UNIDAD

4

DIPLOMATURA EN PROGRAMACION ABAP

MÓDULO 4: ARQUITECTURA

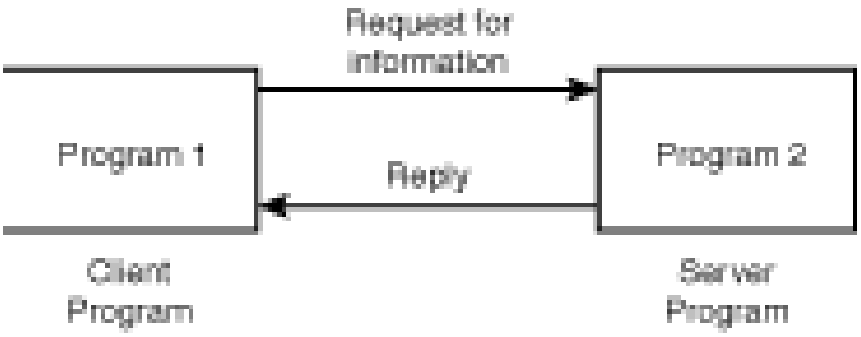
Arquitectura

Arquitectura

Este módulo introduce la arquitectura Cliente/Servidor de SAP y las principales herramientas del Workbench, así como también introduce las nociones básicas de codificación de programas ABAP

Arquitectura Cliente/Servidor

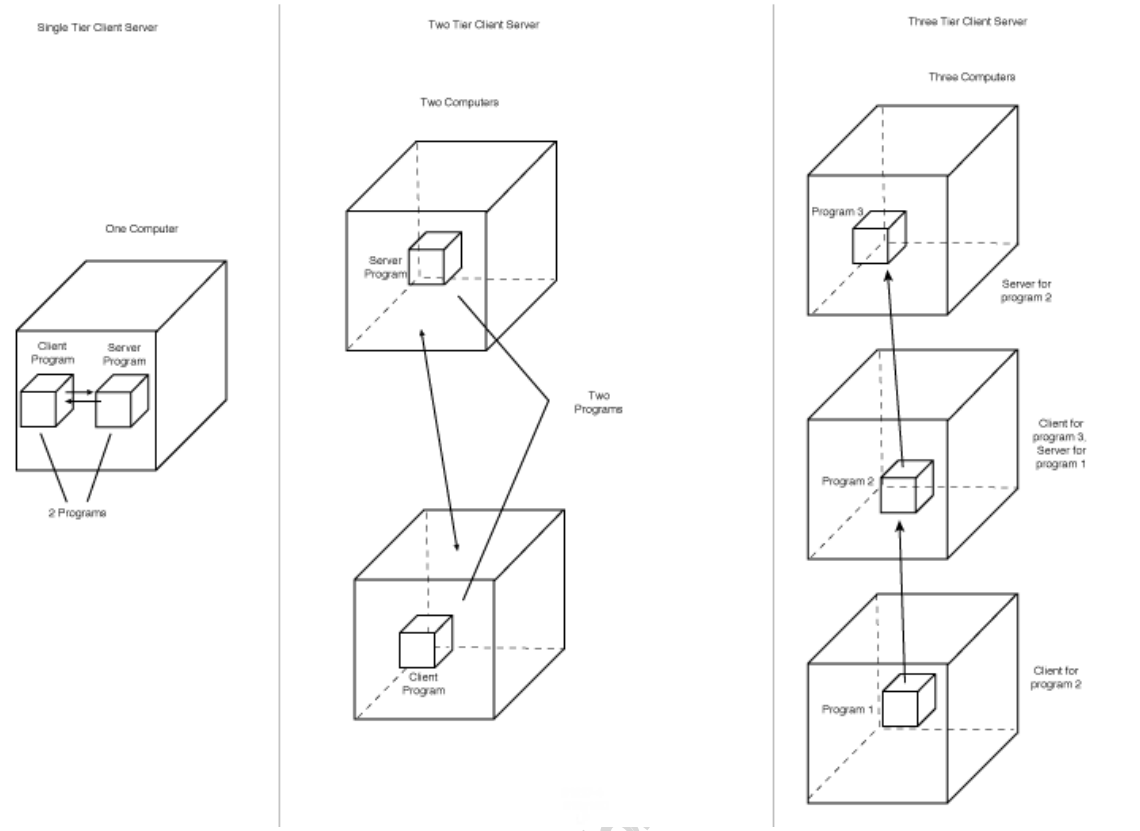
En la arquitectura Cliente / servidor dos programas se hablan el uno al otro:



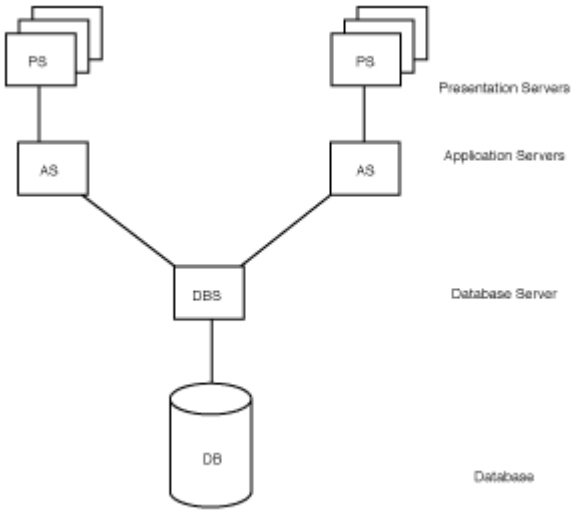
Aquí vemos que el Programa 2 se comunica con el Programa 1 pidiendo alguna información. Programa 1 es el cliente y Programa 2 es el servidor . Programa 2 sirve a Programa 1 con la información solicitada.

Con cliente / servidor, los programas cliente y servidor son procesos independientes. Si el cliente envía una solicitud al servidor, es libre para realizar otras tareas mientras se espera la respuesta del servidor.

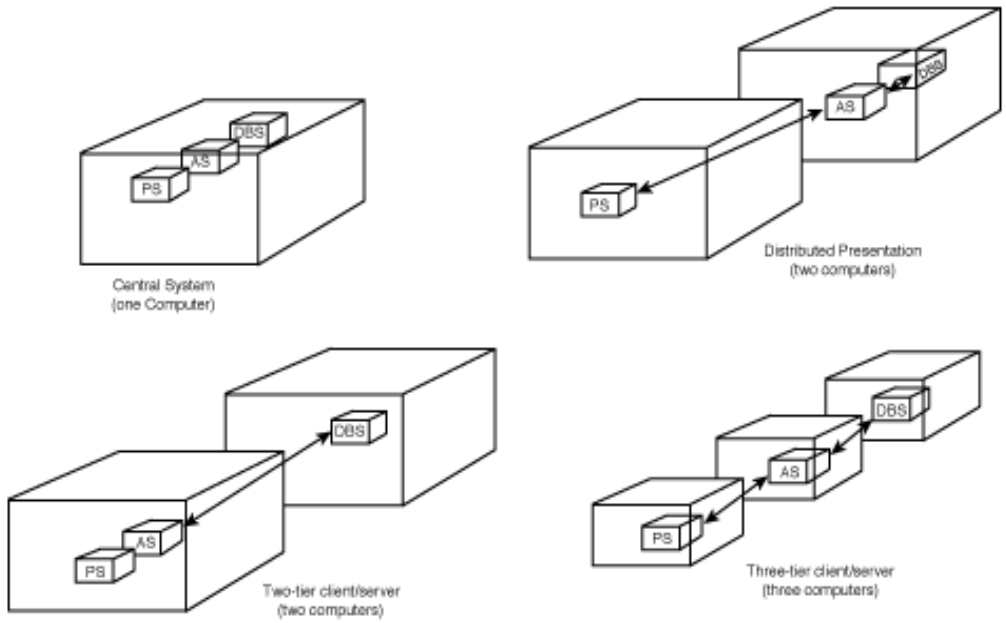
Existen, en general, diferentes configuraciones de esta arquitectura:



ABAP posee en particular una arquitectura "client-server" de 3 capas (además de la base de datos "física", que es la 4ta. capa), lo que se ve en la siguiente figura:



A su vez, existen distintas configuraciones posibles de distribución física de los servidores:



En una configuración de cliente / servidor de tres capas, los servidores de presentación, los servidores de aplicaciones y servidor de base de datos se ejecutan en máquinas separadas. Esta es la configuración más común para grandes sistemas, y es común en producción.

En la configuración de presentación distribuida, los servidores de aplicaciones y bases de datos se combinan en una computadora y los servidores de presentación se ejecutan por separado. Esto se utiliza para los sistemas más pequeños y, a menudo se ve en un entorno de desarrollo.

En la configuración de cliente / servidor de dos capas, los servidores de presentación y aplicación se combinan y el servidor de base de datos es independiente. Esta configuración se utiliza en conjunción con otros servidores de aplicaciones. Se utiliza para un servidor lote (trabajo "Batch" u "offline") cuando se mantiene separado de los servidores en línea. Un SAPGUI (interfaz gráfica de usuario) se instala en ella para proporcionar control local.

Un sistema central es cuando todos los servidores se combinan en una sola máquina, que tiene un centro de configuración. Esto raramente se ve porque describe un sistema independiente de R / 3 con sólo un único usuario (por ejemplo, el minisap que nosotros instalamos en nuestras PCs).

En cada una de las capas hay entonces al menos un servidor: de base de datos, de aplicaciòn y /o de presentación. Empecemos a ver cada uno de ellos en detalle.

Servidor de Base de Datos

El servidor de base de datos es un conjunto de archivos ejecutables que aceptan peticiones de base de datos desde el servidor de aplicaciones. Estas peticiones se transmiten a la RDBMS (sistema de gestión de base de datos relacional, también conocido como el "motor de la base de datos"). El RDBMS envía los datos al servidor de base de datos, que luego pasa la información al servidor de aplicaciones. El servidor de aplicaciones a su vez pasa esa información a su programa ABAP / 4.

Por lo general, existe un equipo físico independiente dedicado a albergar el servidor de base de datos y el RDBMS (por ejemplo, Oracle, Informix, etc.) puede funcionar en ese equipo también, o se puede instalar en su propio ordenador, en forma separada; lo mismo puede pasar con los otros servidores, dando lugar a distintas distribuciones, que se definen al momento de instalar SAP R/3:

Servidor de Aplicaciones

Un servidor de aplicaciones es un conjunto de archivos ejecutables que interpretan colectivamente los programas ABAP / 4 y gestionan la entrada y la salida para ellos. Cuando se inicia un servidor de aplicaciones, estos ejecutables empiezan todos al mismo tiempo. Cuando se detiene un servidor de aplicaciones, todos se cierran juntos. El número de procesos que se inician cuando se trae a colación el servidor de aplicaciones se define en un archivo de configuración llamado el perfil de servidor de aplicaciones .

Cada servidor de aplicación tiene un perfil que especifica sus características cuando se inicia y mientras se está ejecutando. Por ejemplo, un perfil de servidor de aplicaciones especifica:

- Número de procesos y sus tipos
- La cantidad de memoria de cada proceso puede utilizar
- Longitud de tiempo que un usuario está inactivo antes de ser desconectado automáticamente

El servidor de aplicaciones existe para interpretar programas ABAP / 4, y los mismos sólo se ejecutan allí, los programas no se ejecutan en el servidor de presentación. Un programa ABAP / 4 puede iniciar un archivo ejecutable en el servidor de presentación, pero un programa ABAP / 4 no puede ejecutarse allí.

Si su programa ABAP / 4 solicita información de la base de datos, el servidor de aplicaciones formatea la solicitud para enviarla al servidor de base de datos.

Presentación al Usuario con el Servidor de Aplicaciones

El *servidor de presentación* es en realidad un programa llamado *sapgui.exe* . Se suele instalar en la estación de trabajo de un usuario. Para iniciarlo, el usuario hace doble clic en un icono en el escritorio (el de SAP R/3!) o elige una ruta de menú. Cuando se inicia, el servidor de presentación muestra los menús de SAP R / 3 dentro de una ventana. Esta ventana se conoce comúnmente como el SAP GUI, o la interfaz de usuario (o simplemente, la interfaz). La interfaz acepta la entrada del usuario en forma de pulsaciones de teclado, clics del mouseo teclas de función, y envía estas solicitudes al servidor de aplicaciones para su procesamiento. El servidor de aplicaciones envía los resultados de vuelta a la SAPGUI que da formato a la salida de pantalla al usuario.

Herramientas - WorkBench

Un objeto de desarrollo es algo creado por un desarrollador. Ejemplos de objetos de desarrollo son los programas, pantallas, tablas, vistas, estructuras, modelos de datos, mensajes, etc.

