

CLARION 5TM

Primeros Pasos

COPYRIGHT 1994, 1995, 1996; 1997 by TopSpeed Corporation

COPYRIGHT 1998, © Unisoft S. R. L.

Todos los derechos reservados

Esta obra está protegida por el derecho de autor y todos los derechos están reservados. No puede ser copiada, fotocopiada, reproducida, traducida o pasada a medios electrónicos o informáticos sin consentimiento, por escrito, de los titulares de los derechos.

Esta publicación acompaña a Clarion 5. Puede contener errores técnicos o tipográficos. TopSpeed y Unisoft la entregan “como está”, sin garantías, explícitas o implícitas.

TopSpeed Corporation
150 East Sample Road
Pompano Beach, Florida 33064
(+1-954)785-4555

Unisoft S.R.L.
Uruguay 263, piso 5
1015 - Buenos Aires
Tel./Fax: (54-11) 4372-7243/9850; 4374-2298/9469

Traducción: Héctor Daniel Calabia, Belca
Armado: Lilia G. B. de Calabia, Belca. Tel. 4774-6845. Fax: 4777-6683

Revisión y actualización: RC Sistemas. Salta – Argentina. Año 1.999.

Reconocimiento de marcas:

TopSpeed[®] es marca registrada de TopSpeed Corporation.

Clarion5[™] es marca comercial de TopSpeed Corporation.

Btrieve[®] es marca registrada de Btrieve Technologies.

Microsoft[®], Windows[®], y Visual Basic[®] son marcas registradas de Microsoft Corporation.

Todos los demás productos y nombres de empresas son marcas comerciales o registradas de sus respectivos propietarios.

Impreso en la Argentina - Printed in Argentina.

ÍNDICE

1 – INTRODUCCIÓN	7
¡Bienvenido!	7
Niveles de usuario/desarrollador	7
Qué hay en esta sección	9
Dónde encontrar más información	10
Convenciones en la documentación	12
Convenciones tipográficas	12
Convenciones del teclado	12
Información sobre el producto	13
Registro	13
Apoyo Técnico	13
Sistema de respuestas por fax	13
El programa de instalación: Setup	14
Requerimientos del sistema	14
Iniciando el Setup	14
Opciones de instalación	15
Ejecución de Clarion	16
2 – INICIO RÁPIDO: CURSILLO	17
Asistente de Inicio Rápido (Quick Start Wizard)	18
Creación de un diccionario y de una aplicación	18
Asistente de carga rápida (Quick Load Wizard)	23
Añadir un archivo	23
Añadir una relación	25
Asistentes para procedimientos (Procedure Wizards)	28
Uso del Asistente para browse	30

¡Y, finalmente, el Asistente para Aplicaciones!	34
Usar el Dictionary Editor para configurar las opciones del Asistente	34
Uso del Asistente para aplicaciones	35
Qué sigue	38

3 – PANORAMA DEL PROCESO DE DESARROLLO **39**

El Tao de Clarion	39
Clarion es un lenguaje de programación	39
Niveles de abstracción	39
Niveles de abstracción de Clarion	41
Programación controlada por templates	42
El entorno de desarrollo Clarion	45
¿Qué sigue ahora?	48

1 - INTRODUCCION

¡Bienvenido!

Muchas gracias por adquirir Clarion. Tiene ahora en sus manos la herramienta más poderosa y flexible para el desarrollo de aplicaciones. Muy pronto, usted podrá apreciarlo por sí mismo.

Niveles de usuario/desarrollador

Clarion funciona a varios niveles, y respalda a todo nivel de usuario o desarrollador.

Es un sistema de desarrollo automático, manejado mediante asistentes (wizards)

Cuando usted necesita una aplicación “simple” para el mantenimiento de una base de datos, el trabajo le llevará sólo algunos minutos usando Clarion. La clave está en el diccionario de datos. Si el Generador de aplicaciones (*Application Generator*) conoce los archivos o tablas que usted necesita, y cómo se vinculan, puede construir una aplicación completa. De manera que todo lo que usted tiene que hacer es elegir uno o más archivos, e indicar el tipo de relaciones que tienen entre sí (si son dos o más). En el breve cursillo de **Inicio rápido** de este libro encontrará una demostración de lo fácil que es todo el proceso.

