GeneXus

Diseño de Aplicaciones

ARTech

TABLA DE CONTENIDO

Todos los nombre de productos mencionados en este documento son marcas de sus respectivos dueños.

DISEÑO DE APLICACIONES CON *GENEXUS*

INTRODUCCIÓN

El presente documento es una introducción al desarrollo de aplicaciones utilizando ***GENEXUS***. Está dirigido a profesionales de informática que se inician en el uso de ***GENEXUS***.

GENEXUS es una herramienta para el desarrollo de aplicaciones. Su objetivo es ayudar a los analistas de sistemas a implementar aplicaciones en el menor tiempo y con la mejor calidad posible.

A grandes rasgos, el desarrollo de una aplicación implica tareas de análisis, diseño e implementación. La vía de ***GENEXUS*** para alcanzar el objetivo anterior es liberar a las personas de las tareas automatizables (por ejemplo, la implementación), permitiendoles así concentrarse en las tareas realmente difíciles y no automatizables (análisis y diseño).

Desde un punto de vista teórico, ***GENEXUS*** es más una metodología de desarrollo de aplicaciones que una herramienta de software. Como metodología, tiene algunos puntos de contacto con las metodologías tradicionales, pero también aporta enfoques bastante diferentes en otros.

En común con las metodologías tradicionales, se mantiene la importancia del análisis y diseño de la aplicación sobre la implementación. Quizás con ***GENEXUS*** este enfoque se resalta más aún, ya que el usuario de ***GENEXUS*** estará la mayor parte del tiempo realizando tareas de análisis y ***GENEXUS***, en sí, tareas de implementación (por ejemplo, normalización de la base de datos, generación de programas, etc.).

Por otro lado, algunos de los enfoques metodológicos son bastante diferentes que los habituales, como, por ejemplo, el comenzar el análisis de la aplicación por la interfase del mismo con el usuario, la casi nula referencia a la implementación física del sistema, etc.

Para presentar estos nuevos conceptos, y a los efectos de no realizar una presentación demasiado abstracta del tema, se ha elegido una aplicación que se irá desarrollando a través de los distintos capítulos.

El primer capítulo presenta la aplicación y los aspectos iniciales de un desarrollo con ***GENEXUS***.

El segundo capítulo trata el diseño de Transacciones, el tercero de Reportes, el cuarto de Procedimientos, el quinto de Paneles de Trabajo y por último se trata el diseño de Menúes.

Debido a que en todos los capítulos se asume cierto conocimiento sobre Bases de Datos Relacionales, se ha incluido un anexo sobre el tema que describe los conceptos necesarios para este documento. Se recomienda su lectura antes de leer el segundo capítulo.

Por razones didácticas, en este documento no se tratan los siguientes temas:

- ***Ciclo de vida de una aplicación:*** Una aplicación tiene un ciclo de vida, que comienza cuando se planifica la misma y termina cuando la aplicación sale de producción. ***GENEXUS*** acompaña todo este ciclo. Este tema es tratado en el documento Visión General, que recomendamos leer previamente.

- ***Uso de GENEXUS :*** Toda la información respecto a la operación de ***GENEXUS*** en sí (manejo de teclas, edición de texto, etc.) se trata en el Tutorial ***GENEXUS***, que sugerimos repasar posteriormente.

DESARROLLO DE UNA APLICACIÓN

La mejor forma de aprender a utilizar ***GENEXUS*** es realizando aplicaciones con él. La aplicación que se ha elegido es una simplificación de una aplicación real.

SISTEMA DE COMPRAS PARA UNA CADENA DE FARMACIAS.

En una cadena de farmacias se desea implementar un sistema de compras que permita a los analistas de compras realizar dicha tarea con la mayor cantidad de información posible. La función de un analista de compras es decidir los pedidos que se deben efectuar a los proveedores de los distintos artículos.

Funcionamiento del sistema:
Se desea que el analista de compras utilice el computador para definir los pedidos a los distintos proveedores. Una vez que el pedido este hecho y aprobado, se quiere que el computador emita las ordenes de compra. En el momento de hacer un pedido es necesario hacer varias consultas, por ejemplo cuanto hay en stock, cual fue el precio de la última compra, etc.
Los siguientes puntos describen los pasos seguidos para el desarrollo de esta aplicación.

Definir el objetivo

No se debe olvidar que los computadores son meras herramientas. Los usuarios de los sistemas tienen objetivos específicos. Ellos esperan que la aplicación los ayude a alcanzarlos mas rápido, mas fácil, o a un menor costo. Es parte del trabajo de análisis, el conocer esos propósitos y saber por medio de que actividades los usuarios quieren alcanzarlos.
Este objetivo debe poder ser expresado en pocas palabras (uno o dos párrafos) y ser conocido por todas las personas involucradas con el sistema.
En el ejemplo, alguno de los propósitos posibles son:

- "El sistema de compras debe disminuir la burocracia existente para la formulación de un pedido."
- "El sistema de compras debe asistir a usuarios no entrenados en la formulación de pedidos de tal manera que su desempeño se asemeje al de un experto."
- "El sistema de compras debe permitir la disminución del stock existente en las farmacias."

De todos los objetivos posibles se debe elegir uno como el objetivo principal o prioritario. Esto es muy importante para el futuro diseño de la aplicación. Basta con observar como los distintos objetivos anteriores conducen a diseños diferentes.

Para nuestro ejemplo elegiremos el primer objetivo, dado que en la situación real el analista de compras no tenia toda la información que necesitaba. Por lo tanto, debía consultar una serie de planillas manuales y llamar por teléfono a los empleados del deposito para que realizaran un conteo manual del stock.
No se debe confundir el objetivo de la aplicación (el QUE) con la funcionabilidad de la misma (COMO se alcanzará el objetivo).

Definir el equipo de trabajo

A continuación se debe definir cuál será el equipo de personas encargado de la implementación del sistema. Dicho equipo debe tener como mínimo dos personas:

- El analista de sistemas.
- Y un usuario.

Los analistas de sistemas que trabajen en el desarrollo de la aplicación deben cumplir dos condiciones:

- Haber sido entrenados en el uso de ***GENEXUS***
- Tener una orientación a las aplicaciones.

Se recomienda que dichos analistas pasen algún tiempo trabajando con los usuarios en el comienzo del proyecto a los efectos de familiarizarse con los conceptos, vocabulario, problemas existentes, etc.
Como la aplicación se desarrolla de una manera incremental, es MUY IMPORTANTE la participación de los usuarios en todas las etapas del desarrollo. Se recomienda tener un usuario principal disponible para la prueba de los prototipos y tener acceso a los demás usuarios de una manera fluida.
Dado que con ***GENEXUS*** los miembros del equipo estarán la mayor parte del tiempo trabajando en tareas de diseño y no de codificación, se debe tomar como criterio general el trabajar en equipos PEQUEÑOS, por ejemplo, no más de cinco personas.

Obtener una imagen global

Se debe tener entrevistas con el nivel gerencial mas alto que se pueda, de modo de obtener información sobre la posición relativa (e importancia) de la aplicación dentro de toda la organización.

Definir el alcance de la aplicación

Luego de un estudio primario se debe decidir cuál será el alcance de la aplicación para poder cumplir con el objetivo. Para ello se recomienda seguir el Principio de Esencialidad:

"Solo lo imprescindible, pero todo lo imprescindible"

En el ejemplo, una vez que una orden de compra es enviada a un proveedor, se debe controlar como y cuando se fueron entregando efectivamente los productos. Sin embargo vemos que esto no es imprescindible para cumplir el objetivo de la aplicación y por lo tanto no será tratado.

Mantener el diseño simple

Se debe planificar pensando en un proceso de diseño y desarrollo incremental. Comenzando por pequeños pasos y verificando la evolución del modelo frecuentemente con el usuario.

Orientarse a los datos

En esencia, una aplicación es un conjunto de mecanismos para realizar ciertos procesos sobre ciertos datos.
Por lo tanto, en el análisis de la aplicación, se puede poner mayor énfasis en los procesos o en los datos.
En las metodologías tradicionales -como el Análisis Estructurado- el análisis se basa en los procesos. En general, este análisis es Top-Down; se comienza con la definición más abstracta de la función del sistema, y luego se va descomponiendo esta en funciones cada vez más simples, hasta llegar a un nivel de abstracción suficiente como para poder implementarlas directamente.
Sin embargo, este enfoque tiene ciertos inconvenientes:

Las funciones de un sistema tienden a evolucionar con el tiempo, y por lo tanto, un diseño basado en las funciones será más difícil de mantener.
La idea que una aplicación se puede definir por una única función es muy controvertida en aplicaciones medias o grandes.
Frecuentemente se descuida el análisis de las estructuras de datos.
No facilita la integración de aplicaciones.

Si, por el contrario, el diseño se basa en los datos, se puede obtener las siguientes ventajas:

Más estabilidad. Los datos tienden a ser más estables que los procesos y en consecuencia la aplicación será más fácil de mantener.

Facilidad de integración con otras aplicaciones. Difícilmente una aplicación es totalmente independiente de las otras dentro de una organización. La mejor forma de integrarlas es tener en cuenta los datos que comparten. Incluso es posible que en un futuro el propio concepto de aplicación evolucione hacia el concepto de objeto, en donde los usuarios pedirán que se implementen nuevos objetos y no nuevas aplicaciones.

Sin embargo, si bien vemos que orientarse a los datos es beneficioso, la pregunta es como analizar los datos?. La respuesta de ***GENEXUS*** es analizar directamente los datos que el usuario conoce, sin importar como se implementarán en el computador.
El diseño comienza -como veremos más en detalle en el siguiente capítulo- estudiando cuales son los objetos que el usuario manipula. Para cada uno de estos objetos se define cual es su estructura de datos y posteriormente cual es su comportamiento.
De esta manera se alcanzan dos objetivos importantes: el análisis se concentra en hechos objetivos, y este puede ser evaluado directamente por los usuarios, utilizando la facilidad de prototipación de ***GENEXUS***.

DISEÑO DE TRANSACCIONES

El análisis mismo de la aplicación comienza con la definición de las transacciones. De cualquier manera es importante tener en cuenta que todo el proceso de desarrollo es incremental y por lo tanto no es necesario definir en esta etapa todas las transacciones y cada una de ellas en su mayor detalle. Por el contrario lo importante aquí es reconocer las mas relevantes y para cada una de ellas cual es la estructura mas adecuada.
Para poder definir cuales son las transacciones se recomienda estudiar cuales son los objetos (reales o imaginarios) que el usuario manipula. Afortunadamente es posible encontrar la mayor parte de las transacciones a partir de:

La descripción del sistema. Cuando un usuario describe un sistema se pueden determinar muchas transacciones si se presta atención a los sustantivos que utiliza. En el ejemplo:

Analistas de compras
Pedidos
Proveedores
Artículos
Ordenes de Compra

Formularios existentes. Por cada formulario que se utilice en el sistema es casi seguro que existirá una transacción para la entrada de los mismos.

Para cada transacción se puede definir:

Estructura
De que atributos (campos en la metodología tradicional) esta compuesta y que relación tienen entre si.

Pantalla o Form
Cual es el form que tiene. Esto se realiza con un editor especializado.

Fórmulas
Que atributos se calculan a partir de otros atributos. Por ejemplo: Valor = Cantidad * Precio.

Reglas

Conjunto de reglas que debe cumplir la transacción. Por ejemplo cuales son los valores por defecto de los atributos, cuales son los controles en los datos que hay que realizar, etc.

Eventos
Las transacciones soportan la programación dirigida por eventos. Este tipo de programación permite el almacenamiento de código ocioso el cual es activado luego de ciertos eventos - provocados por el usuario o por el sistema.

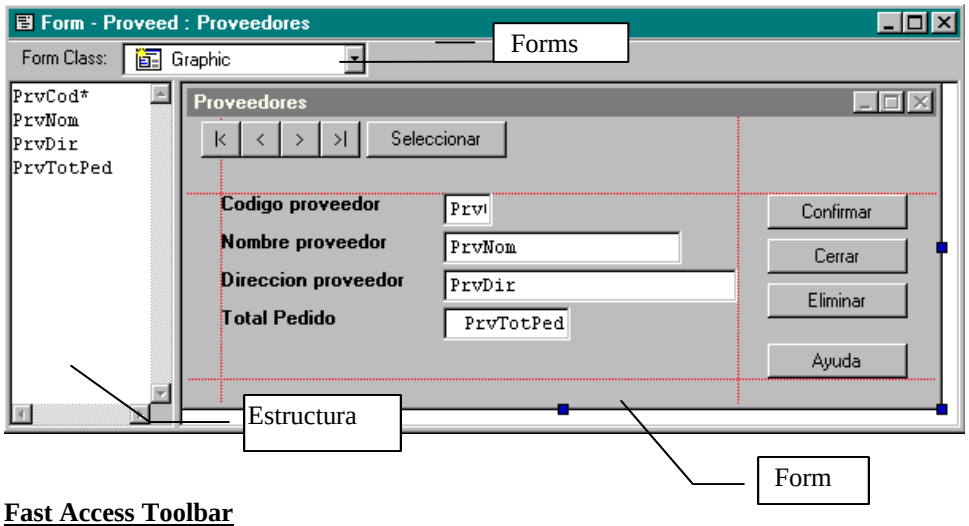
Propiedades
Reglas que definen el comportamiento general de la transacción.

Form Classes
Cada objeto puede tener asociado mas de un form que pertenece a una determinada Form Class. Existen dos Form Classes predefinidas: Graphic y Text. Tipicamente la Form Class Graphic se usa para los ambientes graficos (por ejemplo: Windows) y la form class Text se usa para los ambientes que trabajan en modo texto (por ejemplo: AS/400) El combo box que se muestra en la siguiente figura permite seleccionar un form (de determinada Form Class) de los que se hayan asociado al objeto.

Style Asociado
Styles son básicamente objetos *GENEXUS* (su forma de definición es similar a los otros objetos), pero no son tenidos en cuenta en la normalización o generación de programas; sólo se utilizan para definir estándares. Un ejemplo puede aclarar la idea: supongamos que se quiere definir las transacciones con botones con bitmaps en vez de los usuales de texto. Cuando se crea una transaccion se puede asociar un style en el cual se basara la transaccion.

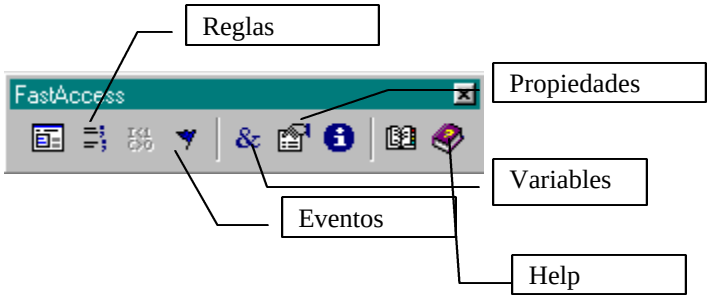
Ayuda
Texto para la ayuda a los usuarios en el uso de la transacción.

Documentación
Texto técnico que se incluye para ser utilizado en la documentación del sistema.



Fast Access Toolbar

Cuando se bare un objeto esta toolbar se habilita con las siguientes opciones:



Estructura de una transacción

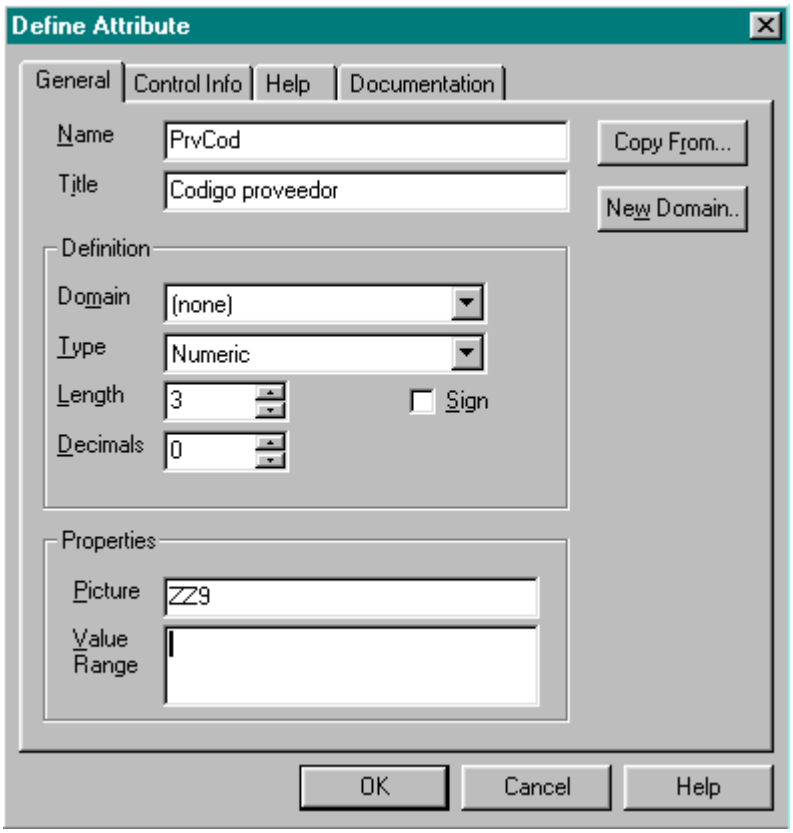
La estructura define que atributos (campos) integran la transacción y como están relacionados. En el ejemplo, la transacción de **Proveedores** posee los siguientes atributos:

PrvCod	Código de proveedor
PrvNom	Nombre del proveedor
PrvDir	Dirección del proveedor

que componen la información que se quiere tener de un proveedor.

Atributos

Para cada atributo se debe definir:



Name: Es el nombre del atributo. Se utiliza para identificar al atributo.

Title: Es un string que acompaña al atributo en los listados de documentación que tenemos disponibles y permite tener una descripción ampliada para éste. También se utiliza en los *Forms* de las transacciones y reportes que se crean por defecto.

Domain: Dominio en que se basa el atributo al definirlo.

Type: tipo de atributo (Numérico, alfanumérico, fecha, Long Varchar, Varchar o DateTime).

El tipo de dato **Long Varchar** (por Variable Character) permite almacenar una cantidad de caracteres indefinida. Se utiliza normalmente para almacenar textos, por ejemplo notas, descripciones, comentarios, etc.

Length: largo del atributo.

Decimals: Número de posiciones decimales.

Picture - Formato del campo que permite una visualización en la entrada y salida, por ejemplo: en campos caracter @! significa que todos los caracteres se aceptan y despliegan en mayúsculas.

Value Range: Rango de valores válidos para el atributo.
Por ejemplo: La siguiente definición:

1:20 30: significa que el valor debe estar entre 1 y 20 o ser mayor que 30
1 2 3 4 significa que el valor debe ser 1 o 2 o 3 o 4
'S' 'N' significa que el valor debe ser 'S' o 'N'

Dominios

Es común cuando estamos definiendo la base de datos, tener atributos que comparten definiciones similares, y que no se puede establecer ninguna relación directa entre ellos. Por ejemplo, es común almacenar todos los nombres en atributos de tipo caracter y largo 25. El uso de dominios permite usar definiciones de atributos genéricos. Por ejemplo en la transacción de proveedores tenemos el nombre (PrvNom) y mas adelante vamos a definir el nombre del analista, entonces podemos definir un dominio *Nombres* de tipo caracter con largo 25 y asociarlo a estos atributos. Por lo tanto si en el futuro cambia la definición de este dominio, los cambios serán propagados automáticamente a los atributos que pertenecen a él.

Atributos Clave

También es necesario definir cual es el atributo o conjunto de atributos que identifican a la transacción (es decir que los valores de los atributos identificadores son únicos), esto se hace poniendo un * al final del o los atributos:

PrvCod*
PrvNom
PrvDir

así se indica que no existen dos proveedores con el mismo Código de Proveedor.

Es importante notar que el concepto de identificador se refiere a la unicidad (si puede haber o no dos proveedores con el mismo número) y no a como se debe acceder al archivo donde se almacenan los proveedores.
Reglas para los identificadores:

- Toda transacción debe tener un identificador.
- Los identificadores tienen valor desde un principio (por ejemplo cuando se crea un proveedor nuevo se debe saber cual será su *PrvCod*)
- No cambian de valor. Por ejemplo al proveedor con código 123 no se lo puede cambiar para 234.

En los casos en los cuales no se puede determinar un identificador se debe optar por crear un atributo artificial (no existente en la realidad) y cuyo valor es asignado automáticamente por el sistema.

Relación entre la estructura y el modelo de datos

GENEXUS utiliza la estructura de la transacción para definir cual es el modelo de datos que se necesita. En particular de la estructura anterior ***GENEXUS*** infiere:

- No existen dos Proveedores con el mismo PrvCod.
- Para **CADA** PrvCod existe solo **UN** valor de PrvNom y de PrvDir.

y con esta información va construyendo el modelo de datos. En este caso a la transacción de Proveedores se le asociara la Tabla:

Tabla: Proveedo

Atributos:
*PrvCod** (con * se indica clave primaria)
PrvNom
PrvDir

Indices:
IPROVEED (*PrvCod*) Clave Primaria

el nombre de la tabla y el del índice son asignados automáticamente, pero luego pueden ser modificados por el analista).
Diremos que la transacción de Proveedores tiene asociada la tabla PROVEEDO en el entendido que cuando se ingresen (o modifiquen, etc.) datos en la transacción estos serán almacenados en la tabla PROVEEDO.
Otras transacciones simples de nuestro ejemplo son:

Analista de compra:

*AnlNro**	Numero de analista
AnlNom	Nombre del analista

Artículos:

*ArtCod**	Código de articulo
ArtDsc	Descripción del articulo
ArtCnt	Cantidad en Stock
ArtFchUltCmp	Fecha ultima compra
ArtPrcUltCmp	Precio ultima compra
ArtUnidad	Unidad del articulo
ArtSize	Tamaño
ArtDisponible	Disponibilidad

Que tendrán asociadas las tablas ANALISTA y ARTICULO respectivamente.

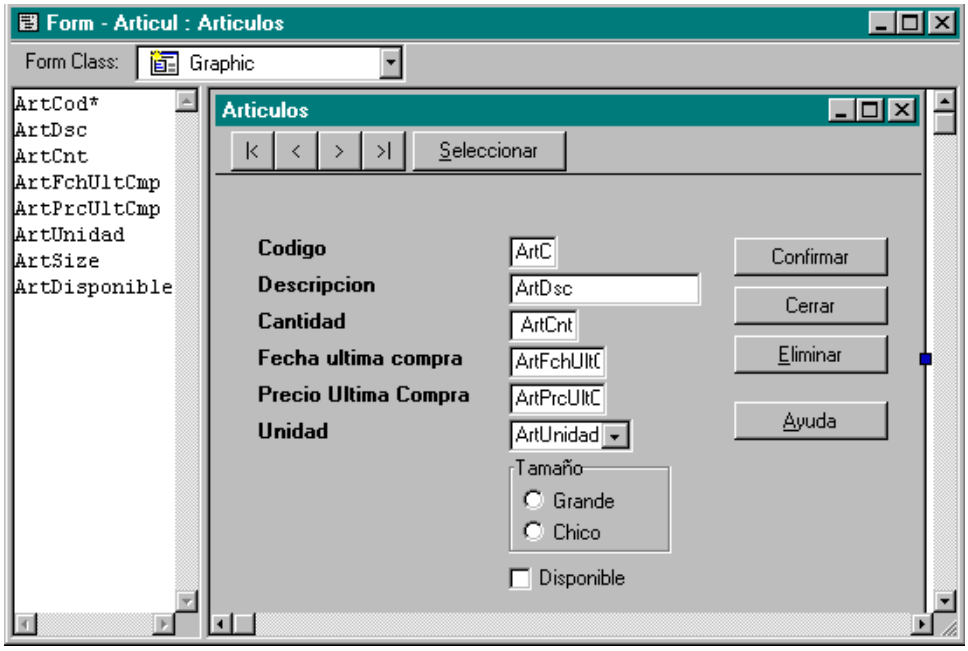
Tabla: ANALISTA

*AnlNro**
AnlNom

Tabla: ARTICULO

*ArtCod**
ArtDsc
ArtCnt
ArtFchUltCmp
ArtPrcUltCmp
ArtUnidad
ArtSize
ArtDisponible

Veamos ahora la definición del form de artículos:



Los atributos *ArtUnidad*, *ArtDisponible* y *ArtSize* no aparecen en el form como los atributos convencionales (de edit). *ArtUnidad* se definió como un **Combo Box**, *ArtSize* como un **Radio Buttom** y *ArtDisponible* como un **Check Box**.

Tipos de controles

Edit

Normalmente los atributos tienen un rango de valores muy grande, (por ejemplo: un nombre, un precio, etc). En estos casos se le permite al usuario entrar el valor del atributo y el sistema se encarga de validarlo. A estos tipos de controles se los llama 'Edit Box'. En el ejemplo, Artcod, ArtDsc, etc.
Sin embargo existen atributos que tienen un rango de valores muy pequeño y que pueden ser desplegados de antemano para que el usuario seleccione uno. De esta forma controlamos que ingresen solo valores válidos. Estos tipos de controles son los que veremos a continuación.

Check Box

Es usado para aquellos atributos que tienen solo dos posibles valores True o False (como en nuestro ejemplo para señalar si existe disponibilidad del artículo). Existe

una única descripción (Disponible) y en caso que este campo este seleccionado el valor será True y en caso contrario será False.

Radio Buttom

Los 'Radio Buttom', en cambio, pueden tener mas de dos valores. Todos los valores se despliegan en el form (en realidad se despliegan sus descripciones, el valor que se almacena es manejado internamente) y solo se puede seleccionar uno. En el ejemplo el tamaño del articulo lo definimos como un Radio Buttom.

Combo Box

Es generalmente usado para aquellos atributos que tienen un rango grande de valores, y que con un Radio Buttom no seria muy practico el manejo. Se despliega un campo de tipo Edit y presionando un botón que se encuentra a la derecha del campo se despliega una lista con todos los valores validos. No es recomendable usar este campo como listas de selección de atributos que tienen valores que no pueden ser determinados a priori, (que son leídos de una tabla). Para estos casos se usan los Dynamic Combobox.

Dynamic Combobox

Un dynamic combobox es un tipo de control similar al combo box, la diferencia es que los valores posibles se leen de una tabla de la base de datos. La forma de operación es similar a la del combo box, solo que los valores desplegados son descripciones leídas de una determinada tabla.

List Box

Este tipo de control tiene asociada una colección de ítems. Cada ítem tiene asociado un par <valor,descripción>. Existe la posibilidad de cargar la colección de ítems tanto en diseño como en runtime. El control da la posibilidad de seleccionar un solo ítem a la vez. El atributo o variable toma el valor en el momento que se selecciona el ítem. La selección se realiza dando click con el mouse en un ítem o con las flechas del teclado.

Dynamic List Box

Este tipo de control tene asociada una colección de ítems. Cada ítem tiene asociado un par <valor,descripción>.

La colección de ítems se carga en runtime desde una tabla de la base de datos. También es posible agregar ítems en forma manual en runtime.
En tiempo de diseño se asocian dos atributos al Dynamic List Box, uno al valor que tendrá el ítem y el otro a la descripción que éste tomará. Ambos atributos deben pertenecer a la misma tabla.
En tiempo de especificación se determina la tabla desde la cual se traerán los valores y las descripciones.

Definición de la transacción de pedidos

Consideremos ahora la transacción de **Pedidos**. El formulario preexistente de pedidos es:

```
Pedido    : 1456              Fecha: 02/01/92
Analista  :   21 Pedro Gomez
Proveedor :  125 ABC Inc.

Código Descripción Cantidad Precio  Valor

 321   Aspirinas     100      120   12000
 567   Flogene       120       50    6000
                                    ------
                               Total 18000
```

Los atributos que integran la transacción son (dejando momentáneamente de lado la información de los Artículos del pedido):

*PedNro**	Numero del pedido
PedFch	Fecha del pedido
PrvCod	
PrvNom	
AnlNro	
AnlNom	
PedTot	Total del pedido

en donde *PedNro* es el identificador del mismo.
Esta transacción tiene algunas características interesantes a destacar: tanto *PrvCod* y *PrvNom*, como *AnlNro* y *AnlNom* son atributos que están presentes en otras transacciones. De esta manera estamos indicando que existe una relación entre los

Proveedores y los Pedidos y también entre los Analistas y los Pedidos, en particular aquí estamos indicando que un Pedido solo tiene UN Proveedor y solo UN Analista.
Se tiene así que la forma de indicar a ***GENEXUS*** la relación entre las distintas transacciones se base en los nombres de los atributos.

