## MENU PRINCIPAL


Fig. Menú principal.

Este es el menú principal de SAP, desde aquí puede acceder a cualquier sitio de SAP.

Para el programador el menú más importante será el de “Herramientas”, dentro de “herramientas” tenemos los siguientes submenús:

“Workbench ABAP/4” -> Desde aquí podemos ir las herramientas de programación de SAP.

“Gestión” -> Esta opción solo es apta para administradores de SAP, desde aquí puede controlar todos los parámetros de SAP. Digo que es solo para administradores porque solo ellos tienen autorización para poder acceder a las opciones que tiene. La pantalla principal sería la siguiente:

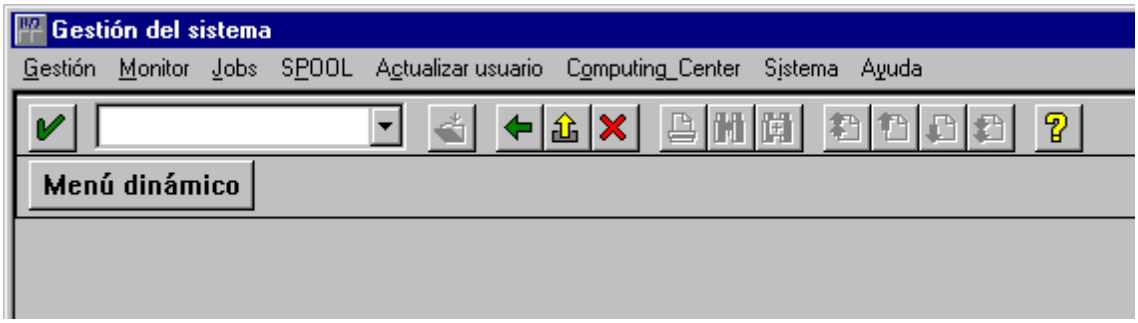


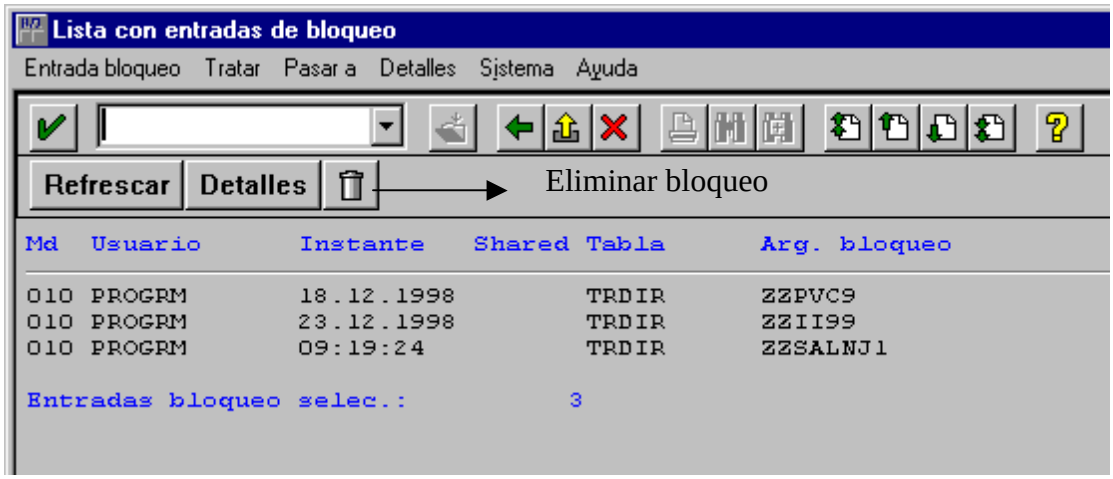

Fig. Gestión.

Hay veces que un programa se queda bloqueado, es decir, cuando intentamos modificar un programa y nos sale el siguiente mensaje: El usuario PROGRM ya esta tratando ZZII5, donde PROGRM es el nombre del usuario y ZZII5 es el nombre del programa al que intentamos acceder.

Para poderlo desbloquear hay dos formas:
La primera es ir al menú “Monitor”, “Entradas bloqueo” y nos saldrá la siguiente pantalla:

A

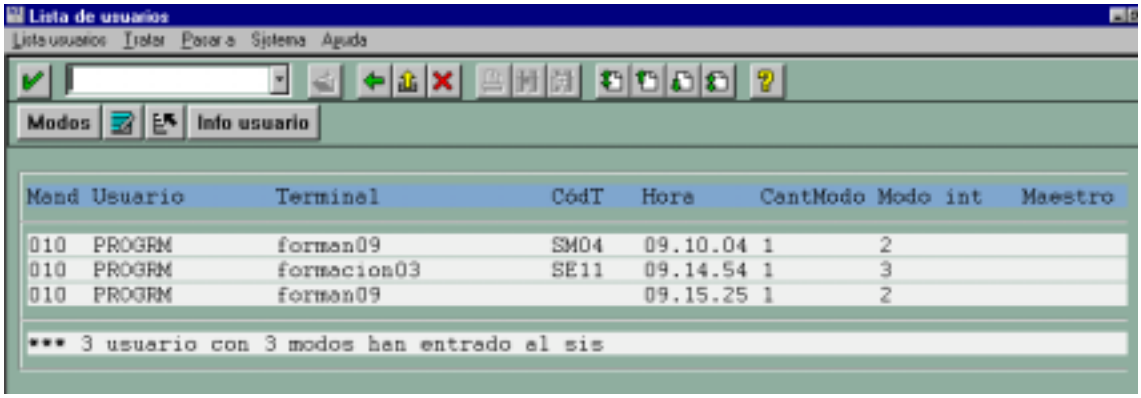
Para saber qué tabla y qué argumento de bloqueo hemos de poner hemos de pulsar el botón que pone “lista” y nos saldrá la siguiente lista de programas bloqueados:




Como vemos, sale la lista de programas bloqueados. Para poder desbloquearlos se pulsa sobre el programa que deseamos desbloquear y pulsamos a la papelera, si no, vamos al menú “Entrada bloqueo”, “Borrar SHIFT+F2” y se quita el bloqueo (siempre y cuando tengamos la autorización pertinente).

Para borrar todas las entradas de bloqueo que tengamos vamos al mismo menú de antes (“Entrada bloqueo”) y pulsamos sobre “borrar todos” y se quitarán las entradas de bloqueo, siempre que tengamos autorización.

También hay una forma más fácil de hacerlo sin necesidad de ser el administrador, esto se hace tomando la sesión. Para hacerlo, desde la pantalla de gestión (Véase Fig. Gestión) vamos al menú “Monitor”, “Supervisar sistema”, “usuarios (todos)”. Entonces nos saldrán los usuarios conectados y los que están bloqueados. La pantalla que sale es la siguiente (en la página siguiente):




En esta imagen solo hay dos usuarios conectados (formación03 y el último forman09), el forman09 que está al principio es el que está bloqueado.

Sé que está bloqueado porque el ordenador donde trabajo es el forman09 y he entrado a las 9:15, por lo tanto el anterior a mí, es decir, el que ha entrado a las 9:10 está bloqueado. El bloqueo lo he hecho expresamente apagando el ordenador cuando estaba conectado.

Para desbloquearlo hacemos clic sobre el usuario que queremos desbloquear, seguidamente vamos al menú “tratar”, “tomar sesión” y entonces tomaremos la sesión bloqueada y podremos utilizar nuevamente los programas.

También desde aquí podremos ver qué usuarios están conectados y también podremos tomar la sesión de otros usuarios, se hace de la misma forma que para desbloquear el programa.

En la pantalla principal de gestión (véase fig. gestión) podemos controlar las colas de impresión y muchas cosas más.

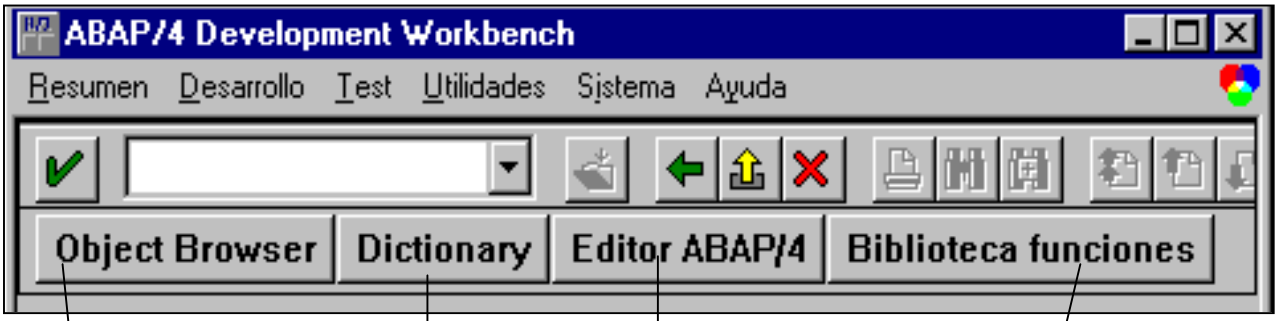
Volviendo al menú de “herramientas” (Véase Fig. Menú principal) tenemos más submenús de los cuales no puedo decir gran cosa ya que desconozco su funcionamiento.

## MENU SISTEMA

El menú “Sistema” siempre aparece, estemos donde estemos y hagamos lo que hagamos. Dentro del menú aparece una serie de opciones que tampoco varían mucho y alguna son de bastante utilidad. Explicaré de una forma rápida algunas de la opciones más importantes, alguna de ellas se explicarán más ampliamente en este manual.

- Para ver los valores fijos de cada usuario, o sea, impresora, idioma, etc.. que tiene por defecto. Tenemos que ir al menú “Sistema”, “Valores prefijados” y “Val.fijos usuario”.
- Para información referente a nuestro usuario como puede ser telefonos, direcciones, etc.. tenemos que ir al menú “Sistema”, “Valores prefijados” y “Dirección usuario”.

## ABAP/4 DEVELOPMENT WORKBENCH



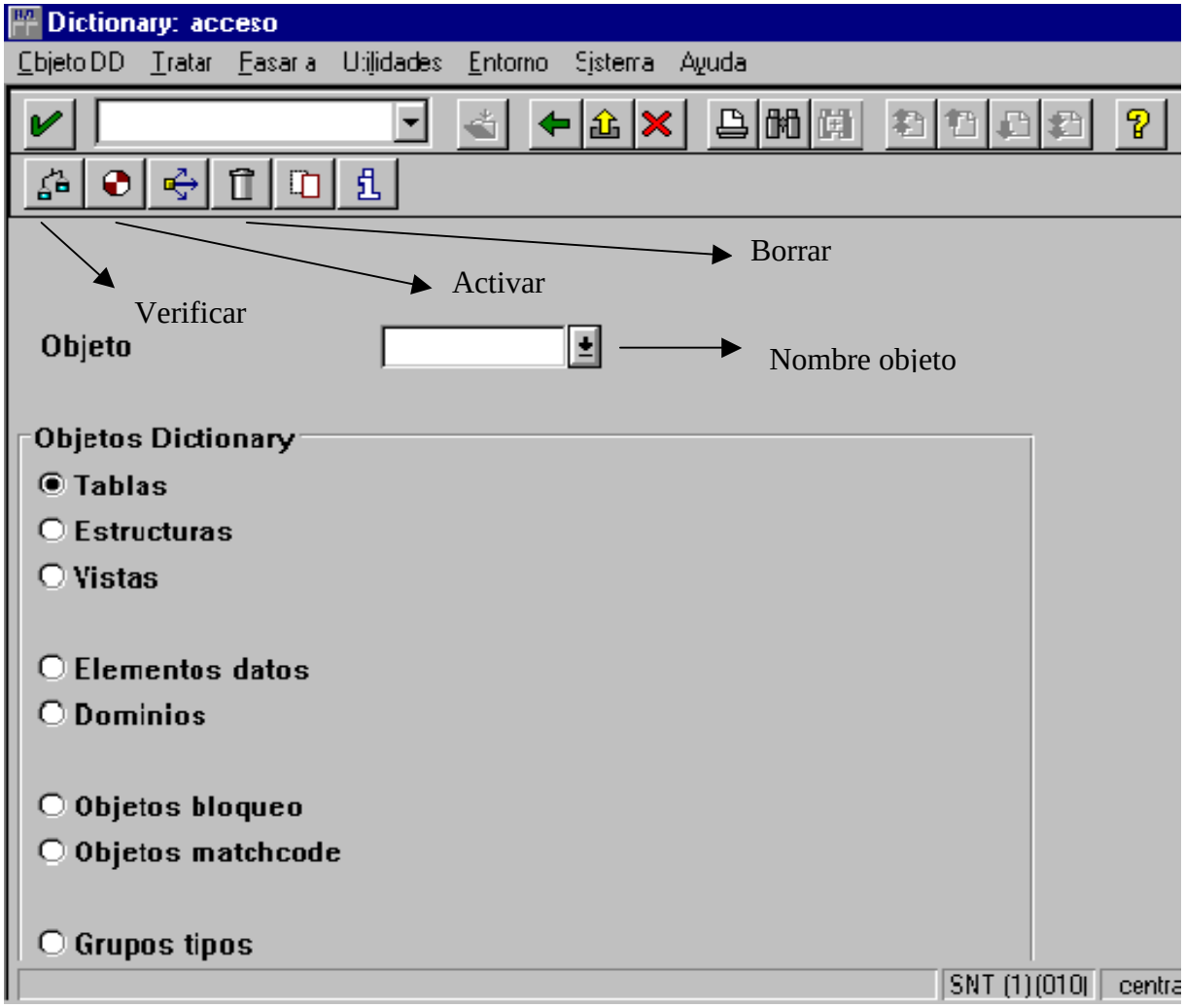

Editor de programas

Aquí están todas las tablas de SAP, es decir, es donde se almacenan todos los datos con que SAP trabaja

Desde aquí podemos crear, borrar, modificar o visualizar todos los objetos que se pueden crear en SAP.

Aquí tenemos la posibilidad de crear, borrar, modificar o visualizar todas las funciones del sistema

## PANTALLA DEL DICCIONARIO



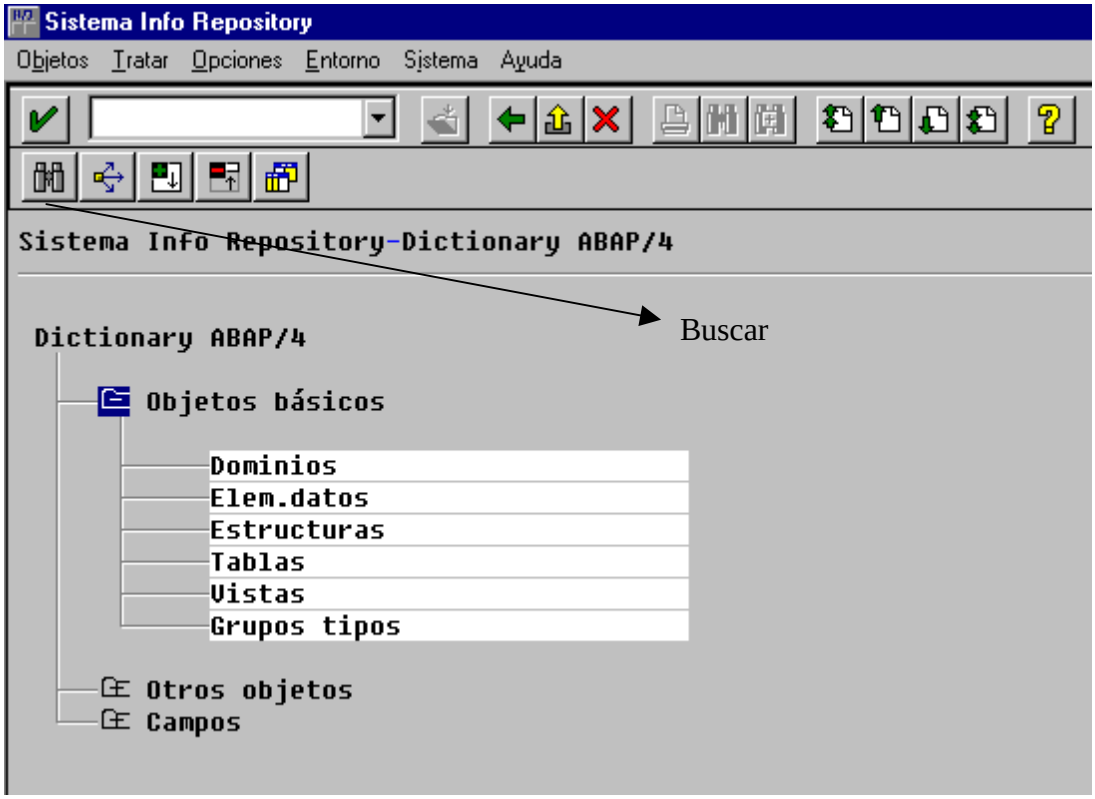

En "objeto" introduciremos el nombre que deseamos crear, borrar, modificar o visualizar.

Los tipos de objetos son:

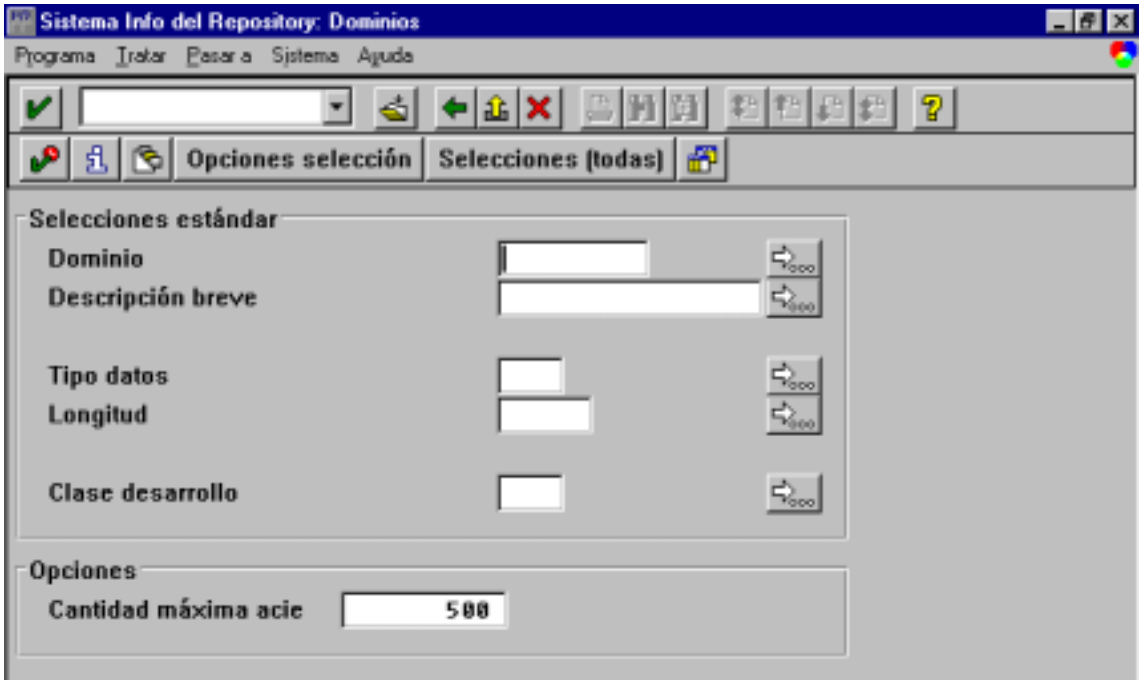
- Tablas
- Estructuras
- Vistas
- Elementos de datos
- Dominios
- Objeto bloqueo
- Objeto matchcode
- Grupos tipo

Después hay tres botones que son: "Visualizar" (icono de gafas), "Modificar" (icono de lápiz) y "crear" (icono objeto en blanco).

Para ver todos los objetos del SAP, vamos al menú "Entorno", "Sistema Info". Ahí nos saldrá una pantalla:




Haciendo doble clic en un campo (ej. : Dominio) podremos buscar los dominios que haya en el SAP. Esta es la pantalla de búsqueda:




En las búsquedas se pueden utilizar caracteres comodín, ej: *.
Con el botón de “Opciones de selección” podemos poner condiciones de búsqueda.

Pulsando F8 realizaremos la búsqueda.

## COMO CREAR UNA TABLA

Para crear una tabla primero hemos de crear un dominio, seguidamente un elemento de datos para ese dominio u otro que exista y por último la tabla que tendrán una serie de elementos ya existente o que los hayamos creado antes.

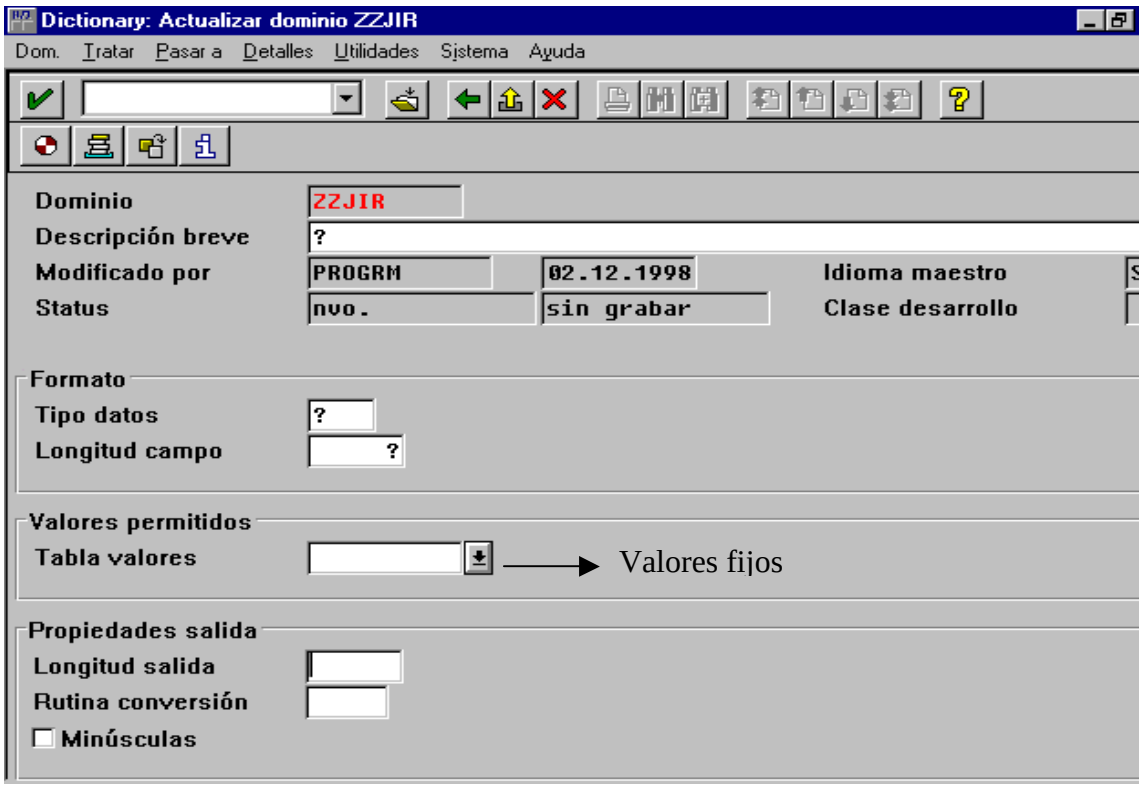
Además de explicar como se realiza un dominio, un elemento de datos y una tabla, también explicaré como las podemos visualizar, modificar y otras operaciones diversas.

## CREAR, MODIFICAR O VISUALIZAR UN DOMINIO

Hemos de ir a la pantalla de DICTIONARY. Después:

Introducir el nombre del Dominio de datos que vamos a crear, visualizar, modificar o buscar.
Pulsar el botón "Objeto Dictionary" llamado "Dominio".
Pulsar el botón que deseamos (está abajo del todo) o ir al menú "Objeto DD" donde se puede escoger lo que vayamos a hacer.

Y nos saldrá la pantalla siguiente:

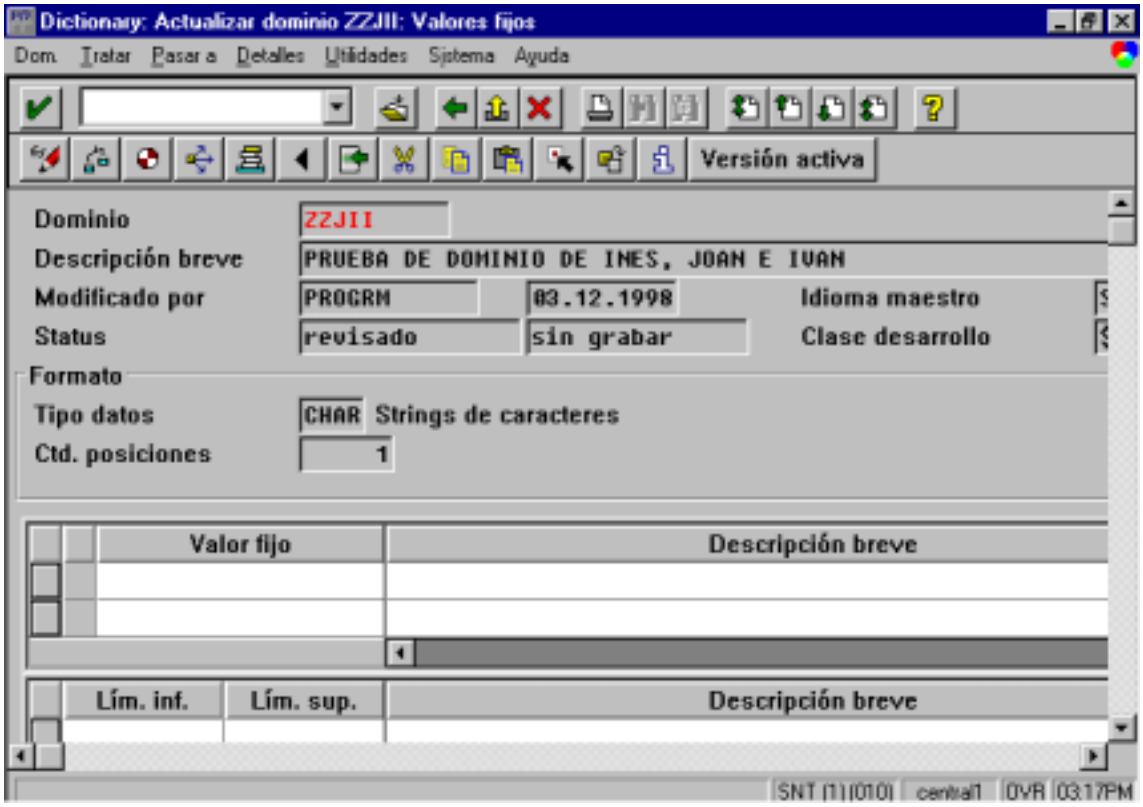



En esta pantalla introduciremos los valores que le daremos al dominio.

Hay que poner una descripción breve de lo que hace o para qué sirve ese dominio.

En "Formato": en "tipo de datos", haciendo doble clic nos saldrán los tipos de datos disponibles. Y "longitud" será la longitud que le demos.

"Valores permitidos": será la tabla de verificación. La tabla de verificación es una tabla en la que hay unos valores. Cada vez que entramos un dato en un campo que tenga una tabla de verificación irá a esa tabla para comprobar su validez.
También podemos introducir límites para un campo.
Para introducir valores fijos vamos al recuadro de "valores permitidos" y pulsamos al botón de "valores fijos" o hacemos doble clic en el campo. Otra forma de llegar es en el menú "Pasar a (o F8)", "Valores fijos". Entonces saldrá esta pantalla:




Como vemos, podemos introducir valores fijos (tantos como queramos) o límites inferior o superior (tantos como queramos).

De vuelta a la pantalla principal de dominios:

En "Propiedades de salida", "Longitud de campo": qué longitud saldrá cuando visualicemos el contenido del campo.

En un dominio también podemos poner valores fijos. Cuando introduzcamos algún valor en un campo y no concuerde con los valores preestablecidos el sistema avisará que el dato introducido no es correcto.

Para Grabar se utiliza este botón:

 → Botón para grabar sin verificar o F11

Cuando grabemos por primera vez nos preguntará por la Clase de desarrollo. Esta clase nos la da la empresa donde estemos y sirve para el transporte a producción. Para practicar se deja en blanco.

Para practicar la grabaremos como Objeto local (Botón que está abajo) nos pondrá como clase de desarrollo '$TMP'.

Una vez creado hemos de verificar la consistencia de la tabla, lo haremos con el botón:

 → Verificar o Ctrl+F2

Después, para poderlo utilizar hay que activarlo, que lo haremos con el siguiente botón:

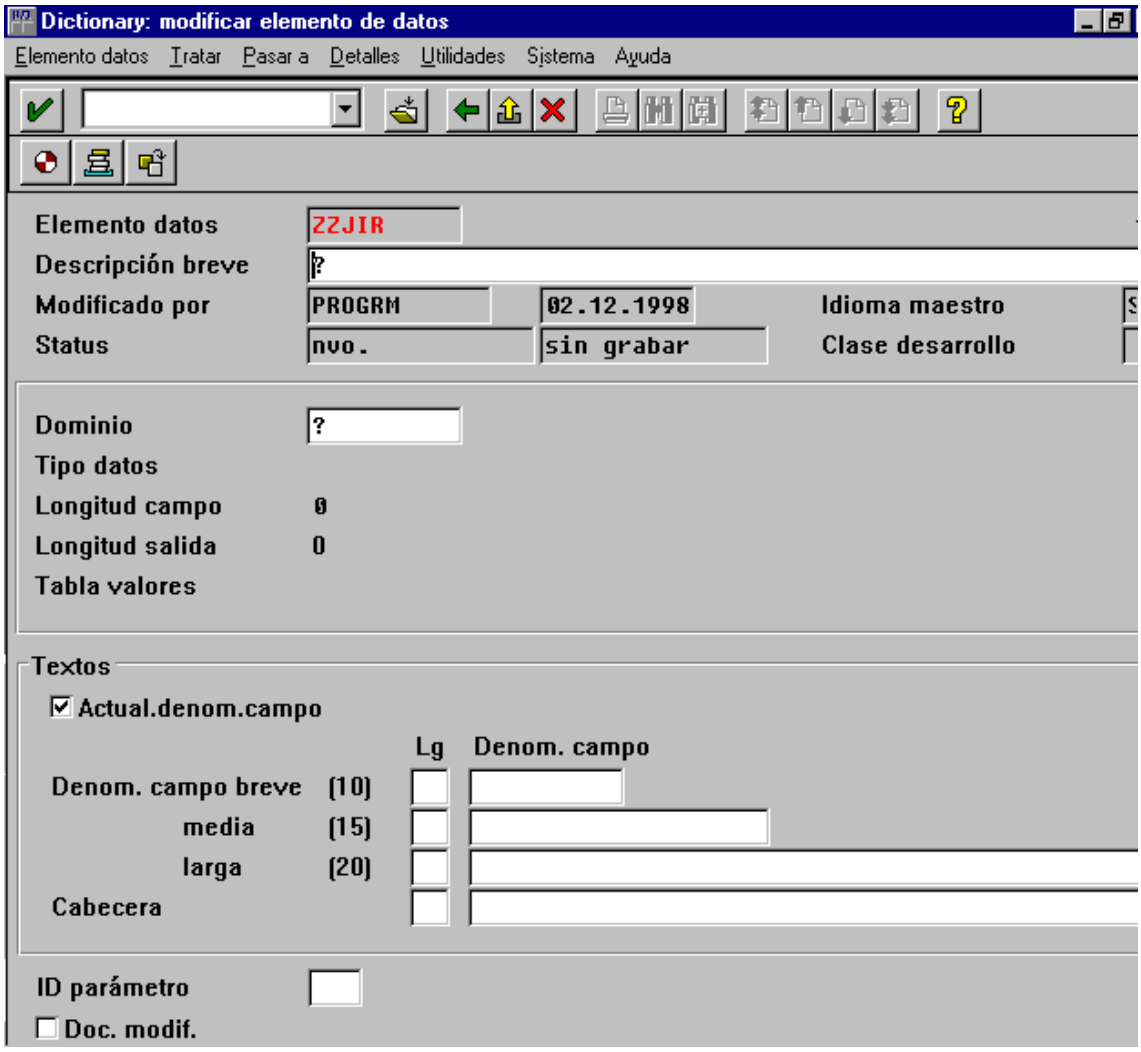
 → Activar CTRL+F3

## CREAR, MODIFICAR O VISUALIZAR UN ELEMENTO DE DATOS

Hemos de ir a la pantalla de DICTIONARY. Después:

- Introducir el nombre del Elemento de datos que vamos a crear, visualizar o modificar.
- Pulsar el botón de Objeto Dictionary llamado “Elemento de datos”.
- Pulsar el botón que deseamos (está abajo del todo) o ir al menú Objeto DD donde se puede escoger lo que vayamos a hacer.

Y nos saldrá la pantalla siguiente:




Como ya hemos dicho los Elementos de datos es información semántica o técnica de un campo. También un elemento puede estar en más de un campo (distinto o igual).

Hay que introducir una descripción para ese Elemento de datos.

En Dominio introduciremos el dominio que tendrá (es un campo obligatorio), si no lo sabemos, ponemos el cursor en el campo y pulsando F4 podremos buscar el dominio que queramos utilizar.

Después pondremos la descripción breve, media o larga del campo

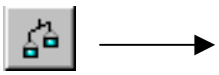
Para Grabar se utiliza este botón:

 Botón para grabar sin verificar o F11

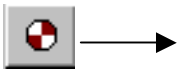
Cuando grabemos por primera vez nos preguntará por la Clase de desarrollo. Esta clase nos la da la empresa donde estemos y sirve para el transporte a producción. Para practicar se deja en blanco.

Para practicar la grabaremos como Objeto local (Botón que esta abajo) y nos pondrá como clase de desarrollo '$TMP'.

Una vez creado hemos de verificar la consistencia de la tabla, lo haremos con el botón:

 Verificar o Ctrl+F2

Después para poderlo utilizar hay que activarlo, que lo haremos con el siguiente botón:

 Activar CTRL+F3

## CREAR, MODIFICAR O VISUALIZAR UNA TABLA

Hemos de ir a la pantalla de DICTIONARY. Después:

- Introducir el nombre de la tabla que vamos a crear, visualizar o modificar.
- Pulsar el botón de "Objeto Dictionary" llamado "Tabla".
- Pulsar el botón de lo que deseamos hacer (está abajo del todo) o ir al menú "Objeto DD" donde se puede escoger lo que vayamos a hacer.

Y nos saldrá la pantalla siguiente:

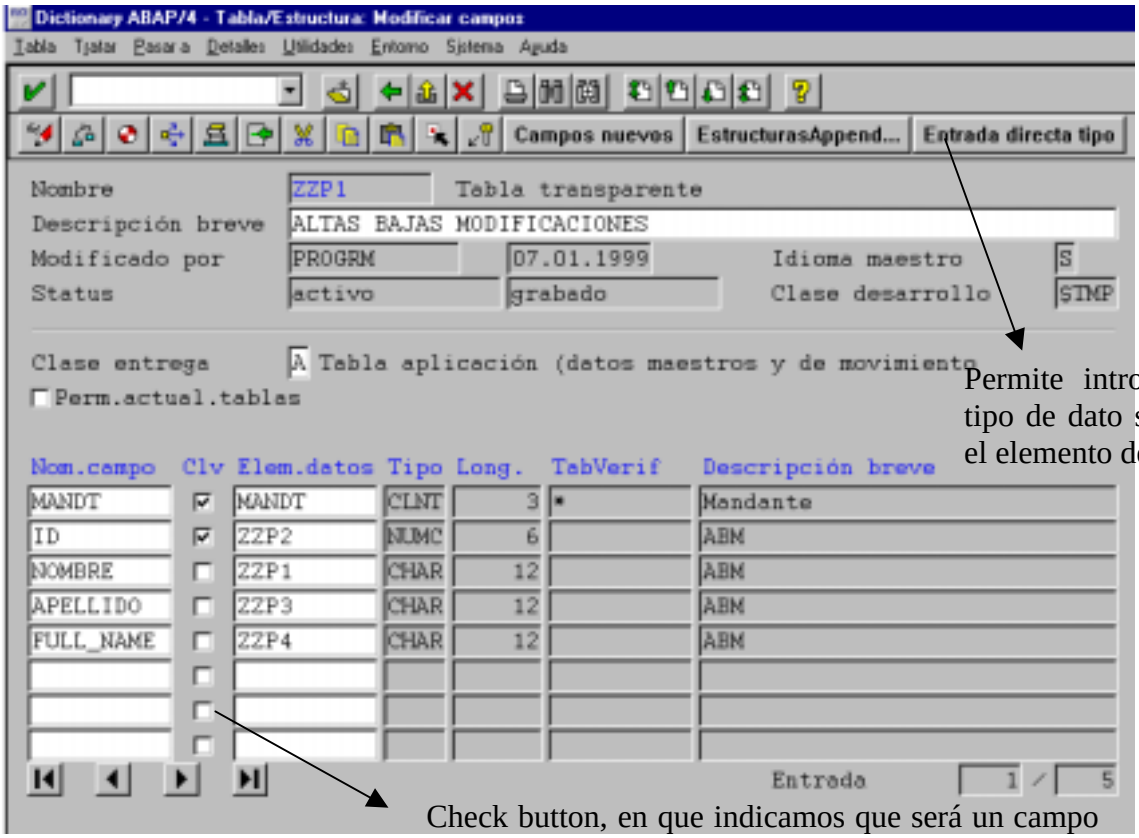
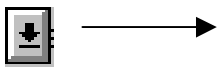



Fig. Tabla.

Los campos que tengan fondo blanco y el carácter '?' son de obligada introducción.

En la clase de entrega haremos clic y le daremos al botón o F4:

 Seleccionar valores de un campo

Y seleccionaremos el tipo de clase de entrega.

También activaremos el Check Box que pone "Perm.actual.tablas" para que se puedan introducir datos y modificar estos.

Debajo de la pantalla, en "campos", introduciremos el nombre del campo. Hay que introducir uno obligatoriamente. Y uno ha de ser el campo clave. Se activa pulsando en el check-button que está debajo de la palabra "Clv".

Después aparecen los Elementos de datos que a su vez tienen Dominios (Estos dominios se asignan automáticamente al crear el elemento de datos).
En “Elementos de datos” podemos introducir un campo que ya sepamos o posicionandonos sobre el campo y pulsando F4 podemos realizar una búsqueda de todos los elementos que haya o una condición de búsqueda. Cuando lo hayamos puesto pulsaremos ENTER para que SAP coga ese elemento de datos. Sí después de eso en “TabVerif.” nos aparece un ‘*’ SAP nos dice que ese campo se puede relacionar con otro campo de una tabla, o sea, una tabla de verificación.

## MANDANTE

En el primer campo se suele poner, por no decir siempre, el campo “Mandt”. Este campo sirve restringir el acceso de un usuario o usuarios a una determinada tabla o tablas. En una empresa puede haber más de una mandante. Para poder declararlo donde pone “nombre de campo” pondremos “Mandt”, en elementos de datos también escribiremos “Mandt”, pulsaremos ENTER para que lo coga y también le diremos que es un campo clave.

## TABLA DE VERIFICACION

La tabla de verificación son tablas en las que hay unos valores. Cada vez que entramos un dato en un campo que tenga una tabla de verificación irá a esa tabla para comprobar su validez. Cuando en el campo que pone “Tab Verif.” sale un ‘*’ indica que puede haber un campo de verificación. Para poder asociar un campo a una tabla de verificación hay que realizar una clave foránea. Más adelante veremos un ejemplo de cómo se asocia a un campo una tabla de verificación y cómo se hace una clave foránea.

## INDICES

Los índices sirven para crear campos claves en nuestra tabla. Lo que se consigue es mejorar la velocidad cuando buscamos por el campo clave de la tabla. Más adelante veremos un ejemplo completo de cómo se hace un pequeño índice.

## CLAVES FORANEAS

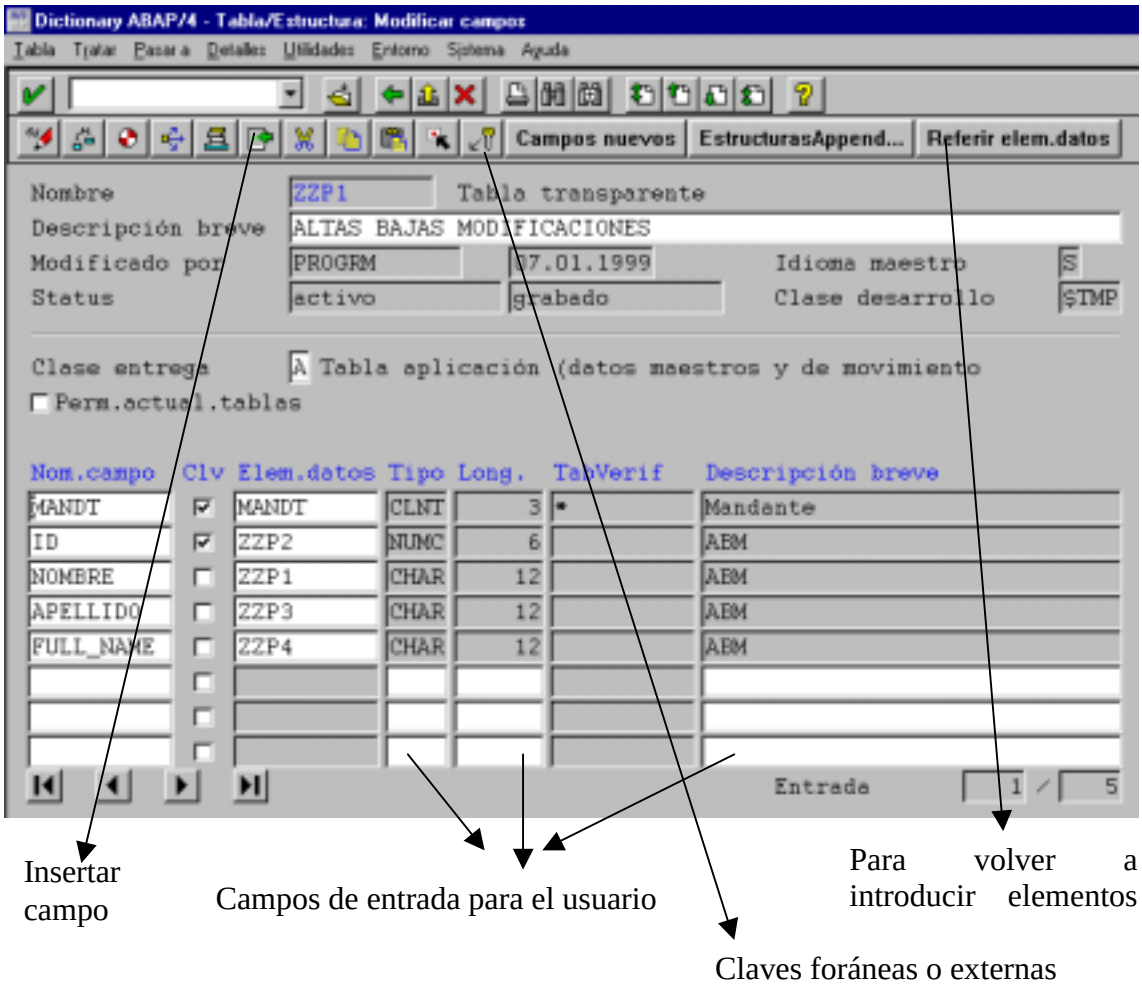
Las claves foráneas nos permiten relacionar una o varias tablas a través de un campo clave. Más adelante veremos un ejemplo completo de cómo crear una clave foránea y una tabla de verificación.

## TABLA SIN ELEMENTOS DE DATOS NI DOMINIOS

En una tabla podremos crear nuestros propios campos sin tener que realizar ni elementos de datos ni dominios. Esto es bueno cuando queremos declarar campos a pelo, es decir, solo poniendo el tipo de campo y su longitud. Para poder activarlo tenemos que ir al menú “Tratar”, “entradas tipo directo”. Para poder quitar esa opción vamos al menú “Tratar”, “Ref. elementos.datos”.

En la imagen de la tabla (Véase  Fig. Tabla.) podemos comprobar que no está activada la opción de entradas de tipo directo

Si pulsamos el botón que pone “Entrada tipo directo” o a través del menú (como ya he explicado antes) nos saldrá la siguiente pantalla:




Como vemos, en este ejemplo hay campos que tienen elementos de datos y otros campos en los que podemos poner el tipo, la longitud y la descripción breve.
Si pulsamos sobre el botón “Referir elem. Datos” o a través del menú “Tratar”, “Referir elem. Datos” volveremos a poder introducir elementos de datos.

## OTROS

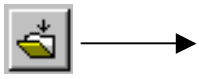
Para insertar nuevos campos utilizaremos el botón “Campos nuevos” o también a través del menú: “Tratar”, “campos nuevos” o CTRL+F9. Con esta opción se insertan bastantes campos, si solo queremos insertar nuevos campos vamos al menú: “Tratar”, “insertar campos”.

Para borrar un campo utilizaremos el menú: “Tratar”, “borrar campo”.

**POR ULTIMO**

Después de poner todos los datos que queramos, tenemos que grabar, verificar y activar.

Para grabar:

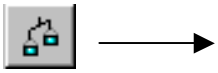
 Botón para grabar sin verificar.

Cuando grabemos por primera vez nos preguntará por la Clase de desarrollo. Esta clase nos la da la empresa donde estemos y sirve para el transporte a producción. Para practicar se deja en blanco.

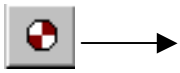
Para practicar la grabaremos como Objeto local (Botón que está abajo) nos pondrá como clase de desarrollo '$TMP'.

Como introducir una clase de desarrollo, se explica más adelante cuando creemos un programa, puesto que su funcionamiento es idéntico.

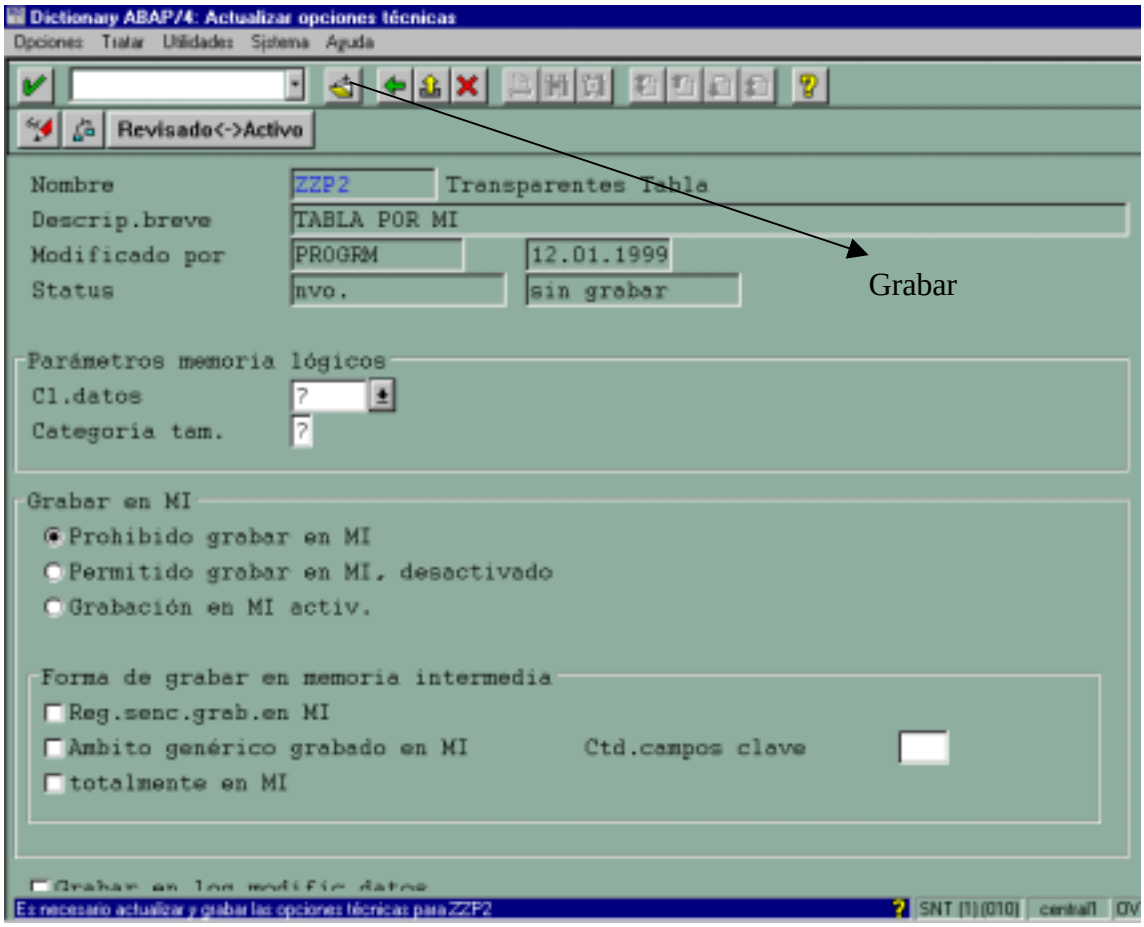
Una vez creado hemos de verificar la consistencia de la tabla, lo haremos con el botón:

 Verificar o Ctrl+F2

Después se ha de activar para poderlo utilizar. Se activa con el botón:

 Activar CTRL+F3

Cuando activemos la tabla, nos dará error, porque seguro que no habéis introducido dos campos que en la pantalla principal (Véase Fig. Tabla) no son visibles, la pantalla que aparece es la que vemos a continuación:



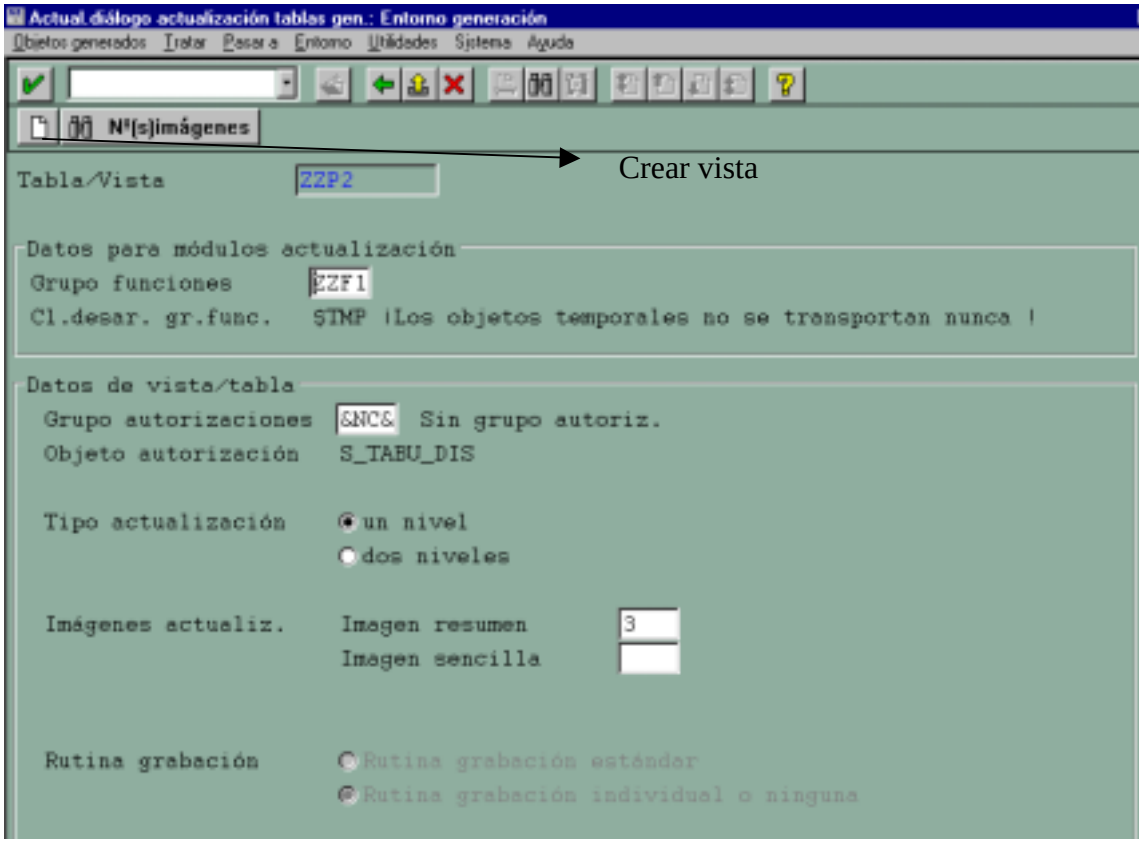

Los campos que faltan son los “Cl. Datos” y “Categoría tam.”
Si pulsamos F4 en los dos campos nos saldrán los datos disponibles a introducir, ponemos cualquiera o los que nos diga el cliente, y a continuación lo grabamos para que SAP los acepte.
Después volvemos pulsando F3 o la tecla de volver. Y automáticamente se activará, sino encuentra ningún otro error.

## VISTAS

Para poder introducir datos o asociar una tabla a un grupo de función para poder trabajar en ella tenemos que crear una vista. No es necesario crear una vista para poder introducir datos como veremos más adelante pero si que es útil para realizar otro tipo de tareas.

Para crear una vista la tabla ha de estar activada con anterioridad.

La vista se crea en el menú “Entorno”, “Gener.Actual.Tablas.” y saldrá la siguiente pantalla:




En el grupo de funciones podemos seleccionar los que hay pulsando la tecla F4 (para ver las existentes).

El grupo de funciones se utiliza para asociar diversos objetos (tablas, dynpros, programas, etc.) entre sí. Se utiliza para el transporte, quiero decir, cuando vamos a transportar nuestro proyecto (programas, tablas, etc.). Solo diciendo que transporte el grupo de funciones, se transportarán automáticamente todos los objetos que estén relacionados con este grupo de función.

Inconvenientes: si utilizamos un grupo de función ya existente puede ser que alguien transporte ese grupo de función (porque ya ha acabado su proyecto) y por la tanto también se transporten nuestros programas. Y no hay forma posible (seguramente SAP la tendrá escondida por ahí) de recuperar el programa o grupos de funciones que hayamos enviado. Por ello es aconsejable que creemos nuestro propio grupo de función.

Ventajas: podemos hacer que el programa cree una o varias dynpros, dependiendo de los objetos relacionados al grupo de funciones, de forma automática y sin tener que intervenir.

En este caso en particular hemos puesto el código de función ‘ZZJ1’.
En “grupo de autorizaciones”, haciendo clic en el campo y otro clic en el matchcode o F4:

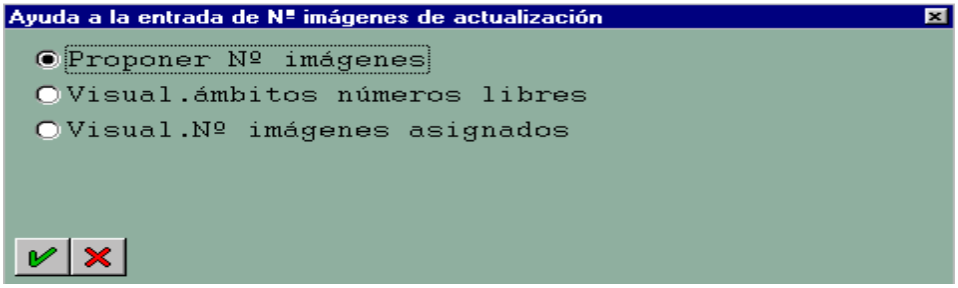
 → Seleccionar valores de un campo

Indicaremos el grupo de autorizaciones, que depende del usuario o usuarios a que vaya dirigido o del departamento donde estemos.

En “tipo actualización” se le suele poner un nivel.

En “imágenes de actualizaciones”:
en “imagen resumen” se pone un número que no ha de estar repetido, este número digamos que es el número de dynpro (explicada más adelante en el manual), se pone cuando queramos utilizar esa tabla en una dynpro, solo nos referiremos al número de la imagen sencilla.

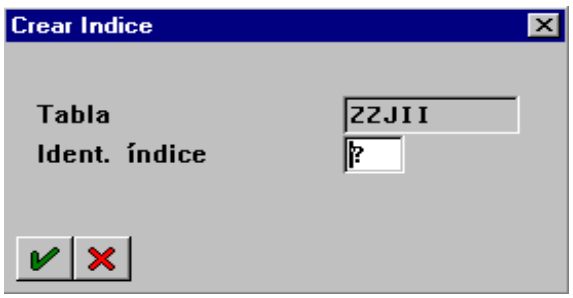
Para saber que imágenes están ocupadas o si queremos que SAP nos asigne una automáticamente, pulsaremos el botón que pone: “Nº(s) imágenes”. Aparecerá la pantalla donde podremos elegir que nos asigne una dynpro o mirar las que están ocupadas. La pantalla que veremos es la siguiente:




Yo suelo elegir la primera opción, ya que SAP mira la primera disponible y la asigna.
Después de introducir todos los datos pertinentes pulsaremos el botón de crear (es el icono que contiene una hoja en blanco)
Pulsamos ese icono para crearla y a continuación la grabamos

## COMO CREAR INDICES EN UNA TABLA

En una tabla también podemos crear índices. Para crearlos podemos pulsar CTRL+F5 o ir al menú: “Pasar A”, “Indices”. Si no hay ninguno creado nos preguntará si lo deseamos crear. Si le decimos que sí nos pedirá la identificación del índice, la pantalla que saldrá será la siguiente:




Si ya existe uno nos aparece esta pantalla:

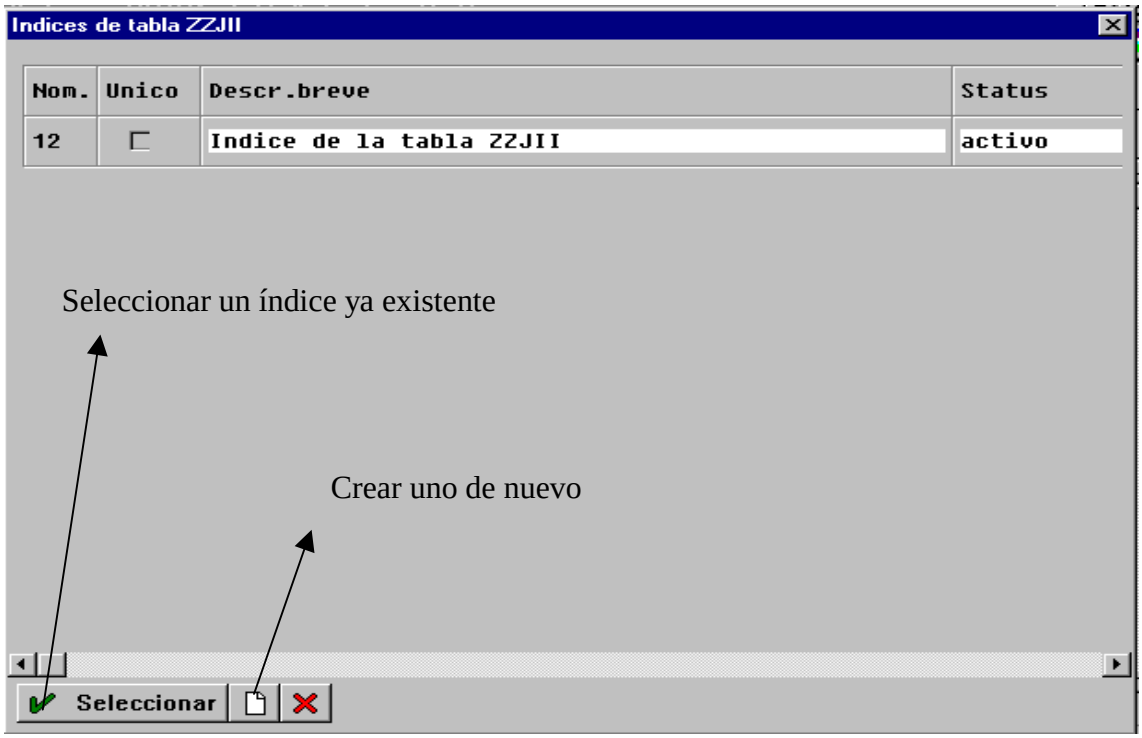
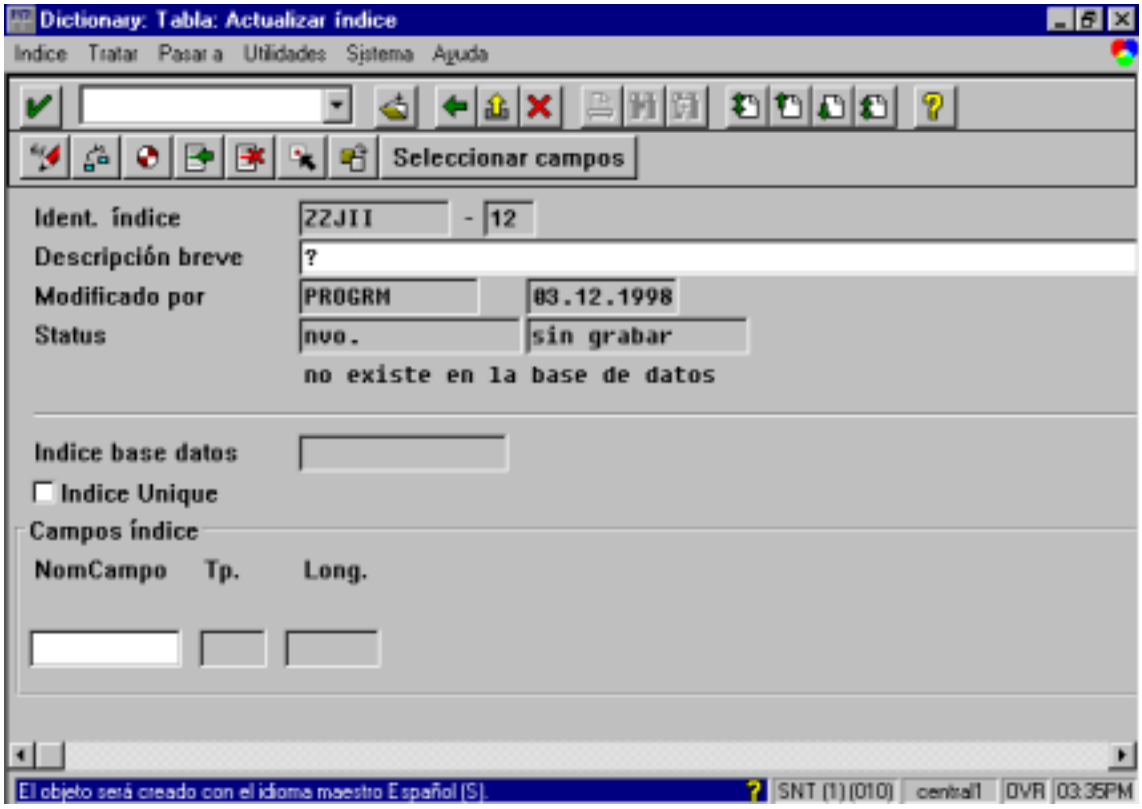



Fig. Ind.

