'X' (todos ellos tienen el grupo atributo 'SEL'). Esto es almacenado cambiando la tabla SCREEN.

9.2.7. Control de páginas

- Para implementar desplazamiento de página, calculas usando el atributo de tabla de control <ctrl>-TOP_LINE.
- Para esto (en PAI), necesitas conocer el número actual de líneas en la tabla de control.
- En PBO, el campo de sistema SY-LOOPC contiene el número actual de líneas de tabla de control. En PAI, contiene el número de líneas actualmente llenas.
- SY-LOOPC solo contiene un valor entre LOOP y ENDLOOP porque cuenta el número de ciclos.
- Note por favor que debes estar preparado para repartir algún overflow

```

PROCESS BEFORE OUTPUT.

LOOP ... .
  MODULE GET_LOOPLINES.
ENDLOOP.

PROCESS AFTER INPUT.

LOOP ... .
  ...
ENDLOOP.

MODULE USER_COMMAND_0200.

```

```

DATA: LOOPLINES LIKE SY-LOOPC
...
MODULE GET_LOOPLINES OUTPUT.
  LOOPLINES = SY-LOOPC.
ENDMODULE.

MODULE USER_COMMAND_0200 INPUT.
...
WHEN 'F21'.
  FLIGHTS-TOP_LINE = 1.
WHEN 'F22'.
  FLIGHTS-TOP_LINES =
  FLIGHTS-TOP_LINES - LOOPLINES.
  IF FLIGHTS-TOP_LINES < 1.
    FLIGHTS-TOP_LINES = 1.
...
WHEN 'F24'.
  FLIGHTS-TOP_LINE =
  FLIGHTS_ITAB_LINES
  - LOOPLINES + 1.

ENDMODULE.

```

9.2.8. Posición del cursor

Sintaxis

GET CURSOR

FIELD f
VALUE v
LINE l
OFFSET o.

SET CURSOR

FIELD f
VALUE v
LINE l
OFFSET o.

Determinar posición del cursor:

¿Cuál entrada de la tabla interna corresponde a la línea de la tabla de control?

ABAP/4

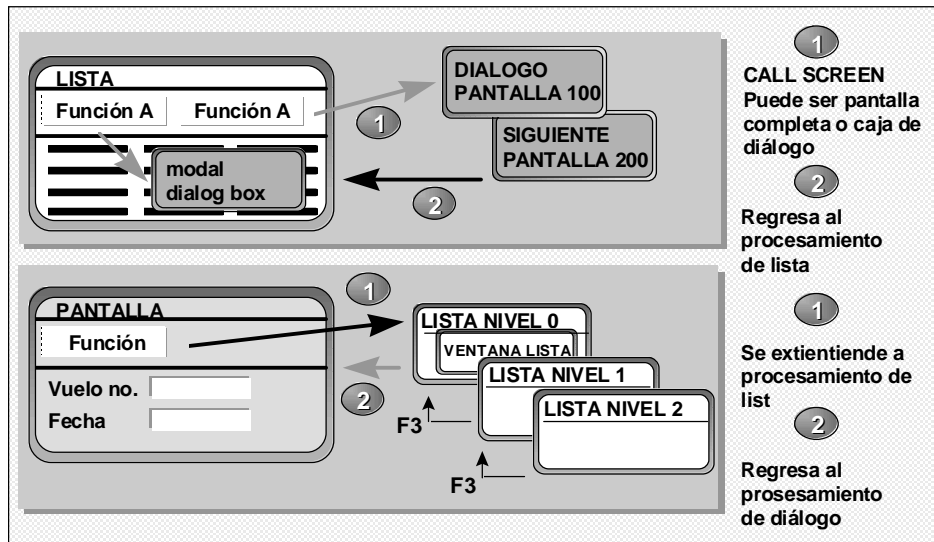
```

DATA: SELLINE LIKE SY-STEPL,
      TABIX LIKE
          CONNECTIONS-TOP_LINE.
...
GET CURSOR LINE SELLINE.
TABIX = CONNECTIONS-TOP_LINE
        + SELLINE -1.
READ TABLE SPFLI_ITAB
INDEX TABIX.
                
```

- El parámetro LINE en la instrucción GET o SET se refiere a el campo de sistema SY-STEPL, el cual es el ciclo de flujo lógico especial indexado.
- Para determinar la entrada de tabla interna que corresponde a la línea de tabla de control seleccionada, se calcula como sigue:
- Línea requerida = <ctrl>-TOP_LINE + línea de posición del cursor determinada -1.
- La instrucción GET CURSOR es apoyada por el código de regreso encontrado en el campo de sistema SY-SUBRC. Si el valor es 0, el cursor es posicionado en un campo. Si el valor es 4, el cursor no está en un campo.

10. Vinculación con los componentes de programas

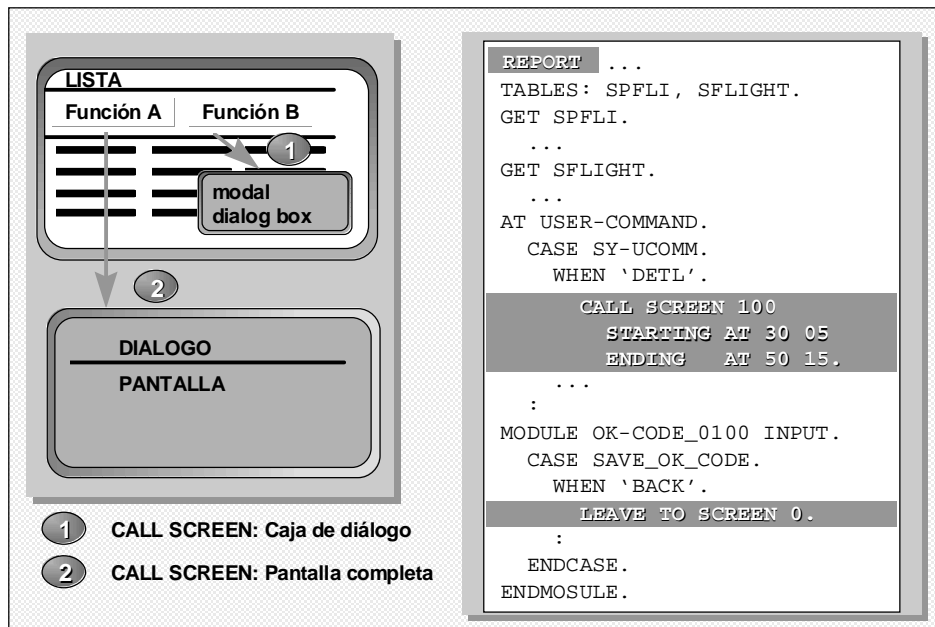
10.1. Diálogos (pantallas) y procedimientos de listas



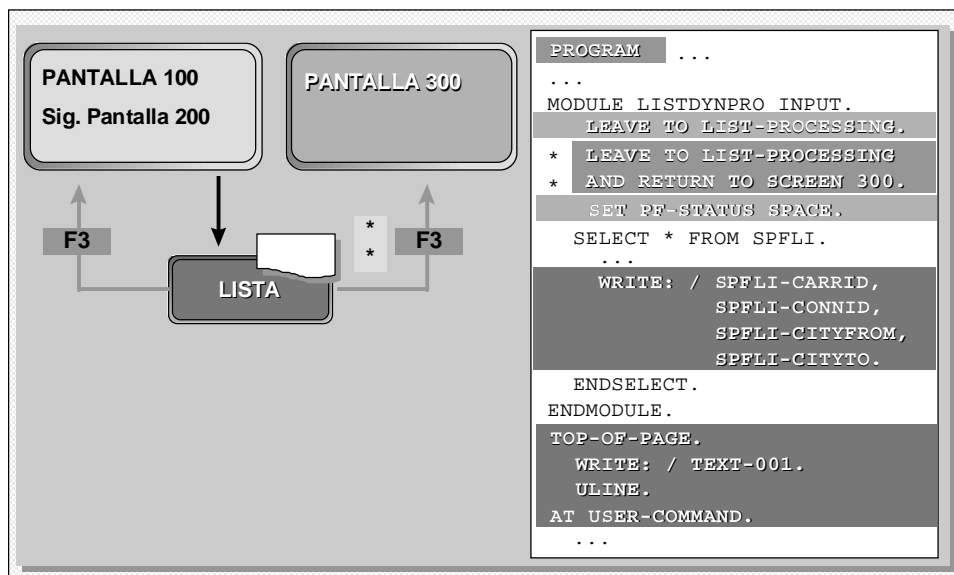
- Puedes combinar procesamiento de diálogo (pantalla) y procesamiento de lista.
- Del procesamiento de lista, puedes activar una secuencia de pantallas. Estas pantallas pueden ser mostradas como ventanas modales o como pantallas completas.
- Del procesamiento de pantallas, puedes activar el procesamiento de lista. Las listas pueden ser mostradas en ventanas o en pantallas de tamaño completo como listas básicas o listas detalladas. En el procesamiento de listas, puedes usar todos los reportes de técnicas interactivas.