Reglas para nombres de atributos

Se debe poner el mismo nombre al mismo atributo en todas las transacciones en que se encuentre, a no ser que ello no sea posible (es el caso de Subtipos que se detallara mas adelante). Por ejemplo al nombre del proveedor se le llama *PrvNom* tanto en la transacción de Proveedores como en la de Pedidos.
Se debe poner nombres distintos a atributos conceptualmente distintos, aunque tengan dominios iguales. Por ejemplo el nombre del proveedor y el nombre del analista tienen el mismo dominio (son del tipo Character de largo 30), pero se refieren a datos diferentes, por lo tanto se deben llamar *PrvNom* y *AnlNom*.

Integridad referencial en las transacciones

Otra característica a destacar es que, cuando se define la estructura de una transacción NO se esta describiendo la estructura de una tabla y SI los datos que se necesitan en la pantalla o en las reglas. Por ejemplo, que en la estructura anterior figure el *PrvNom* no quiere decir que este se encuentre en la tabla de Pedidos, simplemente indica que se necesita tener el *PrvNom* en la pantalla de Pedidos.
La tabla asociada a el cabezal del Pedido será:

Tabla: PEDIDOS

*PedNro**
PedFch
PrvCod
AnlNro
PedTot

Índices:

IPEDIDOS (*PedNro*)	Clave Primaria
IPEDIDO1 (*PrvCod*)	Clave Extranjera
IPEDIDO2 (*AnlNro*)	Clave Extranjera

Observar que *PrvNom* no se encuentra en la tabla PEDIDOS (diremos que fue *'normalizado'*). Y que existe una relación de integridad entre la tabla PROVEEDO,

que tiene a *PrvCod* como clave primaria, y la PEDIDOS que lo tiene como clave extranjera. La relación, que llamaremos de integridad referencial, es:

- Para insertar un registro en la tabla PEDIDOS debe existir el *PrvCod* correspondiente en la tabla PROVEEDO.
- Cuando se elimina un registro de la tabla PROVEEDO no debe haber registros con el *PrvCod* a eliminar en la tabla PEDIDOS.

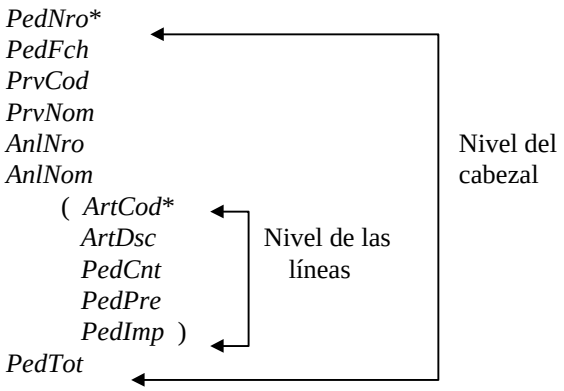
Estos controles de integridad son generados automáticamente por **GENEXUS** y, por ejemplo, en la transacción de Pedidos cuando se digita el *PrvCod* se valida que este exista en la tabla PROVEEDO e incluso se genera una subrutina que permite visualizar los proveedores existentes y seleccionar el proveedor asociado al pedido.

Niveles de una transacción

Volviendo al ejemplo, para terminar de diseñar la transacción de Pedidos se debe definir la información sobre los Artículos del pedido. Sin embargo no es posible definir la estructura de la siguiente manera:

*PedNro**	
PedFch	
PrvCod	
PrvNom	
AnlNro	
AnlNom	
ArtCod	
ArtDsc	
PedCnt	Cantidad pedida
PedPre	Precio pedido
PedImp	Valor del articulo
PedTot	

porque esto significaría que para cada pedido existe solo UN articulo, lo que no se corresponde con el formulario. La estructura correcta es:



donde el identificador es *PedNro* y para cada numero de pedido existe solo una fecha, un código y nombre de proveedor, un número y nombre del analista y un solo total, pero muchos Artículos, cantidades pedidas, precios y importes de la línea .
Los paréntesis indican que para cada Pedido existen muchos Artículos. Cada grupo de atributos que se encuentra encerrado por paréntesis diremos que pertenece a un **NIVEL** de la transacción.
Cabe notar que el primer nivel queda implícito y no necesita un juego de paréntesis. Así, la transacción de Pedidos tiene dos niveles: *PedNro*, *PedFch*, *PrvCod*, *PrvNom*, *AnlNro*, *AnlNom* y *PedTot* pertenecen al primer nivel y *ArtCod*, *ArtDsc*, *PedCnt*, *PedPre* y *PedImp* pertenecen al segundo nivel.
Una transacción puede tener varios niveles y ellos pueden estar anidados o ser paralelos entre si. Por ejemplo si el Pedido tiene varias fechas de entrega:

```
PedNro*
PedFch
PrvCod
PrvNom
AnlNro
AnlNom
  ( ArtCod*
    ArtDsc
    PedCnt
    PedPre
    PedImp  )
  ( PedFchEnt*          Fecha de entrega
    PedImpPag  )        Importe a pagar
PedTot
```

Con esta estructura se define que un Pedido tiene muchos Artículos y muchas Entregas, pero no hay relación directa entre ambas (a no ser el hecho de pertenecer al mismo Pedido). Si por el contrario las fechas de entrega son por articulo (cada articulo tiene sus fecha de entrega particulares), la estructura correcta es:

*PedNro**
PedFch
PrvCod
PrvNom
AnlNro
AnlNom
(*ArtCod**
ArtDsc
PedCnt
PedPre
PedImp
(*PedFchEnt**
PedImpPag))
PedTot

Así como la transacción tiene un identificador, cada nivel de la misma también debe tener uno. Por ejemplo en el Pedido tenemos que el identificador del segundo nivel es *ArtCod*. Con esto se indica que dentro de cada Pedido no existen dos líneas del Pedido con el mismo *ArtCod*, y por lo tanto el siguiente Pedido no es valido:

```
Pedido    : 1456              Fecha: 02/01/92
Analista  :   21 Pedro Gomez
Proveedor :  125 ABC Inc.

Código Descripción Cantidad Precio  Valor

 321   Aspirinas     100      120   12000
 321   Aspirinas     120       50    6000
                                    ------
                               Total  18000
```

porque existen dos líneas del Pedido con el mismo *ArtCod*.

Aquí también se deben tener en cuenta los criterios sobre identificadores que se mencionaron previamente. En particular, es común definir atributos artificiales cuando no hay identificadores naturales, por ejemplo, para el caso en que si puede haber varias veces el mismo articulo en el pedido, se debe definir el Pedido como:

*PedNro**
PedFch
PrvCod
PrvNom
AnlNro
AnlNom
(*PedLinNro**
ArtCod
ArtDsc
PedCnt
PedPre
PedImp)
PedTot

en donde el *PedLinNro* lo puede digitar el usuario o se puede incluir una regla para que lo haga automáticamente la transacción.

Cada nivel de la transacción tiene una tabla asociada:

Tabla: PEDIDOS
*PedNro**
PedFch
PrvCod
AnlNro
PedTot

Tabla: PEDIDOS1
PedNro*
PedLinNro*
ArtCod
PedCnt
PedPre
PedImp

La clave primaria de las tablas asociadas a los niveles subordinados es la concatenación de los identificadores de los niveles superiores (*PedNro*) mas los identificadores del nivel (*PedLinNro*).

La elección de los identificadores es una tarea importante y que puede simplificar o complicar la implementación de un sistema. Supongamos por ejemplo, que en el Pedido no se admite que haya dos líneas con el mismo *ArtCod*. La forma natural de implementar esto es definiendo a *ArtCod* como identificador del segundo nivel. Ahora bien, si por alguna razón se definió a *PedLinNro* como el identificador, ya no tendremos ese control automático y se debe escribir un Procedimiento para realizarlo.
En otras palabras, cuanto mas reglas de la realidad se puedan reflejar en la estructura, menor será la cantidad de procesos que se debe implementar, ganando así en simplicidad y flexibilidad.

Tipos de relación entre los objetos

Muchas veces cuando los usuarios están describiendo la aplicación, es común preguntarles que tipo de relación existe entre los distintos objetos. Por ejemplo cual es la relación entre los proveedores y los pedidos:

> Un proveedor tiene muchos pedidos (relación 1-N).
> Un pedido tiene un solo proveedor (relación N-1).

Como ya vimos, la relación 1-N se puede definir con la estructura:

*PrvCod**
PrvNom
....
(*PedNro**
....
)

Y la relación N-1 como:

*PedNro**
....
PrvCod
PrvNom
....

Vemos que generalmente para cada relación N-1 habrá una relación simétrica 1-N. En estos casos se debe definir la estructura N-1 y no la 1-N.
Otro caso muy común son las relaciones M-N, por ejemplo entre los pedidos y los Artículos:

> Un pedido tiene muchos Artículos.

Un articulo puede estar en muchos pedidos.

En este caso las dos estructuras posibles son:

*PedNro**
PedFch
PrvCod
PrvNom
AnlNro
AnlNom
(*ArtCod**
ArtDsc
PedCnt
PedPre
PedImp)
PedTot

o:

*ArtCod**
ArtDsc
(*PedNro**
PedFch
PrvCod
PrvNom
AnlNro
AnlNom
PedTot
PedCnt
PedPre
PedImp)

Se debe definir solo una de las dos estructuras, en este caso la correcta es la primera, porque el usuario ingresa los pedidos con sus Artículos y no para cada articulo que pedidos tiene.

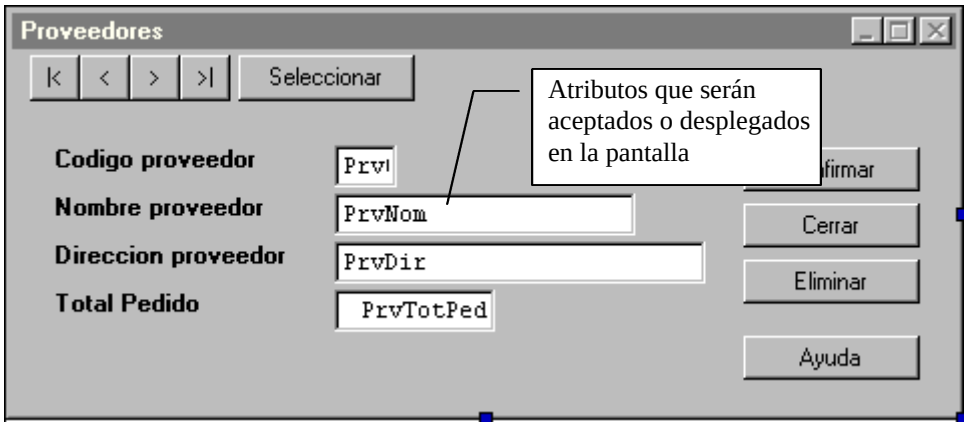
Form de una transacción

Cada transacción puede tener varios forms asociados. Inicialmente **GENEXUS** crea uno por defecto partiendo de los atributos de la misma y luego se puede modificar. Se pueden

agregar o quitar forms a través de las opciones *Edit/Add New Form* o *Edit/Select Forms.* Tambien se puede seleccionar el form con el cual se desea trabajar seleccionándolo del combo box que aparece en la toolbar.

También se puede asociar un style y aplicarlo. El style se asocia en cada objeto en las propiedades del mismo.

Para los Proveedores la pantalla por defecto (en este caso en modo gráfico) es:



Utilizando solo esta pantalla el usuario final puede crear, modificar o eliminar proveedores. Para ello la pantalla tiene un Modo, que indica cual es la operación que se esta realizando (Insertar, Modificar, etc.) y el usuario puede pasarse de un modo a otro sin tener que abandonar la pantalla.
GENEXUS puede generar dos tipos de diálogo para las transacciones, full-screen o campo a campo.

Diálogo full-screen

Este tipo de diálogo se genera para las plataformas donde se utilizan terminales no programables, por ejemplo para el IBM **AS/400**.
El funcionamiento básico del diálogo full-screen es:

1- Se aceptan todos los campos de la pantalla
2- Se realizan todas las validaciones
3- Si hubo error se muestran los mensajes de error y se vuelve al punto 1.
4- Si no hubo error se pide confirmación, si no se confirma se vuelve a 1.

5- Si se confirma la operación esta es realizada y se vuelve a 1.

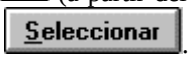
Este proceso tiene leves alteraciones según el modo (Insertar, Modificar, etc.), ya que si se encuentra en modo Insertar se aceptan los identificadores, pero si se esta en modo Modificar no, etc.
El pasaje de un modo a otro se realiza presionando teclas de función. Por ejemplo con F6 se pasa a modo Insertar, con F11 a modo Modificar, etc.

Diálogo campo a campo

Para plataformas con terminales programables, como es el caso de PCs o LANs, ***GENEXUS*** genera las transacciones con una interfaz campo a campo.
En este diálogo, cada vez que un campo es digitado inmediatamente se controla su validez.
También trabaja con modos, pero con la diferencia que el modo es inferido automáticamente por el sistema. Por ejemplo cuando el usuario digita el *PrvCod*, si este existe se pasa automáticamente a modo Modificar y si no existe se pasa a modo Insertar.

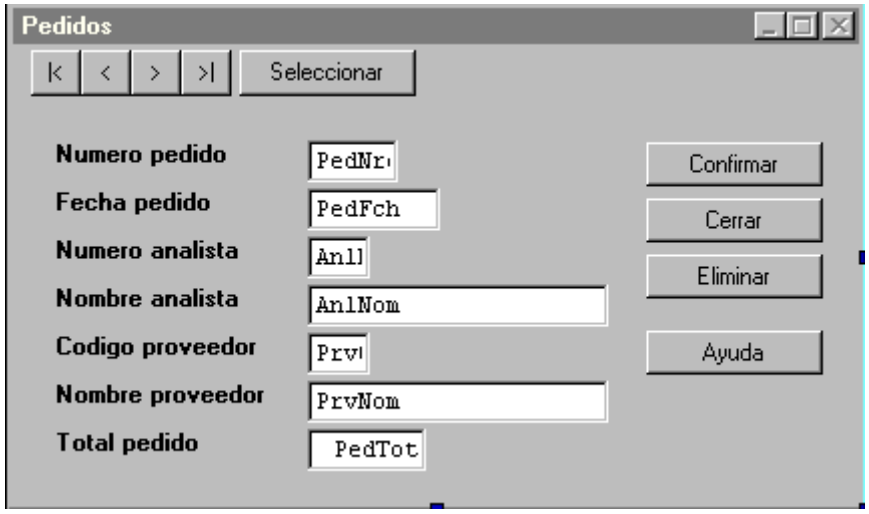
Barra o Botones de Menú

En ambos tipos de diálogo se encuentra disponible una Barra de Menú que tiene las opciones para poder visualizar los distintos datos.

Por ejemplo, se puede elegir ver (para luego modificar) el primer proveedor  , el siguiente  (a partir del que se esta mostrando en pantalla) o poder elegir uno en particular  .

Atributos de entrada y atributos de salida

Consideremos nuevamente la transacción de Pedidos pero sin sus Artículos. La pantalla por defecto es:

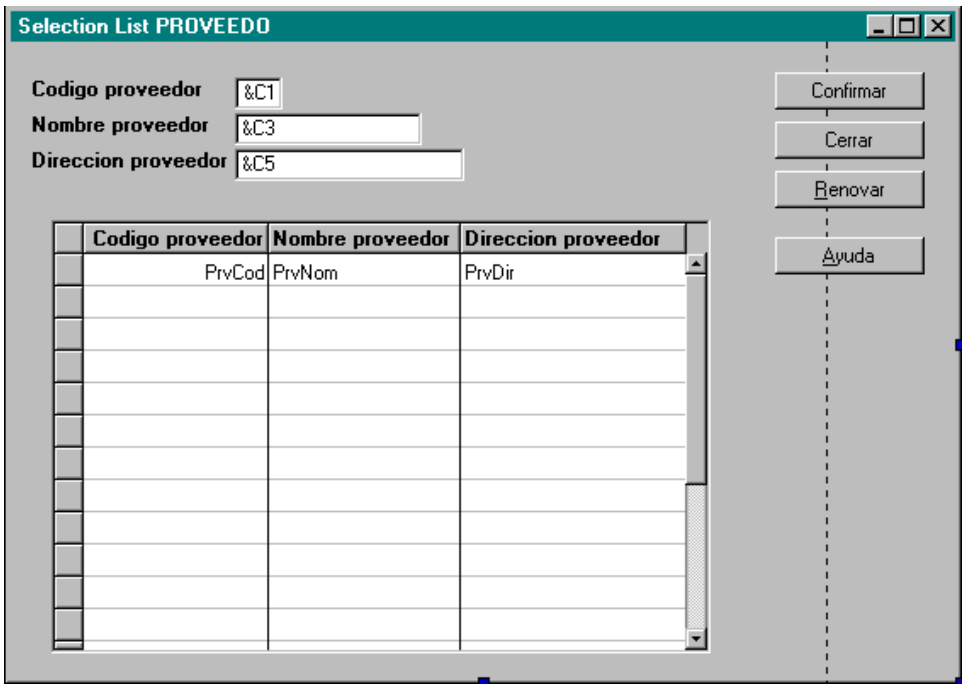


Vemos que hay algunos atributos que deben ser digitados (son atributos de entrada), por ejemplo *PedNro* y *PrvCod* y otros que no, como ser *PrvNom* (son atributos de salida). ***GENEXUS*** automáticamente determina que atributos son aceptados y de cuales solo se muestra su valor, siguiendo las reglas:

- Solo pueden ser aceptados los atributos que se encuentran en la tabla asociada al nivel (en este caso son los atributos de la tabla PEDIDOS, *PedNro*, *PedFch*, *PrvCod*, *AnlNro* y *PedTot*).
- Los atributos definidos como fórmulas no son aceptados.
- En modo Modificar no se aceptan los identificadores.
- Los atributos que tienen una regla de Noaccept no son aceptados.

Facilidad de Prompt

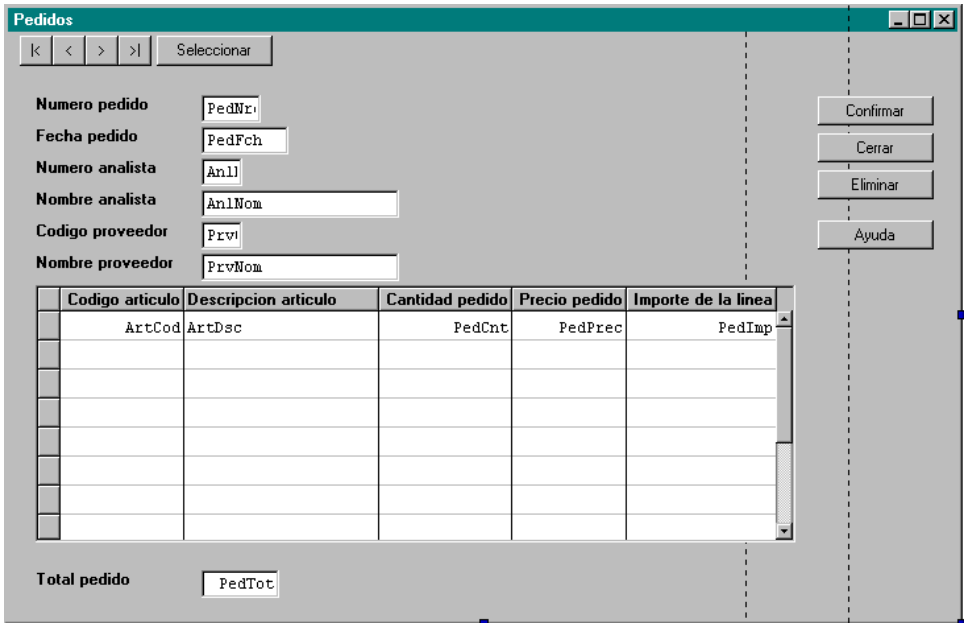
Para los atributos que están involucrados en la integridad referencial de la tabla base del nivel se genera la facilidad de Prompt que permite visualizar cual es el conjunto de valores validos para esos atributos. Por ejemplo, en la transacción de Pedidos presionando una tecla especial se puede visualizar cuales son todos los proveedores, con la siguiente pantalla:



en donde se puede seleccionar el proveedor. Observar que en la primera parte de la pantalla se permite digitar el *PrvCod* ,el *PrvNom*, o el *PrvDir* para poder posicionarse en la lista de proveedores. Esta lista es un work panel de GeneXus que puede ser modificado como cualquier otro objeto.

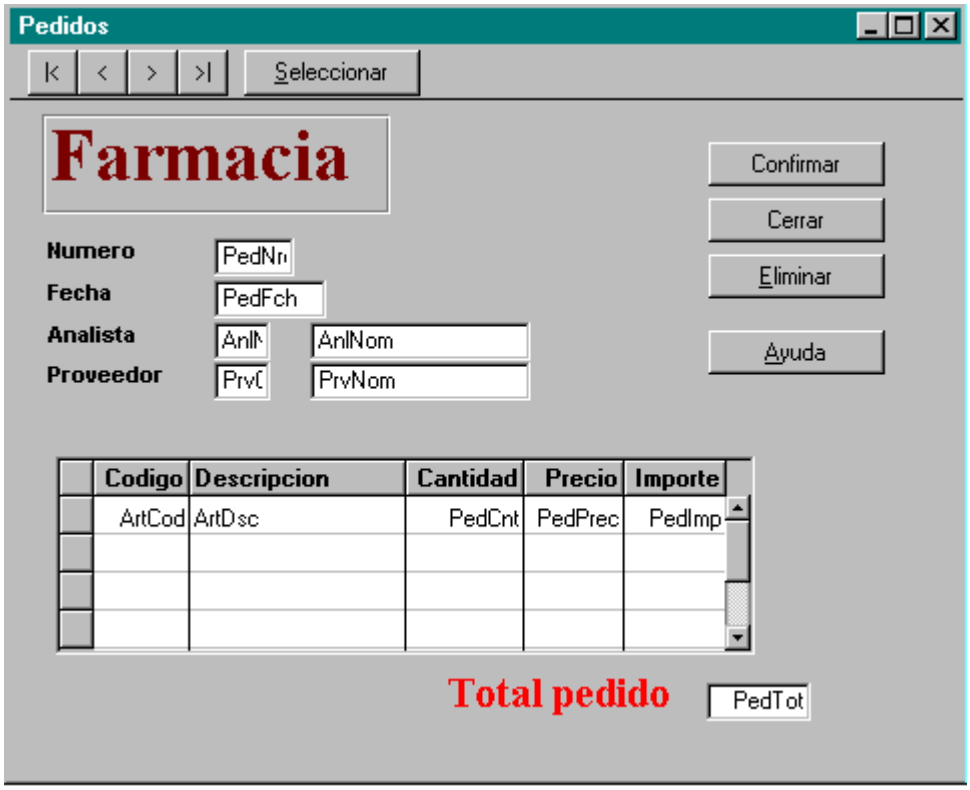
Diálogo para transacciones con varios niveles

Para el caso de transacciones con varios niveles se tiene un proceso de diálogo por nivel. Es decir se realiza la validación y la confirmación de los datos para cada uno de los niveles.
Por ejemplo para la transacción de Pedidos (considerando ahora los Artículos del mismo) la pantalla por defecto es:



En ésta se aceptan primero los datos de entrada del primer nivel (los del cabezal del pedido: Numero, Fecha, Proveedor y Analista), posteriormente se validan y se pide confirmación y luego se realiza el mismo proceso para cada una de las líneas del pedido.

Cuando se genera el form por defecto se diseña un subfile para el último nivel de la transacción, en el ejemplo las líneas del pedido.
Trabajando con el form editor se puede transformar esta pantalla en la siguiente, que simula mejor el formulario de pedidos:



Form Editor

Los objetos como las Transacciones o los Work Panels, tienen uno o varios Formularios asociados. A los efectos de permitir modificar los Formularios que ***GENEXUS*** automáticamente crea por defecto, se provee un Form Editor. Este Form Editor es igual para todos los objetos que tienen Formularios.
El Formulario está formado por un marco de ventana (frame) cuyo título es la descripción de la transacción. Dentro de ella figurarán los distintos tipos de controles de edición del Formulario.

Form Edition Controls

En un Formulario se definen 'controles'. Un control es una área del Formulario que tiene una forma y un comportamiento determinado. Existen los siguientes tipos de controles:

Atributo. Se utiliza para representar atributos o variables.

Texto. Todo fragmento de texto fijo en la pantalla debe colocarse utilizando uno o más controles de este tipo.

Línea. Con este control se dibujan líneas horizontales o verticales.

Recuadro. Permite definir recuadros de distintos tamaños y formas.

SubFile. Permite definir SubFiles. La definición de los subfiles se vera mas adelante en el capitulo de Work Panels.

Botón. Permite definir Botones (también llamados Command Buttons).

Bitmap. Permite definir bitmaps estáticos en la pantalla.
Cada control tiene una serie de propiedades (ancho de línea, color, color de fondo, font, etc.), cuyos valores pueden fijarse en forma independiente.

Tab. Permite definir varios controles dentro de otro. Un Tab control tiene una o varias Paginas y cada Pagina tiene un Title y un area util donde se ponen los controles de esa pagina. Solo una de las paginas puede estar activa a la vez y es la que se puede ver, del resto solo se vera su titulo. Esto es util para cuando se quiere dividir los datos ingresados en la transaccion en distintos grupos de modo que las pantallas queden diseñadas de forma amigable al usuario (transacciones con muchos atributos).

Paleta de herramientas

Mientras se está editando un Formulario, está disponible la paleta de herramientas, en la forma de una ventana adicional, con los botones que representan los tipos de controles disponibles.

Uso de las Herramientas

Sobre los objetos seleccionados con el puntero vamos a poder realizar varias operaciones:

- Cambiar el tamaño y forma de un control.
- Edición de propiedades.
- Move Forward, Move Back, Move to Front y Move to Back. Estas operaciones nos van a permitir cambiar la capa en que se encuentra un control. Cada control se encuentra en una capa distinta del Formulario, por lo que algunos están "más arriba" que otros. Cuando dos objetos se superpongan, el de más arriba será visible totalmente, mientras que el otro sólo en aquellos puntos en que no se encuentre "tapado".
- Move, Copy y Delete.

Toolbar

El Form Editor Toolbar nos permite ubicar los controles en el form, y copiar el tamaño de otros controles ya definidos:

Toolbar

 Alinear a la izquierda.

 Alinear a la derecha.

 Alinear al borde superior.

 Alinear al borde inferior.

 Centrar verticalmente.

 Centrar horizontalmente.

 Separar horizontalmente.

 Separar verticalmente.

 Igualar ancho.

 Igualar altura.