En esta pantalla podemos seleccionar un índice para modificarlo o borrarlo, o crear uno nuevo en el botón en que aparece una hoja en blanco.

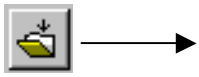
Tanto si lo creamos como si lo modificamos aparecerá esta pantalla:



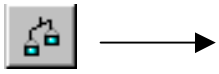

Primero escribiremos una descripción breve sobre la utilidad del índice.

En el "índice unique" le indicaremos si el índice es único o no. Si solo hay un campo clave en la tabla lo podemos poner, si no, es mejor no hacerlo.
En "NomCampo" pondremos los campos que formarán el índice. Se puede poner a mano o con el botón "Seleccionar campo" que está en el "Menú Painter".
En el menú "Pasar A" podemos insertar un nuevo campo o borrarlo.

Después de crearlo lo grabaremos

 Botón para grabar sin verificar.

Una vez creado hemos de verificar la consistencia del índice, lo haremos con el botón:

 Verificar o Ctrl+F2

Después se ha de activar para poderlo utilizar. Se activa con el botón:

 Activar CTRL+F3

Recordar que para activar un índice la tabla ha de estar activada antes.

Cuando esté todo correcto y volvamos a la pantalla anterior nos saldrá una pantalla con los índices de la tabla. La pantalla es la “Fig Ind.”.

Recordar que realizar muchos índices en una tabla repercute en el rendimiento.
¿Por qué repercute?, Pues porque cuando añadimos, modificamos o borramos un registro en la tabla, el sistema ha de actualizar todos los índices de la tabla. Si hay pocos datos no ocurre nada, pero si hay muchos el sistema se ralentiza mucho.

## COMO HACER CLAVES FORANEAS Y TABLAS DE VERIFICACION

Para hacerlas hemos de estar en la pantalla de modificación de tablas, o sea, en la siguiente pantalla:

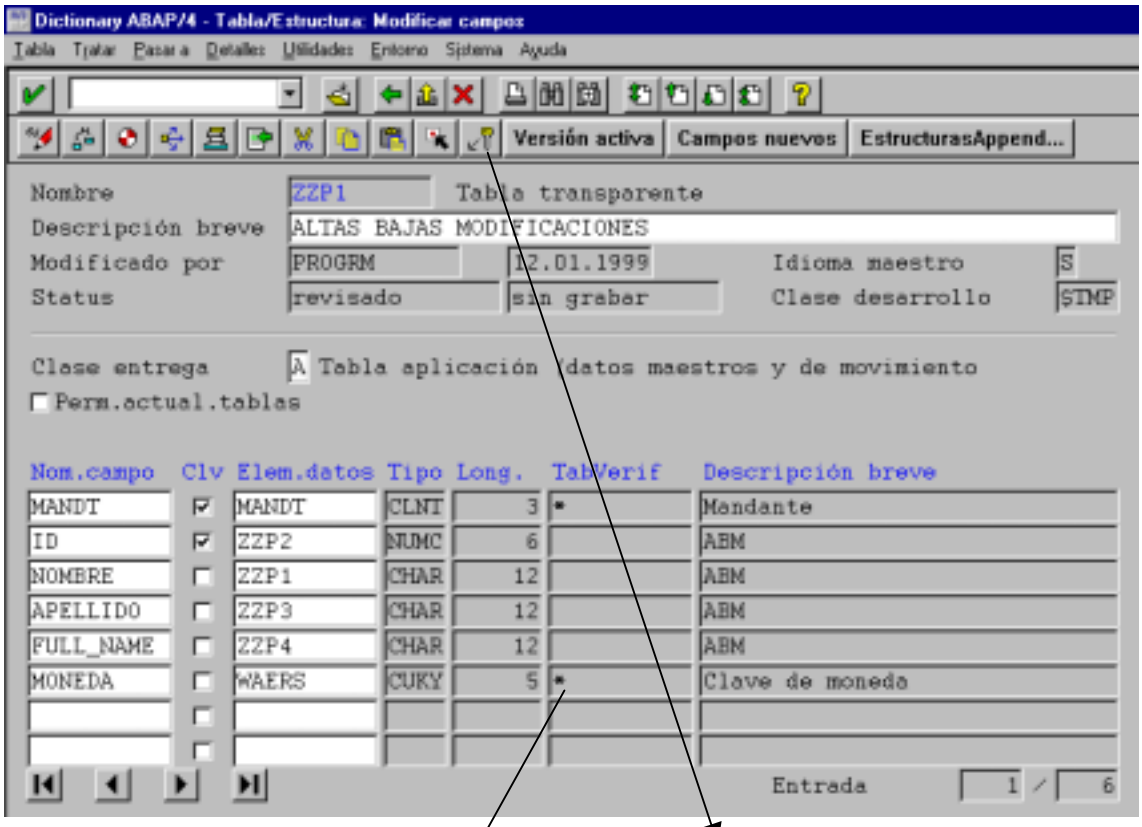



Fig. Tabla

Él ‘*’ puede tener tabla de verificación

Claves foráneas o F8

Cuando sale un ‘*’ en “TabVerif” SAP avisa que puede haber una o más de una tabla de verificación. Para poder asociar ese campo a esa tabla de verificación hay que realizar claves foráneas, también se puede hacer de un campo que no tiene un ‘*’.

Primero lo haremos de un campo que no tiene ‘*’ y seguidamente lo haremos de un campo que tiene un ‘*’.

**TABVERIF SIN ‘*’**

Para poder realizarlas nos hemos de posicionar con el cursor en el campo donde queremos realizar la clave foránea. Cuando estemos le daremos al botón de claves foráneas o pulsaremos F8, entonces nos saldrá la siguiente pantalla:

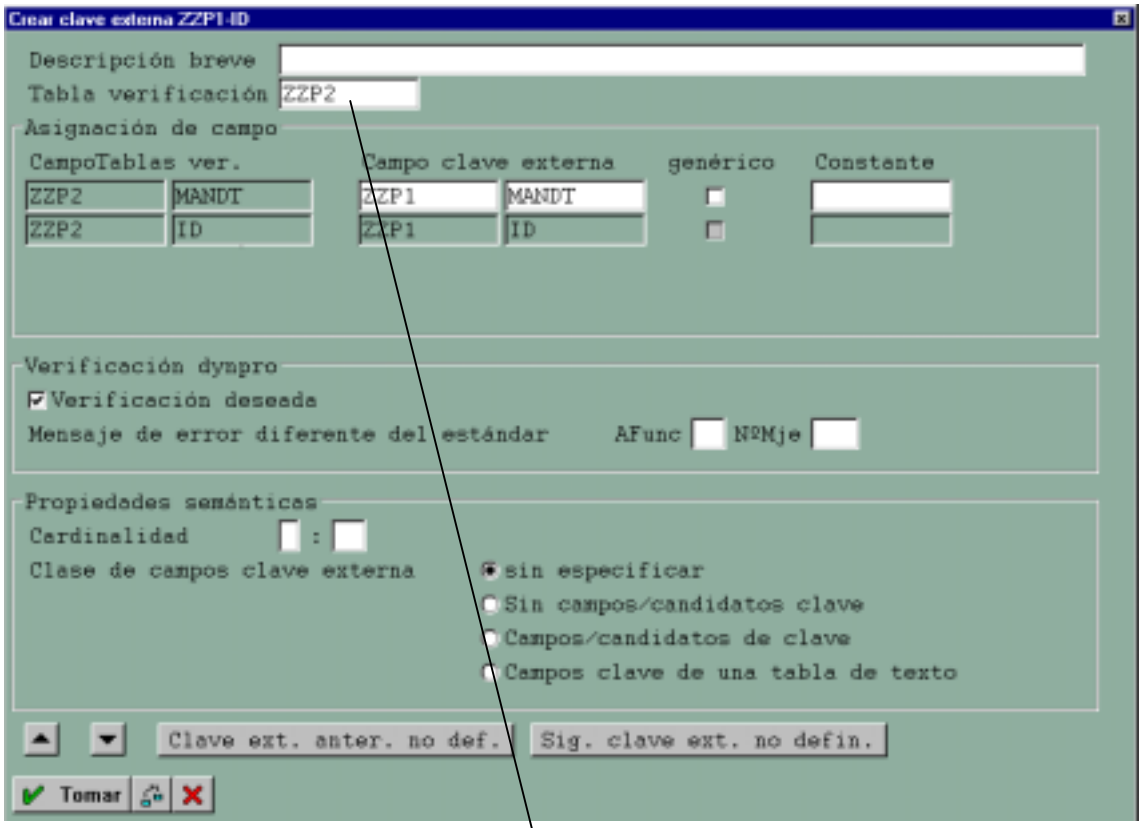
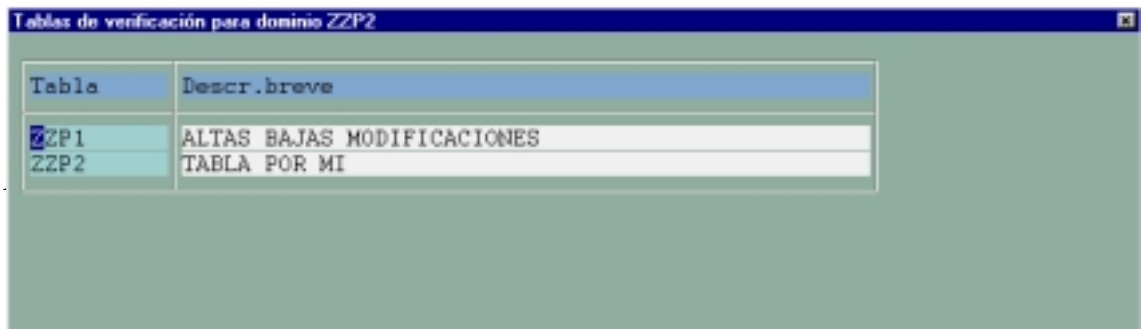



Fig. Claves foráneas

Tabla donde estará el campo a relacionar

Como se ve en esta imagen, ya he puesto la tabla ZZP2 que sé que tiene campos comunes, y pulsado ENTER automáticamente me ha sacado todos los campos de la tabla escogida.

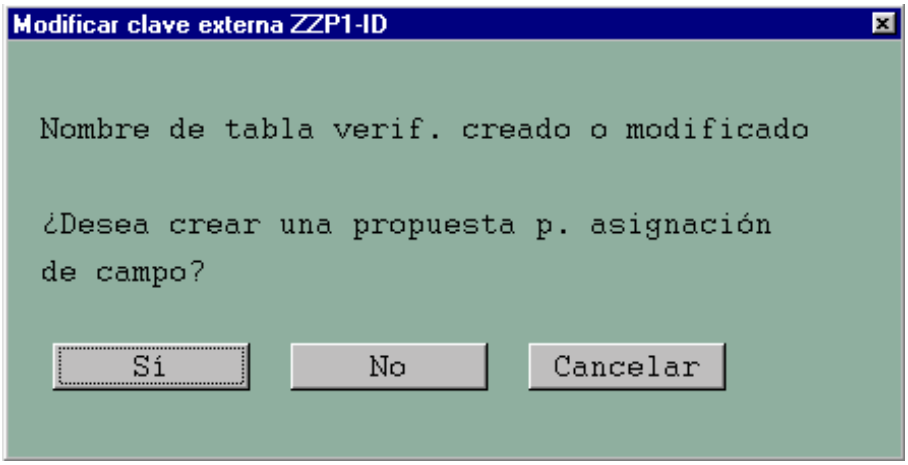
Donde pone “Tabla de verificación” escribiremos qué tabla queremos relacionar con nuestro campo. Para saber qué tablas tienen campos compatibles con nuestro campo posicionamos el cursor donde pone “Tabla verificación”, después pulsamos F4 o le damos al matchcode y nos saldrán las tablas, como en esta imagen (en la página siguiente):



En este caso solo nos salen dos. Vemos que la primera tabla que sale es la nuestra y la segunda la que he creado con un campo que tiene el mismo dominio que el campo que queremos relacionar.

Ya que SAP mira y visualiza las tablas que tengan algún campo con las mismas características que nuestro campo, cuando ponemos un elemento de datos, le damos ENTER y nos aparece un '*' en "TabVerif" es que SAP ha encontrada alguna tabla con el mismo elemento de datos.

Si hacemos doble clic en la tabla que queramos o hacemos el clic en la tabla y después pulsamos el botón de seleccionar nos aparecerá, la siguiente pantalla:

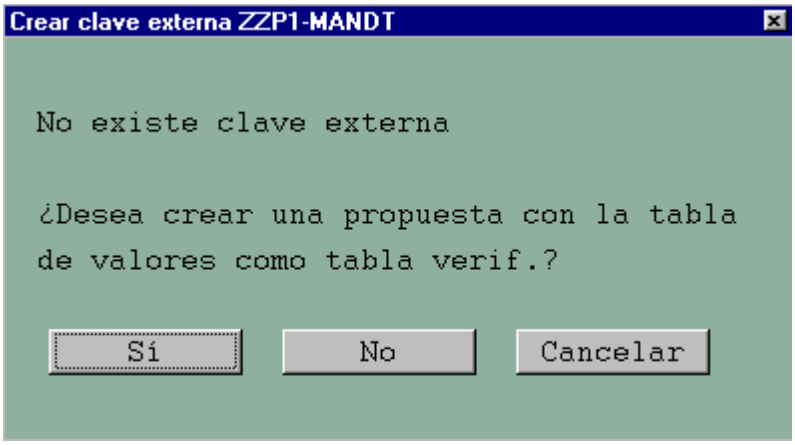



Aquí pulsaremos el botón para que nos muestre los campos que tiene la tabla que hemos escogido, para ver si nos interesa o no.
En la pantalla de claves foráneas (Véase Fig. Claves foráneas) nos aparecerá la tabla que hemos seleccionado con sus campos, ya que en la pantalla anterior le hemos dicho que 'sí' a la propuesta de asignación de campo, si le decimos que no, no nos saldrán los campos de la tabla.

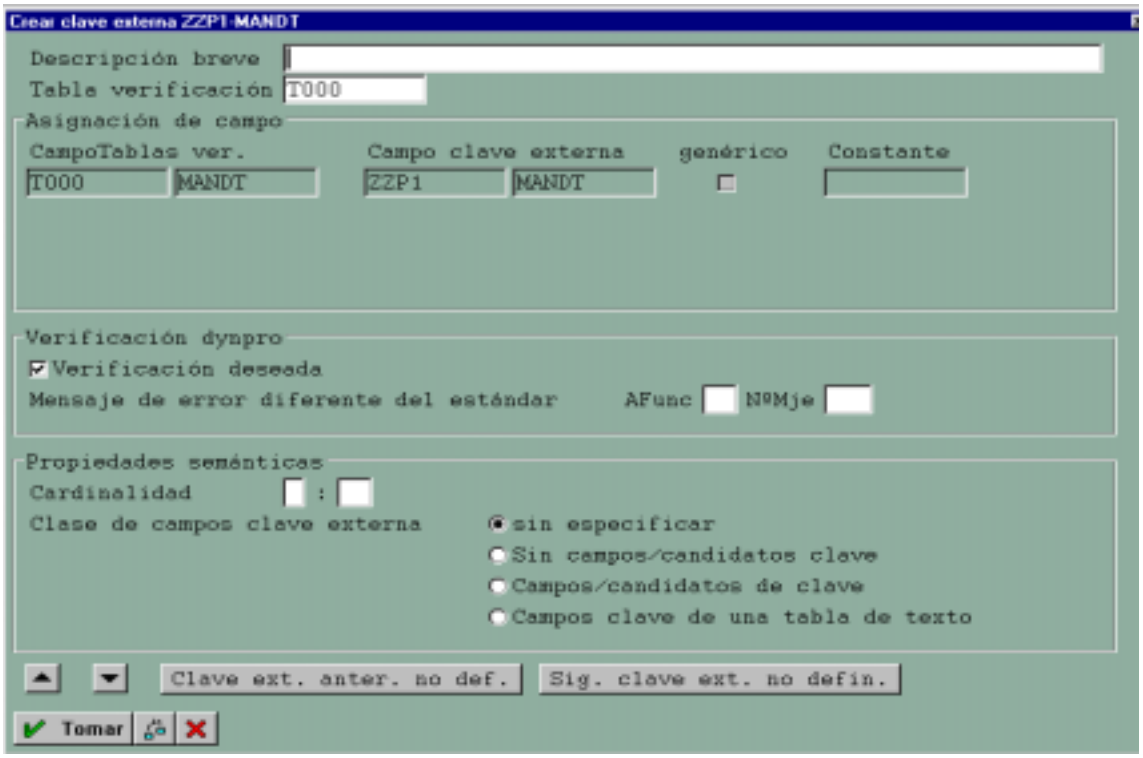
Después de elegir la tabla, es la hora de tomar la tabla que hemos escogido. Para tomarla le damos al botón que pone “Tomar”, y él automáticamente nos pondrá la tabla en el campo TabVerif (Véase Fig. Tabla).

**TABVERIF CON ‘*’**

Si en TabVerif sale un ‘*’, SAP nos dice que hay una tabla de verificación. Los pasos son los mismos que en el anterior, nos posicionamos con el cursor y pulsamos F8 o el botón de claves foráneas, y a continuación nos saldrá la siguiente pantalla:

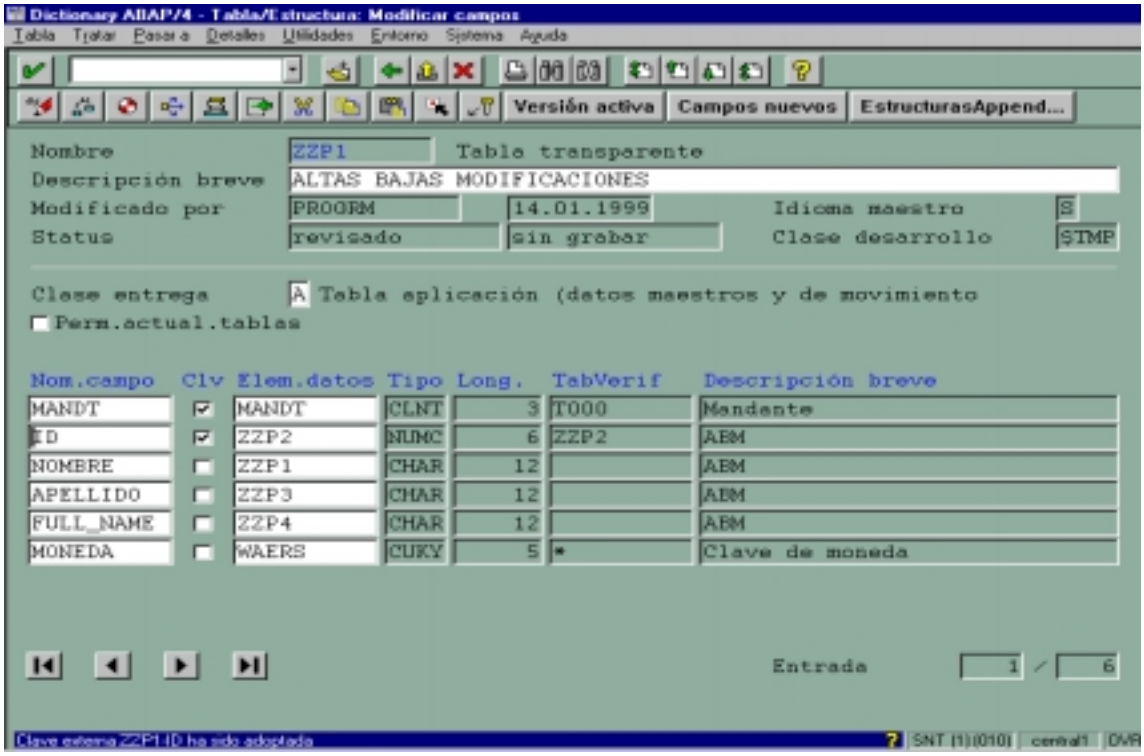



En esta pantalla, nos dice si queremos que SAP nos saque la tabla más conveniente a nuestro campo. Yo suelo poner que sí para que me ponga la tabla de forma automática, si le decimos que no, la tendremos que buscar por nuestra cuenta (y por desgracia hay un montón de tablas). Si le decimos que sí, nos saldrá la siguiente pantalla (en la siguiente página):




Como vemos la tabla T000 es la más adecuada en este caso, ahora, si pulsamos el botón "Tomar" nos tomará automáticamente la tabla. Si vemos que no nos interesa esta tabla podemos buscar la que queramos (como en el ejemplo anterior, o sea, pulsando F4 en el campo de tabla de verificación).

El resultado, si hacemos los dos ejemplos anteriores, sería este:



En la imagen se nos ha olvidado poner una tabla de verificación, pero como ejercicio podéis hacerlo vosotros.

**TABLAS MAS USADAS EN LA TABLA DE VERIFICACION**

Dependiendo del elemento de datos que hayamos puesto, nos puede salir una tabla de verificación u otro.

Ahora pondré un ejemplo de elementos de datos con su tabla de verificación:

| ELEMENTOS DE DATOS | TABLA DE VERIFICACION |
|---|---|
| MANDT (Mandante) | T000 |
| BC_COUNTRY (País) | T005 |
| WAERS (Moneda) | TCURC |

Como vemos estos 3 elementos de datos son internos del sistema, por lo tanto en "TabVerif" se pone automáticamente un '*', y las tablas de verificación son las que SAP aconseja para ese elemento de datos (que se asignan haciendo claves foráneas, como hemos explicado antes).

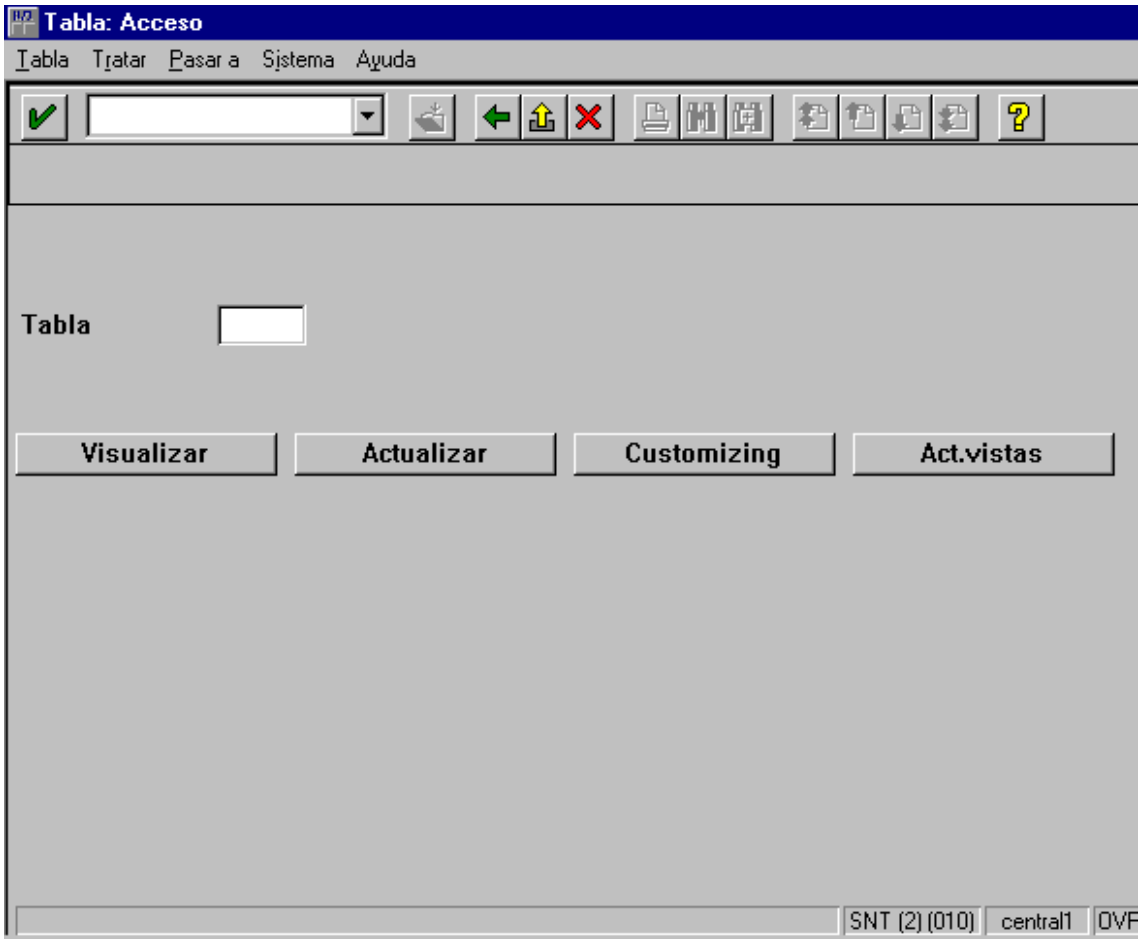
Un consejo, todas las tablas que empiezan por T son tablas de verificación. Si queremos ver las que hay desde la pantalla del Dictionary, en el campo "Tabla" se introduce T* y si pulsamos F4, nos muestra todas las tablas disponibles.

**VER EL CONTENIDO, AÑADIR O MODIFICAR DATOS DE UNA TABLA**

Podemos ver el contenido de una tabla, añadir datos o modificar datos de una tabla. Se puede hacer de dos formas muy sencillas: desde la pantalla de ABAP/4 Development Workbench vamos al menú "Resumen", "Data browser", o en la pantalla donde tratamos una tabla (Véase Fig. Tabla) vamos al menú "utilidades", "contenido tabla". En los dos casos vamos al Data Browser que será explicado más adelante.

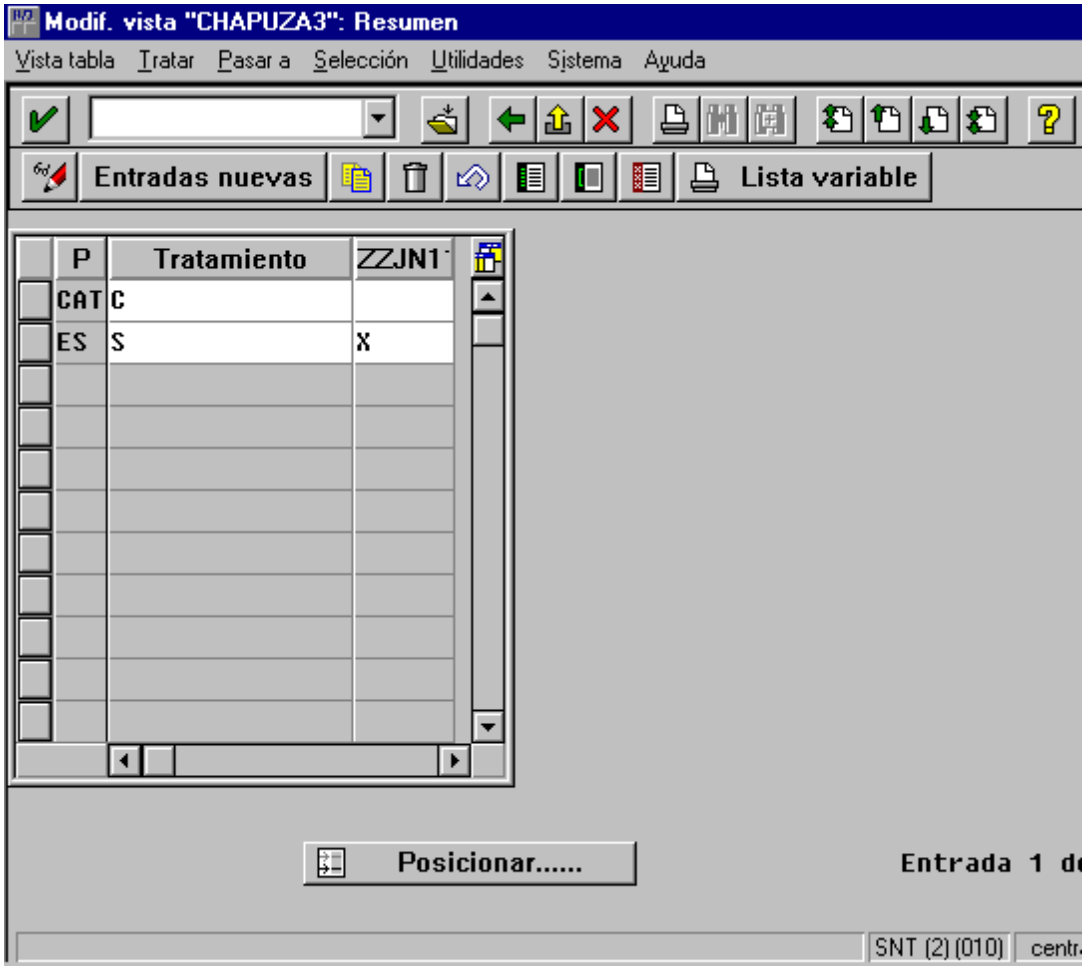
También tenemos otra forma de introducir datos que es a través de las vistas que hayamos creado.

Para ello tenemos que ir al menú "Sistema", "servicios", "actualizar tablas". Esta pantalla la podemos llamar desde cualquier punto de SAP, la pantalla que sale es la siguiente:



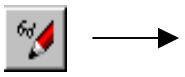

En “tabla” pondremos el nombre de la tabla que queramos visualizar, actualizar, customizing o act. Vistas.

Tanto en “visualizar”, “actualizar” iremos a la siguiente pantalla (En la página siguiente):



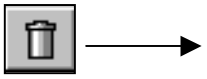

Esta pantalla es idéntica a la vista anteriormente (Véase Fig. Resumen)

Con este icono:

 Cambiar el modo de visualización/modificación

Se podrán modificar o visualizar los datos de la tabla.

Con este icono:

 Borrar entrada o entradas

podremos borrar una entrada o varias dependiendo de cuantas seleccionemos

Para poder seleccionar varias entradas pulsamos el botón de selección.

Con el botón de entradas nuevas podremos introducir nuevos datos a la tabla, para poderlos grabar se utiliza el botón de la carpeta abierta (con el que siempre grabamos).

## TABLAS DEL SISTEMA

SAP tiene una serie de tablas de utilización diversas. Las variables del sistema o las propiedades de una pantalla están almacenadas en una tabla. A continuación explicare las tablas más usadas en SAP.

### SCREEN

SAP coge cualquier pantalla como si fuera una tabla de diccionario, es decir, que para acceder a cualquier objeto hemos de hacer un "loop", lo que quiere decir que cada objeto es un registro y la pantalla una tabla. Los campos de esta tabla ya fueron explicados en el tema de "tablas del sistema". Y la tabla donde se guarda todo esta se llama "SCREEN" que tiene la siguiente estructura:

| Campo | Longitud | Descripción |
|---|---|---|
| NAME | 30 | Nombre del campo o del objeto |
| GROUP1 | 3 | Grupo 1 al cual pertenece |
| GROUP2 | 3 | Grupo 2 al cual pertenece |
| GROUP3 | 3 | Grupo 3 al cual pertenece |
| GROUP4 | 3 | Grupo 4 al cual pertenece |
| ACTIVE | 1 | Si esta activado. se puede ver e introducir datos |
| REQUIRED | 1 | Si se ha de introducir obligatoriamente |
| INPUT | 1 | Si se puede introducir datos en él |
| OUTPUT | 1 | Si se puede visualizar datos en él |
| INTENSIFIED | 1 | Los datos se visualizan con un color intensificado o no |
| INVISIBLE | 1 | Si es visible o invisible |
| LENGTH | 1 | Para acotar la longitud de salida de un campo |
| DISPLAY_3D | 1 | Si se visualiza con una frame de 3D |
| VALUE_HELP | 1 | Si se visualiza con un valor de ayuda |

Todos los campos son tipo carácter o CHAR.

Los atributos que se pueden modificar en tiempo de diseño son lo siguientes:

```
SCREEN-REQUIRED.
SCREEN-INPUT.
SCREEN-OUTPUT.
SCREEN-INTENSIFIED.
SCREEN-INVISIBLE.
SCREEN-ACTIVE.
SCREEN-LENGHT.
```

Los valores que pueden coger estos campos son: 0->Desactivado o 1->Activado.

### SYST

Posiblemente sea la tabla más importante de SAP, en esta tabla se guardan todas las variables del sistema. Solo mostraré las que utilizo más a menudo:

| Campo | Longitud | Descripción |
|---|---|---|

| | | |
|---|---|---|
| SUBRC | | Devuelve un valor cuando se produce un error<br>0 -> No hay errores<br>4 y 8 -> Se ha producido algún error. |
| REPID | | Nombre del programa |
| DATUM | | Fecha de hoy |
| UZEIT | | Hora del sistema |
| PAGNO | | Nº de página del listado |
| TABIX | | Indice de tablas |
| MANDT | | Mandante |
| OPSYS | | Programa |
| UCOMM | | Botón o tecla pulsada |
| LSIND | | Número de ventana, creada a través de la orden WINDOW |
| UNAME | | Nombre del usuario |
| FDPOS | | Posición donde se encuentra el string a buscar de la orden SEARCH |

Para poder utilizar estas variables se ha de poner delante de ellas: SY-variable, ejemplo:

SY-UCOMM.

**OBJETOS MATCHCODE**

Un objeto matchcode es este icono:  sale en aquellos campos donde podemos seleccionar un valor o varios valores.

**CREAR UN MATCHCODE**

Para crearlo tenemos que ir al Dictionary y ahí seleccionar “objetos matchcode”, después escribiremos el nombre que le daremos (máximo de 4 letras), después pulsaremos el botón “crear” y nos saldrá la siguiente pantalla:

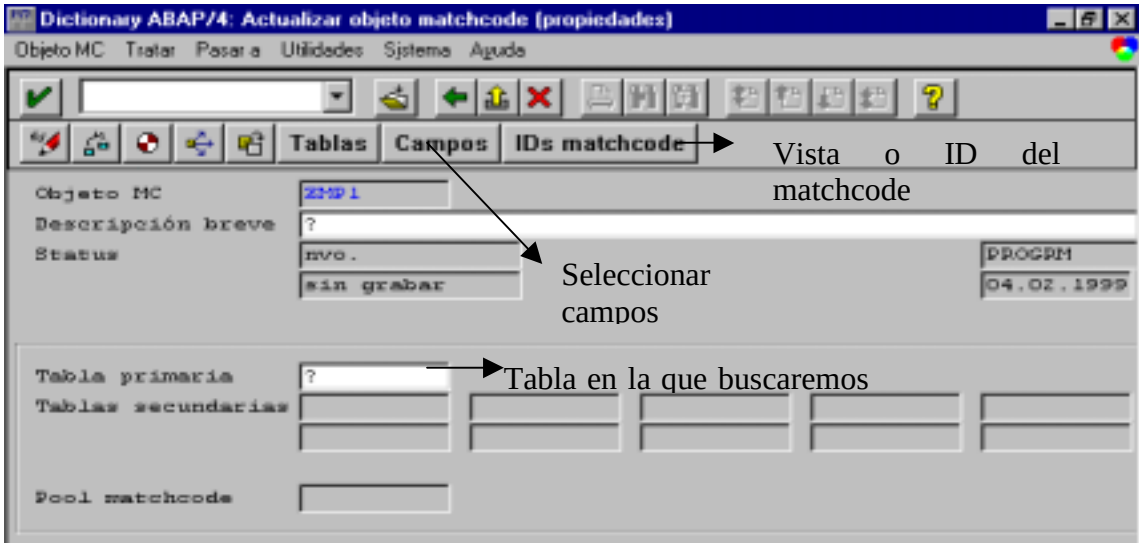


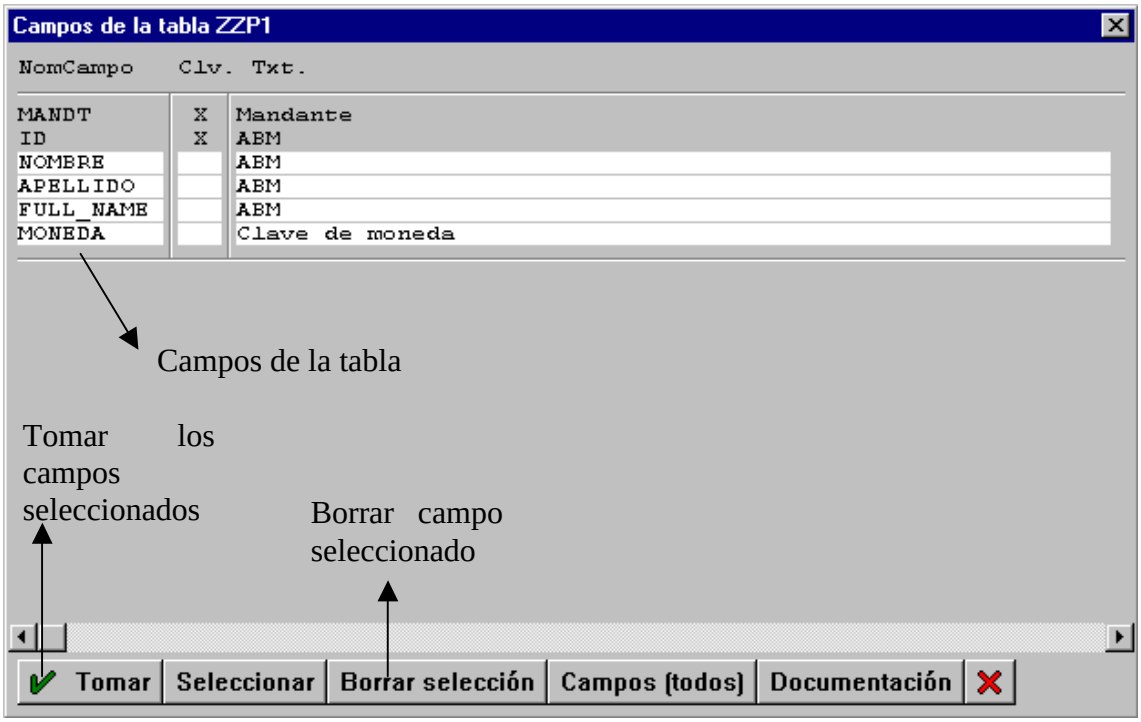

Fig. Matchcode.

Antes de todo hemos de poner la descripción breve y la tabla primaria en esta tabla donde se van a coger los campos a utilizar en el matchcode.
Cuando hayamos introducido estos campos obligatorios es la hora de escoger que campos vamos a utilizar. La parte de selección de campos la llamaremos: vista standard.

Pulsamos el botón “campos” y nos saldrá la siguiente pantalla:


Fig. Campos.

Al principio de todo nos coge los campos clave de nuestra tabla, para coger los campos que no son claves tenemos que pulsar el botón “Seleccionar campos” y nos saldrá la siguiente pantalla:

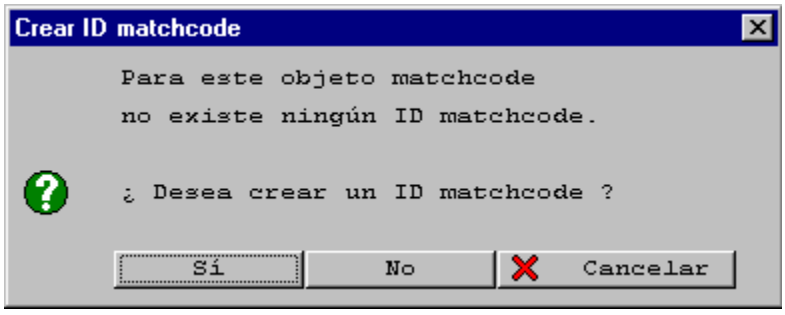



Los campos que tienen un fondo gris es que están seleccionados (los campos clave siempre están seleccionados) y los que están en blanco son los que podemos seleccionar, los podemos seleccionar haciendo clic o doble clic y cuando hayamos seleccionado los campos que queramos pulsamos el botón “tomar”.

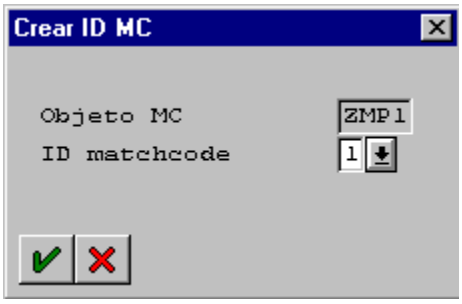
Cuando hayamos tomado los campos nos volverá a la pantalla de campos (Véase Fig. Campos), lo grabaremos para que nos tomé los campos seleccionados y seguidamente volveremos a la pantalla principal de “objeto macthcode” (Véase Fig. Matchcode)
lo generaremos.

Ahora solo nos falta crear la vista/ID o vistas/IDs. Estas vistas o IDs sirven para indicar que campos se realizarán el matchcode, la selección de campos que hemos hecho antes es solo para decir a SAP de que campos se pueden realizar las vistas.

Para realizar la vista pulsamos el botón “IDs matchcode” y nos saldrá la siguiente pantalla:




Esta pantalla nos informa de que nos hay ninguna vista/ID creada y si la queremos crear, esta pantalla solo aparecerá si no tenemos ninguna vista creada. Le diremos que “Sí” y nos saldrá esta otra pantalla:




Aquí le indicaremos el número de ID o vista por defecto nos sale la primera libre pero si queremos ver las que hay creadas le damos al botón de mathcode o F4, cuando hayamos puesto el número de ID pulsaremos “enter” o el botón de confirmar y nos saldrá la siguiente pantalla:

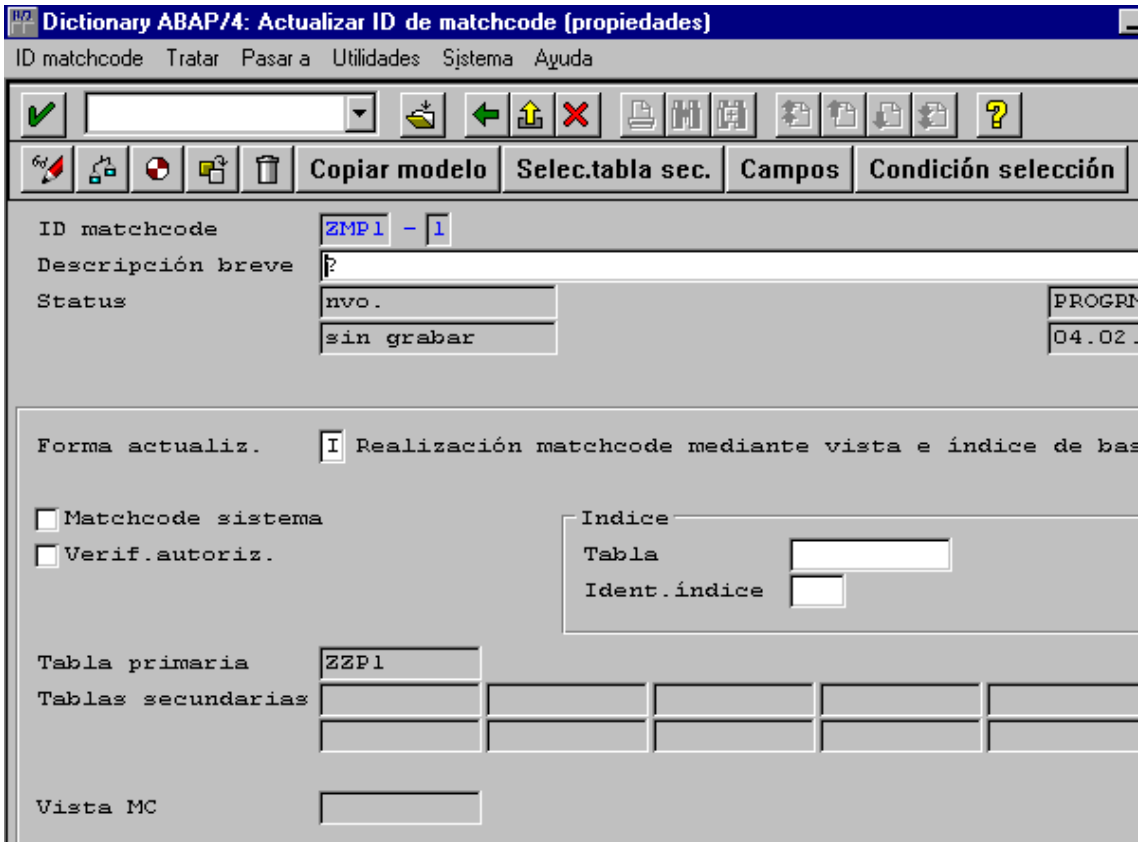



Fig. ID.

Aquí primero le pondremos la descripción breve y después los campos a utilizar. Para escoger que campos vamos a utilizar pulsamos el botón “campos” y nos saldrá esta pantalla (en la página siguiente):

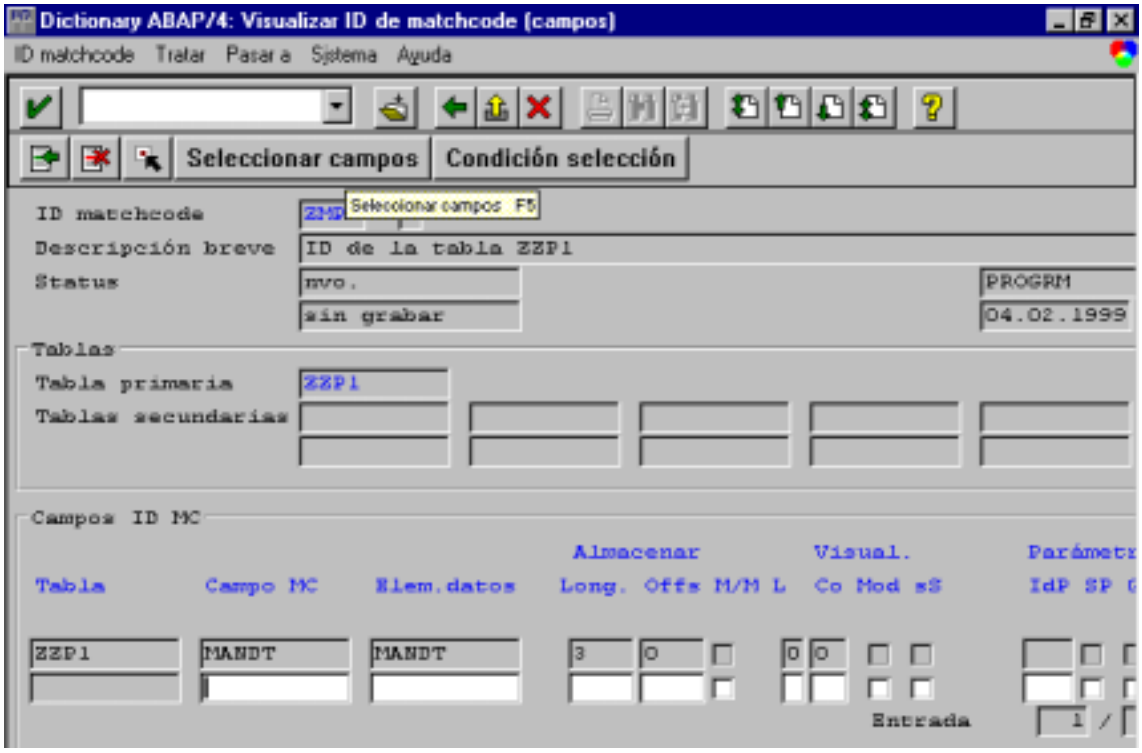
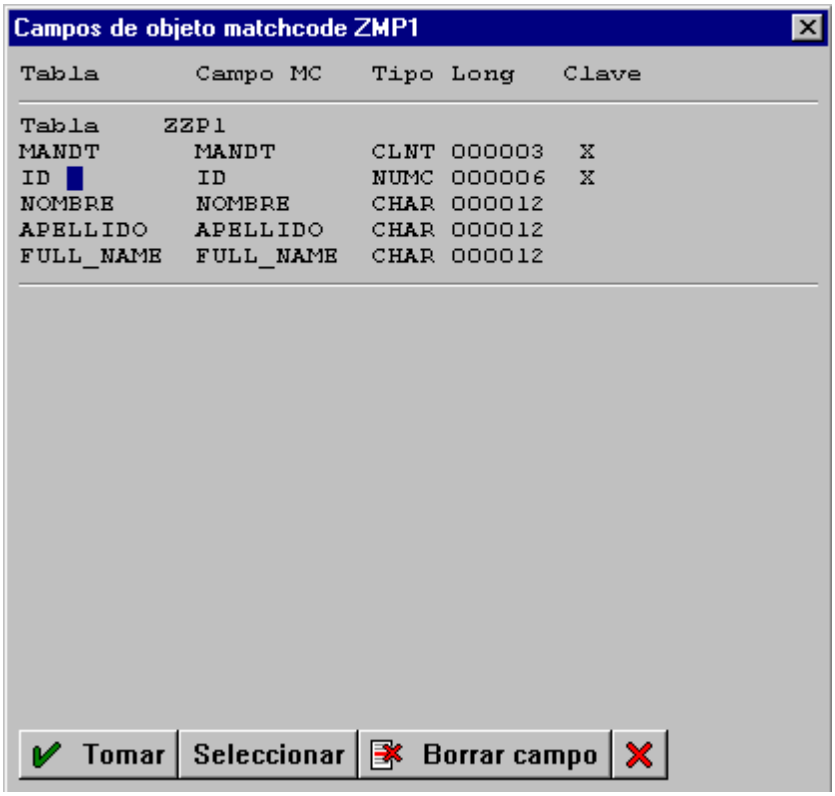



Fig. ID campos.

También aquí nos ha cogido el "mandante" pero no el resto de campos claves y los que no lo son, que los tenemos que escoger. Para escogerlos pulsamos el botón "seleccionar campos" y nos saldrá esta otra pantalla:




En este caso ya he seleccionado los campo a utilizar, que son todos pero podemos escoger los que queramos. Después de haberlos escogidos pulsamos el botón "Tomar".

Si aquí seleccionamos un campo que no hemos seleccionado en la vista standard (la selección de campos que hemos hecho antes) después en el programa no podremos guardar los valores, como veremos más adelante.

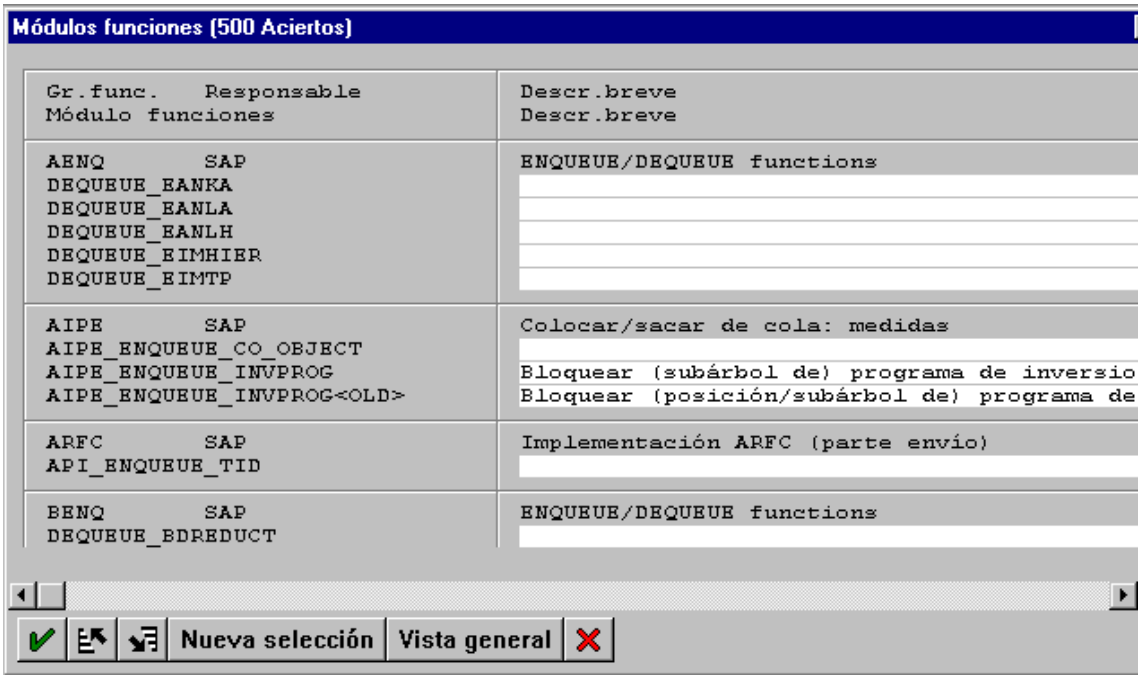
Cuando tomemos los campos nos volverá a la pantalla de crear ID (Véase Fig. ID campos) grabaremos los campos y pulsaremos F3 o el botón “Volver” para volver a la pantalla inicial (Véase Fig. Matchcode) y ahí de nuevo lo volveremos a generar. Cuando lo generemos es posible que nos dé un “Warning” pero no pasa nada.

Después de todos estos pasos ya tenemos el matchcode listo para ser utilizado. Más adelante veremos como utilizar el matchcode en un ABAP.

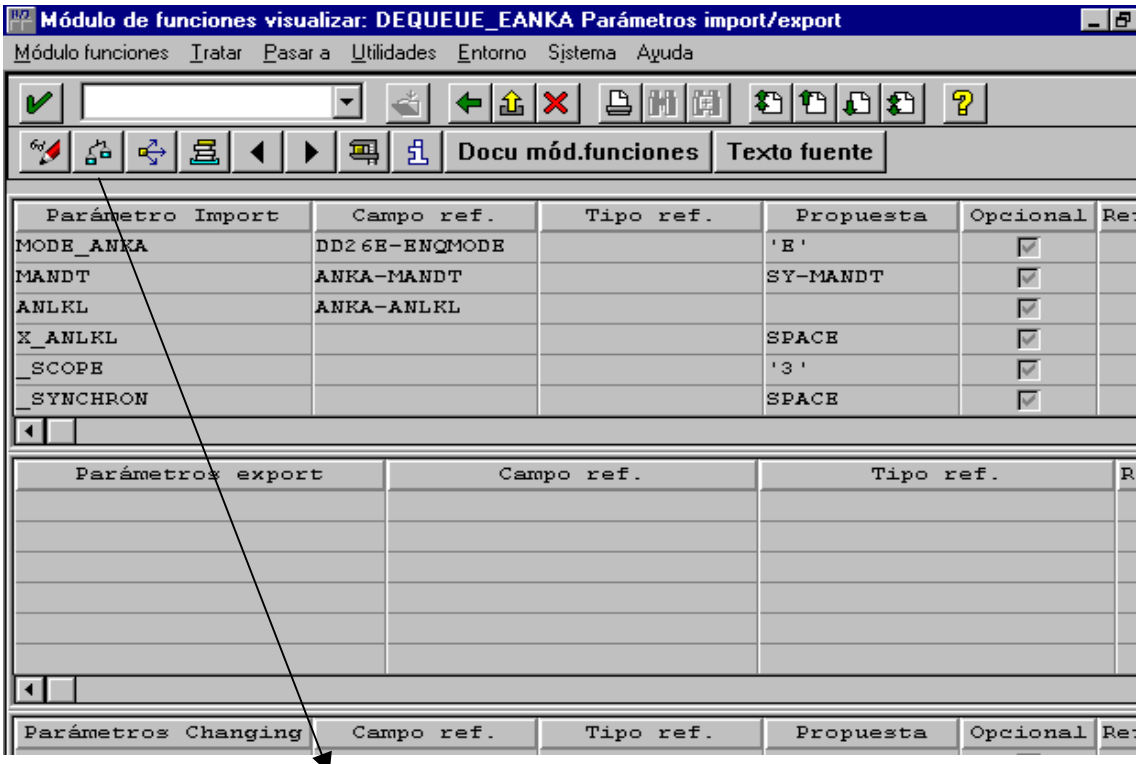
**OBJETOS DE BLOQUEO**

Los objetos de bloqueo solo los podemos ver ya que modificar o crear uno de nuevo solo lo pueden hacer los consultores de SAP.

Para ver los objetos de bloqueo o desbloqueo desde la pantalla de “Biblioteca de funciones” seleccionamos el radiobutton de donde queramos sacar la función y en el campo “Modulo de funciones” escribimos “*queue*” (Bloquear) o “*dequeu*” (para desbloquear) pulsamos F4 y nos saldrá la siguiente pantalla con las funciones de bloqueo:




Si hacemos doble clic sobre una función en el campo “Modulo de función“ de la pantalla de “Biblioteca de funciones” nos saldrá la función escogida y pulsamos el botón de visualizar nos saldrá la pantalla de información de esa función:




Referencia de utilización

Si pulsamos sobre referencia de utilización veremos que programas utilizan esa función y como la utilizan.

## DATA BROWSER

Podemos ver el contenido de una tabla, añadir datos o modificar datos de una tabla. Se puede hacer de dos formas muy sencillas: desde la pantalla de ABAP/4 Development Workbench vamos al menú “Resumen”, “Data browser”, o en la pantalla donde tratamos una tabla (Véase Fig. Tabla) vamos al menú “utilidades”, “contenido tabla”. En los dos casos nos saldrá la siguiente pantalla:

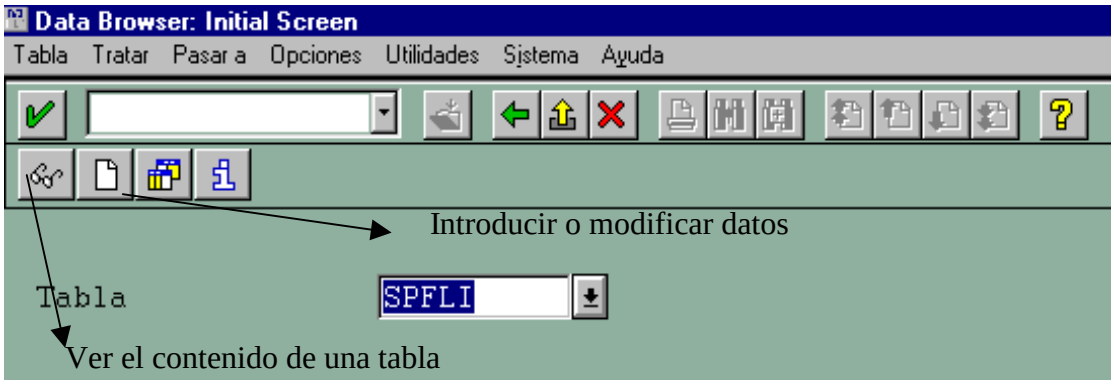
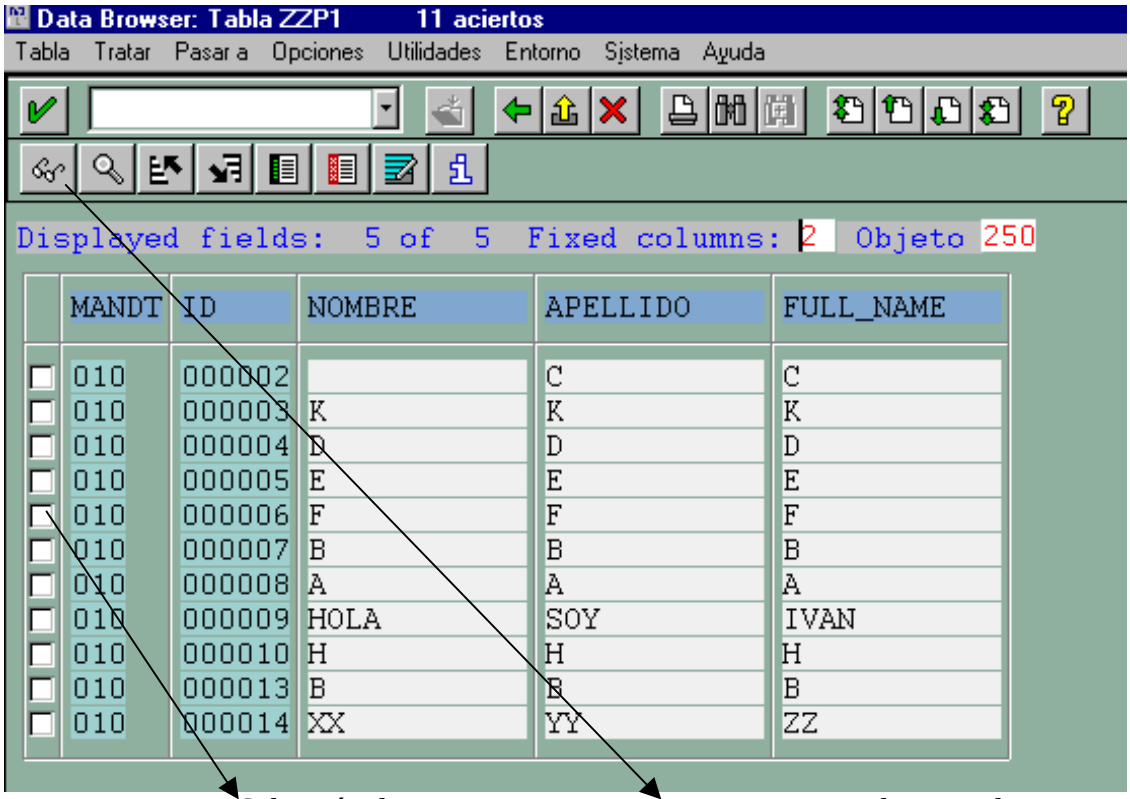



Fig. Data Browser.