El sistema R / 3 contiene herramientas para crear y probar objetos de desarrollo. Estas herramientas se encuentran en el Workbench (banco de trabajo) de Desarrollo de R / 3. Para acceder a cualquier herramienta de desarrollo, hay que acceder al banco de trabajo. El banco de trabajo contiene estas herramientas para ayudarle a crear objetos de desarrollo:

- El editor de programas ABAP / 4, donde se puede crear y modificar código fuente ABAP / 4 y otros componentes del programa
- El diccionario de datos, donde se pueden crear tablas, estructuras y vistas
- El modelador de datos, donde se puede documentar las relaciones entre tablas
- La Biblioteca de funciones, con la que se pueden crear módulos de función globales ABAP / 4
- El diseñador de pantallas y el diseñador de menús, de donde se puede crear una interfaz de usuario para sus programas

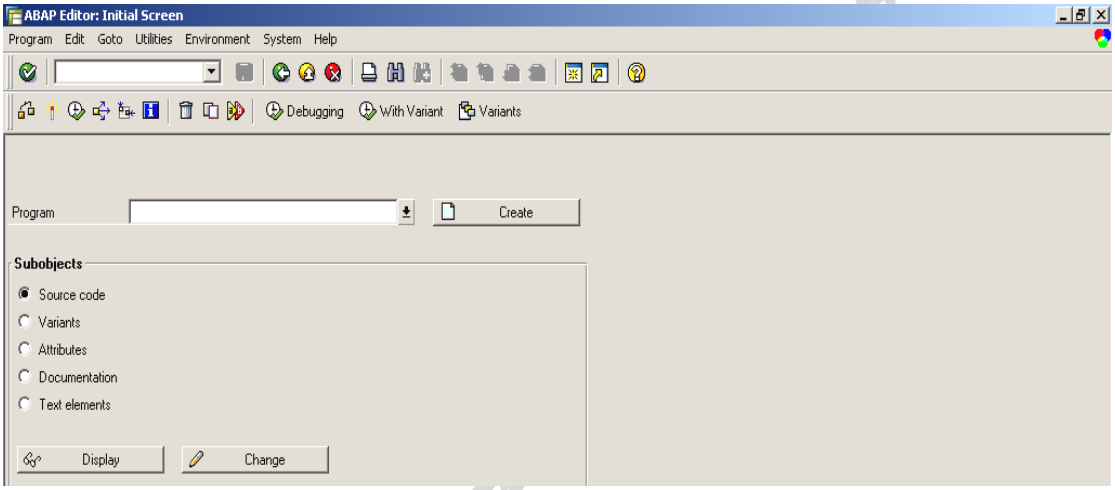
Las siguientes herramientas de prueba y de búsqueda también están disponibles:

- el depurador ABAP / 4
- la herramienta de seguimiento de SQL se utiliza para ajustar sentencias de SQL
- el analizador de tiempo de ejecución para optimizar el rendimiento de su programa
- una herramienta de referencia de utilización para el análisis de impacto de los programas en el sistema
- una herramienta de prueba automatizada para pruebas de regresión
- una herramienta de búsqueda para encontrar objetos en el repositorio de objetos de desarrollo
- el Organizador del Workbench, para registrar cambios en los objetos y promoción de los mismos a la producción

Todos los objetos de desarrollo son portátiles, lo que significa que usted puede copiarlos de un sistema R / 3 a otro (incluso con distintas plataformas y sistemas operativos). Esto se hace normalmente para mover los objetos de desarrollo del sistema de desarrollo al sistema de producción. Si los sistemas de origen y de destino están en diferentes sistemas operativos o utilizan diferentes sistemas de bases de datos, los objetos de desarrollo se ejecutarán tal cual y sin ninguna modificación. Esto es cierto para todas las plataformas soportadas por R / 3 (que son todas las que están estandarizadas por el mercado, sea Windows, Linux, Unix, etc. con Microsoft, Sun, Macintosh, etc.)

Editor ABAP

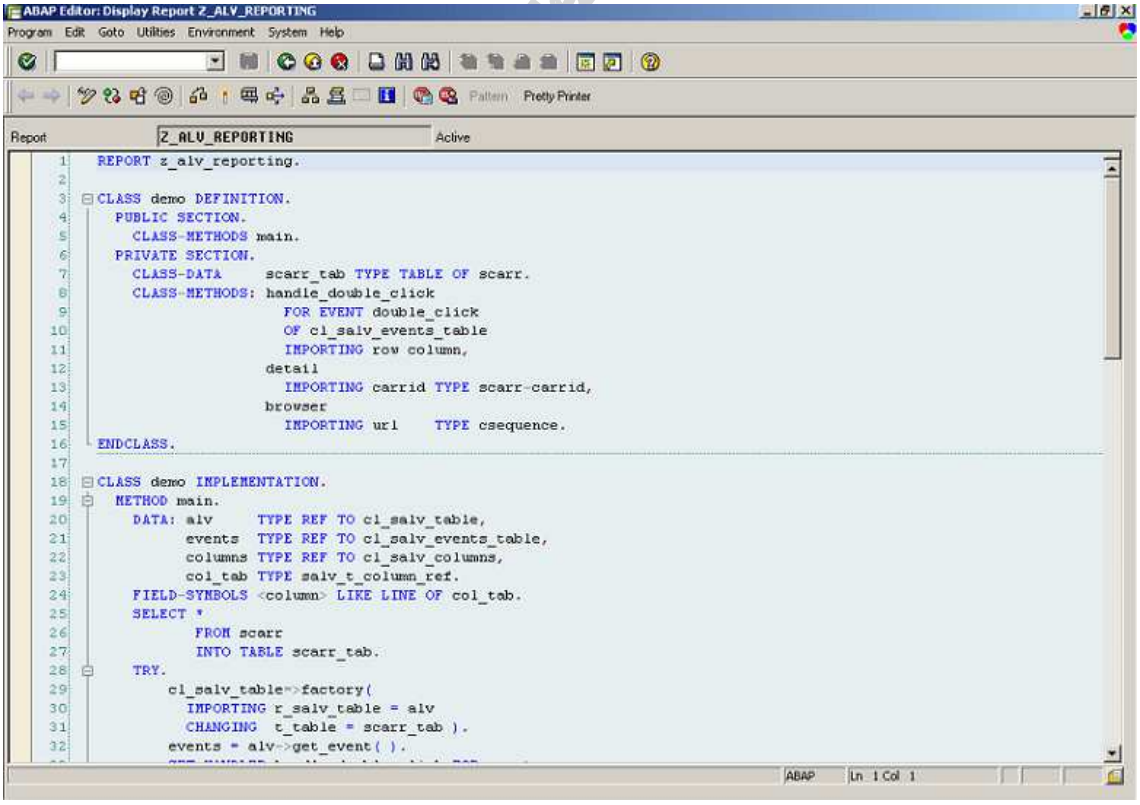
Se ingresa por el Workbench o por la transacción SE38, y se accede a la pantalla principal del editor de programas ABAP:



Si escribo por ejemplo Z* y luego toco DISPLAY, veremos que tenemos el programa y la descripción del programa (para todos los programas que empiezan con Z):

Program Name	Report title
Z_ABAP_UNIT	Program Z_ABAP_UNIT
Z_ABSTRACT_CLASS	Report Z_ABSTRACT_CLASS
Z_AIRLINES	Program Z_AIRLINES
Z_ALV_PRINTING	Program Z_ALV_PRINTING
Z_ALV_REPORTING	Program Z_ALV_REPORTING
Z_ASSIGN_CASTING	Program Z_ASSIGN_CASTING
Z_ASSIGN_CASTING_TYPE	Program Z_ASSIGN_CASTING_TYPE
Z_ASSIGN_COMPONENT	Report Z_ASSIGN_COMPONENT
Z_ASSIGN_DYNAMIC_ACCESS	Report Z_ASSIGN_DYNAMIC_ACCESS
Z_ASSIGN_INCREMENT	Program Z_ASSIGN_INCREMENT
Z_ASSIGN_STATIC	Report Z_ASSIGN_STATIC
Z_ASXML_ELEMENTARY	Program Z_ASXML_ELEMENTARY
Z_ASXML_OBJECT	Program Z_ASXML_OBJECT
Z_ASXML_STRUCTURE	Program Z_ASXML_STRUCTURE
Z_ASXML_TABLE	Program Z_ASXML_TABLE
Z_AUTHORITY_CHECK	Program Z_AUTHORITY_CHECK
Z_BANKING	Report Z_BANKING
Z_BRANCHING	Program Z_BRANCHING
Z_CALL_FUNCTION	Program Z_CALL_FUNCTION
Z_CALL_VERY_SIMPLE_TRANS	Report Z_CALL_VERY_SIMPLE_TRANS
Z_CASE	Report Z_CASE
Z_CHECK	Program Z_CHECK
Z_CLASSIC_PRINTING	Program Z_CLASSIC_PRINTING
Z_CLASSIC_REPORTING	Program Z_CLASSIC_REPORTING
Z_CODE_INSPECTOR	Program Z_CODE_INSPECTOR

Selecciono uno cualquiera y presiono el tilde verde, y aparece entonces el código fuente del programa:



Barra de tareas

La barra de tareas del Editor incluye botones para switchear entre modo display y edición (el lápiz), para chequera sintaxis (los 2 cuadrados), para activar (el "fosforito", el concepto de activación se explicará más adelante) y para ejecutar (el "torniquete", al lado del "fosforito"). A su vez, se puede otorgar formato y sangría automáticas al código con "Pretty Printer" e insertar comandos con sus modificadores y su sintaxis completa con "Pattern":

Editor de Texto Fuente ABAP

Para editar el código fuente de un programa (por ahora, preescrito como el que se ve en la figura anterior), simplemente switchear de modo display a modo ediciòn y escribir.

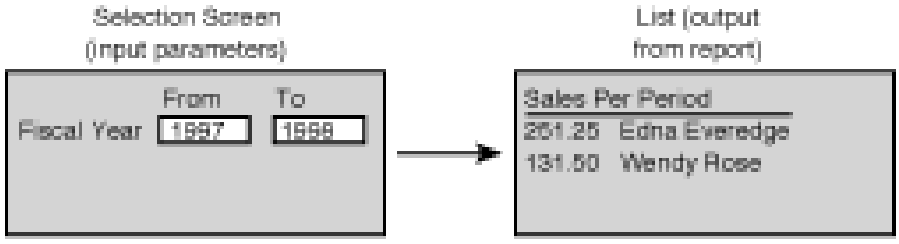
El Editor ABAP tiene las mismas funcionalidades que los demás editores de texto estándar, como copiar, pegar, borrar, etc. Aparecerá una barra contextual con todos estos botones para trabajar con la edición, al activar la misma. Por ejemplo (la figura puede variar ligeramente, de acuerdo con la versión instalada en su PC):

Introducción al Lenguaje

Hay dos tipos principales de programas ABAP / 4:

- informes
- programas de diálogo

El propósito de un informe es leer los datos de la base de datos y escribirlo. Se compone de dos pantallas:



La primera pantalla se denomina pantalla de selección . Contiene los campos de entrada que permiten al usuario introducir criterios para el informe. Por ejemplo, el informe puede elaborar una lista de las ventas para un rango de fechas determinado, por lo que los campos de entrada de rango de fechas aparecerían en la pantalla de selección del informe.

La segunda pantalla es la pantalla de salida. Contiene la lista obtenida . La lista es el resultado del informe, y por lo general no tiene ningún campo de entrada. En nuestro ejemplo, contendría una lista de las ventas que se produjeron dentro del intervalo de fechas especificado.

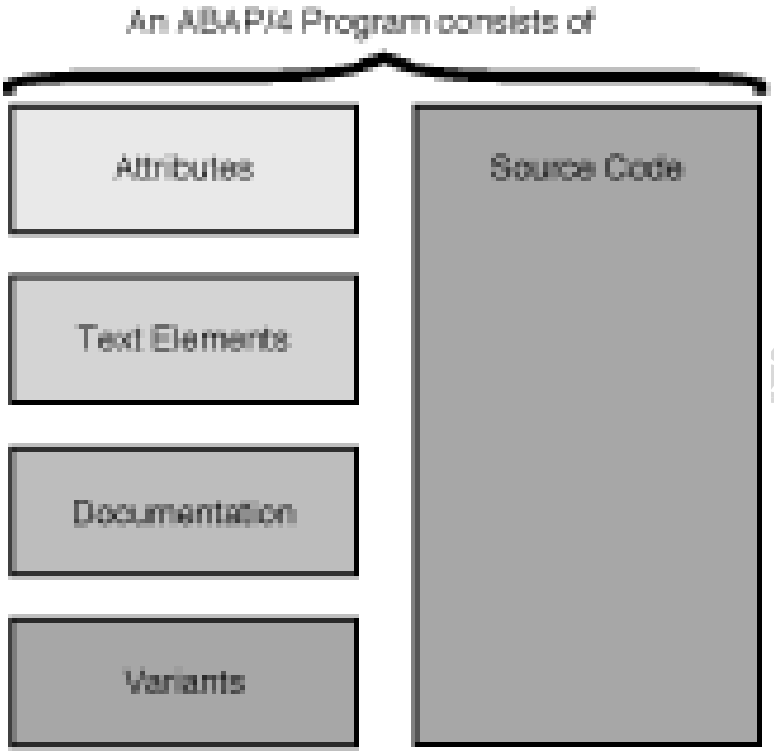
La pantalla de selección es opcional. No todos los informes tienen una. Sin embargo, todos los informes generan una lista.