10.1.1. Ramificación de una lista a un procesamiento de diálogo

- En el procesamiento de lista, hay un número de eventos interactivos que puedes usar para responder a funciones ejecutadas por el usuario. La más importante de estas son AT LINE-SELECTION y AT USER-COMMAND. El código función llamado por el usuario es almacenado en el campo de sistema SY-UCOMM.
- Puedes usar la instrucción CALL SCREEN para llamar pantallas como cuadros de diálogo o fullscreen de la lista.
- La pantalla llamada es un componente del reporte.
- La instrucción SET SCREEN 0 y, en algunos casos, LEAVE SCREEN (o LEAVE TO SCREEN 0) te permite regresar a el lugar donde la pantalla fue llamada originalmente.



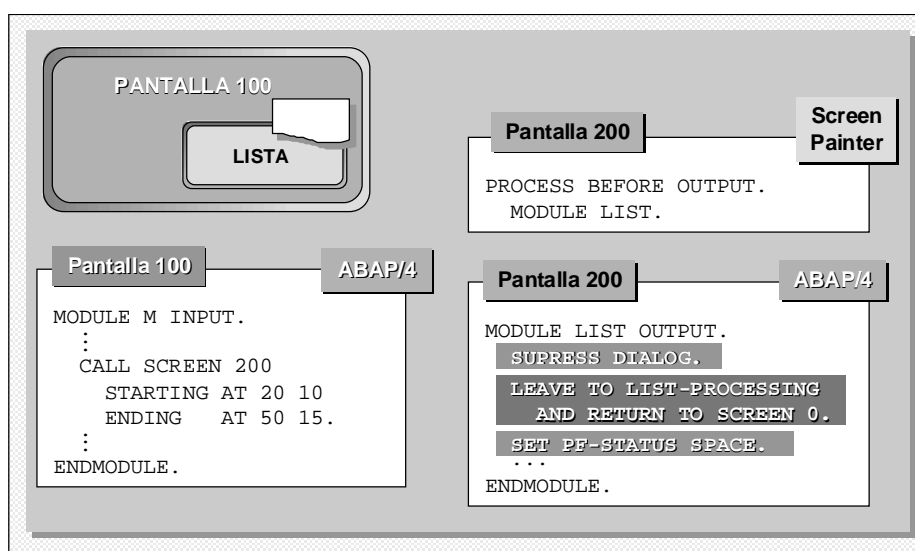
10.1.2. Ramificación de un diálogo a un procesamiento de listas



- Usas la instrucción **LEAVE TO LIST-PROCESSING** para activar de un procesamiento de diálogo a un procesamiento de lista. Para procesar la lista, puedes usar todas las instrucciones asociadas con procesamiento de listas, por ejem. WRITE, ULINE, SKIP etc.; tan pronto los eventos específicos de listas.

- Antes de que la lista sea mostrada, todos los módulos PAI de la pantalla actual son procesados.
- Usa la instrucción **SET PF-STATUS SPACE** para colocar el estatus de lista estándar.
- Cuando el procesamiento de lista ha concluido (como un resultado de que el usuario ha presionado F3 en la lista básica, o programa-driven por la instrucción LEAVE LIST-PROCESSING), el procesamiento de diálogo resumes en la pantalla siguiente en la secuencia. Especificando **AND RETURN TO SCREEN <scr>**, puedes sobre escribir la pantalla siguiente.

10.1.3. Lista en una caja de dialogo modal

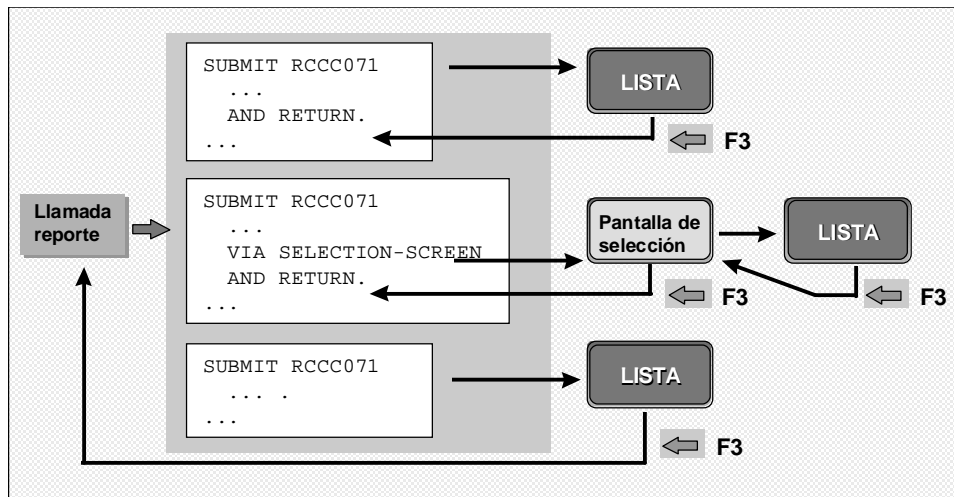


- Si quieres desplegar una lista en un cuadro de diálogo en una pantalla, debes llamar una pantalla, pero puedes eliminar el dialogo con la instrucción SUPPRESS DIALOG en PBO.
- Para asegurar que regresas a el llamado de la pantalla después de dejar la lista , especifica:
LEAVE TO LIST-PROCESSING AND RETURN TO SCREEN 0.

10.1.4. Llamado a un reporte

- Usas la instrucción **SUBMIT** para llamar y comenzar un reporte.
- Si especificas **VIA SELECTION-SCREEN**, el reporte de pantalla seleccionada es mostrado.
- Si especificas **AND RETURN**, regresas al lugar donde el reporte fue llamado cuando éste ha finalizado.

- Llamando un reporte, puedes usar bases de datos lógicas para el proceso leído.



10.1.5. Declaración "SUBMIT"

```
SUBMIT <> VIA SELECTION SCREEN AND RETURN

WITH <parameter> <relational operador> <value> SIGN <s>
WITH <parameter> BETWEEN VALUE1 AND <value2> SIGN <s>
WITH <parameter> NOT BETWEEN VALUE1 AND <value2> SIGN <s>

WITH <select-option> IN <selection table>
... .
```

```
RANGES  SEL_TAB FOR SBOOK-FLDATE.

SUBMIT  RSCCC071 AND RETURN
WITH    CARRID  EQ  SFLIGHT-CARRID
WITH    CONNID  EQ  SFLIGHT-CONNID
WITH    FLDATE  IN  SEL_TAB.
```

- Si quieres pasar a la pantalla de selección (default) cuando se llama un reporte y pasa los valores, hay un número de opciones diferentes que puedes usar, dependiendo de las opciones de selección que requieres.
- Si especificas WITH, puedes predefinir los parámetros y opciones de selección para el reporte

10.1.6. Llamado a una transacción

- De la misma forma que existían dos métodos para cambiar las pantallas de una transacción, existen dos formas para llamar a otra transacción independiente de la transacción que se está ejecutando.