Desde aquí podemos añadir datos a la tabla o también podemos ver su contenido.

Para ver el contenido introducimos el nombre de la tabla, después pulsamos ENTER o le damos al icono de las gafas o en el menú “tabla”, “contenido tabla”. Al pulsar nos saldrá la siguiente pantalla:


Desde aquí podemos realizar los criterios de búsqueda, si queremos hacerlo.
Para ver el contenido de la tabla, con o sin criterios de búsqueda pulsamos al botón de “ejecutar búsqueda” y nos saldrá la siguiente pantalla con el listado de campos, en la siguiente pantalla:




Selección de registros Ver registros seleccionados

Si por ejemplo la tabla tiene muchos campos y deseamos verla de una forma apaisada, podemos hacerlo de dos formas diferentes: la primera es haciendo doble clic en un registro y automáticamente nos saldrán los datos del registro seleccionado uno detrás de otro.
La segunda permite ver un registro o varios registros (a través de una paginación). Para hacerlo tenemos que seleccionar el registro o registros que deseamos ver (con el check box que hay a la izquierda de los registros), después pulsamos sobre el icono de las gafas y nos saldrá una pantalla con el registro o registros seleccionados.

Si queremos añadir o modificar datos en la tabla, desde la pantalla del Data Browser (Véase Fig. Data Browser) pulsamos el botón de la hoja en blanco y nos sale la siguiente pantalla para poder introducir datos:

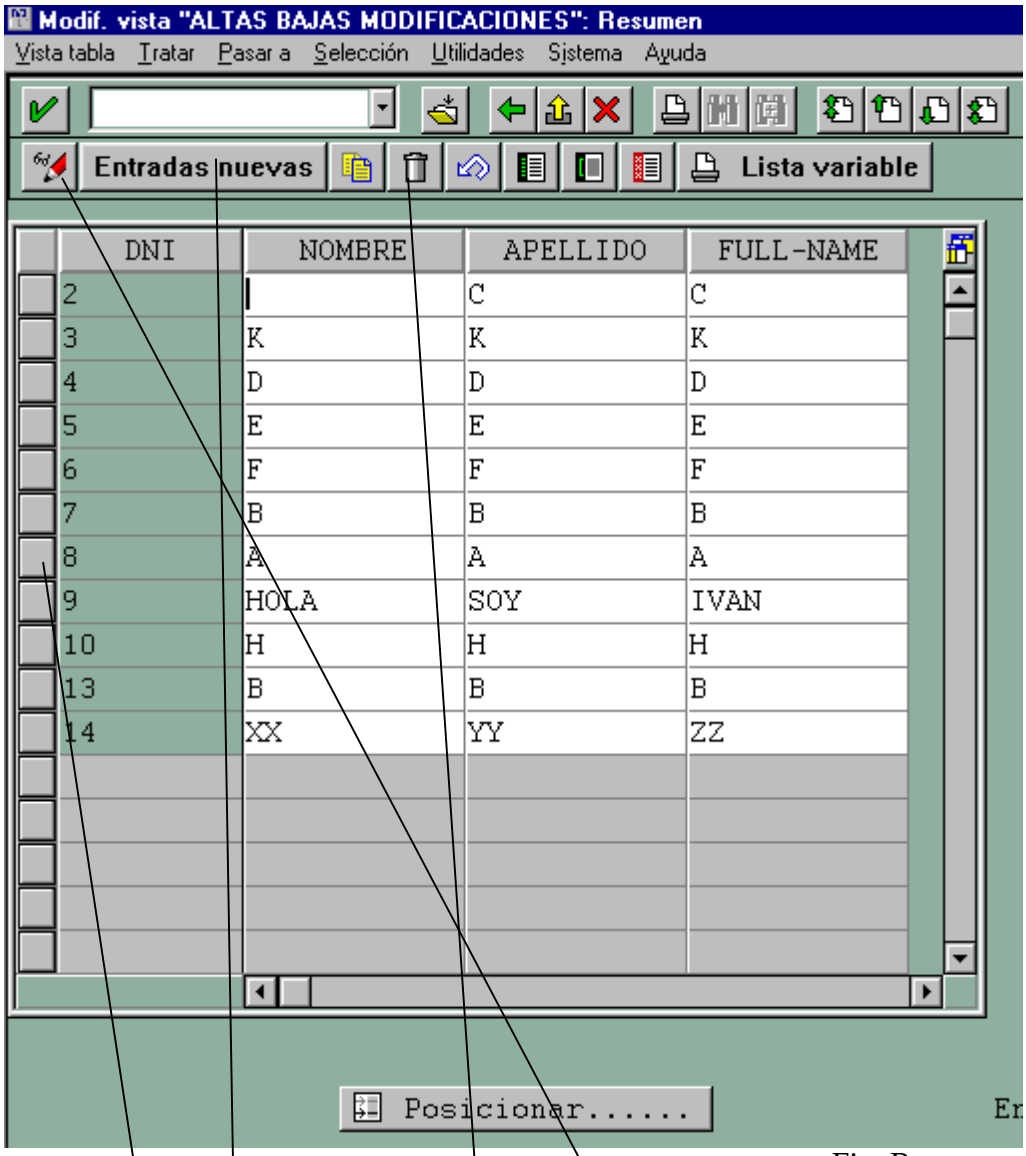

Modif. vista "ALTAS BAJAS MODIFICACIONES": Resumen
Vista tabla Tratar Pasar a Selección Utilidades Sistema Ayuda
Entradas nuevas
Lista variable
DNI NOMBRE APELLIDO FULL-NAME
2 C C
3 K K K
4 D D D
5 E E E
6 F F F
7 B B B
8 A A A
9 HOLA SOY IVAN
10 H H H
13 B B B
14 XX YY ZZ
Posicionar.....
Seleccionador de registros
Un nuevo registro
Borrar un registro
Cambiar de modo visualización/modificación

Fig. Resumen.

## OBJECT BROWSER

Con el “Object browser” podemos crear, modificar, borrar, etc. cualquier objeto de SAP. Para poder acceder a el vamos a la pantalla de “ABAP/4 Development WorkBench” y ahí pulsamos el botón “Object Browser” o en el menú “Resumen”, “Object browser o F5”. Y nos saldrá la siguiente pantalla:

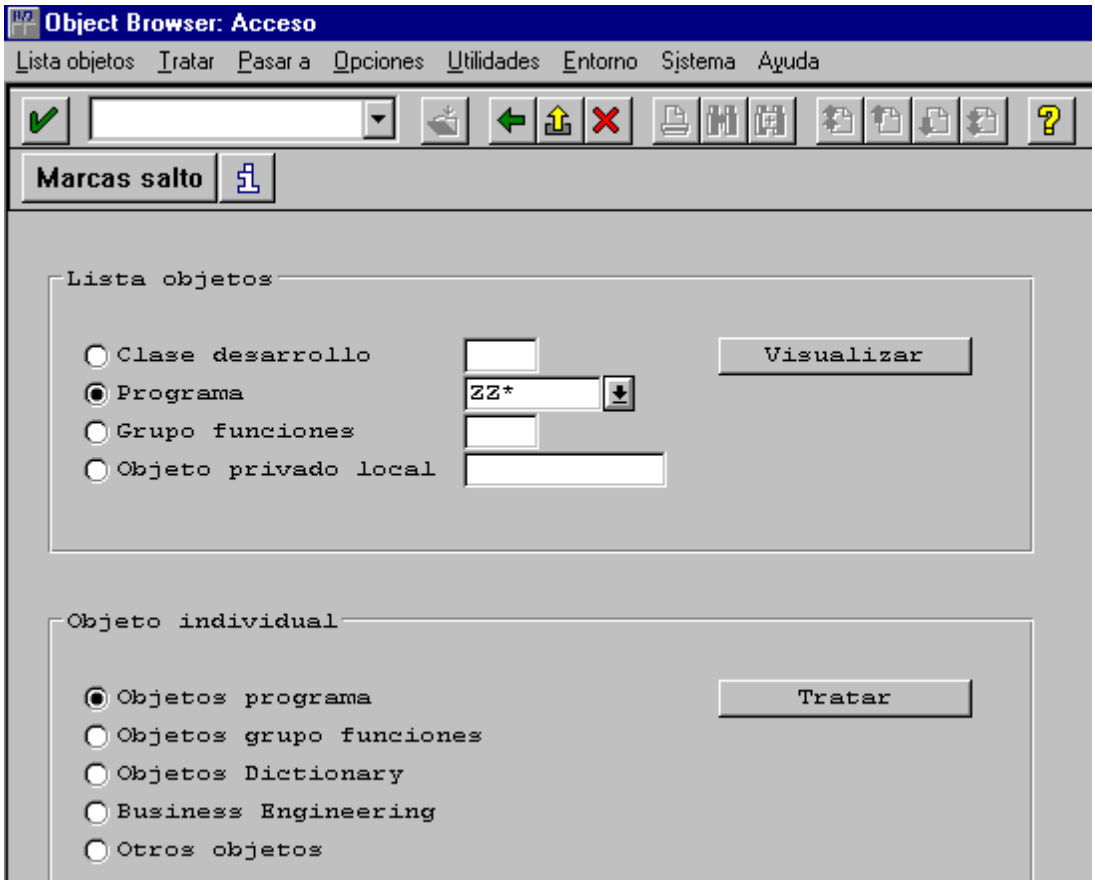
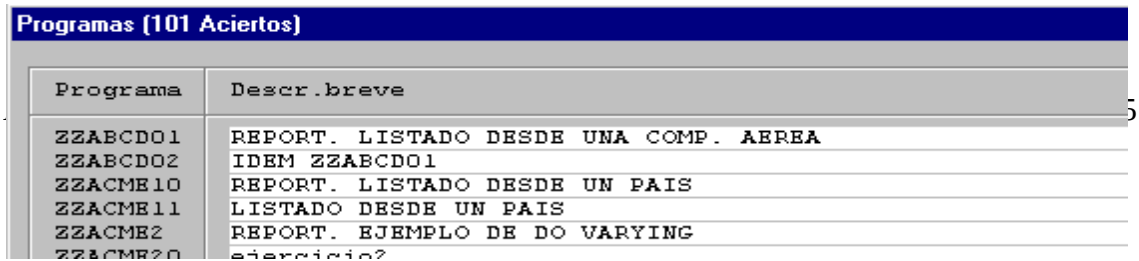



Fig. Object Browser.

En “Lista de objetos” podemos visualizar las clases de desarrollos, programa, grupo funciones, objeto privado local.

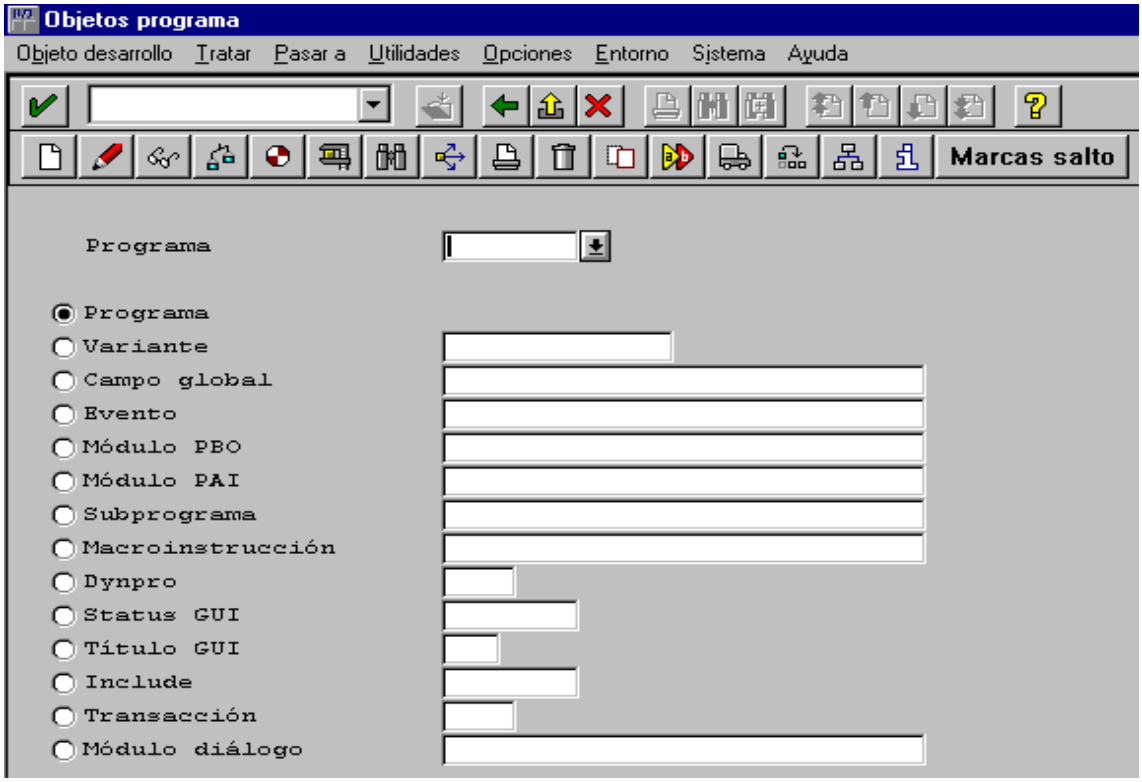
En “Objecto individual” ya podemos crear, borrar, modifcar, etc los Programa, Grupo de funciones, Dictionary, Engineering y otros objetos.

Como ejemplo de “lista de objetos”, listaré los programas que empiezan por zz, para hacerlo escribiremos en el campo “programa” “zz*” y pulsaremos F4 o al matchcode y nos saldrá la siguiente lista de programas:



Ahora seleccionaremos el programa “ZZACME20” haciendo doble clic o un clic y pulsando el botón de confirmar cuando lo hagamos en la pantalla del Object browser (Véase Fig. Objct browser) nos pondrá el nombre del programa y si pulsamos el botón “Visualizar” nos saldrá el programa.

Como ejemplo de “objeto individual” mostraré que podemos hacer en “programa”, para hacerlo seleccionamos el radiobutton “programa” y pulsamos el botón “tratar” y nos saldrá esta pantalla:



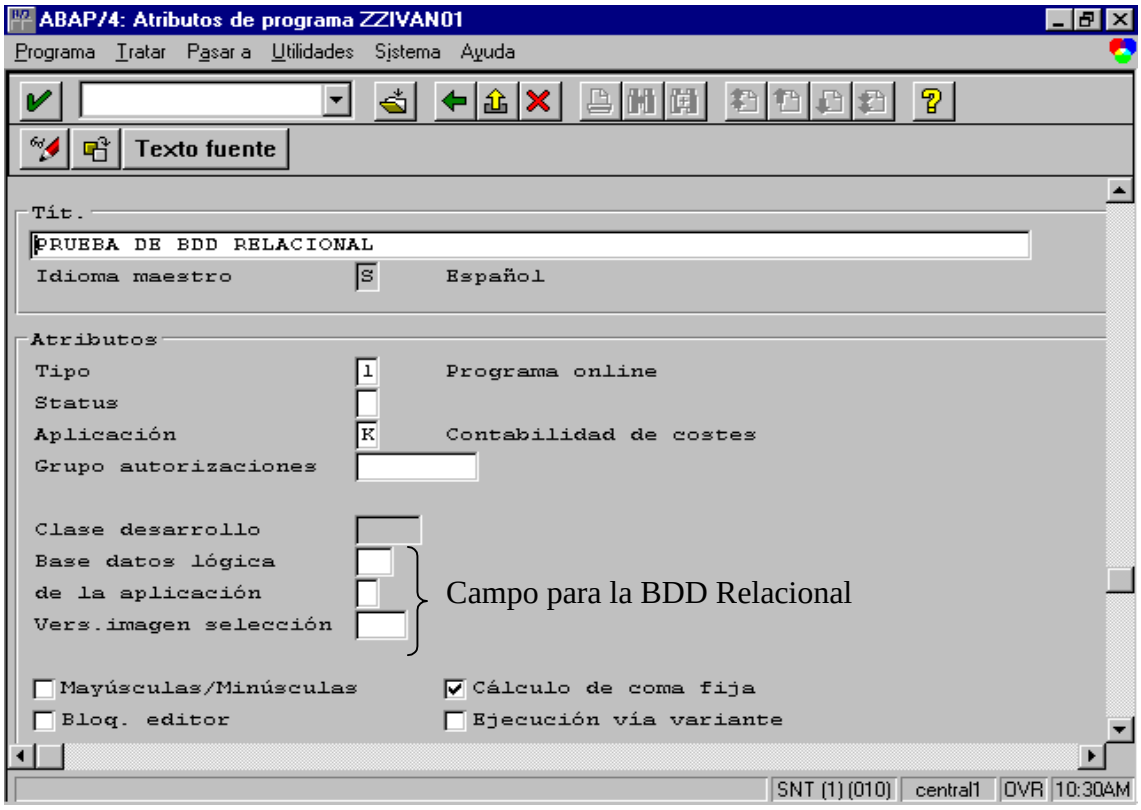

Como vemos podremos hacer cualquiera con los objetos relacionados con el programa, si queremos buscar algún objeto lo tenemos que hacer con el matchcode.

## BASES DE DATOS RELACIONALES

Las BDD Relacionales son un conjunto de tablas relacionadas por un campo común a todas ellas.
Antes hemos visto como relacionarlas (a través de claves foráneas), ahora aprenderemos como utilizarlas en un programa.

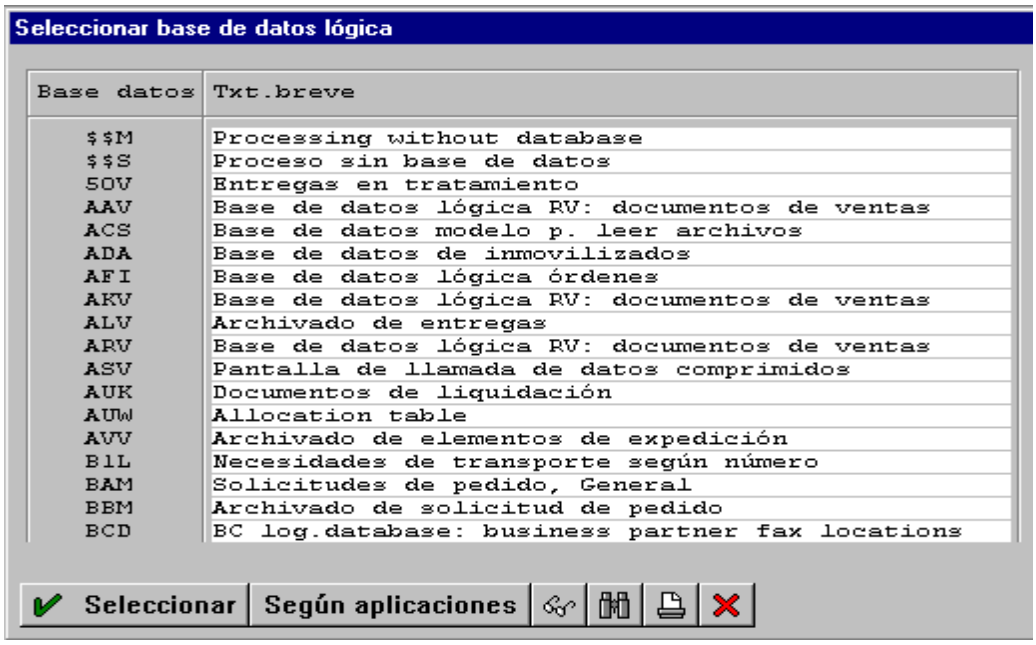
Para utilizar una BDD Relacional, primero hemos de indicarle al programa que vamos a utilizar una BDD Relacional.

Para indicarle al programa que vamos a utilizar una BDD relacional, tenemos que ir a los atributos del programa, e introducir una serie de datos como en la siguiente imagen:

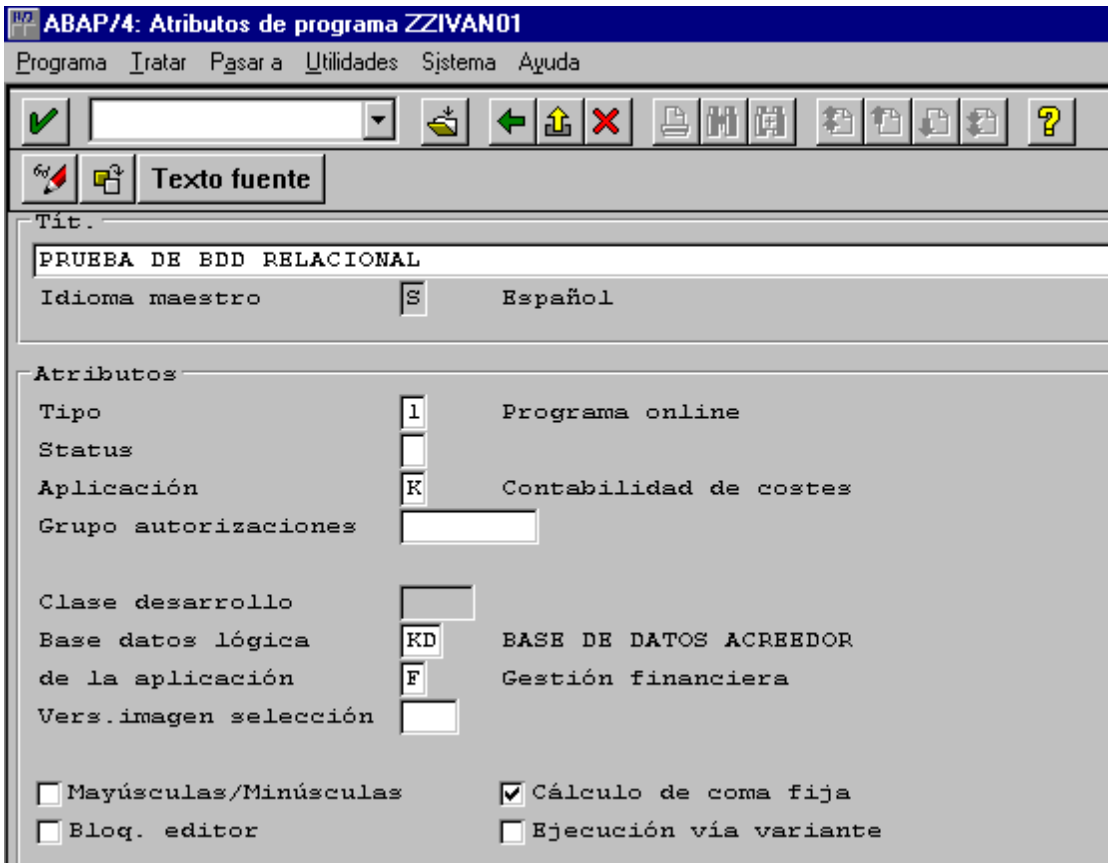



Como vemos en "aplicación" le pondremos el valor K, si queremos ver los que hay utilizaremos el matchcode o pulsando F4. Cuando hallamos seleccionado la "aplicación" pulsaremos "ENTER" y nos saldrán 3 campos para la BDD Relacional.

Para seleccionar la BDD, nos posicionaremos el campo "Base datos lógica" y pulsaremos F4 o al matchcode y nos saldrá una ventana con las BDD Relacionales que tiene SAP. Como en la siguiente imagen:



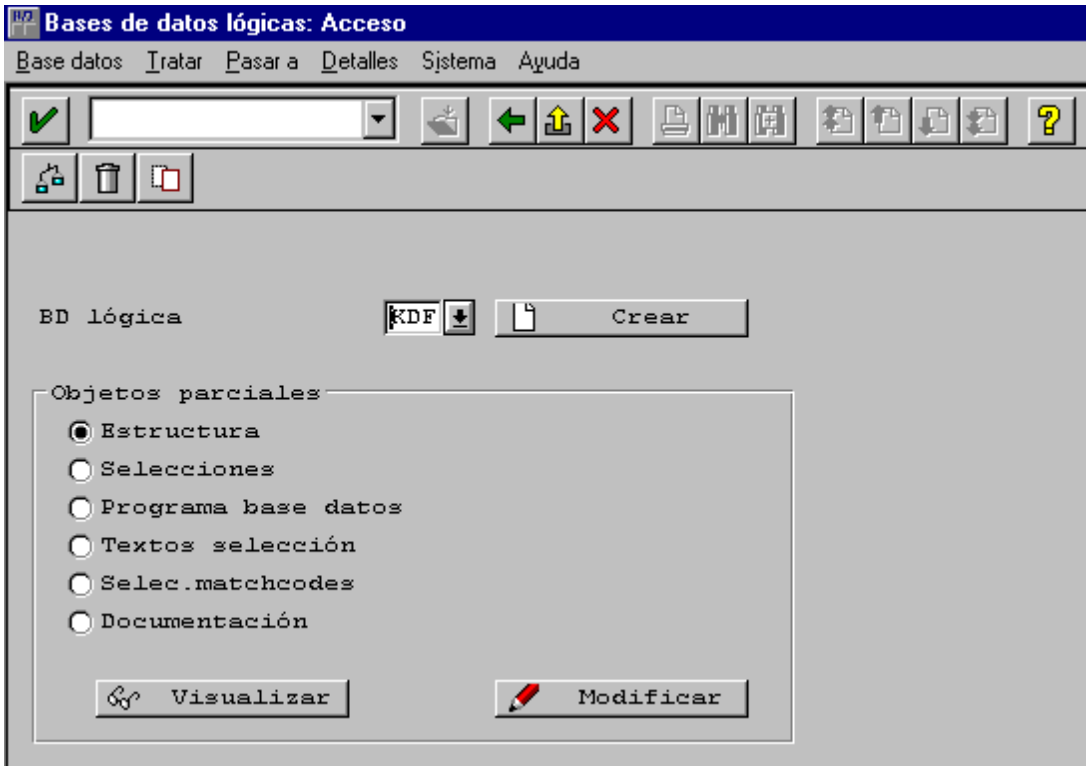

En nuestro caso seleccionaremos la BDD “KDF”, después de seleccionada nos volverá a la pantalla de “atributos” y otra vez pulsaremos “ENTER”, y nos tiene que aparecer esta pantalla:




Después lo grabaremos y ya podremos utilizar nuestra BDD Relacionar en nuestro programa

## VER LA ESTRUCTURA DE UNA BDD RELACIONAL

Para ver la estructura de una BDD Relacional, o sea, ver las que tablas están relacionadas entre sí. Tenemos que ir a la pantalla de "ABAP/4 Development Workbench" y ahí al menú "herramientas", "entorno de programación", "bases de datos lógicas" y nos saldrá la siguiente pantalla:




En el campo "BD lógica" escribiremos el nombre de la BDD Relacional, si no sabemos el nombre podemos buscarlos a través del matchcode o pulsando F4. En nuestro caso la BDD sería KDF a continuación pulsamos visualizar para ver su estructura y nos aparecerá la siguiente imagen:

En esta pantalla salen todas las tablas que están relacionadas entre sí.

Por cada registro de LFA1 habrá 1 o más registros a LFB1 y por cada LFB1 habrá 1 o más registros a LFC1, esto es una estructura jerárquica. Como vemos en la pantalla anterior.

Las lecturas de estas tablas se hace con el GET.

En el programa la parte de tablas, en este caso, se habrán de definir las 3 tablas que forman la parte de BDD lógica que usamos.

Un ejemplo de codificación de lectura:

```
START-OF-SELECTION
        GET LFA1.
        CHECK.      "Selección (opcional, no leer todos)
        GET LFB1.
GET LFC1.   " Opcional: GET lfb1 late, coge él ultimo registro de nivel superior
            porque lo continua leyendo.
END-OF-SELECTION.
```

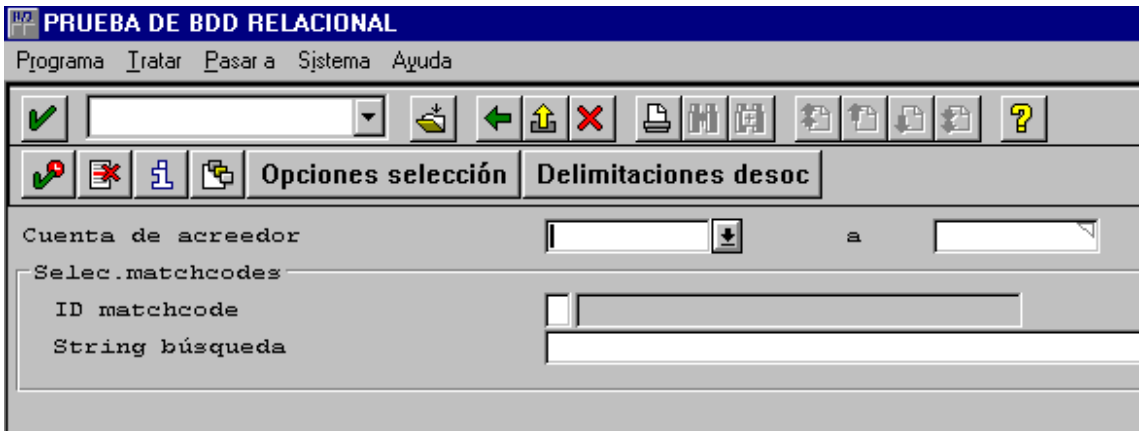
Esto ahorra llenar las llaves de cada acceso del GET.
Hay que tener mucho ¡CUIDADO! Que si los GETS no se ponen por orden de jerarquía, no peta pero hace lee de forma incorrecta.

El hecho de definir una bases de datos lógica los atributos, hace que cuando se ejecute el PGM, nos pida los inputs de todos los campos de clave en las tablas ligadas a la BDD. Lógica.

En el ejemplo anterior he creado el programa "ZZIVAN01" y en los atributos le he puesto que lea la BDD lógica "KDF", ahora en el texto fuente escribiré lo siguiente:

```
REPORT ZZIVAN01
TABLES:  LFA1.
GET LFA1.
```

Si lo ejecutamos nos saldrá la siguiente pantalla:




Como vemos SAP ya nos lo hace todo.

## EDITOR ABAP/4

Este es el editor de programas, se activa pulsando el botón de Editor ABAP/4 en la pantalla principal de ABAP/4.

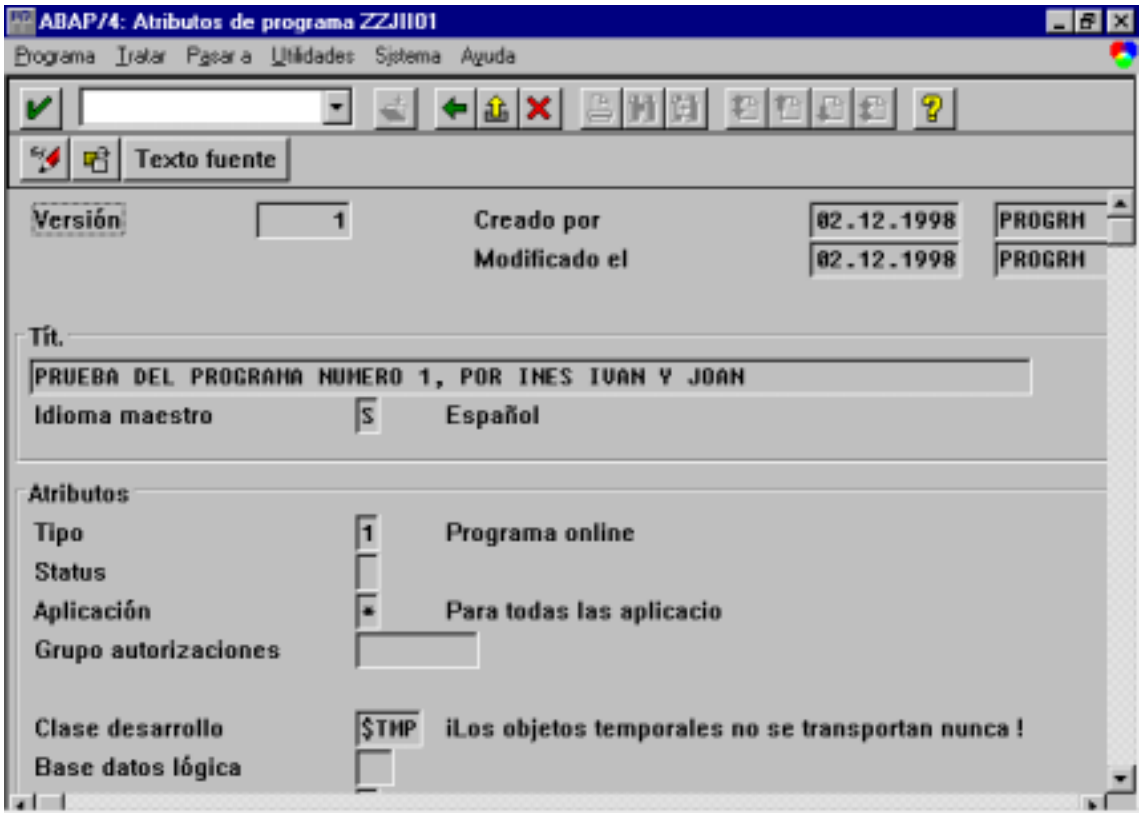

Fig. ABAP/4 Acceso.

En “programa” pondremos el programa que queramos crear, modificar o visualizar.

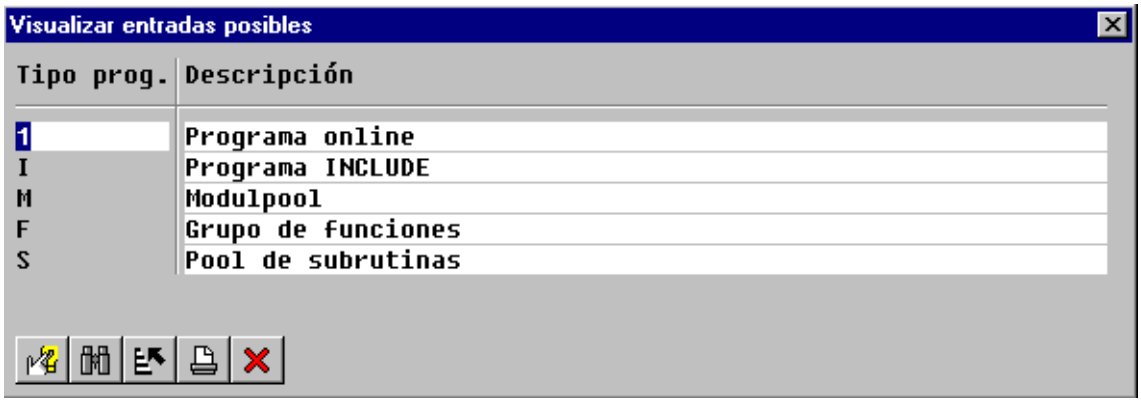
En “objetos parciales” están los tipos de objetos que tiene un programa:

**ATRIBUTOS**

En “atributos” sale la información referente al programa, la pantalla sería la siguiente:




En “Tipo” podremos cambiar el tipo de programa.
Si nos posicionamos en el campo y pulsando F4 o el botón que saldrá a la derecha, nos saldrá el tipo de programas, en esta pantalla:




Los de tipo Report sé utilizan para realizar informes, presentaciones, entrada de datos, etc.

Los de tipo Modulpool son aquellos en que antes de ejecutarse una pantalla se ejecuta un módulo (llamado PBO->Program before output) y después de ejecutarse, cuando confirmemos un dato o pulsemos un botón se ejecuta otro módulo (llamado PAI->program after input). Este tipo de programa puede llamar a otro programa del mismo tipo que tenga PBO y PAI.

En “Aplicación” ponemos los usuarios que podrán ejecutar el programa. Si hay un ‘*’ todos los usuarios lo podrán ejecutar. Si queremos ver los grupos de autorización pulsamos F4 y nos saldrá la pantalla con todos los grupos de autorización.

En “Clase de desarrollo” pondremos la clase, que dependiendo de la empresa en la que estemos será una u otra. *Hay que recordar que las clases que empiezan por el signo ‘$’ son objetos locales (aunque no la grabemos como un objeto local), no se pueden transportar.*
Cuando ya hemos puesto una clase de desarrollo y más tarde la queramos cambiar, veremos que en un principio el campo donde está la clase está desactivada, así pues para poder cambiarla vamos al menú “Programa”, “reasignar” y nos saldrá la siguiente pantalla:


Cuando creamos por primera vez el programa y le damos a cualquier botón, nos saldrá una pantalla en la que nos pedirá si queremos grabar el programa (si pulsamos el botón de grabar directamente ya nos saldrá esta pantalla), si le decimos que sí, nos saldrá esta pantalla:

Si pulsamos el botón que pone ‘ Objeto local’ nos grabará el programa de forma local y por lo tanto no se podrá transportar a producción. Si pulsamos el botón de grabar nos lo grabará con la clase de desarrollo que hemos puesto. Si la clase que hemos puesto no empieza por ‘$’ (Objeto local) nos saldrá la siguiente pantalla:

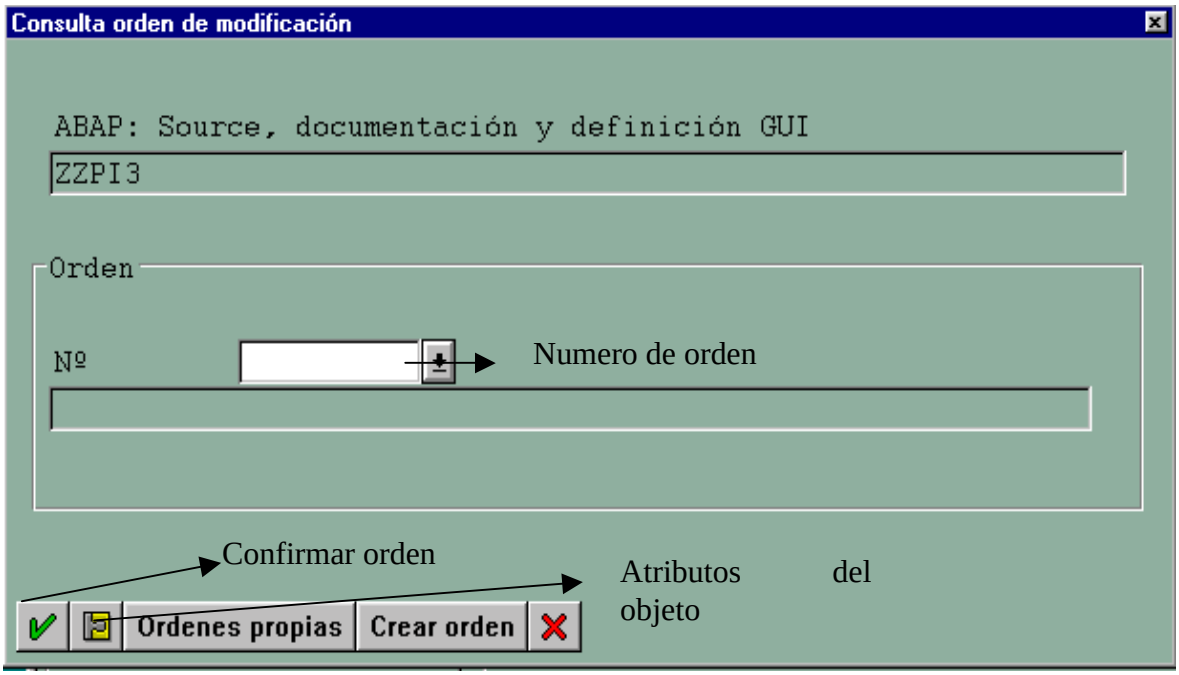
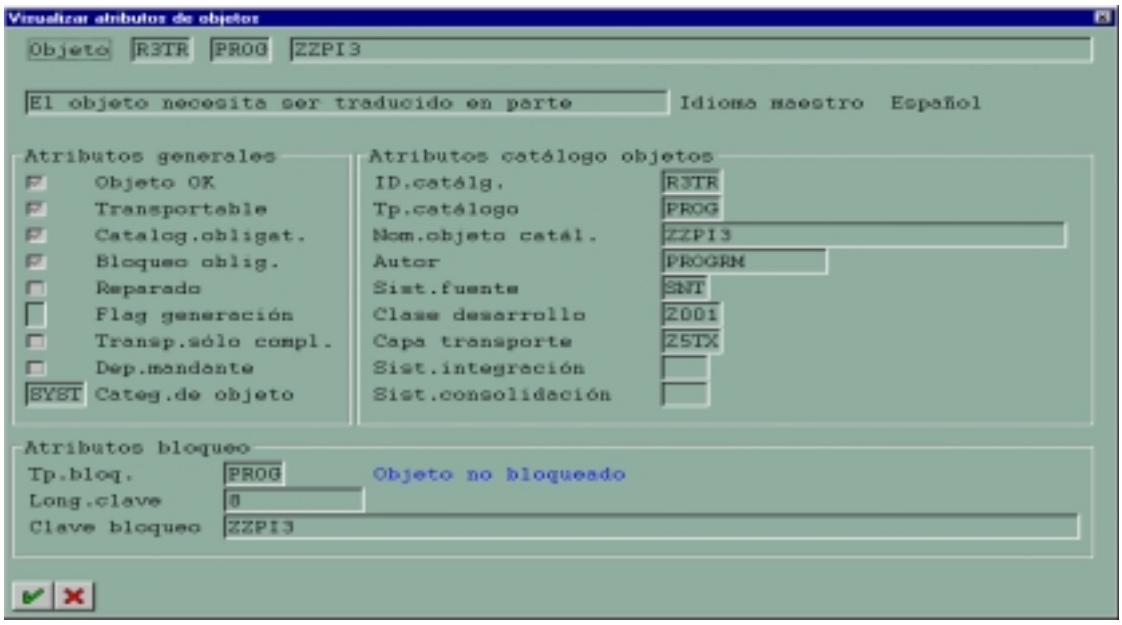



Fig. Orden.

Donde pone Nº de orden le pondremos la orden de transporte.

También podremos crear una orden nueva (siempre y cuando tengamos autorización) o mirar las órdenes que hay con el botón: ‘Ordenes propias’. Cuando hayamos puesto la orden que queramos pulsamos al botón ‘Ordenes propias’.

Si le damos al icono del puzzle nos saldrá una pantalla con los atributos del objeto, la pantalla que sale es la siguiente:

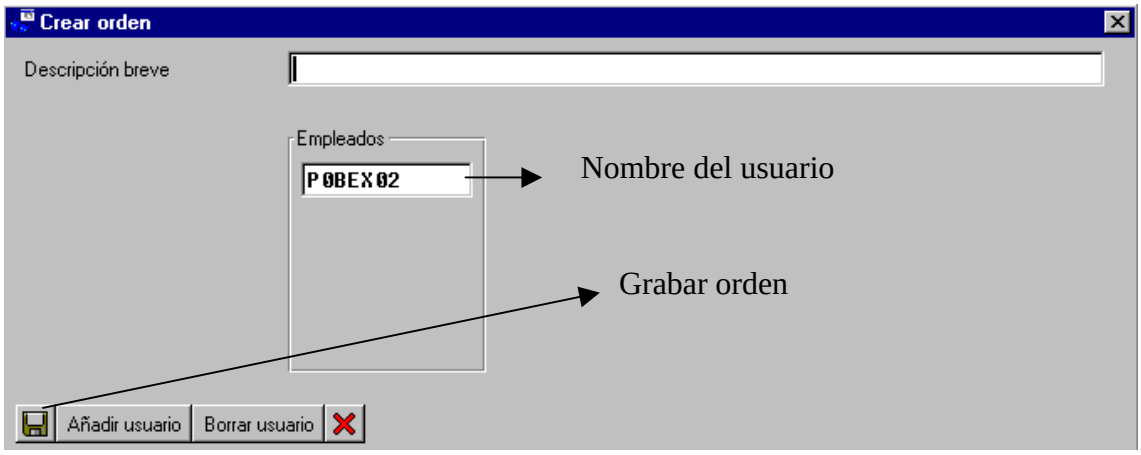


Cuando hayamos introducido la orden, pulsamos el botón de confirmar orden.

El transporte del objeto que hemos creado se explicará más adelante.

## COMO CREAR UNA ORDEN DE TRANSPORTE

Para crear una orden de transporte hemos de estar en la pantalla “Consulta orden modificación” (Véase Fig. Orden) y desde ahí hemos de pulsar el botón “crear orden” y nos saldrá la siguiente pantalla:




En descripción escribiremos un texto breve, por ejemplo, en BASF siempre se pone al principio “E_030_ FI nombre_objeto” donde nombre de objeto es el nombre de una tabla, ABAP, etc. es muy recomendable utilizar una orden de transporte por cada objeto que tengamos creado, ya que es más claro a la hora de transportarlo.

Cuando lo hayamos creado pulsamos el botón de grabar y nos volverá a la pantalla “Consulta orden modificación” (Véase Fig. Orden) ahí confirmamos la orden y podremos transportar el objeto.

Como transportarlo se verá más adelante.

## VARIANTES

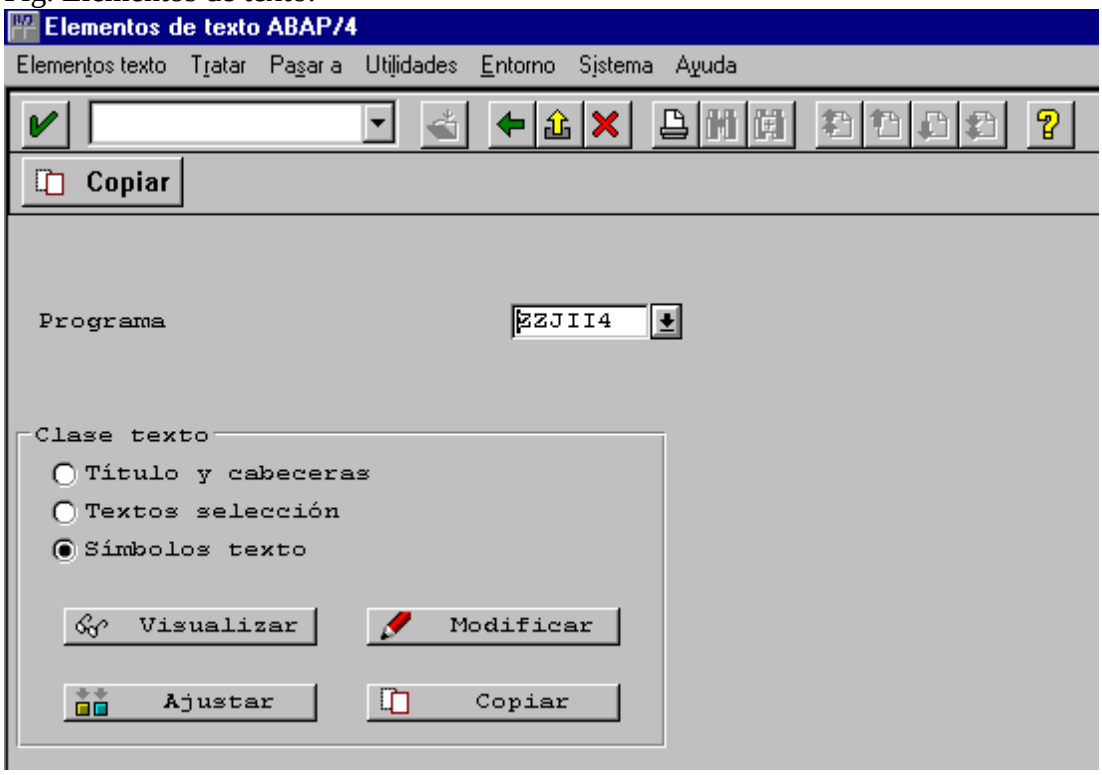
Volviendo a la pantalla del editor ABAP/4 tenemos variantes. Cuando en un programa queremos que se introduzcan unos valores predeterminados y no los que el usuario quiera, se utilizarán variantes. Las variantes tendrán los valores que se pueden introducir. Más adelante veremos un ejemplo completo de variantes.

## DOCUMENTACION

Después está la documentación que como su nombre indica se refiere a la documentación del programa.

## ELEMENTOS DE TEXTO

Los elementos de texto son textos que podemos incluir en el programa de una forma abreviada. Para acceder a ellos desde la pantalla principal (Véase Fig. ABAP/4 Acceso.) pulsamos sobre el botón de elementos de textos. También lo podemos hacer desde el editor de programas, yendo al menú "Pasar a", "elementos de texto". Si accedemos por esos dos sitios o por alguno más (hay alguno más) sale la siguiente pantalla:
Fig. Elementos de texto.




## SIMBOLOS DE TEXTO

Las líneas del editor de programas tienen una anchura determinada, por ello para visualizar mensajes largos o mensajes que queramos repetir varias veces utilizamos los símbolos de texto.
Los símbolos de texto se pueden crear cuando escribimos el programa o en la pantalla principal pulsando el botón de elementos de texto y escribiendo el nombre del programa.
Si estamos escribiendo el programa, para crearlo o modificarlo escribiríamos TEXT-001 (donde 001 es él numero del elemento de texto asociado al texto) y si sobre TEXT hacemos doble clic nos saldrá la pantalla de creación o modificación (sí ya existiera) de un elemento de texto. Un ejemplo sería:

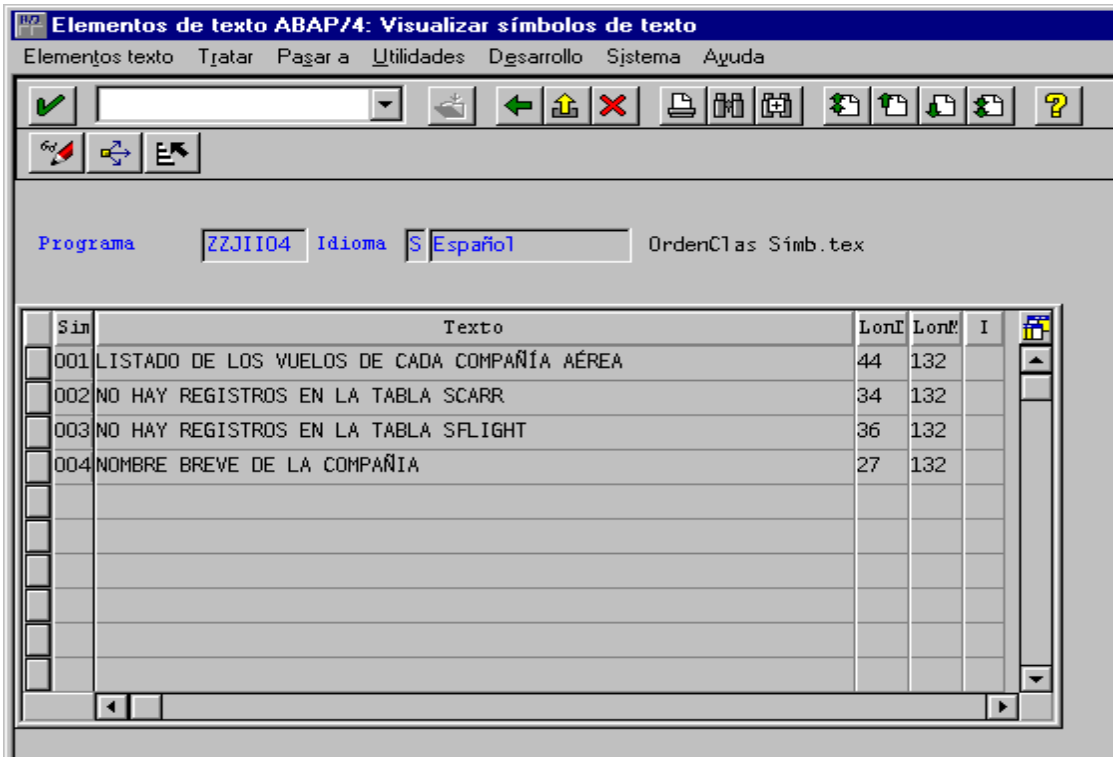
```
WRITE: / TEXT-001 CENTERED
```

Nº de elemento de texto

Escribiría lo que valiera TEXT-001 centrado y después haría un salto de línea.

Desde el editor de programas también se puede hacer a través de los menús “Pasar a”, “elementos de texto” o también desde la pantalla de elementos de datos (Véase Fig. Elementos de datos) seleccionando símbolos de texto.

La pantalla que saldrá cuando lo creemos o modifiquemos (no importa desde donde lo hagamos) será la siguiente:

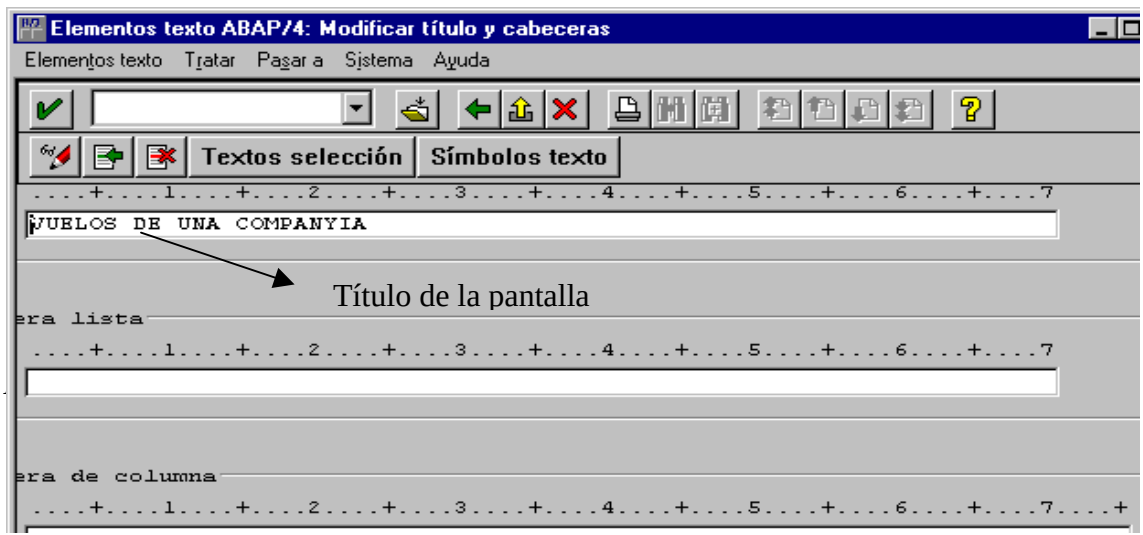



*No hay que olvidarse de grabarlo para poder asociarlo al programa.*

Los elementos de texto son muy útiles ya que nos permiten escribir textos largos sin importarnos la anchura del editor de programas y también cuando queramos traducir nuestro programa a otros idiomas, ya que sólo se traducen los elementos de texto.

## TITULOS/CABECERAS

Volviendo al editor ABAP/4, tenemos Títulos/Cabeceras. Es útil cuando hagamos listados por impresora o por pantalla. Nos saldrán las cabeceras, títulos, etc. que nosotros queramos. La pantalla de Títulos/Cabeceras es la siguiente (En la página siguiente):



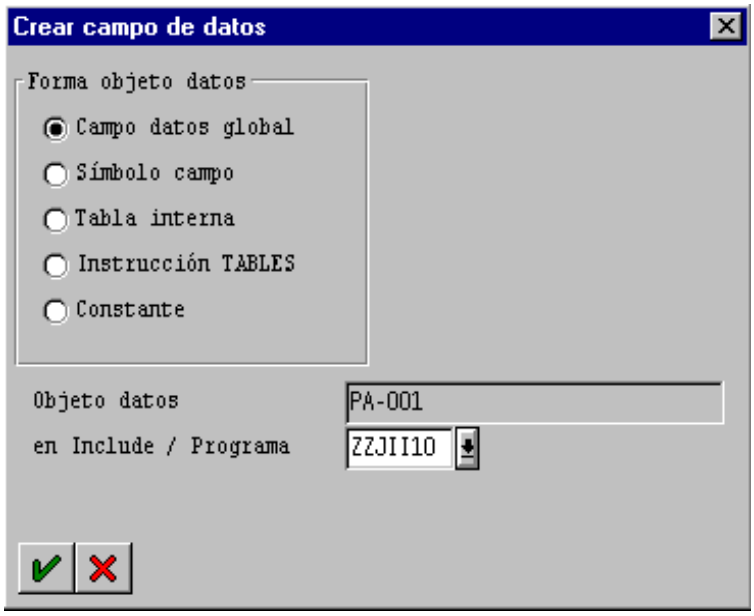
Recordar que si al principio del programa no ponemos la siguiente línea en el Report, 'NO STANDARD PAGE HEADING', no nos saldrán los títulos y cabeceras que hayamos puesto. Esta parte se explicará más adelante.

## TEXTO DE SELECCION

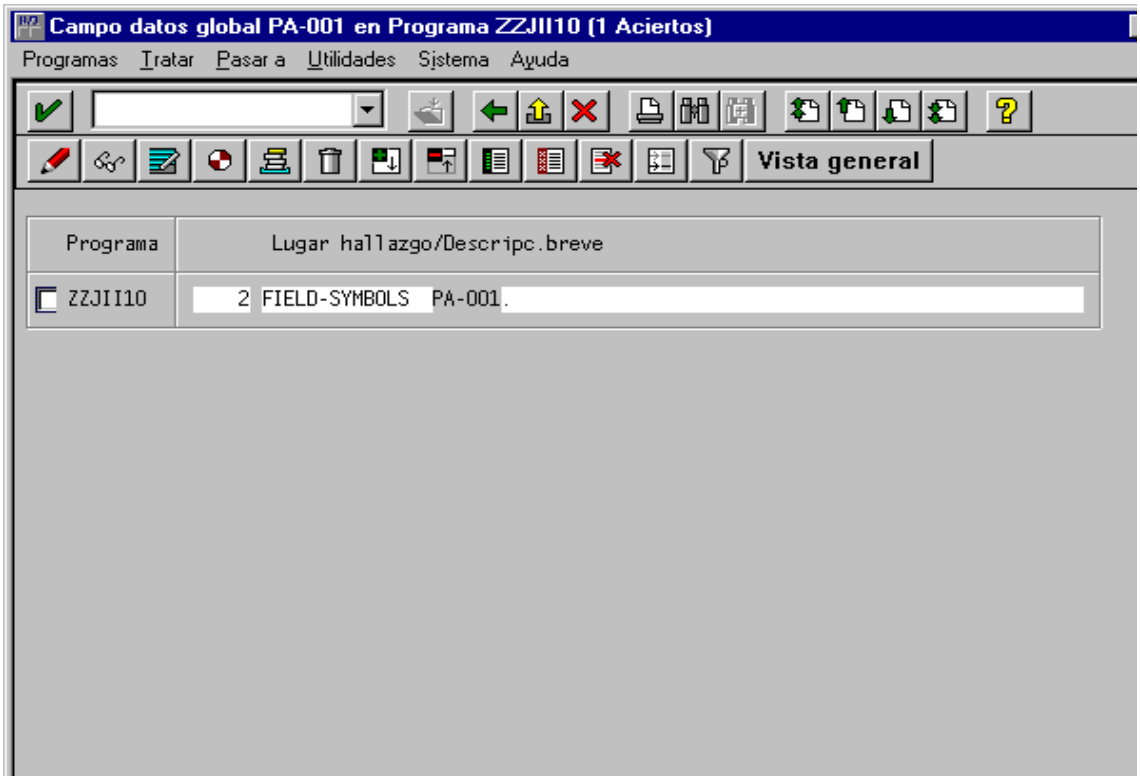
El texto de selección es lo mismo que los elementos de texto pero en este caso referido a campos de un programa.
Se pueden crear o modificar desde cualquier punto del editor ABAP/4, escribiendo el nombre del programa y activando el push botton de texto selección, desde los menús

'Pasar a', 'Texto selección', pero la forma más cómoda es desde el editor de programas (de igual forma que en los elementos de texto).
Escribimos PA-001 (donde 001 será él número de texto selección asociado al texto) y hacemos doble clic en PA y ya podemos crearlo o modificarlo si ya existiera.

Si no existe nos saldrá una pantalla con los objetos que podemos crear:




Después de escoger el objeto nos saldrá la pantalla siguiente:



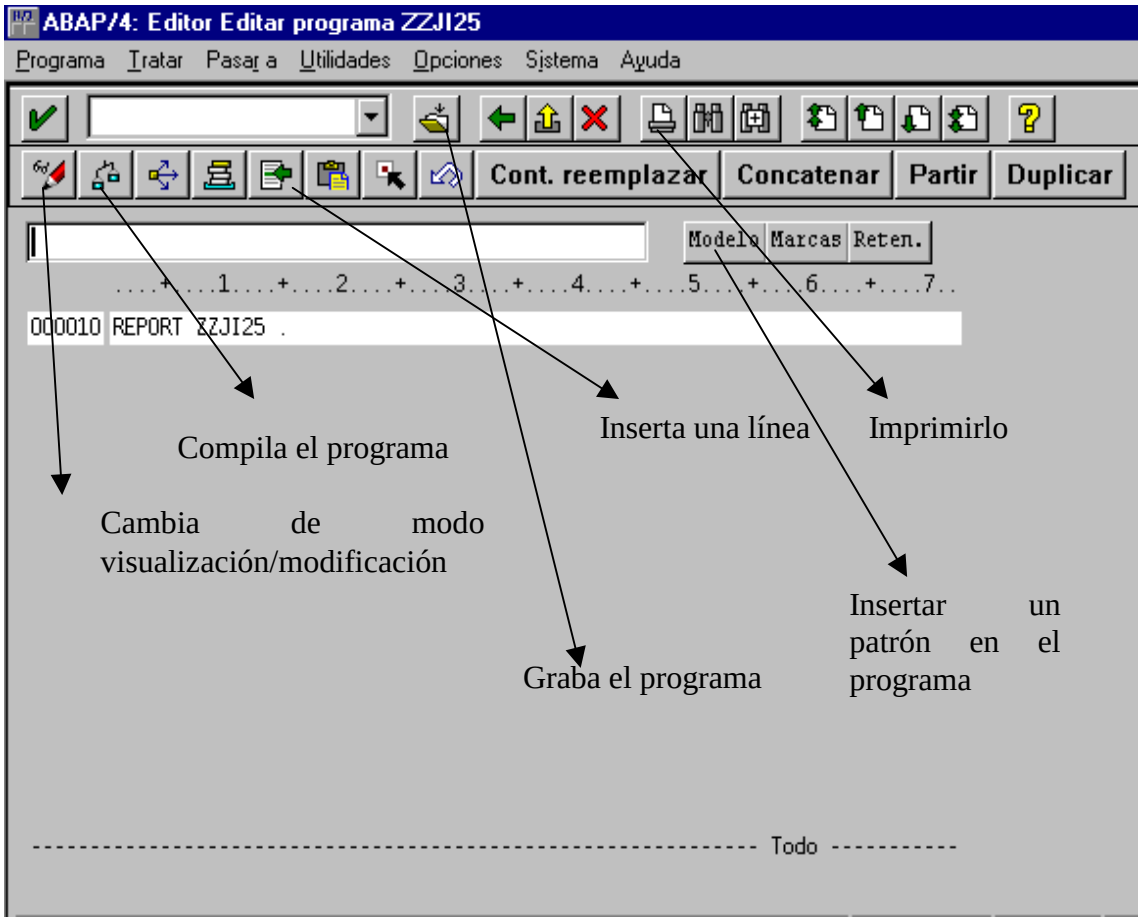

*No hay que olvidarse de grabarlo para poder asociarlo al programa.*

**TEXTO FUENTE**

Por último en el editor ABAP/4 está el texto fuente (yo lo llamo editor de programas), que es donde se escriben los programas en si, para crear un programa o modificar uno ya existente:

- Escribimos el nombre del programa en la etiqueta donde pone programa a crear o modificar
- Activamos el push button de texto fuente y le damos al botón de crear o modificar.