Los programas de diálogo son más flexibles que los informes, por lo que son más complejos a nivel de programa. Pueden contener cualquier número de pantallas, y la secuencia de la pantalla se puede cambiar de forma dinámica en tiempo de ejecución. En cada pantalla puede haber campos de entrada, campos de salida, botones, y un área de desplazamiento, por ejemplo, entre otros.

Estructura de un Programa ABAP

Los informes ABAP / 4 constan de cinco componentes:

- El código fuente
- Atributos
- Los elementos de texto
- Documentación
- Variantes



Sólo son obligatorios el código fuente y los componentes de los atributos del programa. El resto de los componentes son opcionales.

Todos los objetos de desarrollo y sus componentes se almacenan en la base de datos de R / 3. Por ejemplo, el código fuente de un informe se almacena en la tabla de base de datos dd010s .

Los programas ABAP / 4se interpretan, sino que no se compilan. La primera vez que se ejecuta un programa, el sistema genera automáticamente un objeto en tiempo de ejecución . El objeto de tiempo de ejecución es una forma preprocesada del código fuente. Sin embargo, no es un archivo ejecutable que se puede ejecutar en el nivel de sistema operativo. En lugar de ello, se requiere que el sistema R / 3 opere para interpretarlo. El objeto de tiempo de ejecución también se conoce como el formulario generado del programa. Si cambia el código fuente, el objeto de tiempo de ejecución se regenera automáticamente la próxima vez que se ejecute el programa.

Existe una convención de nomenclaturas de programas.La compañía para la que se trabaja es un cliente de SAP. Por lo tanto, los programas que se crean en su empresa se llaman los programas clientes . Los objetos de desarrollo de los clientes deben seguir las convenciones de nomenclatura que están predefinidas por SAP. Estos convenios se denominan rango de nombres del cliente . Para programas, el rango de nombres de cliente es de dos a ocho caracteres y el nombre del programa debe comenzar con la letra y o z (mayúscula o minúscula) . SAP se reserva las letras *a* hasta *x* para sus propios programas.

Por ejemplo, es posible utilizar la letra z seguido de sus dos iniciales de nombre y apellido.

Creación de un Programa ABAP

Lo que sigue es una descripción del proceso que seguirá para crear un programa.

Cuando se conecte a R / 3 para crear su primer programa ABAP / 4, la primera pantalla que verá será el menú principal de SAP. A partir de ahí, usted irá al banco de trabajo, y luego al editor. Se entra con un nombre de programa, y se crea así el programa. La primera pantalla que verá será la de atributos. Debe introducir los atributos del programa y salvarlos, como se indicará enseguida. A continuación se le permitirá continuar con el editor de código fuente. En el editor de código fuente, deberá introducir el código fuente, guardarlo, y luego ejecutar el programa.

Siga este procedimiento para crear su primer programa, paso a paso (puede variar ligeramente la descripción, de acuerdo con la versión de minisap que Usted tenga instalada):

- En el menú principal de R / 3, seleccione el menú de ruta Herramientas-> ABAP / 4 Workbench. Se muestra una pantalla con el título de ABAP / 4 Workbench de Desarrollo.
- Pulse el botón de Editor ABAP / 4 en la barra de herramientas de la aplicación. Se muestra la pantalla inicial del Editor de ABAP / 4.
- En el campo Programa, introduzca el nombre del programa por ejemplo Z01 .
- Pulse el botón Crear. Aparece la pantalla Atributos del Programa ABAP / 4. Los campos que contienen signos de interrogación (o "checkmark") son obligatorios.
- Escriba *Mi primer programa ABAP / 4* en el campo *Título*, o elija un título cualquiera. Por defecto, aparecerá el contenido de este campo en la parte superior de la lista obtenida al final.
- Escriba 1 en el campo Tipo. Un 1 indica que el programa es un informe.
- Escriba un asterisco (*) en el campo de aplicación. El valor en el campo de aplicación indica a qué zona pertenece la aplicación de este programa. La lista completa de los valores se puede conseguir colocando el cursor en este campo para luego hacer clic en la flecha hacia abajo a la derecha de la misma. Por ejemplo, si este programa pertenece a la gestión de inventario, usted pondría una L en el campo de aplicación. Dado que este es un programa simple de prueba, he utilizado un asterisco para indicar que este programa no pertenece a ningún área de aplicación en particular.
- Para guardar los atributos, pulse el botón Guardar de la barra de herramientas estándar. Se muestra la pantalla de entrada del catálogo de objetos Crear.
- Pulse el botón "Objeto Local". La pantalla vuelve a mostrar los atributos del programa. En la barra de estado en la parte inferior de la pantalla, aparece el mensaje "Atributos de programa guardados" .
- Pulse el botón de código fuente en la barra de herramientas de la aplicación del Editor ABAP / 4: se visualiza la pantalla de edición de programas.
- Elija la ruta Configuración-> Modo Editor. Se muestra la pantalla de configuración.
- Elija el botón de radio Modo PC con la numeración de líneas.
- Seleccione el botón Minúsculas.

- Pulse el botón Copiar (la marca de verificación verde). Ahora ha guardado los ajustes del editor. (sólo es necesario establecerlos la primera vez.)
- En la linea 1 aparecerá el nombre del reporte.
- En la línea 2, escriba: write 'Hola mundo SAP'. Utilice comillas simples y ponga un punto al final de la línea. Para ABAP, las sentencias terminan siempre con un punto, ocupen o no varias líneas:

```
1 report z01.

2 write 'Hola mundo SAP'.
```

- Pulse el botón Guardar en la barra de herramientas estándar.
- Para ejecutar el programa, seleccionar la ruta del menú Programa-> Ejecutar. Aparece una pantalla con el título **Mi primer programa ABAP / 4**, y las palabras **Hola mundo SAP** se escriben debajo de ella. Esta es la salida del informe, también conocida como la lista.

Mi primer programa ABAP / 4

Hola mundo SAP

Para volver al editor, pulse el botón de la flecha verde en la barra de herramientas estándar (o la tecla F3).

Estos son los problemas comunes encontrados mientras se crea un programa y sus soluciones

Problema	Solución
Cuando se pulsa el botón Crear, aparece un cuadro de diálogo que dice no crear objetos en el Nombre Rango SAP.	Ha introducido el nombre del programa equivocado. Sus nombres de los programas deben comenzar con y o z . Pulse el botón Cancelar (la X roja) para volver e introduzca un nuevo nombre del programa.
Cuando se pulsa el botón Crear, aparece un cuadro de diálogo con un	Ha introducido el nombre del programa equivocado. Sus nombres de los programas

campo de entrada pidiendo una clave.	deben comenzar con y o z . Pulse el botón Cancelar (la X roja) para volver e introduzca un nuevo nombre del programa.
Usted está recibiendo una pantalla de consulta de solicitud de cambio pidiendo un número de solicitud.	En la pantalla de entrada de catálogo Crear objeto, no introduzca un valor en el campo Clase de desarrollo. Pulse el botón Objeto Local en su lugar.

Resumen de Sentencias ABAP

Hasta ahora hemos utilizado una sola sentencia, *write*, que nos ha servido para mostrar por pantalla en la salida el mensaje que deseamos exhibir (en el ejemplo anterior, 'Hola mundo SAP'). A medida que avancemos en el curso vamos a ir viendo muchos comandos o sentencias más, y en el campus se subirá oportunamente un glosario completo de sentencias para consultar.

Recordar, asimismo, que en el Editor ABAP, con el botón Pattern, se puede ir "construyendo" la sentencia correspondiente con todos sus modificadores (esto se verá en detalle más adelante).

Diccionario de Datos ABAP

El Diccionario de datos R / 3 (o DDIC para abreviar) es una utilidad para la definición de objetos de datos. Usando el DDIC, puede crear y almacenar objetos como tablas, estructuras y vistas. Para invocar el diccionario de datos, ingrese la transacción /nse11 en la ventana de comando.

El DDIC está dentro del sistema R / 3. Usted puede pensar en él como si se sentara encima de una base de datos física como Oracle o Informix y actuara como un control remoto, para generar y enviar instrucciones SQL a la misma. Por ejemplo, puede crear una definición de tabla en el DDIC. Cuando se activa la definición de la tabla, las sentencias SQL se generan y se envían a la RDBMS, haciendo que se cree la tabla real en la base de datos. Cuando desee modificar la tabla, se debe modificar la definición de la tabla en el DDIC. Cuando se activa la tabla otra vez, se genera más código SQL haciendo que el RDBMS pueda modificar la tabla.

NO modifique una tabla o cualquier otra cosa a nivel de RDBMS. La definición de diccionario de datos no se puede luego actualizar por sí misma y estará fuera de sincronización con la base de datos. Esto puede provocar errores de aplicación e incluso conducir a una pérdida de la integridad de los datos. Acceda a la base de datos a través del diccionario (que trabaja, justamente, como aplicativo de "servidor de base de datos").

El diccionario de datos es una fuente de información centralizada.

Los distintos objetos del Diccionario de datos están estructurados en :

Tabla > Campo > Elemento de datos > Dominio

Los elementos de datos describen el significado de un campo independientemente de lastablas donde se utilicen. Es decir, tienen un carácter semántico.

Los dominios describen el campo de valores posibles. Tendrán un carácter técnico.

Ejemplo :

TABLAS : SKB1,SKM1…

CAMPO: STEXT

ELEM. DATOS: STEXT_SKB1

DOMINIO : TEXT50

FORMATO INTERNO : Tipo C de 50 Posiciones

Existen diversos tipos de tablas:

- Tablas TRANSP (transparentes) : Tablas normales relacionales (SQL).

- Tablas POOL : Tablas SAP que se guardan junto a otras tablas SAP en una única tabla física de BDD. Mejorando el acceso a los registros.

- Tablas CLUSTER : Varias tablas que se guardan en un cluster de BDD. Se guardan registros de varias tablas SAP con la misma clave cluster, en el mismo cluster físico de la base de datos.

El diccionario de datos se dice que es integrado y activo. Integrado porque integra el D.D.

con el Screen-Painter, Programas ABAP, Dynpros, etc. y activo porque si modificamos algún objeto del diccionario de datos, el sistema automáticamente regenera el 'Time Stamp' de los programas que utilicen esos objetos.

Podemos clasificar los datos del sistema SAP en datos maestros, datos de movimientos, y datos del sistema.

• Datos maestros : Son datos que no se modifican muy a menudo.

Ej: Materiales, Cuentas, Bancos, Clientes ...

Se almacenarán en tablas transparentes.

• Datos de movimientos : Datos muy volátiles y con gran volumen de generación.

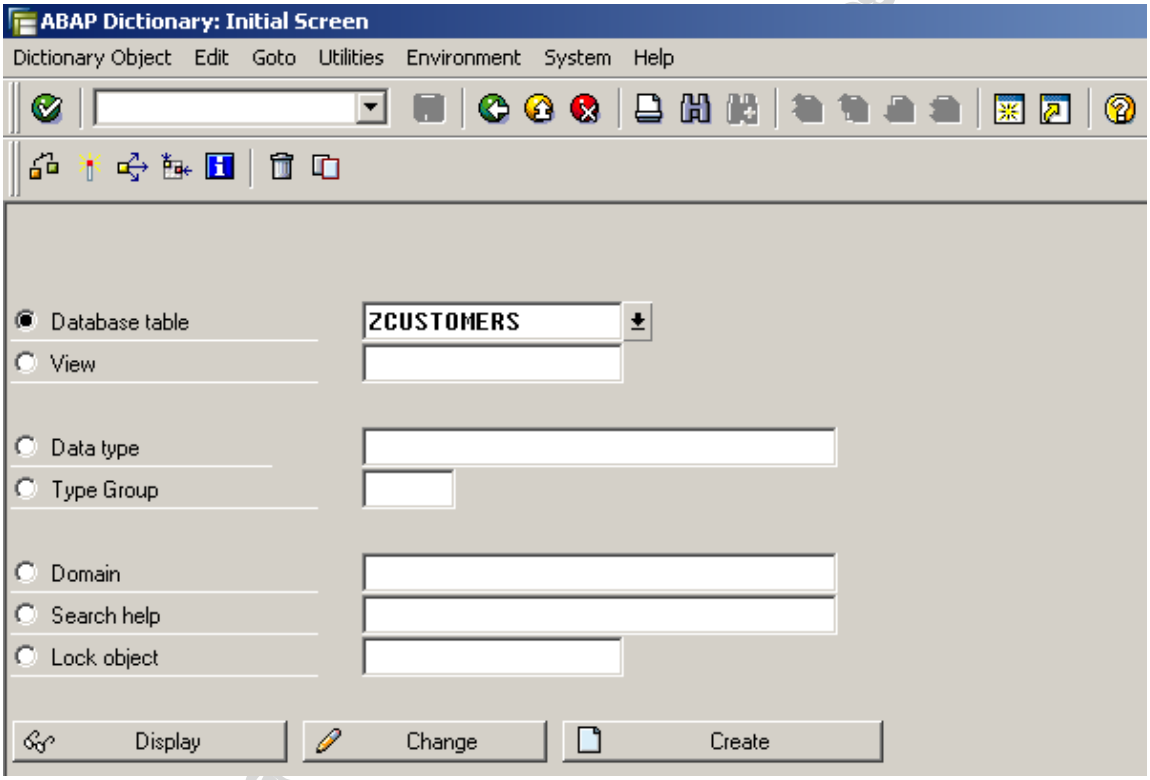
Ej: Facturas, Pedidos...