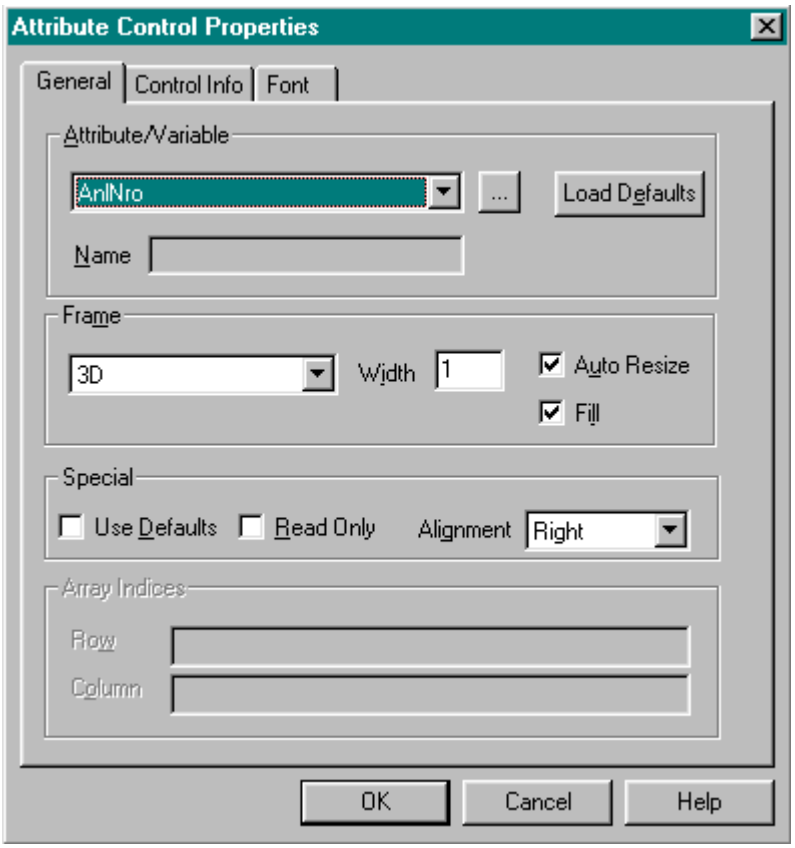
 Igualar tamaño.

 Mostrar Grid

Propiedades de los controles

Vamos a ver las principales propiedades de cada unos de los controles que podemos insertar en un form.

Atributo



Name. En esta opción se permite ingresar un nombre al control que se esta editando. Este nombre será usado luego para asociarle propiedades o eventos al control.

Array Indices. En el caso de utilizarse variables que sean matrices, se habilitan estas opciones, donde se pueden indicar expresiones que definan la fila y la columna.

Frame. Se puede indicar que el atributo esté rodeado por un marco. Es posible indicar el tipo: None, Single o 3D; el ancho de la/s línea/s (Width), si se pinta dentro de él o no (Fill) y si el tamaño y forma deben determinarse a partir del atributo (Auto Resize).

Font. Permite seleccionar el font que se utilizará para el atributo en la pantalla.

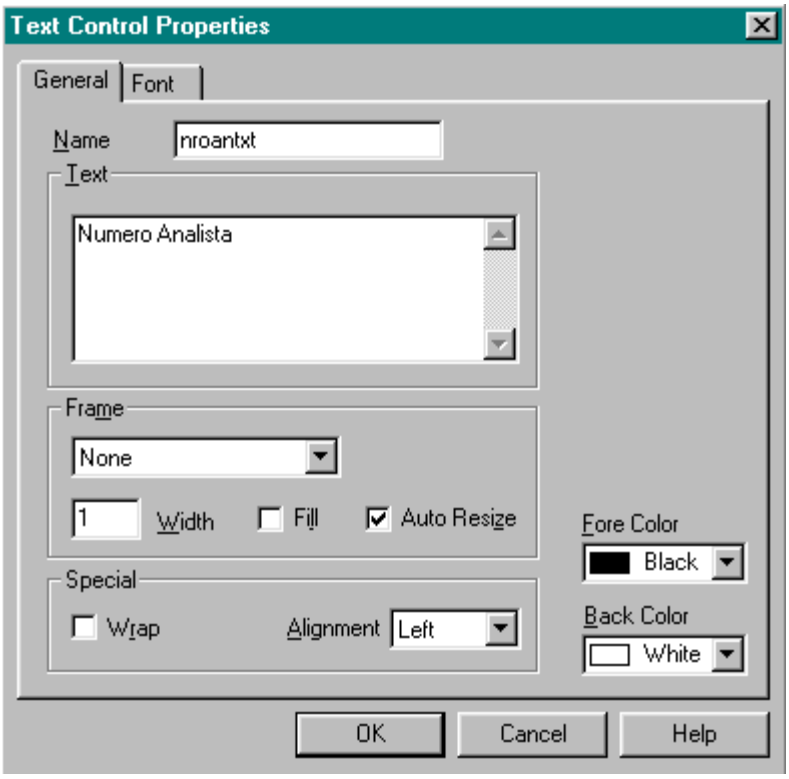
Control Info

Control Type. En los Formularios generados el atributo podrá asociarse a distintos tipos de controles Windows. Se puede seleccionar uno de los siguientes: Combo Box, Dynamic Combobox, Radio Button, Edit, List Box, Dynamic List Box y Check Box. El botón de Setup permite definir características propias a cada tipo de control. Dentro

Fore Color. Este botón es para seleccionar el color con el que se desplegará el valor del atributo y la/s línea/s del frame, si tuviera.

Back Color. Con este botón se puede seleccionar el color con el que se pinta el área asignada al atributo en la pantalla. Sólo tiene efecto si está presente la propiedad Fill.

Texto

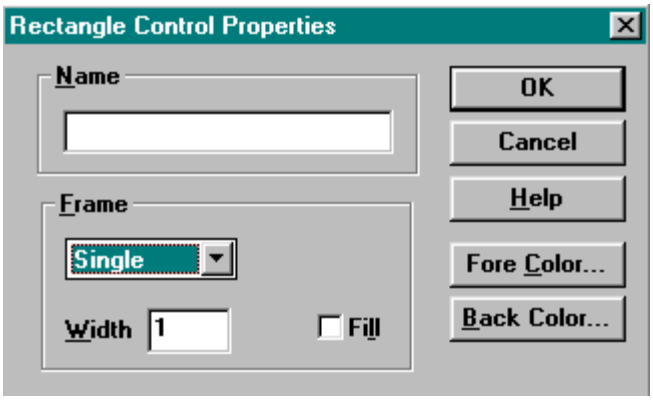


Text. Indica el texto que se quiere insertar en la pantalla. Puede consistir de una o más líneas.

Alignment. El texto puede estar justificado a la izquierda (Left), derecha (Right) o centrado (Center).

Resto de las propiedades. Ver control de Atributo.

Recuadro



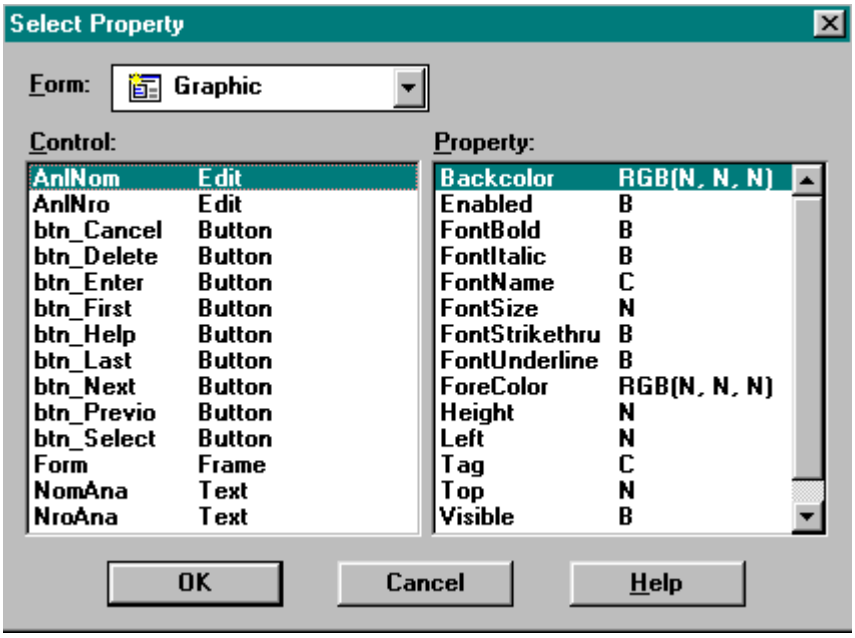
Del combo box se selecciona el tipo del borde del recuadro: single, none (sin borde) o 3D.
Para el resto de las propiedades ver la explicación en el control de Atributo.

Línea

Utiliza el mismo diálogo de edición de propiedades con la salvedad de que la opción Fill aparece deshabilitada.

Editor de Propiedades, Métodos y Eventos

A los controles (que tienen nombre asociado) que usamos en los forms se le pueden asociar propiedades, métodos o eventos, por ejemplo: cambiarle el font a un texto, hacer un texto visible o invisible, deshabilitar un botón, etc. El tipo de propiedades/eventos que se pueden asociar a un control depende del tipo de control. Para acceder al dialogo donde poder asociar propiedades a los controles se usa la opción de menú *Insert/Property* (o *Insert/Event*) cuando se esta editando los eventos, rules o subrutinas de la transacción. El dialogo que aparece es el siguiente:



Fórmulas

Se dice que un atributo es una fórmula si su valor se puede calcular a partir del valor de otros atributos.

En el ejemplo el atributo *PedImp* de la transacción de Pedidos se puede calcular a partir de la *PedCnt* y el *PedPre*:

*PedImp = PedCnt * PedPre*

En cada transacción se puede definir que atributos son fórmulas, pero esta definición es global, es decir toda vez que se necesite el atributo se calcula la fórmula, tanto en transacciones como en los otros objetos **GENEXUS** (reportes, Paneles de Trabajo, etc.).

Existen distintos tipos de fórmulas:

- Horizontales
- Verticales
- De agregación/selección

Fórmulas Horizontales

Una fórmula horizontal se define como una o varias expresiones aritméticas condicionales. Por ejemplo:

*Descuento = PedTot * 0.10 if PedTot > 100 .and.*
PedTot <= 1000;

*= PedTot * 0.20 if PedTot > 1000;*

= 0 OTHERWISE;

En donde el Descuento será del 10% si el total del pedido esta entre 100 y 1000 y del 20% si es mayor y en cualquier otro caso no hay descuento (en las fórmulas con condiciones es saludable definir siempre el OTHERWISE).

La expresión puede utilizar los operadores aritméticos (+. -, *, /, ^) y funciones (por ejemplo 'today()' que devuelve la fecha del día), y para los casos en donde el calculo es muy complicado se puede llamar a una rutina que lo realice en vez de usar formulas:

Descuento = udf('Pdesc', *PedTot*);

En donde 'udf' viene de User Defined Function. 'Pdesc' es el nombre de la rutina llamada, que recibe como parámetro a *PedTot* y devuelve Descuento. Esta rutina puede ser una rutina externa a ***GENEXUS*** o escrita con ***GENEXUS*** utilizando el objeto Procedure.
El importe del pedido también es una fórmula horizontal:

*PedImp = PedCnt * PedPre*

Vemos que la forma de definirla involucra a dos atributos (*PedCnt* y *PedPre*), pero no se indica en que tablas de encuentran. ***GENEXUS*** se encarga de buscar la manera de relacionar *PedCnt* y *PedPre* para poder calcular la fórmula (en este caso es muy fácil porque *PedCnt* y *PedPre* se encuentran en la misma tabla). En los casos en que no es posible relacionar a los atributos ***GENEXUS*** da un mensaje de 'fórmula invalida'.
Ahora bien, cuales son los atributos que pueden estar involucrados?. La respuesta es: en una fórmula horizontal todos los atributos de los cuales depende la fórmula deben estar en la misma **tabla extendida** que la fórmula (ver el concepto de tabla extendida en el anexo sobre Bases de Datos Relacionales).

Cuando se define una fórmula horizontal **GENEXUS** 'sabe' cual es la tabla extendida y controla que todos los atributos utilizados se encuentren en ella.

Fórmulas Verticales

Consideremos ahora el Total del pedido (*PedTot*), este se debe calcular sumando los Importes (*PedImp*) de cada una de las líneas del Pedido. Esta fórmula se expresa como:

PedTot = **SUM**(*PedImp*)

Y es un caso de fórmula vertical, que se define como el tipo de fórmula en que los atributos involucrados no se encuentran dentro de la misma tabla extendida que la fórmula. En el caso de *PedTot*, si observamos el modelo de datos (usando el diagrama de tablas de **GENEXUS**):

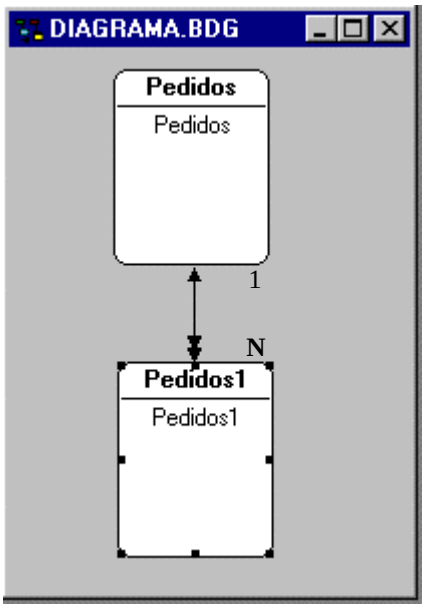
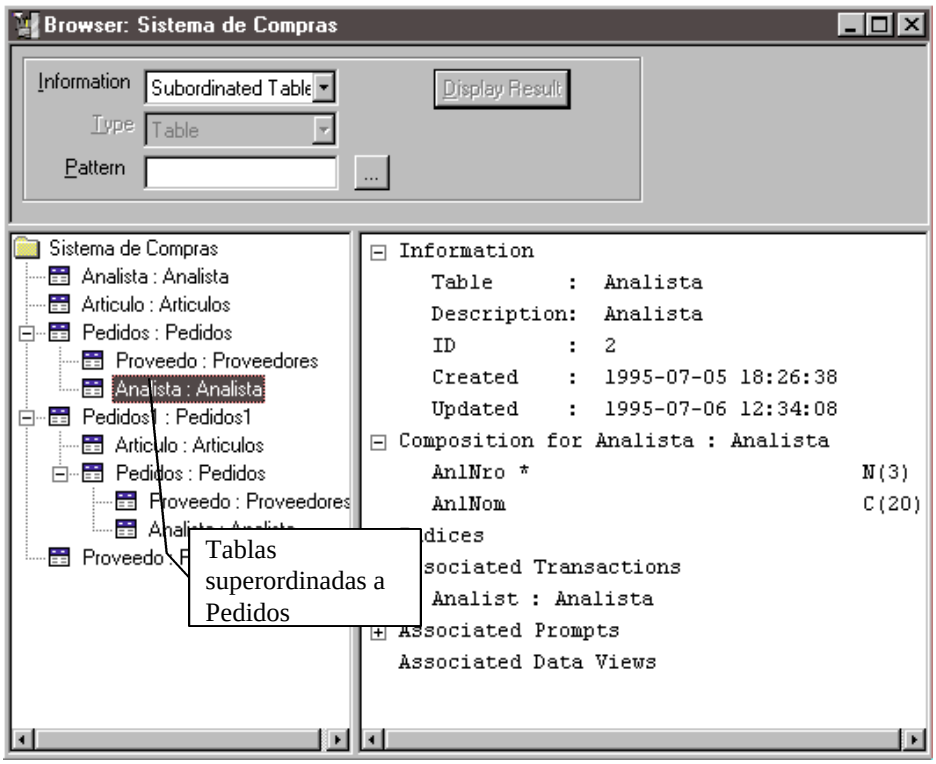


Tabla: PEDIDOS
*PedNro**
PedFch
PrvCod
AnlNro
PedTot

Tabla: PEDIDOS1
*PedNro**
*PedLinNro**
ArtCod
PedCnt
PedPre
PedImp

Tenemos que *PedTot* esta en la tabla PEDIDOS y *PedImp* en la *PEDIDOS1* que es una tabla subordinada a la PEDIDOS.
Podemos utilizar también el Database Browser de **GENEXUS** para determinar que tablas están subordinadas y superordinadas a la de pedidos (eligiendo la opción Superordinated o Subordinated del dialogo del browser):



Además de ver que tablas tiene subordinadas y superordinadas, podemos consultar la estructura de una tabla, que índices tiene (composición de los índices), fórmulas, etc.

Existen dos operadores para fórmulas verticales: **SUM** que suma todos los datos y **COUNT** que cuenta cuantos datos hay.

Fórmulas y Redundancia

En base a los criterios de normalización y dado que por definición una fórmula siempre puede ser calculada, no es necesario que la misma este almacenada y basta con recalcularla cada vez que sea necesario.
Sin embargo el hecho que la fórmula no este almacenada puede ocasionar problemas de performance, debido al tiempo que pueda demorar el recálculo.
Para evitar este inconveniente se puede definir la fórmula como REDUNDANTE. En este caso la fórmula se almacena en la Base de Datos y no es necesario recalcularla

cada vez que se use. Una vez que la fórmula es definida como redundante, **GENEXUS** se encarga de agregar todas las subrutinas de mantenimiento de redundancia a todas las transacciones que utilicen esa fórmula.
Tenemos entonces que la definición de fórmulas redundantes implica un balance de performance por un lado y almacenamiento y complejidad de los programas generados en otro, que el analista debe decidir. Si bien en teoría esta decisión puede ser bastante difícil de tomar, en la practica vemos que la mayor parte de las fórmulas que se definen redundantes son las verticales (mas adelante veremos que hay una forma muy fácil de hacerlo) y pocas veces las horizontales.
La razón de esto es que el calculo de las fórmulas verticales pueden insumir un número indeterminado de lecturas. Por ejemplo para calcular el *PedTot* se deben leer todas las líneas del pedido, que pueden ser una cantidad bastante grande.

Fórmulas de Fórmulas

Una fórmula se calcula a partir del valor de otros atributos. Dichos atributos pueden estar almacenados o ser otras fórmulas. Por ejemplo si en la transacción de Proveedores definimos:

PrvTotPed = SUM(*PedTot*) Total pedido del proveedor

tenemos que para calcularlo se necesita:

PedTot = SUM(*PedImp*)

que a su vez necesita:

PedImp = *PedCnt* * *PedPre*

Para fórmulas de fórmulas **GENEXUS** determina cual es el orden de evaluación correspondiente:

PedImp = *PedCnt* * *PedPre*
PedTot = SUM(*PedImp*)
PrvTotPed = SUM(*PedTot*)

Reglas

Según el Análisis Orientado a Objetos, para cada objeto del sistema se debe definir cual es su estructura y cual su comportamiento. En el caso de las transacciones ya vimos la definición de la estructura. Para definir el comportamiento se usan las REGLAS.
Usando reglas se definen valores por defecto, controles, etc. Veremos algunas de las reglas:

Default

Se utiliza para definir los valores por defecto de algunos atributos, por ejemplo:

```
default( PedFch, today() );
```

El funcionamiento del default varia según el tipo de diálogo (full-screen o campo a campo). En el diálogo full-screen se asigna el valor por defecto si el usuario no digito nada en el campo. En el diálogo campo a campo primero se asigna el valor por defecto y luego se permite modificarlo.

Error

Es la regla para definir los controles que deben cumplir los datos, por ejemplo si queremos que el precio del articulo pedido sea siempre mayor que 100:

```
error( 'El precio debe ser mayor que 100') if PedPre <= 100 ;
```

Cuando se cumple la condición (PedPre <= 100) se despliega el mensaje ('El precio debe ser mayor que 100') en pantalla y no se permite continuar hasta que el usuario ingrese un valor correcto o abandone la transacción.

Una utilización relativamente común es incluir una regla de error() para prohibir alguna de las modalidades de la transacción:

```
error( 'No se permite eliminar Pedidos' ) if delete;
```

Con esta regla se prohibe que el usuario entre en el modo eliminación y así se prohibe la eliminación de Pedidos.

Asignación

Dentro de las reglas de la transacción se permiten definir asignaciones a atributos, dicha Asignación implica una actualización en la tabla base del nivel o en alguna de

las tablas que pertenecen a la tabla extendida del nivel. Veamos un ejemplo, en la transacción de Artículos:

Artículos:
*ArtCod**
ArtDsc
ArtCnt
ArtFchUltCmp
ArtPrcUltCmp

En el atributo *ArtPrcUltCmp* se desea almacenar cual fue el precio del último pedido realizado para ese articulo, y en que fecha fue realizado en *ArtFchUltCmp*. Esto se realiza definiendo en la transacción de Pedidos las siguientes reglas:

ArtPrcUltCmp = *PedPre* if insert;
ArtFchUltCmp = *PedFch* if insert;

Así cada vez que se ingrese una línea del pedido se actualiza la tabla de Artículos con la fecha y el precio correspondiente.

Existe una similitud entre fórmulas y asignaciones, incluso la sintaxis de definición es similar. La diferencia entre ambas es que una fórmula es GLOBAL, es decir vale para todos los objetos **GENEXUS** que la utilicen, mientras que una Asignación es LOCAL, vale solo para la transacción en la cual fue definida.
Cuando un atributo es fórmula este no esta almacenado (a no ser que se lo defina como redundante) y cuando es una Asignación, por ser esta local, si esta almacenado.

Add y Subtract

Las asignaciones que vimos en la sección anterior eran relativamente fáciles, pero existen casos mas sofisticados. Por ejemplo en la misma transacción de Artículos tenemos el atributo *ArtCnt* en donde se quiere mantener cual es el stock que tenemos de cada articulo.
Sin duda la transacción de Pedidos debe modificar ese valor, porque cada pedido nuevo aumenta el stock. También existirá alguna otra transacción que hace disminuir el stock, que podría ser por ejemplo una transacción de Facturación que se realiza cuando se vende el articulo. Como esta transacción esta fuera del alcance del proyecto solo estudiaremos la actualización del stock relacionada con la transacción de Pedidos.

En dicha transacción se debe actualizar *ArtCnt*, esto se podría hacer con la Asignación:

ArtCnt = *ArtCnt* + *PedCnt*;

Pero no debemos olvidar que en la misma transacción se permite crear, modificar o eliminar un pedido, y la asignación anterior solo es correcta si se esta creando uno nuevo, ya que si por ejemplo se está eliminando, la asignación correcta es:

ArtCnt = *ArtCnt* - *PedCnt*;

Entonces, para actualizar *ArtCnt* correctamente se necesitaría también considerar los casos de modificación y eliminación, pero para evitar todo esto ***GENEXUS*** posee la regla add() que lo hace automáticamente:

add(*PedCnt, ArtCnt*);

Con esta regla si se esta insertando una línea del pedido se suma la *PedCnt* a *ArtCnt*, si se esta eliminando se resta y si se esta modificando le resta el valor anterior de *PedCnt* (que se define como old(*PedCnt*)) y le suma el nuevo valor.
Existe una dualidad entre la fórmula vertical SUM y la regla ADD, mas concretamente se puede decir que un SUM redundante es equivalente a la regla ADD. Por ejemplo el *PedTot* se puede definir como:

PedTot = SUM(*PedImp*) y redundante
o
add(*PedImp*, *PedTot*);

Dado que es común que las fórmulas verticales tengan que estar redundantes para tener buena performance se recomienda el uso de la regla ADD y no la fórmula SUM.
La regla SUBTRACT es equivalente pero realiza las operaciones contrarias, es decir cuando se esta insertando resta, en caso de eliminación suma, etc.

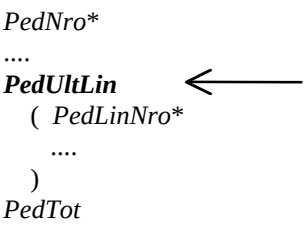
Serial

Esta regla es útil cuando se quiere asignar automáticamente valores a algún identificador de la transacción. Por ejemplo en la transacción de Pedidos tenemos el *PedLinNro* y si queremos que este número sea asignado automáticamente por el sistema se puede usar la regla:

serial(*PedLinNro*, *PedUltLin*, 1);

En donde el primer parámetro define cual es el atributo que se esta numerando, el segundo indica cual es el atributo que tiene el último valor asignado (aquí se trata del último número de línea asignado) y el tercer parámetro el incremento (en este caso se incrementa de a uno). El atributo *PedUltLin* (Ultima línea asignada) debe ser incluido

en la estructura de la transacción en el nivel de *PedNro* (un pedido solo tiene UNA Ultima línea asignada):



De esta manera para cada nueva línea del pedido el programa asigna automáticamente su número.
En el diálogo campo a campo (donde la modalidad era inferida automáticamente), se debe digitar un valor inexistente en *PedLinNro* (usualmente 0) y el programa asigna el valor correspondiente.
En el diálogo full-screen el valor se asigna cuando el usuario presiona Enter en modo Insertar.

Orden de evaluación

La definición de reglas es una forma DECLARATIVA de definir el comportamiento de la transacción. El orden en el cual fueron definidas no necesariamente indica que ese sea el orden en que se encuentran en el programa generado.
GENEXUS se encarga de determinar el orden correcto según las dependencias de atributos existentes.

Veamos un ejemplo:

Supongamos que definimos en la transacción de Artículos un nuevo atributo: *ArtCntMax*, del mismo tipo que *ArtCnt*. Este atributo indicará el stock máximo que se desea tener de ese artículo.
Cuando se ingresa un pedido debemos emitir un mensaje si se sobrepasa el stock máximo de un artículo. Para ello definimos las siguientes reglas en la transacción de pedidos:

msg('Se sobrepasa el stock máximo del artículo')
if *ArtCnt* > *ArtCntMax*;

add(*PedCnt*, *ArtCnt*);

El orden de evaluación será:

add(*PedCnt*, *ArtCnt*);

msg('Se sobrepasa el stock máximo del artículo')
if *ArtCnt* > *ArtCntMax*;

Ya que la regla add() modifica *ArtCnt* y la regla msg() utiliza ese atributo.
Esto implica que en la regla msg() se debe preguntar por *ArtCnt* mayor que *ArtCntMax* y no por *ArtCnt + PedCnt mayor que ArtCntMax*.

Cada vez que se especifica una transacción se puede pedir la opción -**"Detailed Navigation"**- que lista cual es el orden de evaluación. En este listado se explícita en que orden se generarán (usualmente decimos 'se disparan'), no solo las reglas, sino también las fórmulas y lecturas de tablas.

Call y función After

Otra regla muy usada es CALL, con la cual se permite llamar a otro programa. Este puede ser cualquier otro objeto ***GENEXUS***, por ejemplo, de una transacción se puede llamar a un reporte o a otra transacción, o un programa externo.
En el caso de la regla Call es necesario precisar en que MOMENTO se debe disparar el Call. Por ejemplo, si en la transacción de Pedidos se quiere llamar a un reporte que imprime el pedido que se acaba de ingresar, se utiliza la regla:

call('RIMPREC', *PedNro*) if after(trn) ;

En donde 'RIMPREC' es el programa llamado y *PedNro* es el parámetro que se le pasa. Al tener after(trn), el Call se realizará después de haber entrado todo el Pedido.
El uso de after() no es privativo de la regla Call y puede ser utilizado en otras reglas, como ser error o Asignación.
Los after() existentes son:

Confirm

La regla se dispara después de haber confirmado los datos del nivel pero antes de haber realizado la actualización. Se usa para algunos casos de numeración automática cuando no se quiere utilizar la regla serial para evitar problemas de control de concurrencia.

Insert | Update | Delete

Se dispara después de haber insertado, actualizado o eliminado el registro de la tabla base del nivel. Se usa fundamentalmente para llamar a procesos que realizan actualizaciones.