- Si queremos que cuando finalice la transacción, el sistema ejecute otra utilizaremos:

LEAVE TO TRANSACTION 'cod_transacción'.

- Si en cualquier momento queremos ejecutar otra transacción para posteriormente continuar l ejecución de la primera, utilizaremos:

CALL TRANSACTION 'cod_transacción'.

- En este caso el sistema creará una LUW independiente a la anterior y en el caso de producirse un comando BACK, el sistema retornará para ejecutar la primera transacción.

- Un caso típico es querer ejecutar una transacción pero evitando introducir los parámetros de entrada de esta manualmente (es decir, saltando la primera pantalla). En este caso usaremos la instrucción:

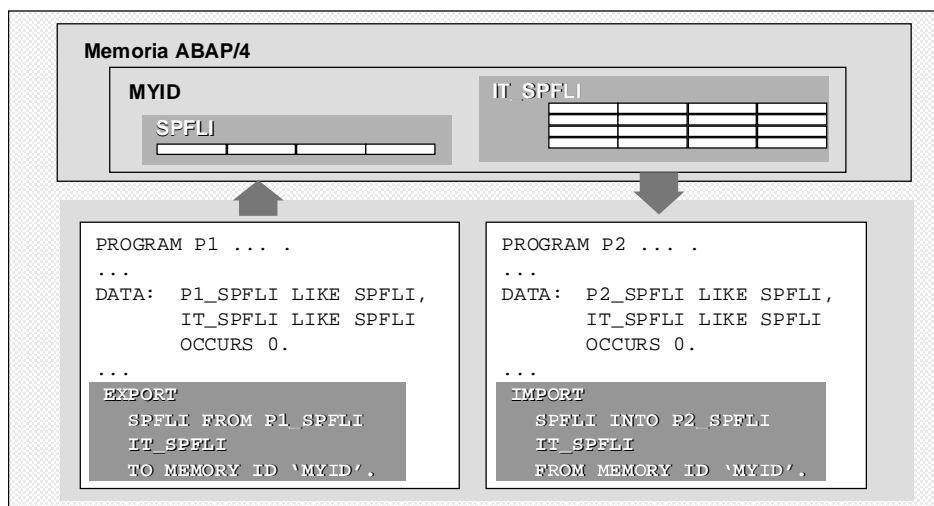
CALL TRANSACTION 'cod_trans' AND SKIP FIRST SCREEN.

- Para lo cual será imprescindible utilizar parámetros almacenado en memoria, ya sea mediante los atributos **SET** (en la primera transacción) y **GET** (en la transacción que llamamos) o codificando explícitamente las instrucciones SET-GET respectivamente:

SET PARAMETER ID 'param' FIELD <campo>.

GET PARAMETER ID 'param' FIELD <campo>.

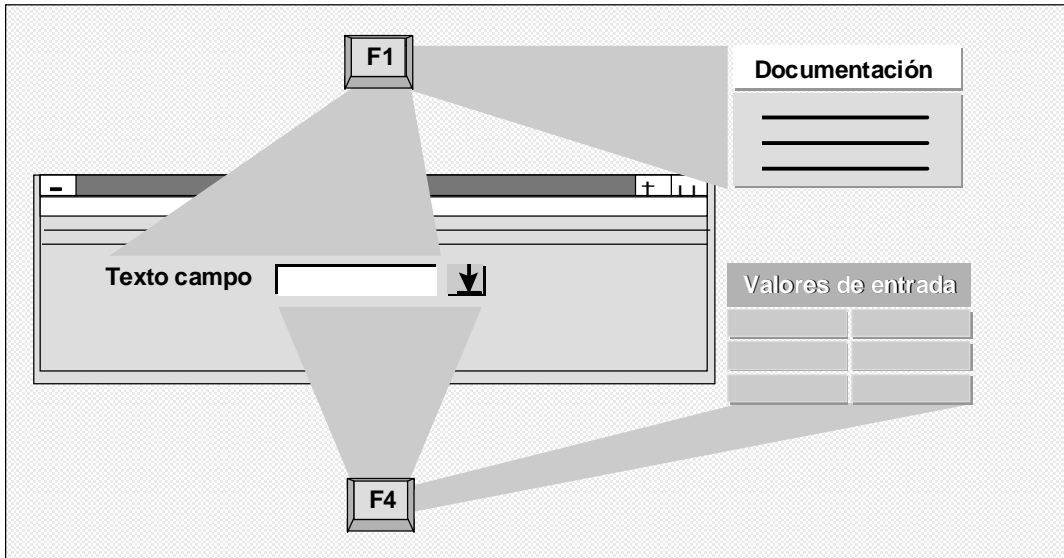
10.1.7. Transferencia de datos entre programas (ABAP/4 Memory)



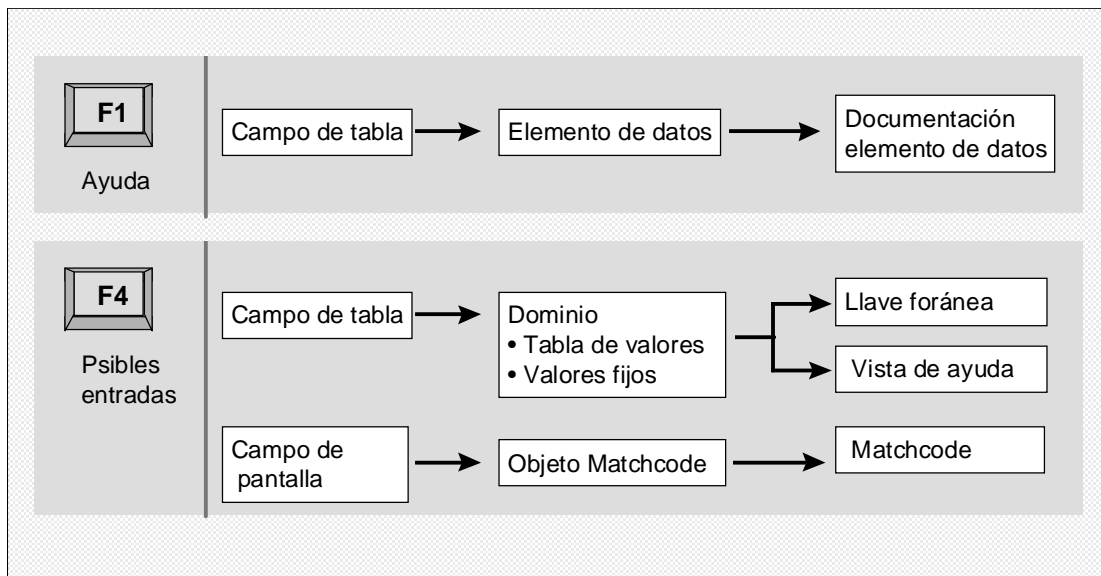
- Para transferir datos entre diferentes programas (reportes o programas de diálogo), usa la memoria SAP y la memoria ABAP/4.
- La memoria SAP es un área memoria específica usada para almacenar valores de campos simples. Es conservada mientras que el usuario este en su sesión. Los contenidos de memoria SAP pueden ser usados como valores dados para los campos de pantalla. Ya que todos **external modi** pueden acceder a la memoria SAP, su conveniencia para transferir datos entre **internal modi** es limitada.
- Puedes usar la memoria ABAP/4 para transferir variables ABAP/4 (campos, cadenas de campos, tablas internas, objetos complejos) entre **internal modi**.
- Cada modo **external** tiene su propia memoria ABAP/4. Cuando el usuario termina un modo external (introduciendo /I en el comando campo), la memoria asociada ABAP/4 es liberada.
- Usa **EXPORT TO MEMORY** para copiar variables ABAP/4 con sus valores actuales (como un dato almacenado) a memoria ABAP/4. Para identificar diferentes datos almacenados, usa la adición ID (máximo 32 caracteres). EXPORT TO MEMORY sobre escribe datos almacenados que ya existen.
- Usa la instrucción **IMPORT FROM MEMORY** para tomar datos de memoria ABAP/4.
- La instrucción **FREE MEMORY ID** te permite liberar un dato almacenado.