Se suelen guardar en tablas tipo CLUSTER todos ellos con formato parecido (documentos).

• Datos del sistema o de control : Muchas tablas con pocos datos. Se suelen guardar en tablas de tipo POOL.

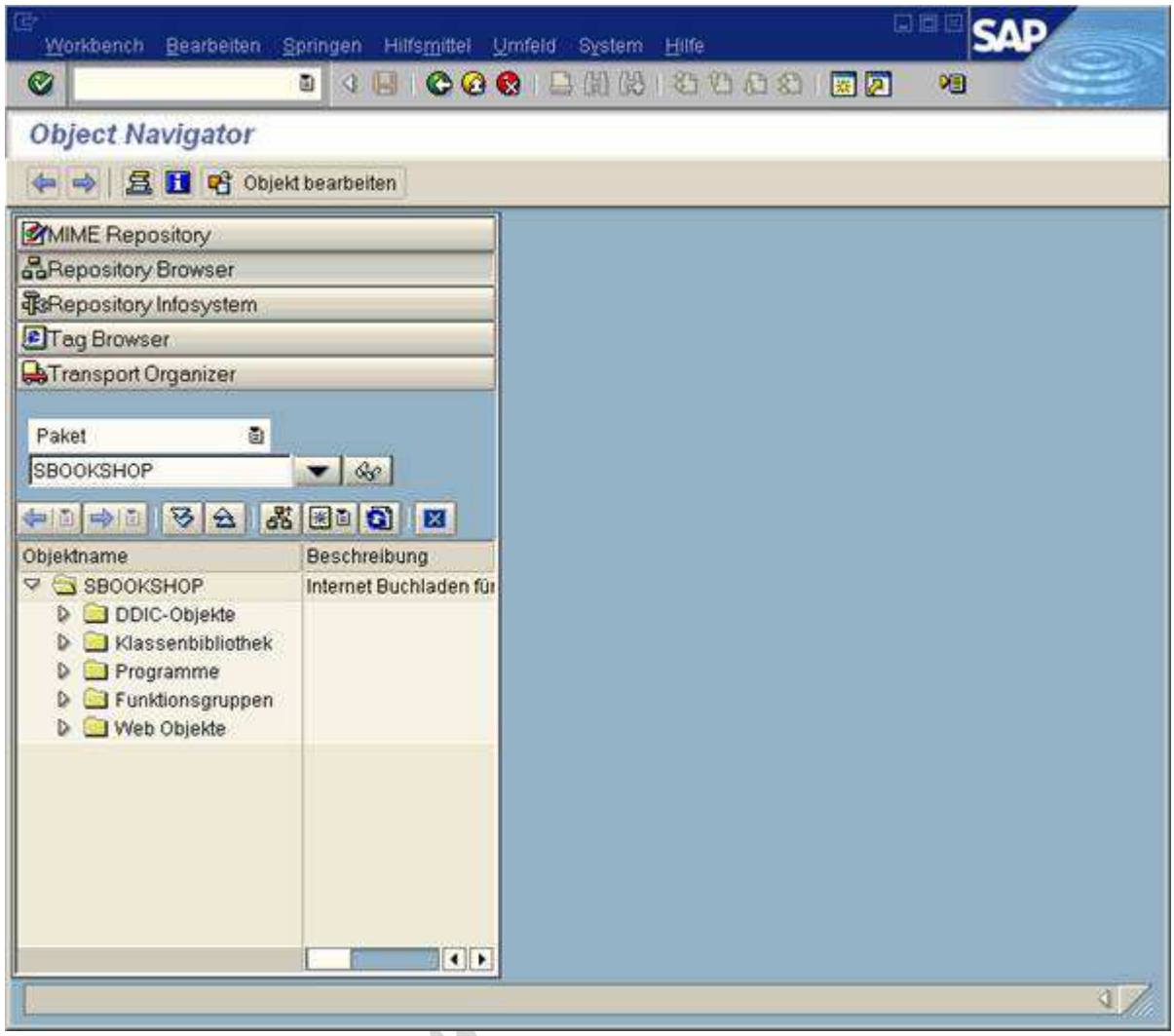
Botones de Acción

Al igual que con el Editor, existe una barra de botones de aplicación en el diccionario, lo cual se muestra en la siguiente figura (puede variar de acuerdo a la versión que tenga instalada), donde se ve la pantalla inicial del diccionario:



Navegación de Objetos SAP

Los objetos de SAP pueden verse con el Objetc Browser, al cual se accede ingresando la transacción /nse80 en la ventana de comandos. En la pantalla principal del mismo, se puede elegir el tipo de objetos a buscar (por ejemplo Locales, o por ejemplo Programas) y luego filtrarlos (por ejemplo, con Z*), y como resultado obtendremos la lista de objetos que cumplen con lo pedido:



Visualización de Objetos del Estándar

Se puede utilizar la misma transacción anterior. Una vez hallado el objeto, se hace doble clic sobre el mismo y se logra así visualizar su contenido.

Actualización de Objetos Cliente

Los objetos creados por el cliente (Z*) pueden ser modificados por su creador y por quienes tengan permiso para acceder a los mismos. Si son locales se modifican directamente, si no, además, hay que transportarlos luego.
Los objetos de SAP no son modificables salvo por consultores autorizados por SAP.

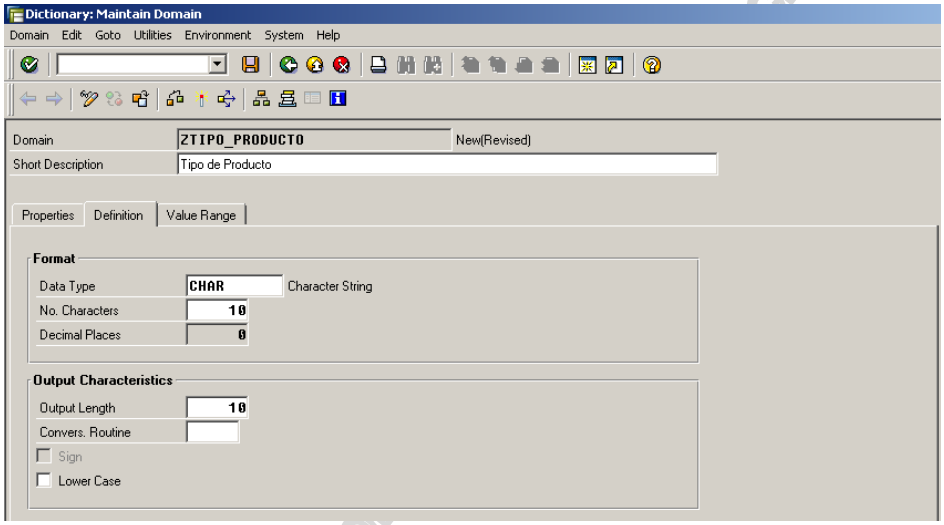
Dominios

Las tablas de la base de datos están formadas por campos. Las tablas de SAP (que llamamos tablas transparentes) tambien están formadas por campos.

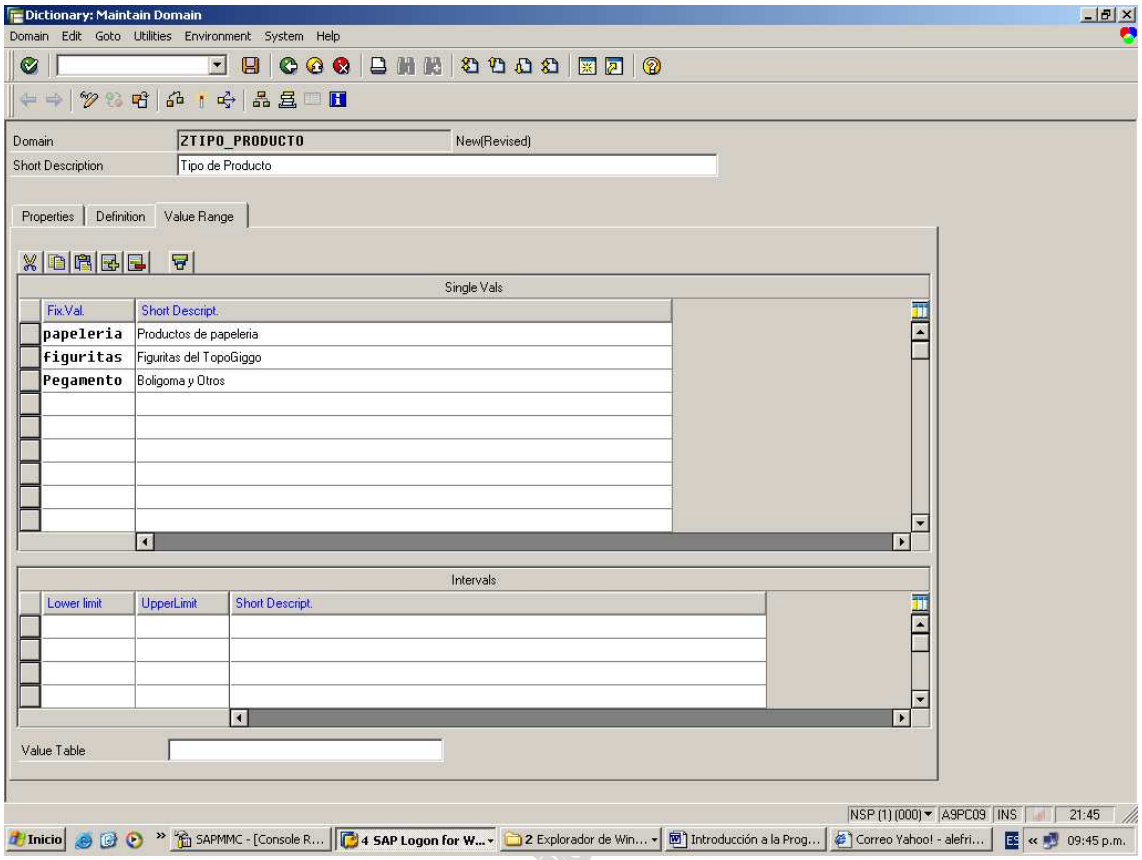
Los campos en la base de datos, están formados basicamente por un nombre, un tipo de datos(char,fecha,numero.....) y unos datos especificativos (longitud, decimales). En SAP, estos datos están agrupados en unas entidades llamadas Elementos de Datos,
Los Elementos de Datos de SAP, contienen una parte semántica (descripciones y ayudas), y una parte técnica que es lo que está mas cerca de la tabla de base de datos, es decir el tipo del campo de la tabla de la base de datos, su longitud, decimales, etc. Esta parte técnica se agrupa en unas entidades llamadas Dominios. Así, podemos crear un dominio como un objeto más de SAP, y lo podemos llamar, por ejemplo, Zdom01. A este dominio ejemplo le podemos asociar el tipo CHAR de 10 caracteres, de modo tal que todos los campos que tengan esta estructura van a pertenecer al mismo dominio y por tanto heredarán sus condiciones (que luego explicaremos en detalle cuàles son).