Level

Se dispara después de haber entrado todos los datos de un nivel. Se usa muchas veces para controles de totales, por ejemplo, si cuando se entra el cabezal del Pedido se digita un total (llamémosle *PedTotDig*) y se quiere verificar que este sea igual que el total calculado la regla es:

> error('El total digitado no cierra con el calculado') if *PedTotDig* <> *PedTot* .and. **after(level(*ArtCod*))**;

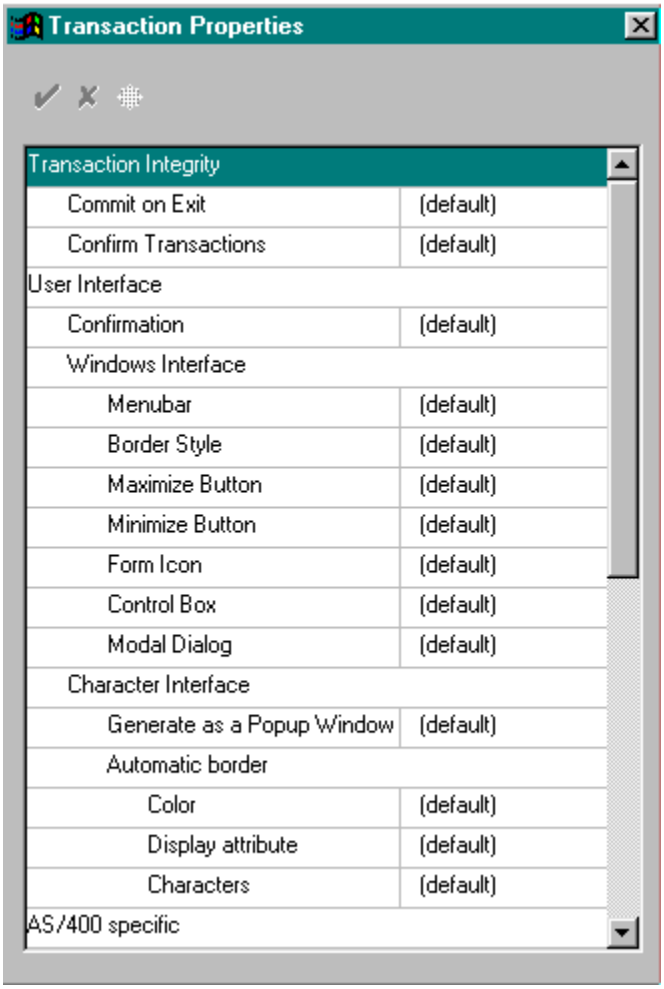
En donde el control recién se realizará cuando se terminaron de entrar todas las líneas del pedido.

Trn

Se dispara después de haber entrado todos los datos de una transacción. Se usa fundamentalmente para llamar a programas que imprimen los datos o a otras transacciones. En ambientes con integridad transaccional esta regla se dispara luego que se realiza el commit de la transacción.

Propiedades

En el editor de propiedades de las transacciones se pueden definir reglas de comportamiento general. A continuación vemos la pantalla para la edición de las mismas:



Por ejemplo vamos a setear propiedades que estén asociadas con el manejo de la integridad transaccional y con la interface. Podemos indicar si deseamos que la transacción haga un commit al final o que no deseamos que se pidan confirmaciones cada vez que se actualiza un nivel, etc.

DISEÑO DE REPORTES

Con Reportes se definen los procesos no interactivos de extracción de datos. Usualmente un Reporte es un listado que se envía a la impresora, aunque también puede ser visualizado en pantalla.

Es un proceso no interactivo, porque primero se extraen los datos y luego se muestran, no habiendo interacción con el usuario durante el proceso de extracción, y es solo de extracción ya que no se pueden actualizar los datos leídos. Para consultas interactivas se utilizan los Paneles de Trabajo y para los procesos de actualización los Procedimientos.

Para cada Reporte se debe definir:

Información
Datos generales del Reporte, por ejemplo nombre, descripción, etc.

Layout
Así como las Transacciones tienen una pantalla los Reportes tienen un layout de la salida.

Condiciones
Qué condiciones deben cumplir los datos para ser impresos.

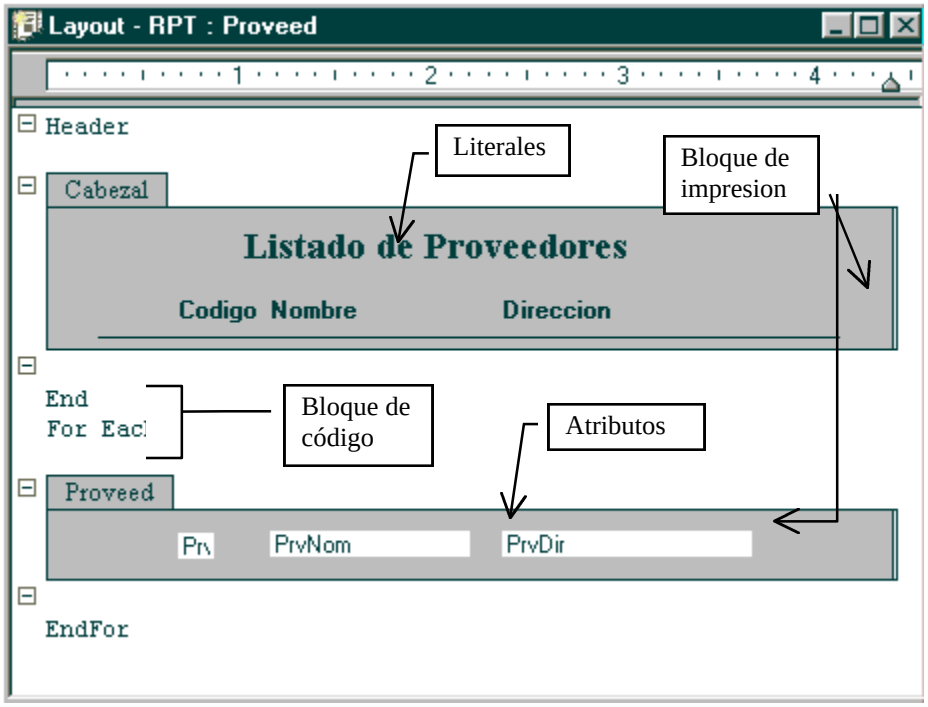
Reglas - Propiedades
Definen aspectos generales del Reporte.

Ayuda
Texto para la ayuda a los usuarios en el uso del Reporte.

Documentación
Texto técnico para ser utilizado en la documentación del sistema.

Layout

En el Layout se define simultáneamente la estructura de la navegación (que datos se quieren listar y en que orden) y el formato de la salida. Para ello se utiliza un lenguaje procedural muy simple que describiremos brevemente a continuación.
Comencemos por un Reporte para listar todos los proveedores:

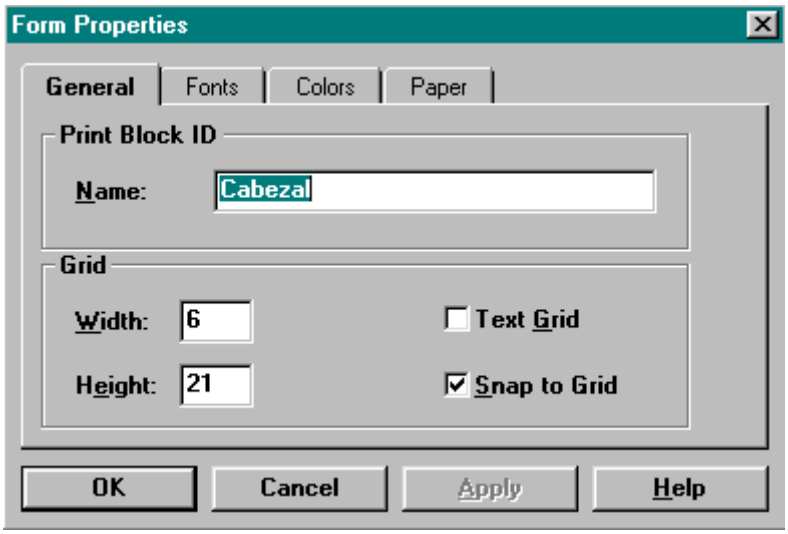


En donde tenemos los comandos Header, End, For each y Endfor. Para describir más fácilmente el funcionamiento de estos comandos veamos el resultado de ejecutar el programa generado correspondiente:

Listado de Proveedores

Codigo	Nombre	Direccion
125	ABC Inc.	18 de Julio
126	GXXXX Corp.	Artigas

La definición de reportes se divide en uno o varios bloques. Estos bloques pueden ser de código o de impresión . En los bloques de impresión se diseña la salida que posteriormente será impresa. Cada bloque de impresión puede tener un conjunto de controles (atributos, variables, textos, etc.). Podemos ver un bloque de impresión como un pedazo de papel. Los bloques de impresión (o Print Blocks) tienen asociadas ciertas propiedades, que pueden seteadas desde un dialogo que se abre al hacer doble-click sobre el bloque de impresión, como ser: los colores a usar, el tipo de papel, el tipo de letra, etc. El dialogo es el siguiente



Para el diseño de los bloques de impresión tenemos disponible una paleta de herramientas similar a las transacciones.

La cual permite insertar atributos, textos, líneas, recuadros, bitmaps y nuevos bloques de impresión. Los controles son definidos de la misma forma en que se hace en el editor del form de las Transacciones (tipo de letra, color, tamaño, etc.).

Todos los bloques de impresión que se encuentren entre los comandos **HEADER** y END son las líneas que se imprimen en el cabezal de la pagina. También existe el comando **FOOTER** que permite imprimir un pie de página en cada hoja.

El comando **FOR EACH** indica que se impriman todos los bloques de impresión que se encuentren entre el FOR EACH y el ENDFOR para cada uno de los datos que se encuentren en la base de datos. En este caso:

PrvCod	PrvNom	PrvDir

se debe imprimir para cada uno de los proveedores.

Comando FOR EACH

Este es, quizás, el comando más importante en los Reportes ya que toda la definición del acceso a la base de datos y la estructura del reporte se realizan solo con este comando.
Usando el FOR EACH se define la información que se va a leer de la base de datos, pero la forma de hacerlo se basa en nombrar los atributos a utilizar y NUNCA se especifica de que tablas se deben obtener, ni que índices se deben utilizar. Por lo tanto, con este comando se define QUE atributos se necesitan y en qué ORDEN se van a recuperar, y ***GENEXUS*** se encarga de encontrar COMO hacerlo. Obviamente esto no siempre es posible, y en estos casos ***GENEXUS*** da una serie de mensajes de error indicando porque no se pueden relacionar los atributos involucrados.
La razón por la cual no se hace referencia al modelo físico de datos es para que la especificación del Reporte sea del mas alto nivel posible, de tal manera que ante cambios en la base de datos la especificación del mismo se mantenga valida la mayor parte de las veces.
El acceso a la base de datos queda especificado por los atributos que son utilizados dentro de un grupo FOR EACH - ENDFOR. Para ese conjunto de atributos ***GENEXUS*** buscará la tabla extendida que los contenga (el concepto de tabla extendida es muy importante en este comando y sugerimos repasar su definición en el Anexo sobre Modelos de Datos Relacionales).
Por ejemplo, en el Reporte anterior dentro del FOR EACH - ENDFOR se utilizan los atributos *PrvCod*, *PrvNom y PrvDir*, ***GENEXUS*** 'sabe' que estos atributos se encuentran en la tabla PROVEEDO y por lo tanto el comando:

For each
PrvCod | PrvNom | PrvDir
Endfor

es interpretado por ***GENEXUS*** como:

For each record in table **PROVEEDO**
PrvCod | PrvNom | PrvDir
Endfor

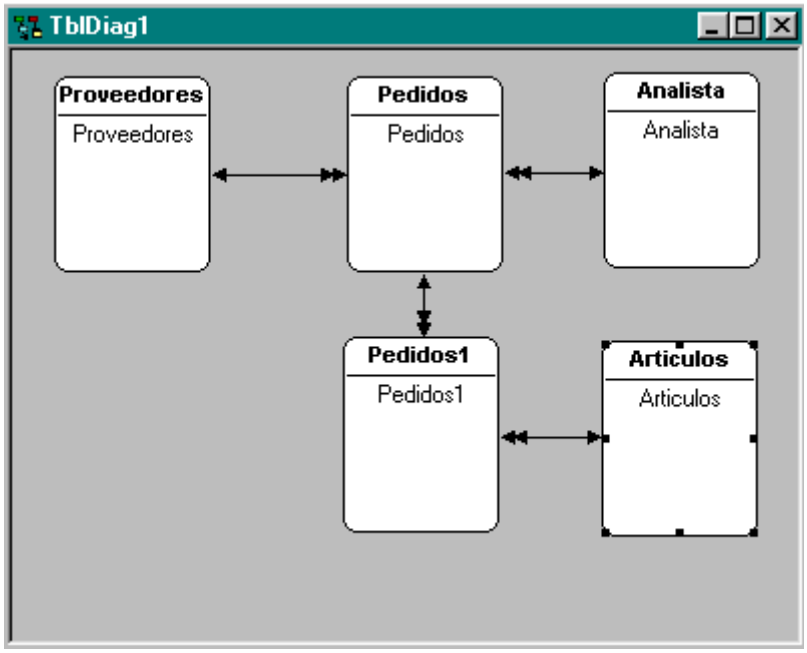
Para clarificar el concepto hagamos un Reporte que involucre datos de varias tablas. Por ejemplo, listar el cabezal de todos los pedidos:

Listado de Pedidos Fecha: 09/08/92

Número	**Fecha**	**Nombre Proveedor**	**Nombre Analista**
321	02/02/92	ABC Inc.	Juan Gomez
654	03/03/92	GXXXX Corp.	Juan Gomez
987	03/03/92	ABC Inc.	Pedro Perez

El layout para este listado es:

Si recordamos el modelo de datos definido en el capítulo Diseño de Transacciones, el Diagrama de Bachman es:



En el listado anterior se requieren datos de las tablas PROVEEDO (*PrvNom*), ANALISTA (*AnlNom*) y PEDIDOS (*PedNro* y *PedFch*). La tabla extendida de PEDIDOS contiene a todos estos atributos y por lo tanto el comando FOR EACH se interpretará como:

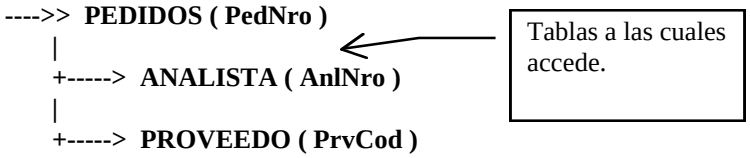
For each record in table **PEDIDOS**
  Find the corresponding *PrvNom* in table **PROVEEDO**
  Find the corresponding *AnlNom* in table **ANALISTA**

PedNro	PedFch	PrvNom	AnlNom

Endfor

Cuando se especifica un Reporte, ***GENEXUS*** brinda un listado (que llamaremos Listado de **Navegación**) donde se indica cuales son las tablas que se están accediendo, en este caso el listado es:

FOR EACH PEDIDOS (Line 6)
Order : PedNro
Index : IPEDIDOS
Navigation filters:
Start from:
First record
Loop while:
Not end of table



END FOR

Donde la flecha doble (--->>) indica cual es la tabla base de la tabla extendida y la flecha simple (---->) las tablas N-1 de la misma. Una interpretación muy práctica de este diagrama es: para la tabla de la flecha doble se leen muchos registros y para cada tabla con flecha simple se lee un solo registro por cada registro leído en la tabla con flecha doble.

Orden

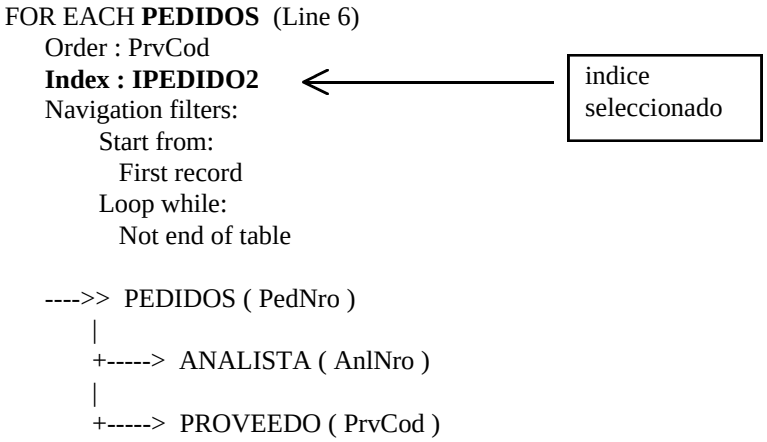
En los Reportes anteriores no se tuvo en cuenta en que orden se deben listar los datos. Cuando no se indica el orden, ***GENEXUS*** busca un orden que favorezca la performance del Reporte.
Para los casos en donde se quiere definir el orden de los datos de forma explícita se utiliza la cláusula ORDER en el comando FOR EACH, por ejemplo, para listar los pedidos ordenados por *PrvCod*:

For each ORDER ***PrvCod***

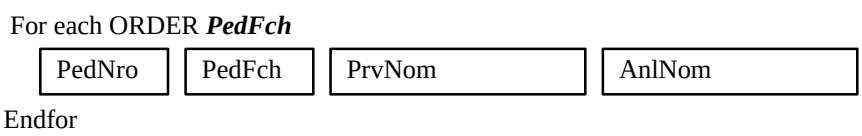
PedNro	PedFch	PrvNom	AnlNom

Endfor

En este caso el índice a utilizar será el IPEDIDO2 (*PrvCod*) de la tabla PEDIDOS, que fue definido automáticamente por ***GENEXUS***. La navegación para este reporte es la siguiente:



Si al listado anterior lo queremos ordenar por *PedFch*:



Como no hay ningún índice definido en la PEDIDOS para *PedFch*, ***GENEXUS*** creará un índice por *PedFch* cada vez que se ejecute el Reporte y lo eliminará al finalizar el mismo y en el Listado de Navegación indicará que esta creando un 'índice temporario'.

Navegación*:*
A temporary index will be created for PedFch on table PEDIDOS

Obviamente este proceso es más lento si lo comparamos con el listado anterior en donde había un índice definido. Dada esta situación el analista debe tomar una decisión:
Crear un índice del usuario. En este caso el Reporte será bastante más eficiente, pero se debe mantener un índice más (lo que implica mayor almacenamiento y mayor procesamiento en las transacciones para mantener actualizado el índice).
Dejar el Reporte con el índice temporario.

No es posible recomendar, a priori, cual de las dos soluciones es la mejor, por lo tanto se debe estudiar caso por caso la solución particular, siendo fundamental la frecuencia con que se ejecutará el Reporte. De cualquier manera si al comienzo no se definió el índice y posteriormente se decide cambiar y definirlo, alcanza con regenerar el Reporte (sin modificar nada del mismo) y éste pasará a utilizarlo.
Los criterios de ordenamiento utilizado por **GENEXUS** son:
Cada grupo FOR EACH puede estar ordenado por CUALQUIER conjunto de atributos pertenecientes a la tabla base del grupo, pero no se puede ordenar por atributos que NO se encuentren en la misma. Por ejemplo el listado anterior no se puede ordenar por *PrvNom* porque la tabla base del grupo es la PEDIDOS que no tiene *PrvNom*.
Si no existe un índice para el orden pedido se crea un índice temporario que se eliminará al finalizar el Reporte. Observar que este proceso es análogo a realizar un SORT cuando se trabajaba con archivos tradicionales.

Condiciones en el FOR EACH

Para restringir los datos que se quieren listar se utiliza la cláusula WHERE. Por ejemplo, para listar sólo los pedidos de hoy:

```
For each
  WHERE PedFch = Today()
```

PedNro	PedFch	PrvNom	AnlNom

```
Endfor
```

Se puede definir varios WHERE dentro del FOR EACH:

```
For each
  WHERE PedFch = Today()
  WHERE PedNro > 100
```

PedNro	PedFch	PrvNom	AnlNom

```
Endfor
```

Que significa que se deben cumplir AMBAS condiciones, o sea que se interpreta como que ambas cláusulas están relacionadas por un AND. Si se quiere utilizar un OR (es decir se desea que se cumpla alguna de las dos condiciones) se debe utilizar un solo WHERE:

```
For each
```

WHERE *PedFch* = Today() OR *PedNro* > 100

PedNro	PedFch	PrvNom	AnlNom

Endfor

GENEXUS posee una serie de mecanismos para optimizar el Reporte en función del orden del FOR EACH y de las condiciones del mismo, pero estas se aplican sólo a las condiciones que tienen la siguiente forma:

```
WHERE <atributo> <operador> <valor>
        <código1>
WHEN NONE
        <código2>
```

Donde:

<atributo>	debe estar almacenado en la tabla base.
<operador>	debe ser = o >=.
<valor>	puede ser una constante o una variable.
<código2>	se ejecuta solo cuando **no** se entra en el For Each

En muchos casos es necesario ejecutar determinado código cuando en un For Each no se encuentra ningún registro. Para esto se usa el comando WHEN NONE. Es importante aclarar que si se incluyen For Eachs dentro del When None no se infieren Joins ni filtros de ningún tipo con respecto al For each que contiene el When None.

Cuando la condición ha sido optimizada en el Listado de Navegación aparece como **'Navigation Filter'** y cuando no, como **'Constraint'**.
Tener en cuenta que cuando la condición tiene esta forma se realizara algún tipo de optimización, pero no quiere decir que otras formas no sean válidas, al contrario, en el WHERE se pueden definir condiciones para atributos que son fórmulas, que no se encuentran en la tabla base, etc.

Determinación de la tabla extendida del grupo FOR EACH

Resulta muy útil saber cómo ***GENEXUS*** determina la tabla extendida que se debe utilizar. El criterio básico es:
Dado el conjunto de atributos que se encuentran dentro de un grupo FOR EACH - ENDFOR, ***GENEXUS*** determina la MINIMA tabla extendida que los contenga.
Consideremos nuevamente el listado:

For each

PedNro	PedFch	PrvNom	AnlNom

Endfor

Según ya vimos todos estos atributos se encuentran dentro de la tabla extendida de PEDIDOS, sin embargo, también se encuentran dentro de la tabla extendida de *PEDIDOS1* (Línea del pedido), pero la tabla extendida de PEDIDOS está contenida en la de *PEDIDOS1* (porque justamente la de *PEDIDOS1* contiene a la de PEDIDOS además de otras tablas), y por lo tanto será la elegida por **GENEXUS**.
Para que **GENEXUS** pueda determinar cual es la tabla extendida del grupo FOR EACH debe haber por lo menos un atributo que pertenezca a la tabla base. Por ejemplo, en el Reporte que estamos realizando hay dos atributos pertenecientes a la tabla base PEDIDOS (*PedNro* y *PedFch*), si ellos no estuvieran y el listado fuera:

For each

PrvNom	AnlNom

Endfor

se podría argumentar que la tabla extendida de PEDIDOS continúa siendo válida, sin embargo por razones de simplicidad **GENEXUS** en este caso indicará que no hay relación entre estos dos atributos y será necesario que se utilice algún atributo de la tabla PEDIDOS para poder generar el Reporte.
También puede darse el caso que exista más de una tabla extendida mínima que contenga a los atributos del grupo FOR EACH - ENDFOR. Ante esta ambigüedad **GENEXUS** escoge la PRIMERA tabla extendida que cumpla con las condiciones anteriores.
Para resolver estos dos problemas (no hay ningún atributo de la tabla base y hay más de una tabla extendida mínima posible), se utiliza la cláusula DEFINED BY dentro del comando FOR EACH, por ejemplo:

For each
Defined by *PedFch*

PedNro	PedFch	PrvNom	AnlNom

Endfor

En el DEFINED BY se debe hacer referencia a por lo menos un atributo de la tabla base que se quiere.
Aún en los casos que no se dan ninguno de los problemas anteriores se recomienda el uso del DEFINED BY para Reportes más o menos complejos porque mejora bastante el tiempo de especificación del Reporte.

Reportes con varios FOR EACH

For Each anidados

Hasta ahora hemos definido reportes en donde existía un solo FOR EACH, pero es posible utilizar mas de un FOR EACH para realizar Reportes más sofisticados. Supongamos que queremos listar para cada proveedor cuales son sus pedidos, por ejemplo:

Fecha: 07/11/95

Listado de Pedidos por Proveedor

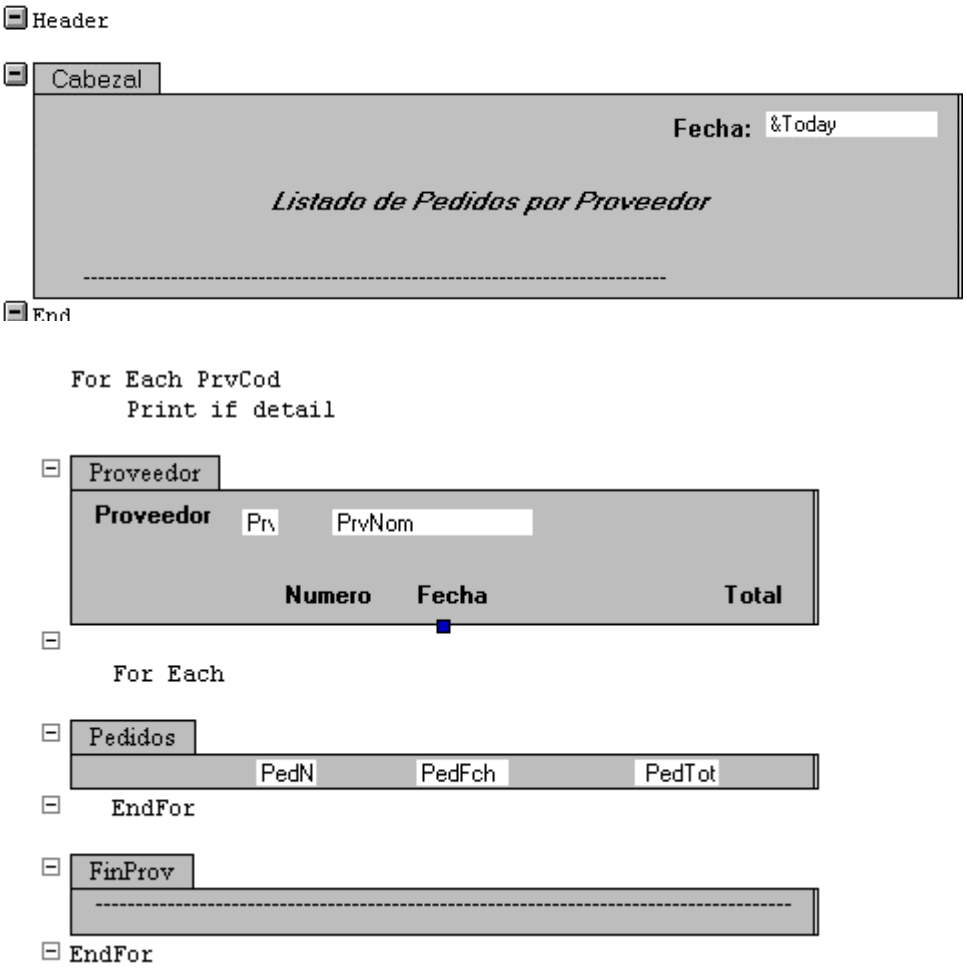
Proveedor 125 ABC Inc.

Numero	**Fecha**	**Total**
400	01/07/95	172,80
423	11/07/95	553,00

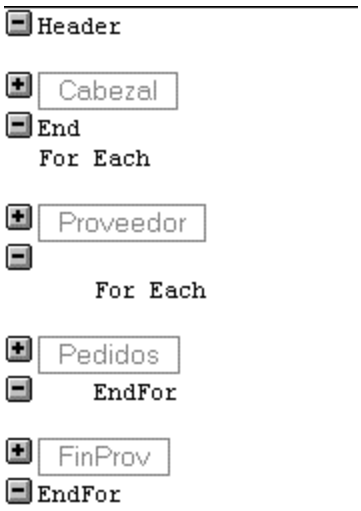
Proveedor 126 GXXXX Corp.

Numero	**Fecha**	**Total**
410	04/11/95	378,00

El layout que se debe definir es (sólo se muestran los grupos FOR EACH):



Si colapsamos todos los print blocks, queda mas claro como se definen los FOR EACH anidados.