11. Funciones Automáticas de ayuda programadas

11.1. Funcionalidad F1 y F4



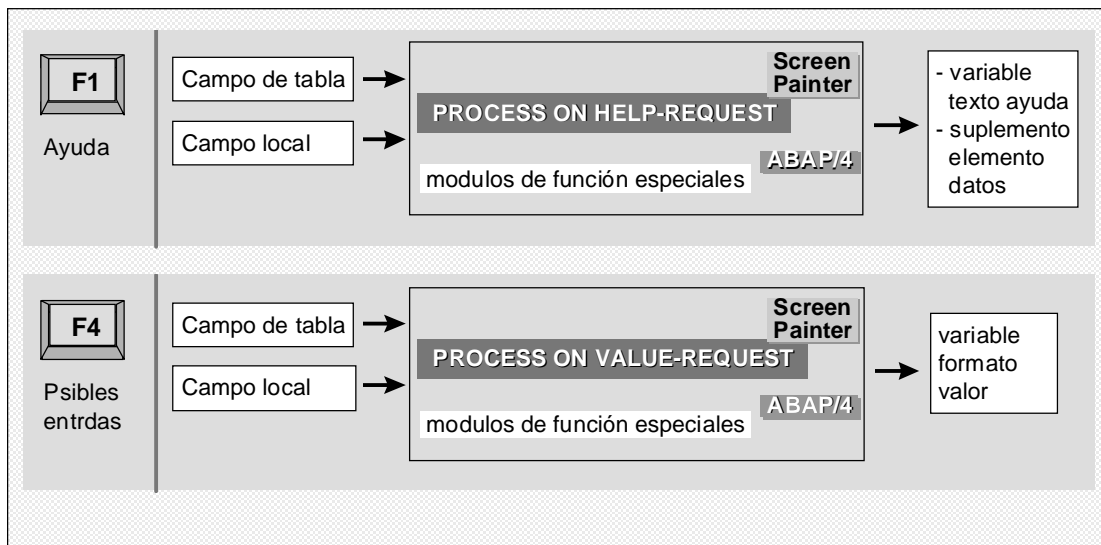
11.1.1. Funciones de Ayuda Automática



- El sistema SAP R/3 provee ayuda al usuario (F1, F4) como funcionalidad estándar. La base de este es la información almacenada en el Diccionario de ABAP/4 el cual sirve como documentación del sistema actual para todas las aplicaciones.

- Ya que el Diccionario de ABAP/4 es integrado activamente dentro del sistema, esta información esta siempre actual y se evita la redundancia.
- Si el usuario llama la función *Help* (presionando F1) estándar, el sistema despliega la documentación del elemento de dato del cual el campo de pantalla se refiere.
- Si el usuario llama la función estándar *Possible entries* (presionando F4), el sistema despliega los valores clave de la tabla check por un campo de diccionario (por ejem. el valor de tabla de la referencia del dominio) a los valores fijos del dominio al cual el campo de pantalla se refiere - si tal referencia existe.
- Si help view existe para la tabla check, estos contenidos son desplegados automáticamente.
- Si hay una relación con un objeto de matchcode (en Screen Painter, un atributo de campo), el matchcode manejado es activado cuando el usuario llamas *Possible entries*.

11.1.2. Programadas



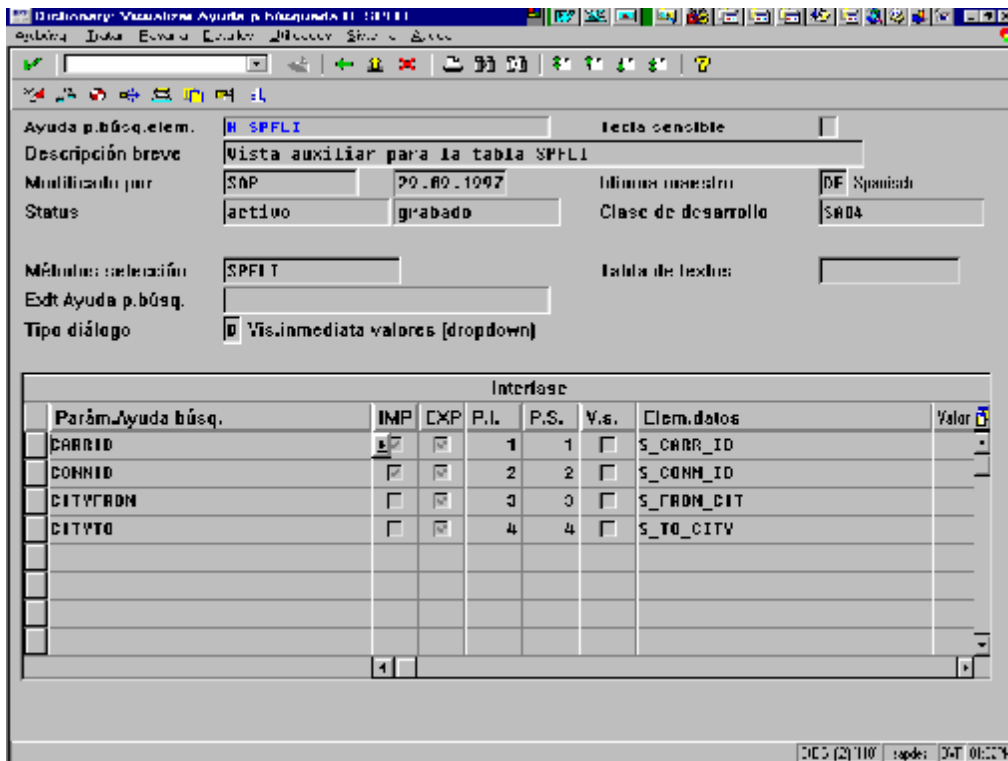
- Si la funcionalidad estándar es insuficiente, puedes modificarlo en el programa de aplicación o programarlo todo tú mismo.
- Para hacer esto puedes usar los eventos **PROCESS ON HELP-REQUEST** y **PROCESS ON VALUE-REQUEST**.

11.1.3. Extensión de la funcionalidad "F4" "HELP VIEW"

- Cuando el usuario presiona F4 para ayuda de valores de entrada, solo los campos claves de la tabla check son mostrados.

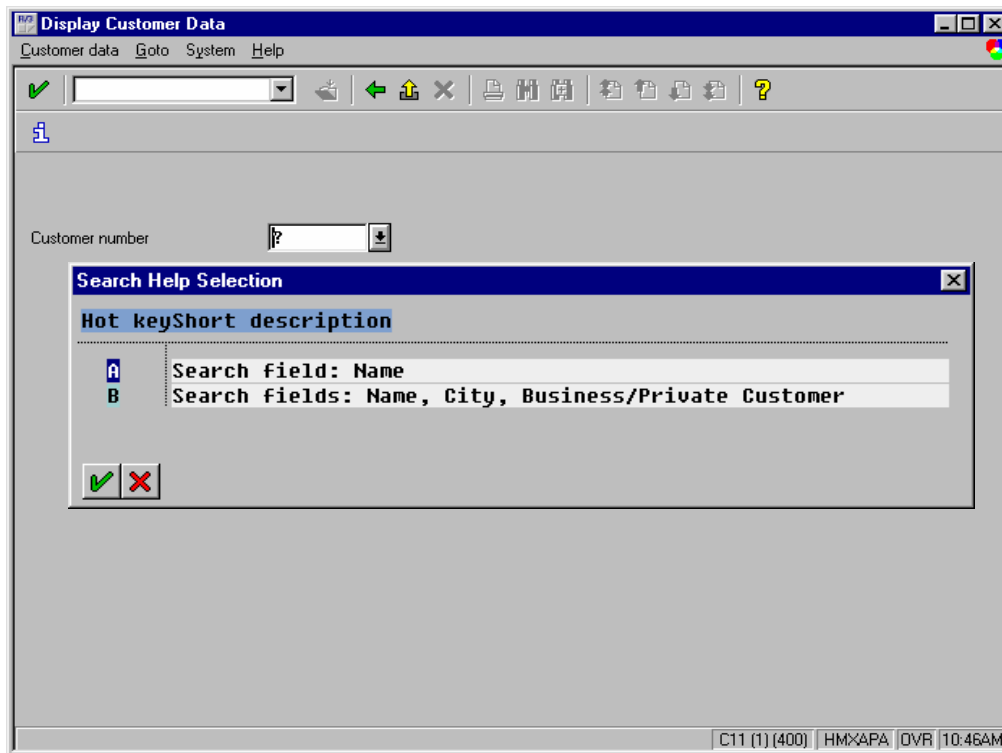
- Si quieres incluir otros campos de la tabla check en el display, debes definir una help view para la tabla check.
- Si la ayuda existe para una tabla, es automáticamente usado en conexión con F4 para el es mostrada.