Después nos aparecerá la misma pantalla que aparece en "Atributos" donde nos pedirá el mensaje breve del programa, tipo de programa, aplicación. Cuando hayamos introducido estos datos lo grabaremos, y nos saldrá la pantalla común cuando se graba un objeto nuevo y después pulsaremos el botón de texto fuente que está en el menú painter y nos iremos al editor de programas.
Y nos aparecerá el editor de programas que es el siguiente:




Y ya podemos escribir nuestro programa. Como en cualquier lenguaje el programa se ha de compilar con este botón:

 → COMPILA UN PROGRAMA

O pulsando CTRL+F2.

Después se ha de ejecutar pulsando F8.

## MENUS DEL TEXTO FUENTE

A continuación explicaré algunas de las opciones más utilizadas o importantes:

- Dentro del menú “Utilidades”, “Upload/Download”, “Upload” sirve para insertar un fichero de texto que tengamos en el disco duro, diskette, cd rom, etc. al programa donde estemos.

- Dentro del menú “Utilidades”, “Upload/Download”, “Upload” Download graba el programa donde estemos a un diskette o al disco duro, lo graba en un fichero de tipo texto.
- Para renumerar las líneas de un programa, vamos al menú “Tratar”, “Otras funciones” y “Numerar”.
- Para deshacer algo que hemos hecho por error, pulsamos CTRL+F5.

## BOTONES DEL TEXTO FUENTE

En la parte derecha hay 3 botones juntos, que son: Modelo, Marcar, Reten.
El más útil de los 3 es el de Modelo.

### MODELO

Este botón permite insertar funciones, perform, select, etc. en el programa. Si lo pulsamos nos saldrá la pantalla siguiente:

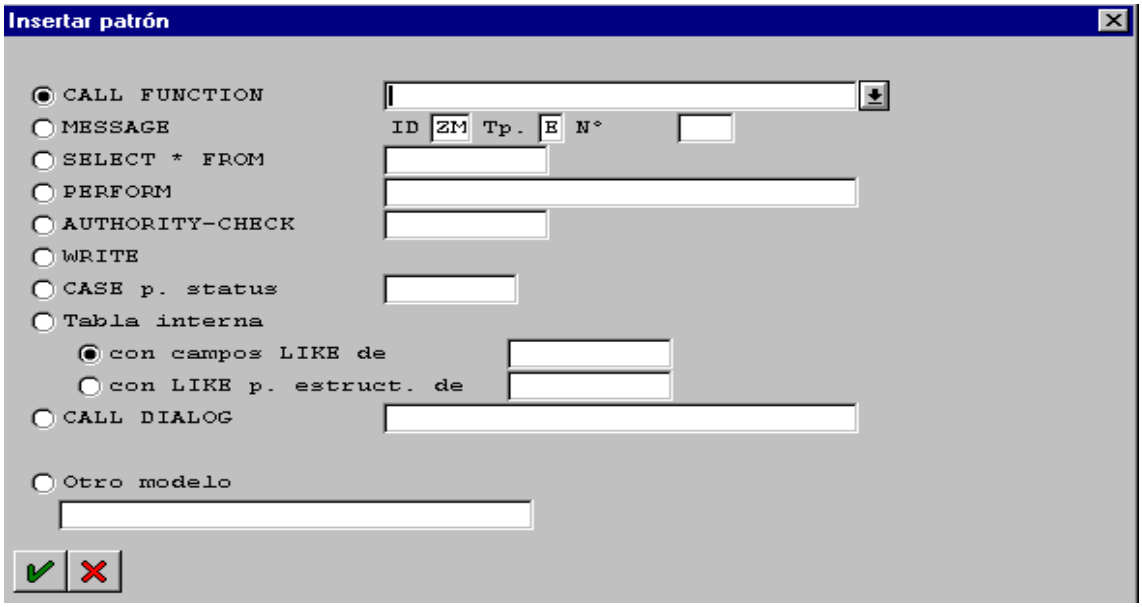
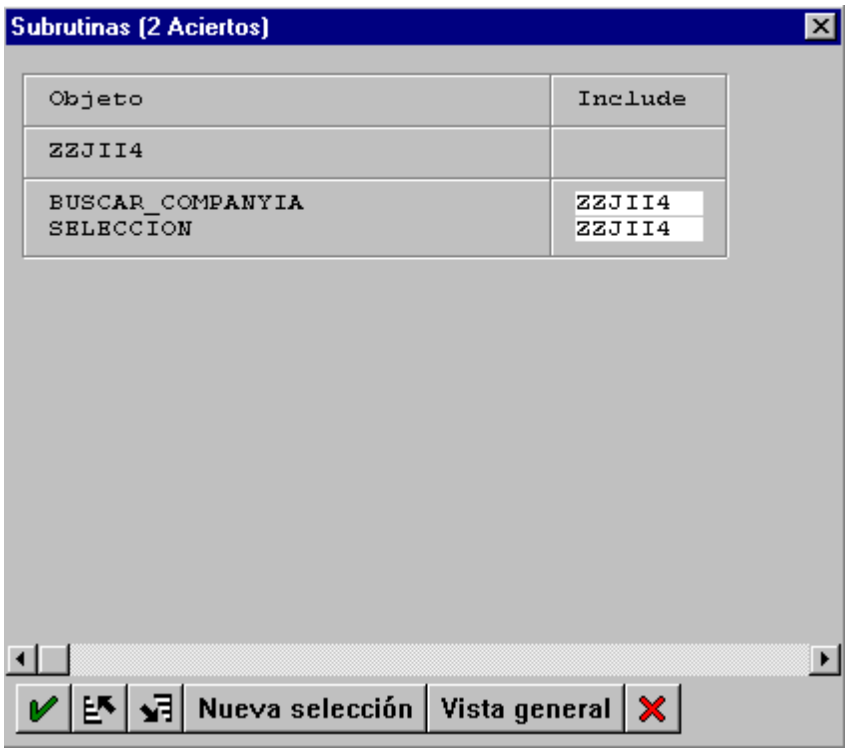


Fig. Modelo.

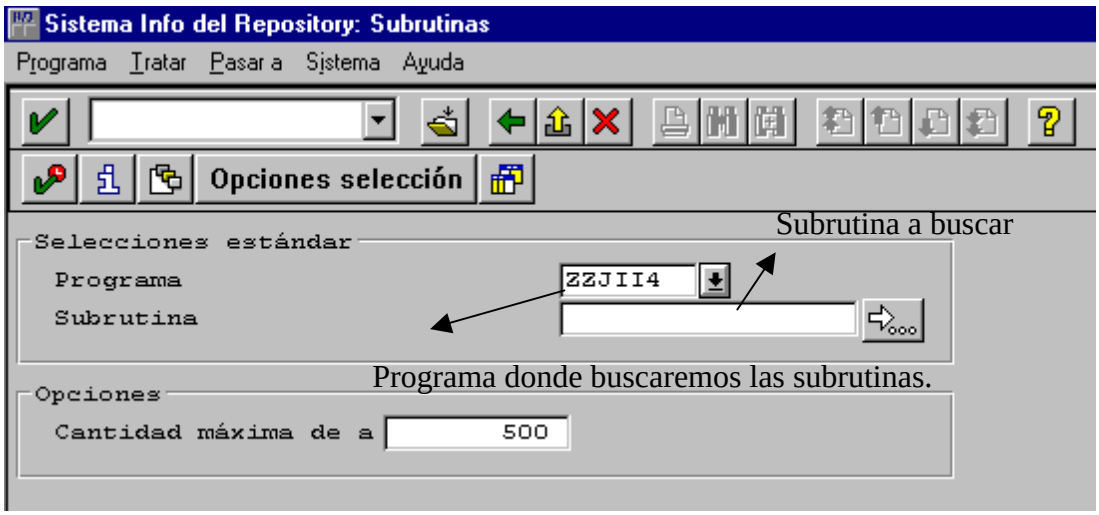
Como vemos podemos insertar casi cualquier cosa. Para ver como funciona vamos a insertar una subrutina externa al programa.

Para hacerlo primero hemos de posicionar el cursor en el campo que pone perform, después le damos al matchcode o pulsamos F4, entonces nos puede salir una pantalla u otra dependiendo si nuestro programa ya tiene alguna subrutina asociada. Si tiene alguna subrutina nos saldrá la siguiente pantalla:

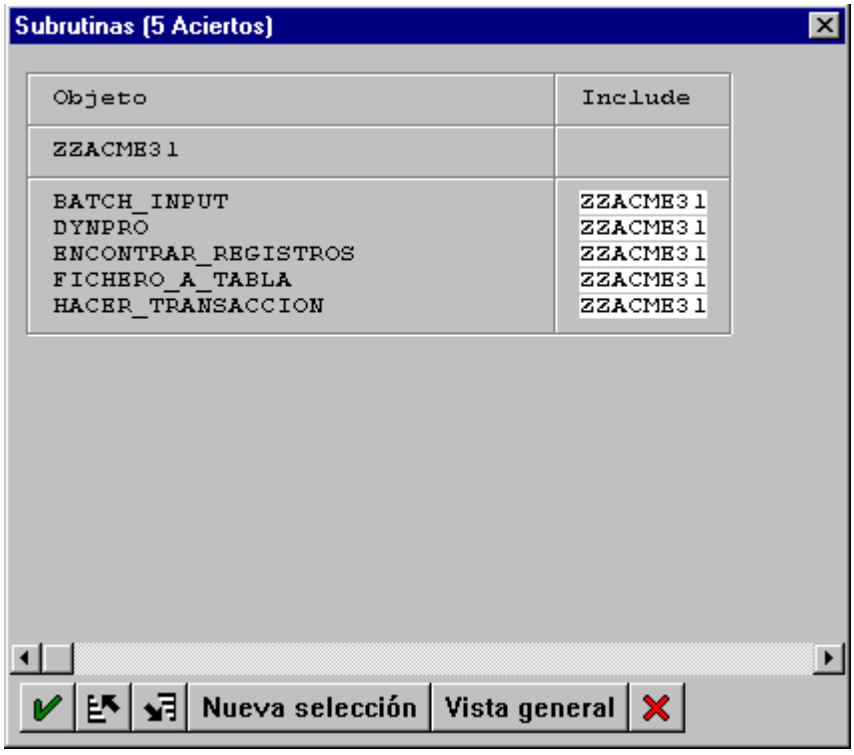
En esta pantalla nos saldrían las subrutinas que tiene nuestro programa, por si queremos insertar alguna en nuestro programa.



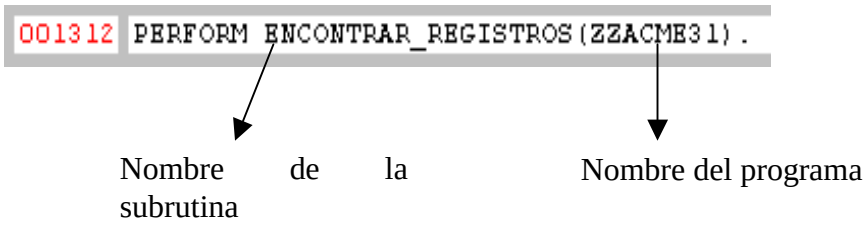

Ahora si le damos al botón "Nueva selección" nos saldrá la pantalla siguiente. Esta pantalla también saldrá cuando no tengamos ninguna subrutina en el programa.




En "programa" pondremos el nombre del programa donde queremos buscar las subrutinas (por defecto sale el programa donde estamos) y en qué subrutina queremos buscar. Ejemplo: yo en programa escribiré ZZACME31 (que sé que tiene subrutinas, vosotros poned un programa que también las tenga). En subrutina, como no sé como se llaman, ni cuantas hay pondré un '*' y pulsaré F8 (o el botón de búsqueda) y en mi caso me saldrá esta ventana con las subrutinas encontradas:

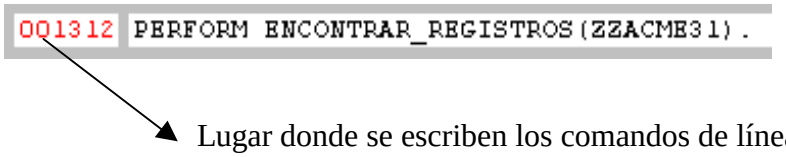



Ahora, haciendo clic en una subrutina y pulsando sobre el icono de confirmar (abajo a la izquierda) o directamente haciendo doble clic sobre la rutina deseada y en la pantalla de modelo (Véase Fig. Modelo) nos saldrá la subrutina escogida. Ahora le damos al botón (también abajo a la izquierda) y SAP nos insertará automáticamente la subrutina. En nuestro caso sería esta:




## COMANDOS DE LINEA

Los comandos de línea, como bien dice su nombre, se escriben en el número de línea de un programa.




Los comandos de línea se pueden dividir en: comandos de una sola línea, comandos de bloques de línea e insertar un trozo de programa en otro.

### COMANDOS DE UNA SOLA LÍNEA

Los comandos de una sola línea son los siguientes:

- J -> Juntar línea.
- In -> Insertar línea(s), donde “n” es el número de líneas a insertar.
- R -> Repetir línea.
- C -> Copiar línea.
- M -> Mover línea.
- D -> Borrar línea.
- S -> Partir línea.
- O -> Overwrite, machaca los espacios en blancos de la línea.
- * -> Posiciona la línea en la primera posición de la página, o doble clic del ratón.

**COMANDOS DE BLOQUE DE LÍNEA.**

Las instrucciones para manipular son parecidas a las anteriores. Explicaré como: mover, copiar y borrar un bloque de líneas.

MOVER

Para mover se utiliza dos veces la orden “MM”, una para indicarle el inicio del bloque y la segunda para indicar el final del bloque. Pulsaremos “ENTER”, cuando lo hagamos veremos como ese bloque cambia de color, para mover el bloque al lugar deseado, nos posicionamos con el curso en la línea que queramos movernos y ahí tenemos dos ordenes para colocarlo que son:

A -> Después de la línea donde estemos.
B -> Antes de la línea donde estemos.

COPIAR

Para copiar tenemos la orden “CC”, y como en la instrucción para mover, la primera va en la línea de comienzo del bloque y la segunda en la línea de final de bloque. Pulsaremos “ENTER” también cambia de color, y para colocar el bloque también tenemos la orden A (Antes) y B (Después) de igual funcionamiento que las explicadas anteriormente.

BORRAR

Para borrar es la orden “DD”, y como en las dos anteriores la primera es para el inicio de bloque y la segunda para el final de bloque. Cuando pulsemos “ENTER” se borrarán las líneas que tenga el bloque.

**INSERTAR UN TROZO DE PROGRAMA EN OTRO**

Para hacerlo tenemos la orden “XX”, que su utilización es la misma que copiar, mover o borrar un bloque, es decir, primero indicamos el inicio del bloque seguidamente indicaremos el final del bloque. Cuando pulsemos “ENTER” veremos como el bloque cambia de color, para colocar el bloque marcado solo tenemos que escribir la “X” en la

línea del programa que deseamos copiar el bloque, y el bloque se insertará una línea después de donde hayamos hecho la "X".

## COMANDOS GENERALES

Los comandos generales nos permiten compilar un programa, visualizar la estructura de una tabla, ayudas, etc.


Lugar donde se escriben los comandos generales

Tenemos las siguientes instrucciones:

- CHECK -> Verificar un programa.
- HELP instrucción -> Ayuda sobre la instrucción especificada.
- SHOW SY -> Enseña las variables del sistema.
- SHOW tabla -> Muestra la estructura de la tabla escrita.
- INSERT -> Inserta líneas.
- T ó TOP -> 1ª línea del programa.
- B ó BOTTOM -> Última línea de programa.
- IC instrucción -> Coloca la estructura en el lugar del cursor.
  Ejemplo: IC IF ó IC SELECT.
- REPLACE valor_inicial valor_final -> Sustituye los valores.
- PP -> Ó Pretty Printer estructura el programa.
- FIND palabra -> Busca la palabra.
- NEXT -> Continua la búsqueda realizada por el FIND.

## MENÚS DEL EDITOR ABAP/4

La pantalla del editor de ABAP/4 (véase fig. ABAP/4 Acceso) tiene unos menús que nos pueden ayudar a realizar las tareas más comunes.

En el menú "programa" podemos:
Crear F5 -> Un programa nuevo.
Modificar F6 -> Un programa ya existente.
Visualizar F7 -> Un programa ya existente.

También podemos verificar de forma simple o de una forma más compleja, generarlo (CTRL + F3) y ejecutarlo con las siguientes opciones:

- Forma directa F8.
- Fondo, debugging SHIFT+F5.
- Análisis de tiempo de ejecución, con variante SHIFT+F6 y de resumen de variante.

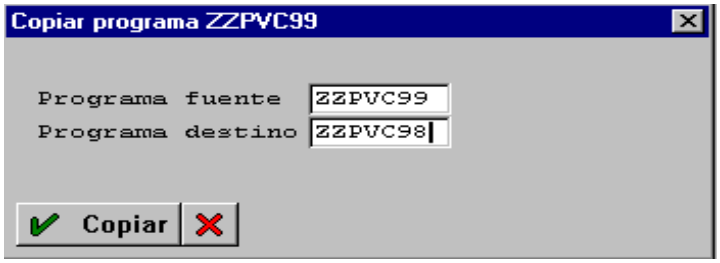
Imprimir SHIFT+F1 -> Imprimir un programa que tengamos.

Activar versión -> esto pone una versión al programa, por ejemplo el SAP que utiliza la empresa en que trabajo tiene la versión 3.1

Pretty Printer -> Nos estructura el programa de forma adecuada y además nos separa los procediemientos y módulos que tengamos en el programa y con la posibilidad de poder insertar un comentario, sobre lo que hace ese procedimiento o formulario.

Copiar CTRL+F5 -> Esta utilidad la utilicé una vez y me gustó bastante como funcionaba. Esta utilidad permite copiar un programa con otro nombre sin perder los objetos o sea dynpros, variantes, etc. que tenga el programa asociados.

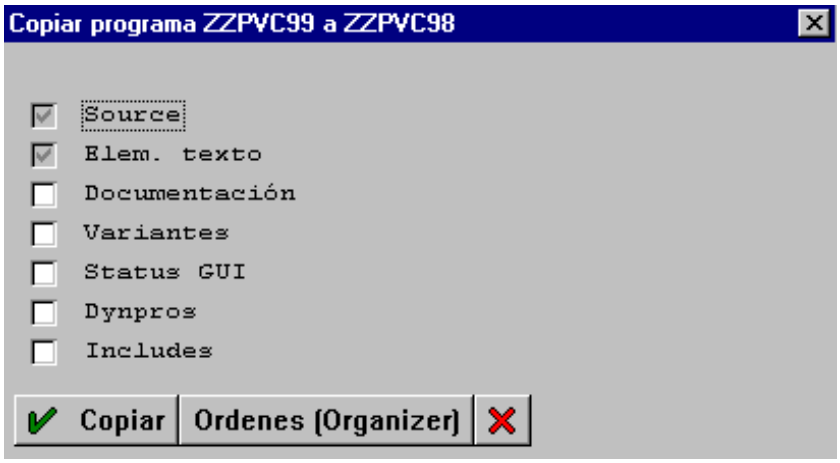
Si pulsamos Copiar nos saldrá esta pantalla:




Aquí, como vemos, indicaremos el programa de origen y el de destino.

Cuando pulsemos el botón de copiar nos saldrá una pantalla con los objetos que queremos copiar (dynpros, includes, variantes, etc.)

La pantalla que sale es la siguiente:

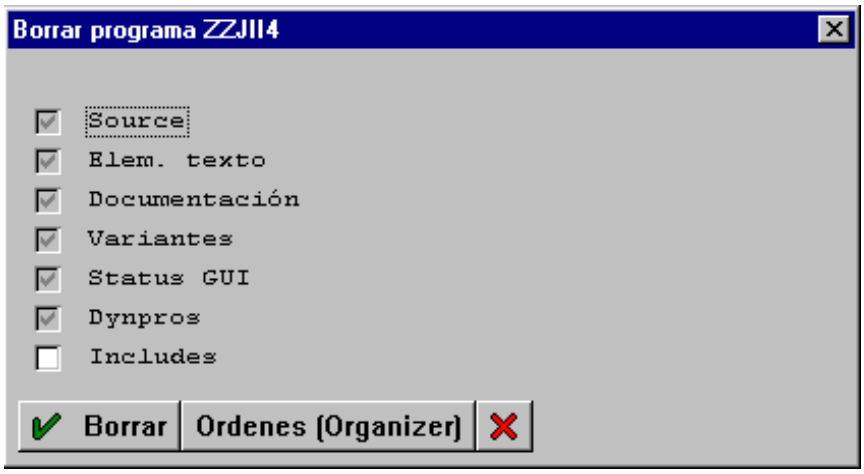



Cuando pulsemos el botón de copiar nos copiará en el programa destino todos los objetos que hayamos seleccionado, después el nuevo programa ya estará listo para ser utilizado.

Renombrar CTRL+F6 -> Como su nombre indica, sirve para cambiar el nombre de un programa.

Reasignar -> Sirve para cambiarle la clase de desarrollo, es decir, la clase que se utiliza para el transporte.

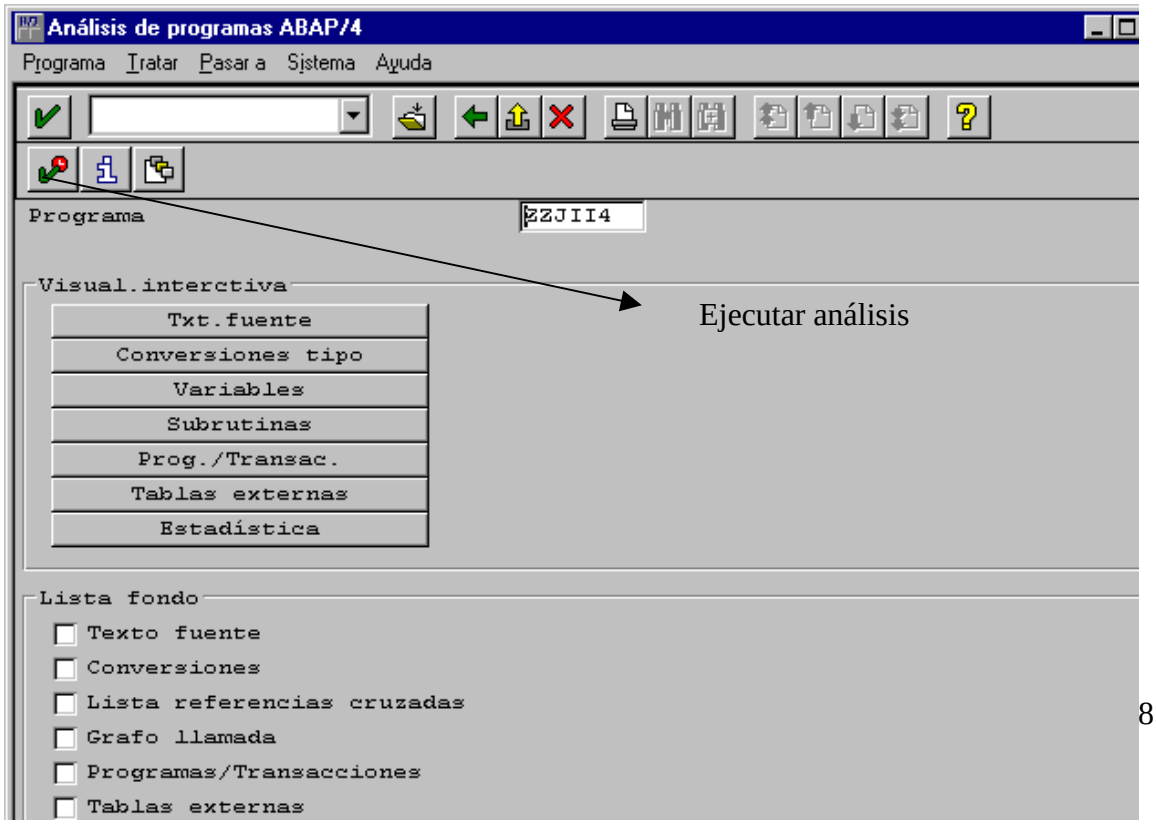
Borrar SHIFT+F -> Borra un programa con sus elementos de texto, documentación, includes, dynpros, etc. que estén relacionados con el programa. La pantalla que sale es la siguiente:




En el menú “utilidades” tenemos los siguientes submenús:

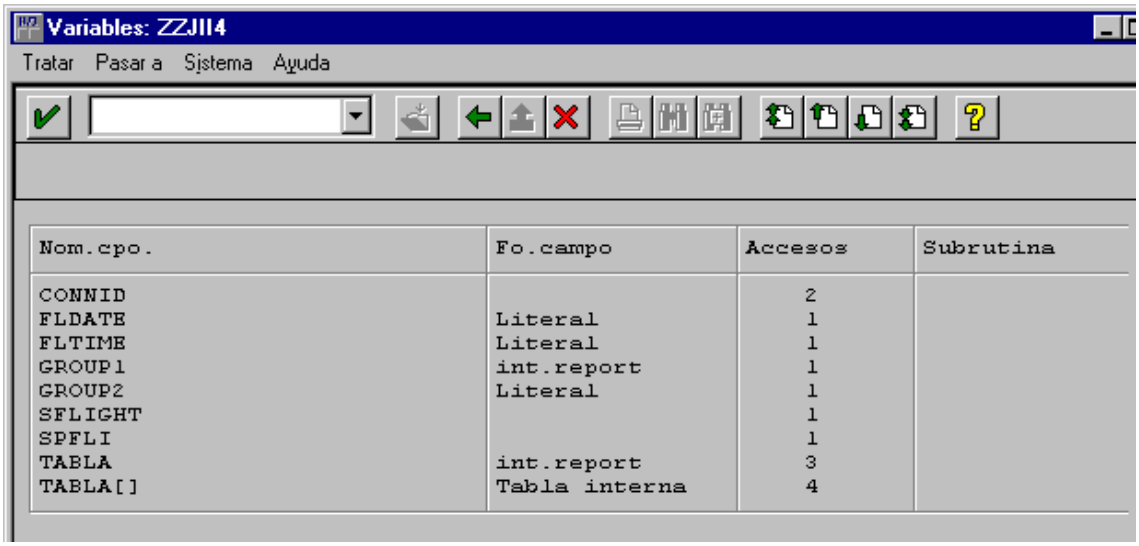
Buscar programa CTRL+SHIFT+F3
Buscar en programa fuente

Análisis de programa -> Desde esta opción podemos ver como tenemos hecho el programa: subrutinas, tablas externas, variables, etc.
La pantalla principal que sale es la siguiente:



Primero hemos de ejecutar el análisis para que nos analice el programa.

Si pulsamos sobre algún botón del box “Visual. Interactiva” nos sale la información que tiene nuestro programa referente al botón pulsado. Ejemplo: si pulsamos sobre variables, saldría la siguiente pantalla:




Como vemos, nos muestra las variables que se utilizan en el programa y las veces que se utilizan.
Editor split screen -> Sirve para comparar dos programas, tanto en código fuente, como en comportamiento.

## TRANSPORTE

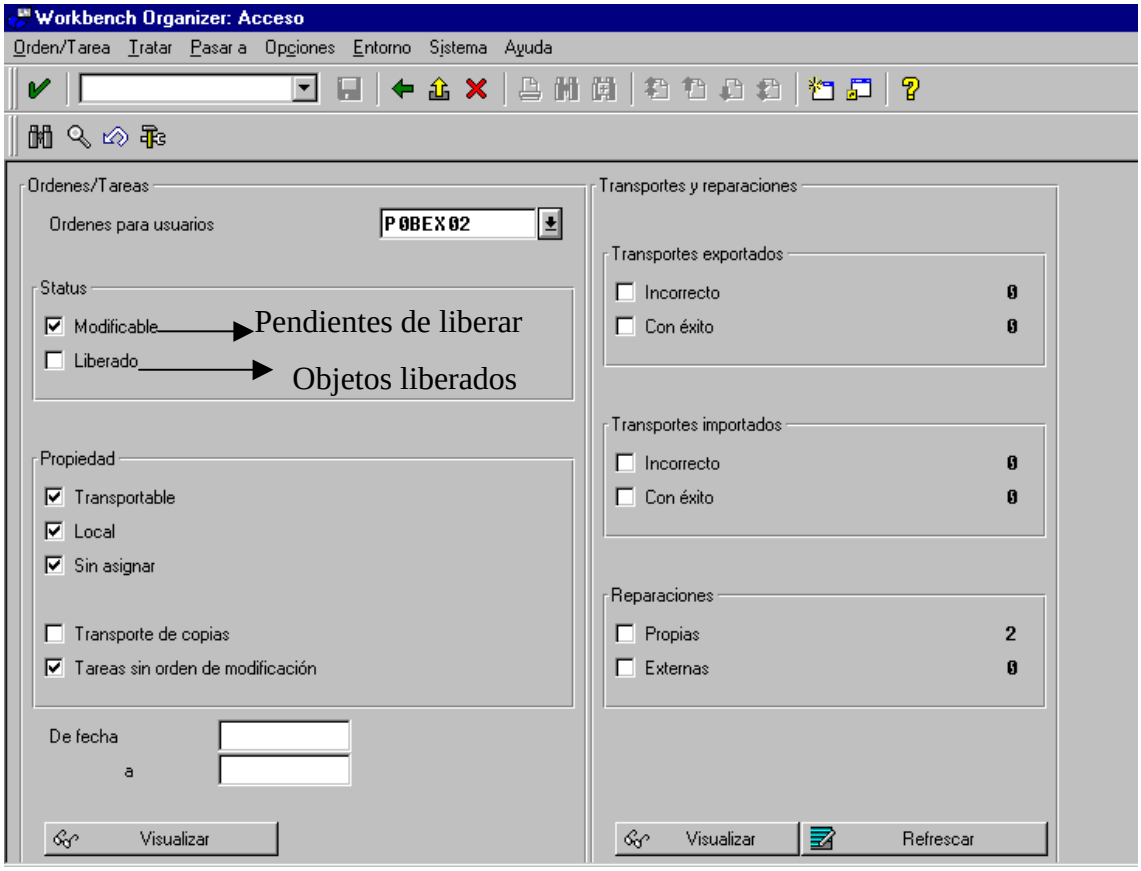
El transporte sirve para mandar los programas a producción, o los programas u objetos que tengan un grupo de función (explicado anteriormente).

Producción es la encargada de implantar el programa o proyecto que hayamos realizado.

La forma de hacer una clase de desarrollo ya está explicada en el capítulo anterior.

También he dicho que la clase de desarrollo nos la dará nuestro jefe de proyecto o en la empresa donde estemos.

Para poder enviar un objeto a transporte, desde la pantalla ABAP/4 Development Workbench menú "Resumen", "Workbench Organizer" y nos saldrá la siguiente pantalla:




En "Ordenes para usuarios" introduciremos el nombre del usuario. Si queremos ver los objetos liberar activaremos la casilla de "liberados", los liberados ya lo veremos más adelante.
Para ver los objetos que podemos transportar como los que no, pulsaremos el botón de confirmar o ENTER. Y nos saldrá la siguiente pantalla:

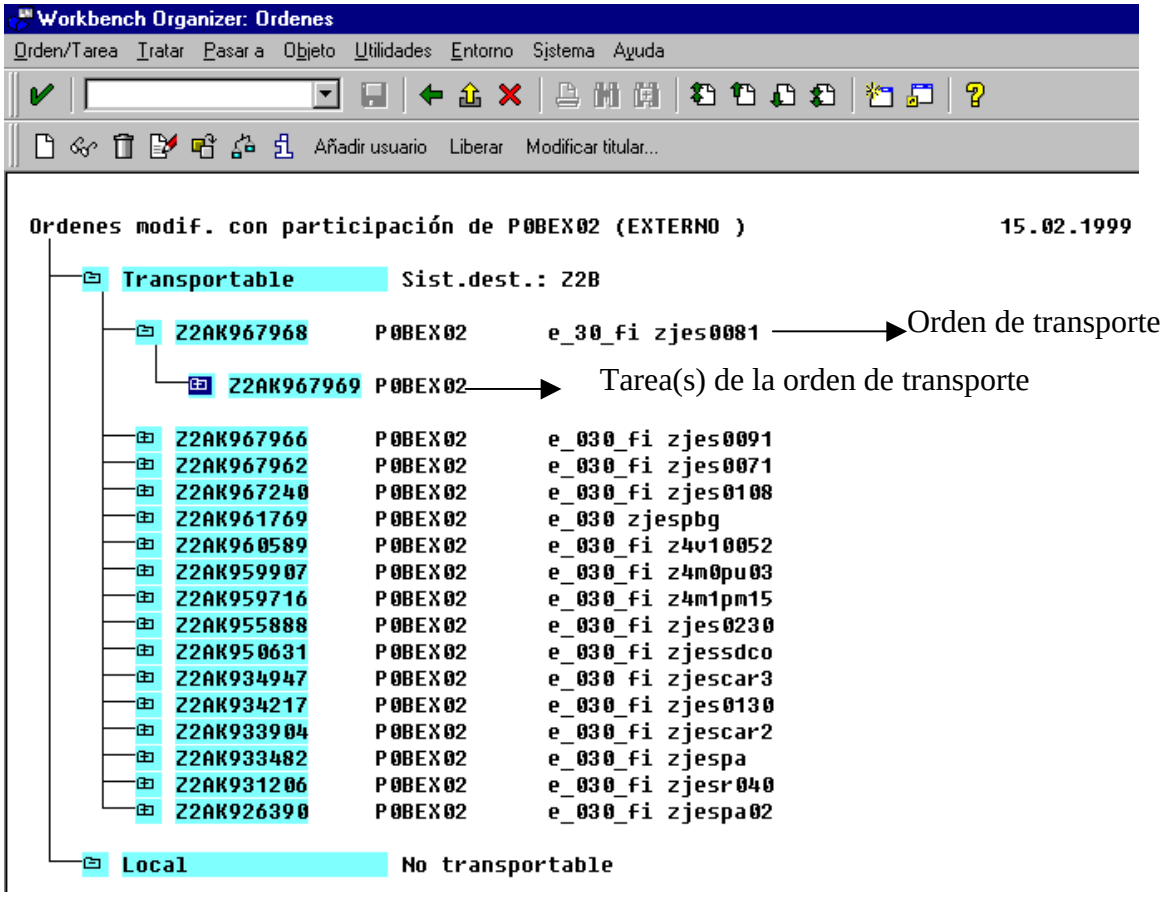
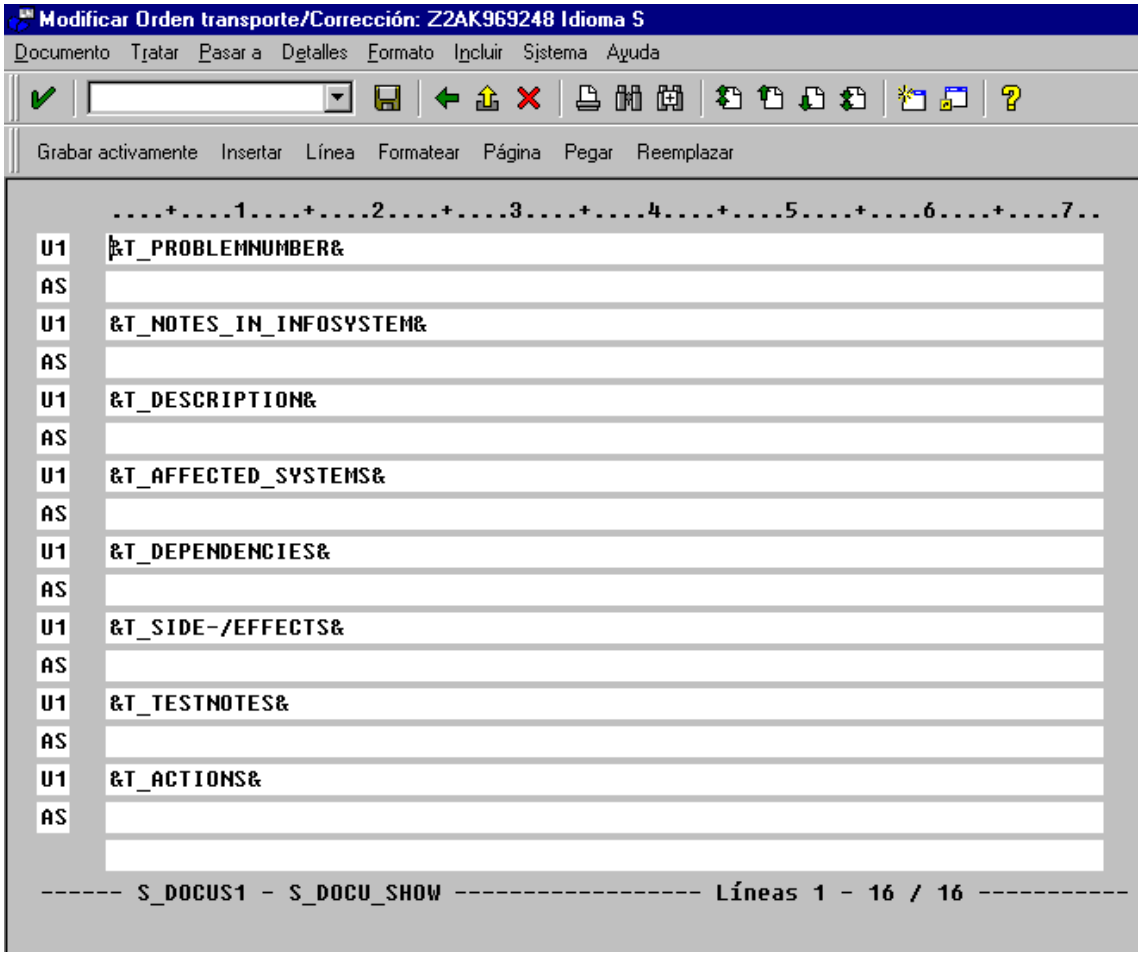



Fig. Ordenes.

Como vemos nos salen los objetos que tiene el usuario “P0BEX02”. Para liberar un objeto tenemos que hacer lo siguiente:

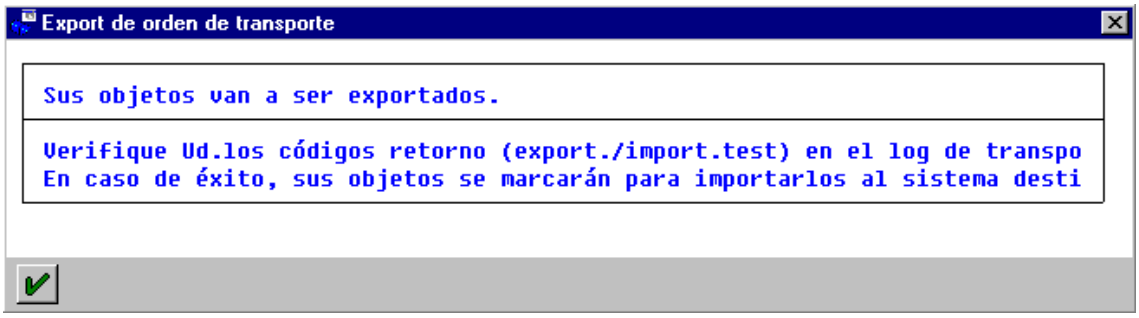
1. Este paso es opcional dependiendo de la empresa. En el caso de BASF se haría de la siguiente forma: para poderla transportar hemos de cambiar el titular de la orden de transporte, se hace pulsando el botón “Modificar titular...” y le cambiaremos el nombre, en caso de BASF se pondría: “P0B9998”.

2. Después de cambiar de titular hemos de desplegar las tareas de la orden de transporte (en este caso es el programa “ZJES0081”) como vemos la orden que hemos creado solo tiene una tarea (aunque puede tener muchas más pero a la hora de transportar es más liado).
   En la tarea que queramos hacemos un clic sobre esa rama y después pulsamos el botón “liberar” y nos saldrá la siguiente pantalla:




Aquí pulsaremos el botón “Grabar activamente” para grabarlos y después volveremos a la pantalla anterior (Véase Fig. Ordenes) y sobre el mismo objeto pulsaremos otra vez el botón liberar y veremos como cambiará de color.

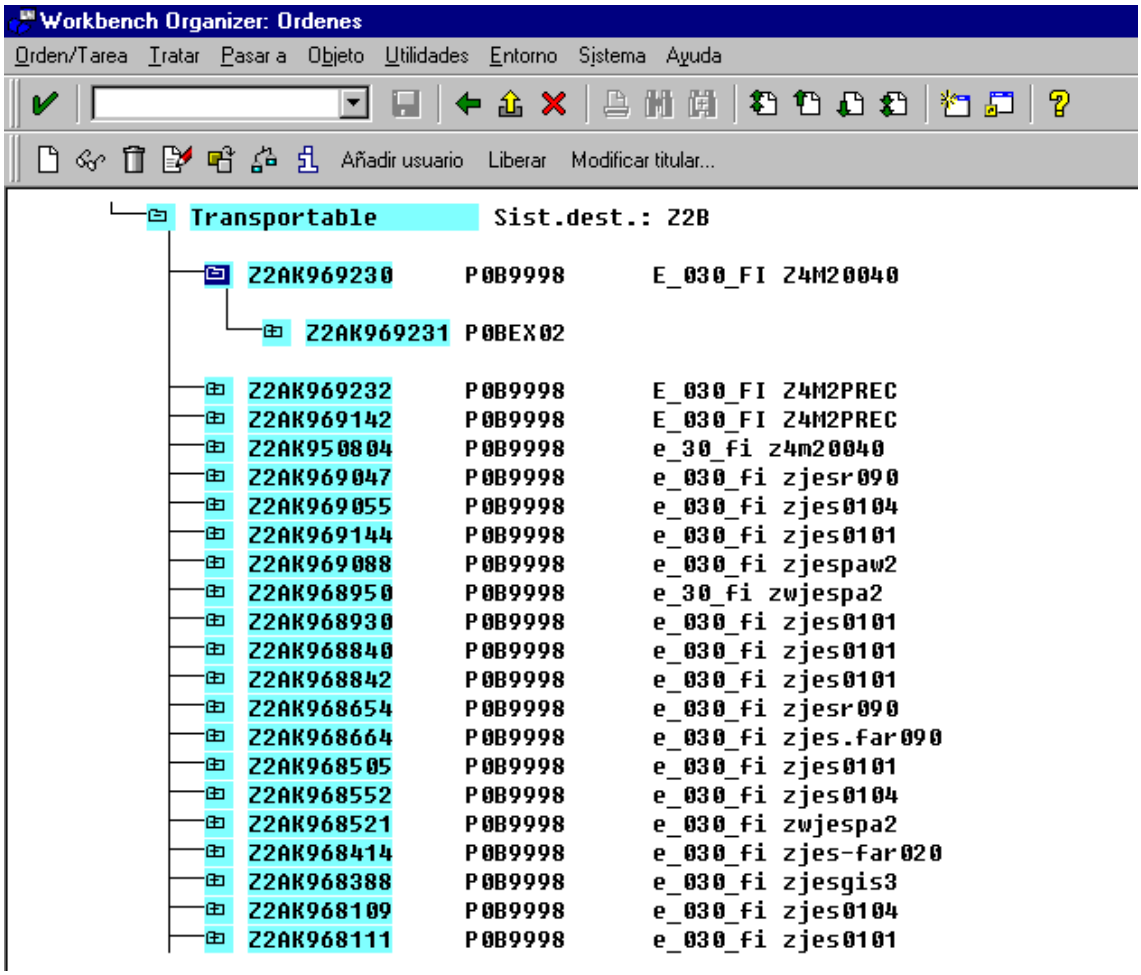
3. Después hemos de liberar la orden, hacemos clic y pulsamos el botón de liberar y nos saldrá la siguiente pantalla diciendonos que el objeto se va a transportar, pulsaremos ENTER y ya tendremos nuestro objeto transportado. La pantalla que sale es esta:




*NOTA: Es bueno apuntarse la orden de transporte de la orden de transporte, en este caso sería la “Z2AK967966”, por si en un futuro la tenemos que utilizar.*

## COMO VER NUESTROS OBJETOS TRANSPORTADOS

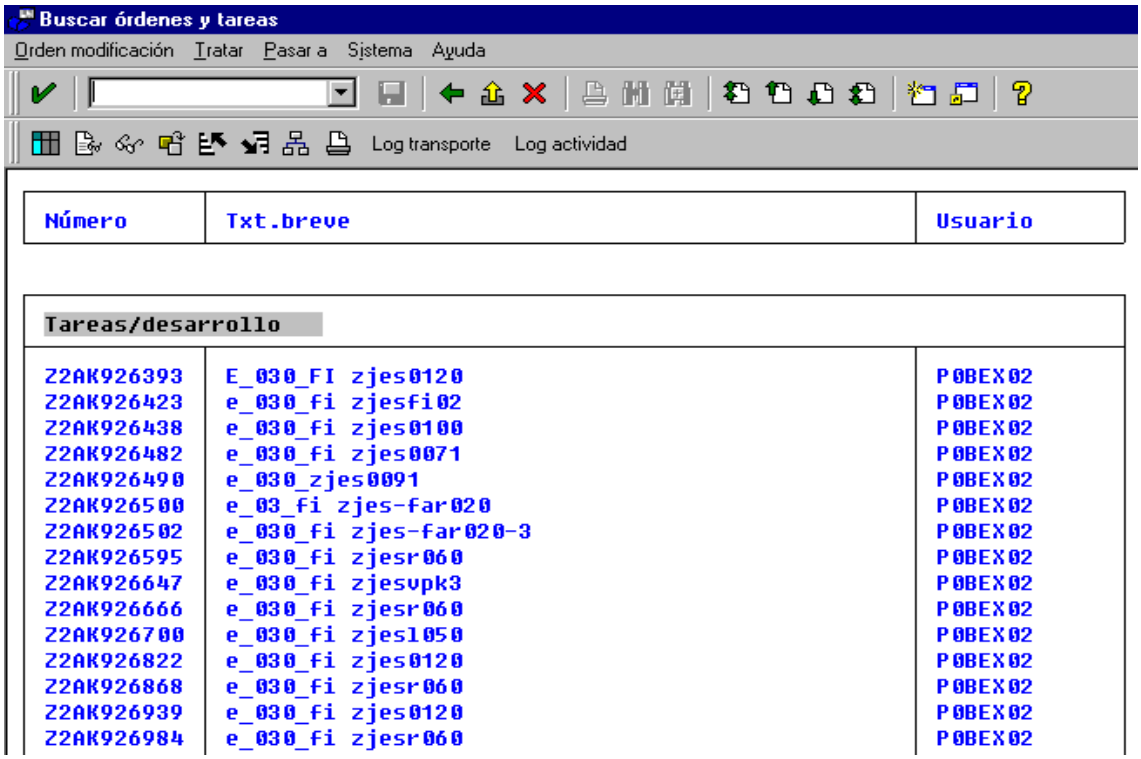
Para ver nuestros objetos transportados tenemos que ir a la pantalla de “Workbech Organizer” y ahí activar la casilla de “liberados” y después pulsar ENTER y en la parte final nos saldrán los objetos liberados, como en la siguiente pantalla:




## BUSCAR OBJETOS

En este caso podemos ver todos los objetos transportados o no de un usuario, por una determinada fecha, etc. Para hacerlos tenemos que ir a la pantalla de “Workbench Organizer” pulsamos sobre los prismáticos y nos saldrá la siguiente pantalla:

Como quiero ver los objetos del usuario “PB0EX02” lo escribo en el campo “usuario” y pulso F8 o ejecutar búsqueda y nos saldrá la siguiente pantalla:




Si queremos saber información sobre un determinado objeto, hacemos sobre el que queramos y después pulsamos sobre el botón “Log transporte” o “Log actividad”.

## BIBLIOTECAS DE FUNCION

La biblioteca de funciones nos permite ver, modificar e insertar funciones que ya tiene SAP, y que nos pueden ayudar a facilitarnos el trabajo. Tenemos funciones para:

- Gestión.
- Interfaces parámetros Import/Export.
- Interfaces parámetros Tabla/excepciones.
- Documentación.
- Texto fuente.
- Datos globales.
- Programa princ.

Para acceder, desde la pantalla de "ABAP/4 Development Workbench", pulsamos el botón "Biblioteca de funciones" y nos saldrá la siguiente pantalla:

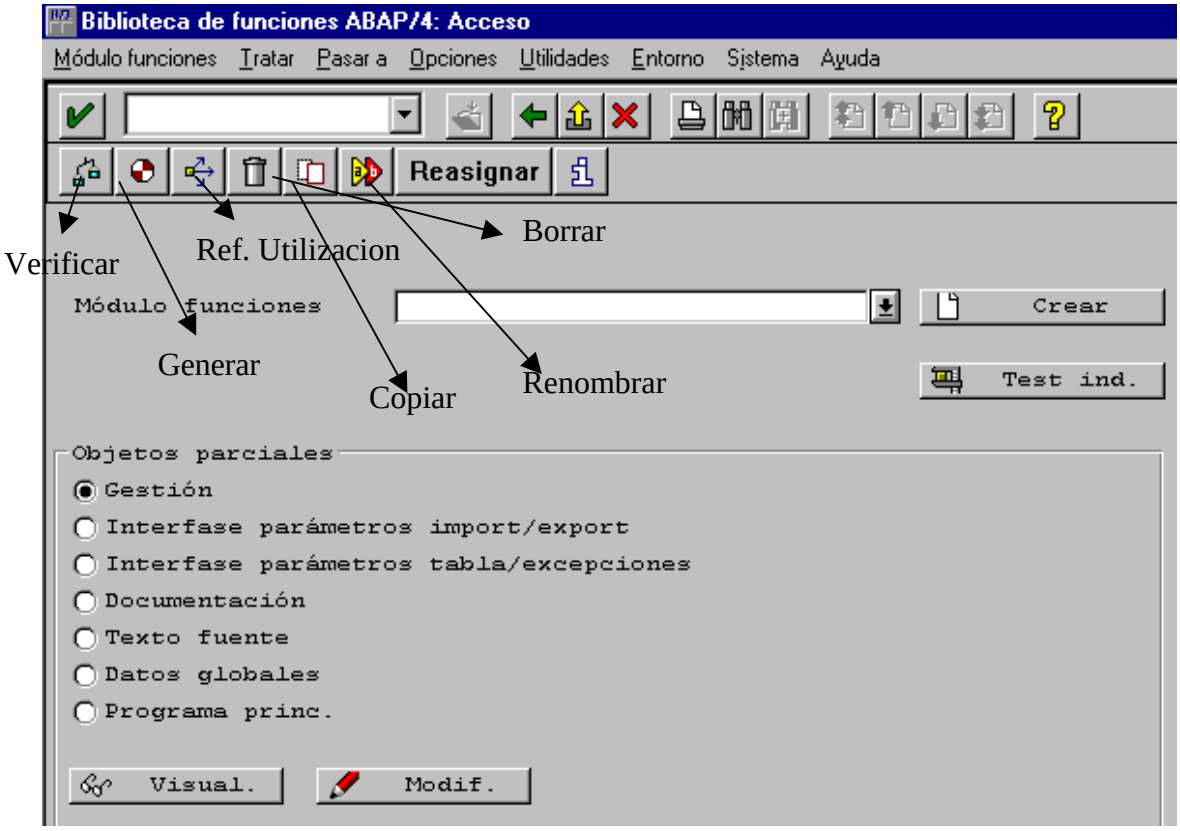


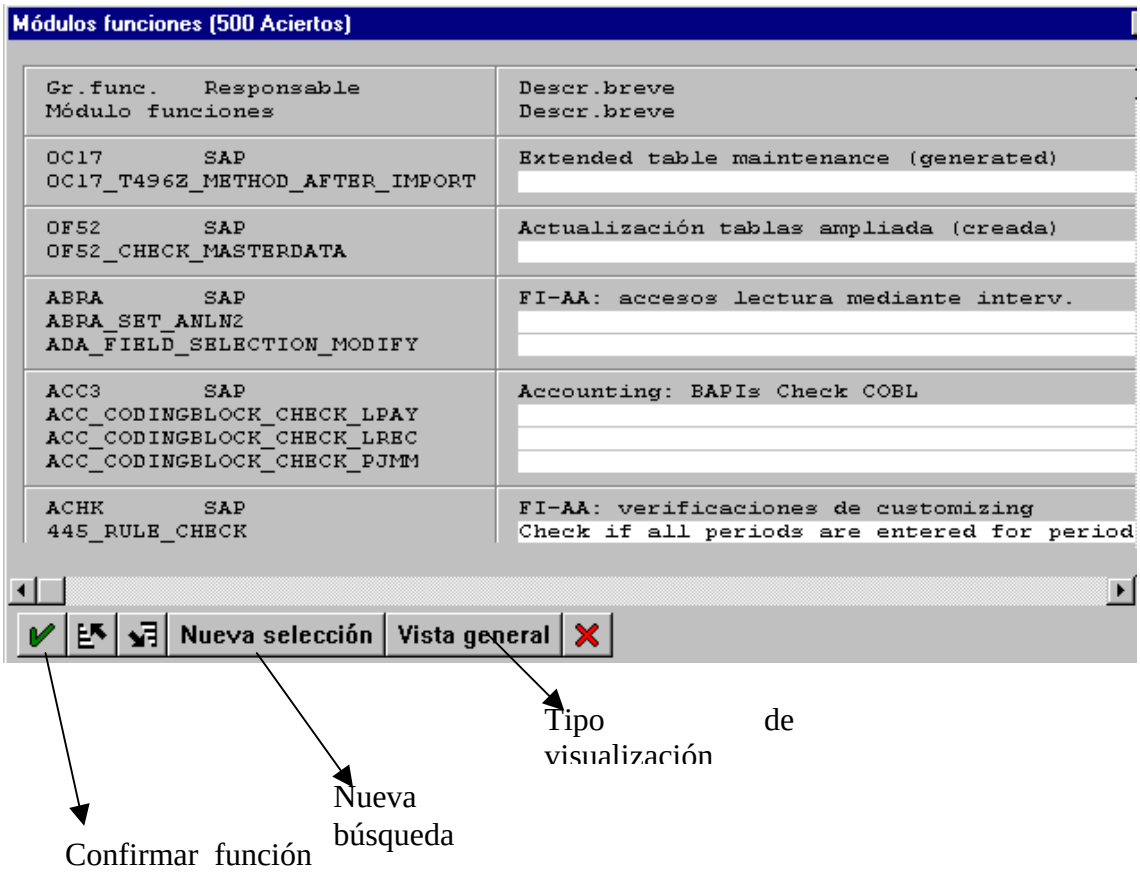

Fig. Biblioteca de funciones.

En el menú "Modul Pool" podemos "crear", "borrar", "modificar", "verificar", etc. de una función.

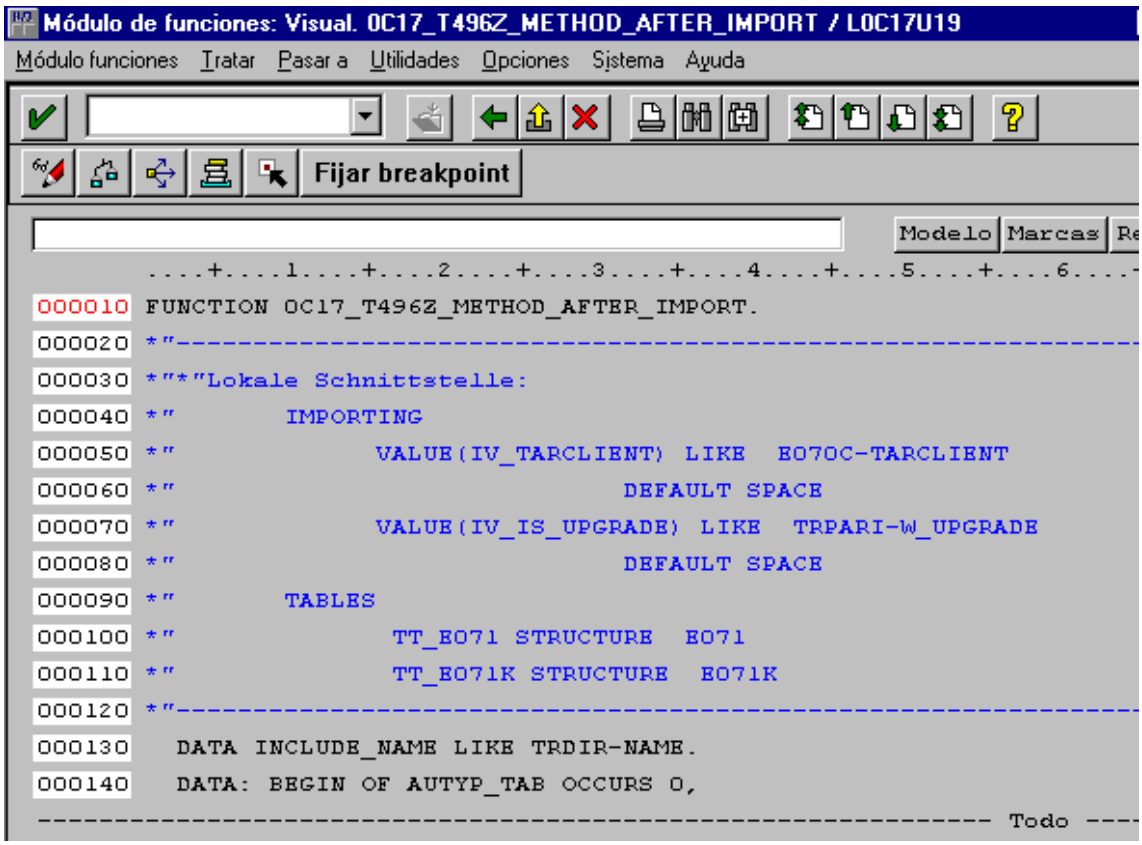
En el botón "Reasignar" podemos cambiar la clase de desarrollo de la función.

> En "Ref. utilización" nos sale una lista con los programas que utilizan la función que haya en "modulo funciones".