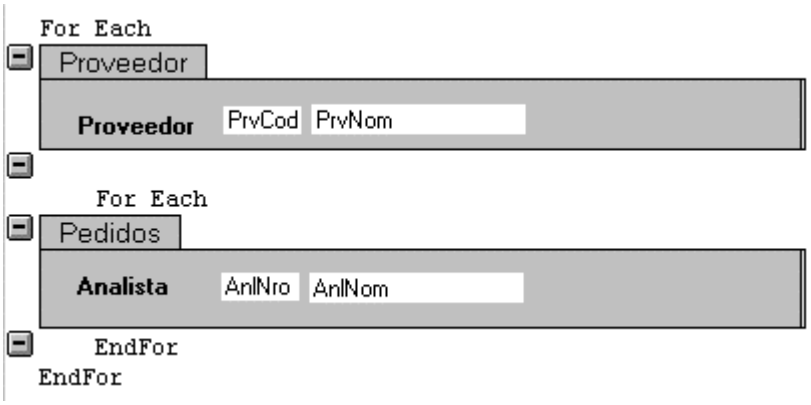
En este Reporte tenemos dos FOR EACH anidados. Si analizamos el primer FOR EACH vemos que utiliza sólo los atributos *PrvCod* y *PrvNom* y por lo tanto su tabla extendida será la de la tabla PROVEEDO, el segundo FOR EACH utiliza los atributos *PedNro* y *PedFch* y su tabla extendida será la de PEDIDOS. Pero el segundo FOR EACH se encuentra anidado respecto al primero y a su vez la tabla PEDIDOS esta subordinada a PROVEEDO (por *PrvCod*) por lo tanto ***GENEXUS*** interpretara el Reporte como:

For each record in table **PROVEEDO**
....
For each record in table **PEDIDOS**
with the same *PrvCod*
....
Endfor
....
Endfor

Y así se listarán, para cada registro en la tabla de proveedores, cuales son sus pedidos en la tabla de pedidos.
Más formalmente, cuando hay dos FOR EACH anidados, ***GENEXUS*** busca cual es la relación de subordinación existente entre la tabla base del segundo FOR EACH y la tabla extendida del primer FOR EACH (en el caso anterior la relación era que la tabla PEDIDOS estaba subordinada a la tabla PROVEEDO por *PrvCod*) y establece de esta manera la condición que deben cumplir los registros del segundo FOR EACH (en el

ejemplo: todos los registros de la tabla PEDIDOS con el mismo *PrvCod* leído en el primer FOR EACH).

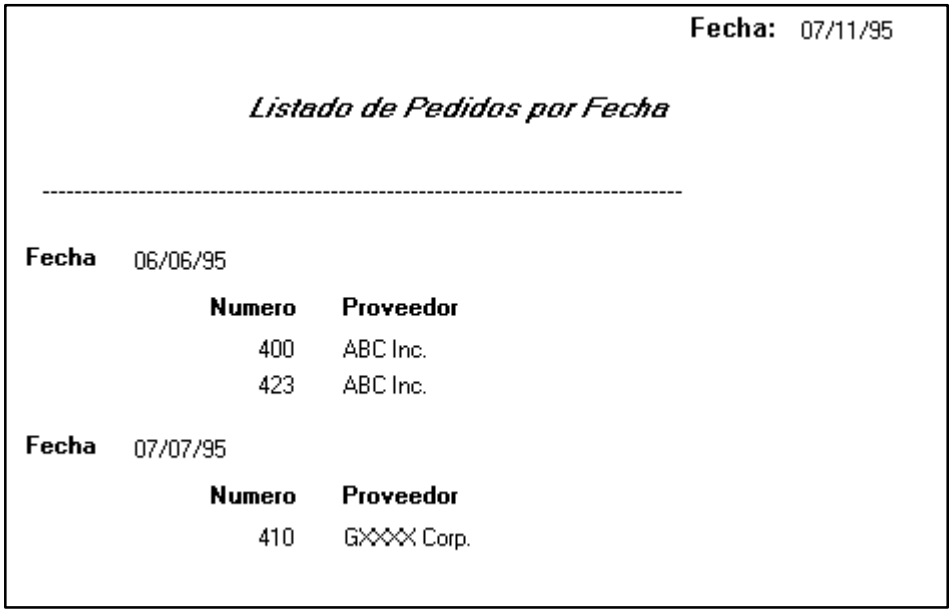
Puede darse el caso en que no exista relación entre ambos FOR EACH, por ejemplo:



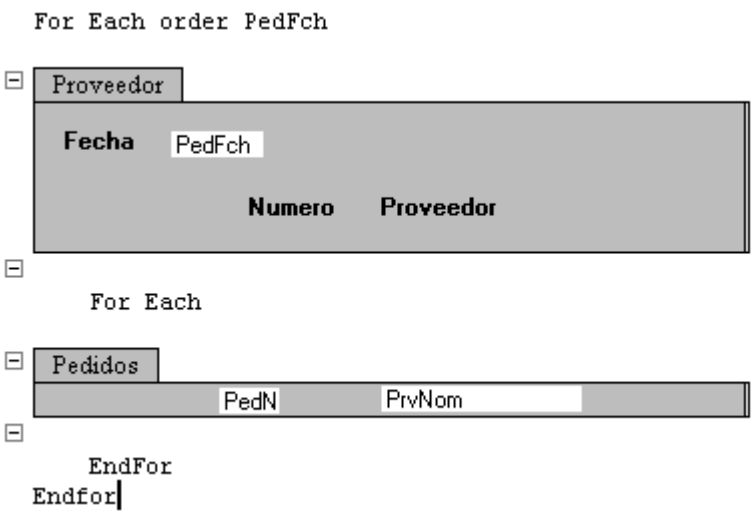
La tabla base del primer FOR EACH es la PROVEEDO (porque los atributos son *PrvCod* y *PrvNom*) y la del segundo FOR EACH es la ANALISTA (porque los atributos son *AnlNro* y *AnlNom*) y no existe relación entre estas dos tablas extendidas. En este caso ***GENEXUS*** da el mensaje 'No existe relación directa' y hará el producto cartesiano entre los dos FOR EACH, es decir, para cada registro de la tabla PROVEEDO (proveedores) se imprimirán todos los registros de la tabla ANALISTA (analistas).

Cortes de Control

Un caso particular de FOR EACH anidados son los llamados cortes de control. Supongamos que queremos listar los pedidos ordenados por fecha:



El layout es:



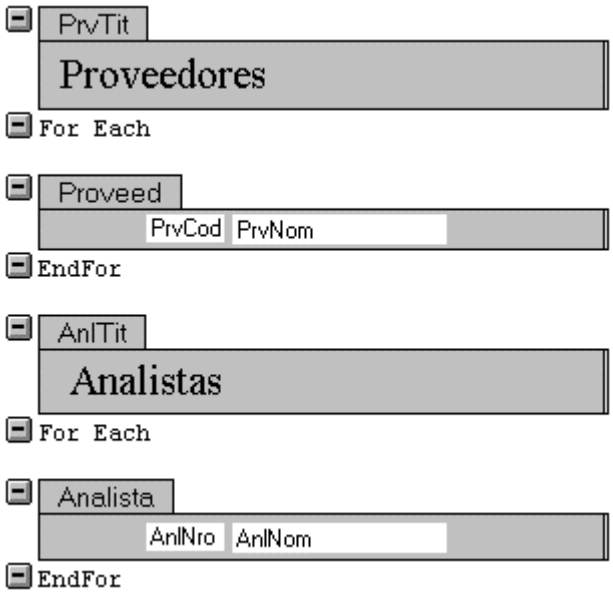
La tabla base del primer FOR EACH es la PEDIDOS (por usar *PedFch*) y la del segundo FOR EACH también (por usar *PedNro* y *PrvNom*). Este es un caso de corte de control sobre la tabla PEDIDOS y ***GENEXUS*** lo indica en el Listado de Navegación con la palabra BREAK en vez de FOR EACH. Los cortes de control pueden ser de varios niveles, por ejemplo, si quisiéramos listar los pedidos por fecha y dentro de cada fecha por proveedor:

```
For each order PedFch
  ....
  For each order PrvCod
    ....
    For each order PedNro
      ....
    Endfor

  Endfor

Endfor
```

Cuando se definen cortes de control es MUY IMPORTANTE indicar el ORDEN en los FOR EACH ya que es la forma de indicarle a ***GENEXUS*** por qué atributos se está realizando el corte de control

For Each paralelos

También se pueden definir grupos FOR EACH paralelos, por ejemplo, si se desean listar los proveedores y los analistas en el mismo listado:



Los listados con FOR EACH paralelos son muy prácticos cuando se quiere hacer listados del tipo 'Listar toda la Información que se tenga sobre un Proveedor', en donde el Layout será del tipo:

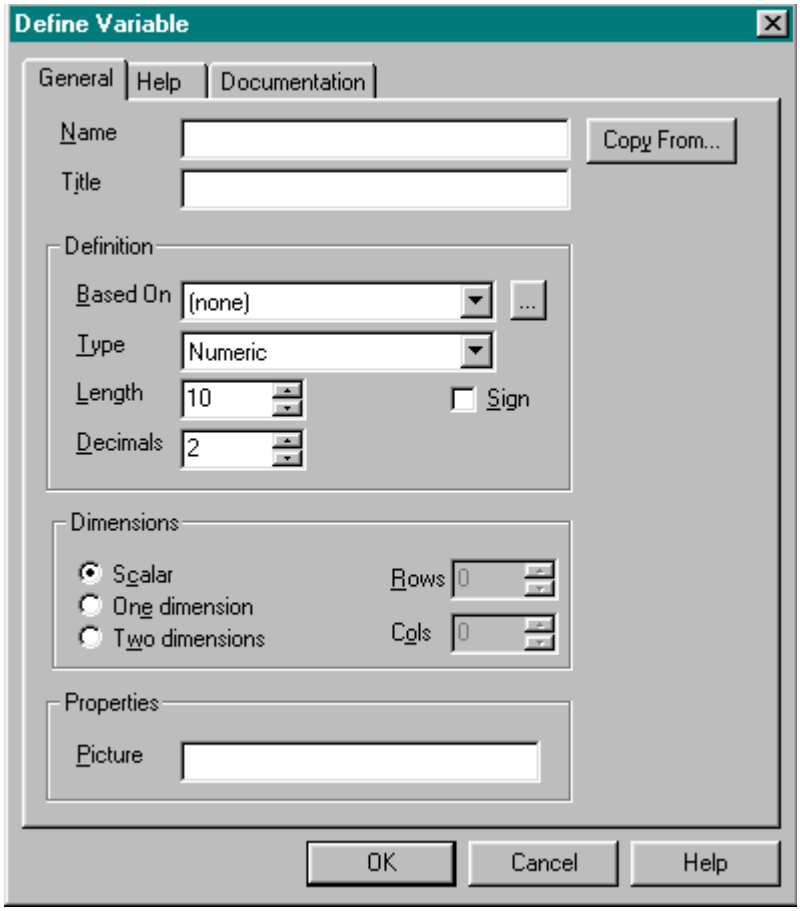
```
For each
    ...  //Información del Proveedor
    For each
       ....  // Información sobre Pedidos
    Endfor
    For each
       ....  // Información sobre Precios
    Endfor
    ....
Endfor
```

Otros comandos

En el Layout se pueden utilizar otros comandos además de los ya vistos. Nos limitaremos aquí a ver solo los más significativos (para un repaso exhaustivo de los mismos sugerimos ver el Reference Manual).

Definición de Variables

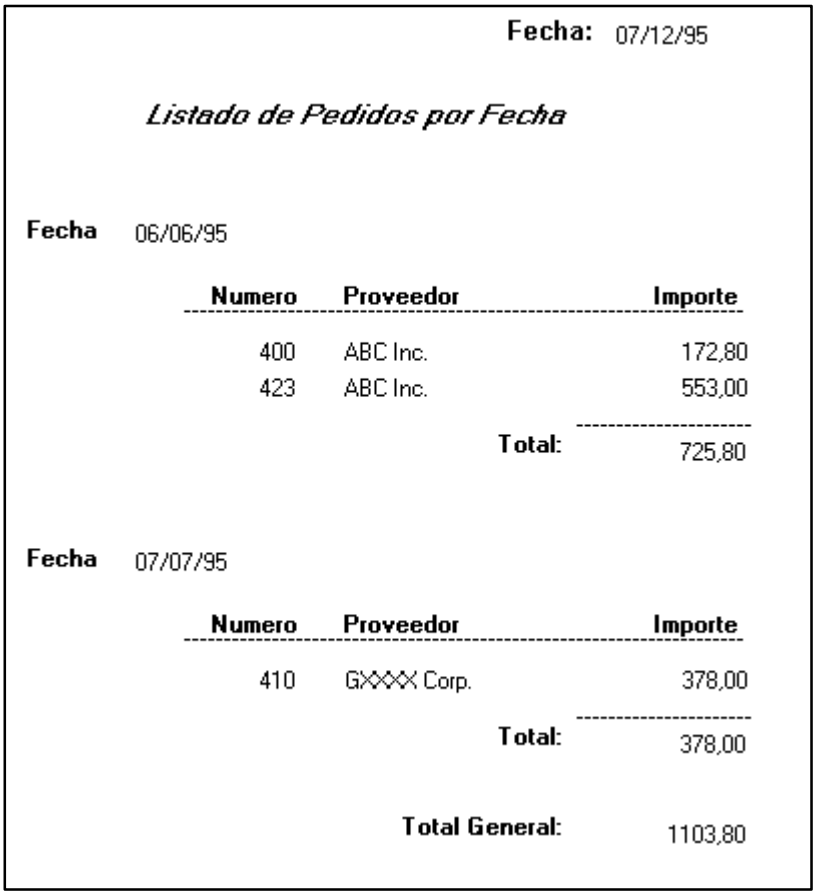
Antes de describir los comandos en si es bueno detenernos en el concepto de Variable. Ya vimos que los datos se almacenan en la base de datos en atributos, pero muchas veces se necesita utilizar Variables, que se diferencian de los atributos por ser locales al Reporte y no estar almacenadas en la base de datos. Una Variable es reconocida por ***GENEXUS*** por ser un nombre que comienza con &, por ejemplo &Today o &Aux.
Existen algunas Variables predefinidas, es decir que ya tienen un significado propio, por ejemplo el ya visto &Today que es una variable con la fecha de hoy o &Page que indica cual es la página que se está imprimiendo, etc.
Se deben definir las características (tipo, largo, decimales, etc.) de las variables que se estén utilizando en el Reporte, con excepción de las predefinidas.



Las variables se pueden definir también en función de otros atributos, usando la opción de Based On. Se recomienda utilizar esta opción siempre que sea posible.

Asignación

Este comando permite asignar valores a variables. Por ejemplo si queremos mostrar totales en un listado:



Fecha: 07/12/95

Listado de Pedidos por Fecha

Fecha 06/06/95

Numero	Proveedor	Importe
400	ABC Inc.	172,80
423	ABC Inc.	553,00
	Total:	725,80

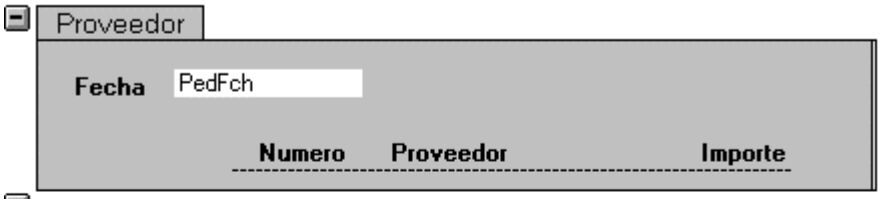
Fecha 07/07/95

Numero	Proveedor	Importe
410	GXXXX Corp.	378,00
	Total:	378,00

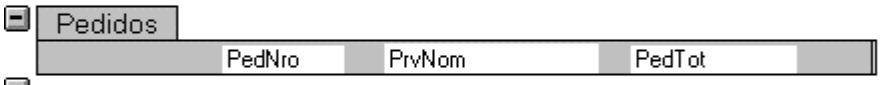
Total General: 1103,80

El layout es:

```
&TotGral = 0
For Each order PedFch
    &TotFch = 0
```




```
    For Each
        &TotFch = &TotFch + PedTot
```



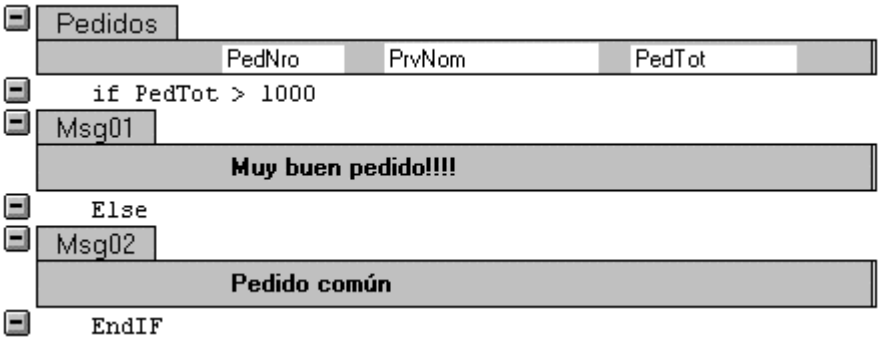

```
    EndFor
```


```
    &TotGral = &TotGral + &TotFch
EndFor
```

Comandos de Control

Los comandos de control son IF-ELSE-ENDIF y DO WHILE-ENDDO. El comando IF-ELSE-ENDIF es muy utilizado para impresión condicional, es decir, imprimir una línea solo cuando se cumple alguna condición.

En el ejemplo anterior:



El comando DO WHILE-ENDDO es muy usado para imprimir varias vías, por ejemplo si se quiere imprimir dos vías del Pedido.

```
FOR EACH order PedNro
        &I = 0
        DO WHILE &I < 2
                &I = &I + 1
                // Imprimir pedido
                 ....
        ENDDO
ENDFOR
```

Llamadas a otros Programas y a Subrutinas

Con el comando CALL se puede llamar a otro programa, éste puede ser otro programa ***GENEXUS*** o un programa externo. El formato del comando es:

CALL(Program_name, Parameter1, ..., ParameterN)

Donde Program_name es el nombre del programa llamado y Parameter1 a ParameterN son los parámetros que se envían, estos parámetros pueden ser atributos, variables y constantes. Los parámetros pueden ser actualizados en el programa llamado.
Cuando desde un Reporte se llama a otro Reporte, que también imprime, es importante pasarle como parámetro la variable predefinida &Line (que contiene el número de línea que se está imprimiendo) para que el Reporte llamado no comience en una página nueva.

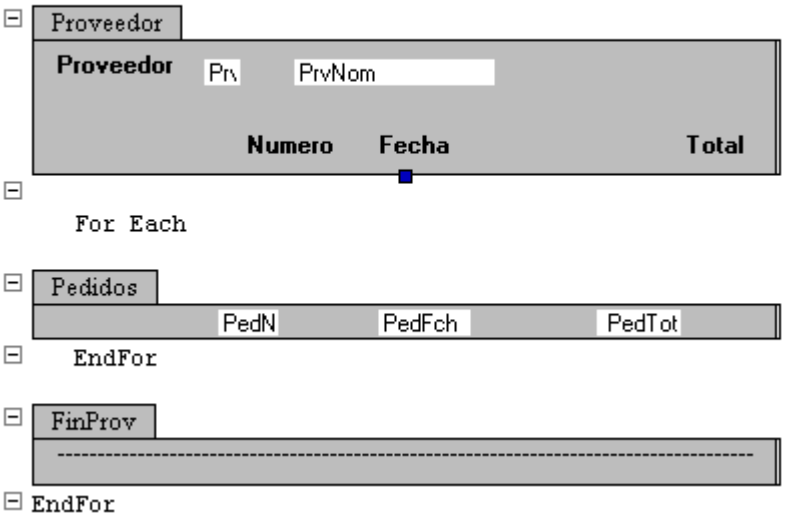
Con el comando DO se llama a una subrutina que se encuentra dentro del mismo Layout. En este caso no son necesarios los parámetros porque están disponibles para la subrutina los mismos datos (variables) que están disponibles en el lugar donde se hace el DO.

```
....
DO 'Nombre-Subrutina'
....


Sub 'Nombre-Subrutina'
    ....
EndSub
```

Print if detail

Volvamos al listado de Pedidos por Proveedor:

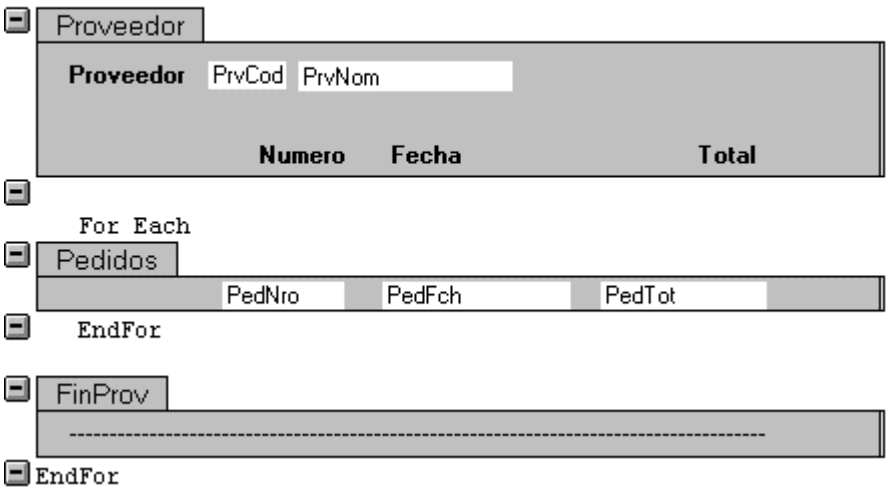
```
For Each PrvCod
    Print if detail
```



Recordemos que el primer FOR EACH tenía como tabla base la PROVEEDO y el segundo la PEDIDOS.

Ahora bien, el primer FOR EACH es independiente del segundo y por lo tanto se imprimirán los datos del primer FOR EACH ANTES de determinar si hay datos en el segundo nivel y como consecuencia se van a imprimir TODOS los proveedores, independientemente de si tienen o no pedidos.
Si se quiere que SOLO se impriman los proveedores que sí tienen pedidos se debe usar el comando Print if detail:

```
For Each PrvCod
    Print if detail
```



Este comando define que el FOR EACH en donde se encuentre debe usar como tabla base la misma tabla base del siguiente FOR EACH. En el ejemplo el primer FOR EACH pasará a estar basado en la tabla PEDIDOS y no en la PROVEEDO y por lo tanto solo se imprimirán los proveedores que tienen pedidos.
Se debe tener en cuenta que, al pasar a estar los dos FOR EACH basados en la misma tabla base, se está definiendo implícitamente un corte de control. Por lo tanto se DEBE definir explícitamente el ORDEN en el primer FOR EACH.
Existen otros comandos como ser EJECT (salta de hoja), NOSKIP, etc. que se detallan en el Reference Manual.

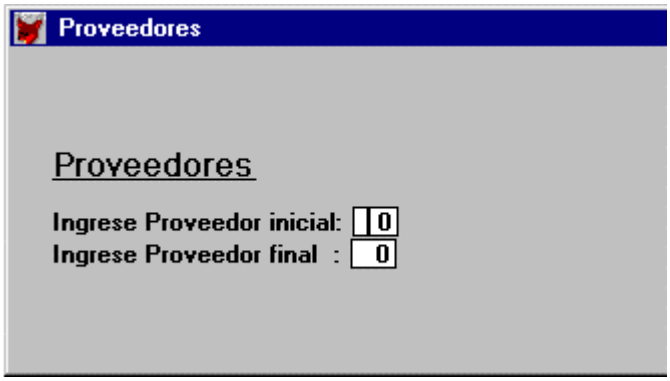
Condiciones

Aquí se definen las condiciones que se quiere imponer a los datos a listar.

Es equivalente a la cláusula WHERE del comando FOR EACH (incluso tienen la misma sintaxis), con la diferencia que el WHERE se aplica a un solo FOR EACH y las condiciones definidas aquí se aplicarán a TODOS los FOR EACH que correspondan. En el Listado de Navegación se indica a que FOR EACH se están aplicando.
Muchas veces se requiere que las condiciones no sean con valores fijos, sino que el usuario elija los valores cuando se ejecuta el Reporte. Por ejemplo, si en el Reporte que lista proveedores deseamos que liste un rango de ellos:

PrvCod >= ask('Ingrese Proveedor inicial: ');
PrvCod <= ask('Ingrese Proveedor final : ');

Cuando se ejecuta el Reporte aparece la pantalla:



Donde se acepta el rango de proveedores a listar. El valor digitado por el usuario se almacena en las variables &ask1, &ask2, etc. según el orden de aparición de los ask() en las condiciones. Estas variables son útiles para imprimir cual fue el valor digitado.

Reglas

Se utilizan para definir aspectos generales del listado. Algunas de las reglas para Reportes son:

Default

Define el valor por defecto de los ask(). En el ejemplo anterior:

default(&ask1, 1);

default(&ask2, 9999);

Las variables &ask1 y &ask2 deben definirse y el orden (1,2,...) es el de definición en las condiciones.

Parm

Es una lista de atributos o variables que recibe el Reporte como parámetros. Por ejemplo, si hacemos un Reporte para imprimir un pedido inmediatamente después de haberlo digitado, en la Transacción se utiliza la regla:

call('RIMPREC', *PedNro*) if after(trn);

y el Reporte será:

Layout

```
FOR EACH
  // Imprime el cabezal del pedido
  ....
  FOR EACH
    // Imprime las líneas del pedido
    ....
  ENDFOR
ENDFOR
```

Reglas

Parm(*PedNro*);

Cuando en Parm() se recibe un atributo, automáticamente este se toma como una condición sobre los datos. En el ejemplo, al recibir como parámetro *PedNro* el primer FOR EACH (que esta basado en la tabla PEDIDOS que contiene *PedNro*) tiene como condición que *PedNro* sea igual al parámetro y no es necesario poner un WHERE. Si se hubiera recibido el parámetro como una variable, entonces sí es necesario definir el WHERE.

Layout

```
FOR EACH
  WHERE PedNro = &Pedido
  ....
ENDFOR
```

Reglas

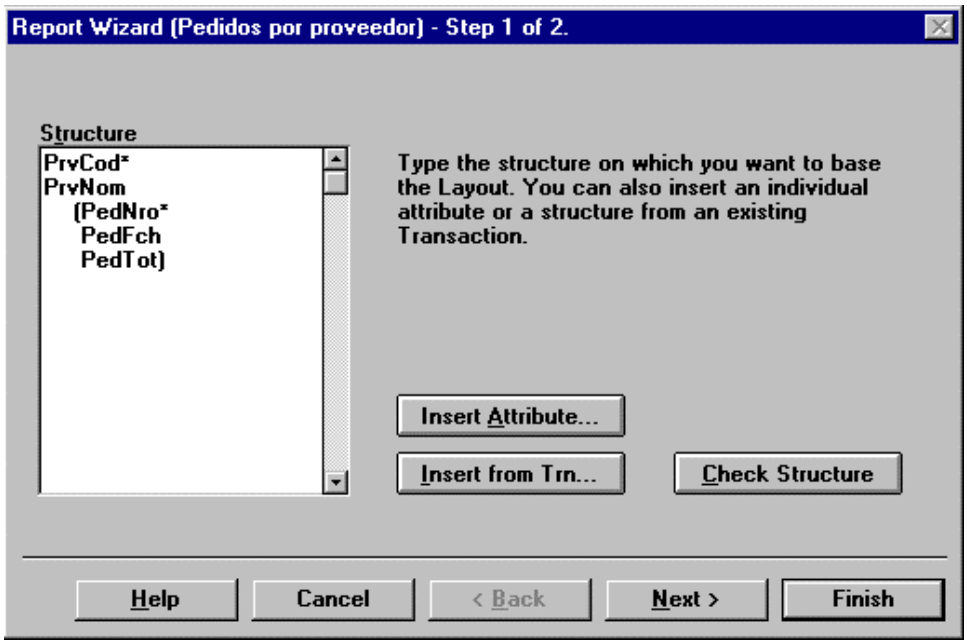
Parm(&Pedido);

Report Wizard

El Report Wizard permite realizar el diseño del layout de los Reportes y Procedimientos de manera mas sencilla. Diseñemos nuevamente el reporte de pedidos por proveedor.

El diseño del layout consiste en dos pasos.