11.1.4. Creación "HELP VIEW"

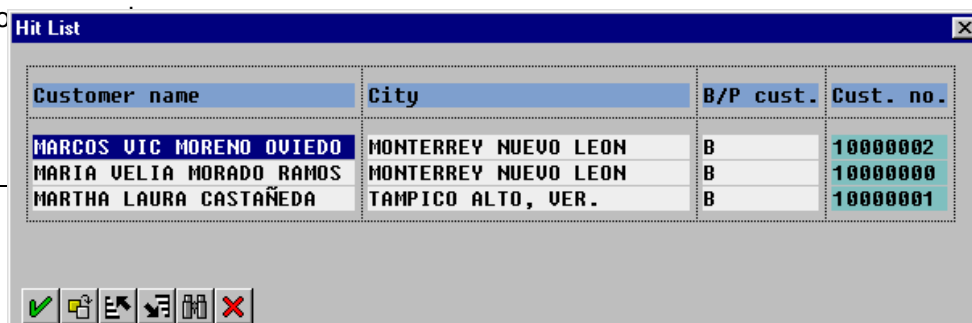


- El mantenimiento de vistas se realiza en el Diccionario de ABAP/4.
- Primero, defines la vista de la tabla y asignas la tabla primaria. Puedes incluir tablas secundarias en la vista, pero esto es posible solo para las tablas las cuales son relacionadas a la tabla primaria mediante la dependencia de una llave foránea.
- Puedes usar cada tabla del diccionario como una tabla primaria por un help view.
- Elige el valor H como tipo.
- En un paso, haces una selección de campo de todas las tablas asignadas (tablas primaria y secundaria).
- Entonces, defines condiciones de selección.

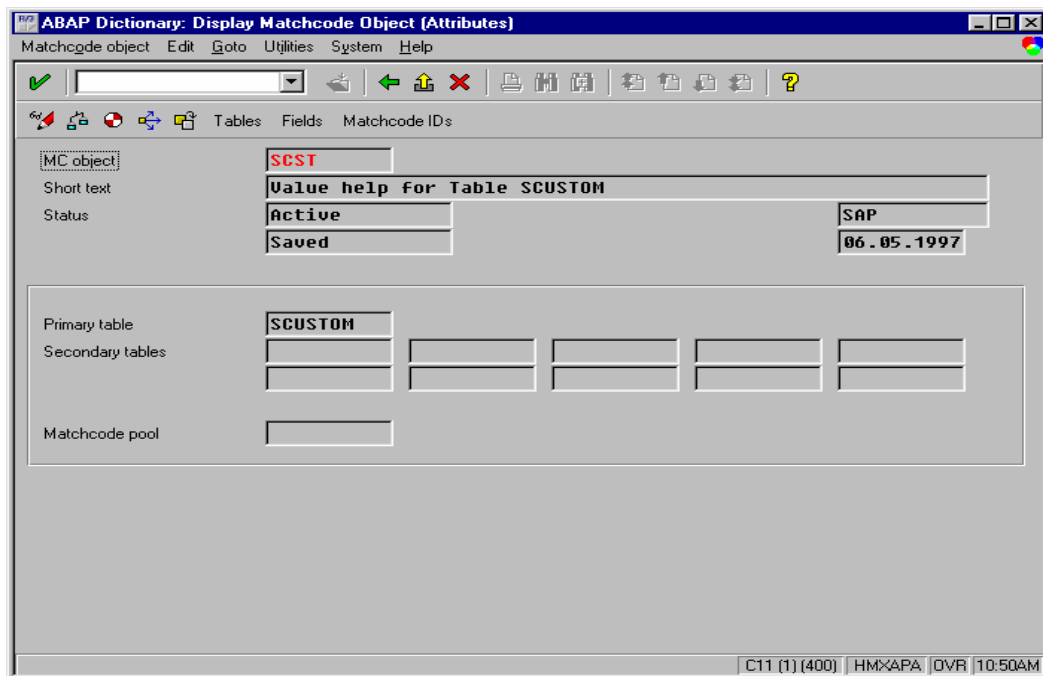
11.1.5. Búsqueda "MATCHCODE"



- El matchcode es un criterio de búsqueda que puedes usar para acceder registros directamente sin conocimientos de la clave (es similar a un índice).
- Matchcode difiere de los índices en que ellos:
 - Pueden ser restringidos por condiciones específicas.
 - Pueden ser formados por varias tablas
 - Son mostrados como ayuda de entrada en el Sistema R/3 (cuando el usuario presiona F4)
- Para un objeto matchcode, estos pueden tomar varios valores (matchcode IDs).
- Si un campos de pantalla incluye una relación a un objeto matchcode (Screen Painter: Atributos de campo), manejando matchcode se activa cuando el usuario presiona a F4 (Posibles entradas), por ejem. Hay varios identificadores de matchcode, son presentados a el usuario para selección.
- Si el usuario elige un identificador de matchcode, el sistema despliega los campos que pertenecen a estos Identificadores para restricción de rango de valor.
- En base de la restricción, el sistema genera una lista resultado de la cual el usuario puede seleccionar