Si queremos ver todas las funciones de gestión, en el campo “Modulo de funciones” escribiremos “*” y pulsaremos F4 y nos saldrá otra pantalla con las funciones que tiene el sistema, la pantalla es esta:



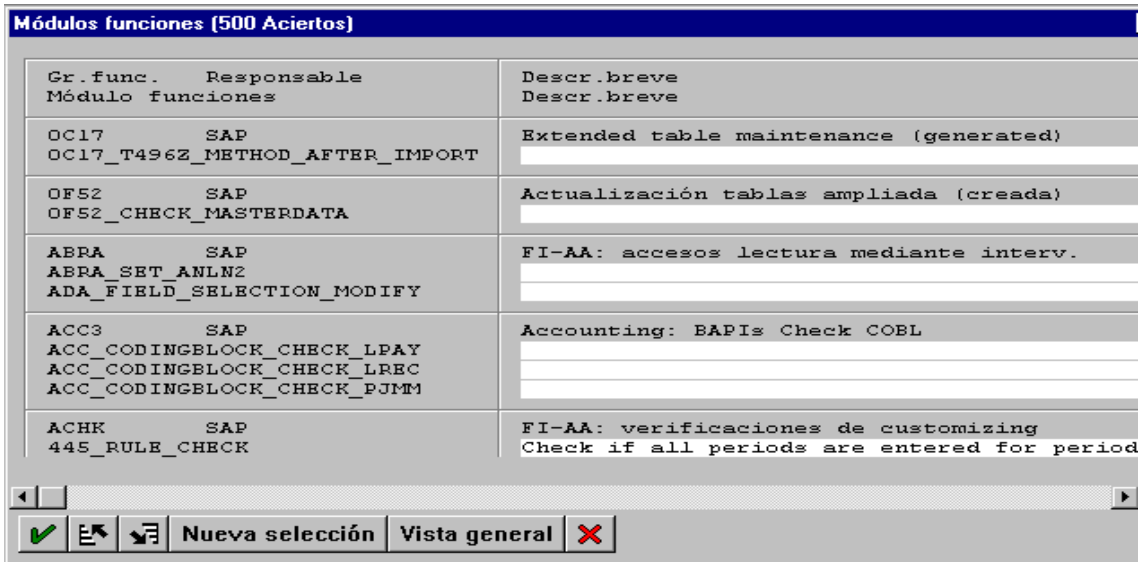

Para seleccionar una función hacemos clic sobre la que deseemos y pulsamos el botón de seleccionar o doble clic en la función deseada. Por ejemplo seleccionemos la primer función que salga, en la pantalla de la biblioteca de funciones (Véase Fig. Biblioteca de funciones). Y en esta pantalla pulsamos el botón “Visualizar” y nos saldrá la siguiente pantalla (en la siguiente página):



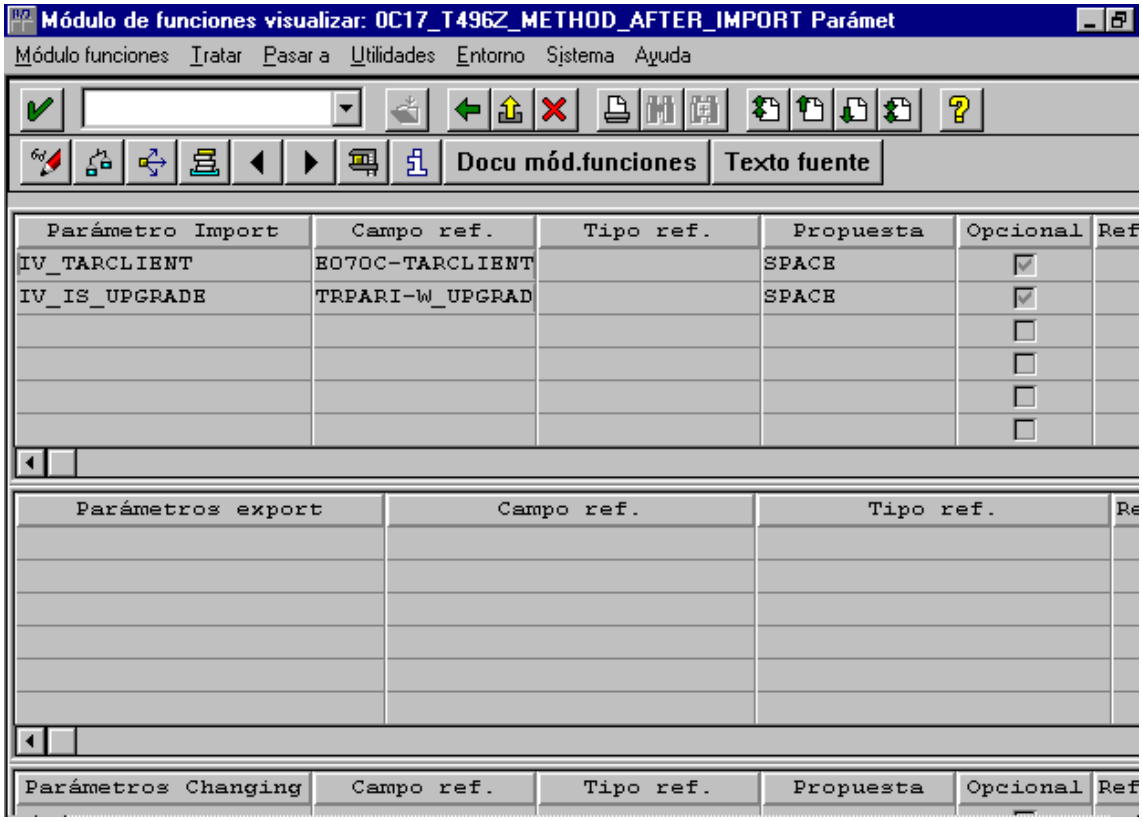

Como vemos, este es el código de la función escogida.

**PARAMETROS DE IMPORT/EXPORT**

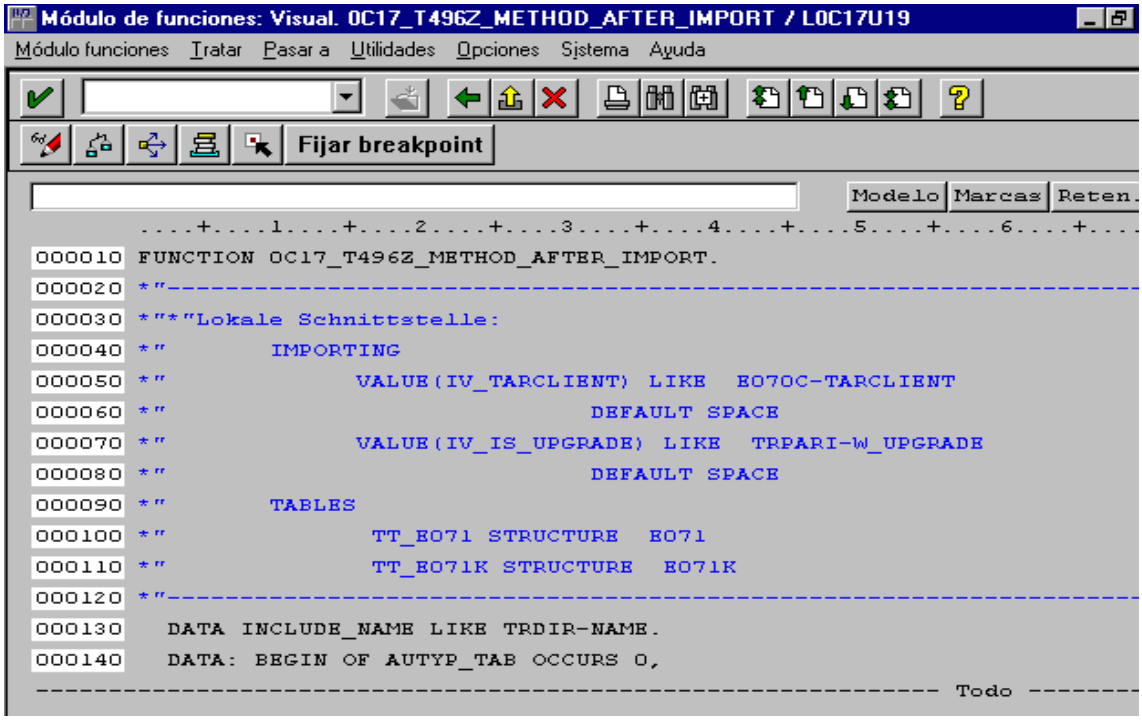
El funcionamiento es idéntico al ejemplo que hemos explicado anteriormente.
Primero vamos a la pantalla de la biblioteca de funciones (Véase Fig. Biblioteca de funciones), después seleccionamos el “radio button” que pone “interfaces parámetros IMPORT/EXPORT” en el campo “módulo funciones” escribimos un “*” y pulsamos F4 y nos sale la siguiente pantalla con las funciones existentes:



Si seleccionamos la primera función y después desde la pantalla de biblioteca de funciones pulsamos el botón “visualizar”, nos saldrá la siguiente pantalla:

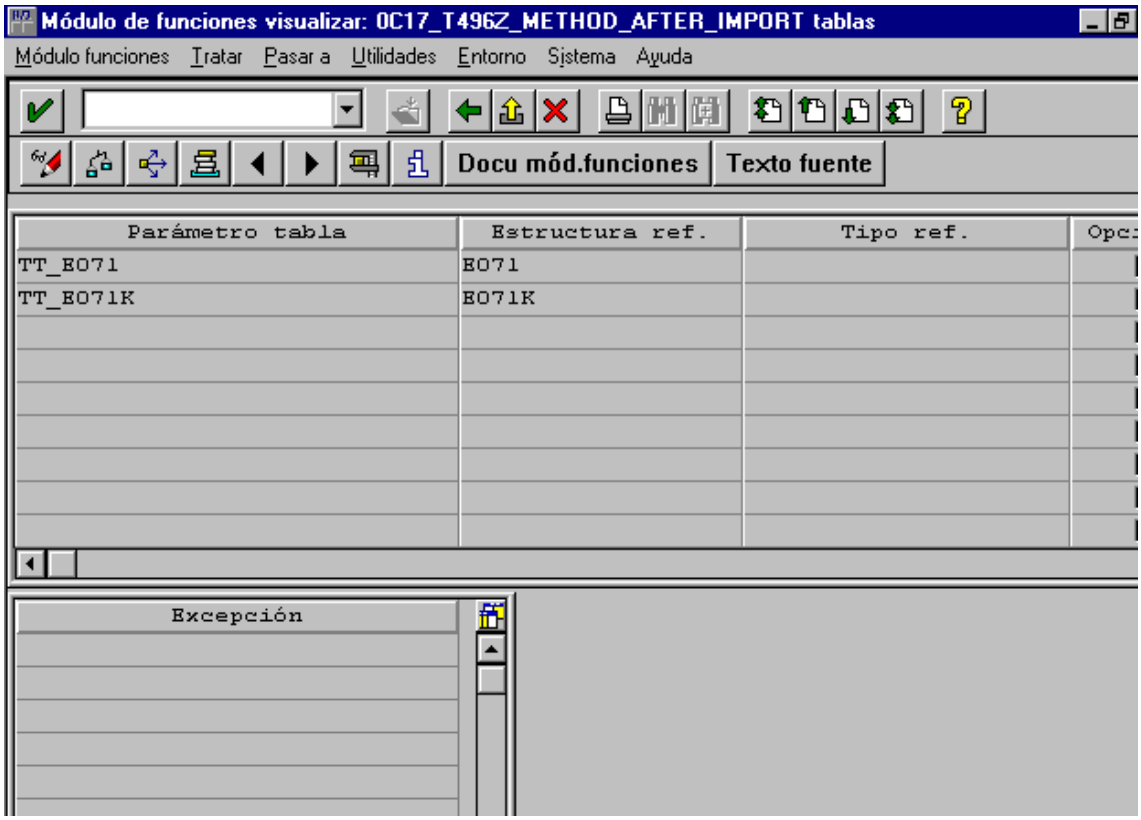



En esta pantalla podemos ver todos los parámetros de la función escogida y si queremos ver el texto fuente de esta función pulsamos el botón de “texto fuente” y nos saldrá esta pantalla con el código de la función:

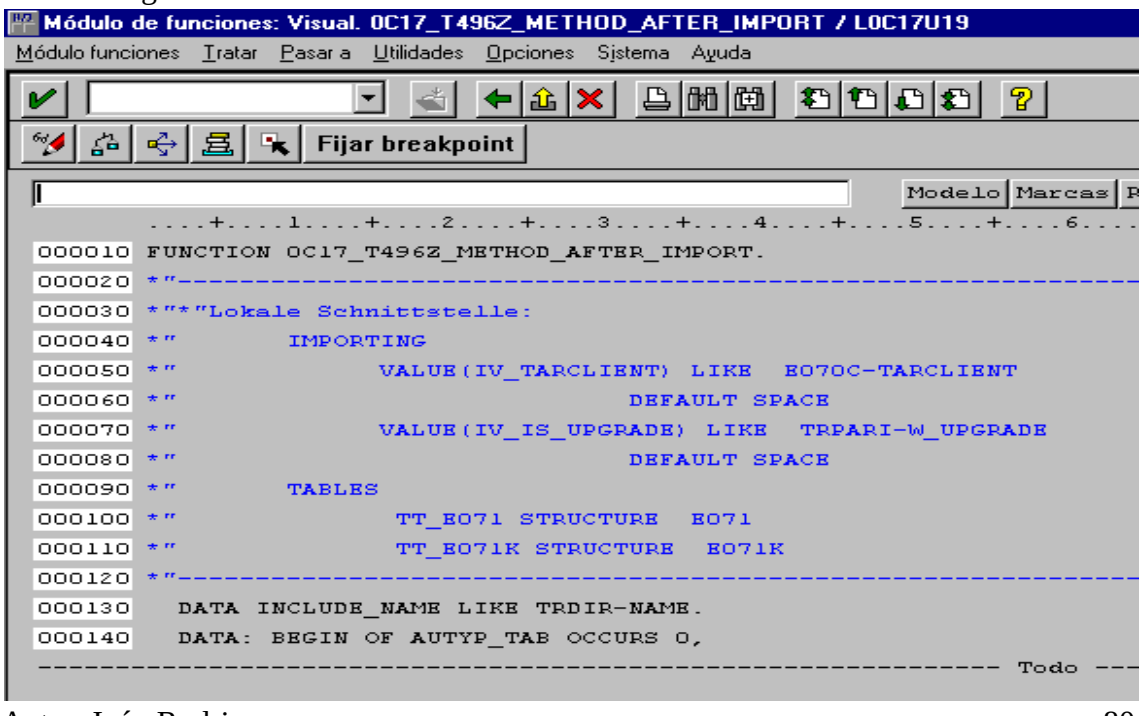


## PARAMETROS TABLA/EXCEPCIONES

Aquí también se opera de la misma forma. En este caso seleccionaremos la primera función, las pantallas son las mismas en los dos casos anteriores, y le daremos al botón visualizar y nos saldrá la siguiente pantalla:




Como vemos nos aparecen todos los parámetros de la función escogida y si queremos ver su código fuente, le damos al botón de “texto fuente” y nos sale la siguiente pantalla con el código fuente:



## LENGUAJE ABAP/4

AL FINAL DE CADA INSTRUCCIÓN LLEVA UN PUNTO '.'.

**ESTRUCTURACIÓN**

La estructura sería la siguiente:

REPORT (Sí es un programa ON LINE)
DECLARACIÓN DE TABLAS DICCIONARIO.
DECLARACIÓN DE TABLAS INTERNAS.
DECLARACIÓN DE VARIABLES.
PANTALLAS DE SELECCIÓN.
EVENTOS.
SUBRUTINAS.

**REPORT**

El REPORT siempre va al principio de un programa. Y tiene las siguientes opciones

- NO STANDARD PAGE HEADING ->Suprime la línea de cabecera del sistema, sin embargo no se suprime la línea de cabecera de columnas. Entonces se visualizan los títulos/cabeceras que hayamos creado.

- LINE-SIZE ->Número de columnas que tendrá una línea.

- LINE-COUNT -> Número de filas que tendrá la pantalla.

- DEFINING DATABASE -> Creo que es la declaración de una base de datos externa.

- MESSAGE-ID xx -> Número de un mensaje estándar, donde xx es la librería donde están los mensajes.

- RESERVE n LINES -> Mantiene líneas libres del listado de salida para evitar que por un salto de página queden separadas líneas de un mismo grupo.

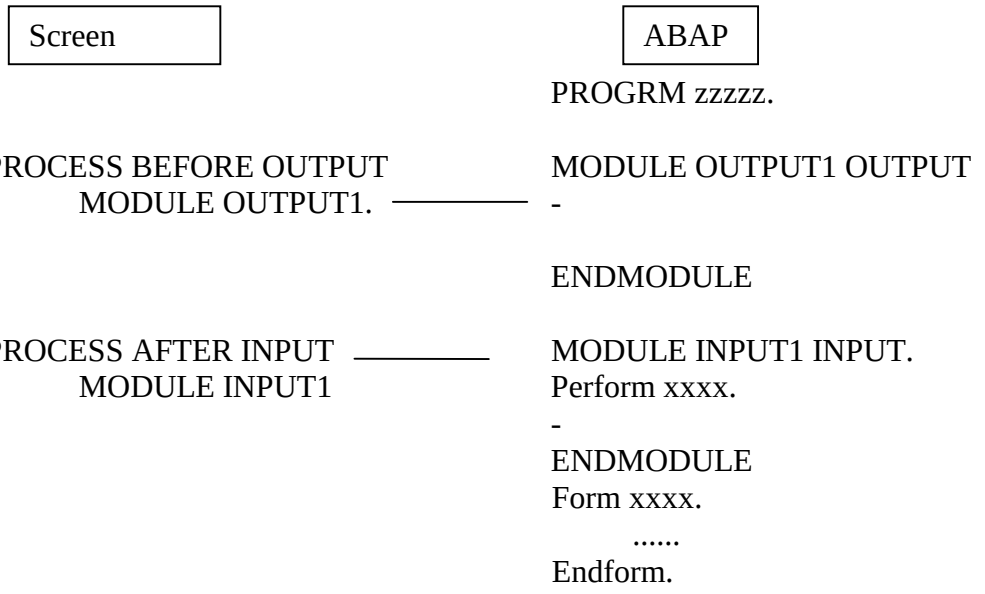
- PAGE-COUNT oo -> oo cantidad máxima de páginas por listado.

Ejemplo: voy a declarar un REPORT que no tenga cabeceras, con 65 columnas y 120 filas.

```
REPORT NO STANDARD PAGE HEADING
       LINE-SIZE 65
       LINE-COUNT 120.
```

**MODUL –POOL**

Los programas de diálogo en ABAP/4 son del tipo M (Modul-Pool). Estos programas solo pueden ejecutarse a través de una transacción.




Los Modul-Pool se utilizan para crear dynpros (o pantallas de dialogo).

Con la llamada MODULE .. en la lógica de proceso de una dynpro, el control se pasa al programa ABAP/4 y se procesa el módulo con el mismo nombre.

Los módulos asignados al evento PBO deberán complementarse con el parámetro OUTPUT y los asignados al PAI con el parámetro INPUT.

El PBO, como su nombre indica, es un evento que se produce antes de la visualización de la dynpro.

El PAI es el evento que se produce cuando en la dynpro se pulsa algún botón o tecla que confirma los datos introducidos. Aquí es donde vamos a trabajar con esos datos.

**DECLARACIÓN DE TABLAS DICCIONARIO**

Las tablas de diccionario son donde guardamos los datos.
Para poderlo declarar se utiliza la orden TABLES. La sintaxis sería esta:

```
TABLES tabla1, tabla2, etc.
```

**DECLARACIÓN DE TABLAS INTERNAS**

Se pueden declarar de varias formas diferentes:

**OPCION 1**

La definición de tablas internas es más complicada que las de diccionario, que son muy fáciles de declarar. La sintaxis sería la siguiente:

```
DATA: BEGIN OF nom_tabla  ( OCCURS num-ocurrencias),
                          Campo1 LIKE tabla_dicc-nom_campo,
                          campo2(longitud) TYPE  (tipo).
      END OF nom_tabla.
```

Donde:

Nom_tabla->Es el nombre de la tabla interna que vamos a declarar.

OCCURS num_ocurrencias->Sería como el número de registros que tendría la tabla, es opcional poner esta cláusula.

Al final del OCCURS pondremos una coma. Y si no está la cláusula, la pondremos después de nom-tabla.

Campo1->El nombre de la variable que declaramos (suele tener el mismo nombre que el campo de la tabla de diccionario).

Tabla_dicc->Nombre de la tabla de diccionario.

Nom_campo->Nombre del campo de la tabla interna. Al final pondremos una coma.

Siempre tenemos que poner un guión entre el nombre de la tabla de diccionario y el campo de esa tabla, ya que si no lo hacemos, el SAP no lo reconoce y da error.

Campo2-> Es igual que declarar una variable a nuestro gusto (se explica después, la declaración de variables). Un campo no ha de ser necesariamente igual que un campo de una tabla de diccionario.

Podemos declarar tantos campos como queramos.

Un ejemplo de declaración sería este:

```
DATA: BEGIN OF TABLA OCCURS 0,
          COUNTRY LIKE TABNA-COUNTRY,
          CITY (16) TYPE C,
END OF TABLA.
```

Como vemos, para separar cada línea utilizamos la coma y al final de la instrucción ponemos un punto.

**OPCION 2**

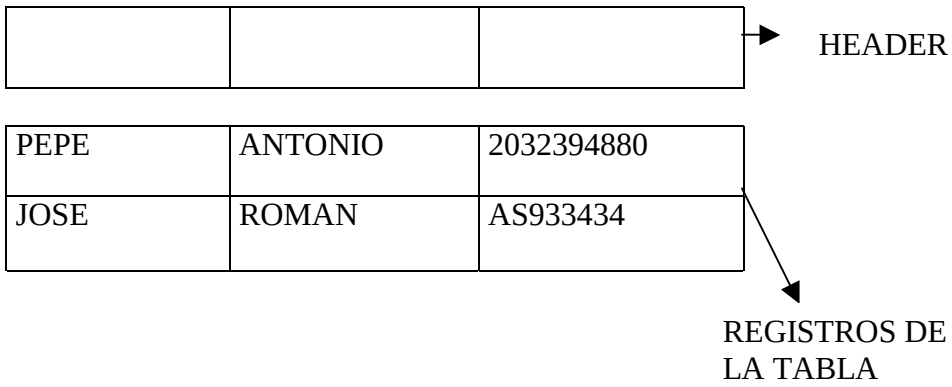
Si queremos que una tabla interna sea exactamente igual que una tabla de diccionario, podemos utilizar el INCLUDE que es más cómodo que declarar campo a campo. La sintaxis sería la siguiente:

```
DATA: BEGIN OF tabla-int.
            INCLUDE STRUCTURE tabla-dicc.

DATA:END OF tabla-int.
```

Como vemos, en el END OF se pone la DATA: , mientras que en la opción 1 no se ponía.

**COMO FUNCIONAN...**

En las tablas existe un HEADER. El HEADER es un registro de cabecera con el que siempre trabajaremos, para añadir, borrar o modificar algún registro de la tabla.




Antes de realizar cualquier operación (altas, modificaciones, etc.), hemos de mover los datos al header.

**DECLARACIÓN DE VARIABLES**

Podemos declarar las variables de tres formas:

- Asignar los atributos del campo de una tabla de diccionario con una variable.
- Declarar una variable del tipo que queramos.
- Una variable con la estructura de otra.

**ASIGNAR LOS ATRIBUTOS DEL CAMPO DE UNA TABLA DE DICCIONARIO CON UNA VARIABLE**

La forma de hacerlo sería la siguiente:

```
DATA: Variable LIKE tabla_dicc-nom_campo.
```

Donde:

Tabla_dicc->Nombre de la tabla de diccionario.

Nom_campo->Nombre del campo de la tabla interna. Al final pondremos un punto.

*Siempre tenemos que poner un guión entre el nombre de la tabla de diccionario y el campo de esa tabla, ya que si no el SAP no lo reconoce y da error.*

**DECLARAR UNA VARIABLE DEL TIPO QUE QUERAMOS**

La forma de hacerlo sería la siguiente:

DATA: Variable (tamaño) TYPE (tipo) [VALUE valor][SPACE][DECIMALS n.]

Los tipos de variable que podemos declarar son los siguientes:

- I-> Entero.
- N-> Numérico.
- P-> Empaquetado.
- F-> Coma flotante.
- C-> Carácter.
- D-> Fecha ( AAAAMMDD).
- T-> Hora (HHMMSS).
- X-> Hexadecimal.

VALUE es por si le queremos dar un valor a la variable cada vez que se ejecute el programa. El valor ha de ser del mismo tipo que la variable.

SPACE -> es un campo predefinido.

DECIMALS n -> Solo válido para variables de tipo I, N y F indica el número de decimal que va ha tener el número.

Si no declara el tipo de variable a un campo SAP asume que es de tipo C.

***NOTA: Para pasar carácter a paquet. Si el número final es 0 no lo tiene en cuenta con lo cual, pasamos el carácter a numérico y de numérico (Como no admite decimales) y si hemos de pasarlo a P con decimales lo multiplicaremos por el número de decimales antes de pasarlo.***

**UNA VARIABLE CON LA ESTRUCTURA DE OTRA**

Hay una forma más sencilla de declarar una variable con la estructura de otra variable (Un ejemplo con la estructura de una tabla de diccionario), la sintaxis sería la siguiente:

DATA: variable1 LIKE variable2.

Donde variable1 tendría la estructura de variable2, un ejemplo sería el siguiente:

DATA: tabla like SPFLI..

Tabla tendrá la misma estructura que la tabla de diccionario SPFLI.

Hay que recordar que sólo podemos añadir un sólo registro, es decir, no es una tabla interna sino una variable que contiene la estructura de otra variable.

**FIELD-GROUPS**

Define un grupo de campos distintos agrupándolos bajo el nombre del FIELD-GROUP.

Mediante INSERT se incluyen campos.
EXTRACT se asignan valores a este campo.
LOOP podemos acceder a la información.

**FIELD_GROUPS:**

HEADER, grupo_campos_1, grupo_campos_2.

INSERT campo_xh campo_yh campo_zh INTO HEADER.

INSERT campo_x1 campo_y2 campo_z3 INTO grupo_campos_1.

INSERT campo_x1 campo_y2 campo_z3 INTO grupo_campos_2.

GET tabla_1.

EXTRACT grupo_campo_1.

GET tabal_2.

EXTRACT grupo_campos_2.

El grupo HEADER tiene que declararse siempre, en este grupo están los campos por los que se realizan la clasificación.

**INSTRUCCIONES DE ENTRADAS DE DATOS**

Estas instrucciones nos permite introducir datos, para después ser utilizados en el programa. Ha esta forma de introducir de datos también se les puede llamar pantallas de selección.

En una programa se pueden declarar tantas pantallas de selección como queramos.

Para crear una sencilla pantalla de selección la estructura sería la siguiente:

SELECTION-SCREEN BEGIN OF BLOCK nom-bloque (WITH FRAME nom-titulo.)

INSTRUCCIONES

SELECTION-SCREEN END OF BLOCK nom-bloque.

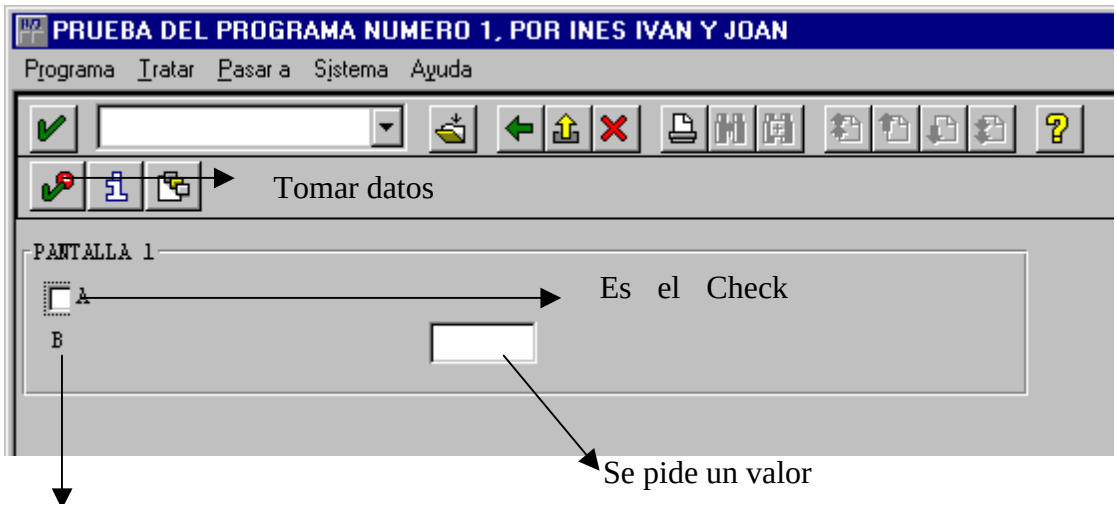
Nom_bloque-> Es el nombre del bloque. Va escrito sin comillas, ni simples ni dobles.

Nom_titulo-> El título de la ventana va escrito sin comillas ni simples ni dobles.

Un ejemplo sería el siguiente:

```
REPORT ZZJII01.
SELECTION-SCREEN BEGIN OF BLOCK AAA WITH FRAME TITLE TEXT-001.
   PARAMETERS: A AS CHECKBOX,
                        B LIKE TABNA-ID.
SELECTION-SCREEN END OF BLOCK AAA
```

El resultado en pantalla sería el siguiente:




El botón que pongo “tomar datos” sirve para realizar las ordenes que estén después de los SELECTION-SCREEN que halla al inicio del programa.

Los SELECTION-SCREEN se utilizan para hacer pantallas sencillas, es decir, que nos pida unos datos y de esos datos introducidos sacar una pantalla de información referente a los campos introducidos. Si queremos hacer pantallas más complejas utilizaríamos la SCREEN-PAINTER, explicada más adelante.

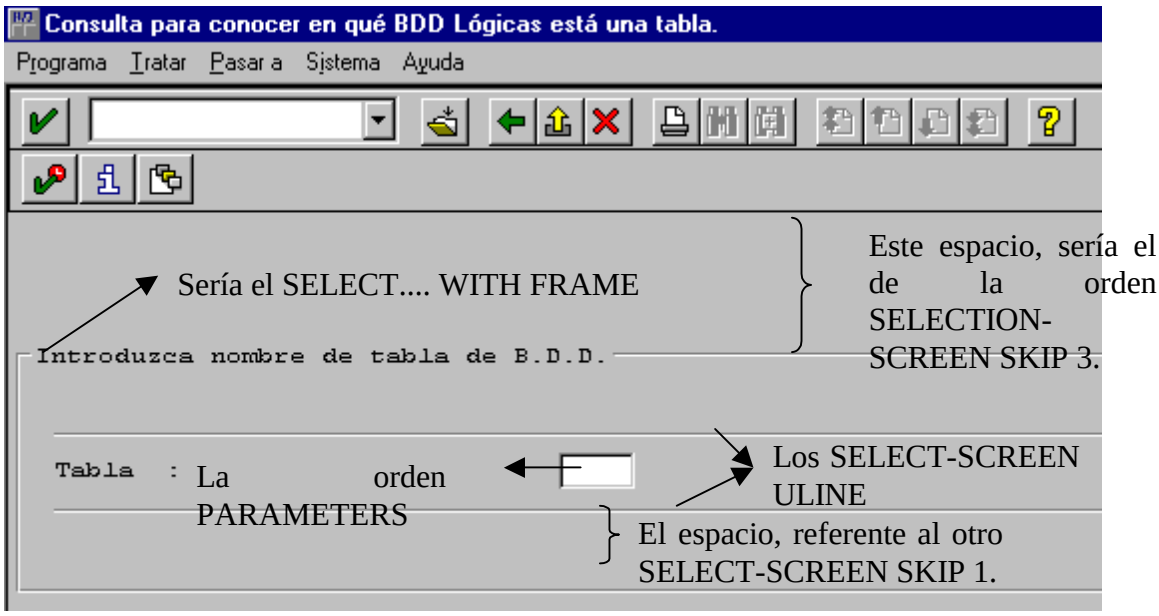
Un pequeño ejemplo de cómo haríamos una pantalla para pedir datos se haría de la siguiente forma:

```
SELECTION-SCREEN SKIP 3.
SELECTION-SCREEN BEGIN OF BLOCK AAA WITH FRAME TITLE TEXT-001.
SELECTION-SCREEN SKIP 1.
SELECTION-SCREEN ULINE.
*
PARAMETERS : TABLA(4) TYPE C.
*
SELECTION-SCREEN ULINE.
```

```
SELECTION-SCREEN SKIP 1.
SELECTION-SCREEN END OF BLOCK AAA.
```

- El primer SELECTION-SCREEN SKIP 3 -> Sirve para saltar 3 líneas, desde la posición donde este el cursor.
- El segundo SELECTION... realizo un "frame" o una especie de ventana, que servirá para pedir datos dentro de ella. Como veis utilizo un símbolo de texto para poner el título del "frame".
- El tercero realiza un salto de línea.
- El cuarto dibuja una línea que ocupara todo el ancho del "frame".
- Con la orden PARAMETERS pide un valor, en este caso, el nombre de una tabla.
- El quinto SELECTION... dibuja otra línea
- El sexto realiza un salto de línea.
- Y el último SELECTION-SCREEN cierre el bloque.

Si ejecutásemos estas ordenes en pantalla nos saldría lo siguiente:




Como veis el "frame" realizado con la orden "SELECTION-SCREEN BEGIN OF BLOCK AAA WITH FRAME TITLE TEXT-001" nos realiza como una ventana donde englobamos los datos que hay en su interior.

Cuando expliquemos los Report interactivos veremos más ejemplos de este tipo de ordenes y muchas más.

Para pedir datos en la SELECTION-SCREEN o fuera de ella, se utilizan las siguientes instrucciones:

**SET PARAMETER**

Con esta instrucción podemos guardar un valor en memoria para poder ser recuperado en la orden PARAMETERS... MEMORY ID id. La sintaxis sería la siguiente:

SET PARAMETER ID ‘id’ FIELD valor.

“id” -> Es el identificador donde estará almacenado valor.
“valor” -> es lo que se guardará, desde una variable hasta un simple string.

Podemos guardar una cosa en un programa y recuperarlo en otro, sabiendo el nombre del “id” no hay ningún problema. Un ejemplo sería este:

REPORT SAPMZTS1.

SET PARAMETER ID ‘HK’ FIELD ‘Test de parámetros’.

Y lo recuperaría:

REPORT SAPMZTS2.

PARAMETERS TEST(16) MEMORY ID HK.

El resultado en pantalla sería el siguiente:

TEST test de parámetros.

**PARAMETERS**

Su sintaxis es la siguiente:

PARAMETERS: variable LIKE campo-tabla-dicc.
Idem variable(longitud) TYPE [opciones-campo]
Idem variable AS CHECKBOX [DEFAULT ‘X’].
Idem variable RADIOBUTTON [opciones-resto]

En opciones de campo tenemos las siguientes posibilidades:

- OBLIGATORY-> hace que al lado del campo en la pantalla de entrada de parámetros, aparezca un ‘?’ indicando su obligada entrada.
- LOWER CASE -> convierte el texto introducido a minúsculas, si no se pone, por defecto aparece en mayúsculas, cuando lo utilizamos para pedir un fichero que es de tipo UNIX **es obligatorio** ponerlo ya que sino cuando lo abramos el fichero dará error en la apertura.
- MATCHCODE OBJECT nombre-match -> Asocia un matchcode al campo creado.
- MEMORY ID id -> Sirve para indicarle un valor por defecto a un campo, este valor se guarda en memoria, en el identificador id, que puede tener como máximo 3 caracteres. Para guardar un
- MODIF ID grupo -> Indicamos que un objeto se puede modificar, para ello ha de estar en un grupo, que se lo indicaremos en “grupo”. Para guardar un valor se utiliza la orden SET PARAMETER.

En el resto de opciones tenemos las siguientes posibilidades:

- GROUP n -> Valido solo para los ”radiobutton”, indica a que grupo pertenece, donde “n” es el número del grupo.
- DEFAULT ‘valor’ -> Indica si por defecto el “radiobutton” o el “checkbox” estará activo, por defecto está desactivado (valor vale una espacio en blanco), para activarlo hemos de poner “X” en valor.
- MODIF ID grupo -> Indicamos que un objeto se puede modificar, para ello ha de estar en un grupo, que se lo indicaremos en “grupo”.

Como vemos podemos pedir datos de dos formas (las dos primeras), las dos últimas son añadidos a la entrada de datos.

Podemos pedir un campo que sea igual que un campo de una tabla interna o pedir un campo como nosotros queramos que sea.

**RANGES**

RANGES nombre_rango FOR campo.

```
DATA: BEGIN OF nombre_rango OCCURS 10,
            SIGN (1),
            OPTION (2),
            LOW LIKE campo_li,
            HIGH LIKE campo_hi,
            END OF nombre_rango.
```

La sentencia RANGES es la forma más cómoda de definir una tabla interna para el parámetro IN (lo mismo que un “=”) en un SELECT.

Se genera una tabla interna transparente para el usuario con la siguiente estructura:

SIGN puede tomar los valores ‘I’ INCLUDE (Por defecto) o ‘E’ EXCLUDE.

OPTION puede tomar los valores EQ, NE, LT, GT, LE, GE, BT, NB, CP, NP.

LOW y HIGH marca los valores entre.

Esta tabla se utiliza cuando queremos buscar más de un dato de un mismo campo en un bucle o en sentencias de condición.

**SELECT-OPTIONS**

Los SELECT-OPTIONS internamente es una tabla que tiene los valores, con los cuales se puede trabajar.

SELECT-OPTION nombre_selección FOR CAMPO_1

DEFAULT ‘xxxx’ TO ‘yyyy’

Genera una línea en la pantalla de selección con un texto al principio y la posibilidad de entrar datos.

El nombre de la selección puede tener como máximo 8 caracteres.

El parámetro FOR indica el campo en el que se van a comprobar las entradas.

DEFAULT propone valores únicos o intervalos.

Se genera una tabla interna transparente para el usuario con la siguiente estructura:

Nombre_selección-SIGN (1),
Nombre_selección-OPTION (2),
Nombre_selección-LOW LIKE campo_li,
Nombre_selección-HIGH LIKE campo_hi,
Nombre_selección-LOW = valor,
Nombre_selección-HIGH = valor.

SIGN puede tomar los valores I, INCLUDE (Por defecto), o E, EXCLUDE.

OPTION puede tomar los valores EQ, NE, LT, GT, LE, GE, BT, NB, CP, NP.

LOW y HIGH marcan los valores entre el máximo y el mínimo.

También podemos realizar una selección de varios campos, no de solo uno. La realizarlo la sintaxis sería la siguiente:

```
SELECT-OPTIONS:
        Campo1 FOR.. ,
        Campo2 FOR...
```

Es conveniente saber que cuando utilizamos el campo de un “select-options” en una condición hemos de poner “IN” que es lo mismo que “=”. No se pone el “=” ya que si no, no realiza bien la comparación por ello siempre hemos de poner “IN”.

**EVENTOS**

Los eventos pueden ser de dos tipos:

- Asincronos->Puede pasar en cualquier momento.
- Síncronos->Pasa con periodicidad.

Tenemos los siguientes eventos:

**INITIALIZATION**

En este evento sé inicializarian todas las variables de los programas (o al menos aquellas que su ámbito de utilización sea en todo el programa). Ejemplo:

```
INITIALIZATION.

        Cont = 0, Suma = 0.
```

**START-OF-SELECTION**

Este evento se lanza cuando se procesa la instrucción REPORT... En ese momento se empieza a ejecutar el código que se encuentra entre REPORT y la palabra SART-OF-SELECTION. Inmediatamente después se procesa el bloque contenido dentro de este evento.

Se utiliza cuando hay una selección de datos. Se utiliza siempre, cuando queremos cargar los datos, desde una tabla o fichero, para después utilizarlo en el programa.

**END-OF-SELECTION**

El código asociado a este evento se procesa cuando se termina la selección de datos de tablas o bases de datos lógicas.
Se puede forzar por código la interrupción de la selección de datos con la instrucción STOP. Si en nuestro código se procesa esta instrucción no se leerán más entradas y se procesará inmediatamente el código correspondiente al evento END-OF-SELECTION.
Otra alternativa es usar la instrucción EXIT, que cancela el procesamiento del Report y muestra la lista.

**TOP-OF-PAGE**

Este evento se procesa antes de que el primer dato salga en cada página. Se suele usar para poner el título a las páginas o cabeceras, en combinación con la opción de la sentencia REPORT ...NO STANDARD PAGE HEADING.

Si añadimos la opción ...DURING LINE-SELECTION se ejecutará sólo en listas secundarias que eventualmente se generen como consecuencia de una acción en el Report actual.

**END-OF-PAGE**

Este evento se lanza cuando se crea automáticamente una nueva página. Esto ocurrirá cuando el número de líneas procesado sobrepase el número de líneas fijado para la página en la sentencia REPORT, o cuando la sentencia RESERVE n LINES. Después de esta sentencia “RESERVE” se forzará nueva página si en la misma no hay al menos n líneas libres. ‘n’ puede ser una variable o una constante literal.
No se procesará END-OF-PAGE si se refuerza nueva página por código (NEW-PAGE).

**AT LINE-SELECTION**

A partir de la selección de una línea de pantalla (haciendo doble clic) se desencadenan una serie de eventos.

**AT PFn**

A partir de la selección de una tecla de función, se desencadenan una serie de eventos.

**AT USER-COMMAND**

Cuando hay un botón de selección y lo seleccionamos, a partir de su selección, se desencadenan unas acciones.

**AT SELECTION-SCREEN**

Define la pantalla de selección, tal como se presenta al usuario. Al hacer un enter o confirmar una pantalla de selección se ejecutan las órdenes que haya a continuación. Tiene las siguientes opciones:

- Podemos hacer que cuando se pulse F4 se ejecute las instrucciones que contenga el AT SELECTION-SCREEN. La sintaxis es la siguiente:

  AT SELECTION-SCREEN ON VALUE REQUEST FOR campo.

  Este "campo" suele ser un nombre de campo que se pide con la orden PARAMETERS.
- ... OUTPUT -> con esta opción hacemos que se ejecute este evento antes de que se visualice la pantalla, por ejemplo, cambiar los atributos de un programa.

**LLAMADA A OTROS PROGRAMAS**

Se pueden llamar a programas externos, internos y funciones. Los internos son más conocidos como subrutinas. Las funciones son programas que el SAP tiene para ayudar al programador en tareas diversas.

Los parámetros se pueden pasar:

Por valor -> los parámetros formales son copia de los actuales.

Por valor y resultado -> al abandonar el subprograma, los parámetros formales se traspasan al actual

Por referencia -> las variaciones en el parámetro formal tienen efecto inmediato en el actual.

Las tablas internas siempre se pasan por referencia, ya que se definen como globales.

**SUBRUTINAS O PROGRAMAS INTERNOS**

Para llamar a una subrutina se utiliza la orden PERFORM:

```
PERFORM nom-subrutina.
```

La sintaxis para definir una subrutina sería la siguiente:

```
FORM nom-subrutina.

    INSTRUCCIONES.
ENDFORM.
```

En los PERFORM también podemos pasar parámetros. Su sintaxis sería:

```
PERFORM nom-subrutina  TABLES nombre_tabla
                       USING parametro1 parametro2
                       CHANGING campo_3.
```

Y en la subrutina sería:

```
FORM nom-subrutina   TABLES nombre_tabla
                     USING campo_1
                     VALUE campo_2
               CHANGING VALUE campo_3

ENDFORM.
```

TABLES y USING se pasan por referencia.
VALUE se pasa por valor.
CHANGING VALUE al terminar el subprograma transfiere el valor de los parámetros formales a los actuales.

Los parámetros se pasan por orden, es decir, el primer parámetro del PERFORM con el primero del FORM. No importa que los nombres sean diferentes o iguales, pero sí deben ser del mismo tipo.

También se puede llamar a una subrutina que está en otro programa. Su sintaxis sería la siguiente:

```
PERFORM nom-subrutina (USING parámetro1 parámetro 2 )(nom-programa).
```

Entre paréntesis al final del PERFORM pondríamos el nombre del programa. El paso de parámetros es opcional.

**PROGRAMAS EXTERNOS**

Para llamar a un programa externo se utiliza la orden SUBMIT.

```
SUBMIT nom-programa.
```

Por defecto el programa que es llamado no vuelve al programa desde el que se llama, sino que pierde control y termina cuando termina el programa llamado.
Para que no suceda esto se pone el “AND RETURN”.

```
SUBMIT nom-programa AND RETURN.
```

También podemos llamar a programas a través de transacciones, para llamar se utiliza la orden CALL TRANSACTION... Su sintaxis es la siguiente:

```
CALL TRANSACTION “xxxx”
                    USING “tabla BDC”
                    MODE “modo de visualización”
                    UPDATE “modo actualización”.
```

Modo visualización:

- A -> Visualizar todo.
- E -> Visualizar sólo errores.
- N -> No visualizar nada.

Modo de actualización:

- S -> continuar con el proceso cuando termine la actualización (síncrono).
- A -> Continuar inmediatamente con el proceso.

Códigos de retorno y campos del sistema: el código de retorno informa del proceso de la transacción a la que se llama:

- 0 -> Con éxito.
- <=1000 -> Error en el programa de diálogo.
- > 1000 -> Error en la actualización.

## FUNCIONES

Las funciones se llaman a través de la orden CALL FUNCTION**.** Las funciones son muy útiles ya que como he dicho facilitan la tarea al programador.
SAP incorpora muchas funciones.

La sintaxis es la siguiente:

```
CALL FUNCTION función

 IMPORTING
        OPCIONES
EXPORTING
        OPCIONES
TABLES
        OPCIONES
CHANGING
        OPCIONES
EXCEPTIONS
        OPCIONES
```

Si se quiere saber las opciones de una determinada función, poniendo SHOW FUNCTION nombre-función o SHOW FUNCTION *, en la línea de comandos nos sacará la ayuda de esa función o de cualquier otra función.

EXPORTING

Pasa campos, cadena de caracteres o tablas internas a la función indicada. Los campos no son modificables en la función. En determinadas funciones esta función es opcional.

IMPORTING

Pasa campos, cadena de caracteres o tablas internas procedente de la función a nuestro programa. En determinadas funciones es opcional.

TABLES

Pasa tablas internas por referencia. En determinadas funciones es opcional. Aquí las tablas pueden modificarse en la función.

CHANGING

Pasa campos, cadena de caracteres o tablas internas a la función indicada. Aquí los campos también pueden modificarse en la función. Esta función también es opcional.

EXCEPTIONS

Las excepciones son valores que devuelve la función, y que sirven para saber si la función ha realizado su trabajo con éxito o no.
Las dos excepciones siguientes están predefinidas por el sistema y tienen un especial significado:

- OTHERS -> Cubre todas las excepciones del usuario en la función llamada.
- ERROR_MESSAGE -> .....

**EJEMPLOS DE FUNCIONES**

Voy a explicar dos funciones muy utilizadas en SAP que son la Download y la Upload. La Download (como he explicado en el Texto fuente de un programa) sirve para pasar los datos de una tabla interna a un fichero de texto.
El Upload es todo lo contrario, pasa los datos de un fichero de texto a una tabla interna.

DOWNLOAD

La sintaxis del Download es la siguiente:

```
CALL FUNCTION 'DOWNLOAD'
     EXPORTING
```

```
        FILENAME          = 'nombre'
        FILETYPE          = 'tipo'
        ITEM              = ‘comentario’
IMPORTING                                          PARAMETROS
        FILESIZE = FSIZE                           OPCIONALES
        ACT_FILENAME = FNAME
        ACT_FILETYPE = FTYPE
TABLES
       DATA_TAB           =  tabla_interna
 EXCEPTIONS
        CONVERSION_ERROR    = 1
        INVALID_TABLE_WIDTH = 2
        INVALID_TYPE        = 3.
```

En EXPORTING tenemos:

FILENAME -> Nombre del archivo donde se guardarán los datos de la tabla.
FILETYPE -> Tipo del archivo, existen cuatro: ASC(ASCII), BIN (Binario), DAT, WK1(Formato EXCEL).
ITEM -> Comentario que le damos al fichero.

En IMPORTING, aunque es opcional, nos devuelve los bytes que ocupa el fichero (FILESIZE), el nombre y tipo de fichero que se ha creado.

En TABLES, tenemos:

DATA_TAB -> De que tabla interna se van a sacar los datos que vamos a guardar en el fichero.

EXCEPTIONS, tenemos:

Son las excepciones para los errores.

Hay que recordar que sólo podemos utilizar una tabla interna para hacer el download. Si queremos pasar el contenido de una tabla de diccionario a un fichero primero hemos de pasar dicho contenido a una tabla interna. En el siguiente ejemplo completo, pasamos el contenido de una tabla de diccionario a una tabla interna:

```
REPORT ZZJII12.
TABLES: TABNA.
DATA: BEGIN OF TAB OCCURS 0.
     INCLUDE STRUCTURE TABNA.
DATA: END OF TAB.

PERFORM LLENAR-TABLA.

CALL FUNCTION 'DOWNLOAD'
    EXPORTING
        FILENAME          = 'C:\ZZJII.TXT'
```

```
      FILETYPE          = 'ASC'
   TABLES
      DATA_TAB          =  TAB
   EXCEPTIONS
      CONVERSION_ERROR    = 1
      INVALID_TABLE_WIDTH = 2
      INVALID_TYPE        = 3.

FORM LLENAR-TABLA.
  SELECT * FROM TABNA.
      MOVE-CORRESPONDING TABNA TO TAB.
      APPEND TAB.
  ENDSELECT.
ENDFORM.
```

Paso los datos de la tabla de diccionario a la tabla interna

UPLOAD

La sintaxis es completamente idéntica a la del DOWNLOAD.

```
CALL FUNCTION 'UPLOAD'
      EXPORTING
            FILENAME        = 'nombre'
            FILETYPE        = 'tipo'
            ITEM            = 'comentario'
      IMPORTING
            FILESIZE = FSIZE
            ACT_FILENAME = FNAME
            ACT_FILETYPE = FTYPE
      TABLES
           DATA_TAB         =  tabla_interna
       EXCEPTIONS
            CONVERSION_ERROR    = 1
            INVALID_TABLE_WIDTH = 2
            INVALID_TYPE        = 3.
```

PARAMETROS OPCIONALES

En EXPORTING tenemos:

FILENAME -> Nombre del archivo donde se leen los datos a guardar en la tabla.
FILETYPE -> Tipo del archivo, existen cuatro: ASC(ASCII), BIN (Binario), DAT, WK1(Formato EXCEL).
ITEM -> Comentario que le damos al fichero.

En IMPORTING, aunque es opcional, nos devuelve los bytes que leemos del fichero (FILESIZE), el nombre y tipo de fichero que se ha creado.

En TABLES, tenemos:

DATA_TAB -> En que tabla interna se van a guardar los datos que vamos a leer del fichero.

EXCEPTIONS, tenemos:

Son las excepciones para los errores.

También, en este caso, donde se guardan los datos ha de ser una tabla interna. Un ejemplo completo sería este:

```
REPORT ZZJII13.
TABLES: TABNA.
DATA: BEGIN OF TAB OCCURS 0.
     INCLUDE STRUCTURE TABNA.
DATA: END OF TAB.

CALL FUNCTION 'UPLOAD'
    EXPORTING
FILENAME           = 'C:\IVAN.TAB'
        FILETYPE           = 'ASC'
TABLES
    DATA_TAB           =  TAB
    EXCEPTIONS
        CONVERSION_ERROR    = 1
        INVALID_TABLE_WIDTH = 2
        INVALID_TYPE        = 3.

LOOP AT TAB.
    WRITE: / TAB-COUNTRY.
ENDLOOP.
```

**IMPRESIÓN DESDE UN ABAP**

Desde un programa de SAP podemos imprimir de dos formas diferentes: La primera es imprimirlo después de haberlo ejecutado. Y imprimir un report mientras se ejecuta.

**DESPUES DE HABERLO EJECUTADO**

De esta forma podemos imprimir casi cualquier cosa: listados, contenidos de tablas, estructura de tablas, etc...

Para imprimir utilizaremos el siguiente icono:

 ⟶ Imprimir o SHIFT+F1

Este icono lo podemos ver en casi todas las pantallas de SAP, si lo pulsamos nos saldrá la siguiente pantalla:

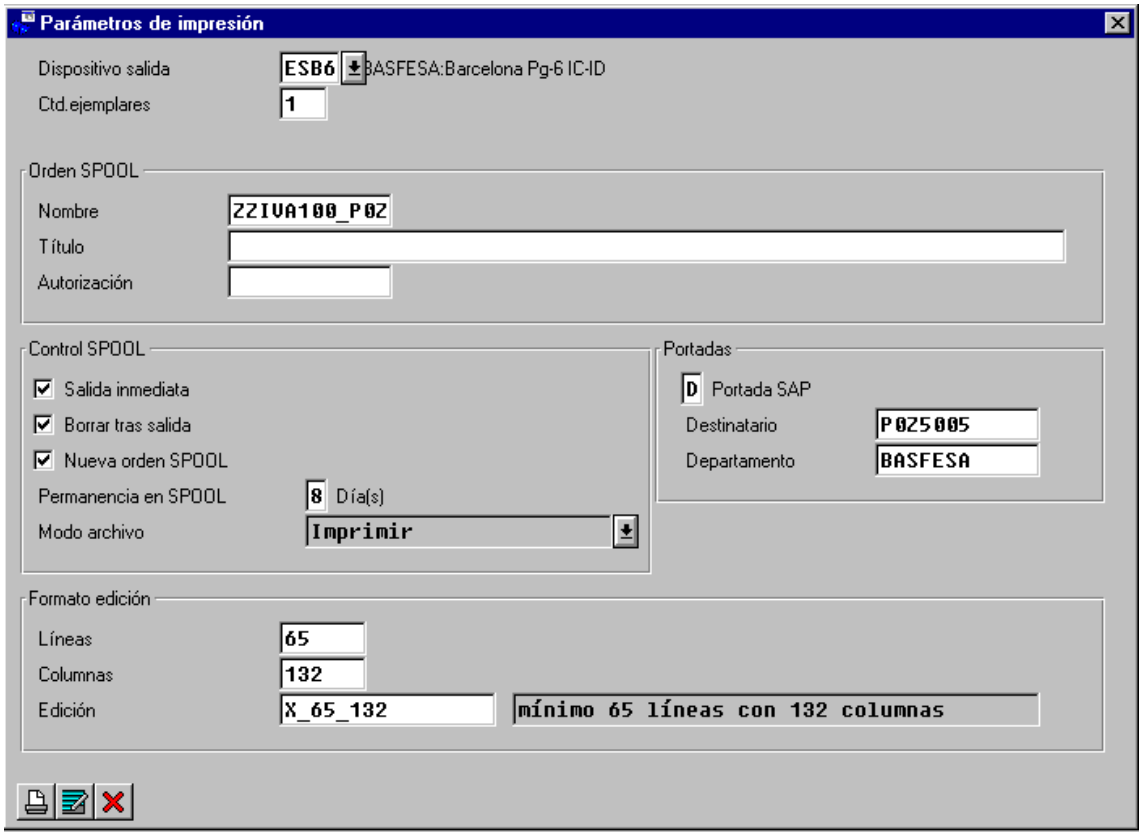


Fig. Ventana impresión.

En esta pantalla controlaremos los parámetros de nuestra impresión, desde que impresora utilizaremos hasta las líneas y columnas de la impresión. Esta pantalla sale siempre que queramos imprimir un informe, una tabla, o cualquier cosa desde SAP.

**MIENTRAS SE EJECUTA**

Esta segunda forma se realiza a través de un ABAP y se suele utilizar para imprimir listados.

Podemos imprimir de tres formas diferentes: llamando a un report, ejecutar e imprimir e imprimir directamente.

En las dos primeras formas hemos de llamar a un función para poner las propiedades de la impresión, la pantalla que nos saldrá es la que hemos visto anteriormente (Véase Fig. Ventana impresión), aunque podemos hacer que no se muestre. La función sería la siguiente:

```
Call function 'GET_PRINT_PARAMETERS'
    Exporting
      Layout              = 'X_65_132'
      Line_count          = 65
```

Parametros opcionales

```
      Line_size            = 132
      No_dialog            = 'X'
   Importing
      Out_parameters       = PRIPAR
      Out_archive_parameters = ARCPAR
      Valid                = VAL
   Exceptions
      Archive_info_not_found = 1
      Invalid_print_params   = 2
      Invalid_archive_params = 3
      Others               = 4.
```

Esta función tiene muchos parámetros, pero normalmente se suele utilizar estos.

En “Exporting” tenemos:

- Line_count -> Número de líneas por página.
- Line_size -> Tamaño de cada línea.
- No_dialog -> Si esta en blanco quiere decir que se va a mostrar la pantalla de impresión (Véase Fig. Ventana impresión), si contiene una ‘X’ no se mostrará la pantalla de impresión.

En “Importing” tenemos:

- Out_parameters -> Devuelve los valores de cómo se va a imprimir. Le pasamos una variable con la estructura “pri_params”.
- Out_archive_parameters -> Devuelve los valores de cómo se va a imprimir en un archivo. Le pasamos una variable con la estructura de “arc_params”.
- Valid -> Si devuelve un espacio es que algún valor no es válido y se devuelve cualquier otra cosa los datos son validos. Se le pasa una variable de tipo char de longitud uno.

**LLAMANDO A UN REPORT**

Podemos imprimir un listado llamando a otro programa que es la realiza el listado, pero en vez de sacarlo por pantalla lo sacará por impresora. Para hacerlo utilizaremos la orden SUBMIT...TO SAP-SPOOL. Cuya sintaxis es la siguiente:

```
SUBMIT <programa> TO SAP-SPOOL
        [<parametros>|SPOOL PARAMETERS <pripar>]
        [ARCHIVE PARAMETERS <arcpar>]
        [WITHOUT SPOOL DYNPRO].
```

Las opciones que estan en corchetes son opcionales

- SPOOL PARAMETERS -> Son los parámetros de impresión.
- ARCHIVE PARAMETERS -> Son los parámetros de impresión de un archivo.
- WITHOUT SPOOL DYNPRO -> No nos mostrará la pantalla de impresión (Véase fig. Ventana impresión). Si no ponemos esta opción SAP nos mostrará automáticamente la ventana de impresión

Un ejemplo de cómo la utilizaríamos sería de la siguiente forma:

```
report zziva140.

submit zziva130 to sap-spool
         without spool dynpro.
```

Aquí es lo que hago es que imprimo el listado que saca el programa ZZIVA130 directamente por impresora sin mostrar pantalla de impresión.

Si queremos que nos salga de una forma más adecuada, se haría con el siguiente ejemplo:

```
report zziva140.

Data: val,
      pripar like pri_params,
      arcpar like arc_params.

Call function 'GET_PRINT_PARAMETERS'
    Exporting
       Layout              = 'X_65_132'
       Line_count          = 65
       Line_size           = 132
       No_dialog           = 'X'
    Importing
       Out_parameters      = pripar
       Out_archive_parameters = arcpar
       Valid               = VAL
    Exceptions
       Archive_info_not_found = 1
       Invalid_print_params   = 2
       Invalid_archive_params = 3
       Others              = 4.

if val <> space and sy-subrc = 0. ⟶  Compruebo que no existe nigún error
   submit zziva130 to sap-spool
            spool   parameters pripar
           archive parameters arcpar
           without spool dynpro.
endif.
```

En este ejemplo primero muestro la pantalla de impresión (Véase fig. Ventana impresión), para que el usuario pueda escoger como quiere realizar la impresión una vez escogida.
El “if” después de la función sirve para controlar de que todo ha ido correcto, si es todo correcto me dispongo a imprimir.

**EJECUTAR E IMPRIMIR**

Este método suele ser el más común de todos. Lo primero que se suele hacer es llamar a la función "GET_PRINT_PARAMETERS" para que el usuario pueda introducir las propiedades de impresión.
Después de esto validaremos que todo es correcto, si lo es, pondremos la instrucción NEW-PAGE PRINT... Con esta instrucción haremos que cuando hagamos visualizaremos algo en vez de salir por pantalla salga por la impresora. La sintaxis de la instrucción sería la siguiente:

```
New-page print on
  [new-section]
  [parameters PRIPAR]
  [archive parameters ARCPAR]
  [no dialoag].
```

- new-section -> Pone la variable del sistema sy-pagno (número de página) a 1.
- Parameters ... -> Parametros de impresión, se le pasa una variable con la estructura pri_params.
- archive parameters... -> Parametros de impresión por archivo, se le pasa una variable con la estructura arc_params.
- no dialog -> No muestra la pantalla de impresión (Véase fig. ventana de propiedades)

Para desactivar esta orden utilizaremos la orden: NEW-PAGE PRINT OFF.

Un ejemplo completo de cómo se imprimiria se haria de la siguiente forma:

```
REPORT ZZIVA130 LINE-SIZE 130.
*---------------------------------------------------------------------
* tablas de diccionario
*---------------------------------------------------------------------
TABLES: ZZTABPRU10.
*---------------------------------------------------------------------
* variables del programa
*---------------------------------------------------------------------
DATA: VAL,
    PRIPAR LIKE PRI_PARAMS,
    ARCPAR LIKE ARC_PARAMS.

* llama a un procedimiento para que prepare la impresion
PERFORM PREPARAR_IMPRESION.
* Comienza el listado
SELECT * FROM ZZTABPRU10.

 WRITE: /3 ZZTABPRU10-KUNNR, 17 ZZTABPRU10-BUKRS, 26 ZZTABPRU10-
BANKS,
      35 ZZTABPRU10-BANKL, 52 ZZTABPRU10-BANKN, 72 ZZTABPRU10-
BKONT,
```

```
        83 ZZTABPRU10-BVTYP, 93 ZZTABPRU10-BKREF,
        121 ZZTABPRU10-XEZER.
ENDSELECT.
* Cuando termina el listado desactivo el modo de impresión
NEW-PAGE PRINT OFF.
*---------------------------------------------------------------------
* Encabezado de página
*---------------------------------------------------------------------
TOP-OF-PAGE.
  WRITE: /3 'Deudor', 15 'Sociedad', 25 'País', 35 'Código'
        , 52 'Cuenta', 71 'Clave', 80 'Tipo banco', 96 'Referencia',
        116 'Autorización'.
  WRITE: /25 'banco', 35 'bancario', 52 'bancaria', 71 'control',
        80 'colaborador', 116 'domiciliaria'.
ULINE.
*&---------------------------------------------------------------------*
*&      Form  PREPARAR_IMPRESION
*&---------------------------------------------------------------------*
*       text                                                          *
*----------------------------------------------------------------------*
FORM PREPARAR_IMPRESION.
* Saco las opciones del listado
CALL FUNCTION 'GET_PRINT_PARAMETERS'
     EXPORTING
        LAYOUT               = 'X_65_132'
        LINE_COUNT             = 65
        LINE_SIZE            = 132
    IMPORTING
        OUT_PARAMETERS         = PRIPAR
        OUT_ARCHIVE_PARAMETERS = ARCPAR
        VALID                = VAL
    EXCEPTIONS
        ARCHIVE_INFO_NOT_FOUND = 1
        INVALID_PRINT_PARAMS   = 2
        INVALID_ARCHIVE_PARAMS = 3
         OTHERS                = 4.

* Si todos los datos son correctos, activo el modo de impresión
IF VAL <> SPACE AND SY-SUBRC = 0.
   NEW-PAGE PRINT ON
   NEW-SECTION
   PARAMETERS PRIPAR
   ARCHIVE PARAMETERS ARCPAR
   NO DIALOG.
ENDIF.
ENDFORM.                    " PREPARAR_IMPRESION
```

Como vemos en este ejemplo llamo a un procedimiento para que me preparé la impresión, primero mostrando la ventana de impresión y segundo activando el modo de impresión si todos los datos son correctos.

Después con el "SELECT" voy mostrando los registros de la tabla, como esta activado el modo de impresión, por la impresora, al principio de todo se muestra un encabezado que también se imprime.

Cuando termina el "SELECT" desactivo el modo de impresión, para que cuando vuelve ha hacer un "WRITE" me lo saque por pantalla en vez de por la impresora.