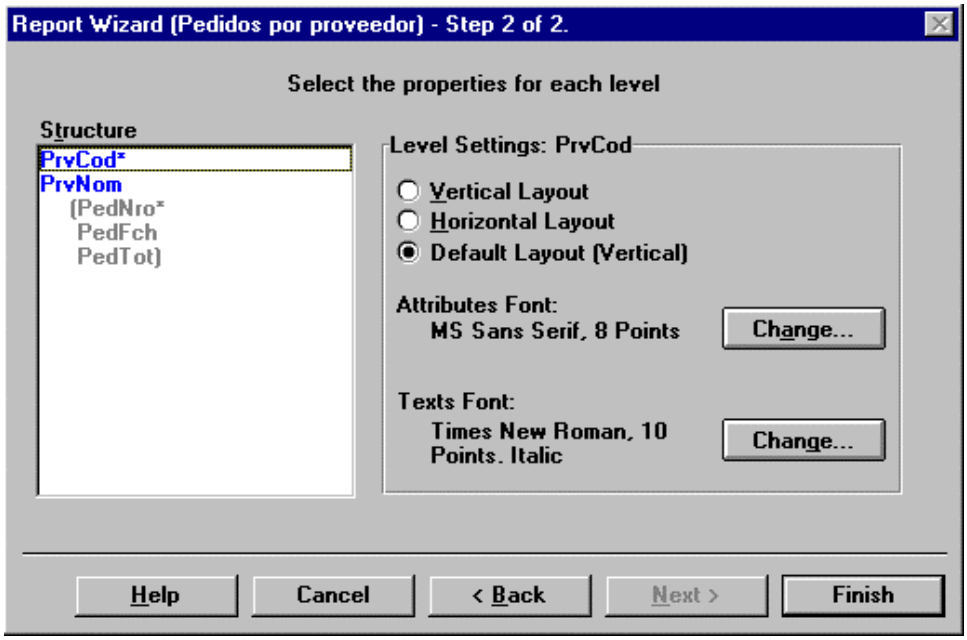
Paso 1: Requiere que se provea de una estructura de datos. Notar que dicha estructura de datos debe incluir atributos definidos en las transacciones. La estructura de datos usará una estructura sintáctica similar a la usada en las transacciones, sin embargo, los paréntesis se usan para indicar niveles de FOR EACH (en vez de niveles de una transacción) y el asterisco (‘*’) se usa para indicar el orden deseado (y no para indicar la unicidad como en las transacciones). Las reglas que son usadas para definir la estructura de una transacción también se aplican al definir la estructura del layout.



Para diseñar nuestro reporte definimos una estructura en donde en el primer nivel tenemos los datos del proveedor y en el segundo los del pedido. Esto va a generar dos for each anidados en donde el exterior va a estar ordenado por código de proveedor y el interno por numero de pedido.
Una vez que se ha definido correctamente la estructura, se presiona el botón de Next para pasar al siguiente paso.

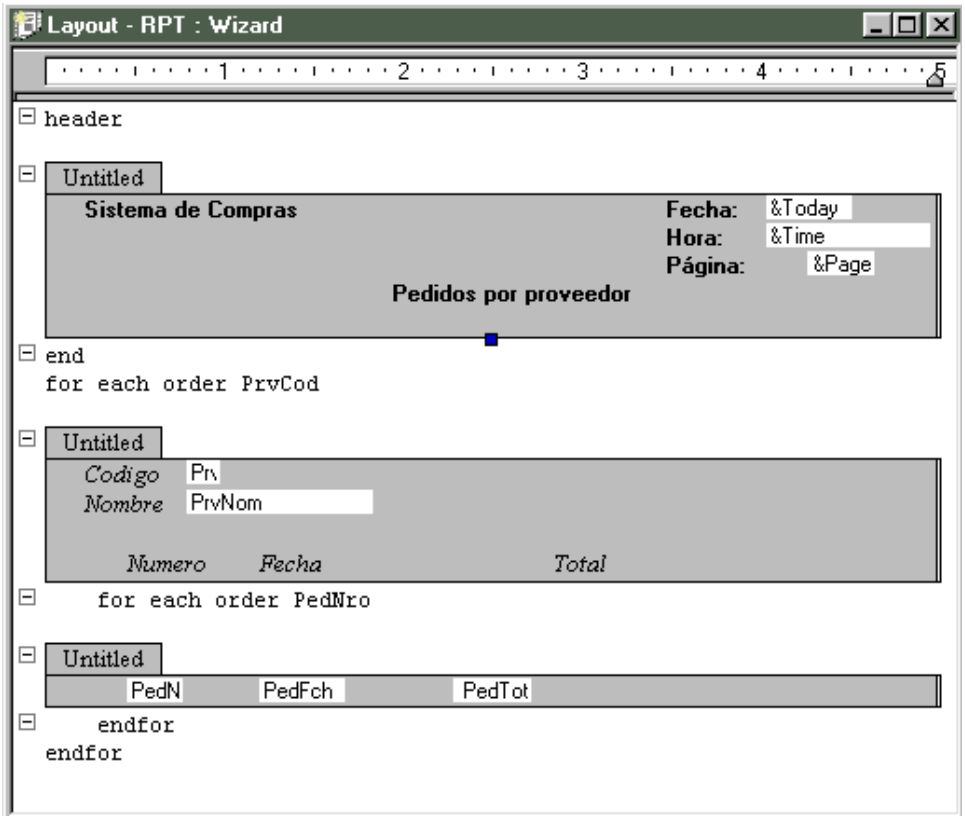
Paso 2: Permite definir otras características del layout. Se permite elegir los fonts para representar los atributos o textos, también se permite definir si los cabezales de los atributos para cada uno de los niveles se despliegan horizontalmente o verticalmente.

El dialogo del paso 2 para la estructura de datos usada en el paso 1 se ve de la siguiente manera:



Como se puede ver, el nivel para el cual estamos definiendo el tipo del layout se presenta en un color diferente. Para este nivel, seleccionaremos el tipo vertical de layout y para el segundo nivel nos aseguraremos que el check box correspondiente al tipo de layout vertical no este marcado, de esta manera el display del nivel interior será horizontal. En nuestro ejemplo cambiamos los fonts de los títulos, mientras que los de los atributos dejamos el default.
Presionando el botón de "Finish" se graban las definiciones para el paso actual y se sale del dialogo del Report Wizard.

El reporte generado es el siguiente:



Una vez que todas las opciones del Wizard fueron aceptadas, se genera el layout del reporte, el cual puede ser modificado como cualquier otro reporte. Vemos que el diseño del layout es idéntico al que definimos anteriormente en forma manual.

Properties

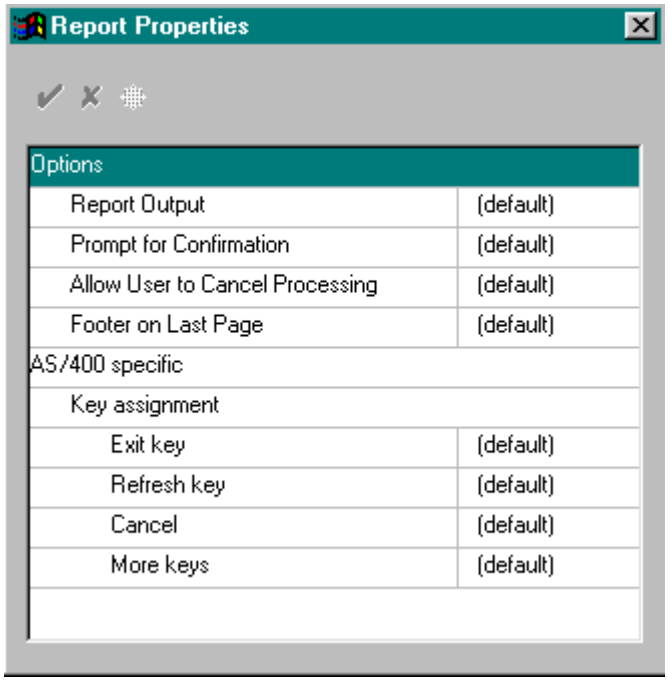
En el editor de propiedades de los reportes podemos definir:

Report Output: Si el reporte va a salir solo por pantalla, impresora o se le pregunta al usuario en tiempo de ejecución.

Prompt for Confirmation: Si deseamos que se pida confirmación al usuario.

Allow User to Cancel Processing.

Footer on Last Page: Imprimir el footer en la última pantalla.



DISEÑO DE PROCEDIMIENTOS

Los Procedimientos son el objeto ***GENEXUS*** con el que se definen todos los procesos no interactivos de actualización de la base de datos y las subrutinas de uso general.
Este objeto tiene todas las características de los Reportes (cualquier Reporte se puede realizar como un Procedimiento) más algunas características particulares para la actualización de la base de datos. Por esta razón se recomienda haber leído previamente el capítulo Diseño de Reportes.
Vamos a ver entonces los comandos disponibles en los procedimientos para la actualización de la base de datos

Modificación de datos

La modificación de datos de la base de datos se realiza de forma implícita, ya que no hay un comando específico de modificación (como podría ser REWRITE), sino que basta con asignar un valor a un atributo dentro de un For Each para que automáticamente se realice la modificación.

Por ejemplo supongamos que queremos tener un proceso batch que actualiza el atributo *ArtPrcUltCmp* para adecuarlo según la inflación para todos los Artículos almacenados:

Programa:

```
For each
  ArtPrcUltCmp = ArtPrcUltCmp * (1 + &Inf)
Endfor
```

Reglas:

```
Parm( &Inf );
```

Se puede modificar más de un atributo dentro del mismo FOR EACH, por ejemplo, si también queremos modificar el atributo *ArtFchUltCmp*, poniéndole la fecha de hoy:

```
For each
  ArtPrcUltCmp = ArtPrcUltCmp * (1 + &Inf)
  ArtFchUltCmp = Today( )
Endfor
```

La modificación en sí se realiza en el ENDFOR y no cuando se realiza la asignación del atributo.
En estos dos ejemplos se modificaron atributos de la tabla base del FOR EACH, pero también es posible actualizar atributos de las tablas N-1 (tabla extendida) con el mismo mecanismo de asignación de atributos.
No todos los atributos pueden ser modificados. Por ejemplo, no se pueden modificar los pertenecientes a la clave primaria de la tabla base del FOR EACH (en este caso *ArtCod*). Tampoco se pueden modificar los atributos que pertenecen al índice (el orden) con el que se está accediendo a la tabla base.
En el Listado de Navegación se informa cuales son las tablas que se están actualizando marcando con un + estas tablas. Por ejemplo:

------>> + **ARTICULO**

Eliminación de datos

Para eliminar datos se utiliza el comando DELETE dentro del grupo FOR EACH-ENDFOR. Por ejemplo, si queremos eliminar todos los pedidos con fecha anterior a una fecha dada:

```
For each
where PedFch < ask('Eliminar pedidos con fecha menor')
        defined by PedFch
        For each
                defined by PedCnt

                // Se eliminan todas las lineas del pedido
                DELETE
        Endfor
        // Después de eliminar las lineas se elimina el cabezal
        DELETE
Endfor
```

Creación de datos

Para crear nuevos registros en la base de datos se usan los comandos NEW-ENDNEW. Tomemos por ejemplo un Procedimiento que recibe como parámetros *AnlNro* y *AnlNom* e inserta estos datos en la tabla de analistas:

Programa:

New
AnlNro = &Nro
AnlNom = &Nom
Endnew

Reglas:

Parm(&Nro, &Nom);

Con este procedimiento se pueden crear registros en la tabla de analistas utilizando una subrutina.
El comando NEW realiza el control de duplicados, y solo creará el registro si ya no se encuentra en la tabla. Con la cláusula WHEN DUPLICATE se programa la acción a realizar en caso de duplicados. Por ejemplo, en el caso anterior si se quiere que si el analista ya estaba registrado se actualice el *AnlNom*:

New
AnlNro = &Nro
AnlNom = &Nom
When Duplicate
For each
AnlNom = &Nom
Endfor
Endnew

Observar que para realizar la modificación de atributos las asignaciones se deben encontrar DENTRO de un grupo FOR EACH-ENDFOR.

Controles de integridad y redundancia

En los Procedimientos el único control de integridad que se realiza automáticamente es el de control de duplicados en el comando NEW.
Pero NO se controla automáticamente la integridad referencial, por ejemplo, si hacemos un Procedimiento que crea automáticamente Pedidos, no se controlará que el *PrvCod* que se cargue exista en la tabla de proveedores. O cuando se elimina el cabezal de un pedido, se debe tener esto en cuenta para eliminar las líneas del mismo explícitamente.
El mismo criterio se aplica para las fórmulas redundantes. Estas deben ser mantenidas explícitamente con los comandos de asignación/update.

DISEÑO DE PANELES DE TRABAJO

Con Paneles de Trabajo (o Work Panels) se definen las consultas interactivas a la base de datos. Si bien éste es un objeto muy flexible que se presta para múltiples usos, es especialmente útil para aquellos usuarios que utilizan el computador para pensar o para tomar decisiones.
La forma de programar los Paneles de Trabajo esta inspirada en la programación de los ambientes gráficos, especialmente la programación dirigida por eventos.

Para cada Panel de Trabajo se debe definir:

Información
Datos generales del Panel de Trabajo, por ejemplo nombre, descripción, etc.

Form
Al igual que las transacciones, se debe definir el form con el que interactuará el usuario.

Eventos
Aquí se define la respuesta a los eventos que pueden ocurrir durante la ejecución del Panel de Trabajo. Por ejemplo que respuesta dar cuando el usuario presionó Enter o un botón, etc.

Condiciones
Define que condiciones deben cumplir los datos para ser presentados en la pantalla. Son equivalentes a las Condiciones de los Reportes y Procedimientos.

Reglas
Definen aspectos generales del Panel de Trabajo, por ejemplo orden de los datos a mostrar, parámetros, etc.

Ayuda
Texto para la ayuda a los usuarios en el uso del Panel de Trabajo.

Documentación
Texto técnico para ser utilizado en la documentación del sistema.

Propiedades
Propiedades generales del Work Panel.

En este capítulo solo se presentarán las características y usos más salientes de los Paneles de Trabajo, pero se recomienda leer el capítulo de Paneles de Trabajo en el Manual de Usuario ***GENEXUS***, para una visión más profunda.

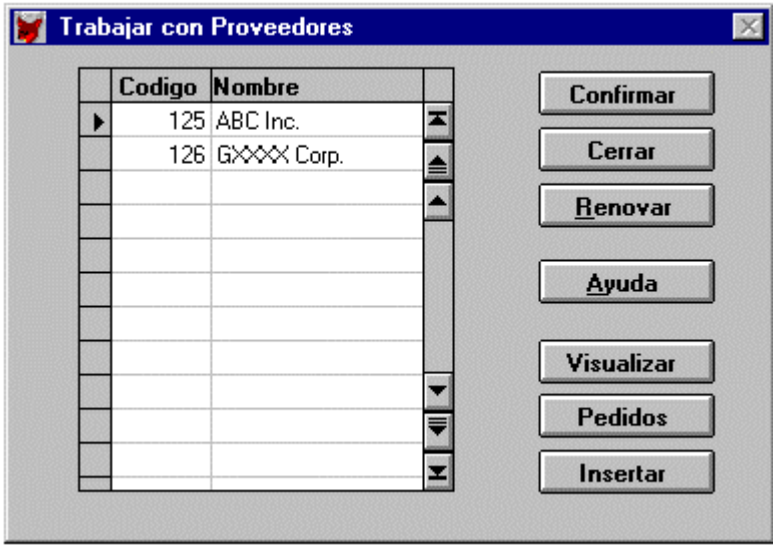
Ejemplo

Para ilustrar más fácilmente el uso de Paneles de Trabajo, vamos a realizar una serie de consultas encadenadas.
Estas consultas comienzan con un Panel de Trabajo de proveedores, donde el usuario selecciona para luego realizar alguna acción, en este caso para cada proveedor seleccionado se puede:

Visualizar los datos del proveedor.

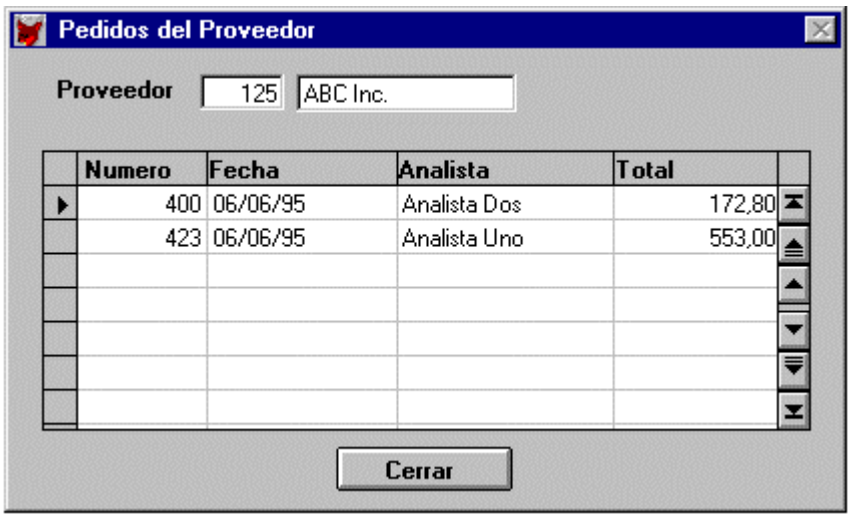
Visualizar los pedidos del proveedor.

El form será:



si, por ejemplo, el usuario presiona el botón de Visualizar en el proveedor 125, se abre una ventana donde se muestran los datos del proveedor:

Si se presiona el botón de pedidos se presentará la siguiente pantalla con los pedidos del proveedor:

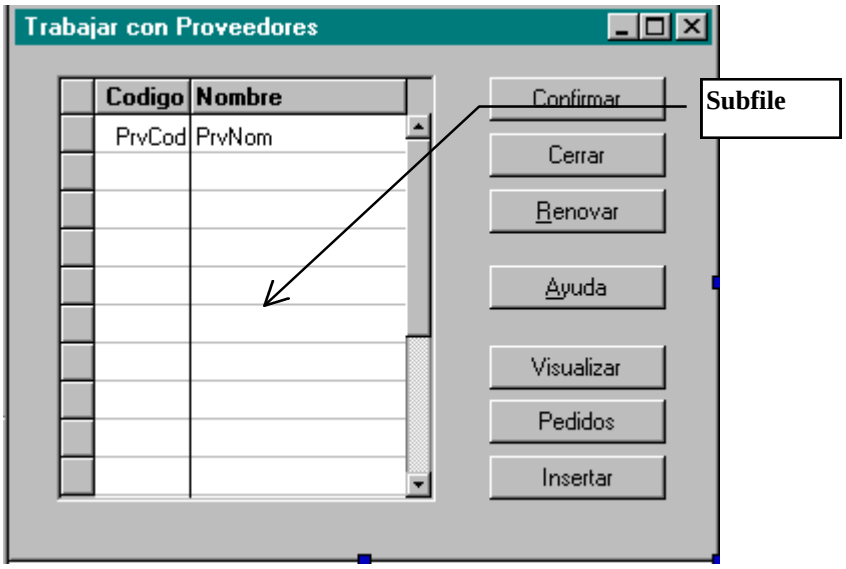


A su vez este Panel de Trabajo podría extenderse, permitiendo seleccionar algún pedido para visualizar qué artículos fueron pedidos y así sucesivamente.
Las acciones que vimos en el primer Panel de Trabajo se aplican a una línea, pero existen otras acciones que no se aplican a ninguna línea en particular, por ejemplo Insertar un

nuevo proveedor. Si se elige esta acción se pasa el control a la transacción de proveedores para definir uno nuevo.

Renovar - es una acción incluída automáticamente por *GENEXUS* y, si se elige, se vuelven a cargar todas las líneas. Más adelante nos detendremos en esta acción.
Veamos ahora como se define el primer Panel de Trabajo del ejemplo.

Form del Work Panel

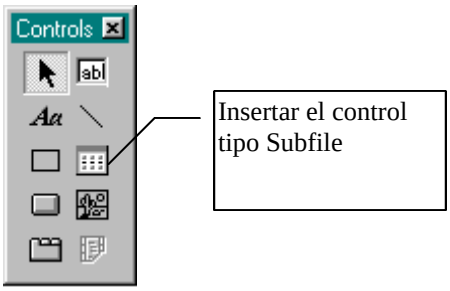


Este form se define de una manera similar a los forms de las transacciones.

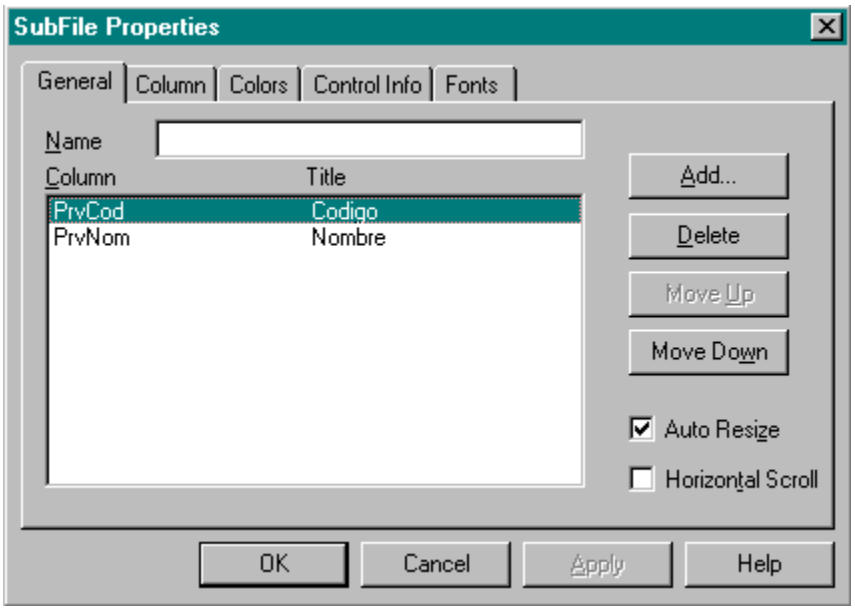
Cuando se define un subfile en el form se está indicando que se va a mostrar una cantidad indefinida de datos (en este caso los proveedores). Si bien sólo se pueden ver simultáneamente las líneas presentes en la pantalla, ***GENEXUS*** provee todos los mecanismos para que el usuario se pueda 'mover' (llamaremos 'paginar') dentro de la lista, como por ejemplo ir a la primera página, a la siguiente, etc.
El Panel de Trabajo anterior tiene un SubFile compuesto de *PrvCod* y *PrvNom*. Los SubFile pueden contener además de atributos, variables y arrays de una y dos dimensiones.

Subfile

Si seleccionamos el botón de “subfile” nos va a permitir incorporar un subfile en el form del work panel y definir el tamaño que deseamos.



Una vez pasteado el control en el form se abre automáticamente el siguiente dialogo:

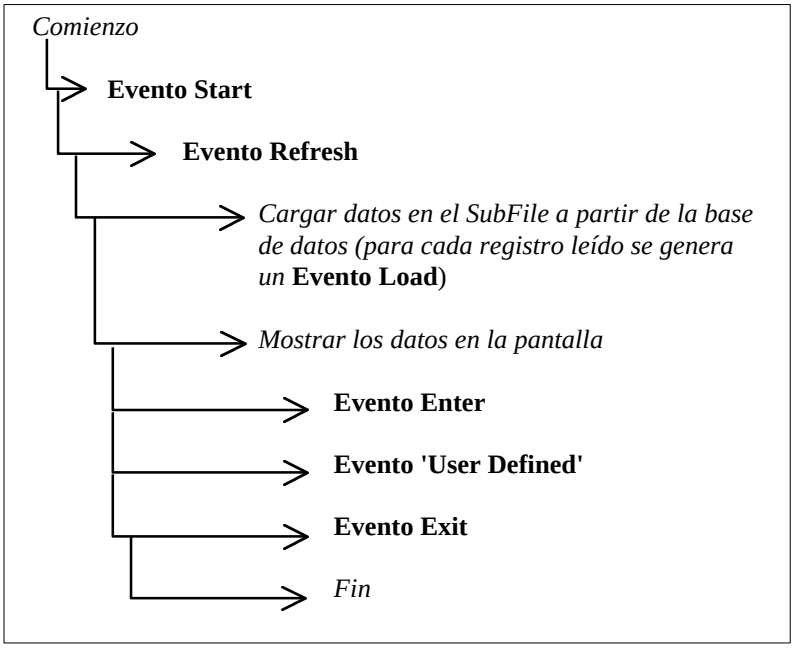


Para seleccionar los atributos que van a ser incluidos en el subfile se debe presionar el boton “Add”. Con el tab “Column” vamos a poder cambiar los títulos de las columnas (por defecto se define el titulo del atributo). Los botones “MoveUp” y “Move Down” nos van a permitir cambiar el orden en que se van a desplegar los atributos. Con los tabs “Colors” y “Fonts” podemos cambiar los colores y el tipo de letra de las columnas. Con el tab Control Info se selecciona el tipo de control a utilizar, y dado el tipo, indicar la información necesaria para ese tipo, las opciones disponibles son: Edit, Check box, Combo y Dynamic Combo (o el default del atributo o variable de la columna).

Podemos también ajustar manualmente el ancho de cada columna desmarcando la opción de “Auto Resize”, si esta opción queda marcada el ancho de la columna lo determina ***GENEXUS*** en función del largo del atributo o del titulo de la columna.
El check box “Horizontal Scroll” permite que en tiempo de ejecución aparezca una barra de scroll horzontal (además de la de scroll vertical que aparece automáticamente)

Eventos

La forma tradicional de programar la interfaz entre el usuario y el computador (incluídos los lenguajes de cuarta generación) es subordinar la pantalla al programa. Es decir, desde el programa se hacen 'calls' a la pantalla. En Paneles de Trabajo es lo opuesto: el programa está subordinado a la pantalla, ya que una pantalla realiza 'calls' al programa para dar respuesta a un evento.
Esta forma de trabajar es conocida como EVENT DRIVEN (dirigida por eventos) y es típica de los ambientes GUI (Interfaz de Usuario Gráfica), en donde ya ha demostrado su productividad, fundamentalmente porque se necesita programar sólo la respuesta a los eventos y no la tarea repetitiva que implica el manejo de toda la interfaz.
La estructura de eventos es, de forma resumida, la siguiente:



Para programar cada uno de los eventos se utiliza el mismo lenguaje que ya vimos en Reportes y Procedimientos, aunque no están permitidos ni los comandos relacionados con la impresión (Header, etc.) ni los de actualización de la base de datos (New, Delete, etc.). También existen comando específicos para Paneles de Trabajo.

Evento Start

```
Event Start

    &Fecha = &Today
Endevent
```

En este caso, se utiliza el evento Start para mover la fecha actual a la variable &Fecha que por ejemplo se podrá mostrar en la pantalla. Esta técnica es muy útil para inicializar valores por defecto, sobre todo para aquellos que son complejos (por ejemplo, &Valor = udf('PX', ...)).

Evento Enter

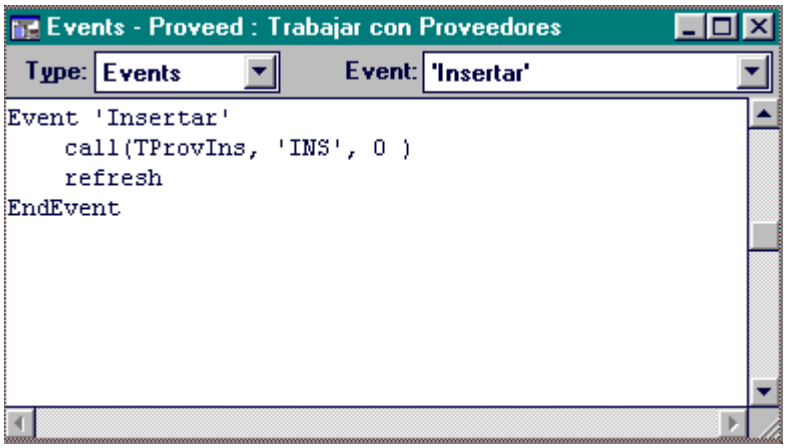
Este evento se dispara cuando el usuario presiona ENTER. Consideremos el siguiente ejemplo: borrar un conjunto de proveedores. Entonces se define una nueva variable en el subfile, &Op, y por cada línea que se le asocie la opción 4 se llama a un procedimiento que borra los clientes. El código asociado al evento enter en este caso es el siguiente:

```
Event Enter

   For each line

        If &op = '4'
            Call(PDltPrv, PrvCod)
        Else
            If &op <> ' '
                &msg = concat(&op, ' no es una opción valida')
                Msg( &msg )
            Endif
        Endif

   Endfor

Endevent
```

El evento Enter es un poco más extenso y vale la pena detenernos en él. En primer lugar se utiliza el comando FOR EACH LINE que realiza un FOR EACH, pero no sobre los datos de la base de datos, sino sobre el SubFile.