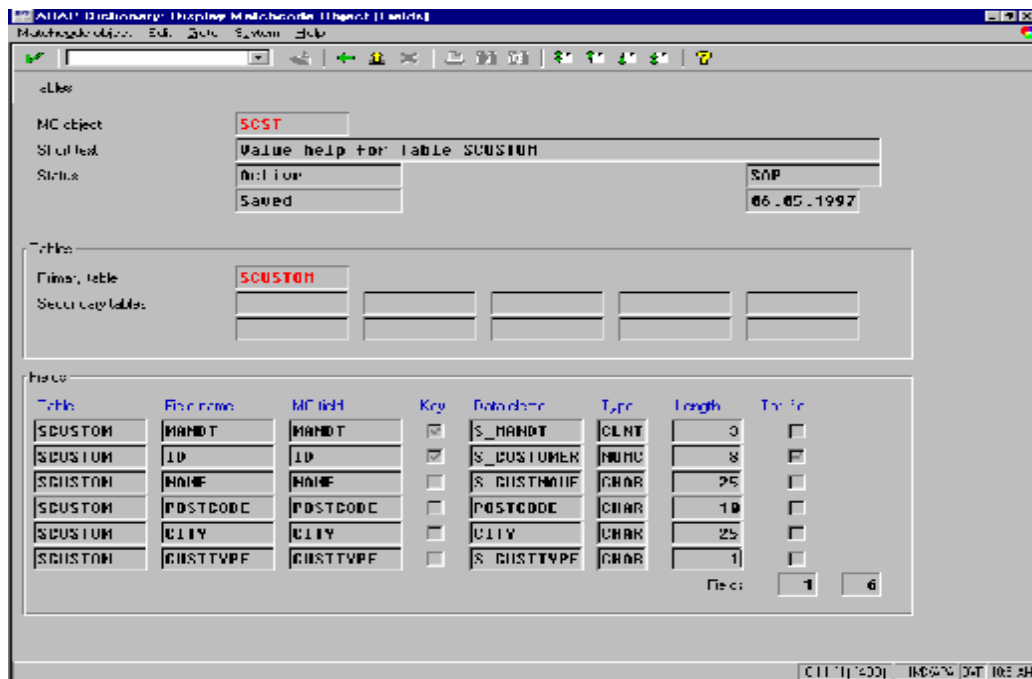


11.1.5.1. Definición "MATCHCODE OBJECT"



- Se da mantenimiento a objetos matchcode en el Diccionario de ABAP/4. El nombre debe ser de 4 caracteres.
- Primero, asignas una tabla primaria. Puedes entonces incluir tablas secundarias en la vista, pero esto es posible solo para esas tablas que están relacionadas a la tabla primaria vía la dependencia de la llave foránea.
- Una lista de tablas para almacenar datos matchcode es necesario solo en casos excepcionales (para ciertos tipos de matchcode lds). Como regla, debes usar views con objetos matchcode. Entonces, una lista de tabla no es requerida en la base de datos.

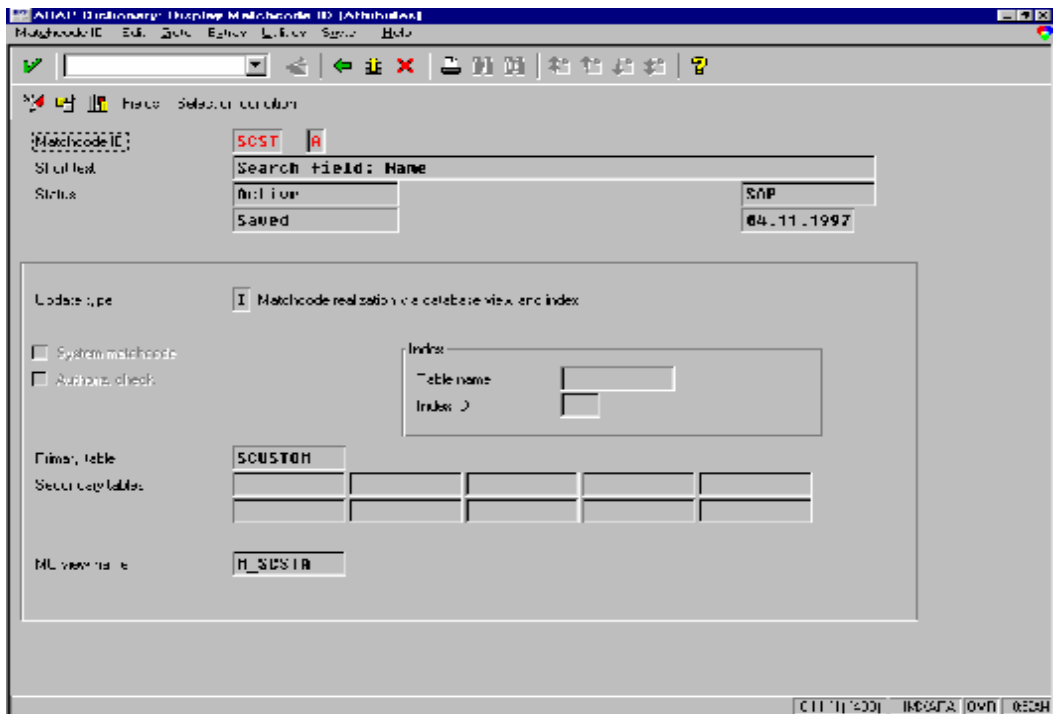
11.1.5.2. Selección de campos



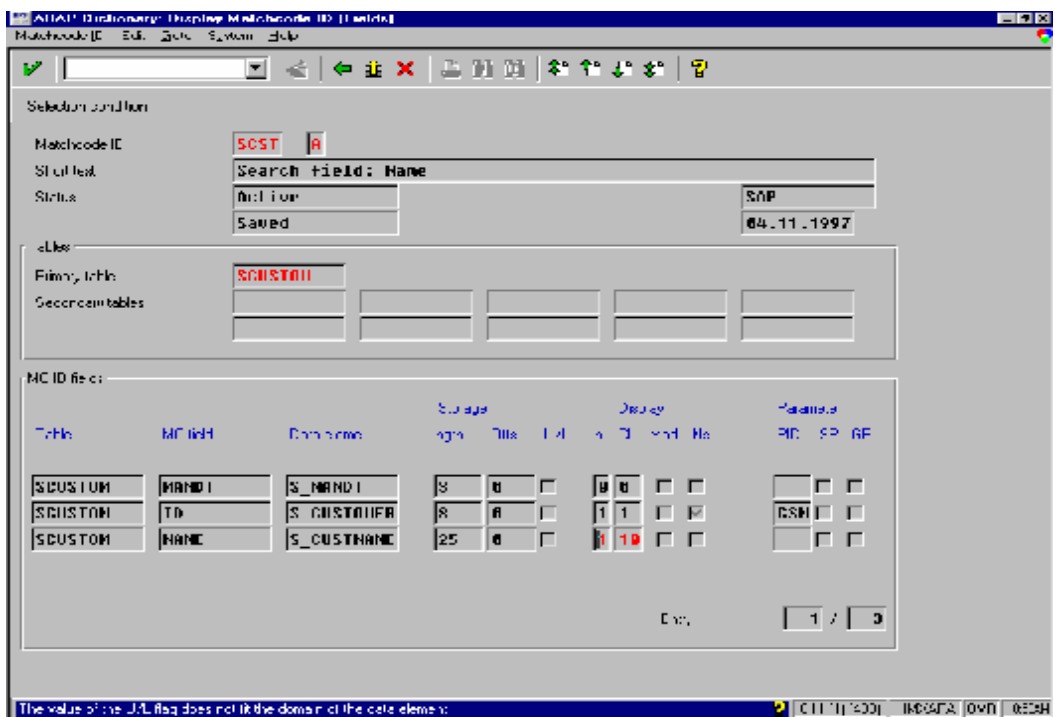
- Cuando has asignado las tablas relevantes para un objeto matchcode, elige los campos. Para hacer esto, defines el nivel de objeto el conjunto básico de campos que se pueden usar definiendo matchcode IDs.
- Debes señalar uno de esos campos como un campo de búsqueda - usualmente, este es el campo clave de una tabla primaria. Después de una búsqueda matchcode, Los contenidos de este campo están colocados en el campo de pantalla asignado en tiempo de ejecución.
- Después de hacer esas definiciones, activas el objeto matchcode.

11.1.5.3. Definición "MATCH CODE ID"

- Cuando defines un matchcode ID para un objeto matchcode, debes especificar un tipo actualizado.
- Si es posible, usa tipo I actual (vistas: datos no redundantes!). En este caso, el sistema genera una vista (de acuerdo a la selección de campo de ID – específico). Esta vista automáticamente es creada en la base de datos cuando el matchcode es activado.