**IMPRIMIR DIRECTAMENTE**

Se realiza con la orden NEW-PAGE PRINT... que esta explicada con anterioridad. Para imprimir directamente escribiremos:

```
NEW-PAGE PRINT ON NO DIALOG.
```

Y a partir de entonces, cuando hagamos un"WRITE" nos lo sacara por la impresora en vez de por la pantalla.

Cuando queramos que no imprima más, escribiremos:

```
NEW-PAGE PRINT OFF.
```

**GRAFICOS EN SAP**

Desde SAP tenemos un amplio surtido de abánicos, pero que a su debida compleijada solo explicaremos solo tres funciones para hacer gráficos pertenecientes todas ellos al grupo "SAP Business Graphics".
El resto de funciones gráficas les remito a que consulten los ejemplos que SAP incluye;

| Programas | Descripción |
|---|---|
| GRBUSGxx | Ejemplos de gráficos en 2D, 3D y 4D. |
| GRHIERxx | Ejemplos de gráficos jerarquicos |
| GRSTATxx | Ejemplos de gráficos de funciones trigonométricas y estadísticas |
| GRPORTxx | Ejemplo de gráficos de portafolios |

**GRAFICOS EN DOS DIMENSIONES**

Para hacer gráficos en dos dimenensiones tenemos la función "GRAPH_2D". Esta función posee bastantes opciones pero solo explicaré las necesarias para hacer un simple gráfico. La sintaxis seria la siguiente:

```
Call function 'GRAPH_2D'
    Exporting
        Display_type = '' ‘
        Mail_allow = '' ‘
        Titl       = ' '
        Winpos = ' '
```

```
        Winszx = '50'
        Winszy = '50'
    Tables
        Data      =.
```

En Exporting tenemos:

- Display_type -> Tipo de grafico que se va a visualizar inicialmente, tenemos las siguientes posibilidades:

  - 'VB' -> Vertical Bar.
  - 'VS' -> Stacked Vertical Bars.
  - 'HO' -> Horizontal Bars.
  - 'HS' -> Stacked Horizontal Bars.
  - 'TD' -> Perspective Bars.
  - 'VT' -> Vertical Triangles.
  - 'ST' -> Stepped lines.
  - 'MS' -> Stepped Areas.
  - 'LN' -> Lines.
  - 'SA' -> Stacked Areas.
  - 'MA' -> Shaded Areas.
  - 'PI' -> Pie Chart.
  - 'TP' -> Perspective Pie Chart.
  - 'PO' -> Polar diagram.
  - 'PP' -> Polar relative.

- Mail_allow -> Si lleva una 'X' indica que el gráfico puede ser enviado a través del correo de SAPOffice.
- Titl -> Es título del gráfico.
- Winpos -> Indica en que posición se va a poner el gráfico la primera vez que se visualize, las posibles posiciones son estas:

  - SPACE -> Sin especificar.
  - '1' -> Arriba a la izquierda.
  - '2' -> Arriba al centro.
  - '3' -> Arriba a la derecha.
  - '4' -> Centro a la izquierda.
  - '5' -> Centro al centro.
  - '6' -> Centro a la derecha.
  - '7' -> Abajo a la izquierda.
  - '8' -> Abajo al centro.
  - '9' -> Abajo a la derecha.

- Winszx -> Tamaño 'x' de la ventana en %, por defecto tiene el valor 50, si ponemos esta opción hemos de poner la claúsula "winpos" ya que si no, no tendrá efecto lo que pongamos.
- Winszy -> Tamaño 'y' de la ventana en %, por defecto tiene el valor 50. Como en el caso anterior hemos de poner también la claúsula "winpos" para que los cambios tengán efecto.

En Tables tenemos:

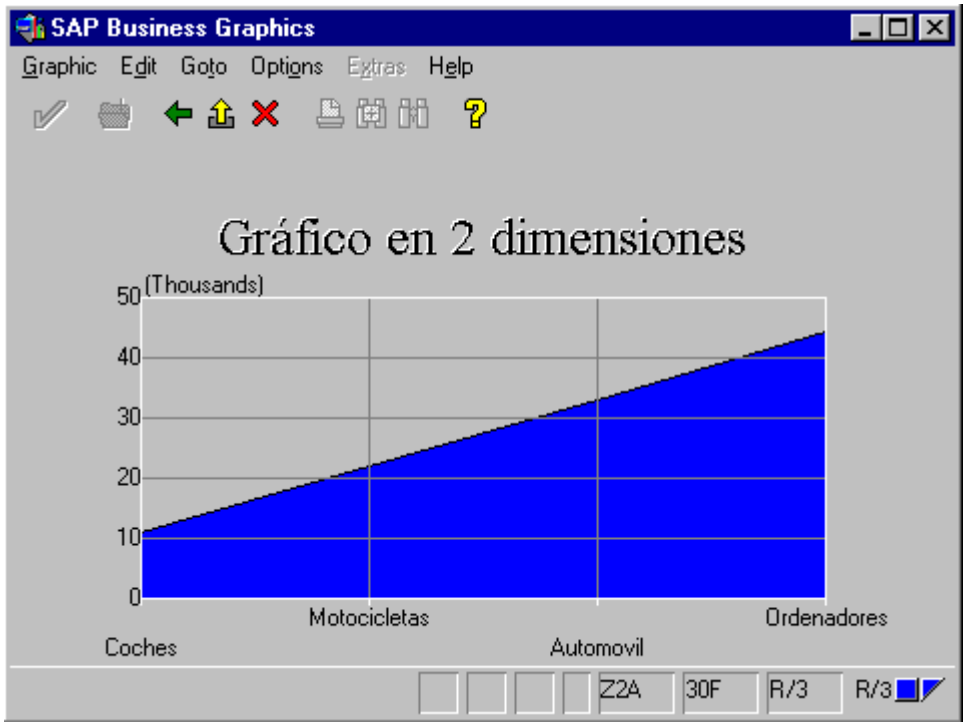
- Data es el nombre de la tabla interna con los valores a representar gráficamente, la estructura ha de ser la siguiente:

```
Data: Begin of data occurs 1,
      TEXT(25),
      VALUE Type P,
     End of data
```

Un ejemplo de esta función sería el siguiente:

```
Report ZZIVAN140.

Data: Begin of data occurs 1,
            TEXT(25),
            VALUE Type P,
      End of data



Data-text = 'Coches'.
Data-value = 11111.
Append data.

Data-text = 'Motocicletas'.
Data-value = 22222.
Append data.

Data-text = 'Automovil'.
Data-value = 33333.
Append data.

Data-text = 'Ordenadores'.
Data-value = 44444.
Append data.

Call function 'GRAPH_2D'
     Exporting
        Display_type = 'SA'
        Mail_allow = ''
        Titl       = 'Gráfico en 2 dimensiones'
        Winpos = '5'
        Winszy = '60'
        Winszy = '60'
     Tables
        Data       = DATA.
```

El resultado en pantalla sería el siguiente :




## GRAFICOS EN TRES DIMENSIONES

Para hacer gráficos en tres dimensiones tenemos la función “GRAPH_3D”, como en el caso anterior, solo explicaré las opciones necesarias para hacer un sencillo gráfico en tres dimensiones. La sintaxis sería la siguiente:

```
Call function 'GRAPH_3D'
      Exporting titl       = ' '
                display_type = ' '
                dim1       = ' '
                dim2       = ' '
                mail_allow = ' '
                winpos = ' '
                winszx = '50'
                winszy = '50'
     Tables    data = .
```

En Exporting tenemos:

- Display_type -> Tipo de grafico que se va a visualizar inicialmente, tenemos las siguientes posibilidades:

  - ‘TO’ -> Torres.
  - ‘PY’ -> Pyramids.
  - ‘HO’ -> Edges.
  - ‘HS’ -> Wedges.
  - ‘TD’ -> Network.

- Dim1 -> El título de la primera dimensión.
- Dim2 -> El título de la segunda dimensión.
- Mail_allow -> Si lleva una 'X' indica que el gráfico puede ser enviado a través del correo de SAPOffice.
- Titl -> Es título del gráfico.
- Winpos -> Indica en que posición se va a poner el gráfico la primera vez que se visualize, las posibles posiciones son estas:

    - SPACE -> Sin especificar.
    - '1' -> Arriba a la izquierda.
    - '2' -> Arriba al centro.
    - '3' -> Arriba a la derecha.
    - '4' -> Centro a la izquierda.
    - '5' -> Centro al centro.
    - '6' -> Centro a la derecha.
    - '7' -> Abajo a la izquierda.
    - '8' -> Abajo al centro.
    - '9' -> Abajo a la derecha.

- Winszx -> Tamaño 'x' de la ventana en %, por defecto tiene el valor 50, si ponemos esta opción hemos de poner la claúsula "winpos" ya que si no, no tendrá efecto lo que pongamos.
- Winszy -> Tamaño 'y' de la ventana en %, por defecto tiene el valor 50. Como en el caso anterior hemos de poner también la claúsula "winpos" para que los cambios tengán efecto.

En Tables tenemos:

- Data es el nombre de la tabla interna con los valores a representar gráficamente, la estructura ha de ser la siguiente:

```
data: begin of data occurs 1,
        Text(25),
        Value1 type p,
        Value2 type p,
        Value3 type p,
        Value4 type p,
End of data.
```

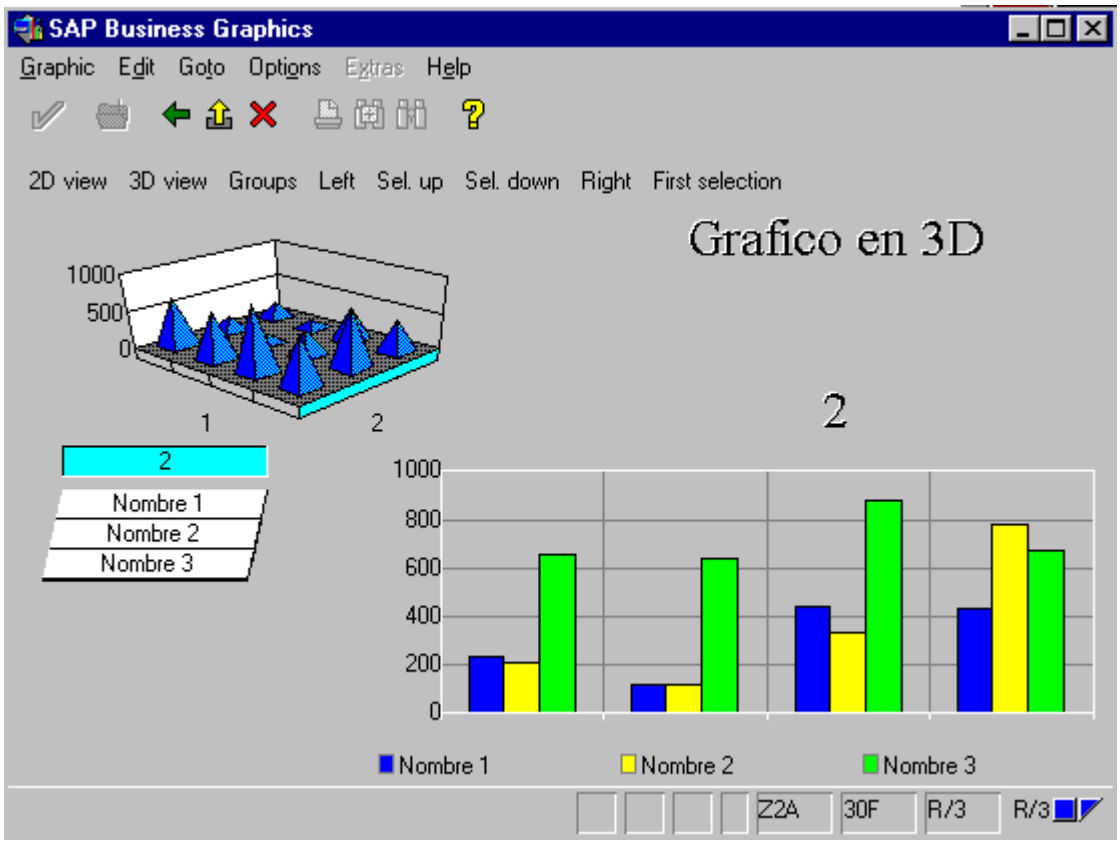
Podemos colocar hasta seis "value", si ponemos más no dará error pero no saldrá por pantalla.

Un ejemplo de esta función sería la siguiente:

```
deport ZZIVA150.

data: begin of data occurs 1,
        Text(25),
        Value1 type p,
        Value2 type p,
```

```
        Value3 type p,
        Value4 type p,
    End of data.

data-text = 'Nombre 1'.
data-value1 = 232.
data-value2 = 121.
data-value3 = 444.
data-value4 = 433.
append data.
data-text = 'Nombre 2'.
data-value1 = 212.
data-value2 = 123.
data-value3 = 331.
data-value4 = 783.
append DATA.
data-text = 'Nombre 3'.
data-value1 = 656.
data-value2 = 643.
data-value3 = 881.
data-value4 = 673.
append data.

Call function 'GRAPH_3D' Exporting titl       = 'Grafico en 3D'
                display_type = 'PY'
                dim1       = '1'
                dim2       = '2'
                mail_allow = ' '
                winpos = '5'
                winszx = '70'
                winszy = '70'
     Tables    data = data.
```

Y el resultado en pantalla sería el siguiente:




**GRAFICOS EN 2D, 3D Y 4D.**

Con esta función podemos crear gráficos de dos, tres y cuatro dimensiones, sobre en esta función explicaré como se realiza un gráfico con esta función de manera sin muchas complicaciones. La sintáxis seria la siguiente:

```
call function 'GRAPH_MATRIX'
      exporting
          titl  = ' '
          valt  = ' '
          dim1  = ' '
          dim2  = ' '
          dim3  = ' '
          winpos = ' '
          winszx = '50'
          winszy = '50'
      tables
          data  =
          tdim1 =
          tdim2 =
          tdim3 =
          opts  =.
```

En Exporting tenemos:

- Dim1 -> Título de la primera dimensión.
- Dim2 -> Título de la segunda dimensión.
- Dim3 -> Título de la segunda dimensión.
- Mail_allow -> Si lleva una 'X' indica que el gráfico puede ser enviado a través del correo de SAPOffice.
- Titl -> Es título del gráfico.
- Valt -> Es la unidad de medida.
- Winpos -> Indica en que posición se va a poner el gráfico la primera vez que se visualize, las posibles posiciones son estas:

  - SPACE -> Sin especificar.
  - '1' -> Arriba a la izquierda.
  - '2' -> Arriba al centro.
  - '3' -> Arriba a la derecha.
  - '4' -> Centro a la izquierda.
  - '5' -> Centro al centro.
  - '6' -> Centro a la derecha.
  - '7' -> Abajo a la izquierda.
  - '8' -> Abajo al centro.
  - '9' -> Abajo a la derecha.

- Winszx -> Tamaño 'x' de la ventana en %, por defecto tiene el valor 50, si ponemos esta opción hemos de poner la claúsula "winpos" ya que si no, no tendrá efecto lo que pongamos.
- Winszy -> Tamaño 'y' de la ventana en %, por defecto tiene el valor 50. Como en el caso anterior hemos de poner también la claúsula "winpos" para que los cambios tengán efecto.

En Tables tenemos:

- Data es el nombre de la tabla interna con los valores a representar gráficamente, la estructura ha de ser la siguiente:

```
data: begin of data occurs 1,
        p type p,
end of data.
```

- Tdim1, Tdim2 y tdim3: Son los nombres de la primera, segunda y tercera dimensión respectivamente. Cuya estructura es la siguiente:

```
data: begin of tdim occurs 1,
        c(80) type c,
  end of tdim.
```

- Opts -> Son las opciones del gráfico a visualizar. La estructura ha de ser la siguiente:

```
data: begin of opts occurs 1,
```

```
          c(80) type c,
        end of opts.
```

Un ejemplo sería el siguiente:

```
report zziva160.
data: begin of data occurs 1,
       p type p,
     end of data.

*--- optionen-tabelle ----------------------------------------------*
data: begin of opts occurs 1,
        c(80) type c,
     end of opts.

data: begin of tdim1 occurs 1,
        c(80) type c,
     end of tdim1.

data: begin of tdim2 occurs 1,
        c(80) type c,
     end of tdim2.

data: begin of tdim3 occurs 1,
        c(80) type c,
     end of tdim3.


data: tyear1(5) value '#1991',
      tyear2(5) value '#1992',
      tyear3(5) value '#1993',
      tyear4(5) value '#1994'.

data: tprod1(9),
      tprod2(9),
      tprod3(9),
      tprod4(9),
      tprod5(9).

tprod1 = 'prod 1'.
tprod2 = 'prod 2'.
tprod3 = 'prod 3'.
tprod4 = 'prod 4'.
tprod5 = 'prod 5'.

data: tland1(20),
      tland2(20),
      tland3(20),
      tland4(20),
```

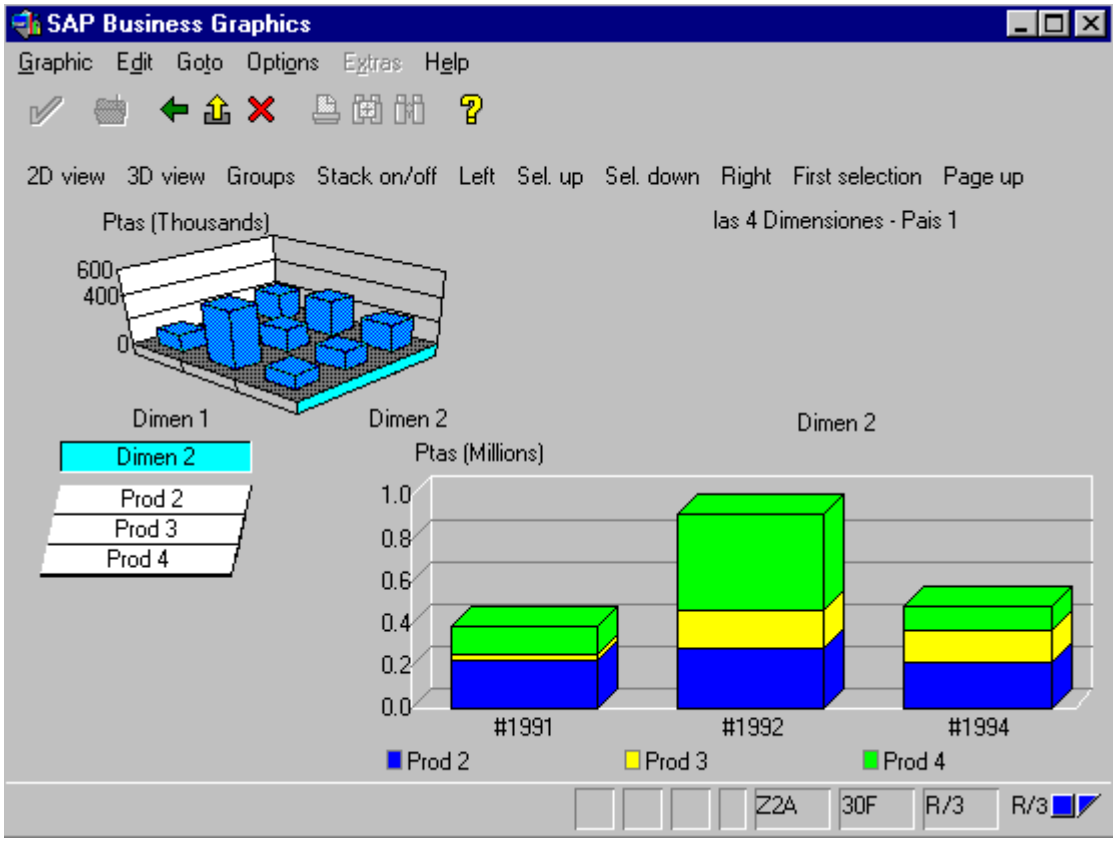
```
      tland5(20),
      tland6(20).

tland1 = 'pais 1'.
tland2 = 'pais 2'.
tland3 = 'pais 3'.
tland4 = 'pais 4'.
tland5 = 'pais 5'.
tland6 = 'pais 6'.

perform fill_data.

  refresh opts.
*--- erstes bild: auswaehlen ----------------------------------------*
  write 'fifrst = pu' to opts-c. append opts.
*--- 2d-graphiktyp: perspektivische balken ----------------------------*
  write 'p2type = td' to opts-c. append opts.
*--- art der faerbung: gleichmaessig ----------------------------------*
  write 'p3ctyp = pl' to opts-c. append opts.

*--- 1ª dimensión
  refresh tdim1.
  move tyear1 to tdim1.
  append tdim1.
  move tyear2 to tdim1.
  append tdim1.
  move space  to tdim1.
  append tdim1.
  move tyear4 to tdim1.
  append tdim1.
*--- 2ª dimensión
  refresh tdim2.
  move space  to tdim2.
  append tdim2.
  move tprod2 to tdim2.
  append tdim2.
  move tprod3 to tdim2.
  append tdim2.
  move tprod4 to tdim2.
  append tdim2.

*--- 3ª dimensión
  refresh tdim3.
  move tland1 to tdim3.
  append tdim3.
  move space  to tdim3.
  append tdim3.
  move tland3 to tdim3.
  append tdim3.
```

```
  call function 'graph_matrix'
      exporting
          titl  = 'las 4 dimensiones'
          valt  = 'ptas'
          max1 = '4'
          max2 = '4'
          max3 = '4'
          dim1  = 'dimen 1'
          dim2  = 'dimen 2'
          dim3  = 'dimen 3'
          winpos = '5'
          winszx = '70'
          winszy = '70'
      tables
          data  = data
          tdim1 = tdim1
          tdim2 = tdim2
          tdim3 = tdim3
          opts  = opts.

*&---------------------------------------------------------------------*
*&      form  fill_data
*&---------------------------------------------------------------------*
*       text                                                          *
*----------------------------------------------------------------------*
form fill_data.
  data-p = 153470.
  append data.
  data-p = 243470.
  append data.
  data-p = 124567.
  append data.
  data-p = 179037.
  append data.
  data-p = 234980.
  append data.
  data-p = 287513.
  append data.
  data-p = 253430.
  append data.
  data-p = 223440.
  append data.
  data-p =  24567.
  append data.
  data-p = 180037.
  append data.
  data-p = 129830.
  append data.
  data-p = 145530.
  append data.
```

```
data-p = 132470.
append data.
data-p = 453470.
append data.
data-p =  24456.
append data.
data-p = 119807.
append data.
data-p = 288710.
append data.
data-p = 166656.
append data.
data-p = 300430.
append data.
data-p = 723110.
append data.
data-p =  22767.
append data.
data-p = 195522.
append data.
data-p =  38970.
append data.
data-p =  89635.
append data.
data-p = 166970.
append data.
data-p = 401470.
append data.
data-p =  29967.
append data.
data-p = 112957.
append data.
data-p =  37860.
append data.
data-p =  77450.
append data.
data-p = 253150.
append data.
data-p = 343570.
append data.
data-p = 768867.
append data.
data-p = 236790.
append data.
data-p = 122750.
append data.
data-p = 328760.
append data.
data-p = 292150.
append data.
```

```
  data-p = 356570.
  append data.
  data-p = 268867.
  append data.
  data-p =  36790.
  append data.
  data-p = 125680.
  append data.
  data-p = 178893.
  append data.
  data-p = 333150.
  append data.
  data-p = 373570.
  append data.
  data-p = 168867.
  append data.
  data-p = 226790.
  append data.
  data-p = 278940.
  append data.
  data-p = 177784.
  append data.
endform.
```

Como vemos el llenado de la tabla es largo, en este caso se han introducido 40 registros. No se porqué pero si se introducen menos de 40 da error de que no hay datos suficientes. El resultado por pantalla sería el siguente:




**INSTRUCCIONES**

**INSTRUCCIONES DE CONTROL DE FLUJO**

Tenemos 5 instrucciones de control de flujo y son las siguientes:

**IF**

IF condición

INSTRUCCIONES

ELSEIF condición → Los ELSEIF no necesitan ENDIF

INSTRUCCIONES

( ELSE

INSTRUCCIONES )

ENDIF

Los operandos que podemos utilizar en la condición, son los de siempre: AND, OR, NOT y XOR.

Para comparar dos datos se utilizan los siguientes operadores:

- EQ-> '='
- GT-> '>'
- LT-> '<'
- GE-> '>='
- LE-> '<='
- NE-> '<>'

Para saber si un campo es nulo se utilizaría la siguiente instrucción: IS INITIAL. Ejemplo:

```
IF código IS INITIAL.
   WRITE: / 'El código es nulo'.
ELSE
   INSTRUCCIONES.
END-IF.
```

**DO**

DO condición(es) o n TIMES

INSTRUCCIONES

ENDDO

Aquí se ejecutan las instrucciones hasta que se cumpla la condición, o bien podemos poner que el bucle se repita n veces.

**WHILE**

WHILE condición(es)

INSTRUCCIONES

ENDWHILE

Aquí, mientras se cumpla la condición ejecutará las instrucciones que estén dentro del bucle.

**CASE**

```
CASE variable
    WHEN condición(es)
            INSTRUCCIONES
     WHEN OTHERS
            INSTRUCCIONES
ENDCASE
```

En este caso, dependiendo del valor de una variable, hará una rama u otra, y si no hay ninguna instrucción que cumpla la condición, entonces hará las instrucciones que estén en WHEN OTHERS.

**CHECK**

CHECK sería como hacer un IF y un CONTINUE todo junto. La sintaxis de esta instrucción sería la siguiente:

```
CHECK condición.
```

Su funcionamiento lo explicaré de una forma visual que se entenderá mejor. El primer ejemplo sería en un módulo:

```
MODULE MODIFY_SCREEN OUTPUT.
    CHECK MODO = 'PRUEBA'
    LOOP AT SCREEN.
            .
            .
ENDMODULE.
```

Si no se cumple la condición se va fuera del módulo.

Si se cumple la condición continua con las instrucciones que tiene debajo.

Otro ejemplo se daría en un LOOP su funcionamiento sería el mismo que en un módulo.

```
LOOP AT SCREEN.
    CHECK SCREEN-GROUP1 = 'IVA'.
    SCREEN-INPUT = 0.
    MODIFY SCREEN.
ENDLOOP.
```

Si se cumple

No se cumple la condición

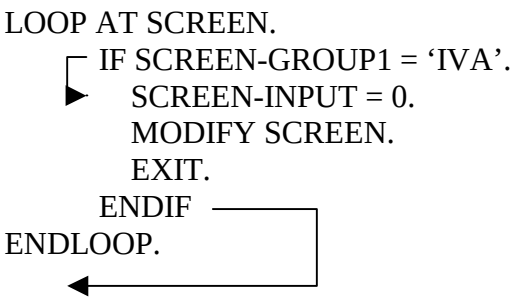
Aquí cuando no se cumple la condición va al final del LOOP, si se cumple continua con las instrucciones que hay debajo.

## INSTRUCCIONES DE RUPTURA DE UN CONTROL DE FLUJO

Tenemos dos instrucciones que nos permiten alterar el control de las instrucciones de flujo, son las siguientes: EXIT y CONTINUE.

### EXIT

Esta instrucción sirve para salir de un bucle de cualquier tipo (WHILE, IF, CASE, DO, LOOP, etc.), su funcionamiento gráficamente sería el siguiente:




Como vemos el EXIT hace que salga inmediatamente del LOOP, o de cualquier otra instrucción de bucle.

### CONTINUE

El CONTINUE lo que hace es que va al final del bucle, pero no sale de él, sin ejecutarse las instrucciones que hay debajo del CONTINUE. Su funcionamiento sería el siguiente:

```
DO 4 TIMES.

  IF SY-INDEX = 2.
    CONTINUE.
  ENDIF.
  WRITE SY-INDEX.

ENDDO.
```

El resultado por pantalla sería el siguiente:

```
1       3       4
```

Como vemos el 2 no se visualiza por que cuando es 2 hacemos un CONTINUE y por lo tanto no hace el WRITE.

LECTURA DE TABLAS DE DICCIONARIO

Para leer este tipo de tablas se utiliza la orden SELECT. Su sintaxis es la siguiente:

SELECT[ SINGLE ] campo1 campo2 campoN o *

[INTO CORRESPONDING FIELDS OF TABLE tabla-interna ]

FROM tabla-diccionario

[WHERE campo_tabla[ BETWEEN valor_1 AND valor_2 ] ]

[ORDER BY campo1 campo2 campoN ]

INSTRUCCIONES

ENDSELECT.

IMPORTANTE: La última opción antes de las INSTRUCCIONES lleva punto.

Si queremos leer todos los campos de una tabla de diccionario pondremos él '*'.

SINGLE->sólo lee un registro y el ENDSELECT no se pone

INTO CORRESPONDING FIELDS OF TABLE tabla_interna-> Mueve los campos de una tabla de diccionario a una tabla interna (Los campos han de tener el mismo nombre y ser del mismo tipo).

FROM tabla_diccionario-> De que tabla de diccionario leemos.

WHERE campo_tabla-> Leeremos los registros que cumplan esa condición o condiciones.
La opción BETWEEN sirve para indicarle un rango de valores comprendidos entre valor1 y valor2. Siempre va después del WHERE.

En la orden WHERE podemos comparar máscaras, por ejemplo:

WHERE campo_tabla LIKE (máscara)

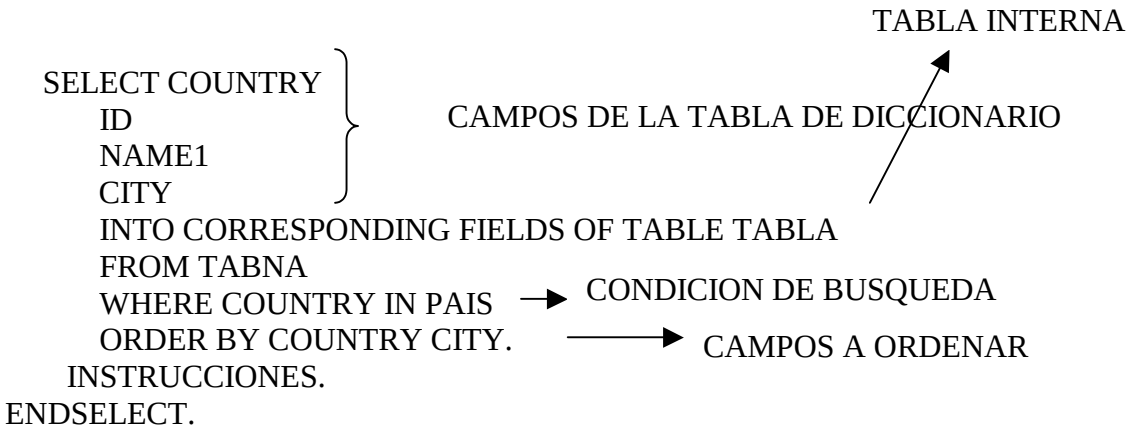
La máscara puede ser por ejemplo: '_AAA' o 'BBB%'

'_' -> un valor cualquiera, '%'-> una cadena de valores cualesquiera.

ORDER BY campo1 campo2-> Podemos sacar la lectura ordenada por los campos que queramos.

GROUP BY ->No sé como se utiliza.

Un ejemplo de SELECT sería el siguiente:




**CONSEJOS**

Cuando dentro de un "select" llamamos a forms, y en ellos utilizamos variables que significan lo mismo o tiene el mismo significado que algún campo que utilizamos en la tabla del "select" es más conveniente utilizar los campos del la tabla del "select".

LECTURA DE TABLAS INTERNAS

Se utiliza la orden LOOP, su sintaxis sería la siguiente:

```
LOOP AT tab-int ( OPCIONES )
      INSTRUCCIONES
ENDLOOP
```

Otra forma de leer una tabla interna se realiza con la orden READ...

También para leer una sola vez una tabla interna, se utiliza la instrucción:

```
READ TABLE nombre_tabla_interna
```

Presentando las siguientes variantes

```
READ TABLE nombre_tabla_interna WITH KEY 'xxxx'.
```

Lee la tabla comparando carácter a carácter el principio de las entradas de la tabla con el argumento de búsqueda.

```
READ TABLE nombre_tabla_interna BY SY-INDEX.
```

SY-INDEX número de la línea que queremos leer, por defecto lee la primera.

```
READ TABLE nombre_tabla_interna INDEX num_reg.
```

Lee un registro en concreto, indicado en num_reg. Muy utilizado cuando hacemos paginación.

Para saber cuantos registros o más concretamente líneas tiene una tabla interna utilizaríamos la orden DESCRIBE, cuya sintaxis es la siguiente:

```
DESCRIBE TABLE tabla_interna LINES var_num_lineas.
```

Nombre de la tabla

Variable donde guardamos el número de líneas

## INSTRUCCIONES DE RUPTURA

### AT FIRST

Cuando se lee el primer registro de la tabla interna, se ejecutan las órdenes que hay dentro.

```
AT FIRST.
      INSTRUCCIONES.
ENDAT.
```

### AT NEW

Cuando cambio el valor de un campo y es el primer valor, se ejecutan las instrucciones que hay dentro. Para que eso suceda, la tabla ha de estar ordenada por ese campo.

```
AT NEW campo.
      INSTRUCCIONES.
ENDAT.
```

### AT END OF

Cuando es el último valor de un grupo de campos, se ejecutan las órdenes que hay dentro. Para que eso suceda, la tabla ha de estar ordenada por ese campo.

```
AT END OF campo.
      INSTRUCCIONES.
ENDAT.
```

Se suele poner detrás de un AT NEW que tenga un campo en común.

### AT LAST

Cuando es el último registro de la tabla, se ejecutan las órdenes que hay dentro.

```
AT LAST OF campo.
```

```
        INSTRUCCIONES.
ENDAT.
```

**CONSEJOS**

Podemos usar la sentencia SUM para sumar las cantidades de los registros de la tabla interna que hayan intervenido en el evento.

Cuando realizamos un AT NEW... sobre un campo hay que saber que los campos que estén situados a la derecha pierden su valor. Para comprenderlo mejor veamos un ejemplo:

Imaginemos que tenemos la siguiente tabla con los siguientes datos:

| CODIGO | NOMBRE | APELLIDO | OTROS DATOS |
|---|---|---|---|
| 0001122 | JOSE ANTONIO | PEREZ | CAMINERO |
| 0002231 | FERNANDO | MORIENTES | EXMAÑICO |
| 0022123 | ERIC | CANTONA | THE KING |

Y realizamos un AT NEW del campo “nombre”, cuando estubieramos dentro del AT NEW.. los datos de la tabla quedaría así:

| CODIGO | NOMBRE | APELLIDO | OTROS DATOS |
|---|---|---|---|
| 0001122 | JOSE ANTONIO | PEREZ | CAMINERO |
| 0002231 | FERNANDO | | |
| 0022123 | ERIC | CANTONA | THE KING |

Aquí se produce la ruptura

Como vemos cuando se produce la ruptura, los valores de los campos que están situados a la derecha se pierden.

Para solucionar esto, se puede hacer de dos formas:

- La primera es realizar una ruptura sin utilizar la orden AT NEW ... , o sea a mano, esto conlleva que se tenga que utilizar más variables y controlar todos los casos.
- La segunda es más fácil, solo utilizaríamos una variable para saber si hemos entrado en la ruptura o no. El esquema de cómo lo haríamos sería el siguiente:

```
Entrado = False.
LOOP AT MI_TABLA.
    AT NEW nombre.
        Entrado = True.
    ENDAT.
    IF Entrado.
```

```
            WRITE: / Nombre,  Apellido, Otros_datos
            Entrado = False.
        ENDIF.
    ENDLOOP.
```

Como veis solo utilizaríamos una variable para controlar la ruptura. Antes de entrar en el LOOP la variable vale "False" (No ha entrado) y cuando entre en el AT NEW le pondré el valor "True" (Ha entrado) y cuando salga del AT NEW mirare con un IF si he entrado, si es así visualizare los campos que quiera (o cualquier otra operación que queramos) y volveré a ponerla a "False" para que no vuelve a entrar el IF hasta que no se produzca una nueva ruptura.

Tenéis que saber que los campos de la derecha, al campo con el cual queremos hacer la ruptura, solo se pierden cuando estamos dentro del AT NEW pero cuando salimos los volvemos recuperar.

## INSTRUCCIONES DE ORDENACION

La instrucción para ordenar una tabla interna es el SORT, la sintaxis es:

SORT tab-int [ BY campo1 [ ASCENDING ] campo2 etcétera [ ASCENDING / DESCENDING ] ]

Recordar que si ponemos en el campo1 ASCENDING el resto de campos se ordenan de forma DESCENDING.

Ejemplos:

```
SORT TABLA.
```

Aquí al no indicarle criterios la ordena por todos los campos exceptuando los de tipo: I, F y P de forma ascendente.

```
SORT TABLA BY CIUDAD PAIS.
```

Ordena TABLA por los campos CIUDAD y PAIS de forma descendente.

```
SORT TABLA BY CIUDAD ASCENDING
              PAIS DESCENDING.
```

Ordena TABLA por los campos CIUDAD (Ascendentemente) y PAIS (Descendentemente).

## OPERACIONES CON LAS TABLAS INTERNAS

### AÑADIR

Para añadir registros se utiliza la instrucción APPEND. Para poder añadir, primero hemos de mover los datos al HEADER y entonces haremos el APPEND.
Su sintaxis será la siguiente:

```
APPEND tabla-int.
```

Crea el contenido de la línea de cabecera en una nueva línea de la tabla delante de la línea actual.

### MODIFICAR

Para modificar los registros se utiliza la orden MODIFY, y como en el APPEND, primero hemos de poner los datos a modificar en el HEADER, modificarlos y ejecutar la orden MODIFY para modificarlo.

Su sintaxis sería la siguiente:

```
MODIFY tabla-int.
```

Otra opción sería:

```
MODIFY nombre_tabla_interna INDEX i
```

Sobreescribe el contenido de la línea de la tabla i con el contenido de la cabecera, la entrada tiene que existir.

### BORRAR

Esta orden es idéntica a la anterior pero con la diferencia que no modifica sino borra. Y primero hemos de mover los datos al HEADER para después borrarlos.
Su sintaxis sería la siguiente:

```
DELETE tabla-int.
```

Otra opción:

```
DELETE nombre_tabla_interna INDEX i
```

Borra la entrada de la tabla i.

### COLLECT

El COLLECT es parecido al MODIFY, pero funciona de otra manera.
El COLLECT busca los campos no numéricos de una tabla interna que sean iguales a los datos que hay en el HEADER y si encuentra uno que sea igual, entonces suma los campos numéricos.

## COMIC WORK Y ROLLBACK

El COMIC WORK sirve para confirmar las actualizaciones que hacemos (añadir, borrar o modificar), si no lo ponemos las actualizaciones las hace cuando sale del programa.

ROLLBACK es todo lo contrario que el COMIC WORK, es decir, no confirma las actualizaciones que realicemos.

## REFRESH

Borra toda la tabla. Su sintaxis:

```
REFRESH tabla-int.
```

## CLEAR

Inicializa la línea de cabecera, borra la HEADER, área para la manipulación de datos de las tablas. Su sintaxis:

```
CLEAR tabla-int.
```

## FREE

Borra todos los datos de la tabla y libera la memoria. Su sintaxis:

```
FREE tabla-int.
```

## OPERACIÓN CON LAS TABLAS DE DICCIONARIO

## AÑADIR

Podemos añadir registro a registro o varios registros a la vez. En los dos casos se utiliza la orden INSERT pero con diferentes opciones.

### REGISTRO A REGISTRO

La sintaxis sería la siguiente:

```
INSERT INTO tabla-dicc [CLIENT SPECIFIED] VALUES estruct.
```

Los datos que están en estructura se guardarían en la tabla de diccionario que le indiquemos en tabla-dicc.

Las estructuras ya hemos aprendido como se declaran.

Un ejemplo de cómo lo haríamos sería el siguiente:

```
TABLES SPFLI.                          WA tiene la estructura de SPFLI
DATA WA LIKE SPFLI.
WA-CARRID = 'LH'.                      Guardo los datos a la estructura WA
WA-CITYFROM = 'WASHINGTON'.
 .....
INSERT INTO SPFLI VALUES WA.           Guardo lo que contiene la estructura WA
                                       en la tabla de diccionario SPFLI
WA-CARRID = 'UA'.
WA-CITYFROM = 'LONDON'.                Hacemos los mismos pasos que antes
 .....
INSERT SPFLI FROM WA.
```

Como vemos es muy fácil insertar datos a una tabla.

La opción CLIENT SPECIFIED no sé para que se utiliza.

**A TRAVES DE UNA TABLA INTERNA**

Lo que quiero decir con esto, es que podemos guardar una tabla interna en una tabla de diccionario con una sola instrucción sin tener que hacer ningún bucle.

También sería con la orden INSERT pero la estructura sería la siguiente:

```
INSERT tabla-dicc [CLIENT SPECIFIED] FROM TABLE tabla-int
[ACCEPTING DUPLICATE KEYS].
```

Como he dicho antes CLIENT SPECIFIED no sé para que se utiliza.
La cláusula ACCEPTING DUPLICATE KEYS sirve para que cuando insertemos datos con claves iguales no nos dé error o ignore lo que introduzcamos.

Un ejemplo de su funcionamiento es el siguiente:

```
TABLES SPFLI.
DATA ITAB LIKE SPFLI OCCURS 10 WITH HEADER LINE.
                                       Voy guardando los datos
ITAB-CARRID = 'UA'. ITAB-CONNID = '0011'.   en el header de la tabla
APPEND ITAB.   Añado a la tabla lo que hay en el header

ITAB-CARRID = 'LH'. ITAB-CONNID = '1245'.
APPEND ITAB.                           Repito el paso
ITAB-CARRID = 'AA'. ITAB-CONNID = '4574'.   anterior 2 veces más
APPEND ITAB.
.................
INSERT SPFLI FROM TABLE ITAB ACCEPTING DUPLICATE KEYS.
```

Esta opción es más conveniente, porque con esta opción sólo hacemos un acceso a la tabla de diccionario mejorando el rendimiento global del programa.

Es decir cuantos menos accesos hagamos al disco mejor será el rendimiento, por ello utilizaremos las tablas internas (que se almacenan en memoria) para realizar las altas, bajas o modificaciones.

Recordar, para que funcione bien la tabla interna ha de tener la misma estructura que la tabla de diccionario.

**MODIFICAR**

Podemos modificar una tabla de diccionario de 3 formas diferentes.

**UN SOLO REGISTRO**

Para modificar un sólo registro utilizamos la orden UPDATE. Su sintaxis es la siguiente:

```
UPDATE tabla_dicc [CLIENT SPECIFIED] FROM estruc.
```

En estruc se guardan los datos a modificar.
Ahora veremos un ejemplo para comprender como realiza la modificación:

```
TABLES SPFLI.
DATA WA LIKE SPFLI.

MOVE 'AA'         TO WA-CARRID.
MOVE '0064'       TO WA-CONNID.
MOVE 'WASHINGTON' TO WA-CITYFROM.
 .....
UPDATE SPFLI FROM WA.
```

} Guardo en la estructura WA los nuevos datos

→ **Modifica la tabla SPFLI**

```
MOVE 'LH'     TO SPFLI-CARRID.
MOVE '0017'   TO SPFLI-CONNID.
MOVE 'BERLIN' TO SPFLI-CITFROM.
 .....
UPDATE SPFLI.
```

} **Repetimos los pasos anteriores**

En este ejemplo CARRID y CONNID son campos claves de la tabla SPFLI, entonces la orden UPDATE busca todos los registros cuyos CARRID y CONNID tengan los valores ‘AA’ y ‘0064’, y los registros que encuentre los reemplazará por los nuevos datos. En el segundo caso sería los mismo pero lo buscaría con los valores ‘LH’ Y ‘0017’.

**VARIOS CAMPOS A LA VEZ**

Para este campo utilizamos también la orden UPDATE, pero de diferente forma, su sintaxis es esta:

UPDATE tabla_dicc [CLIENT SPECIFIED] SET <S1> .. <Sn> [WHERE condición].

En el SET pondremos los campos a modificar con sus nuevos valores, o sea, S1 a Sn lo podríamos desglosar de la siguiente manera.

f = n

Donde ‘f’ es el nombre del campo a modificar y ‘n’ es su nuevo valor.

f = f + g

Al campo ‘f’ se le suma lo que vale ‘g’.

f = f - g

Al campo ‘f’ se le resta lo que vale ‘g’.

En el WHERE podemos buscar los registros que cumplan una determinada condición o condiciones.

Un ejemplo sería este:

```
TABLES SFLIGHT.
UPDATE SFLIGHT SET PLANETYPE = 'A310'
                   FLPRICE = FLPRICE - '10000'
                   WHERE CARRID = 'LH'.
```

En esta caso lee todos los registros de la tabla SFLIGHT cuyo campo clave CARRID tenga el valor LH. El campo PLANETYPE tendrá el valor ‘A310’ y al campo FLPRICE se le restará ‘10000’ del valor que tenga.

**DE UNA TABLA INTERNA**

También podemos modificar una tabla de diccionario con los valores de una tabla interna. La sintaxis sería la siguiente:

UPDATE tabla-dicc [CLIENT SPECIFIED] FROM TABLE tabla-int.

Un ejemplo práctico sería el siguiente:

```
TABLES SPFLI.
DATA ITAB LIKE SPFLI OCCURS 10 WITH HEADER LINE.
ITAB-CARRID = 'UA'. ITAB-CONNID = '0011'.
APPEND ITAB.
ITAB-CARRID = 'LH'. ITAB-CONNID = '1245'.
APPEND ITAB.
```

Voy guardando los datos a modificar en la tabla interna



→ Modifica la tabla de diccionario con los nuevos valores de la tabla interna

```
ITAB-CARRID = 'AA'. ITAB-CONNID = '4574'.
APPEND ITAB.
.....
UPDATE SPFLI FROM TABLE ITAB.
```

Su funcionamiento es similar al UPDATE de un sólo registro. Pero aquí va buscando por los campos claves CARRID y CONNID cuyos valores estén en la tabla interna, que ha de tener la misma estructura que la tabla de diccionario (en el ejemplo se ve como se declara).

Como he dicho en el INSERT, esta forma es más eficiente que no hacer el UPDATE por un sólo registro.

Recordar que para que funcione bien la tabla interna ha de tener la misma estructura que la tabla de diccionario.

Hay que decir que en los 3 casos si ponemos un valor que no existe nos dará un error controlado por la variable del sistema SY-SUBRC.

## MODIFICAR Y AÑADIR

Esta otra orden tiene un funcionamiento similar al UPDATE pero la diferencia radica en el hecho de que mientras el UPDATE sólo modifica, si encuentra el registro buscado, el MODIFY añade este registro a la tabla de diccionario si no encuentra el registro a buscar.
También aquí se puede modificar y añadir de 2 formas diferentes, a saber, un registro solamente o registros procedentes de una tabla interna.

### UN SOLO REGISTRO

Su sintaxis es idéntica al UPDATE, sólo cambia el nombre de la orden. Su sintaxis es esta:

```
MODIFY tabla_dicc [CLIENT SPECIFIED] FROM estruc.
```

Un ejemplo práctico es el siguiente:

```
TABLES SPFLI.
DATA WA LIKE SPFLI.

MOVE 'AA'         TO WA-CARRID.
MOVE '0064'       TO WA-CONNID.
MOVE 'WASHINGTON' TO WA-CITYFROM.
 .....
MODIFY SPFLI FROM WA.
```

} Guardo en la estructura WA los nuevos datos

⟶ **Modifica la tabla**

Aquí como en el UPDATE hace la búsqueda por los campos clave CARRID y CONNID que tendrán los siguientes valores: ‘AA’ y ‘0064’. Si estos valores no los encuentra los añadirá a la tabla.

**POR UNA TABLA INTERNA**

Como en el UPDATE su funcionamiento es exactamente el mismo. Su sintaxis sería esta:

MODIFY tabla-dicc [CLIENT SPECIFIED] FROM TABLE tabla-int.

El ejemplo práctico sería el mismo que en el UPDATE.

```
TABLES SPFLI.
DATA ITAB LIKE SPFLI OCCURS 10 WITH HEADER LINE.
ITAB-CARRID = 'UA'. ITAB-CONNID = '0011'.
APPEND ITAB.
ITAB-CARRID = 'LH'. ITAB-CONNID = '1245'.
APPEND ITAB.
ITAB-CARRID = 'AA'. ITAB-CONNID = '4574'.
APPEND ITAB.
.....
MODIFY SPFLI FROM TABLE ITAB.
```

} Voy guardando los datos a modificar en la tabla interna

→ Modifica la tabla de diccionario con los nuevo valores de la tabla interna

La explicación sería la misma, busca por los campos clave CARRID y CONNID y si encuentra los valores los reemplaza por los ya puestos sino los añade.

Recordar que para que funcione bien la tabla interna ha de tener la misma estructura que la tabla de diccionario.

También recomiendo utilizar esta última opción para mejorar el redimiento del programa y del sistema.

**BORRAR**

Borrar también se puede hacer de 3 formas diferentes.

**UN SOLO REGISTRO**

La sintaxis del DELETE es esta:

DELETE tabla-int [CLIENT SPECIFIED] FROM estruc.

El ejemplo sería este:

```
TABLES SPFLI.
DATA WA LIKE SPFLI.
```

```
MOVE 'AA'        TO WA-CARRID.
MOVE '0064'      TO WA-CONNID.
DELETE SPFLI FROM WA.
```

Aquí busca los campos clave CARRID y CONNID que les ponemos los valores ‘AA’ y ‘0064’ y si los encuentra los borra.

**VARIOS REGISTROS**

Aquí podemos borrar dependiendo de las condiciones que le pongamos. La sintaxis sería esta:

```
DELETE FROM tabla_dicc [CLIENT SPECIFIED] WHERE condición.
```

En condición pondremos las condiciones por la que queremos borrar. Un ejemplo sería este:

```
TABLES SFLIGHT.
DELETE FROM SFLIGHT WHERE PLANETYPE = 'A310'
                AND      CARRID = 'LH'.
```

Borra todos los registros cuyo PLANETYPE y CARRID tengan los valores ‘A310’ y ‘LH’ respectivamente.

**A TRAVÉS DE UNA TABLA INTERNA**

Como en todos los casos anteriores también se puede borrar dependiendo de los valores que tenga una tabla interna. La sintaxis sería esta:

```
DELETE tabla_dicc [CLIENT SPECIFIED] FROM TABLE tabla-int.
```

El ejemplo sería este:

```
TABLES SPFLI.
DATA ITAB LIKE SPFLI OCCURS 10 WITH HEADER LINE.
ITAB-CARRID = 'UA'. ITAB-CONNID = '0011'.
APPEND ITAB.
ITAB-CARRID = 'LH'. ITAB-CONNID = '1245'.
APPEND ITAB.
ITAB-CARRID = 'AA'. ITAB-CONNID = '4574'.
APPEND ITAB.
.....
DELETE SPFLI FROM TABLE ITAB.
```

En este caso borra los registros de SPFLI dependiendo de los valores que tenga los campos clave CARRID y CONNID de la tabla interna.

Recordar que para que funcione la tabla interna ha de tener la misma estructura que la tabla de diccionario.

También recomiendo esta última opción para un uso eficiente de los recursos. No me cansaré de repetir que cuantos menos accesos al disco hagamos mejor para el rendimiento del sistema. Es más rápido trabajar con una tabla interna y después pasarla a una de diccionario, que no trabajar todo el rato con una tabla del diccionario.

## CONFIRMACION O NO DE LOS CAMBIOS

En las tablas internas hemos de confirmar o desconfirmar los cambios que hayamos hecho (altas, bajas y modificaciones).

Para confirmar los cambios realizados es la orden COMMIT WORK.
Y para decir todo lo contrario, o sea, que no confirme los cambios se utiliza la orden ROLLBACK WORK.

SAP almacena todas las bajas, modificaciones y altas que hacemos en un área especial de trabajo, entonces cuando hacemos un COMMIT WORK (SAP realiza la operaciones que le hemos indicado) y si hacemos un ROLLBACK WORK (borrar las operaciones que le hemos indicado, por lo tanto nos la realiza), también puede ocurrir que ese área de trabajo se llene y entonces el sistema realiza un COMMIT WORK automático.

## ATRIBUTOS DE UNA TABLA

En SAP podemos saber cuantos registros o líneas tiene una tabla, la instrucción con su sintaxis sería esta:

DESCRIBE TABLE tabla_interna [LINES <lin<] [OCCURS <occ>].

“lin” -> Devuelve el número de líneas que tiene la tabla.
“occ” -> Número de ocurrencias de la tabla o el número inicial de líneas de la tabla.

Un ejemplo:

```
DATA: BEGIN OF LINE,
        COL1 TYPE I,
        COL2 TYPE I,
      END OF LINE.
DATA ITAB LIKE LINE OCCURS 10.
DATA: LIN TYPE I, OCC TYPE I.
DESCRIBE TABLE ITAB LINES LIN OCCURS OCC.
WRITE: / LIN, OCC.
DO 1000 TIMES.
  LINE-COL1 = SY-INDEX.
  LINE-COL2 = SY-INDEX ** 2.
  APPEND LINE TO ITAB.
ENDDO.
DESCRIBE TABLE ITAB LINES LIN OCCURS OCC.
WRITE: / LIN, OCC.
```

El resultado sería el siguiente:

```
     0        10
 1.000        10
```

FICHEROS EN SAP

Tenemos dos tipos de ficheros en SAP: locales y secuenciales.

**SECUENCIALES**

**¿CÓMO ABRIRLOS?**

Para abrir un fichero secuencial se utiliza la orden OPEN DATASET, su sintaxis es la siguiente:

```
OPEN DATASET nom-fichero FOR ( OUTPUT
                               INPUT
                               APPENDING )

                             (  BINARY MODE  )
                             ( IN            )
                             (  TEXT MODE    )

                             ( AT POSITION posición. )
```

Nom-fichero -> Puede ser una variable, que contiene la ruta del fichero.

OUTPUT -> Escribe un fichero, si ya existe lo borra, ni lee ni modifica.
INPUT -> Lee un fichero, no se puede escribir en él.
APPENDING -> Añade registros a un fichero, no se puede escribir.

BINARY MODE -> Los datos los graba en forma binaria, no legible para el usuario.
TEXT MODE -> Lo graba en formato ASCII, por lo tanto legible para el usuario.

AT POSITION -> En que posición del fichero deseas hacer la modificación.

**¿CÓMO LEERLOS?**

Para leerlos se utiliza la orden READ DATASET, su sintaxis es la siguiente:

```
READ DATASET nom-fichero INTO variable ( LENGTH longitud. )
```

Nom-fichero -> Puede ser una variable, que contiene la ruta del fichero.

Variable -> Es donde vamos a guardar los datos leídos. Puede ser una variable normal o la estructura de una tabla interna o externa.

Longitud -> cuantos bytes (eso creo) queremos que lea. Esta opción es opcional, ya que por defecto la longitud depende de la variable.

SAP no tiene ninguna orden para indicar el final del fichero, para ello se utiliza la variable del sistema SY-SUBRC. SY-SUBRC vale 4 cuando ha llegado al final del fichero y 0 cuando todavía no lo ha hecho.

Un ejemplo de cómo se leería un fichero, sería este:

```
OPEN DATASET 'MI-FICHERO' FOR INPUT
WHILE SY-SUBRC = 0
        READ ....
        INSTRUCCIONES
END-WHILE
CLOSE DATASET 'MI-FICHERO'
```

**¿CÓMO ESCRIBIR EN ELLOS?**

Para escribir en ellos, se utiliza la orden TRANSFER. Su sintaxis es la siguiente:

```
TRANSFER variable TO nom-fichero.
```

Variable -> Es donde están los datos a guardar. Puede ser una variable normal o la estructura de una tabla interna o externa.

Nom-fichero -> Puede ser una variable, que contiene la ruta del fichero.

**¿CÓMO BORRARLOS?**

Para borrarlos se utiliza la orden DELETE, su sintaxis es esta:

```
DELETE DATASET nom-fichero.
```

Nom-fichero -> Puede ser una variable, que contiene la ruta del fichero.

¡IMPORTANTE! : Esta orden borra un fichero, no el registro del fichero.

**¿CÓMO CERRALOS?**

Para cerrar un fichero secuencial se utiliza la orden CLOSE, su sintaxis es esta:

```
CLOSE DATASET nom-fichero.
```

Nom-fichero -> Puede ser una variable, que contiene la ruta del fichero.

**OTRAS COSAS...**

Si utilizamos un mismo fichero bastantes veces ya sea para grabar, buscar u otras cosas, es más cómodo utilizar una variable que no ir escribiendo cada vez el nombre del fichero.

Es lo mismo poner esto:

```
OPEN DATASET 'C: \MIS DOCUMENTOS\FICHERO1.TEXT' FOR ....
```

Que esto:

```
DATA: FICHERO LIKE RLGRAP-FILENAME
                    VALUE  'C: \MIS DOCUMENTOS\FICHERO1.TEXT'

OPEN DATASET FICHERO1 FOR....
```

Como se ve, la ruta del fichero sólo se escribe una vez en todo el programa, lo que resulta más cómodo y práctico.

RLGRAP-FILENAME es una variable del sistema, que sirve para declarar una variable de tipo fichero.

TRATAMIENTO DE CADENAS

**CONCATENATE**

Se utiliza para varias variables en una sola, su sintaxis es la siguiente:

CONCATENATE var1, var2, ... INTO var-destino (SEPARATED BY carac.)

La opción SEPARATED BY... sirve para indicar el carácter de separación entre las variables. Un ejemplo sería este:

```
DATA: C1(10)  VALUE  'Sum',
      C2(3)   VALUE  'mer',
      C3(5)   VALUE  'holi ',
      C4(10)  VALUE  'day',
      C5(30),
      SEP(3)  VALUE ' - '.

CONCATENATE C1 C2 C3 C4 INTO C5 SEPARATED BY SEP.
WRITE / C5.
```

La salida en pantalla sería:

Sum - mer - holi - day

**CONDENSE**

La sintaxis es la siguiente:

CONDENSE string [NO-GAPS].

Sirve para eliminar los espacios en blanco por la izquierda y si hay espacios en blanco entre las palabras los convierte a un solo espacio en blanco. Si ponemos la cláusula NO-GAPS nos elimina todos los espacios en blanco que tenga el string.

**TRANSALATE**

La siguiente orden tiene dos posibilidades:

1. Convierte un string a mayúsculas / minúsculas. Su sintaxis es la siguiente:

   TRANSLATE string TO UPPER CASE. → Pasa el string a mayúsculas
   TRANSLATE string TO LOWER CASE. → Pasa el string a minúsculas

2. Compara dos cadenas y los caracteres comunes los elimina. Su sintaxis:

   TRANSLATE string1 USING string2.

   El resultado de la comparación lo guarda en string1. Ejemplo:

```
DATA: STRING1 (10) VALUE 'AbCdEfGhIj',
      STRING22 (20) VALUE 'AxbXCydYEzfZ'.

TRANSLATE STRING1 USING STRING2.
WRITE / STRING1.
```

   La salida en pantalla sería esta:
   xXyYzZGhIj

**REPLACE**

Reemplaza el contenido de un string por otro string en otro string. La sintaxis es esta:

REPLACE STR1 WITH STR2 INTO STRING [LENGHT long].

STR1 -> Variable donde esta el string que será sustituido.
STR2 -> Variable donde esta el string que sustituirá a STR1
STRING -> String donde buscare STR1.
Long -> La longitud que voy a sustituir.
Ejemplos:

```
DATA: T(10) VALUE 'abcdefghij',
      STRING LIKE T,
      STR1(4) VALUE 'cdef',
      STR2(4) VALUE 'klmn',
      STR3(2) VALUE 'kl',
      STR4(6) VALUE 'klmnop',
      LEN TYPE I VALUE 2.

STRING = T.
REPLACE STR1 WITH STR2 INTO STRING.
WRITE / STRING.
```

La salida en pantalla sería la siguiente: abklmnghij

```
STRING = T.
REPLACE STR1 WITH STR2 INTO STRING LENGTH LEN.
WRITE / STRING.
```

La salida en pantalla sería la siguiente: abklmnefgh

```
STRING = T.
REPLACE STR1 WITH STR3 INTO STRING.
WRITE / STRING.
STRING = T.
```

La salida en pantalla sería la siguiente: abklghij

```
REPLACE STR1 WITH STR4 INTO STRING.
WRITE / STRING.
```

La salida en pantalla sería la siguiente:  abklmnopgh

En esta ultima instrucción se pierden los dos últimos caracteres, por que STRING es 10 caracteres y la sustitución completa ocuparía 12 caracteres. Por ello se pierden los 2 últimos.

**OVERLAY**

Añade los caracteres de un string que no se encuentran en otro string, su sintaxis:

OVERLAY STR1 WITH STR2 [ONLY STR3.]

Si la cláusula ONLY es omitida los espacios que haya en STR1 serán sobrescritos.
La cláusula ONLY indica que solo se reemplazará los carácter que haya en STR3.

Un ejemplo:

```
DATA: T(10) VALUE 'a c e g i ',
      STRING LIKE T,
      OVER(10) VALUE 'ABCDEFGHIJ',
      STR(2) VALUE 'ai'.
```

```
STRING = T.
OVERLAY STRING WITH OVER.
WRITE / STRING.
STRING = T.
OVERLAY STRING WITH OVER ONLY STR.
WRITE / STRING.
```

La salida en pantalla sería esta:

```
aBcDeFgHiJ
A c e g I
```

**SEARCH**

Nos permite buscar un carácter o caracteres en un string o en una tabla.

SEARCH string FROM string _buscar [OPCIONES.]

Si la búsqueda ha sido un éxito SY-FDPOS devolverá en que posición la ha encontrado. Si no es así SY-SUBRC valdrá 4.

El string a buscar (string_buscar) puede tomar las siguientes formas

| String_buscar | Proposito |
|---|---|
| <cadena> | <cadena> (Una secuencia de caracteres a buscar). Los espacios en blanco los ignora. |
| .<cadena>. | La <cadena> a buscar. Los espacios en blanco no se ignoran |
| *<cadena> | Caracteres comodín y al final la cadena a buscar |
| <cadena>* | Primer la cadena seguido de caracteres comodines |

Un string puede separarse por espacios, comas, puntos, puntos y comas, periodos, interrogantes, exclamaciones, los don puntos, signo más y signo igual.