Para cada una de las líneas del SubFile, se pregunta por el valor de &op, si &op es '4' se llama al procedimiento DltPrv, y en otro caso, da un mensaje indicando que la opción no es válida. Notar que se podrían haber definido mas opciones y preguntar por ellas en este evento.

Eventos definidos por el usuario (User Defined Events)



Este es un evento definido por el usuario, para el cual se debe definir el nombre ('Insertar'). En este evento se llama a la transacción de proveedores (por más detalles sobre como llamar de un Panel de Trabajo a una Transacción ver el capítulo de Paneles de Trabajo en el Manual de Referencia). Luego de llamar a la transacción, se ejecuta el comando Refresh, que indica que se debe cargar nuevamente el SubFile, ya que se ha adicionado un nuevo proveedor. También se podría haber usado el comando con el parámetro keep (**Refresh Keep)** que hace que una vez que el comando refresh se ejecuta el cursor queda posicionado en la misma línea del subfile que estaba antes de la ejecución.

Evento Refresh

Los datos que se presentan en pantalla son los cargados en el SubFile, que a su vez fueron cargados desde la base de datos.
En un ambiente multi-usuario, los datos de una pantalla pueden haber quedado desactualizados si desde otra terminal fueron modificados. Cuando el usuario desea actualizar los datos, debe dar click sobre el boton Refresh (o presionar la tecla F5), cuyo efecto es cargar nuevamente el SubFile.
En algunos casos, desde otro evento también se necesita cargar nuevamente el SubFile. Por ejemplo, cuando en otro evento se llama a una transacción que actualiza los datos (es el caso del Evento 'Crear Proveedor').

Tenemos, por lo tanto, que el SubFile se carga (es decir, se genera el evento Refresh) cuando:

- Se carga la pantalla - por lo tanto después del evento Start siempre hay un evento Refresh.
- El usuario dio click sobre el botón Refresh (o presiono la tecla F5).
- Se ejecutó el comando Refresh en otro evento.

Carga de datos en el Panel de Trabajo

Como vimos en el diagrama de eventos, después del evento Refresh, se carga el SubFile con los datos obtenidos de la base de datos.
Para determinar de que tablas se deben obtener los datos, GENEXUS utiliza el mismo mecanismo descripto para los grupos FOR EACH en el capítulo Diseño de Reportes. Pero en este caso no hay ningún FOR EACH!.
Sin embargo, se considera que cada Panel de Trabajo tiene un FOR EACH implícito, que contiene los siguientes atributos:

- Todos los atributos que se encuentran en la pantalla (incluso los que se encuentran en la parte fija de la misma).
- Todos los atributos que se encuentren en los Eventos, con la excepción de aquellos que se encuentren dentro de un FOR EACH explícito.
- Todos los atributos mencionados en las reglas Hidden y Order

Con estos atributos GENEXUS determina cual es la tabla extendida correspondiente.
Para cada uno de los registros encontrados, se genera un evento Load. Así, la relación entre el evento Refresh, el FOR EACH implícito y el evento Load es la siguiente:

```
Event Refresh
  ...
Endevent

FOR EACH

  Event Load
    ...
  Endevent

ENDFOR
```

Los datos que se leen en este FOR EACH implícito, se cargan en el SubFile, que es el que se presenta en pantalla. Esta carga se realiza 'bajo pedido', es decir que al

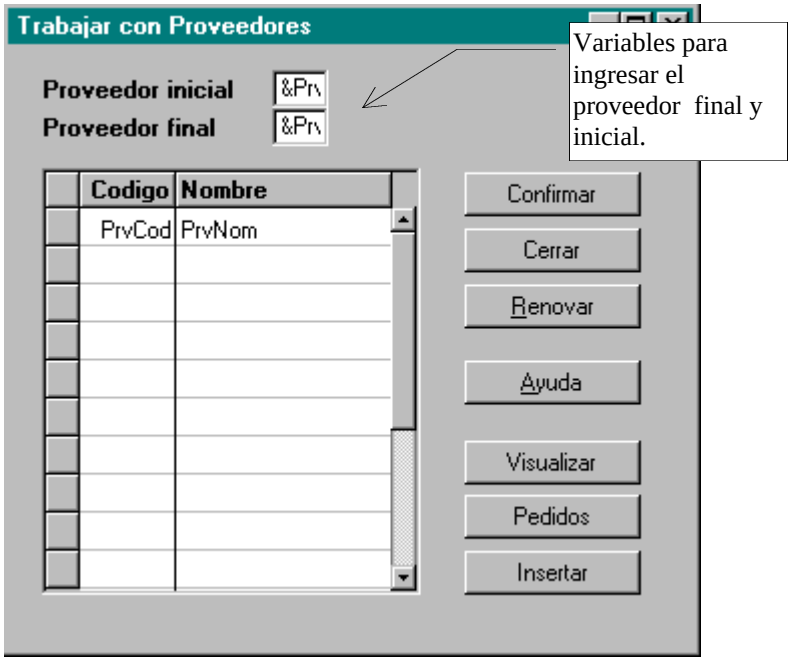
comienzo sólo se carga la primera página y, a medida que el usuario requiere más información, se van cargando las siguientes páginas. Existe una propiedad para indicar que se quiere cargar el SubFile de una sola vez, si fuera necesario.
Igual que en los Reportes y Procedimientos en el Listado de Navegación del Panel de Trabajo se indica qué tablas se están utilizando, índices, etc.
Puede darse el caso que no exista el FOR EACH implícito. Esto se debe a que no hay atributos ni en la pantalla ni en los eventos ni en las reglas Hidden y Order (porque se usan sólo variables). En este caso se genera sólo UN evento Load y es en éste donde se deben cargar los datos en el SubFile (utilizando el comando Load), por ejemplo:

```
Event Refresh
  ...
Endevent

Event Load
    ...
    For each
        ...
        Load
    Endfor

Endevent
```

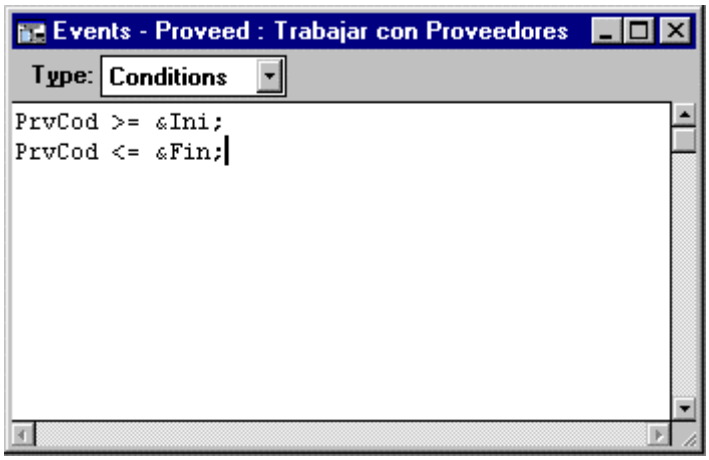
Observar que en este caso se pierde la facilidad de carga 'bajo pedido' y por lo tanto antes de mostrar los datos se carga todo el SubFile.

Condiciones

Aquí se definen las restricciones a los datos de la misma forma que en los Reportes.
Lo interesante de las condiciones en los Paneles de Trabajo, es que en las mismas se puede utilizar variables que se encuentran en la pantalla, de tal manera que el usuario, cambiando los valores de estas variables, cambia los datos que se muestran en el SubFile sin tener que salir de la pantalla.
Por ejemplo, si queremos visualizar sólo un rango de proveedores, agregamos en la pantalla las variables:



Y definiendo las condiciones:



se obtiene el efecto deseado.

Todas las consideraciones sobre optimización de condiciones vistas en el capítulo Diseño de Reportes son igualmente válidas aquí.

Reglas

En Reglas se aplican la mayor parte de las reglas de los Reportes, más algunas particulares:

Order

Define cual es el orden de los datos del SubFile. Es equivalente a la cláusula ORDER del FOR EACH y se aplica todo lo dicho en el capítulo Diseño de Reportes sobre el tema (incluída la creación de índices temporarios).
Por ejemplo si queremos ver los proveedores por orden alfabético:

order(*PrvNom*);

Noaccept

A diferencia de las Transacciones, en los Paneles de Trabajo ningún atributo es aceptado (siempre se muestra su valor pero no se permite modificarlo). Para variables se sigue el siguiente criterio:

- Todas las variables presentes en la pantalla son aceptadas, a no ser que tengan la regla NOACCEPT.

Por ejemplo, en un Panel de Trabajo que deseemos poner la fecha en el form, se necesita la regla:

noaccept(&Fecha);

Search

Cuando el SubFile es demasiado extenso, muchas veces se quiere brindar la posibilidad de que el usuario pueda 'posicionarse' en alguna línea determinada del mismo, en forma directa, sin tener que avanzar página por página. Por ejemplo si en nuestro Panel de Trabajo de proveedores queremos que exista la posibilidad de buscar un proveedor según su nombre, se debe definir la regla:

search(*PrvNom* >= &Nom);

y en la pantalla se debe agregar la variable &Nom:

De esta manera, cada vez que el usuario digite algo en &Nom, luego de dar Enter, el SubFile quedará posicionado en el primer proveedor con *PrvNom* mayor o igual al digitado. En los casos de nombres es muy usual la regla:

search(*PrvNom* .LIKE. &Nom);

En donde el SubFile se posiciona en el primer proveedor cuyo *PrvNom* contenga a &Nom (en el ejemplo si en &Nom se digitó 'xx' se posicionará en 'GXxxxx Corp.').
Si bien esta regla es muy útil para que el usuario avance rápidamente en un SubFile, se debe tener en cuenta que no hay ningún tipo de optimización sobre la misma. Si se quieren utilizar los criterios de optimización se deben usar las Condiciones.

Bitmaps

Los bitmaps son usados usualmente para mejorar la presentación de las aplicaciones. Se pueden incorporar en los forms de las transacciones, reportes y work panels. ***GENEXUS*** provee dos métodos diferentes para agregar un Bitmap:

Fixed Bitmaps

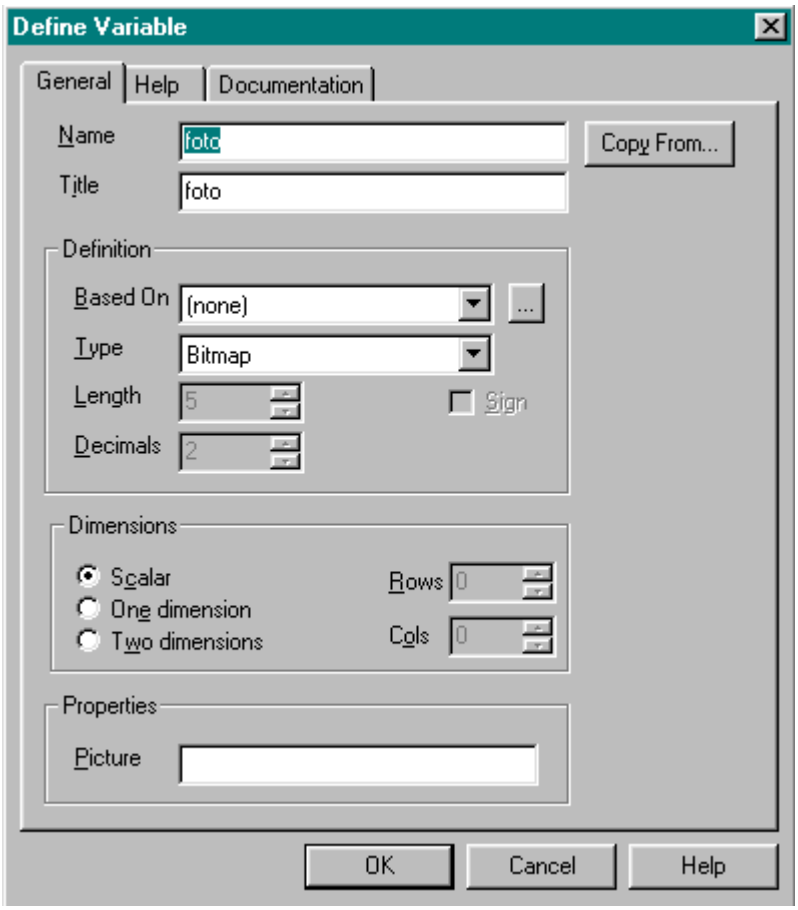
Se puede agregar un Bitmap seleccionando el botón de Bitmap de la paleta de herramientas. De esta forma tenemos que indicar su ubicación. Agreguemos un logo en el work panel que permite visualizar los datos de un proveedor. El nuevo form se vera de la siguiente forma:

En este caso siempre aparecerá el mismo Bitmap en el form.

Dynamic Bitmaps

Este método tiene como diferencia que el Bitmap lo asigno a una variable y usando la función **Loadbitmap** puedo cargar el Bitmap que yo desee. Por ejemplo se puede tener un Bitmap que despliegue la foto de cada proveedor.

Definición de una varible de tipo bitmap:



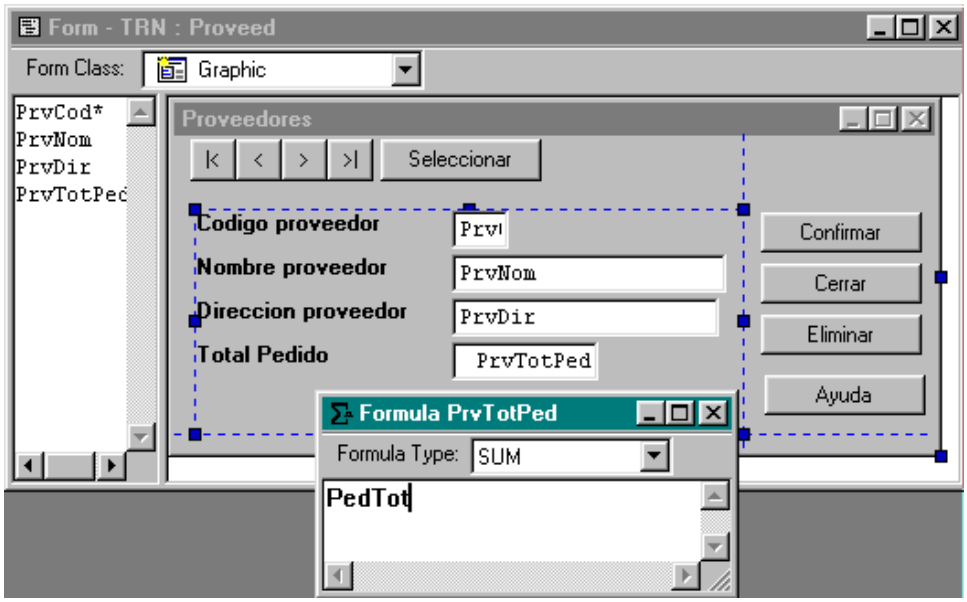
En el evento load hay que cargar el bitmap para el proveedor pasado por parámetro.

En el atributo PrvFoto se carga la ubicación del archivo (bmp).

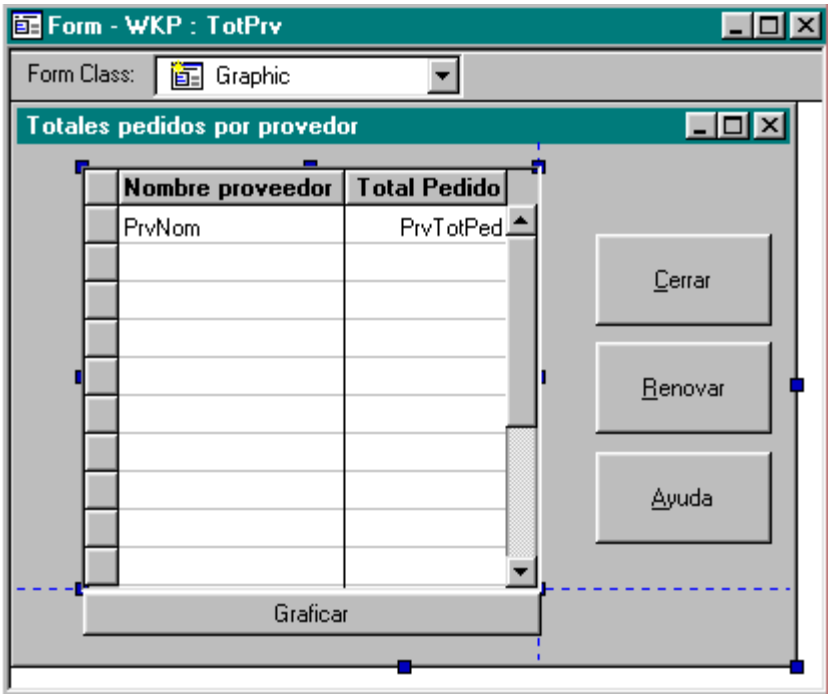
En tiempo de ejecución se verá de la siguiente forma:

Gráficas

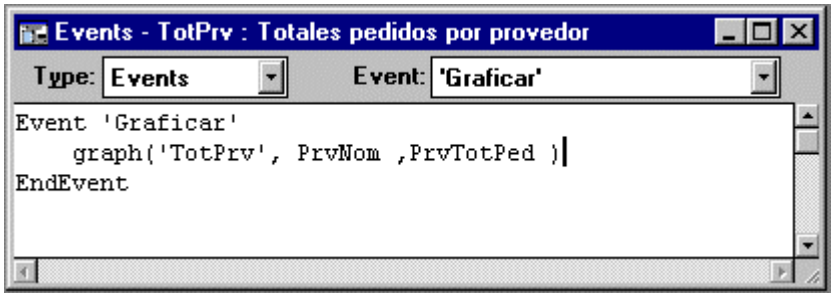
Grafiquemos el total pedido por Proveedor. Vamos a definir entonces una formula que me calcule el total pedido en la transacción de proveedores:



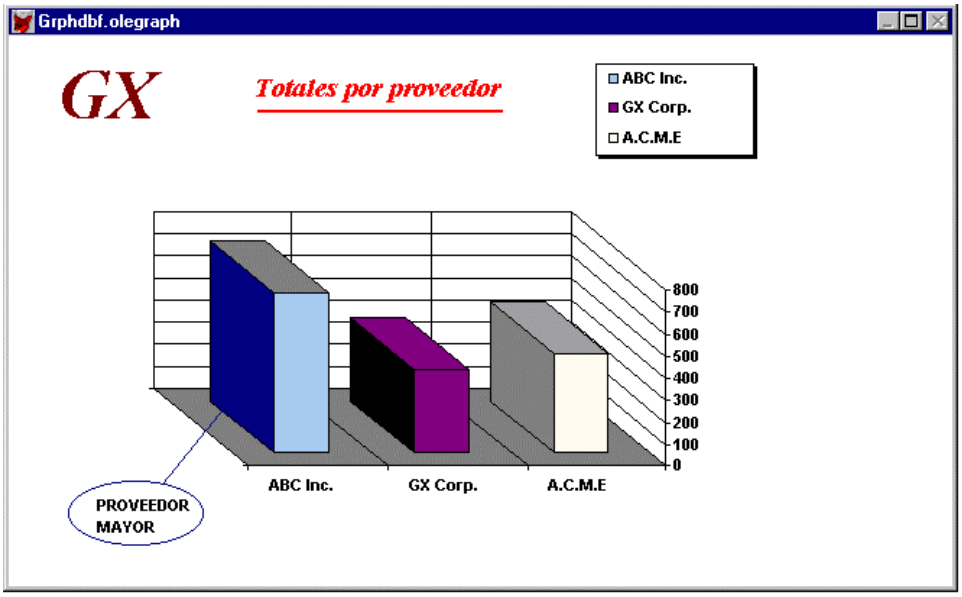
Definimos entonces un nuevo atributo, “PrvTotPed” y decimos que este atributo es una formula Sum vertical del atributo PedTot, de esta forma estamos definiendo el total Pedido de un proveedor como la suma de todos los totales de los pedidos de ese proveedor. Recordemos que el atributo PedTot había sido definido como una formula, sumando todos los importes de la línea que a su vez era también una formula.
De esta forma tenemos disponible este atributo para usarlo en la definición del work panel.

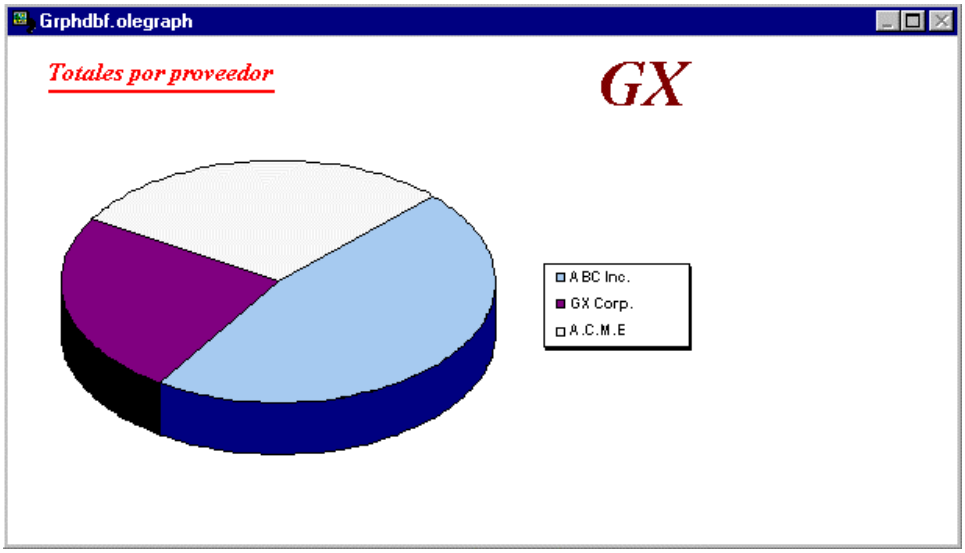


En este work panel definimos el botón “Graficar” el cual va a tener asociado el siguiente evento:

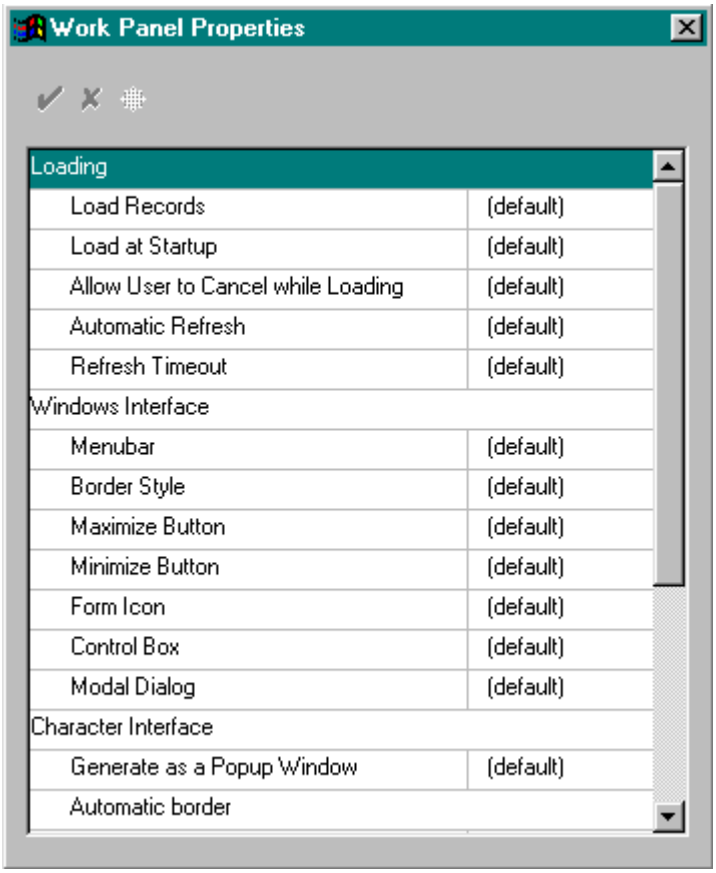


De esta manera cuando el usuario presiona el botón de graficar visualizara el total pedido por los proveedores en forma gráfica. Vemos a continuación ejemplos de los tipos de gráficos que podemos diseñar:


Grphdbf.olegraph
GX
Totales por proveedor
ABC Inc.
GX Corp.
A.C.M.E
800
700
600
500
400
300
200
100
0
ABC Inc.
GX Corp.
A.C.M.E
PROVEEDOR MAYOR


Grphdbf.olegraph
Totales por proveedor
GX
ABC Inc.
GX Corp.
A.C.M.E

Properties

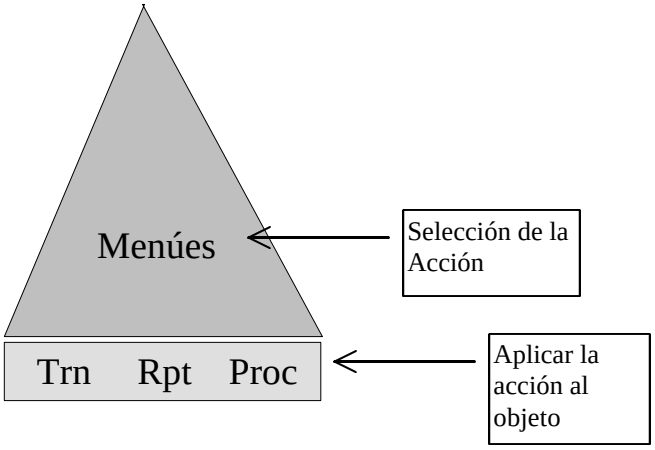


Generar como una ventana Popup

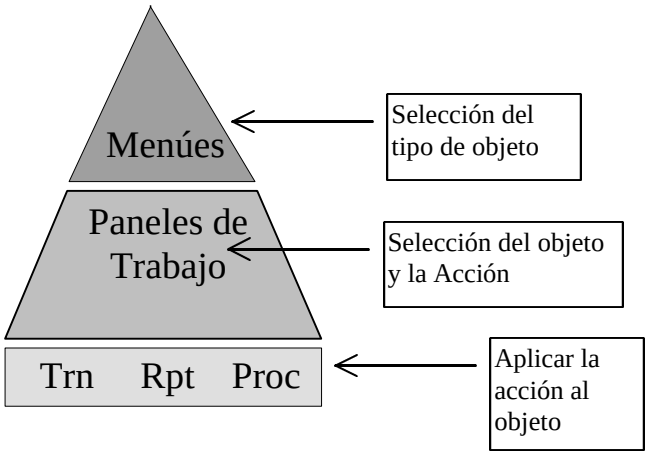
Se utiliza para indicar que el Panel de Trabajo debe cargarse como una ventana y no borrar la pantalla anterior, tiene sentido para ambientes que no tienen interfaces gráficas.
En el segundo Panel de Trabajo del ejemplo se utilizó esta regla.
Por mayor información sobre las propiedades de los distintos objetos sugerimos consultar al manual de referencia.