Los pasos para crear un dominio podrían ser estos:

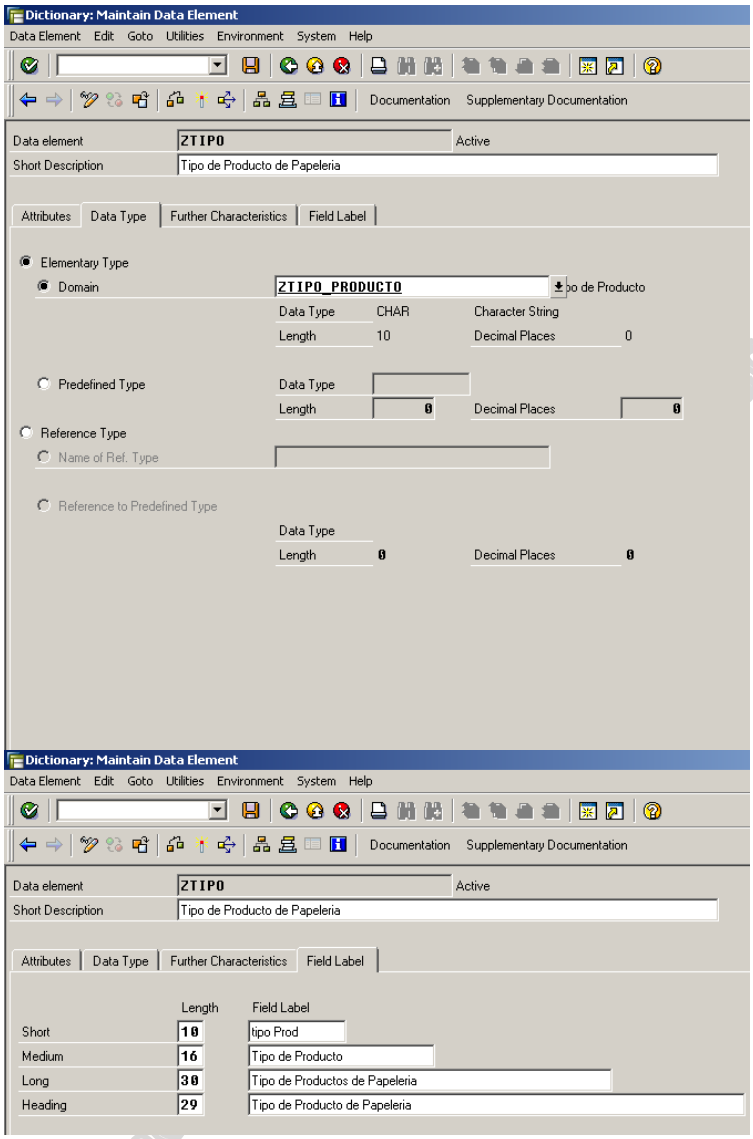
- Ir a la transacciòn SE11 y elegir crear un dominio:



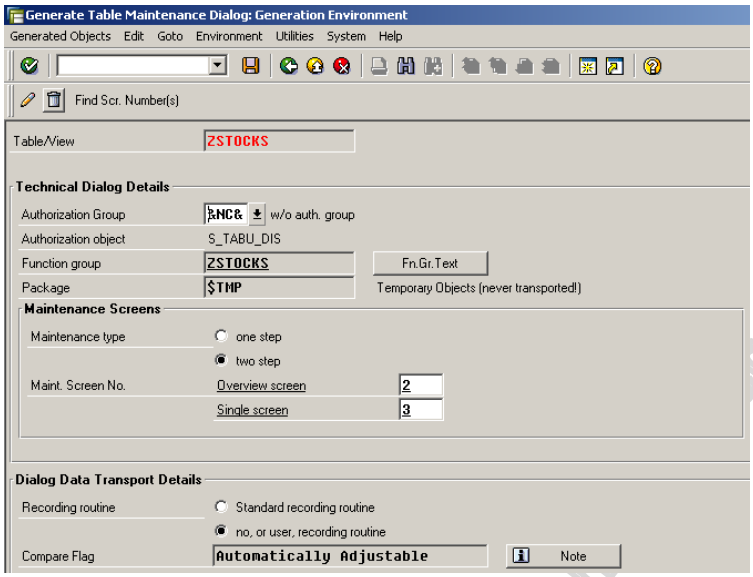
- En el tab (solapa)de Value Range puedo especificar qué datos incluyo:



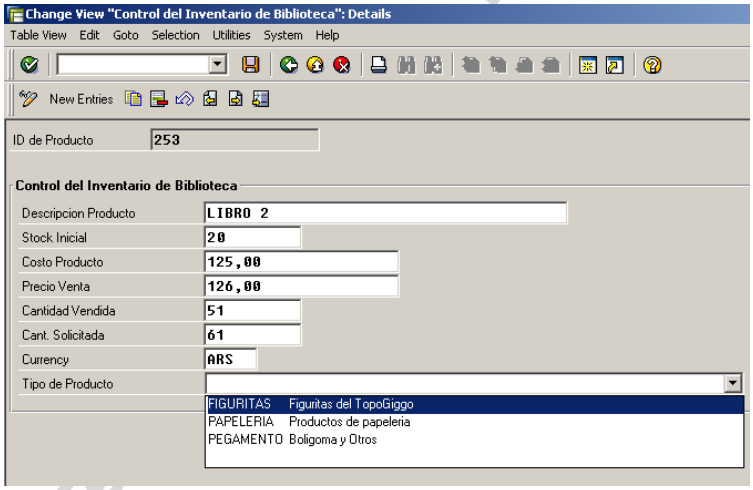
- Luego lo activo, y regreso a la SE11, edito la tabla (como se verá en el apartado siguiente) y creo un campo nuevo llamado por ejemplo Ztipo al que le asignare un dominio en el elemento de datos:



- Para activar esto, vuelvo a ir a Generate Table Mantenaince y elimino los seteos viejos y los vuelvo a crear nuevamente. Con esta operación activo la nueva vista:

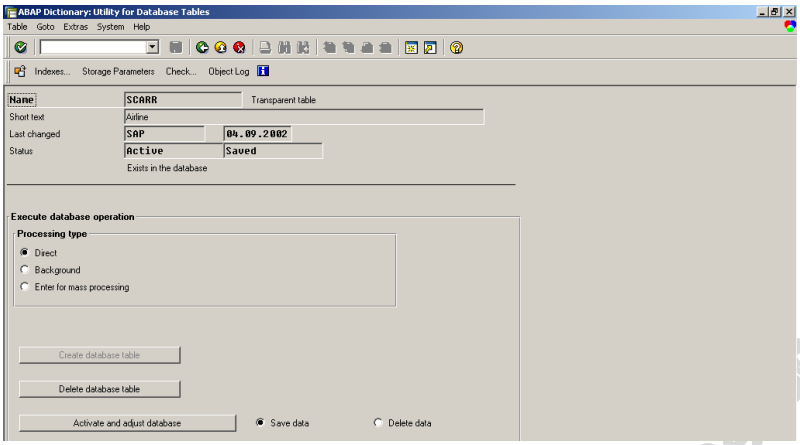


- Luego con la SE16 (Data Browser) puedo volver a ver y encontrare el nuevo campo:



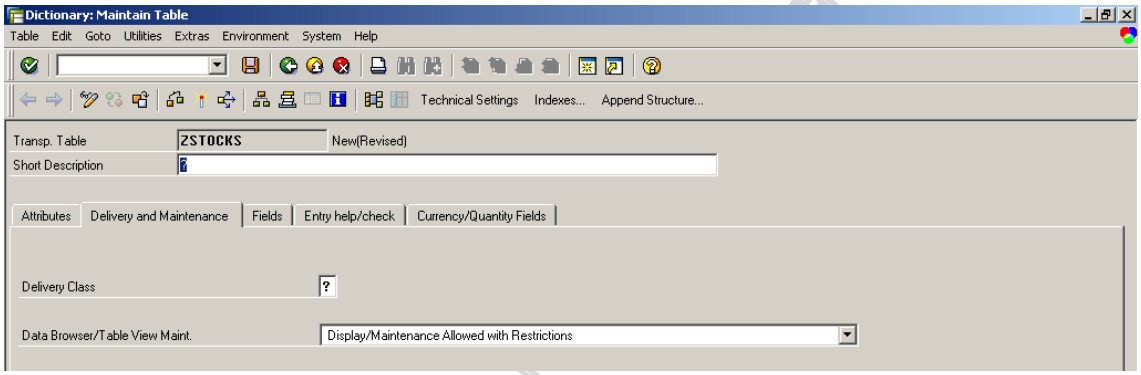
Modificación de tablas

Accedemos con la transacción SE14, que sirve para editar Tablas:

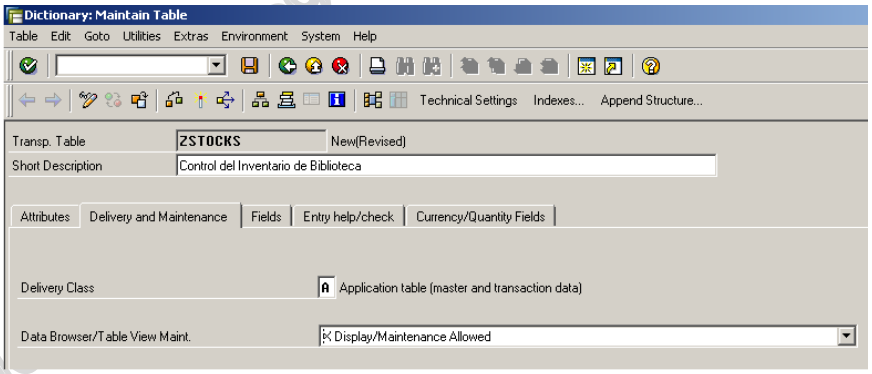


Paso a Paso:

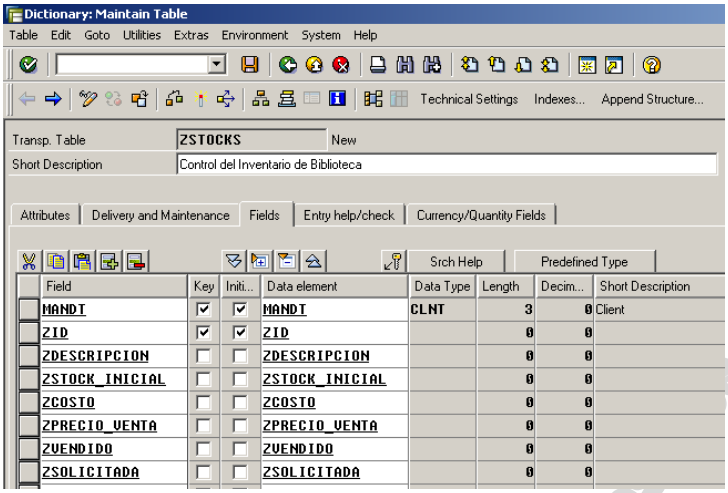
- creo la tabla



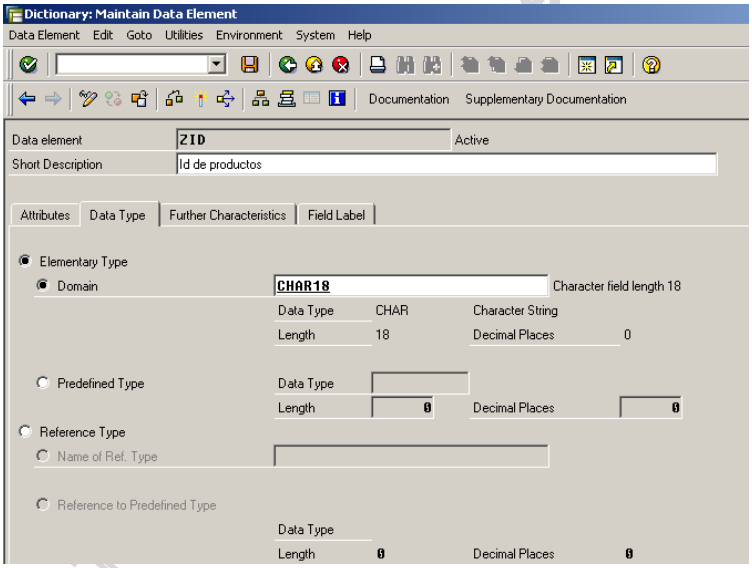
- Descripción



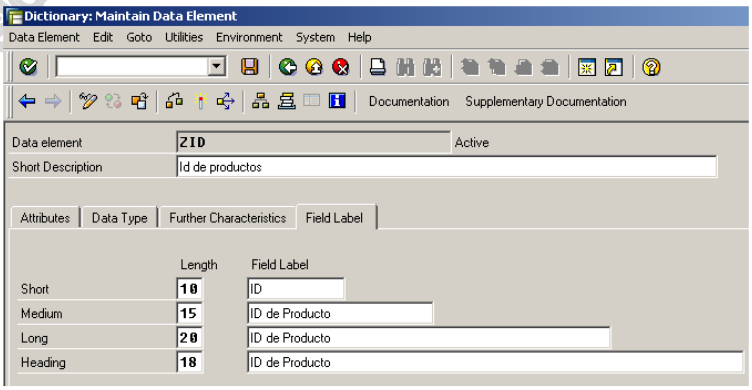
- Defino los campos:



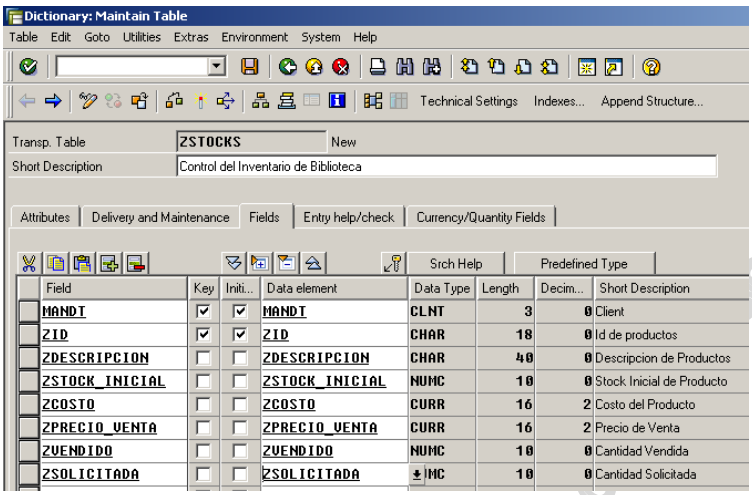
- Ahora especifico cada elemento de datos creado. (haciendo doble clic en el nombre del elemento de datos):



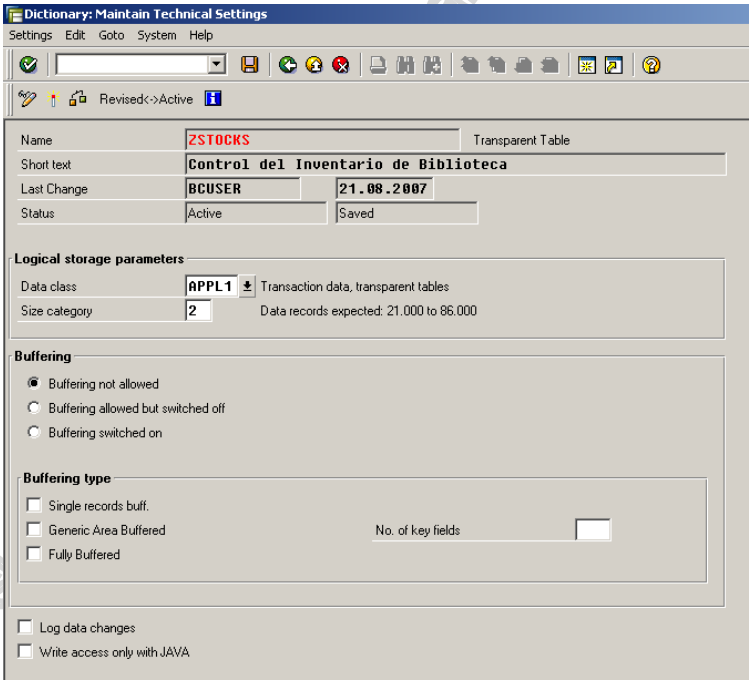
- Y en el tab de Field Label



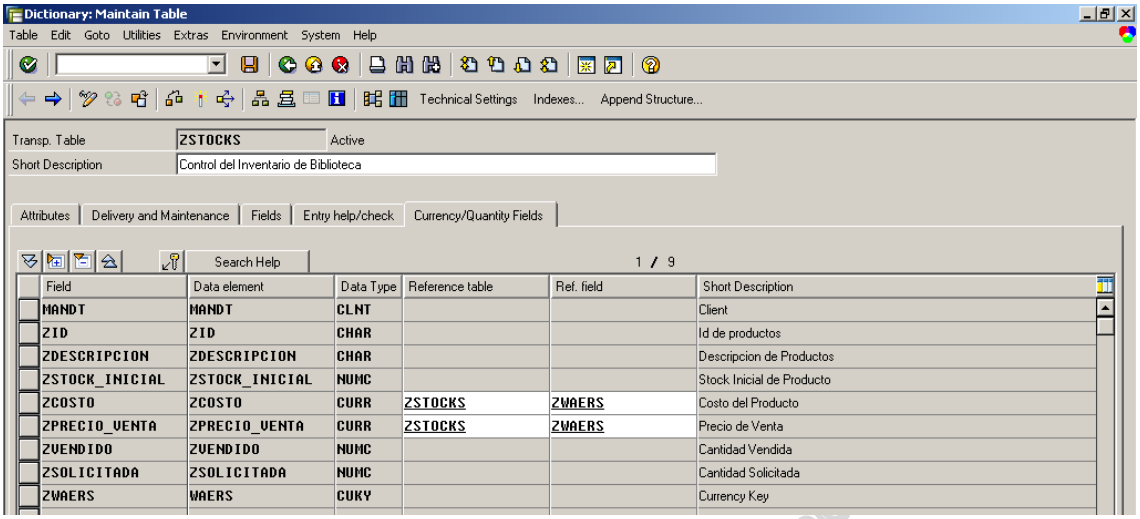
- Para que esto se vea en el tabla debo activarlos.



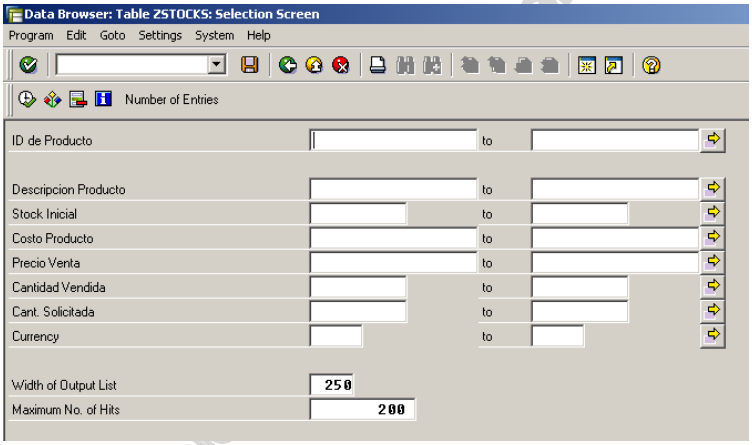
- Cuando creamos el campo zwaers, ojo que se auto completa por que el elemento waers es un campo de datos Standard. En Opciones técnicas de la tabla, defino la clase de datos y el tamaño



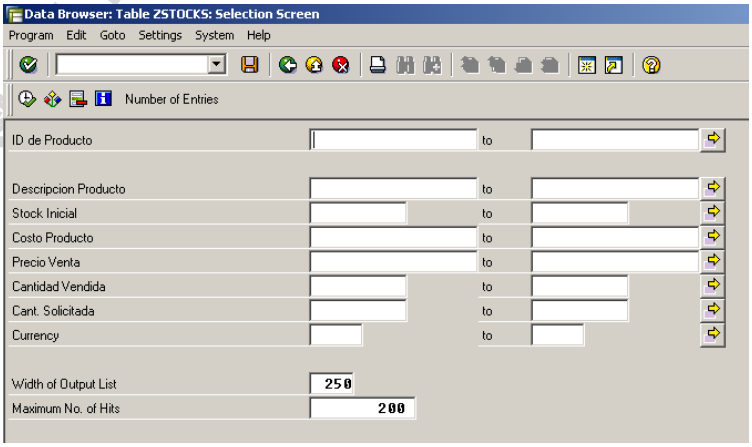
- Ojo que para poder grabar debemos antes especificar los detalles de los campos de monedas:



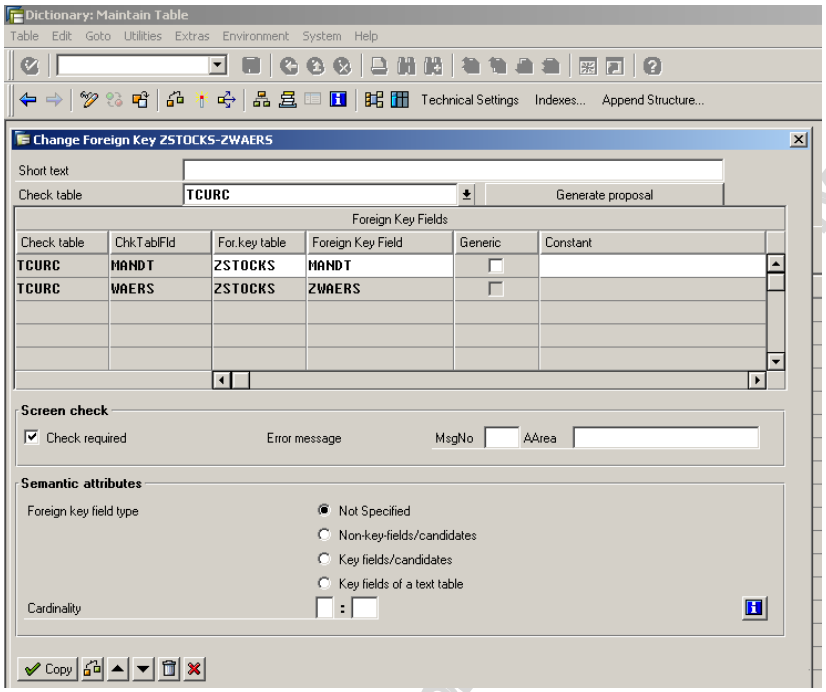
- Si voy al boton de Contents puedo ver el



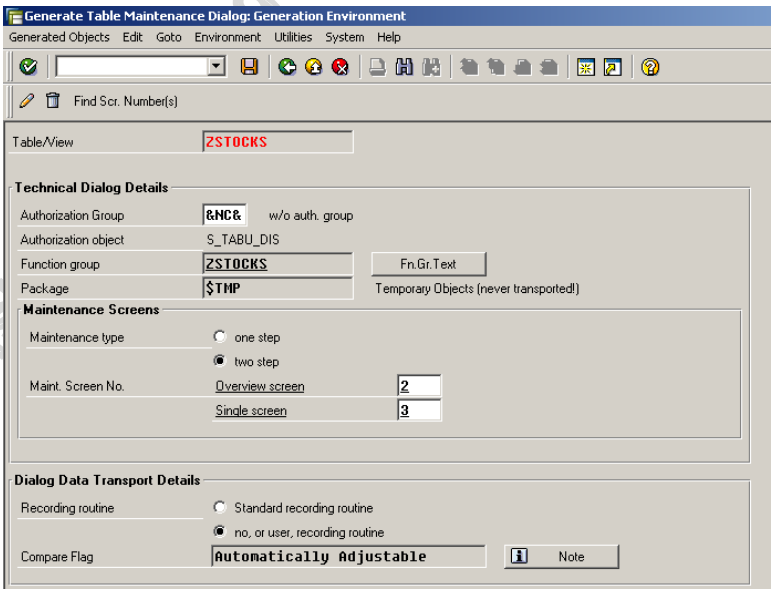
- Ya a esta altura con la SE16 (Data Browser), puedo ver ya la tabla y dar de alta registros en ella. Con esto agrego datos manualmente.



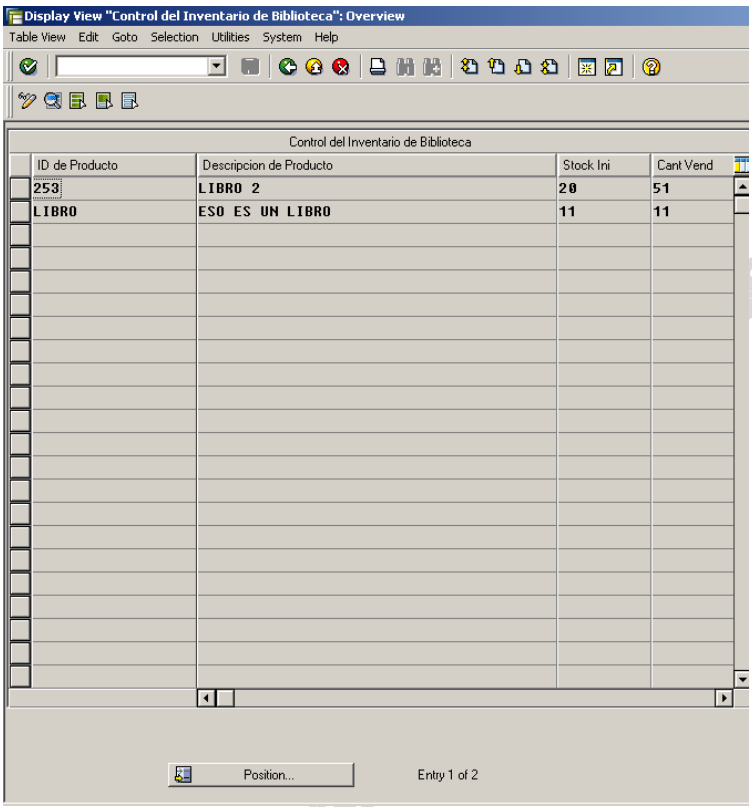
- Ojo que aquí el campo Currency puedo agregarle cualquier texto o nro. Para solucionar eso debo especificar que sea clave foranea. Para esto me paro en el campo y presiono la llavecita.