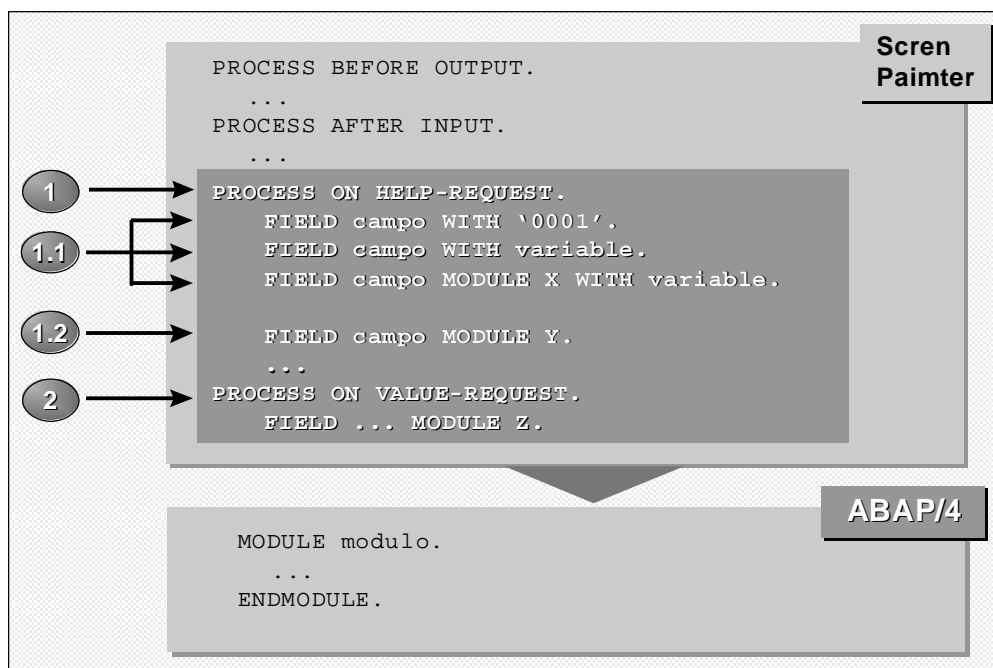


11.1.5.4. Selección de campos



- Ahora puedes elegir campos de nivel matchcode ID. Para hacer esto, defines esos campos que estas usando como criterio de búsqueda para este matchcode ID.
- Aquí, puedes referirte a esos campos que han sido seleccionados con el objeto matchcode. Este normalmente incluiría el campo marcado como nivel de objeto de búsqueda del campo, así que este campo puede ser señalado como no visible y usado como un criterio de búsqueda.
- Para campos que tienen un parámetro ID asignado a ellos, puedes activar el parámetro SET/GET funcionando.
- En suma, puedes definir condiciones de selección de campos específicos para matchcode IDs. Esto significa que solo las entradas de las tablas de base de datos relevantes que actualmente coinciden por el matchcode ID son esos donde los contenidos de campos corresponden a las condiciones de selección.

11.1.6. Programación de la funcionalidad de "F1" y "F4"



- Los eventos **PROCESS ON HELP-REQUEST (POH)** y **PROCESS ON VALUE-REQUEST (POV)** te permiten la programación de la funcionalidad F1 y F4.
- Cuando el usuario presiona F1 en un campo, POH es ejecutado. Similarmente, presionando F4 en un comienzo de campo POV. Significa que debes asignar procesamiento POH y POV a el campo de pantalla con la instrucción FIELD.
- **PROCESS ON HELP-REQUEST (1):**
 - Mostrando suplementos de cada elemento(1.1):

Hay varias formas en que puede direccionar el elemento de dato para suplementar la documentación (ver la información de PROCESS en la pantalla de flujo lógico). Puedes definir suplementos a la documentación de elemento de datos también en el diccionario de ABAP/4 o en la lista de campo del Screen Painter.

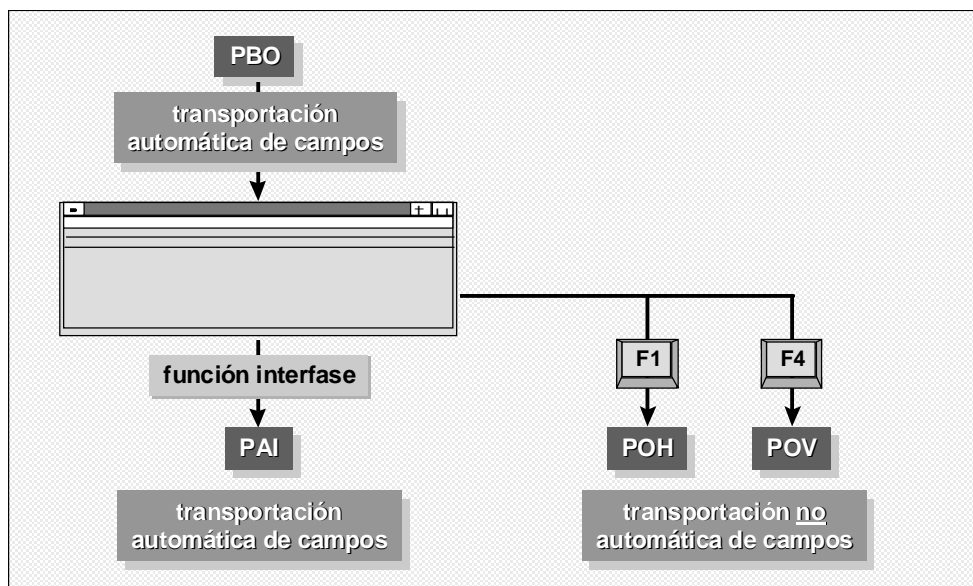
- Mostrando la variable de texto de help (1.2):

En el ABAP/4 module pool, puedes llamar módulos de funciones que muestren la documentación de elementos de datos.

- **PROCESS ON VALUE-REQUEST (2) :**

Hay un número de módulos de funciones que te permiten asignar y formatear valores de entradas de campos (restricción genérica de tablas check, valor mostrado usando una tabla interna, variable matchcode, etc.).

11.1.7. Flujo de los eventos "POH" y "POV"



- Cuando el usuario presiona F1 ó F4, el POH o POV también es procesado. No hay campo transportado automáticamente de una pantalla a un módulo pool.
- Cuando el procesamiento POH o POV ha terminado, el sistema vuelve a mostrar la pantalla pero no ejecuta PBO o a algún campo transportado automáticamente. Esto significa que un valor seleccionado durante el proceso de POV debe ser transportado a un campo de pantalla relevante de otra manera.
- En el procesamiento POH y POV, varios módulos de funciones están disponibles para implementar los pasos necesarios en un diálogo F1 o F4.

11.1.8. Módulos de funciones

- Los módulos de funciones que puedes usar cuando el procesamiento programado F1/F4 ejecuta las siguientes tareas:
 1. Transporta datos entre campos de pantallas y el programa ABAP/4 actual.
 2. Muestra texto de ayuda con la funcionalidad estándar F1.
 3. Muestra y selecciona valores de entrada con la funcionalidad F4.
- Los módulos de función individuales ejecutan un amplio rango de diferentes funciones. Mientras, por ejemplo, hay módulos de funciones especiales para transportar datos entre la pantalla y el programa ABAP/4, otros ofrecen este y funcionalidad adicional en forma integrada.
- El módulo de función lista en el cuadro que son liberados para clientes.

11.1.9. Transportación de campos desde y hacia la pantalla

```
CALL FUNCTION 'DYNP_VALUES_READ'
  EXPORTING
    DYNNAME      = nombre programa
    DYNUMB       = numero pantalla
  TABLES
    DYNPFIELDS  = campos de la pantalla a leer
  EXCEPTIONS
    ....
```

```
CALL FUNCTION 'DYNP_VALUES_UPDATE'
  EXPORTING
    DYNNAME      = nombre programa
    DYNUMB       = numero pantalla
  TABLES
    DYNPFIELDS  = campos de la pantalla a regresar
  EXCEPTIONS
    ....
```

- Si quieres ejecutar operaciones de transporte de campos con procesamiento POH / POV, puedes usar el módulo de función **DYNP_VALUES_READ** para transportar campos de la pantalla con POH y POV, y **DYNP_VALUES_UPDATE** para transportar cambios de la pantalla con POV.
- Los parámetros DYNNAME y DYNUMB te permiten determinar la pantalla que quieres leer, o a la que quieres escribir valores.
- Para la tabla de parámetro DYNPFIELDS, necesitas una tabla interna con estructura DYNPREAD del diccionario de ABAP/4. En esta tabla, especificas los nombres de los campos de pantalla relevante.