Un ejemplo de su utilización:

```
DATA STRING(30) VALUE 'This is a little sentence.'.

WRITE: / 'Searched', 'SY-SUBRC', 'SY-FDPOS'.

ULINE /1(26).
SEARCH STRING FOR 'X'.

WRITE: / 'X', SY-SUBRC UNDER 'SY-SUBRC',
          SY-FDPOS UNDER 'SY-FDPOS'
SEARCH STRING FOR 'itt   '.

WRITE: / 'itt   ', SY-SUBRC UNDER 'SY-SUBRC',
            SY-FDPOS UNDER 'SY-FDPOS'
```

```
SEARCH STRING FOR '.e .'.

WRITE: / '.e .', SY-SUBRC UNDER 'SY-SUBRC',
          SY-FDPOS UNDER 'SY-FDPOS'.

SEARCH STRING FOR '*e'.
WRITE: / '*e ', SY-SUBRC UNDER 'SY-SUBRC',
          SY-FDPOS UNDER 'SY-FDPOS'.

SEARCH STRING FOR 's*'.
WRITE: / 's* ', SY-SUBRC UNDER 'SY-SUBRC',
          SY-FDPOS UNDER 'SY-FDPOS'.
```

El resultado sería este:

| Cadena a buscar | SYB-SUBRC | SY-FDPOS |
|---|---|---|
| X | 4 | 0 |
| Itt | 0 | 11 |
| .e . | 0 | 15 |
| *e | 0 | 10 |
| s* | 0 | 17 |

En OPCIONES tenemos las siguientes posibilidades:

- ABREVIATED -> Busca en “string” una palabra que contenga al menos un carácter especificado en “string_buscar”.
- STARTING AT n1 -> Empieza a buscar en la posición indicada por “n1”. SY-FDPOS devuelve la posición refiriéndose al inicio de “n1” y no a la posición del inicio del string a buscar. Si buscamos en una tabla interna “n1” es en la línea o registro a buscar.
- ENDING AT n2 -> Busca hasta la posición “n2”. En una tabla interna hasta que línea buscaremos.
- AND MARK -> Si encontramos algo, todos los caracteres del string a buscar (y todos los caracteres del medio si usamos ABREVIATED) son convertidos a mayúsculas.

Un ejemplo de estas opciones:

```
DATA: STRING(30) VALUE 'This is a fast first example.',
      POS TYPE I,
      OFF TYPE I.

SEARCH STRING FOR 'ft' ABBREVIATED.
WRITE: / 'SY-FDPOS:', SY-FDPOS.
POS = SY-FDPOS + 2.

SEARCH STRING FOR 'ft' ABBREVIATED STARTING AT POS AND MARK.
WRITE / STRING.
```

```
WRITE: / 'SY-FDPOS:', SY-FDPOS.
OFF = POS + SY-FDPOS -1.
WRITE: / 'Off:', OFF.
```

La salida en pantalla sería la siguiente:

```
SY-FDPOS:    10
This is a fast FIRST example.
SY-FDPOS:    4
Off:         15
```

Si la búsqueda se realiza a través de una tabla interna, hemos de mirar también la variable del sistema SY-TABIX que devuelve en que línea o registro se encuentra el string, el resto de variables del sistema funcionan de la misma manera.

```
TYPES: BEGIN OF LINE,
         INDEX   TYPE I,
         TEXT(8) TYPE C,
       END OF LINE.
DATA: ITAB TYPE LINE OCCURS 10 WITH HEADER LINE,
      NUM(2) TYPE N.
DO 10 TIMES.
  ITAB-INDEX = SY-INDEX.
  NUM = SY-INDEX.
  CONCATENATE 'string' NUM INTO ITAB-TEXT.
  APPEND ITAB.
ENDDO.
SEARCH ITAB FOR 'string05' AND MARK.
WRITE: / '"string05" encontrado en la línea', (1) SY-TABIX,
         'en la posición', (1) SY-FDPOS.
SKIP.
READ TABLE ITAB INDEX SY-TABIX.
WRITE: / ITAB-INDEX, ITAB-TEXT.
```

La salida sería la siguiente:

```
'string05' found at line 5 with offset 4
         5 STRING05
```

Hay otra opción para saber si hay un carácter o caracteres en un string, la orden es ‘CA’ y se utiliza en los ‘IF’ o en cualquier orden de condición. Ejemplo:

```
IF string CA ‘V’.
      .
      .
```

Devolvería TRUE si en ‘string’ esta la ‘V’ y FALSE si no lo esta.

Esta opción es más rápida y útil que la orden SEARCH, pero todo depende de cuando y como queremos utilizar la orden ‘SEARCH’ o ‘CA’.

**SHIFT**

Con esta orden permite coger caracteres de un string. La sintaxis:

SHIFT string [ BY n PLACES] [modo].

Si se omite la cláusula BY... se coge la primer posición por la izquierda.

Si se pone BY... coge la “n” primeras posiciones por la izquierda.

Los modos que podemos escoger son los siguientes:

- LEFT -> Coge los caracteres por la izquierda.
- RIGHT-> Coge los caracteres por la derecha.
- CIRCULAR -> los caracteres que coge por la izquierda los coloca por la derecha.

Un ejemplo:

```
DATA: T(10) VALUE 'abcdefghij',
    STRING LIKE T.

STRING = T.

SHIFT STRING.
WRITE / STRING.

STRING = T.
SHIFT STRING BY 3 PLACES LEFT.
WRITE / STRING.

STRING = T.
SHIFT STRING BY 3 PLACES RIGHT.
WRITE / STRING.

STRING = T.
SHIFT STRING BY 3 PLACES CIRCULAR.
WRITE / STRING.
```

El resultado sería el siguiente:

```
bcdefghij
defghij
  abcdefg
defghijabc
```

**STRLEN**

Devuelve la longitud de un string. Ejemplo:

```
Long = STRLEN( 'IVAN RODRIGO' ).
```

ó

```
Long = STRLEN( TEXT-001 ).
```

*****

En el editor de programas, si lo ponemos al principio de la línea (si no da error al compilar) lo que escribamos después del * SAP lo considerará un comentario. Ejemplo:

* Esto es un comentario.

**"**

También desde el editor de programas o texto fuente, lo que escribamos a continuación del " lo considera un comentario, no importa en que parte de la línea estemos lo considerará un comentario. Ejemplo:

LOOP AT MI_TABLA.  "Le la tabla interna MI_TABLA

TRATAMIENTO DE CAMPOS EN DYNPROS

SAP tiene dos instrucciones que nos permite controlar si un campo a cambiado de valor en una dynpro, que quiero decir, pues cuando un campo de una dynpro cambia de valor se ejecutará él modulo que le indiquemos.

Más adelante veremos que es una dynpro y un módulo, por ahora explicaré la sintaxis de estas dos ordenes.

**FIELD**

Esta orden sirve cuando un campo es modificado que se vaya a un módulo para su comprobación. La orden "FIELD" se escribe o declara en la "PAI" (En la lógica de proceso de la "SCREEN PAINTER"), Su sintaxis sería la siguiente:

```
FIELD campo MODULE modulo.
```

En campo podemos poner una variable, un campo de una tabla interna o de diccionario.

A continuación pondré un ejemplo gráfico de donde pondríamos la orden "FIELD"

```
      ....+....1....+....2....+....3....+....4....+....5....+....6....+....7
000010 PROCESS BEFORE OUTPUT.
000020  MODULE STATUS_0001.
000030 *
000040 PROCESS AFTER INPUT.
000050  FIELD TABLA-CARRID MODULE CHEQUEO_CARRID.
000060  MODULE USER_COMMAND_0001.
000070
```

Como vemos en esta imagen el “FIELD” lo ponemos en la “PAI” (Process After Input), se suele poner el “FIELD” antes de los módulos.

Ahora él módulo “CHEQUEO_CARRID” contendría las siguientes instrucciones:

```
000970 MODULE CHEQUEO_CARRID INPUT.
000980   REFRESH TABLA. PA = 1.
000990   TOTAL_PAG = 0.
001000   SELECT * FROM SPFLI
001010          WHERE CARRID = TABLA-CARRID.
001020     MOVE-CORRESPONDING SPFLI TO TABLA.
001030     APPEND TABLA.
001040     ADD 1 TO TOTAL_PAG.
001050   ENDSELECT.
001060 ENDMODULE.
```

**CHAIN**

“El CHAIN” es como el “FIELD” pero podemos hacer dos tipos de “CHAIN”: Múltiples campos con múltiples módulos y múltiples campos con un solo módulo.

Los “CHAIN” son muy útiles cuando queremos que un campo o varios que cambian de valor nos realicen una determinada cosa. Pues no tenemos que preocuparnos de controlar cuando el valor del campo o campos varíe, ya que el SAP nos lo controla a través de la orden “CHAIN”.

Lo que no sé si por cada “PAI” puede haber más de un “CHAIN” o un solo, pero personalmente con un solo va de sobra.

**MULTIPLES CAMPOS CON MULTIPLES MODULOS**

Como el “FIELD” también se declararía en la “PAI” de la lógica de proceso. Mejor veamos un ejemplo:

```
      ....+....1....+....2....+....3....+
000010 PROCESS BEFORE OUTPUT.
000020  MODULE STATUS_0001.
000030 *
000040 PROCESS AFTER INPUT.
000050 CHAIN.
000060    FIELD: TABLA-CARRID.
000070    FIELD: TABLA-CONNID.
000080    MODULE CHEQUEO_CARRID.
000090    MODULE IR.
000100 ENDCHAIN.
000120  MODULE USER_COMMAND_0001.
```

Como vemos el “CHAIN” lo ponemos antes de llamar a cualquier módulo.
SAP asocia el primer “FIELD” con el primer “MODULE” y el segundo “FIELD” se asocia con el segundo “MODULE” y así respectivamente.

En cada “FIELD” podemos poner más de un campo aunque en este caso solo ponemos un campo.

Si cualquiera de los dos campos que hay en el “FIELD” cambia de valor entonces se ejecuta él módulo que corresponde a cada “FIELD”.

También hay que decir que primero se escriben todos los “FIELD” que queremos controlar y después los módulos. En el ejemplo anterior solo dos “FIELD” y dos “MODULE” pero puede haber todos los que queramos.

El código del primer module sería este:

```
000980 MODULE CHEQUEO_CARRID INPUT.
001000   REFRESH TABLA. PA = 1.
001010   TOTAL_PAG = 0.
001020   SELECT * FROM SPFLI
001030          WHERE CARRID = TABLA-CARRID.
001040     MOVE-CORRESPONDING SPFLI TO TABLA.
001050     APPEND TABLA.
001060     ADD 1 TO TOTAL_PAG.
001070   ENDSELECT.
001080 IF TOTAL_PAG = 0.
001090    MESSAGE E006 WITH TABLA-CARRID.
001100 ENDIF.
001120 ENDMODULE.
```

En este ejemplo compruebo que existe la compañía aérea introducida, si la encuentra guarda en una tabla interna las conexiones de vuelo que tiene esa compañía, si no la encuentra muestra un mensaje de error.

El segundo “MODULE” llamaría a una “DYNPRO” a través de una transacción.

**MULTIPLES CAMPOS CON UN SOLO MODULO**

Aquí es más fácil de entender ya que solo se llama a un módulo. La declaración la haría también el “PAI” y se haría de la siguiente forma:

```
000010 PROCESS BEFORE OUTPUT.
000020  MODULE STATUS_0001.
000030 *
000040 PROCESS AFTER INPUT.
000050 CHAIN.
000060    FIELD: TABLA-CARRID,TABLA-CONNID.
000080    MODULE CHEQUEO_CARRID.
000100 ENDCHAIN.
000120 MODULE USER_COMMAND_0001.
```

Como vemos aquí controlamos que cuando uno de los dos campos cambia de valor se procesa el módulo que tiene debajo.

El código de módulo sería el siguiente (En la página siguiente):

```
000980 MODULE CHEQUEO_CARRID INPUT.
000990   REFRESH TABLA. PA = 1.
001000   TOTAL_PAG = 0.
001010   SELECT * FROM SPFLI
001020          WHERE CARRID = TABLA-CARRID AND CONNID = TABLA-CONNID.
001030     MOVE-CORRESPONDING SPFLI TO TABLA.
001040     APPEND TABLA.
001050     ADD 1 TO TOTAL_PAG.
001060   ENDSELECT.
001070 IF TOTAL_PAG = 0.
001080    MESSAGE E006 WITH TABLA-CARRID.
001090 ENDIF.
001100 ENDMODULE.
```

Como vemos si cada vez que varíe uno de los dos campos que controlo (TABLA-CONNID y TABLA-CARRID) vuelvo a buscar todos los vuelos.

FORMATEO DE LISTADOS

**FORMAT INTENSIFIED OFF**

Fija atributos de pantalla, por defecto es ON.

**WRITE**

Si queremos imprimir una sola cosa, un comentario, se haría de la siguiente forma:

WRITE 'PRUEBA POR PANTALLA'.

Si queremos imprimir más de un dato se haría de la siguiente forma:

WRITE: dato1, dato2, etcétera.

Los datos puede ser cualquier cosa desde una variable, un elemento de texto, etc.

El WRITE también tiene opciones:

- / -> Al acabar el WRITE realiza un salto de línea.

  WRITE: / 'PRUEBA DE IMPRESIÓN'.

  Como se ve él '/ ' se puede separar con una coma o sin coma:

  WRITE: /, 'PRUEBA DE IMPRESIÓN'.

  Aunque creo que el efecto no es el mismo.
- N-> Sería el tabulador.

  WRITE: 5 'PRUEBA DE IMPRESIÓN'.

  El texto lo escribiría en la posición 5.

- COLOR color -> Visualiza un dato con un número de color determinado.

  Los colores los podemos ver en el icono que sale en la parte superior Derecha. El icono es este:

   → COLORES EN SAP

  Haciendo clic sobre este icono nos saldrá un menú contextual y sí elegimos la opción: OPTIONS nos saldrán los colores del SAP.

  Ejemplo:

  WRITE: 'HOLA' COLOR 2.

  Para escribir un texto en un color inverso, sería así:

```
WRITE: 'HOLA' INVERSE COLOR 3.
```

- ... AS CHECKBOX

  La salida es una caja de selección. El contenido del primer carácter del campo escrito es interpretado como el estado de la caja de selección; un espacio es *no seleccionada*, *X* es *seleccionada.*

  Si no queremos que el usuario cambie el estado de la caja de selección durante la ejecución ésta se escribirá añadiendo INPUT OFF, de este modo sólo podrá ser cambiada mediante código. Por ejemplo:

```
DATA CAJA(1) TYPE C VALUE SPACE.
WRITE CAJA AS CHECKBOX.
WRITE CAJA AS CHECKBOX INPUT OFF.
```

- ... AS SYMBOL

  Podemos escribir ciertos símbolos en pantalla usando este modificador. Para ello es necesario haber escrito previamente en nuestro código:

```
INCLUDE <SYMBOL>.
```

  Por ejemplo, la sentencia:

```
WRITE: SYM_RIGHT_HAND AS SYMBOL, 'apunta a la derecha'.
```

  Apunta a la derecha.

- ... AS ICON

  También se pueden añadir iconos a un Report. Para ello antes de escribir un WRITE que contenga AS ICON hay que poner INCLUDE <ICON>. La lista de iconos también se puede ver en la ayuda asociada a esa palabra.

  Por ejemplo:

```
WRITE: ICON_GREEN_LIGHT AS ICON, 'semáforo verde'.
```

  Los iconos y los símbolos normalmente ocupan más de un carácter. Podemos saber en tiempo de ejecución cual es su longitud, con la sentencia:

```
DESCRIBE FIELD NOMBRE_IC_O_SYMBOL LONGITUD.
```

  Donde LONGITUD es una variable que almacenará el número de caracteres que ocupa el icono o el símbolo.

- ... AS LINE.

  La forma de hacer líneas horizontales o verticales en un Report u que éstas formen cuadrículas es escribir los caracteres SY-VLINE (línea

vertical) y SY-ULINE (línea horizontal). Son equivalentes a los caracteres "|" y "-". La forma exacta en la que aparecen estos segmentos depende de los caracteres adyacentes. Cuando tenemos un carácter de línea en una posición y otro en la posición adyacente automáticamente se produce la unión de ambas. Si en la posición adyacente no hay otro carácter de línea ese carácter permanece inalterado.

En la mayoría de los casos esta técnica es suficiente para hacer cuadrículas. Pero hay veces en las que las uniones no se producen de la forma en la que queremos. En esos casos será necesario echar mano de caracteres especiales: LINE_TOP_LEFT_CORNER, LINE_BOTTOM_MIDDLE_CORNER que realizan esta operación.

Para poder usar estos caracteres es necesario incluir en nuestro código INCLUDE <LINE>.

- también tenemos diferentes opciones, para el formato de salida:
  ... NO-ZERO.
  ... NO-SIGN.
  ... DD/MM/YY.
  ... MM/DD/YY.
  ... DD/MM/YYYY.
  ... MM/DD/YYYY.
  ... DDMMYY.
  ... CURRENCY w.
  ... DECIMALS d.
  ... ROUND R.
  ... UNIT u.
  ... EXPONENT e.
  ... USING EDIT MASK mask.
  ... UNDER g.
  ... NO-GAP.
  ... LEFT-JUSTIFIED.
  ... CENTERED.
  ... RIGHT-JUSTIFIED.

**SKIP**

La orden SKIP realiza salto o saltos de línea. Su sintaxis sería la siguiente:

SKIP n.

n-> Es el número de líneas que saltará.

Si no ponemos nada haremos un salto de línea.

Ejemplo: SKIP.

es lo mismo que hacer:

```
SKIP 1.
```

También es posible moverse a una línea determinada dentro del Report actual con SKIP TO LINE ... (p.ej: SKIP TO LINE 10)

**ULINE**

Esta orden dibuja una línea por pantalla.

Si escribimos:

```
ULINE.
```

A secas dibujará una línea que ocupará toda la pantalla.

Si queremos dibujar una línea desde una posición y de una determina longitud, se haría de la siguiente forma:

```
ULINE AT 35(15).
```

Aquí empieza la línea en la posición 35 y tendrá una longitud de 15. Y si además queremos que después haga un salto de línea:

```
ULINE AT /35(15).
```

**NEW-PAGE**

Provoca salto de página.

**NEW-LINE**

Provoca salto de línea.

**POSITION n**

Inicia la siguiente impresión en la línea n. Es posible posicionarse hacia atrás en la misma página.

**SET BLANK LINES ON**

Por defecto el sistema suprime líneas que contienen sólo caracteres en blanco. Si se activa esta opción aparecen en el listado.

**FORMAT**

Con esta instrucción podemos cambiar el aspecto de lo que vamos a escribir en el report, FORMAT tendría las siguientes opciones:

- FORMAT COLOR n ON -> Todas las líneas que se escriben entre esta sentencia y la correspondiente FORMAT COLOR n OFF. Se escribirán en un color determinado.
- FORMAT INTENSIFIED ON -> Se refiere al color de fondo.
- FORMAT INVERSE ON -> Al color de fondo y de la cadena que se escribe.
- FORMAT INPUT ON -> Habilita la entrada en los campos que se escriban a continuación hasta encontrar el correspondiente FORMAT INPUT OFF. Esta opción se usa en campos que se escriben ...AS CHECKBOX o ... AS RADIOBUTTOM..
- FORMAT HOTSPOT ON -> hace que cuando el cursor se sitúe sobre las líneas o palabras que se escriban a continuación tome aspecto de *mano*, permitiendo su selección con un simple click. Esta opción se inhabilita con el correspondiente FORMAT HOSTPOT OFF.
- FORMAT RESET -> inicializa todas las opciones de formato. Esta instrucción equivale a: FORMAT COLOR OFF INTENSIFIED OFF INVERSE OFF HOTSPOT OFF INPUT OFF.

**SET PF-STATUS 'nombre'.**

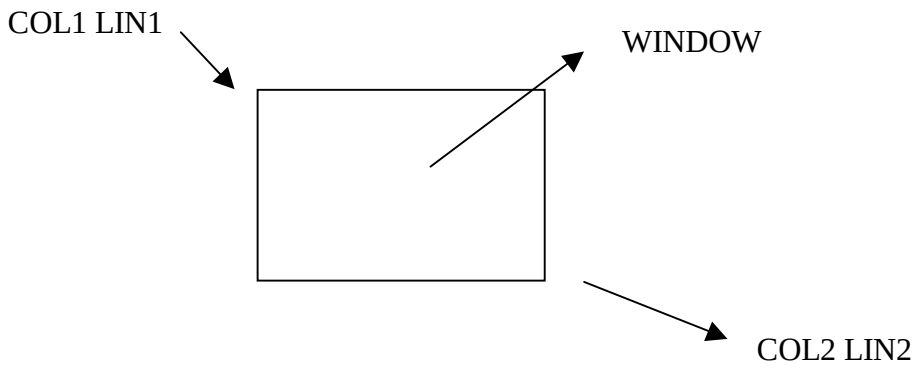
Esto para que nos muestre un menú painter echo por nosotros.

La forma de hacer un menú painter se explica más adelante.

**WINDOW**

Esta orden sirve para crear ventanas (podemos crear hasta 9). Es muy útil para hacer listados secundarios. La sintaxis de esta orden es:

```
WINDOW STARTING AT <col1> <lin1>
 ENDING AT <col2> <lin2>
```

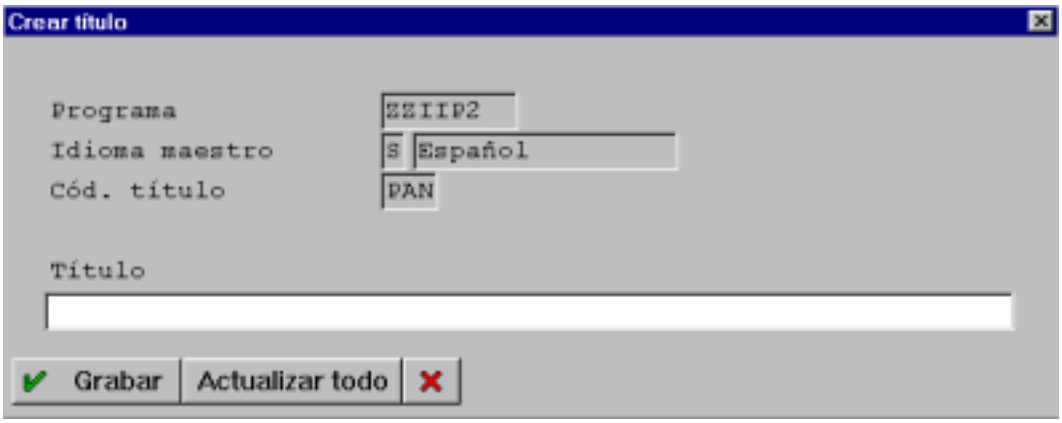



La orden WINDOW se puede utilizar en cualquier parte del programa y/o evento, no sólo en AT LINE-SELECTION.

Por defecto SAP pone un título a la ventana, para cambiar el título se utiliza la orden:

SET TITLEBAR ‘nombre-título’.

Cuando hallamos escrito esta orden, hacemos doble clic en nombre-título y nos dirá que ese título no esta creado y preguntará si lo queremos crear. Si decimos que sí nos saldrá esta ventana para introducir el título:




Cuando lo hallamos introducido lo grabaremos y volveremos al editor de programa.
Ahora para que SAP “coja” ese título tenemos que grabar el programa y después compilarlo, si no lo hacemos SAP ignorará el título que hallamos creado.

La orden SET TITLEBAR..... va después de la orden WINDOW.

**READ CURRENT LINE**

Todavía no sé su funcionamiento.

**MODIFY CURRENT LINE**

Se utiliza para modificar la línea donde esta el cursor posicionado. Su sintaxis sería la siguiente:

MODIFY CURRENT LINE LINE FORMAT opciones

CURRENT LINE es la línea donde esta el cursor

LINE FORMAT, qué formato queremos ponerle a la línea.

Opciones, qué valores queremos darle a la línea.

Un ejemplo sería cambiar de color una línea, en este ejemplo pondremos una línea de color gris y azul (SAP las utiliza para hacer las cabeceras):

MODIFY CURRENT LINE LINE FORMAT COLOR = 1.

INSTRUCCIONES ARÍTMETICAS

La siguiente tabla muestro como se puede realizar las operaciones aritméticas más comunes en ABAP/4 de dos formas diferentes:

| Operación | Sintaxis | Sintaxis |
|---|---|---|
| | Expresión matemática | Instrucción |
| Suma | <p> = <n> + <m>. | ADD <n> TO <m>. |
| Resta | <p> = <m> - <n>. | SUBTRACT <n> FROM <m>. |
| Multiplicación | <p> = <m> * <n>. | MULTIPLY <m> BY <n>. |
| División | <p> = <m> / <n>. | DIVIDE <m> BY <n>. |
| División entera | <p> = <m> DIV <n>. | --- |
| Resto división | <p> = <m> MOD <n>. | --- |
| Exponenciación | <p> = <m> ** <n>. | --- |
| Exponenciación con base e. (e=2.7182818285) | <p> = EXP( <n> ). | --- |
| Logaritmo de base e | <p> = LOG( <n> ). | |
| Logaritmo base 10 | <p> = LOG10( <n> ). | |
| Raíz cuadrada | <p> = SQRT( <n> ). | --- |
| FUNCIONES TRIGRONOMETICAS | | |
| ACOS | <p> = ACOS( <n> ). | |
| ASIN | <p> = ASIN( <n> ). | --- |
| ATAN | <p> = ATAN( <n> ). | --- |
| Coseno | <p> = COS( <n> ). | --- |
| Seno | <p> = SIN( <n> ). | --- |
| Tangente | <p> = TAN( <n> ). | --- |
| TRAMIENTO DE NUMEROS | | |
| Valor absoluto | <p> = ABS( <n> ). | --- |
| Signo del número:<br>1 Sí N > 0,<br>0 Sí N = 0 y<br>-1 Sí N < 0 | <p> = SIGN( <n> ). | --- |
| Devuelve el entero más bajo | <p> = CEIL( <n> ). | --- |
| Devuelve el entero más alto | <p> = FLOOR( <n> ). | --- |
| Devuelve la parte entera | <p> = TRUNC( <n> ). | --- |
| Devuelve la parte fraccionaria | <p> = FRAC( <n> ). | |

Como vemos solo las operaciones más sencillas se pueden hacer a través de una instrucción (ADD, SUBTRACT, DIVIDE y MULTIPLY), para realizar operaciones más complejas se utiliza la orden “=” o también la instrucción COMPUTE. Esta orden se pone al principio de la expresión aritmética pero el funcionamiento es el mismo, ejemplo:

```
COMPUTE A = B + C.
```

Un ejemplo de las instrucciones de “tratamiento de números” sería la siguiente:

```
DATA N TYPE P DECIMALS 2.
DATA M TYPE P DECIMALS 2 VALUE '-5.55'.
N = ABS( M ).   WRITE:   'ABS:  ', N.
N = SIGN( M ).  WRITE: / 'SIGN: ', N.
N = CEIL( M ).  WRITE: / 'CEIL: ', N.
N = FLOOR( M ). WRITE: / 'FLOOR:', N.
N = TRUNC( M ). WRITE: / 'TRUNC:', N.
N = FRAC( M ).  WRITE: / 'FRAC: ', N.
```

La salida en pantalla sería la siguiente:

```
ABS:          5.55
SIGN:         1.00-
CEIL:         5.00-
FLOOR:        6.00-
TRUNC:        5.00-
FRAC:         0.55-
```

**SUM**

La orden SUM suma todos los campos de una tabla interna que sea de tipo I (Entero), P (Empaquetado) y F (Coma flotante).

El resultado de la suma lo guarda en el mismo campo. Se utiliza cuando hay una ruptura de fin de campo. Veamos un ejemplo:

```
AT END OF COUNTRY.
        SUM.
        SKIP.
        WRITE  ‘El total de ventas de una país es: ‘, TABLA-COUNTRY.
ENDAT.
```

Como vemos el resultado de la suma lo guarda en el mismo campo.

Para sumar solo campo, lo haríamos de la siguiente forma:

```
AT END OF COUNTRY.
        WRITE  ‘El total de ventas de una país es: ‘, SUM(TABLA-COUNTRY).
ENDAT.
```

*Para utilizar la orden SUM, la tabla ha de estar ordenada si no dará error*

**CNT**

Calcula el número de entradas diferentes que se producen en un campo, ejemplo:

```
AT LAST
    WRITE    'El total de ventas de una país es: ', CNT(TABLA-COUNTRY).
ENDAT.
```

*Para utilizar la orden CNT, la tabla ha de estar ordenada si no dará error.*

**OPERACIONES CON ESTRUCTURAS**

En SAP podemos sumar, restar, dividir y multiplicar estructuras. Para ellos tenemos las ordenes: ADD-CORRESPONDING, SUBTRACT-CORRESPONDING, DIVIDE-CORRESPONDING y MULTIPLY-CORRESPONDING. La sintaxis sería la siguiente:

```
ADD-CORRESPONDING ESTRUC1 BY ESTRUC2.
```

La sintaxis para las 4 ordenes es la misma. El resultado de la operación se guarda en ESTRUC1. No hay que decir, que para se realice la operación en los campos de las dos estructuras los campos han de ser iguales (nombre y tipo).

Un ejemplo sería este:

```
DATA: BEGIN OF RATE,
      USA TYPE F VALUE '0.6667',
      FRG TYPE F VALUE '1.0',
      AUT TYPE F VALUE '7.0',
    END OF RATE.
DATA: BEGIN OF MONEY,
      USA TYPE I VALUE 100,
      FRG TYPE I VALUE 200,
      AUT TYPE I VALUE 300,
    END OF MONEY.
MULTIPLY-CORRESPONDING MONEY BY RATE.
WRITE / MONEY-USA.
WRITE / MONEY-FRG.
WRITE / MONEY-AUT.
```

INSTRUCCIONES DE ASIGNACION

**MOVE**

El MOVE sirve para copiar el valor de una variable en otra. Su sintaxis sería la siguiente:

```
MOVE dato-origen TO variable-destino.
```

Es idéntica al MOVE del COBOL. Ejemplo:

MOVE 5 TO suma.

El dato de origen puede ser cualquier cosa: texto, número, variables, etc.

La variable de destino ha de ser una variable, ya que por ejemplo no podemos hacer esto:

MOVE suma TO 5. ⟶ INCORRECTO

Dentro del MOVE tenemos una opción para mover una tabla a otra. La sintaxis sería:

MOVE-CORRESPONDING tabla-origen TO tabla-destino.

Se suele utilizar por ejemplo para mover el contenido de una tabla de diccionario a una interna. Es un complemento ideal al INCLUDE STRUCTURE (explicado anteriormente). Ejemplo:

MOVE-CORRESPONDING TABNA TO TABLA1

Tabla de diccionario

Tabla interna

Hay que decir que sólo copia los datos cuyos campos tengan el mismo nombre.

=

Como hemos visto en las operaciones aritméticas, su funcionamiento es sencillo. Ejemplo:

RESUL = 45.

Aquí asigno a RESUL el valor 45.

Recomiendo utilizar él '=' en vez del MOVE ya que es más cómodo y sencillo, a no ser que queremos mover datos entre estructuras que recomiendo el MOVE-CORRESPONDIG.

ATRIBUTOS DE UN CAMPO

En SAP podemos saber como es una variable. La instrucción y la sintaxis sería la siguiente:

DESCRIBE FIELD <f> [LENGTH <l>] [TYPE <t> [COMPONENTS <n>]]
[OUTPUT-LENGTH <o>] [DECIMALS <d>]
[EDIT MASK <m>].

LENGHT -> Longitud de la variable
TYPE -> Tipo de la variable o estructura.
OUTPUT-LENGHT-> Longitud del campo de salida.
DECIMALS-> Número de decimales.

EDIT MASK ->Mascará de la variable.

Un ejemplo de cada opción:

```
DATA: TEXT(8), LEN TYPE I.
DESCRIBE FIELD TEXT LENGTH LEN.
```
El contenido de LEN sería 8.

```
TABLES SPFLI.
DATA: NUMTEXT(8) TYPE N, TYP.
DESCRIBE FIELD NUMTEXT TYPE TYP.
WRITE TYP.
DESCRIBE FIELD SPFLI-FLTIME TYPE TYP.
WRITE TYP.
```

La salida en pantalla sería:
N T

```
DATA: FLOAT TYPE F, OUT TYPE I, LEN TYPE I.
DESCRIBE FIELD FLOAT LENGTH LEN OUTPUT-LENGTH OUT.
```

El resultado de LEN valdría 8 y el campo OUT valdría 22.

```
DATA: PACK TYPE P DECIMALS 2, DEC.
DESCRIBE FIELD PACK DECIMALS DEC.
```

DEC valdría 2.

## COMPROBACIONES DE AUTORIZACIONES EN ABAP/4

- Una autorización es una característica de los campos de un objeto.
- Las autorizaciones se pueden agrupar en perfiles.
- Los perfiles se pueden agrupar en perfiles compuestos.
- Los perfiles compuestos se asignan a los usuarios.

Para comprobar autorizaciones se utiliza la orden: AUTORITY-CHECK... cuya sintaxis es la siguiente:

```
AUTORITY-CHECK OBJECT ID ‘CAMPO1’ FIELD <FIELD1>
                      ID ‘CAMPO2’ FIELD <FIELD2>.
```

Los campos cuyos valores no deban comprobarse pueden excluirse mediante el parámetro DUMMY.

```
AUTORITY-CHECK OBJECT ID ‘CAMPO1’ FIELD <FIELD1>
                      ID ‘CAMPO2’ DUMMY.
```

BLOQUEO LÓGICO DE OBJETOS

Para prevenir una actualización en paralelo de los mismos objetos de datos, se pueden bloquear para el resto de usuarios.

El bloqueo lógico se debería efectuar en el evento PAI de la dynpro o del report, cuando vayamos a realizar alguna operación(leer, borrar, modificar o añadir) con las tablas internas o externas.

Una vez realizada la modificación se debería desbloquear.

Para bloquear se utiliza la orden:

ENQUEUE <nombre_objeto_bloqueo>.

Para desbloquear:

DEQUEUE <nombre_objeto_bloqueo>.

## BATCH-INPUT

Los BATCH-INPUT son programas que se utilizan para realizar una serie de tareas sin intervención del usuario.
Estas tareas se pasan a un fichero de colas, en forma de sesiones de batch input. Posteriormente, Para ver este fichero de cola iremos al menú “Sistema”, “Servicios”, “Batch input”, “Tratar” o el código de transacción SM35, para transferir los datos hacia las bases de datos SAP.

Cualquier proceso de diálogo o batch de un sistema SAP está controlado por dynpros. Las dynpros (programas dinámicos) contienen exactamente un paso de diálogo.
Y un paso de diálogo consta de las siguientes partes:

- Un evento PBO (Process Before Output), en el que se prepara una pantalla para la salida.
- Las entradas efectuadas por el usuario.
- Y un evento PAI (Process After Input), en el que se procesan las entradas realizadas por el usuario.

El funcionamiento de los BATCH-INPUT no es complicado si se sabe lo que se ha de hacer. Su codificación suele ser bastante larga.
Como un BATCH-INPUT es siempre complicado de explicar, lo mejor es ver un ejemplo donde se ve mucho mejor lo que se quiere hacer.

```
REPORT ZZJII14.
**********************************************************************
**
* BATCH-INPUT
**********************************************************************
**

DATA: BEGIN OF BDC_TAB OCCURS 0.
     INCLUDE STRUCTURE BDCDATA.
DATA: END OF BDC_TAB.
DATA: BEGIN OF TABLA OCCURS 0,
    NOMBRE LIKE RF02D-KUNNR,
    TELF LIKE KNA1-TELF1,
    FAX LIKE KNA1-TELFX,
    DIRECCION LIKE KNA1-ADRNR,
    END OF TABLA.
DATA: FICHERO LIKE RLGRAP-FILENAME VALUE 'C: \ZZJII.DAT'.
* PASO EL FICHERO A UNA TABLA INTERNA
CALL FUNCTION 'UPLOAD'
    EXPORTING
        FILENAME = FICHERO
    TABLES
        DATA_TAB = TABLA
    EXCEPTIONS
        CONVERSION_ERROR = 01
        INVALID_TABLE_WIDTH = 02
```

Paso del fichero a la tabla interna

```
        INVALID_TYPE = 03
        NO_BATCH = 04
        UNKNOWN_ERROR = 05.

OPEN DATASET 'IVAEINE' FOR OUTPUT IN BINARY MODE.
* GENERAR EDC-DATA (PASO DE LA TABLA INTERNA A LA EXTERNA).
LOOP AT TABLA.
   IF NOT TABLA-TELF IS INITIAL.
    CASE TABLA-DIRECCION.
      WHEN 'BARCELONA'.
        CONCATENATE '93' TABLA-TELF INTO TABLA-TELF.
      WHEN 'HUESCA'.
        CONCATENATE '974' TABLA-TELF INTO TABLA-TELF.
      WHEN 'TARRAGONA'.
        CONCATENATE '977' TABLA-TELF INTO TABLA-TELF.
    ENDCASE.
   TRANSFER TABLA TO 'IVAEINE'.
  ENDIF.
ENDLOOP.
CLOSE DATASET 'IVAEINE'.
```

Modifico los datos del fichero. En este caso pongo el prefijo.

```
* LLAMADA AL BATCH-INPUT
CALL FUNCTION 'BDC_OPEN_GROUP'
   EXPORTING
      CLIENT = SY-MANDT
      GROUP = 'ZCURSOS'
      USER = SY-UNAME.
```

Abro el proceso del BATCH INPUT.

```
* GENERAR BDC_DATA (PASO DE LA TABLA INTERNA A LA EXTERNA)
OPEN DATASET 'IVAEINE' FOR INPUT IN BINARY MODE.
WHILE SY-SUBRC = 0.
    READ DATASET 'IVAEINE' INTO TABLA.
    IF SY-SUBRC = 0.
      REFRESH BDC_TAB.
      PERFORM DYNPRO USING:
            'X'   'SAPMF02D'  '0101',
            ' '   'RF02D-KUNNR' TABLA-NOMBRE,
            ' '   'RF02D-D0110'  'X',
            ' '   'BDC_OKCODE'   '/0',
            'X'   'SAPMF02D'    '0110',
            ' '   'KNA1-TELF1'   TABLA-TELF,
            ' '   'KNA1-TELFX'   TABLA-FAX,
            ' '   'KNA1-ADRNR'   TABLA-DIRECCION,
            ' '   'BDC_CODE'    '/11'.
      CALL FUNCTION 'BDC_INSERT'
         EXPORTING
```

CUERPO PRINCIPAL

```
            TCODE = 'XD02'
TABLES
            DYNPROTAB = BDC_TAB.
    ENDIF.
ENDWHILE.
CALL FUNCTION 'BDC_CLOSE_GROUP'.
CLOSE DATASET 'IVAEINE'.
DELETE DATASET 'IVAEINE'.
```

**Cierro el BATCH**

```
FORM DYNPRO USING DYNBEGIN NAME VALUE.
   IF DYNBEGIN = 'X'.
     CLEAR BDC_TAB.
     MOVE: NAME TO BDC_TAB-PROGRAM,
          VALUE TO BDC_TAB-DYNPRO,
          'X' TO BDC_TAB-DYNBEGIN.
     APPEND BDC_TAB.
   ELSE.
     CLEAR BDC_TAB.
     MOVE: NAME TO BDC_TAB-FNAM,
          VALUE TO BDC_TAB-FVAL.
     APPEND BDC_TAB.
 ENDIF.
 WRITE:/ BDC_TAB-FNAM, BDC_TAB-FVAL.
ENDFORM.
```

Dependiendo si es una pantalla o un campo lo añado de una forma u otra a la tabla BDC

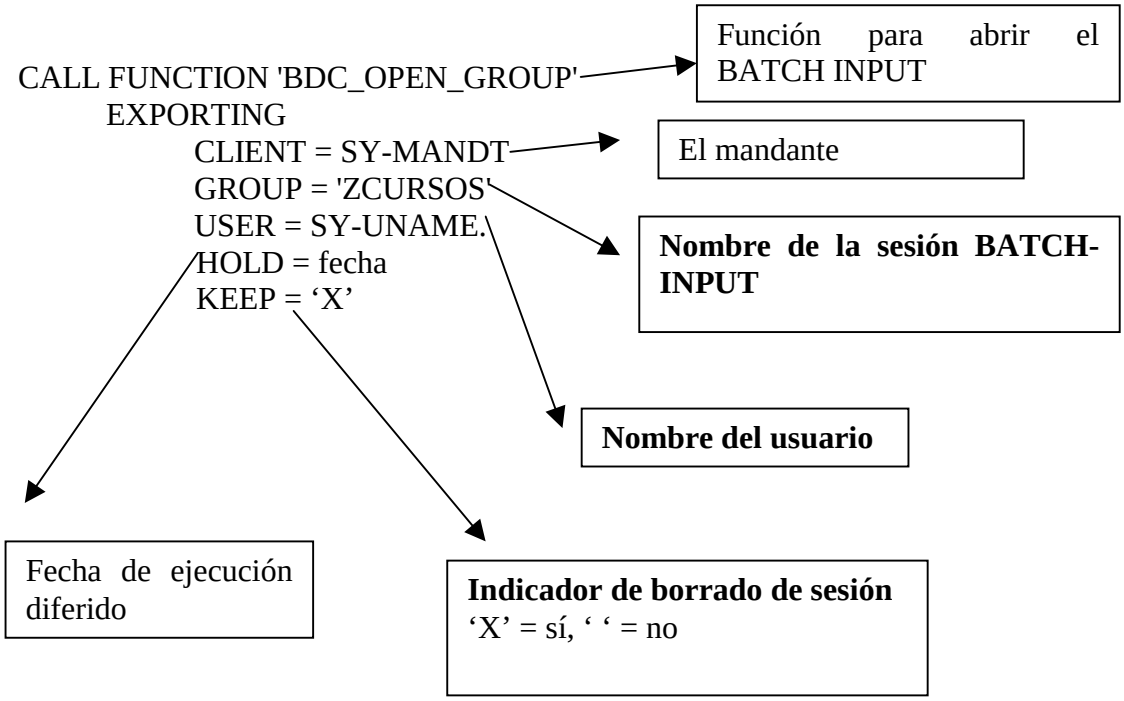
En este programa leo de un fichero secuencial llamado “fichero” (que es una variable con la ruta del fichero).
La función UPLOAD pasa el contenido de un fichero a una tabla interna

Después leo la tabla interna pasando los datos de la tabla interna a otro fichero secuencial (llamado: ‘IVAEINE’), realizando las siguiente correcciones:

1º Solo grabaré los que tienen teléfono.
2º Aquellos que tengan teléfono, miraré su provincia y les pondré el prefijo correspondiente (No compruebo si ya tiene prefijo).

Cuando ya tengo el fichero secuencial con todos los datos corregidos, lo cierro para poder abrirlo como lectura (al estar abierto como OUTPUT no se puede leer).

Después abro el proceso BATCH-INPUT, con la siguiente función:




Después voy leyendo registros comprobando que no ha llegado al final del fichero.
Si no estoy en el final es cuando paso a una DYNPRO lo que tiene que hacer.

A la DYNPRO le paso los siguientes parámetros: Dynbegin, Programa, Dynpro, nombre del campo y contenido del campo.

Dynbegin -> Indica si es el inicio de una nueva pantalla.

Programa -> Nombre del programa al que llamamos.

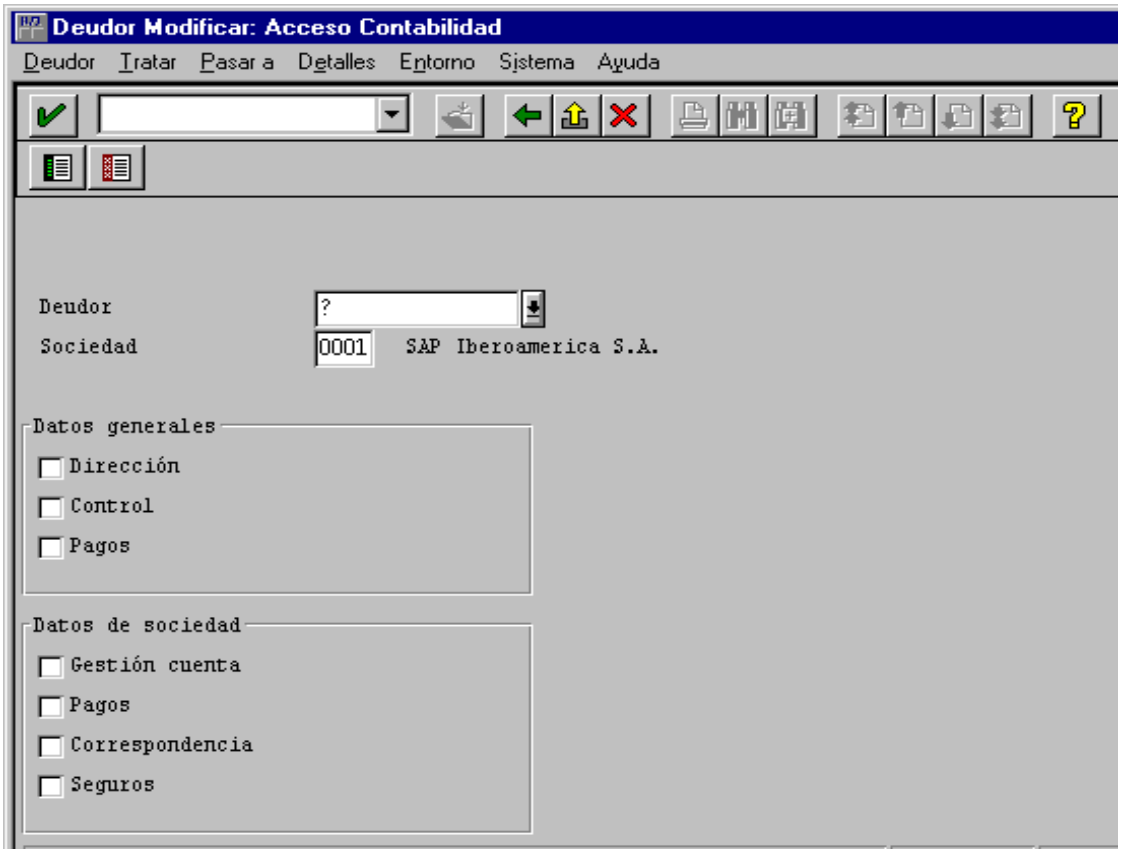
Dynpro -> Número de Dynpro que tiene el programa, o el número de pantalla.

Nombre del campo -> Es el nombre del campo de la pantalla del programa que hemos llamado.

Contenido del campo -> El valor que tendrá el campo.

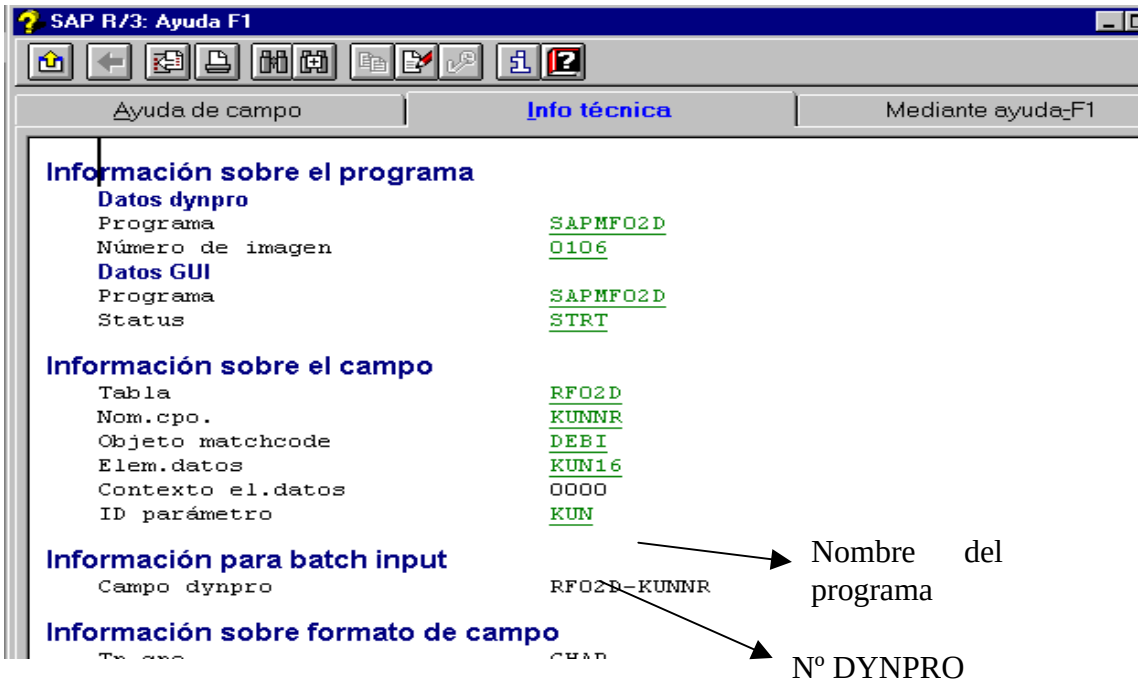
Para saber el nombre de pantalla, su Dynpro, el nombre de campo, a qué tabla pertenece, etc. haremos lo siguiente:

En el ejemplo la pantalla está situada, desde el menú principal en:
"Menú financias", "Gestión financiera", "Deudores". Al pulsar en "Deudores" nos saldrá la pantalla de deudores, vamos al menú "Datos maestros" y pulsamos en "modificar" (por ejemplo) y nos saldrá la siguiente pantalla:




A esta pantalla la llamaremos Pant-1.

Si nos posicionamos sobre cualquier campo (ya sea de entrada de datos o los checkbox o cualquier otro tipo de campo) y pulsamos F1 nos saldrá una pantalla con la información referente a ese campo y también referente al programa en cuestión. La pantalla que sale es la siguiente:



Estos dos datos los utilizaremos para hacer el BATCH-INPUT, en el programa tenemos lo siguiente:

```
PERFORM DYNPRO USING:
            'X'   'SAPMF02D'  '0106',
            ' '   'RF02D-KUNNR' TABLA-NOMBRE,
            ' '   'RF02D-D0110'  'X',
            ' '   'BDC_OKCODE'   '/0',
            'X'   'SAPMF02D'     '0110',
            ' '   'KNA1-TELF1'   TABLA-TELF,
            ' '   'KNA1-TELFX'   TABLA-FAX,
            ' '   'KNA1-ADRNR'   TABLA-DIRECCION,
            ' '   'BDC_CODE'     '/11'.
```

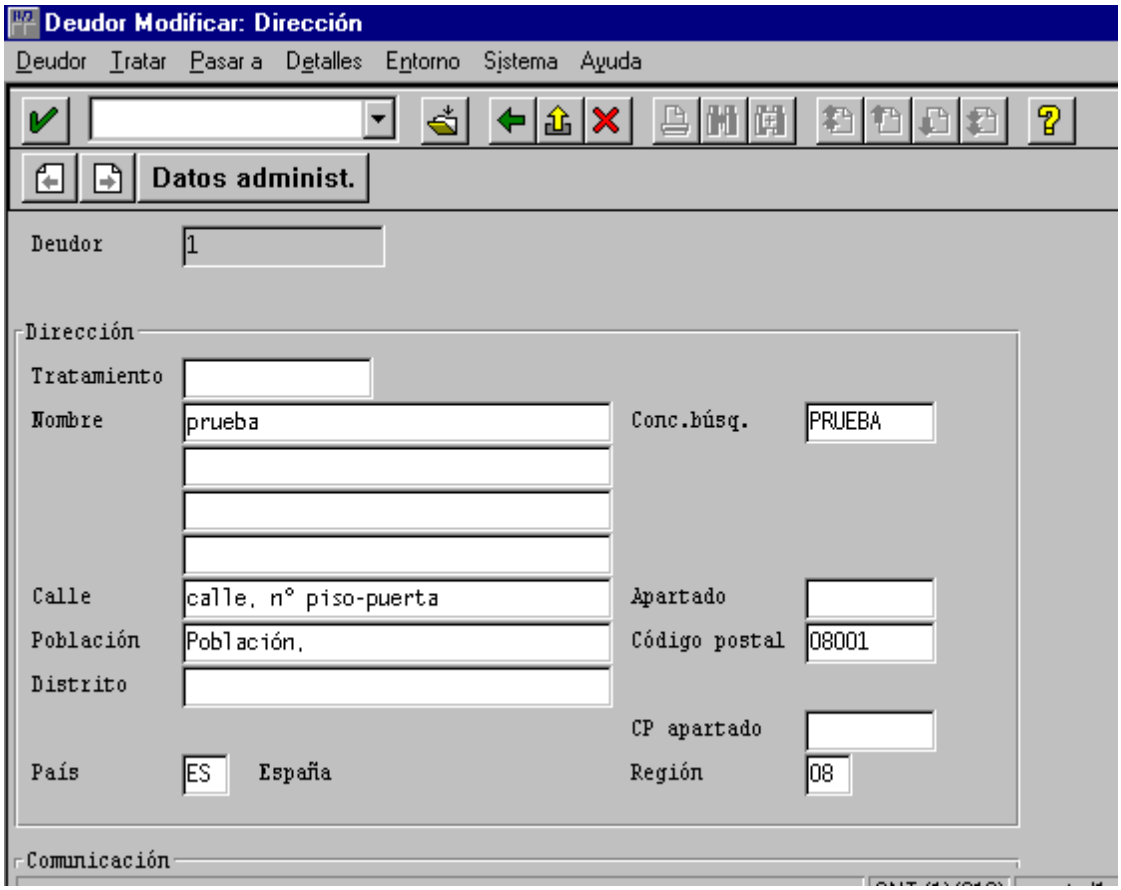
En la primera línea le indicamos que cargue el programa ‘SAPMF02D’ con la DYNPRO ‘0106’ y le indicamos al sistema que es el inicio de una nueva pantalla.

En la segunda línea ponemos en el campo ‘RF02D-KUNNR’ (El campo Deudor) el valor TABLA-NOMBRE (Este valor está guardado en el fichero).

En la tercera línea activamos el checkbox Dirección (dentro del recuadro Datos Generales, en la pantalla: Pant-1). El nombre del checkbox es 'RF02D-D0110' (Recordar que para saberlo, nos posicionamos sobre el checkbox y pulsamos F1) y su valor será ‘X’ (para activarlo).

En la cuarta línea hacemos un ENTER, se hace poniendo en el nombre del campo: BDC_OKCODE y le damos el valor ‘/0’ (Código del Enter).

La quinta línea nos va a la pantalla de dirección que es la siguiente, tiene el nombre ‘SAPMF02D’y la DYNPRO ‘110’ y le indicamos que es el inicio de una nueva pantalla, la pantalla es la siguiente:




La llamaremos Pant-2.

En el resto de líneas, excepto en la última, guardo él numero de teléfono, fax y la dirección en los campos 'KNA1-TELF1', 'KNA1-TELFX' y 'KNA1-ADRNR' respectivamente. Para saber su nombre nos posicionamos en el campo y pulsamos F1.

En la última línea pulsamos F11, que sirve para grabar los datos introducidos. Para ello en el nombre ponemos ‘BDC_OKCODE’ y el valor será ‘/11’.

Como hemos visto llamamos a un subprograma al que le pasamos como parámetros todos las operaciones a realizar.
El subprograma llamado se llama DYNPRO, ahí añadimos a la tabla BDC_DATA las operaciones a realizar (Esta tabla ha de tener la misma estructura que la tabla BDCDATA). El subprograma sería el siguiente:

```
FORM DYNPRO USING DYNBEGIN NAME VALUE.
    IF DYNBEGIN = 'X'.
      CLEAR BDC_TAB.
      MOVE: NAME TO BDC_TAB-PROGRAM,
           VALUE TO BDC_TAB-DYNPRO,
           'X' TO BDC_TAB-DYNBEGIN.
      APPEND BDC_TAB.
    ELSE.
      CLEAR BDC_TAB.
```

```
      MOVE: NAME TO BDC_TAB-FNAM,
          VALUE TO BDC_TAB-FVAL.
      APPEND BDC_TAB.
  ENDIF.
  WRITE:/ BDC_TAB-FNAM, BDC_TAB-FVAL.
ENDFORM.
```

Si DYNBEGIN tiene una 'X' quiere decir que llamamos a una pantalla, y cuando llamamos a una pantalla hemos de poner el nombre del programa (BDC_TAB-PROGRAM), la dynpro que tiene (BDC_TAB-DYNPRO) y si es inicio de una nueva pantalla (BDC_TAB-DYNBEGIN). Después con la orden APPEND la añadimos.

Si no llamamos a una nueva pantalla hemos de poner en qué campo introducimos datos o si pulsamos alguna tecla de función.
En todo caso hemos de poner que campo es (BDC_TAB-FNAM) y el valor que tiene (BDC_TAB-FVAL).

Después de añadirlo a la tabla hay que ejecutar una función del SAP para insertarlo en el fichero de colas (o eso creo). Se realiza a través de la función:

```
CALL FUNCTION 'BDC_INSERT'
          EXPORTING
             TCODE = 'XD02'
          TABLES
             DYNPROTAB = BDC_TAB.
```

TCODE es el código de la transacción

En TABLES ponemos en qué tabla están las operaciones a realizar por el BATCH-INPUT.

Cuando ya hemos introducido todos los datos del fichero hemos de cerrar el proceso de BATCH-INPUT, que se realiza a través de la función interna:

```
CALL FUNCTION 'BDC_CLOSE_GROUP'.
```

Después cerramos el fichero secuencial con los datos y a continuación lo borramos.

A partir de ahora, doy una explicación de un BATCH INPUT que sale en un manual bastante bueno, que viene a complementar lo comentado por mí antes.
Seguro que os aclarará las dudas que os queden de los BATCH-INPUT:

TABLA BDCDATA

| PROGRAM | DYNPRO | DYNBEGIN | FNAM | FVAL |
|---|---|---|---|---|
| Programa1 | Nnnn | X | | |
| | | | Campo_1 | Valor_1n |
| | | | Campo_2n | Valor_2n |
| | | | BDC_OKCODE | /0 |
| Programa2 | Mnnn | X | | |
| | | | Campo_1m | Valor_1m |
| | | | Campo_2m | Valor_2m |
| | | | BDC_OKCODE | /11 |

Antes de escribir el ABAP/4 de batch input, hay que obtener la información necesaria para su realización, para ello hay que simular paso a paso la función de la aplicación que queremos simular, la información de PROGRAM, DYNPRO, FNAM (nombre de campo para BATCH_INPUT), y TCODE, la obtenemos del menú “Sistema”, “Status”, “información técnica”.

El proceso a seguir para llenar la tabla BDCDATA es

1/ PROGRAM, DYNPRO, DYNBEGIN.

2/ FNAM, con el nombre del campo de DYNPRO a tratar, FVAL el nuevo valor.

3/ Repetir el paso 2 tantas veces como sea necesario.