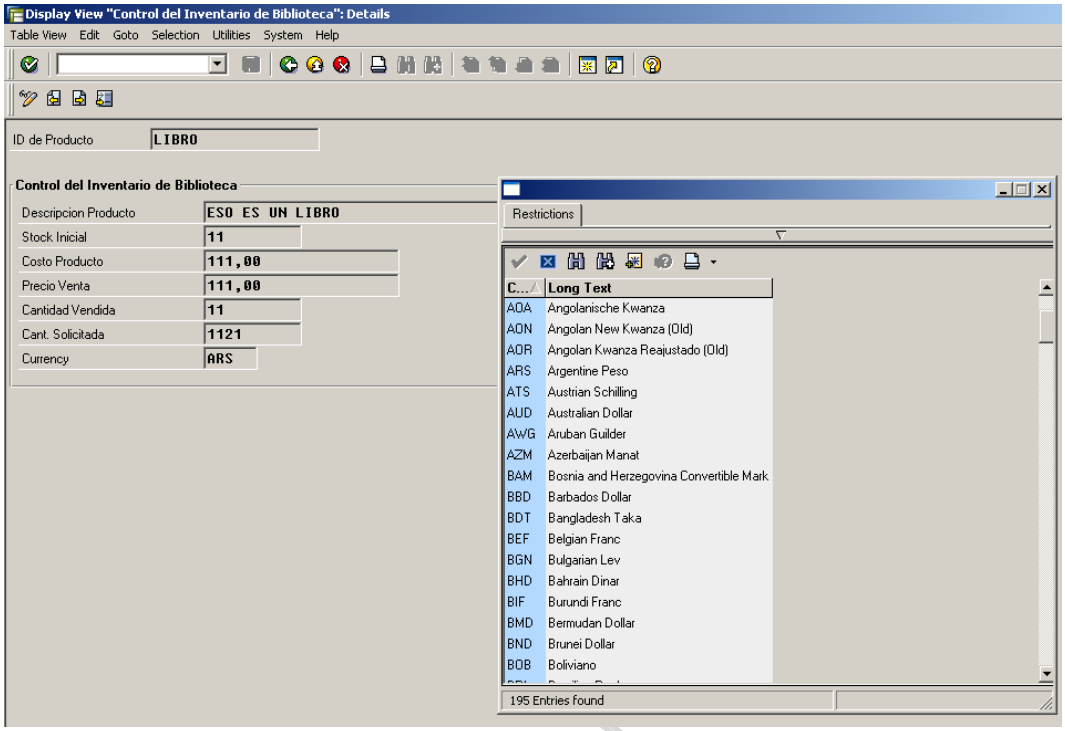
- Copio la clave propuesta y me completa que chequeara la información ingresada en el campo con la tabla TCurC. Luego activo la tabla y cuando vuelvo a la SE16 a agregar un registro. Luego de generar la tabla con el comando de Generate Table Manteince y defino los siguientes parámetros:



- Puedo ir al SM30 y puedo ya ver los cambios realizados y agregar registros a mi tabla:



- O bien añadir registros, aquí puedo ver que las configuraciones de moneda si estan funcionando y que en el campo Currency ya no puedo poner cualquier cadena de texto:



Tipos de datos

Los tipos de datos que se pueden utilizar en ABAP /4 son:

Tipos	Long. por defecto	Posible longitud	Valor inicial	Descripción
C	1	1-32000	ESPACIOS	Texto
F	8	8	0.0E+00	Punto flotante
I	4	4	0	Entero
N	1	1-32000	'0000'	Texto numérico
P	8	1-16	0	Número Empaquetado
X	1	1-29870	x'00'	Hexadecimal
D	8	8	00000000	Fecha YYYYMMDD
T	6	6	000000	Hora HHMMSS

Tipos de variables

Podemos decir que existen 2 tipos de variables; las que ya están definidas por R/3 y las que el programador define para realizar sus procedimientos dentro del programa. Pero esto no quiere decir que los primeros no pueden ser accedidos por el usuario:

- **Variables del Sistema**: Estas variables han sido definidas en el sistema y se encuentran en la estructura SYST, son flags que nos indican, dependiendo de la operación que se realice, un estado de dicha operación, es decir, nos pueden indicar si se realizó la operación correctamente, la fecha de sistema, la hora del sistema, etc. Como se puede en la siguiente figura, existe un gran número de estas variables, aunque las más usuales son:

  SY-SUBRC -> Código de retorno posterior a determinadas operaciones.
  SY-INDEX -> Cantidad de repeticiones de bucles.
  SY-DATUM -> Fecha del día.
  SY-PAGNO -> Número de página.
  SY-ULINE -> CONSTANTE: Dibuja una línea horizontal.

  Todas estas variables se leen e imprimen de la misma forma que las variables que son definidas por el programador, la única diferencia consiste en que este tipo de variables no pueden ser modificadas por el programador, únicamente cambia su valor dependiendo de las operaciones efectuadas dentro del programa.

- **Variables de programa**: Se utiliza la forma básica de declaraciones de datos para definir variables en un programa, con la siguiente sintaxis:

  DATA v1 [(longitud)] [TYPE t] [DECIMALS d] [VALUE 'xxx']

  o:

  DATA v1 LIKE v2 [VALUE 'xxx'].

  donde: v1 es el nombre de la variable,
  v2 es el nombre de la variable previamente definida en el programa o es el nombre de un campo que pertenece a una tabla o a una estructura en el Diccionario de Datos,
  (longitud) es la longitud,
  t es el tipo de dato,
  d es el número de dígitos después del punto decimal (máximo 14, únicamente usado con un dato de tipo p),
  'xxx' es el valor inicial de la variable.

  Ejemplos:

  ```
  data f1 type c.
  data variable2 like f1.
  data max_valor type i value 100.
  data fecha type d value '19980211'.
  data flag type c value is initial*.
  ```
  *IS INITIAL indica que la variable flag tiene el valor inicial de una variable de tipo c.

Si no se indica el tipo de la variable, será tipo carácter (Texto) y si no se indica longitud, la que tenga el tipo por defecto (ver tabla en figura anterior).

Ejemplos:

DATA VAR_CAR. . Creará una variable texto de longitud 1.
DATA VAR_CAR(8). Creará una variable texto de longitud 8.

Los nombres de las variables pueden ser de longitud 1 a 20 caracteres. Pueden contener cualquier carácter excepto ( ) + . , :

Con el parámetro TYPE podemos utilizar otros tipos de datos.

Ejemplos:

DATA NUM_CAR(5) TYPE N.
DATA NUMERO(2) TYPE P.
DATA FECHA_LIMITE TYPE D.

Con el parámetro LIKE podemos declarar una variable con los mismos atributos de longitud y tipo que una variable de base de datos (con las mismas propiedades).

Ejemplos:

DATA ACREEDOR LIKE LFA1-LIFNR.
o
DATA LIFNR LIKE LFA1-LIFNR.

A pesar de que el nombre de las variables puede ser el que el programador desee, es muy recomendable que cuando se utilice el comando LIKE, el nombre de las variables sea el mismo que el del campo de la tabla del diccionario de datos al que se hace referencia.
Con el parámetro VALUE podemos inicializar la variable con un valor distinto al que tiene por defecto.

Ejemplo:

DATA CONTADOR TYPE P VALUE 1.

Un registro de datos es un conjunto de campos relacionados lógicamente en una estructura.

Ejemplo:

```
DATA: BEGIN OF PROVEEDOR,
LIFNR LIKE LFA1-LIFNR,
NAME1 LIKE LFA1-NAME1,
CIUDAD(20) VALUE 'BARCELONA',
FECHA TYPE D,
END OF PROVEEDOR.
```

Posteriormente el acceso a los campos del registro de datos será :

```
WRITE: PROVEEDOR-NAME1,
PROVEEDOR-FECHA.
```

También usaremos la instrucción DATA para declarar tablas internas.
La diferencia entre una tabla interna y una tabla del diccionario de datos es que las primeras guardan los datos en memoria y no en la base de datos, mientras que la segunda guarda la información en la base de datos.
Una estructura es definida en le diccionario de ABAP/4 como las tablas y pueden ser direccionadas desde los programas de ABAP/4. Cualquier campo en la definición de la estructura en el diccionario de ABAP/4 es automáticamente reflejada en todos los programas. Mientras que los datos en las tablas de datos se almacenan de manera permanente, las estructuras únicamente contiene datos durante el tiempo de ejecución del programa. La única diferencia es que no se genera ninguna tabla en la base de datos.

Ejemplo:

```
DATA: BEGIN OF MEJORES_PROVEEDORES OCCURS 100,
NOMBRE LIKE LFA1-NAME1,
CIUDAD LIKE LFA1-ORT1,
VENTAS LIKE LFC3-SOLLL,
END OF MEJORES_PROVEEDORES.
```

La cláusula OCCURS determina el número de líneas guardadas en memoria principal. Esto no significa que el tamaño máximo de la tabla sea el indicado, ya que si este se desborda los datos se guardan en un fichero de paginación, bajando lógicamente el tiempo de proceso de las tablas internas, pero evitando que el área global de almacenamiento destinado por SAP para tablas internas se agote.
Las tablas internas se declaran, inicializan y referencian como un registro de datos.
También podemos utilizar la misma estructura que una tabla de base de datos, para ello utilizaremos la instrucción INCLUDE STRUCTURE.

Ejemplo:

```
DATA BEGIN OF SOCIEDADES OCCURS 10.
INCLUDE STRUCTURE T001.
DATA END OF SOCIEDADES.
```

Si se desea chequear la longitud o el tipo de una variable podemos utilizar la instrucción DESCRIBE FIELD.

Sintaxis:

DESCRIBE FIELD campo LENGTH longitud.
DESCRIBE FIELD TYPE tipo.
DESCRIBE FIELD OUTPUT-LENGTH long_salida.
DESCRIBE FIELD DECIMALS PLACES decimales.

Para chequear la longitud de un campo utilizamos la cláusula LENGTH.
Para conocer el tipo de datos del campo utilizamos TYPE.
Para conocer la longitud de salida utilizamos OUTPUT-LENGTH.

Para saber el número de decimales que tiene una cierta variable utilizaremos la cláusula DECIMALS.

Para inicializar las variables utilizamos la sentencia: CLEAR <campo>.

CLEAR inicializa al valor que tiene asignado como valor inicial(ver tabla) sin tener en cuenta las cláusulas VALUE que haya.
La asignación e inicialización de los registros de datos funciona de la misma forma que en las variable normales. Un CLEAR inicializa todos los campos del registro. Podremos conseguir una asignación mas potente con MOVE-CORRESPONDING:

MOVE-CORRESPONDING <reg1> TO <reg2>.

Esta instrucción mueve de reg1 a reg2 aquellos campos que tengan idéntico nombre.

Declaración de constantes.

Para declarar una constante se procede igual que para definir una variable, únicamente que la sentencia para definirlas es diferente como se muestra en la siguiente sintaxis:

constants c1 [(longitud)] [tipo t] [decimales d] [valor 'xxx'],

donde: c1 es el nombre de la constante,
longitud es la longitud,
tipo especificación del tipo de dato,
decimales es el número de dígitos después del punto decimal.

Ejemplos:

constants myname (20) value ‘Martín’.
constants birthday type d value ‘19760321’.
constants zero type i value is initial.

Declaración de tipos

El parámetro TYPES es utilizado para crear tipos de datos y estructuras de datos:

Types t1 (longitud) [tipo t] [decimales d]

Ejemplos:

```
types: surname (20) type c,
begin of address
name type surname,
end of address.
Data: address1 type address,
address2 type address.


data counts type i.
types: company like spfli_carrid,
no_flights like counts


types: surname(20) type c,
street(30) type c,
cp(10) type n,
city(30) type c,
phone(20) type n,
fecha like sy_datum.
types: begin of address,
name type surname,
code type cp,
town type city,
str type street,
end of address.
types: begin of phone_list,
adr type address,
tel type phone,
end of phone_list.
data datos1 type phone_list.
```

Asignación de valores

Existen diversas formas de asignar valores a una variable en ABAP/4. Una asignación directa, como resultado de una operación aritmética o como resultado de una conversión automática entre campos con valores de diferente tipo de datos. La instrucción MOVE realiza un transporte del contenido del var1 al campo var2:

MOVE <var1> TO <var2>.