11.1.10. Determinación del contenido de los campos de pantalla (POV, POH)

- Para el módulo de función DYNP_VALUES_READ, listas los nombres de los campos de pantalla que requieres para procesar POH/POV a nivel ABAP/4 en la primer columna de tú tabla interna DYNPFIELDS (con pasos de ciclos de campos o campos de tabla de control, también pasas el ciclo índice en la segunda columna).
- Después de un procesamiento con el módulo de función, los valores de campo de pantalla aparecerán en la tercer columna de la tabla.

11.1.11. Transportación de valores a los campos de pantalla (POV)

- Por cada módulo de función DYNP_VALUES_UPDATE, listas los nombres de los campos de pantalla que quieres transportar a los campos de pantalla relevantes en la primera columna de tú tabla interna DYNPFIELDS (con pasos de campos de ciclo o campos de tabla de control, también das un ciclo índice en la segunda columna). En la tercer columna introduces los valores.
- Después de procesar con el módulo de función, los valores son transportados a los campos de pantalla.
- Puedes usar solo el módulo de función DYNP_VALUES_UPDATE con POV.

11.1.12. Visualización de los valores de entrada para el campo de la tabla (POV)

CALL	FUNCTION	HELP_VALUES_GET_EXTEND	
EXPORTING	DYNAME	=	Program name
	DYNUM	=	Screen number
	FIELDNAME	=	Field name
	INPUT_VALUE	=	Input value to restric selection condition
	TABNAME	=	Name of a table
	SHRINK	=	Value restriction yes/no
	...		
Importing	RETURNCODE	=	Return code: 0 = OK 4 = Nothing found
	...		
TABLES	GETFIELDS	=	Screen field names to be used for value restriction
	UPDFIELDS	=	Screen field names and values
for			assigning values to screen fields.

- El módulo de función HELP_VALUES_GET_EXTEND te permite desplegar la tabla por un campo de pantalla particular - con la opción de usar valores en otros campos de pantalla para restringir la selección. Esto también transporta los valores (s) seleccionados por el usuario a el campo de pantalla.

- Pasas el campo de pantalla que has definido por referencia de un campo de tabla del Diccionario a el parámetro FIELDNAME; y el nombre de la tabla del campo de pantalla a el parámetro TABNAME.
- Con el identificador mostrado, especificas si un valor mostrado puede ser seleccionado (DISPLAY = SPACE, por default) o no (DISPLAY = 'X').
- El parámetro SHRINK te permite especificar si el rango de valor de la tabla check es restringido.
- Usas el parámetro TABLES GETFIELDS para especificar los nombres de esos campos de pantalla whose valores actuales have to be taken into account on data selection (por ejem. este contenido de campos de pantalla son leídos por el programa de pantalla para ejecutar un transporte de campo de la pantalla).
- Usa TABLES-parámetro UPDFIELDS para especificar los nombres de esos campos de pantalla que son llenados con valores a un transporte de campo (por ejem. estos campos de pantalla son llenados con valores - transporte de campo a una pantalla).
- La tabla de parámetros GETFIELDS y UPDFIELDS ambos necesitan una tabla interna con la estructura DYNPFLD.

ANEXO 1

ABAP EDITOR

Se pueden especificar los comandos del editor en cualquiera de las formas siguientes:

- Como comandos de cabecera (en la línea de comandos sobre las líneas).
- Comandos de línea sobrescribiendo los números de línea.
- Pulsando las teclas de función o seleccionando las opciones de menú.

COMMANDOS DE CABECERA.

A(TTACH) n	Visualiza el texto desde la línea n.
B(OTTOM)	Ir al final.
T(OPE)	Ir al principio.
+	Siguiente página.
-	Página Anterior.
FIND c	Si la cadena contiene blancos o caracteres especiales, se deberán acotar entre caracteres especiales no contenidos en la cadena a localizar. Ejemplo: FIND / - /
	El comando no distingue entre mayúsculas y minúsculas.
N(EXT)	Busca y se desplaza a la siguiente ocurrencia de la cadena solicitada desde la posición actual del cursor.
R(EPLACE) c1 c2	Reemplaza la cadena c1 por la cadena c2 en todo el texto. 'c1' y 'c2' pueden tener distintas longitudes. Si una de las cadenas contiene blancos o caracteres especiales se deberán acotar, ambas entre caracteres especiales, ver lo indicado en FIND. Ejemplo: R /empty-/ blanks/
F(ETCH) prog	Realiza la edición del programa indicado abandonando el programa actual.
	Guarda el contenido del editor en un almacenamiento intermedio.
S(AVE)	El comando UPDATE borra cualquier texto de almacenamiento intermedio. Si se produce una caída de sistema, normalmente el texto es recuperado del almacenamiento intermedio.
RES(TORE)	Restaura el texto desde el almacenamiento intermedio, sobrescribiendo el existente.
RES(TORE) AKTIV	Restaura la versión activa en el DLIB.
SAVEAS prog	Salva el programa con otro nombre.
U(PDATE)	Salva el contenido del editor.
CHECK	Checa la sintaxis del programa.
PCF(ETCH)	Carga un fichero contenido en el PC.
PC(DOWN)	Escribe el contenido del editor en un fichero de PC.
HELP word	Visualiza la ayuda sobre la palabra indicada.
I(NSERT) n	Inserta n líneas al final del texto.

IC word	Inserta la estructura de la sentencia indicada, esto es valido para las sentencias: CASE, DO, FOR, IF, LOOP, MESSAGE, MODULE, SELECT, SHIFT, SORT, TRANSFER, WHILE y WINDOW.
IC FUNCTION func	Inserta la estructura de un CALL FUNCTION para la función indicada.
IC SELECT tab	Inserta la estructura del SELECT para la tabla indicada.
IC...	Inserta en la posición del cursor... <ul style="list-style-type: none"> *f - FORM bloque de comentario. *m - MODULE bloque de comentario. *. * - Línea de comentario *....text.....* *- * - Líne de comentario *_____* *- *1 - Area de comentario con línea en blanco. ** - Línea de comentario *****. **n - Area de comentario con n líneas en blanco (1 <= n <= 5)
PP	Pretty Print del programa.
PRINT	Imprime el contenido del editor.
RENUM(VER)	Renumera líneas.
SHOW tab	Visualiza los campos de la tabla indicada.
SHOW FUNCTION func	Visualiza el módulo de la función indicada.

COMANDOS DE LINEA:

*	Considera la línea como primera línea en la pantalla.
T+	Ir a la primera línea
B-	Ir a la última línea
>	Inserta las líneas de programa del include.
<	Elimina las líneas de código del include y restaura éste.
u	Escribe el bloque incluido en el fichero del INCLUDE e inserta la sentencia include correspondiente.
A	Línea de destino de una operación de copia o movimiento, el texto seleccionado se incluirá en la línea posterior.
B	Línea de destino de una operación de copia o movimiento, el texto seleccionado se incluirá en la línea anterior.
O	Overlay el contenido de C o M sobre la línea indicada.
C	Copia esta línea.
CC...CC	Copia el bloque de líneas.
M	Mueve la línea indicada.
MM...MM	Mueve las líneas indicadas.
I	Inserta una nueva línea.
In	Inserta n líneas.
N	Inserta un área de comentario.
D	Borra la línea.
DD...DD	Borra el bloque de líneas.
R	Repite la línea.
Rn	Repite la línea n veces.
RR...RR	Repite el bloque de líneas.
J	Junta las líneas actual y siguiente.
S	Parte la línea a la posición del cursor.
SH...SH	Desplaza el bloque de líneas a la posición del cursor.
WW...W W	Marca el bloque de líneas en el archivo intermedio general.

W Copia el contenido del archivo intermedio general.
XX...XX Copia el bloque indicado en el archivo intermedio X.
X Copia el archivo intermedio X en la línea siguiente.
YY...YY Ver XX.
Y Ver Y
ZZ...ZZ Ver XX.
Z Ver X.
CLEAR Borra los buffers X, Y, Z.
PR...PR Imprime el bloque de líneas.