4/ DBC_OKCODE código de terminación para pasar al siguiente estado
/0 intro /11 grabar /nn → PF nn

**STATUS DE LAS SESIONES**

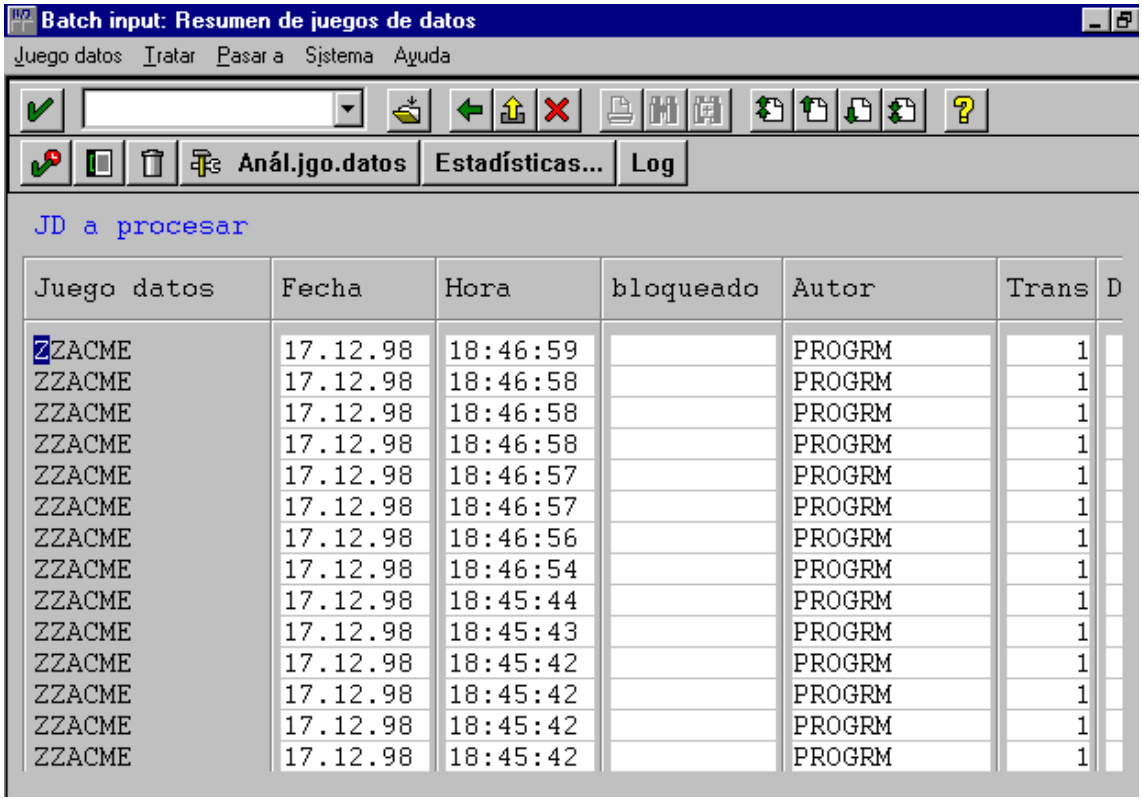
Acceso: SISTEMA -> SERVICIOS -> BATCH_INPUT

Procesadas terminada OK Incorrecta terminada con errores
A procesar grabada sin procesar Siendo grabada
Siendo procesada BATCH

**CONTROLES**

/N saltar transacción.
/BDNO salir BATCH INPUT.
/BDE visualizar todo.
/BDEL borrar transacción.
/BDA visualizar solo errores.
**¿COMO VER LOS PROCESOS?**

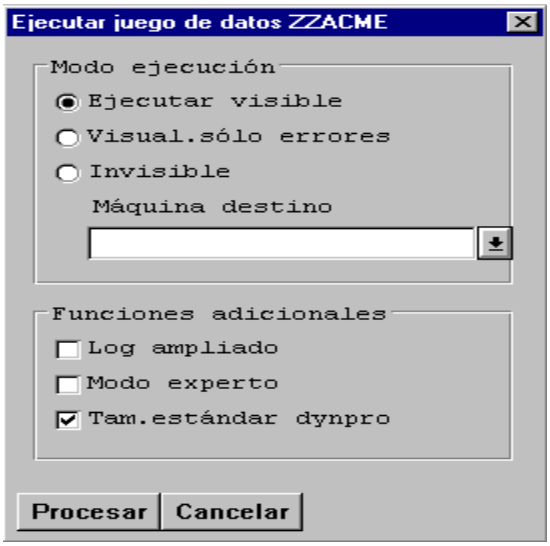
Como ya hemos dicho antes, SAP guarda en una cola los batch input generados. Para poder acceder a ellos desde cualquier parte del SAP hay que ir al menú “sistema”, “servicios”, “batch input”, “tratar”.
Nos saldrá una pantalla. Pulsando en el botón que pone resumen, saldrá otra pantalla con los batch input que hay en la cola. La pantalla es la siguiente:




| Juego datos | Fecha | Hora | bloqueado | Autor | Trans | D |
|---|---|---|---|---|---|---|
| ZZACME | 17.12.98 | 18:46:59 | | PROGRM | 1 | |
| ZZACME | 17.12.98 | 18:46:58 | | PROGRM | 1 | |
| ZZACME | 17.12.98 | 18:46:58 | | PROGRM | 1 | |
| ZZACME | 17.12.98 | 18:46:58 | | PROGRM | 1 | |
| ZZACME | 17.12.98 | 18:46:57 | | PROGRM | 1 | |
| ZZACME | 17.12.98 | 18:46:57 | | PROGRM | 1 | |
| ZZACME | 17.12.98 | 18:46:56 | | PROGRM | 1 | |
| ZZACME | 17.12.98 | 18:46:54 | | PROGRM | 1 | |
| ZZACME | 17.12.98 | 18:45:44 | | PROGRM | 1 | |
| ZZACME | 17.12.98 | 18:45:43 | | PROGRM | 1 | |
| ZZACME | 17.12.98 | 18:45:42 | | PROGRM | 1 | |
| ZZACME | 17.12.98 | 18:45:42 | | PROGRM | 1 | |
| ZZACME | 17.12.98 | 18:45:42 | | PROGRM | 1 | |
| ZZACME | 17.12.98 | 18:45:42 | | PROGRM | 1 | |

Fig. Batch-input.

Haciendo doble clic sobre los batch input en cola ejecutaremos el batch input. Si lo hacemos saldrá una pantalla de cómo queremos ejecutarlo. La pantalla que sale es la siguiente:

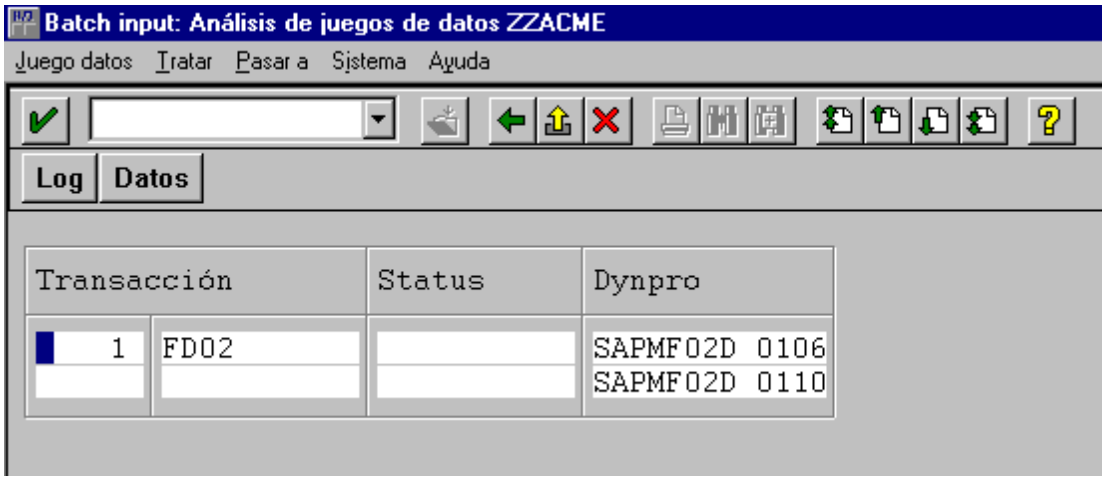



En el modo de ejecución podemos hacer que el batch input se ejecute de tres formas diferentes:

- Ver todo el proceso (Opción “Ejecutar visible”).
- Sólo mostrará los errores del batch input (Opción “Visul. Sólo errores”)
- No mostrará nada (Opción “invisible”).

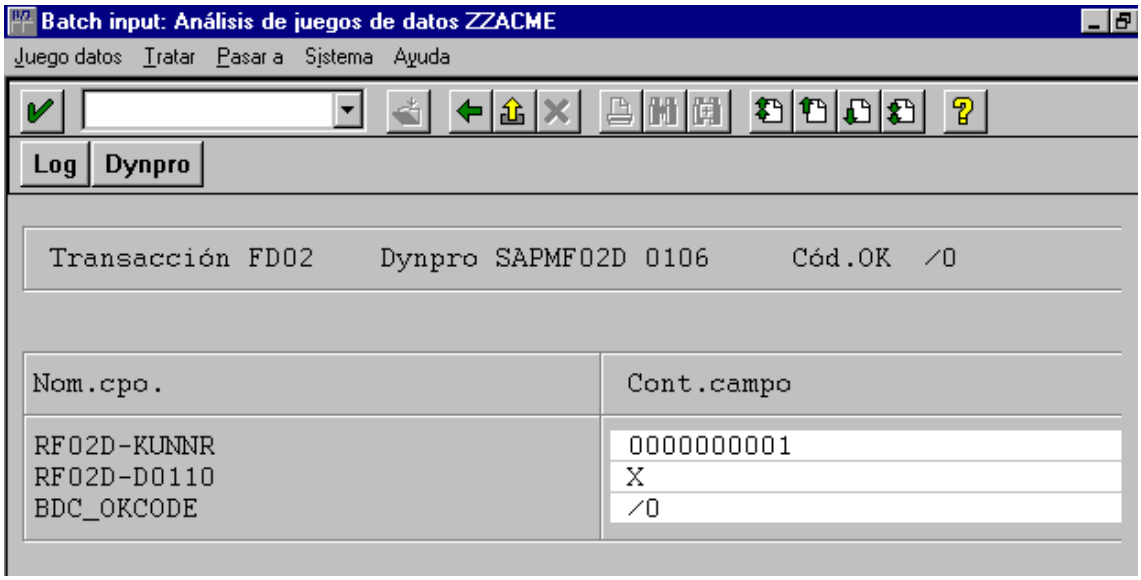
Cuando esté todo de acuerdo pulsaremos enter o el botón de procesar y se comenzará a ejecutar, como nosotros queremos, el batch input.

También hay otras posibilidades si seleccionamos un batch input y pulsamos el botón “Anál.Jgo.datos”. Sale la pantalla siguiente:

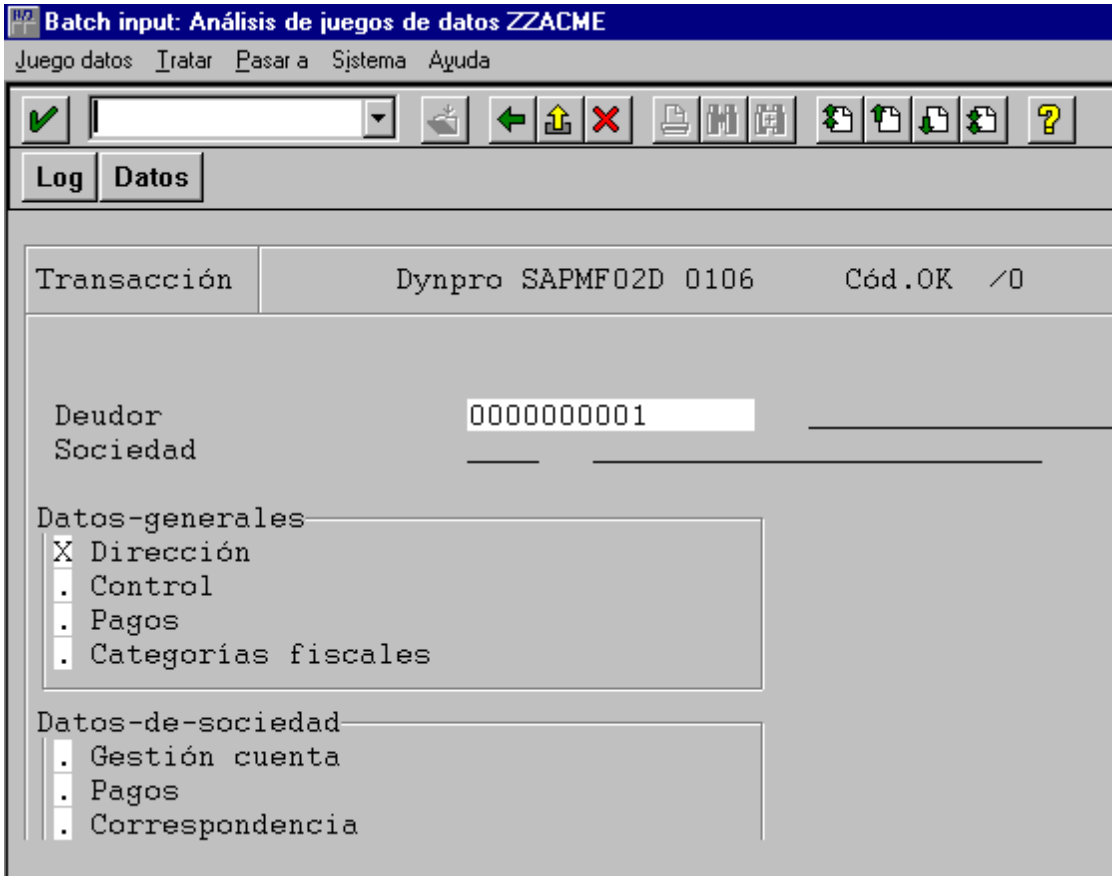



Aquí se muestra la transacción y sus Dynpros correspondientes.

Si seleccionamos uno y pulsamos el botón “Datos” sale una pantalla con los datos de ese batch input, la pantalla que sale es la siguiente:




Y si además pulsamos el botón “Dynpro” se ven los campos seleccionados, la pantalla que sale es esta:




¿CÓMO CREAR UN BATCH INPUT DE FORMA AUTOMATICA?

Con esto quiero decir que SAP nos permite ir a las pantallas y campos a utilizar y él automáticamente nos hará el batch input. Solo nos pone lo principal, es decir, lo que valdrán esos campos ya es cosa nuestra.
Desde cualquier pantalla vamos al menú “sistema”, “servicios”, “batch input”, “tratar”. (Fig. batch-input).

Ahí pulsamos el botón grabación o vamos al menú “Pasar a”, “grabación” o F8. Y entonces nos saldrá la pantalla siguiente:

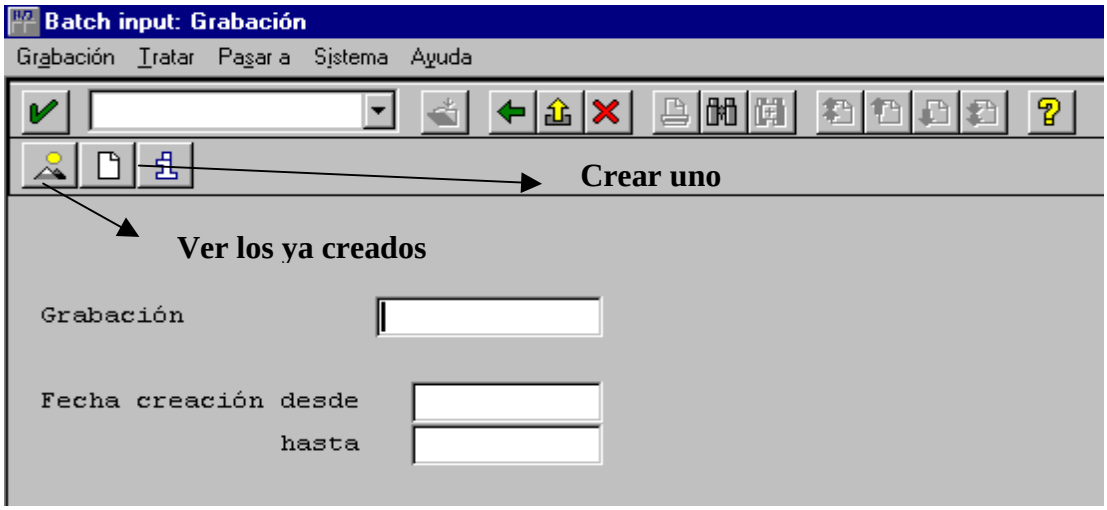
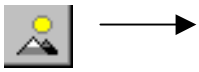


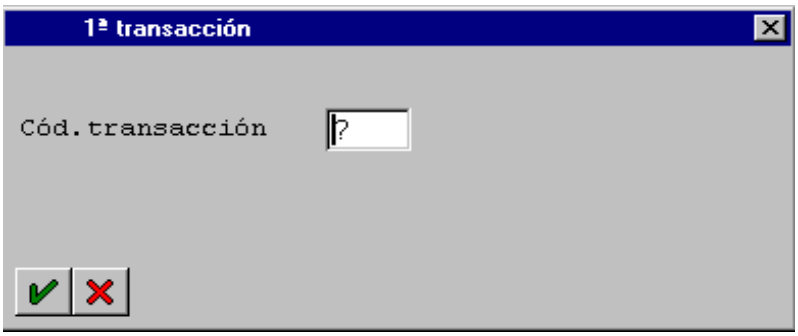

Fig. Grabación.

En grabación se pone el nombre del programa donde se guarda todo el proceso. Como muestra, le pondremos el nombre "ENRIQUE".
Y le daremos al botón que tiene el icono de la hoja en blanco o vamos al menú "grabación", "crear" o pulsamos F5.
Si le damos al botón del icono de la montaña o en el menú "Pasar a", "Resumen" o F8 iremos a los ya creados y los podremos ver, modificar, borrar, etc. Este es el icono:

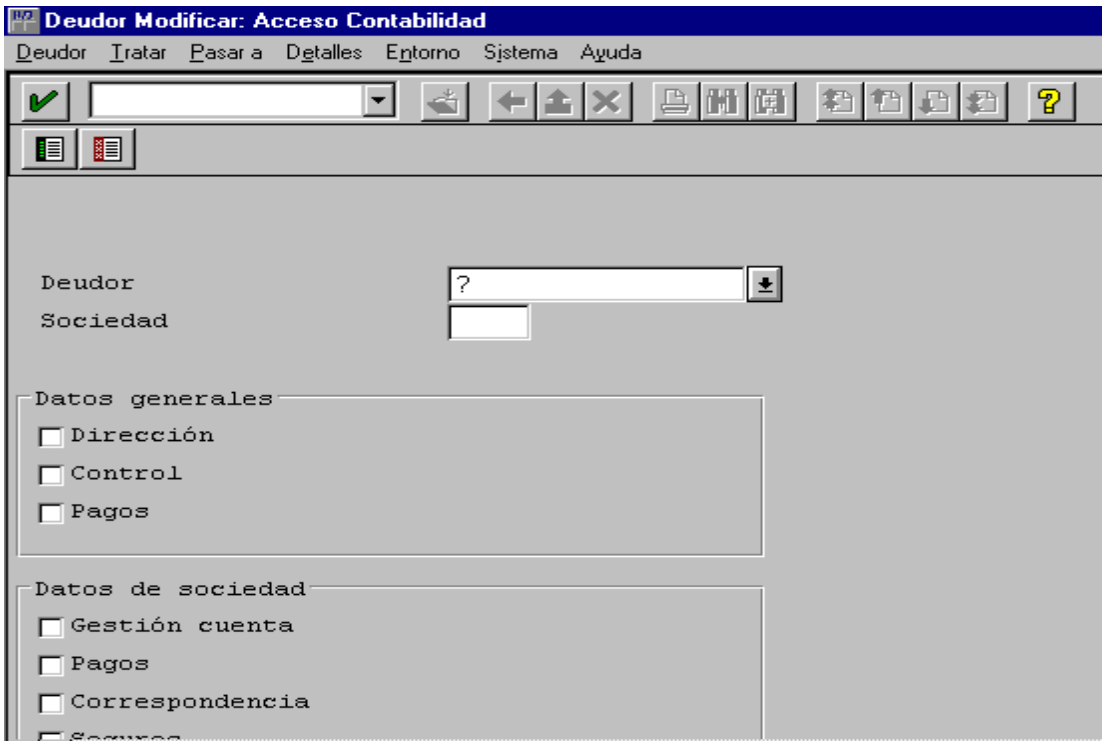


Resumen o F8

Cuando le demos a crear saldrá esta pantalla:

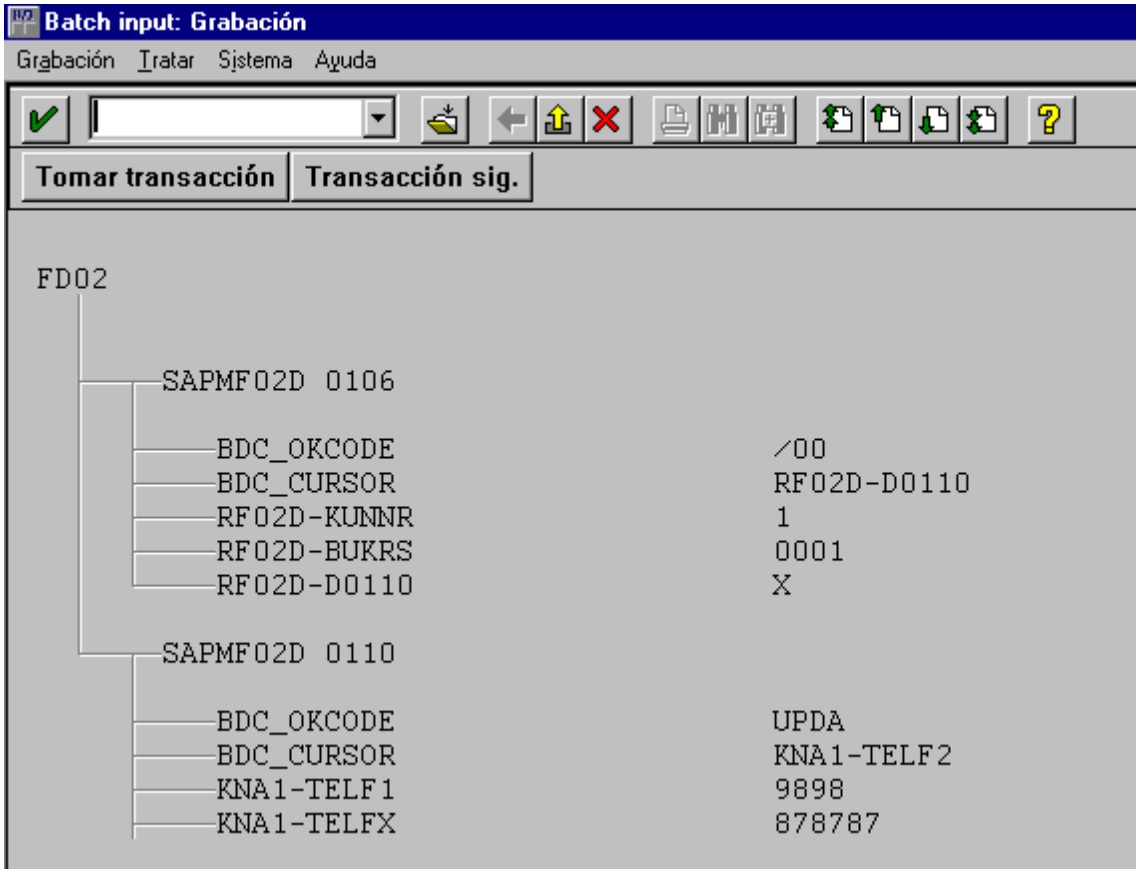



Aquí nos pide la transacción del programa. En nuestro caso pondremos la transacción: FD02. Y pulsaremos Enter.
Veremos que se nos ha ido al programa del que queremos hacer el batch input.
En nuestro ejemplo iremos a la pantalla siguiente:



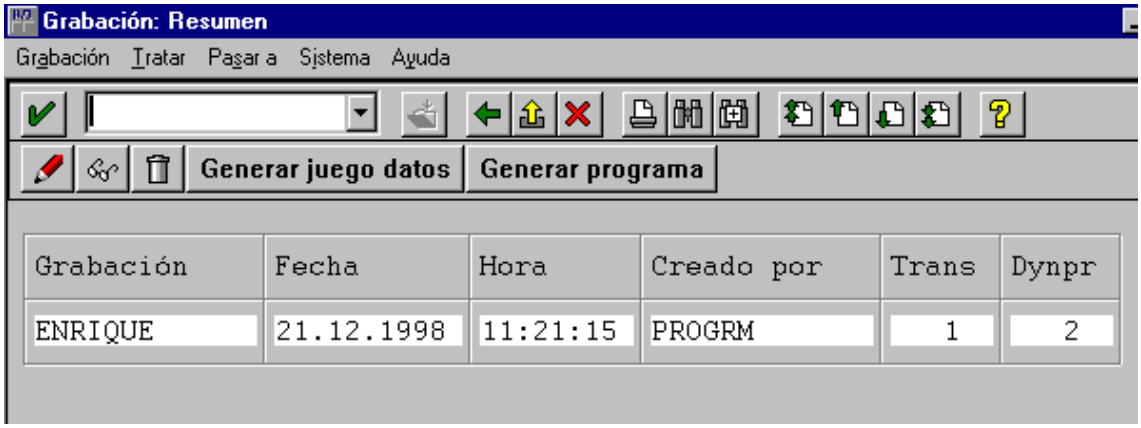

En el programa introduciremos algún dato (correcto por supuesto) en los campos que vamos a utilizar para hacer el batch input.

Después de todo esto lo grabamos pulsando F11 o con el icono de siempre, al hacer esto sale la siguiente pantalla (en la página siguiente):

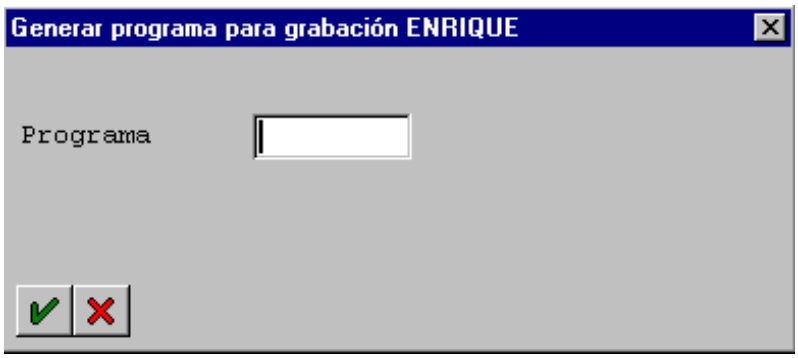



Se ve como salen los campos en que hemos introducido algo y las pantallas a las que hemos ido.

Después de esto lo grabaremos y nos preguntará si deseamos tomar la transacción. Diremos que sí y la transacción será tomada, o también pulsando el botón “Tomar transacción”.
Después volvemos a través del menú “Grabación”, “finalizar” o SHIFT+F3. Y volveremos a la pantalla de grabación (Fig. Grabación). Si pulsamos sobre el icono de resumen o F8 saldrá la siguiente pantalla con los procesos creados:



El icono del lápiz permite modificar el proceso de la lista.
Las gafas permiten ver los procesos de la lista.
Y la papelera borra cualquier proceso que esté en la lista.
Y si pulsamos sobre el botón "Generar programa" nos preguntará el nombre del programa donde lo queremos que lo copie, la pantalla que sale es esta:




Aquí le pondremos: ZZJII10. Pero podremos ponerle cualquier nombre. Seguidamente nos saldrá la misma pantalla que cuando creamos un nuevo programa (o sea los atributos del sistema). Cuando los hallamos grabado nos saldrá el código fuente de este programa.

AVISOS

Con los batch input hay un problema si se ejecuta un batch input en dos o más ordenadores con resoluciones diferentes.
Es decir si en una pantalla se muestran 4 campos (pero tiene 6) SAP toma 4 esos campos como una pantalla. Si movemos la pantalla, SAP asume que los campos 5 y 6 están en otra pantalla.
Para arreglar esto, en teoría, si vamos al icono de antena 3 (el icono que salen los colores rojo, verde y azul arriba a la izquierda) y pulsamos ahí sale un menú contextual, después pulsamos en "default size". La pantalla se pone en la resolución, fuente de letras, número de colores, etc. que tiene por defecto SAP. Pero hay ordenadores (como por ejemplo en el que estoy haciendo el manual) que si lo pulsas, sale el siguiente error:

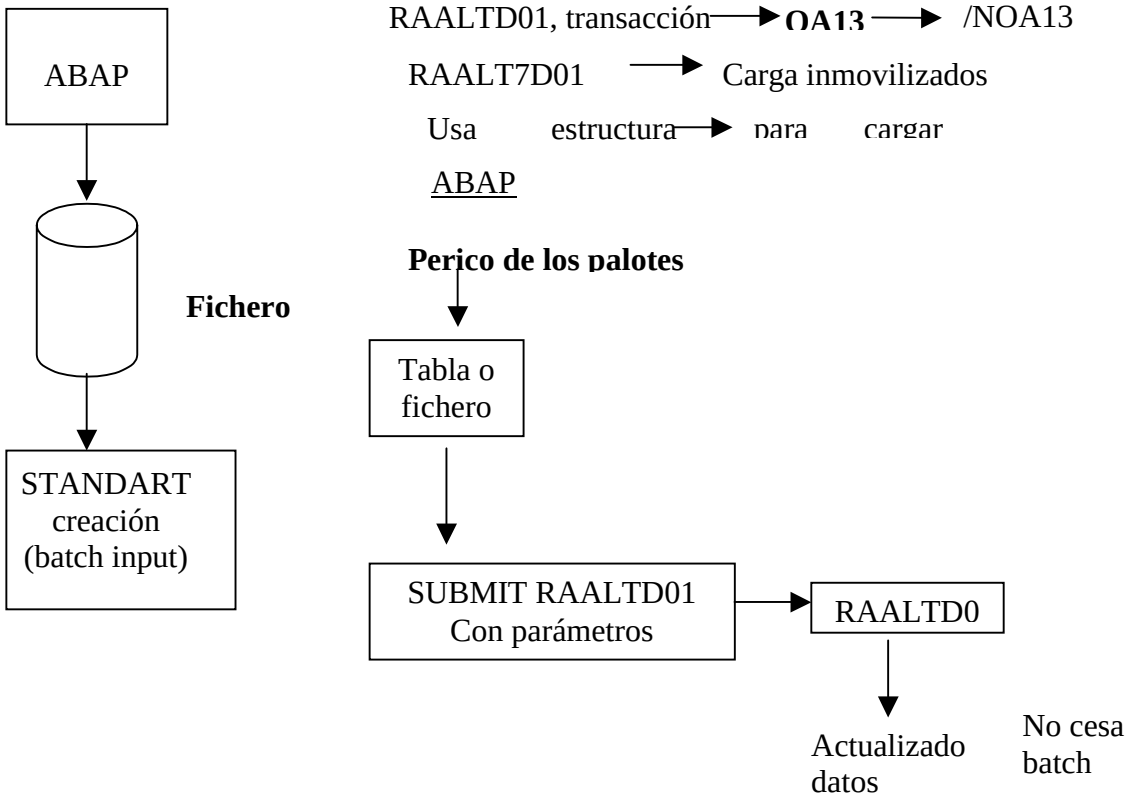

En definitiva, rezar que no te salga este error o que todas las pantallas tengan la misma resolución.

## DIRECT INPUT

Existen standars creados a medida para las empresas en cuestión, cuya función consiste en pasar batch-inputs, al ejecutar una transacción determinada.
También se pueden llamar con un SUBMIT desde un ABAP creado al efecto, que carga un fichero con los datos elegidos.

Sería: ejemplo RMMHBIMZ
RMDATGEN
RMDATIND
} INMIW1

Batch – input maestro materiales




En el caso del DIRECT INPUT las estructuras deben estar llenas, eso quiere decir que los campos no actualizados también se deben informar con '/', aunque se usan includes que ya tienen estos casos previstos y por tanto no se tienen que picar en el ABAP.

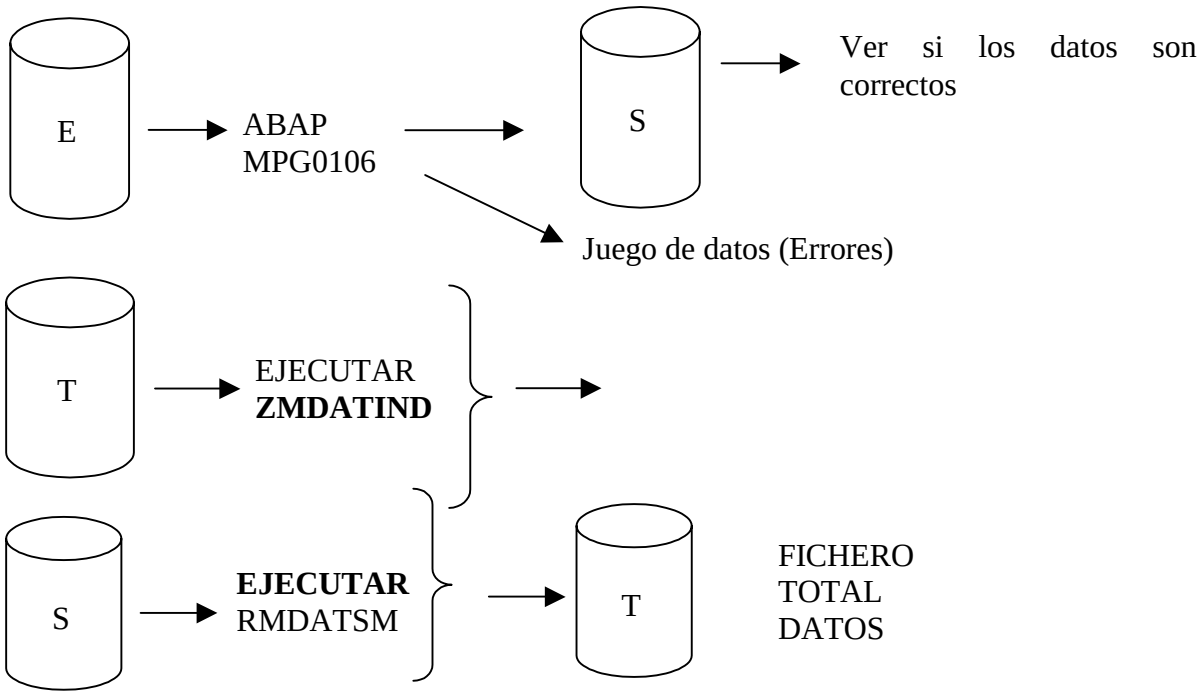
Todos los R* → son los creados a las empresas por SAP en casos especializados.

BMM00 → Estructura datos transacción para batch – input, maestro de materiales.

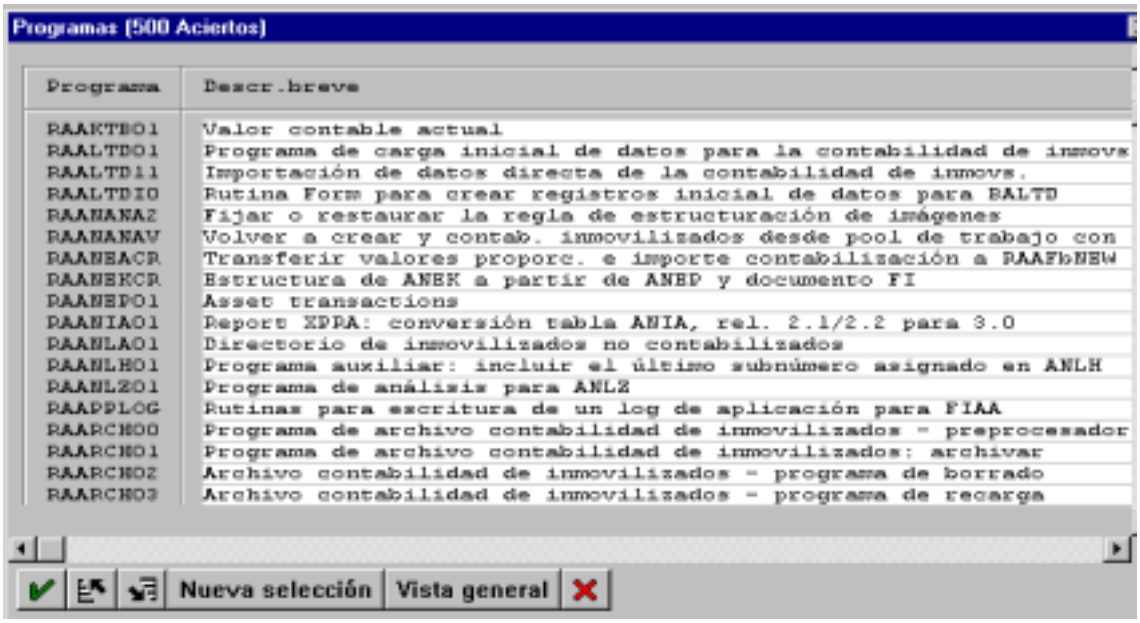
BMMH1 → Datos principales para maestro-material batch-input.
RAALTD01 → Carga de inmovilizados.
/NO13 → Batch-input de datos de AH.

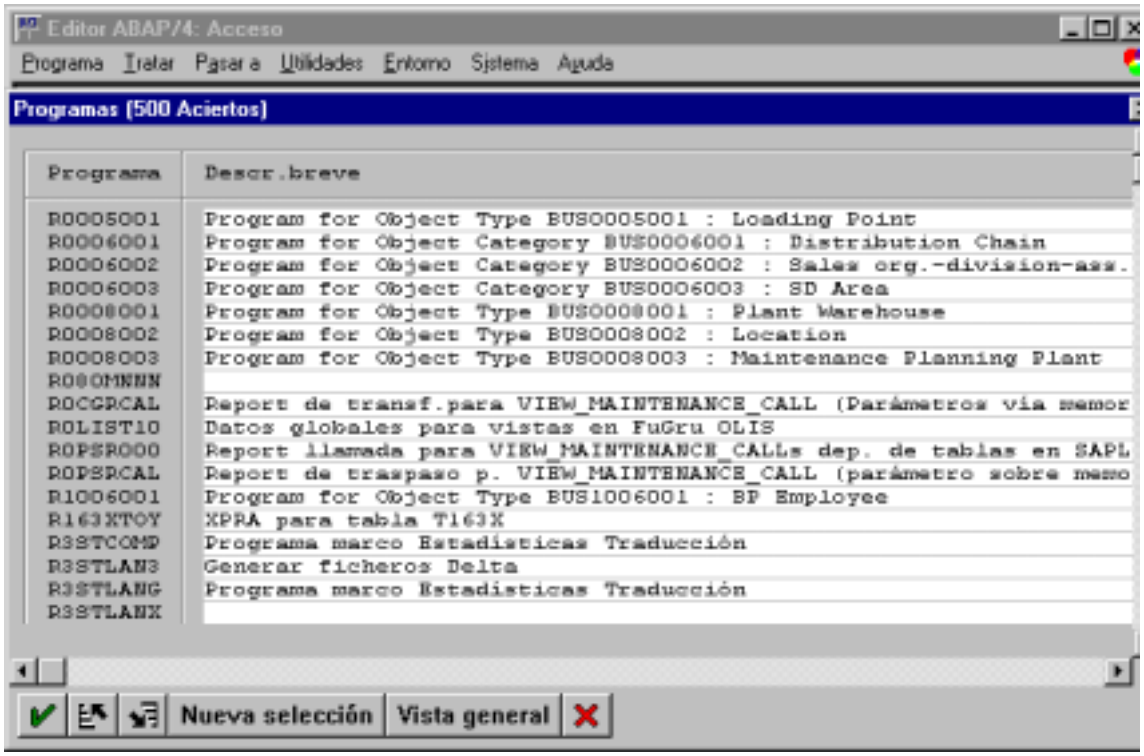
RC2ZM ⟶ Estructura con VERID.

ZMPG0106
Creación de ficheros base para introducir en la pantalla de selección del RMDATIND (BATCH-INPUT).
Desde Excel se toma el fichero con 'UPLOAD', se modifica registro a registro grabando el fichero de salida para batch-input con formato preestablecido por el standard.




Lista de programas relacionados con el direct input.




Editor ABAP/4: Acceso
Programa Tratar Pasar a Utilidades Entorno Sistema Ayuda
Programas (500 Aciertos)
Programa
Descr.breve
R0005001 Program for Object Type BUS0005001 : Loading Point
R0006001 Program for Object Category BUS0006001 : Distribution Chain
R0006002 Program for Object Category BUS0006002 : Sales org.-division-ass.
R0006003 Program for Object Category BUS0006003 : SD Area
R0008001 Program for Object Type BUS0008001 : Plant Warehouse
R0008002 Program for Object Type BUS0008002 : Location
R0008003 Program for Object Type BUS0008003 : Maintenance Planning Plant
R0@OMNNN
ROCGRCAL Report de transf.para VIEW_MAINTENANCE_CALL (Parámetros vía memor
ROLIST10 Datos globales para vistas en FuGru OLIS
ROPSR000 Report llamada para VIEW_MAINTENANCE_CALLs dep. de tablas en SAPL
ROPSRCAL Report de traspaso p. VIEW_MAINTENANCE_CALL (parámetro sobre memo
R1006001 Program for Object Type BUS1006001 : BP Employee
R163XTOY XPRA para tabla T163X
R3STCOMP Programa marco Estadísticas Traducción
R3STLAN3 Generar ficheros Delta
R3STLANG Programa marco Estadísticas Traducción
R3STLANX
Nueva selección
Vista general

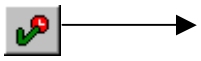
## REPORT INTERACTIVO

Los reports interactivos es una forma de combinar las instrucciones de formato de listados, los eventos y pantallas de selección, la unión de estas tres cosas nos permite, por ejemplo, que un usuario introduzca un dato (a través de las ordenes de introducción de datos) seguidamente controlar si el usuario a pulsado alguna tecla (a través de los eventos) y por último dependiendo de que tecla se ha pulsado visualizar un tipo de listado u otro(a través de la ordenes de formateo de listados).

A continuación mostraré una serie de ejemplos sencillos pero combinan las dos o tres elementos anteriormente dichos.

EJEMPLO 1

Retomemos el pequeño ejemplo que mostraba cuando explicaba las pantallas de selección, pero esta vez lo ampliaremos. En el ejemplo anterior pedíamos un dato (que es el nombre de una tabla) y cuando pulsemos este botón:

 Tomar datos o F8

Este botón sirve para realizar las ordenes que estén después de los SELECTION-SCREEN que halla al principio del programa.

Después de pulsar este botón, visualizare en que tablas lógicas aparece la tabla introducida.

A continuación escribe el programa completo para después comentarlo.

```
*&
*&---------------------------------------------------------*
*& Descripción:Consulta para conocer en qué Base de Datos
*&  Lógica  se  encuentra  asignada  una  tabla  de  SAP.
*
*&     Módulo:     Base(para     todos     los     módulos).
*
*&    Autor    :    S.B.C.    NORMA    CONSULTING,    S.A.
*
*&        Fecha        de        creación:        01.02.99
*
*&---------------------------------------------------------*
*&            LOG            DE            MODIFICACIONES:
*
*&-------ASUNTO-----------    ---AUTOR---     ---FECHA--- *
*&XXXXXXXXXXXXXXXXXXXXXXX    XXXXXXXXXXX     XX.XX.XXXX  *
*&---------------------------------------------------------*
*&
REPORT ZIBDD010 NO STANDARD PAGE HEADING LINE-SIZE 80.
*&---------------------------------------------------------*
```

```
*& Tablas de BDD
*&---------------------------------------------------------------------*
*&
TABLES: TLDB, TLDBT.
*&
*&---------------------------------------------------------------------*
*& Tablas internas
*&---------------------------------------------------------------------*
*&
DATA: BEGIN OF INTAB OCCURS 0,
      DBNAME  LIKE  TLDB-DBNAME,
      SYSTEM  LIKE  TLDB-SYSTEM,
      END  OF INTAB.
*&
*&---------------------------------------------------------------------*
*& Variables internas
*&---------------------------------------------------------------------*
*&
DATA : NUMLIN(3) TYPE I.
*&
*&---------------------------------------------------------------------*
*& Pantalla de selección
*&---------------------------------------------------------------------*
*&
SELECTION-SCREEN SKIP 3.
SELECTION-SCREEN BEGIN OF BLOCK AAA
                 WITH FRAME TITLE TEXT-001.
SELECTION-SCREEN SKIP 1.
SELECTION-SCREEN ULINE.
*
PARAMETERS : TABLA(4) TYPE C.
*
SELECTION-SCREEN ULINE .
SELECTION-SCREEN SKIP 1.
SELECTION-SCREEN END OF BLOCK AAA .
*&
*&---------------------------------------------------------------------*
*& Cabecera de página
*&---------------------------------------------------------------------*
*&
TOP-OF-PAGE.
*& Cabecera general.
  WRITE: /1 'Título  :', SY-TITLE.
  WRITE: /1 'Programa:', SY-REPID,
       /1 'Fecha   :', SY-DATUM,
       /1 'Página  :', SY-PAGNO.
  WRITE: /1 'Usuario :', SY-UNAME,
       /1 'Hora    :', SY-UZEIT.
  ULINE. SKIP 2.
*               "Cargar cabecera
```

Tablas donde están los datos referentes a las bases de datos lógicas

Pido el nombre de la tabla.

Cada vez que se cambie de página escriba esta cabecera

```
  FORMAT INTENSIFIED OFF.
*&
*&---------------------------------------------------------------------*
*& Selección
*&---------------------------------------------------------------------*
*&
START-OF-SELECTION.
*& Lectura de tabla madre.
*&
  SELECT * FROM TLDB.
    CHECK TLDB-EINTRAG CS TABLA.
         LOOP AT INTAB WHERE DBNAME = TLDB-DBNAME
        AND SYSTEM =  TLDB-SYSTEM.
    ENDLOOP.
*
    IF SY-SUBRC NE 0.
     MOVE-CORRESPONDING TLDB TO INTAB.
     APPEND INTAB.
    ENDIF.
  ENDSELECT.
*
  SORT INTAB.
*
  ULINE.
  FORMAT COLOR 1.
  WRITE: / SY-VLINE, 'BASES DE DATOS LOGICAS QUE CONTIENEN',
                TABLA, 80 SY-VLINE.
  ULINE.
*
  DESCRIBE TABLE INTAB LINES NUMLIN.
*
  IF NUMLIN GT 0.
    FORMAT COLOR 3.
    LOOP AT INTAB.
      SELECT SINGLE * FROM TLDBT WHERE SPRAS = SY-LANGU
                           AND DBNAME = INTAB-DBNAME
                          AND SYSTEM = INTAB-SYSTEM.
      IF SY-SUBRC = 0.
       WRITE: / SY-VLINE,
              INTAB-DBNAME,
              INTAB-SYSTEM,
              TLDBT-TEXT,
            80 SY-VLINE.

      ENDIF .
    ENDLOOP.
  ELSE.
    FORMAT COLOR 6.
    WRITE: / SY-VLINE, 'No existen datos para la tabla:',
```

Evento de selección de datos

Averiguo si la tabla ya esta introducida en la tabla interna

Si la tabla no ha sido encontrada en el LOOP, muevo los datos de tabla de diccionario al header de la interna y después la añado

Ordeno la tabla interna

Realizo una especie de encabezado

Guardo en NUMLIN cuantas líneas o registros tiene la tabla interna

Si hay datos en la interna, me dispongo a visualizarlos

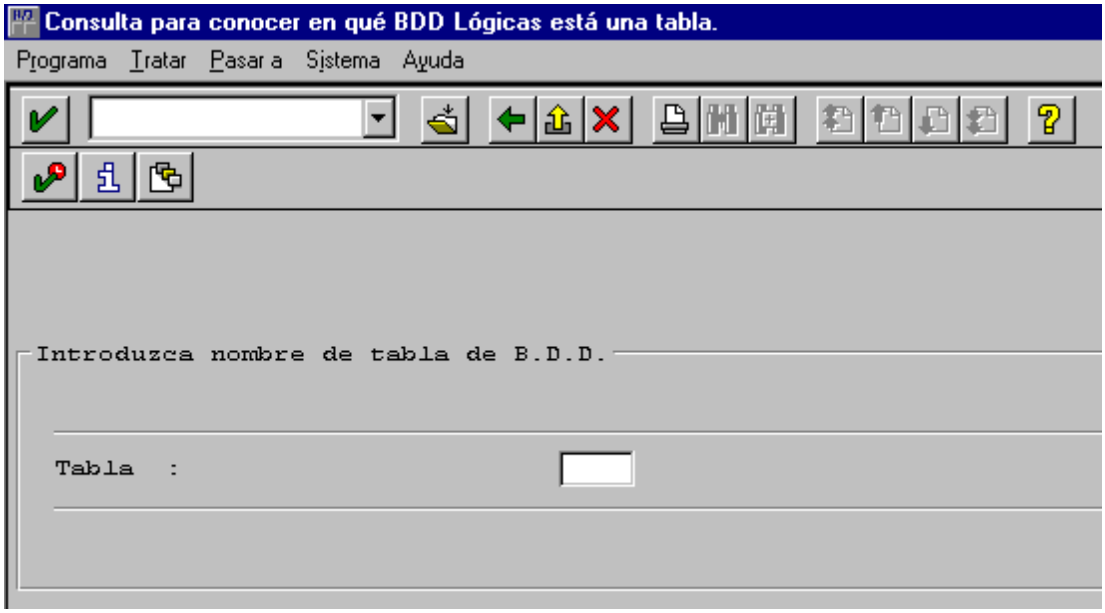
Busco otro dato referente a la tabla introducida

Si el dato a buscar es encontrado, muestro en que BDD lógica esta

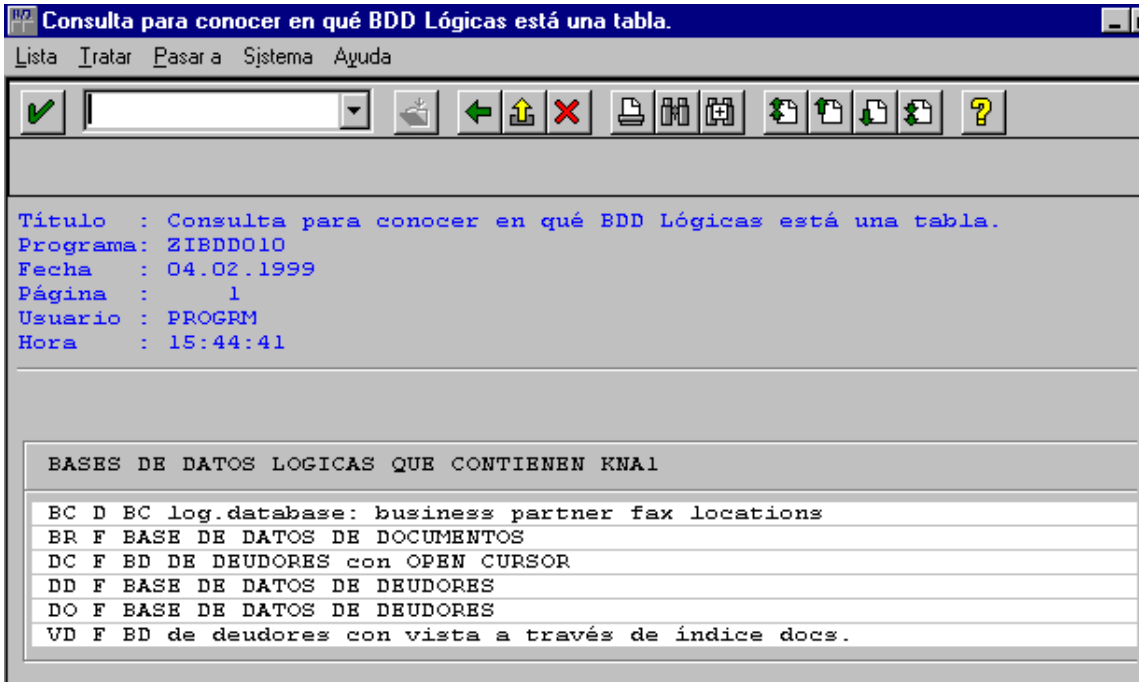
Si no hay datos en la tabla interna, muestro un mensaje diciéndolo

```
          TABLA, 80 SY-VLINE.
  ENDIF.
*
  FORMAT COLOR OFF.  ⟶ Desactivo  todos  los
                     ⟶ Por último  dibujo una línea que ocupa toda la pantalla
  ULINE.
```

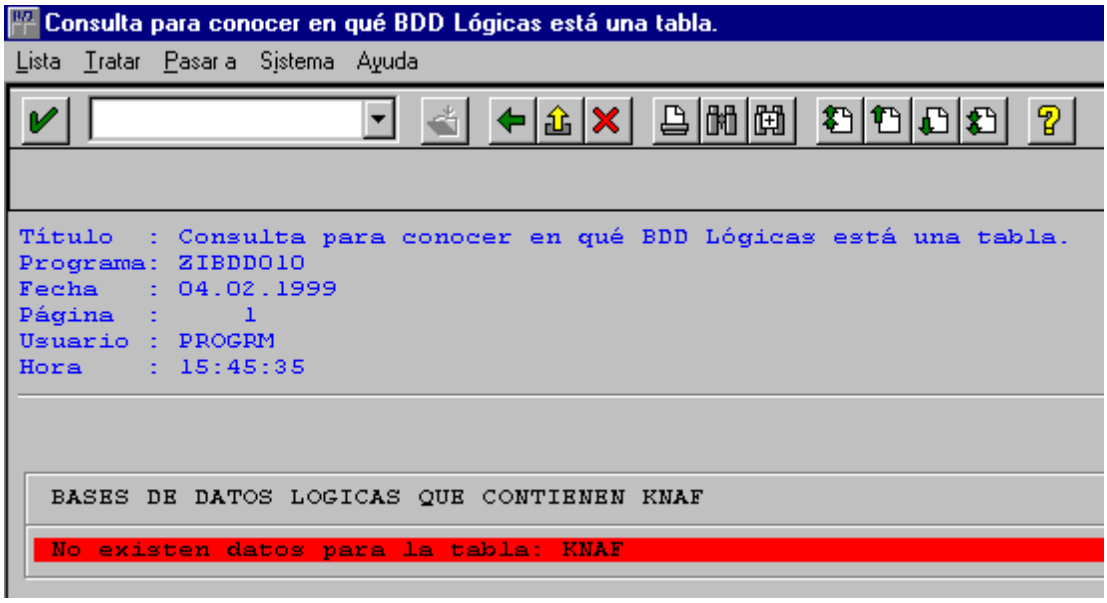
La pantalla que saldría cuando ejecutaríamos el programa sería la siguiente:




Si introducimos la tabla “KNA1” que se que existe, i pulsamos el botón de tomar datos, nos saldrá la siguiente pantalla (que es la misma que antes, pero sin la pantalla de selección y algún botón menos):



Si introducimos una tabla que no existe nos saldrá esta otra pantalla:



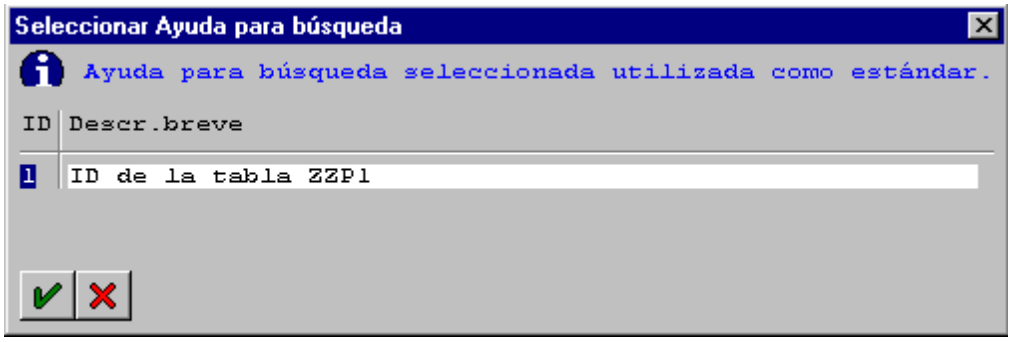

EJEMPLO 2

En este ejemplo asociaremos a un campo que introducimos datos un matchcode ya creado por nosotros. Como ya he explicado como se hacía un matchcode, ahora explicaré como se haría a través de programa, el código del programa sería este:

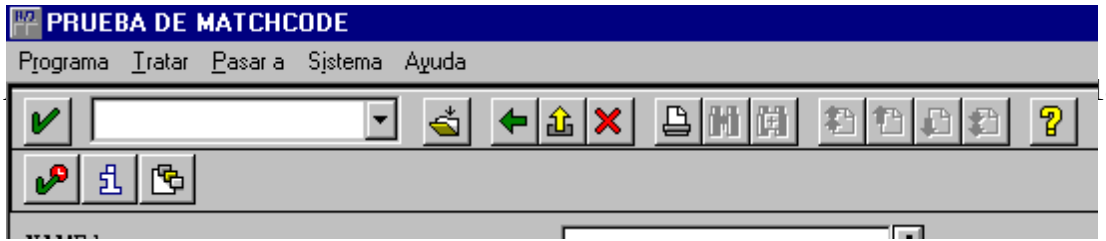
```
REPORT ZZIVAN20.

PARAMETERS NAME1(20)  MATCHCODE OBJECT ZMP1.
```

Nombre del matchcode

Como veis es muy sencillo, pero si lo ejecutaríamos saldría la siguiente pantalla:

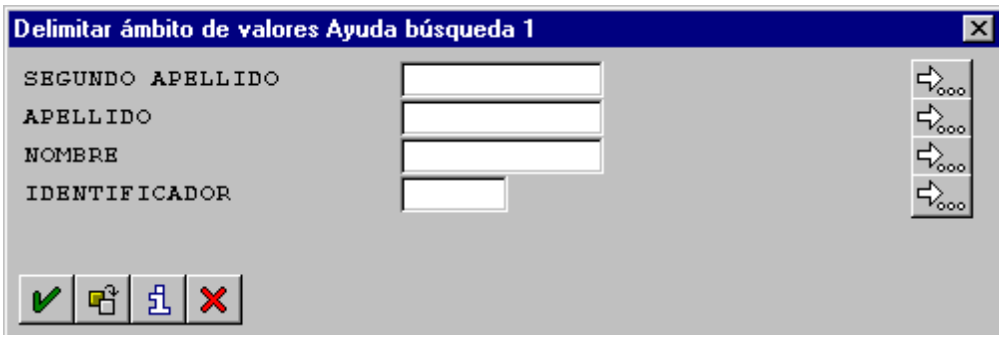



En esta pantalla nos saldría las vistas o IDs que hemos creado, en este caso solo una, cuando seleccionemos la vista o IDs que queramos nos aparecerá el programa, que es este:

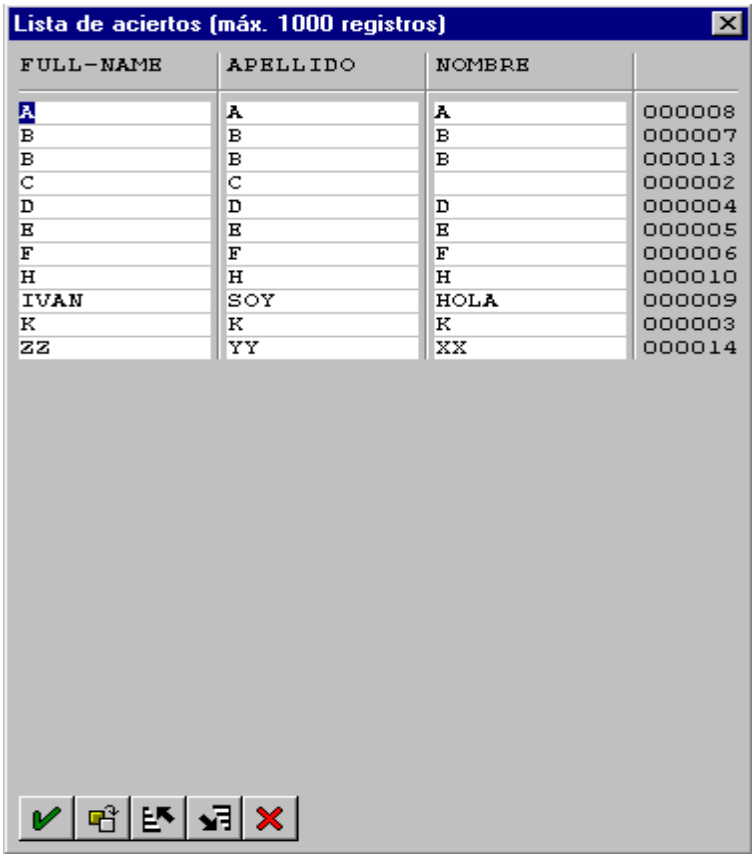


Nuestro
MATCHCODE

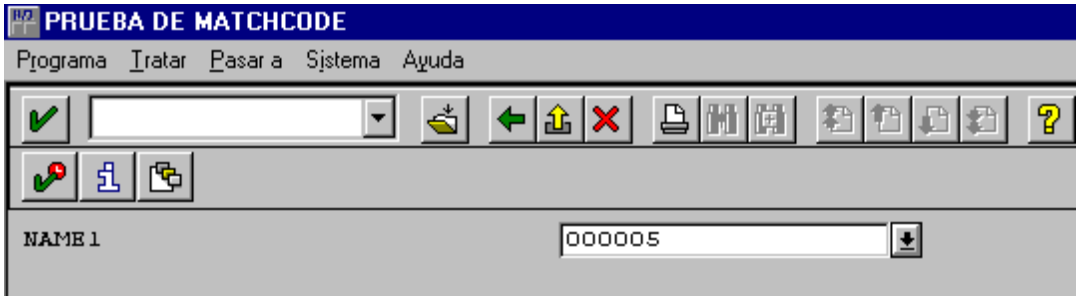
Si seleccionamos el matchcode nos saldrá esta otra pantalla:




Donde podemos indicar los ámbitos de búsqueda, cuando los hayamos puesto o no queramos poner ningún ambito, ejecutaremos el botón de confirmar. Y nos saldrá estra otra pantalla (en la siguiente página):



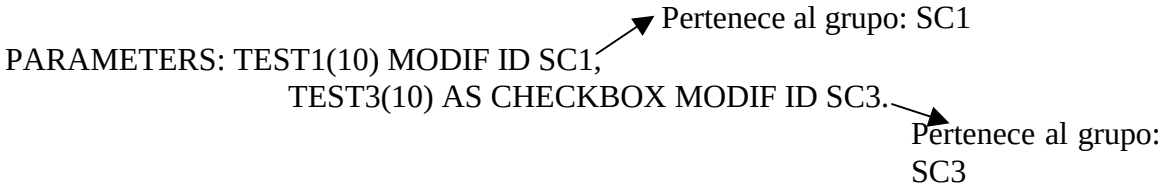

Para seleccionar un registro simplemente hacemos doble clic en un registro. Cuando seleccionemos uno nos guardará el campo clave, en el campo donde hemos hecho el matchcode. Tal que así:




Como ya he explicado cuando haciamos el matchcode, si hay campo que hemos seleccionado en la vista o ID que hemos creado pero ese camopo no esta en la vista standard no guardara el valor del registro seleccionado.

EJEMPLO 3

En este ejemplo vamos a crear un programa que modifique los atributos de la pantalla en tiempo de ejecución, los atributos de pantalla solo se pueden modificar antes de que aparezca la pantalla, para hacerlo utilizaremos el evento AT SELECTION-SCREEN OUTPUT y la propiedad ... MODIF ID ... en la orden “parameters”. El programa en cuestión sería el siguiente:




```
AT SELECTION-SCREEN OUTPUT.

LOOP AT SCREEN.
        IF SCREEN-GROUP1= 'SC1'.
          SCREEN-INTESIFIED = '1'.
          MODIFY SCREEN.
          CONTINUE.
        ENDIF.
        IF SCREEN-GROUP1= 'SC3'.
          SCREEN-INPUT = '1'.
          MODIFY SCREEN.
          CONTINUE.
        ENDIF.
ENDLOOP.
```

El primer "IF" me controla si hay algún objeto que pertenezca al grupo "SC1", si es así le indico que se visualice con un color intensificado.

El segundo me controla si hay algún objeto que pertenezca al grupo "SC2", si es así, le indico que ese objeto es de solo lectura, en este caso es un "checkbox".

Cada vez que cambio la propiedad de algún campo he de realizar la orden "MODIFY SCREEN" para confirmar que realmente quiero modificar la pantalla.