Podemos sustituir esta última instrucción por:

<var2> = <var1>.

que es la simplificación de:

COMPUTE <var2> = <var1>.

donde la palabra clave COMPUTE es opcional.

También es posible referenciar o asignar valores a una parte de la variable utilizando el offset:

VARIABLE**+offset(longitud)**

Offset es la posición a partir de la cual se moverá el dato
Longitud es la cantidad de caracteres que se desean mover.

Ejemplo:

```
DATA: VAR1(15) VALUE 'ESCADOR MADRID',
VAR2(15) VALUE 'HOLA'.
MOVE VAR1+8(6) TO VAR2+5(6).
WRITE VAR2.
```

Resultado:

HOLA BCN.

MOVE VAR1+8(6) TO VAR2+5(6).

HOLA MADRID

Si se desean utilizar variables en el offset o la longitud se usará la instrucción WRITE TO, la cual es la única forma de poder hacer esto.

Ejemplo:

```
OFF1 = 10.
OFF2 = 5.
LEN = 4.
WRITE VAR1+OFF1(LEN) TO VAR2+OFF2(LEN).
```

Conversión de tipo.

Si intentamos realizar una asignación de variables de distinto tipo, ABAP/4 intenta realizar una conversión automática de tipo.

Operaciones aritméticas en Abap/4.

En ABAP/4 las 4 operaciones aritméticas básicas se pueden implementar:

- Con la instrucción COMPUTE y los símbolos + , - , / , *.

COMPUTE var1 = <Exp. Aritmética>.

donde la palabra COMPUTE es opcional.

Por ejemplo:

Código:

```
data: var1 type n value ’3’,
var2 type c value ‘2’,
var3 type n .
var3 = var1 + var2. ‘o podría ser COMPUTE var3 = var1 +
var2., el resultado es el mismo
Write ‘VAR3 =’, var3.
```

Salida:

Var3 = 5.

Si utilizamos paréntesis dejaremos un espacio en blanco precediendo y siguiendo al paréntesis.

- Con las instrucciones: ADD TO, SUBSTRACT FROM, MULTIPLY BY y DIVIDE BY.

También dispondremos de funciones matemáticas para los números de coma flotante: EXP, LOG, SIN, COS, SQRT, DIV, MOD, STRLEN.

Procesando campos de tipo texto

ABAP/4 ofrece algunas instrucciones para el procesamiento de cadenas de texto. Para realizar un desplazamiento del contenido de un campo utilizamos SHIFT:

SHIFT <campo>. Realiza un desplazamiento de un carácter hacia la izquierda.
SHIFT <campo> BY <n> PLACES (RIGHT). Realiza un desplazamiento de n

caracteres hacia la izquierda o si se especifica hacia la derecha, introduciendo blancos por el lado opuesto.

Ejemplo:

HOLA
SHIFT campo BY 2 PLACES.
LA

SHIFT <campo> BY 2 PLACES CIRCULAR (RIGHT).

Realiza un desplazamiento cíclico hacia la izquierda o si se especifica hacia la derecha.

Ejemplo:
HOLA
SHIFT campo BY 2 PLACES CIRCULAR.
LAHO

Podemos reemplazar el contenido de ciertos campos con la instrucción REPLACE:

REPLACE <cadena1> WITH <cadena2> INTO <campo>.

Reemplaza 'cadena1' por 'cadena2' dentro de la variable 'campo'. Si la variable del sistema SY-SUBRC <> 0 es que 'cadena1' no existe dentro de 'campo'. REPLACE únicamente sustituirá la primera aparición de 'cadena1'.

Ejemplo:

Código:

data field(10).
move 'ABCB' to field.
replace 'B' with 'string' into field.
write: field.

Salida:

AstringCB

Existe otra instrucción de sustitución, TRANSLATE:
TRANSLATE <campo> TO UPPER CASE pasa a Mayúsculas.
TO LOWER CASE pasa a Minúsculas.

Ejemplo:

Código:

```
data var1(20) type c value 'Escador'.
translate var1 to UPPER CASE.
write:/ var1.
```

Salida:

ESCADOR.

La instrucción SEARCH busca una cadena dentro de un campo o una tabla:
SEARCH <campo>/<tabla> FOR <cadena>.
Si el resultado es positivo SY-SUBRC = 0. En caso de que sea una tabla interna, SY-TABIX contiene la línea de la tabla donde se ha encontrado.

Ejemplo:

NOMBRE	*SEXO*
Ariel Martínez	M
Lorena Jimenez	F
Miriam Trinado	F
Sergio Mondragon	M

Código:

```
tables: tabla1.
data var1(10) type c value 'Lorena'.
search tabla1 for var1.
if sy-subrc = 0.
Write: 'LA CADENA FUE ENCONTRADA EN LA LINEA:', sy-tabix.
else.
Write: 'LA CADENA NO FUE ENCONTRADA'.
endif.
```

Salida:

LA CADENA FUE ENCONTRADA EN LA LINEA: 2

En este ejemplo, se puede ver que sy-subrc es igual a cero, lo que significa que si existe la cadena dentro de la tabla1, una vez que entra al if, imprime con la ayuda de la variable del sistema la línea en la que se encuentra.

Para borrar los blancos de una cadena utilizaremos CONDENSE:
CONDENSE <campo> (NO-GAPS).

Esta instrucción borra todos los blancos que se encuentren comenzando la cadena por la izquierda y en caso de encontrar series de blancos intermedios dejará únicamente uno por serie.

Ejemplo :
“ CURSO DE ABAP/4” “CURSO DE ABAP/4”

La cláusula NO-GAPS borra todos los blancos estén donde estén.

Control de flujo en los programas Abap/4.

En ABAP, como en todos los lenguajes estructurados, disponemos de una serie de instrucciones para subdividir el programa en bloques lógicos que se ejecutarán cuando se cumpla una cierta condición.
Para introducir una condición utilizaremos la sentencia IF ... ELSE ... ENDIF que podrá aparecer en distintas modalidades:

```
IF <Cond.>.
    ...
ENDIF.
```

```
IF <Cond.>.
    ...
ELSE.
    ...
ENDIF.
```

```
IF <Cond.>.
  ...
ELSEIF.
    ...
ELSEIF.
    ...
ELSE.
    ...
:
ENDIF.
```

En las condiciones utilizamos los clásicos operadores lógicos y relacionales:

Y AND
O OR
Igual = , EQ
Distinto <> , NE
Mayor > , GT
Menor < , LT
Mayor o igual >= , GE
Menor o igual <= , LE

Además existen operadores adicionales para comparar cadenas de caracteres:

<f1> CO <f2> (Contains Only): f1 sólo contiene caracteres de f2.

En caso de ser cierta SY-FDPOS contiene la longitud de f1 y si es falsa contiene el offset del 1er. carácter que no cumple la condición.

Ejemplos:

'ABCDE' CO 'XYZ' => SY-FDPOS = 0
'ABCDE' CO 'AB' => SY-FDPOS = 5
'ABCDE' CO 'ABCDE' => SY-FDPOS = 5

<f1> CN <f2> (Contains Not Only) : Negación de la anterior.
<f1> CA <f2> (Contains Any) : f1 contiene como mínimo algún carácter de f2.
Si es cierta SY-FDPOS contiene el offset del 1er. carácter de f1 que está en f2 y si es falsa contiene la longitud de f1.

Ejemplos:

'ABCDE' CA 'CY' => SY-FDPOS = 2
'ABCDE' CA 'XY' => SY-FDPOS = 5

<f1> NA <f2> (Contains Not Any) : Negación de la anterior.
<f1> CS <f2> (Contains String) : f1 contiene la cadena f2. Si la condición es cierta SY-FDPOS contiene el offset donde empieza f2 en f1 y si es falsa contiene la longitud de f1.

Ejemplos:

'ABCDE' CS 'CD' => SY-FDPOS = 2.
'ABCDE' CS 'XY' => SY-FDPOS = 5.
'ABCDE' CS 'AB' => SY-FDPOS = 0.
' ABC' CS ' AB' => SY-FDPOS = 1.
'ABCDEF ' CS ' ' => SY-FDPOS = 0.

<f1> NS <f2> (Contains No String) : Negación de la anterior.
<f1> CP <f2> (Contains Pattern) : f1 corresponde al patrón f2.

En el patrón podemos utilizar: + como cualquier caracter, * como cualquier cadena de caracteres, # para utilizar los caracteres +,*,# en la comparación. Si la condición es cierta SYFDPOS contiene el offset de f2 en f1 y si es falsa contiene la longitud de f1.

Ejemplos:

'ABCDE' CP '*CD*' => SY-FDPOS = 2.
'ABCDE' CP '*CD' => SY-FDPOS = 5.
'ABCDE' CP '++CD+' => SY-FDPOS = 0.
'ABCDE' CP '+CD*' => SY-FDPOS = 5.
'ABCDE' CP '*B*D*' => SY-FDPOS = 1.

<f1> NP <f2> (Contains No Pattern) : Negación de la anterior.

También podremos utilizar operadores especiales:

IF <f1> BETWEEN <f2> AND <f3>. Para chequear rangos.
IF <f1> IS INITAL. Para chequear valores iniciales.

Si queremos ejecutar diferentes instrucciones en función del contenido de un campo podemos utilizar la sentencia CASE:

```
CASE <campo>.
WHEN <valor1>.
....
WHEN <valor2>.
....
:
WHEN OTHERS.
....
ENDCASE.
```

Por último existe la instrucción condicional, ON CHANGE OF ... ENDON, que permitirá la ejecución de un bloque de instrucciones, si se ha producido un cambio de valor de un cierto campo durante el acceso a base de datos o una tabla interna (como procesar una tabla interna o un acceso a base de datos, ya lo veremos más adelante):

```
ON CHANGE OF <campo>.
....
ENDON.
```

Proceso de bucles

Para realizar procesos repetitivos utilizaremos DO y WHILE. La instrucción DO permite ejecutar un bloque de instrucciones tantas veces como se especifique:

```
DO <n> TIMES.
...
ENDDO.
```

En la variable del sistema SY-INDEX tendremos un contador del número de repeticiones. Es posible anidar DO's. En ese caso el SY-INDEX hará referencia al bucle en proceso.

La instrucción WHILE permite ejecutar un bloque de instrucciones mientras se cumpla una condición:

```
WHILE <cond>.
...
ENDWHILE.
```

De la misma forma que la instrucción DO, WHILE permite anidar bucles.

Sentencias de control

Las sentencias descritas a continuación se utilizarán para terminar el procesamiento de un bucle o proceso.
La instrucción *CHECK <cond>* realiza un chequeo de <cond> de forma que si dentro de un bucle la condición es falsa, saltará todas las instrucciones que siguen al CHECK e iniciará la siguiente pasada al bucle. Fuera de un bucle si la condición es falsa, saltará todas las instrucciones que siguen al CHECK hasta el final del evento o programa en proceso (si no estamos usando eventos, lo que se verá más adelante).

Ejemplo:

```
DO 4 TIMES
CHECK SY-INDEX BETWEEN 2 AND 3.
WRITE SY-INDEX.
ENDDO.
```

Produce la siguiente salida:

2 3
Aquí el programa termina la primera y la cuarta pasada del bucle sin procesar la sentencia WRITE porque SY-INDEX no está entre 2 y 3.

La instrucción *EXIT,* dentro de un bucle saldrá del bucle y fuera de un bucle saldrá del programa. Si la instrucción EXIT está dentro de varios bucles anidados, únicamente saldrá del bucle en proceso.

Ejemplo:

```
DO 4 TIMES.
IF SY-INDEX = 3.
EXIT.
ENDIF.
WRITE SY-INDEX.
ENDDO.
```

Produce la siguiente salida:

1 2

El sistema termina el procesamiento del bucle en la tercera pasada sin ejecutar el WRITE ni la cuarta pasada.

La instrucción *CONTINUE* termina un bucle incondicionalmente. Sólo se puede usar dentro de bucles. Después de la sentencia CONTINUE, el sistema salta todas las sentencias siguientes en el bloque en el que estamos y continúa con la siguiente pasada.

Ejemplo:

```
DO 4 TIMES.
IF SY-INDEX = 2.
CONTINUE.
ENDIF.
WRITE SY-INDEX.
ENDDO.
```

Produce la siguiente salida:

1 3 4
El sistema termina la segunda pasada del Do sin procesar la sentencia WRITE.

La instrucción *STOP* finaliza el report (programa) en ejecución, pero antes ejecutaremos el evento END-OF-SELECTION (que se verá más adelante).

La instrucción *LEAVE* finaliza el report (programa) en ejecución, sin ejecutar el evento ENDOF-SELECTION (lo cual se verá más adelante).