Diálogos Objeto-Acción

Con el uso de Paneles de Trabajo, se tiene la oportunidad de definir para la aplicación una arquitectura orientada a diálogos Objeto-Acción. Pero previo a introducir este concepto veamos cual era la forma tradicional.
Si observamos una aplicación desarrollada por los métodos tradicionales, veremos que cuanto mayor es, más grande será el árbol de menúes que se tiene para acceder a los programas que efectivamente trabajan con los datos. La arquitectura de este tipo de sistemas normalmente es:



Esta situación se puede apreciar claramente en los sistemas con muchas consultas. En general, todas estas consultas dependen de un menú, pero no se encadena una consulta con las otras. Y al no estar encadenadas, se da la paradoja de que cuanto más consultas se implementan en el sistema, menos son usados por el usuario, porque le resulta difícil acordarse de las diferencias entre ellas.
Diremos que este tipo de aplicación tiene un diálogo Acción-Objeto, porque primero se elige la acción a realizar (por ejemplo modificar un proveedor, o listar los pedidos) y luego se elige el objeto al cual aplicar la acción (que proveedor o los pedidos de que fecha).
Con Paneles de Trabajo se pueden implementar sistemas en donde el diálogo puede ser Objeto-Acción, donde primero seleccionamos el objeto con el cual vamos a trabajar y luego las acciones que aplicamos al mismo. En este caso la arquitectura será:



De esta manera el usuario, una vez que selecciona el tipo de objeto con el que va a trabajar, ya puede utilizar el Panel de Trabajo que le permite seleccionar el objeto en sí, y luego, las acciones a aplicar sobre el mismo.
Esta arquitectura permite desarrollar aplicaciones complejas, pero que se mantienen fáciles de usar.

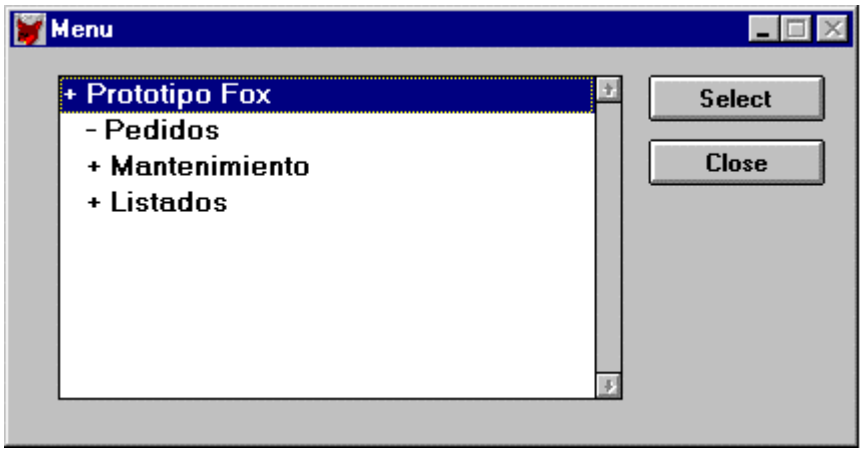
DISEÑO DE MENÚES

Este objeto permite definir los diferentes menúes de la aplicación. La definición de un menú es muy fácil, y prácticamente lo único que se requiere es definir las opciones del mismo.

Ejemplo:

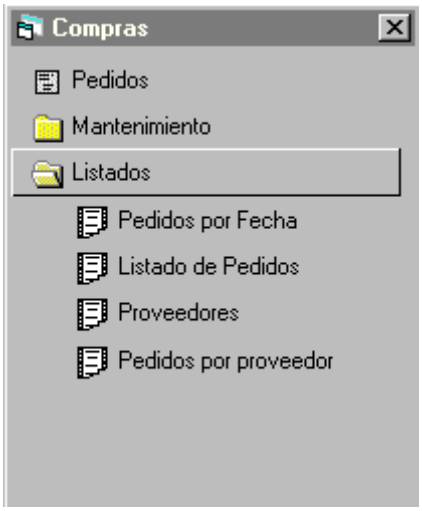
Menú Principal del Sistema de Compras

El menú anterior cuando es generado para FoxPro Windows es:

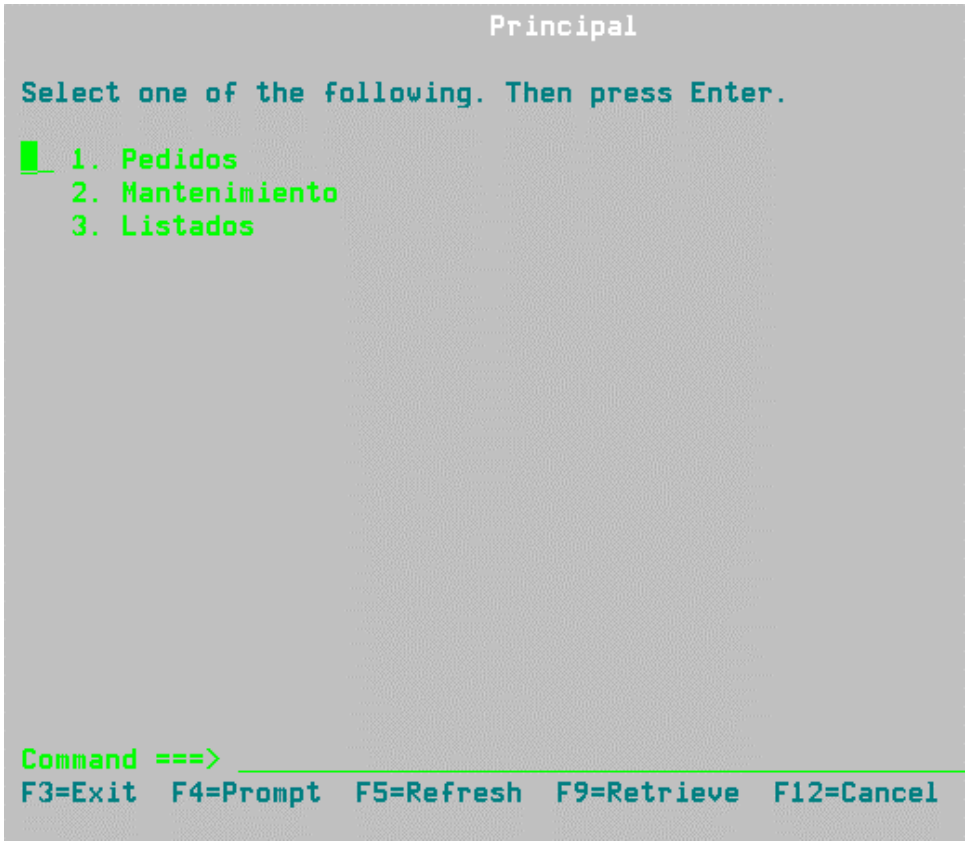


Es un menú de selección implícita, es decir la opción se selecciona directamente con el Mouse.

El mismo menú generado para Visual Basic tiene el siguiente formato:



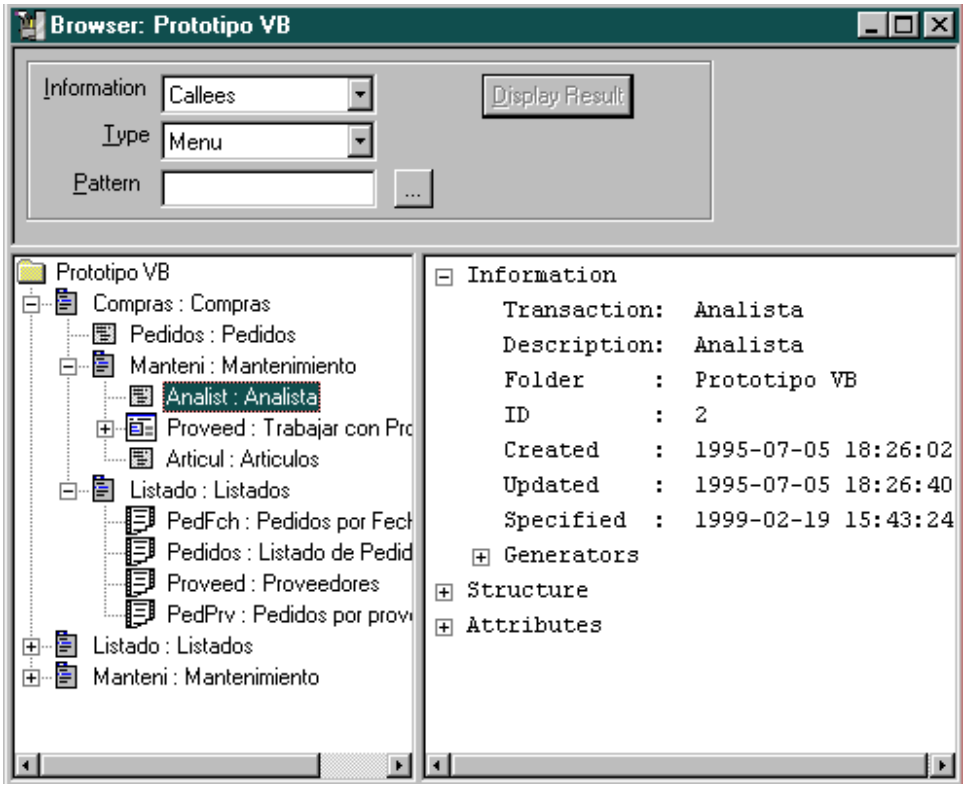
El mismo menú generado para el ambiente AS/400 tiene el siguiente formato:



En donde se utiliza la técnica de selección explícita, donde para seleccionar la opción se debe digitar el número de la misma en el campo de selección.
Los menúes generados para AS/400 siguen las especificaciones CUA 89 (Common User Access) de IBM.

Call Browser

Este browser despliega todos los objetos que se llaman a través de los comandos call y udp.
Dado un programa podemos ver su árbol de llamadas (a que programas llama) y desde que programas es llamado (eligiendo la opción Callees o Callers del dialogo del browser).
En nuestro ejemplo vamos a ver el árbol de llamadas del menú Compras:



Desde este browser podemos editar cualquier objeto que aparezca en las listas.

ANEXO A. MODELOS DE DATOS RELACIONALES

El propósito de este Anexo es explicar una serie de conceptos sobre Bases de Datos Relacionales relevantes para el uso de **GENEXUS**.
Un Modelo de Datos esta compuesto por:

Tablas

En una Base de Datos Relacional la única forma de almacenar información es en Tablas.
Una Tabla es una matriz (tiene filas y columnas) con un nombre asociado (ej. PROVEEDO). A las columnas les llamaremos Atributos, y a las filas Registros o Tuplas. Por ejemplo la Tabla de Proveedores:

PROVEEDO

PrvCod	**PrvNom**	**PrvDir**
125	ABC Inc.	Xxxxxxxxx
126	GXXXX Corp.	Yyyyyyy

Una Tabla se diferencia de un archivo convencional porque debe cumplir las siguientes reglas:

- Todos los atributos representan la misma información para cada una de las filas (o registros). Es decir en el atributo *PrvNom* se almacenan los nombres de los proveedores, y no puede ocurrir que para algunos proveedores en ese atributo se almacene la dirección del mismo. En la práctica esta regla implica que no existen tablas con diferentes tipos de registros.
- No existen dos registros con exactamente la misma información.
- El orden de los registros no contiene información. Es decir se pueden reordenar los registros sin perder información.

Clave primaria y Claves candidatas

Dado que no puede haber dos registros con la misma información, siempre se puede encontrar en una tabla el atributo o conjunto de atributos cuyos valores no se duplican. Se llama a ese atributo o conjunto de atributos Clave Primaria de la tabla.
En realidad pueden existir varios de esos conjuntos, a cada uno de ellos lo llamamos Clave Candidata. Por ejemplo en la tabla PROVEEDO tanto *PrvCod* como *PrvNom* son claves candidatas, ya que no existen dos proveedores con el mismo código, pero tampoco hay dos proveedores con el mismo nombre.
En una Base de Datos Relacional sólo una de las claves candidatas debe ser definida como la Clave Primaria. También se debe respetar la siguiente regla: no deben existir dos tablas con la misma clave primaria.

Por ejemplo consideremos el siguiente diseño:

Tabla: **PROVEEDO**

Atributos:
*PrvCod**
PrvNom

Tabla: **PROVDIR**

Atributos:
*PrvCod**
PrvDir

Tanto la PROVEEDO como la PROVDIR tienen la misma clave primaria. Esto era relativamente común cuando el criterio de diseño era separar las actualizaciones (por ejemplo dado que *PrvDir* cambia mas que *PrvNom*, entonces es mejor definir un archivo diferente en donde el registro que se modifica es menor, y por lo tanto de mejor performance). Sin embargo en una Base de Datos Relacional el criterio de diseño es diferente (ver sección de Normalización) y es preferible una sola tabla con los tres atributos:

Tabla: **PROVEEDO**

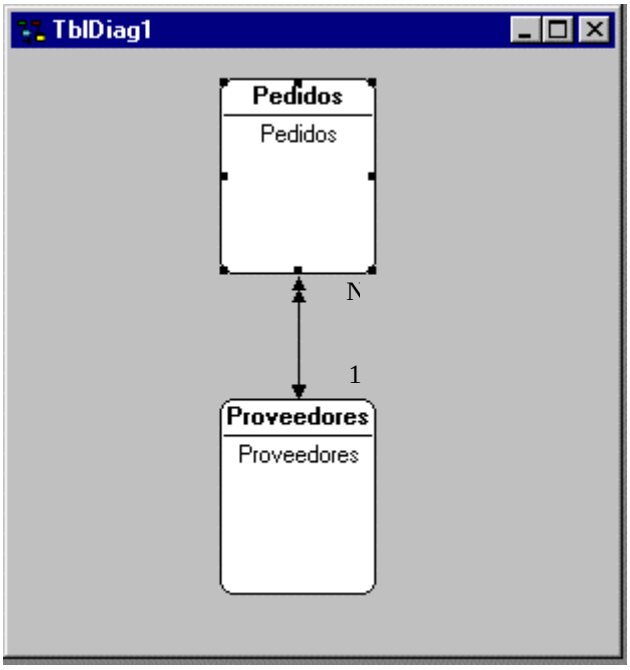
Atributos:
*PrvCod**
PrvNom
PrvDir

Indices

Son vías de acceso eficientes a las Tablas. Existen tres tipos de índices: Primarios, Extranjeros y del Usuario.
El índice primario es el que se define para la Clave Primaria y, por lo tanto, se usa en el control de unicidad de los registros. Con **GENEXUS** todos los índices primarios son definidos automáticamente a partir de los identificadores de las transacciones.
Los índices extranjeros son utilizados para hacer eficientes los controles de integridad inter-tablas. También son definidos automáticamente.
Los índices del usuario se definen, fundamentalmente, para recorrer los datos por determinado orden de una forma eficiente. Por ejemplo en la tabla PROVEEDO se puede definir un índice por *PrvNom*, que es muy útil para los listados ordenados por nombre. Los índices del usuario NO son definidos automáticamente.
En una Base de Datos Relacional los índices se utilizan sólo por problemas de performance, pero siempre es posible acceder a los datos de la tabla por cualquiera de los atributos de la misma.

Integridad Referencial

Son las reglas de consistencia entre los datos de las distintas tablas.
Según vimos en el ejemplo, existe una relación entre la tabla PEDIDOS (Pedidos) y la PROVEEDO (Proveedores):



La relación entre estas dos tablas se determina analizando los atributos que tienen en común, por ejemplo aquí tenemos que el atributo común es *PrvCod*, que es la clave primaria de la tabla PROVEEDO (Proveedores) y se encuentra en la tabla PEDIDOS (Pedidos) y, por lo tanto, ambas tablas están relacionadas y la relación es 1-N.
Con relación 1-N se indica que un Proveedor tiene varios Pedidos y que un Pedido sólo tiene un Proveedor. También es usual decir que la tabla de proveedores esta superordinada a la de pedidos, y la de pedidos esta subordinada a la de proveedores.
Esta relación implica que los datos de ambas tablas no son independientes y cada vez que se realicen modificaciones en una de las tablas, se deben tener un cuenta los datos de la otra tabla. A estos controles se les llama de integridad referencial y son:

- Cuando se elimina un registro en la tabla superordinada (PROVEEDO), no deben existir registros correspondientes en la tabla subordinada (PEDIDOS).
- Cuando se crean registros en la tabla subordinada (PEDIDOS), debe existir el registro correspondiente en la tabla superordinada (PROVEEDO).

Para hacer eficiente el primer control se utiliza el índice extranjero por *PrvCod* en la tabla PEDIDOS y para el segundo el índice primario también por *PrvCod* en la tabla PROVEEDO.

El diagrama anterior se ha inspirado en los conocidos Diagramas de Bachman, y tiene justamente la virtud de explicitar la integridad referencial del modelo de datos.

Normalización

El proceso de normalización determina en que tablas debe residir cada uno de los atributos de la base de datos.
El criterio que se sigue es básicamente determinar una estructura tal de la base de datos, que la posibilidad de inconsistencia en los datos sea mínima.
Por ejemplo, si tenemos que almacenar información sobre Proveedores y Pedidos, se podría tener la siguiente estructura:

Tabla: **PROVEEDO**	Tabla: **PEDIDOS**
*PrvCod**	*PedNro**
PrvNom	*PedFch*
PrvDir	*PrvCod*
	PrvNom
	AnlNro
	PedTot

en donde vemos que ambas tablas tienen dos atributos en común (*PrvCod* y *PrvNom*). En el caso de *PrvCod* es necesario que se encuentre en ambas tablas dado que es el identificador de los Proveedores, y por lo tanto debe estar en la de Pedidos para indicar a que Proveedor corresponde el Pedido.
La situación es diferente para *PrvNom*, ya que no es necesario que esté en la tabla de Pedidos, porque si conocemos el *PrvCod* podemos determinar cual es su *PrvNom* (basta con buscar en la tabla PROVEEDO por la clave primaria). Si de cualquier manera se almacena el *PrvNom* en la tabla PEDIDOS tenemos que esta estructura tiene más posibilidades de inconsistencia que si no estuviera allí. Por ejemplo si un Proveedor cambia su *PrvNom*, el programador debe encargarse de tener un programa que lea todos los Pedidos de ese Proveedor y le ponga el nuevo *PrvNom*.

Entonces la estructura correcta (diremos 'normalizada') será:

Tabla: **PROVEEDO**	Tabla: **PEDIDOS**
*PrvCod**	*PedNro**
PrvNom	*PedFch*
PrvDir	*PrvCod*
	AnlNro
	PedTot

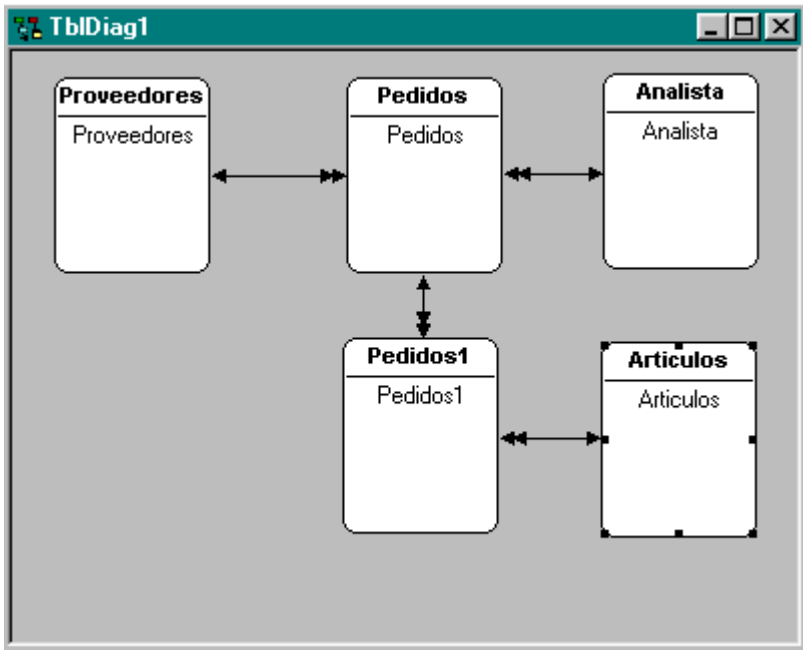
El proceso de normalización (que se acaba de introducir de una manera muy informal) se basa en el concepto de Dependencia Funcional, cuya definición es:
Se dice que un atributo depende funcionalmente de otro si para cada valor del segundo sólo existe UN valor del primero.
Por ejemplo *PrvNom* depende funcionalmente de *PrvCod* porque para cada valor de *PrvCod* sólo existe UN valor de *PrvNom*.
En realidad la definición es algo mas amplia, ya que se define que un conjunto de atributos dependa funcionalmente de otro conjunto de atributos.

Tabla extendida

Como vimos en la sección de Normalización, el criterio de diseño de la base de datos se basa en minimizar la posibilidad de inconsistencia en los datos. Una base de datos diseñada de esta manera tiene una serie de ventajas importantes (tal es así que actualmente la normalización de datos es un standard de diseño), pero se deben tener en cuenta también algunos inconvenientes.
El inconveniente más notorio es que los datos se encuentran dispersos en muchas tablas, y por lo tanto cuando se quieren hacer consultas mas o menos complejas a la base de datos se debe consultar una cantidad importante de tablas. Así, por ejemplo, para listar los Pedidos por Proveedor es necesario consultar las tablas PROVEEDO y PEDIDOS.
Para simplificar esta tarea ***GENEXUS*** utiliza el concepto de Tabla Extendida, cuya definición es:

> Dada una tabla de la base de datos, la tabla extendida de la misma son los atributos que pertenecen a la tabla y a todas las tablas que tengan una relación N-1 (directa o indirectamente) con la primera. A la primera tabla se le llama de Tabla Base y a las otras de Tablas N-1.

Si hacemos un diagrama de Bachman del modelo de datos del ejemplo:



vemos que la tabla extendida de Pedidos comprende a todos los atributos de la tabla Pedidos, más todos los atributos de Proveedores y todos los de Analistas, pero no los atributos de la Linea del pedido o de Artículos (porque si bien están relacionados, su relación no es N-1).

En el siguiente diagrama se muestra, para cada una de las tablas del modelo, cual es su tabla extendida:

Tabla base	**Tabla extendida**
Proveedores	Proveedores
Analistas	Analistas
Pedidos	Pedidos, Proveedores, Analistas
Linea_pedidos	Linea_pedidos, Pedidos, Proveedores, Analistas, Artículos
Artículos	Artículos

El concepto de tabla extendida es muy utilizado en ***GENEXUS***, sobre todo en Reportes, Procedimientos y Paneles de Trabajo en donde siempre se trabaja con tablas extendidas y no con tablas físicas. Por ejemplo el comando FOR EACH determina una tabla extendida y no una tabla física.

ANEXO B. FUNCIONES, REGLAS Y COMANDOS

FUNCIÓN	DESCRIPCIÓN
MANEJO DE FECHAS	
Day	Numero de día de una fecha
Month	Numero de mes de una fecha
Year	Año de una fecha
Today	Fecha de hoy
Dow	Numero del día de la semana
Cdow	Nombre del día de una fecha
Cmonth	Nombre del mes de una fecha
Ctod	Pasar de tipo carácter a tipo fecha
Dtoc	Pasar de tipo fecha a carácter
Ymdtod	Armar fecha a partir de año, mes y día
Addmth	Sumar un mes a una fecha
Addyr	Sumar un año a una fecha
Age	Diferencia en años entre dos fechas
Eom	Fecha fin de mes de una fecha
MANEJO DE STRINGS	
Str	Pasar un numérico a string
Substr	Substring de un string
Concat	Concatenar dos strings
Space	Definir un strings con “n” blancos
Len	Largo de un string
Trim	Quita los caracteres en blanco de un string
Ltrim	Quita los caracteres en blanco al principio de un string
Rtrim	Quita los caracteres en blanco al final de un string
Upper	Convierte un string a mayúsculas
Lower	Convierte un string a minúsculas
Int	Parte entera de un numérico
Round	Redondear un valor numérico
Val	Pasar un string a numérico
OTRAS	

Old	Valor del atributo antes de ser modificado
Previous	Valor del atributo en la última inserción
Time	Hora
Systime	Hora del sistema
Sysdate	Fecha del sistema
Userid	Identificación del usuario
Usercls	Clase de usuario
Wrkst	Workstation
Ask	Para pedir valores por pantalla
Udf	Función definida por el usuario
Udp	Procedimiento definido por el usuario
Null(<Att|Var>)	Si un atributo tiene valor nulo o no
Nullvalue(<Att|Var>)	Devuelve el valor nulo de un atributo
LoadBitmap	Carga un bitmap
RGB (<R>,<G>,<B>)	Retorna el numero que representa al color RGB determinado por los 3 parametros
Color(<GXColor>)	Retorna el numero que representa al color pasado como parametro

ORDEN DE DISPARO DE LAS REGLAS EN LAS TRANSACCIONES	
After(Insert)	Luego que se inserto el registro
After(Update)	Luego que se actualizo el registro
After(Delete)	Luego que se dio de baja el registro
After(Confirm)	Luego que se confirmo el nivel (todavía no se hizo la modificación del registro)
After(Trn)	Luego que termina la transacción
After(<Att>)	Después de un atributo
After(Level(<Att>))	Luego que finaliza determinado nivel
Level(<Att>)	Nivel de una transacción
Modified()	Si fue modificado algún atributo de un nivel
Insert	Si la transacción esta en modalidad insert
Update	Si la transacción esta en modalidad update
Delete	Si la transacción esta en modalidad delete

REGLAS	DESCRIPCIÓN	T	W	R	P
Default	Dar un valor por defecto a un atributo				
Equal	Asigna un valor en modalidad insert y funciona como filtro en update y delete.				
Add	Sumar un valor a un atributo teniendo en cuenta la modalidad de la transacción				
Subtract	Restar un valor a un atributo teniendo en cuenta la modalidad de la transacción				
Serial	Numerar serialmente atributos				
Msg	Desplegar un mensaje				
Error	Dar un error				
Noaccept	No aceptar atributo o variable en la pantalla				
Call	Llamar a un programa				
Submit	Someter una llamada a un programa				
Parm	Para recibir los parámetros				
Nocheck	No realizar el control de integridad referencial				
Allownulls	Permitir ingresar valores nulos				
Refcall	Hacer un call cuando falla el control de integridad referencial				
Refmsg	Mensaje que se despliega cuando falla el control de integridad referencial				
Prompt	Asociar al prompt un objeto definido por nosotros				
Noprompt	Inhibe la lista de selección				
Accept	Aceptar atributo o variable en la pantalla				
Default_mode	Modalidad por defecto al ingresar en una transacción				
Noconfirm	No pedir confirmación al final de un determinado nivel				
Color	Dar colores a los atributos, variables y fondo				
Order	Orden de la tabla base del Work Panel				
Search	Condición de búsqueda sobre el subfile				
Hidden	Para incluir variables o atributos en el subfile				
Noread	Inhibe la lectura de atributos				
Printer	Para seleccionar el printer file en el				

	AS/400				

T = Transacciones
W = Work Panels
R = Reportes
P = Procedimientos

Las casillas que aparecen marcadas indican que la función aparece disponible para ese objeto.

COMANDOS	DESCRIPCIÓN	T	W	R	P
Header	Comienzo del cabezal				
Footer	Comienzo del pie de pagina				
For each	Para recorrer los registros de una tabla				
Exit	Permite salir de un while				
Do while	Repetición de una rutina mientras se cumpla una condición				
Do Case	Ejecuta algún bloque según que condición se cumpla				
If	Ejecuta un bloque de comandos en caso que la condición evalúe verdadera				
Commit	Commit				
Rollback	Rollback				
Print if detail	Imprime si existen registros en el subfile.				
Return	Finaliza la ejecución y vuelve al programa llamador				
&<a> = <Exp>	Asignación de una variable				
Lineno	Numero de línea donde los datos serán impresos				
Prncmd	Para pasar comandos especiales a la impresora				
Pl	Largo de pagina				
Mt	Margen superior				
Cp	Provoca un salto de página si quedan menos de “n” líneas por imprimir				
Noskip	No pasar a la siguiente línea de impresión				
Eject	Salto de página				
Call	Llamar a un programa				
Submit	Someter un programa				
Msg	Desplegar un mensaje				
Do	Llamar a una subrutina				
Sub - EndSub	Bloque de subrutina				
ATT = <Exp>	Actualización de atributo				
New - EndNew	Insertar registro				
Delete	Borrar un registro				

T = Transacciones (eventos)
W = Work Panels (eventos)
R = Reportes
P = Procedimientos

Las casillas que aparecen marcadas indican que la función aparece disponible para ese objeto.