Así, el Asistente para Aplicaciones (*Application Wizard*) puede crear una aplicación completa usando el concepto genérico de las aplicaciones Clarion. Lo llamamos el concepto *browse-form* (“procedimiento de visualización/ficha”), que se extiende directamente desde la estructura misma de la base de datos. La aplicación funciona así: (1) El usuario final navega por la base de datos -- por toda ella o parcialmente, en un archivo de datos o múltiples bases de datos relacionadas-- desplazándose por una lista, en la que cada ítem representa un registro. La ventana en que esto se realiza se llama “*browse*” (pronúnciese /bráus/). (2) El usuario selecciona un registro de la lista para realizar una acción, como modificar los datos. Esto generalmente se hace en una ventana separada, en que los campos de la base de datos aparecen en cajas de edición separadas. Esto se llama una “ficha o formulario de actualización” (*update “form”*). Una ficha puede también aceptar datos nuevos. (3) Opcionalmente, el usuario final puede consultar un valor de una tabla vinculada durante el ingreso de datos a la ficha. Esto abre otro *browse* en una ventana separada. El usuario puede elegir un ítem, cerrando la ventana y colocar el valor en la caja de edición de la ficha. Esto

se llama “consulta” (*lookup*). Esto sería la descripción más genérica. Además, cada ventana de visualización o *browse*, que puede navegarse desde la barra de herramientas, se abre en un hilo de ejecución separado, con su propio *buffer* de registro, lo que permite un manejo de datos más seguro. Se pueden seleccionar múltiples ordenamientos por claves (“*keys*”) dando CLIC sobre una lengüeta. Un DOBLE CLIC sobre un registro de la lista abre una ficha de actualización, con comprobación automática de uso concurrente (para aplicaciones multiusuario) y, opcionalmente, restricciones de integridad referencial, que mantienen las relaciones de la base de datos.

Cualquiera puede hacer todo esto. Se empieza sencillamente, escogiendo un archivo de datos de la lista.

Es un ambiente de desarrollo visual

Con Clarion, colocar un control en una ventana hace mucho más que en otras herramientas rápidas para el desarrollo de aplicaciones, que típicamente una vez colocado el control para la interfaz con el usuario, esperan que *usted* escriba el código para hacerlo funcionar. Con Clarion, usted añade un *template* o plantilla, que contiene el control y todos los elementos de datos requeridos y el código ejecutable. Esto significa que no hay que escribir el código, basta un CLIC para obtener una solución completa: el control de interface del usuario, y el código que lo hace trabajar. Es más, cada *template* tiene su propia interfaz de usuario. Cuando se observan las propiedades del *template*, se encuentra la lengüeta “Actions” (acciones). Mediante acciones como responder a una caja de diálogo, escoger un ítem de una lista desplegable, o llenar un casillero, puede modificarse el funcionamiento del *template* de modo que responda exactamente a las necesidades de la aplicación. Durante el cursillo que aquí presentamos, habrá muchas oportunidades para configurar estas acciones. El cursillo sobre el **Generador de aplicaciones** en la sección *Aprendiendo Clarion* lo introduce a todas las herramientas de desarrollo rápido de aplicaciones (RAD) que tiene el programa.

Cuando se usa la interfaz de los *templates* para especificar tales comportamientos, el Generador de aplicaciones escribe el código fuente Clarion que los implementa. Se trata de un código orientado a objetos, construído de la biblioteca de clases de aplicación de Clarion (*Application Builder Class [ABC] Library*), por lo que es un código compacto y muy eficiente. Mediante los *templates*, usted puede realizar una enorme cantidad de programación totalmente configurada a sus necesidades, sin necesidad de escribir ni una sola línea de código a mano.

La magnitud y el alcance del lenguaje Clarion pueden parecer imponentes a primera vista. La Referencia del Lenguaje (*Lenguaje Reference*) tiene unas 1000 páginas. Una de las grandes ventajas de tener el sistema de *templates*, adicionalmente a la capacidad de escribir código manualmente, es que usted puede *irse introduciendo* al lenguaje, al ritmo y profundidad que desee. Si no le agrada el modo en que los *templates* resuelven un problema en particular, puede usarlos al principio, sólo para salir del paso. Luego, puede utilizar la interfaz del propio *template* para hacerlo más a su gusto. Finalmente, cuando domina el lenguaje Clarion, puede escribir una solución propia, con completa libertad. (Nota: además puede comprar *templates* de terceras partes para tener otro modo de resolver los problemas).

Los usuarios avanzados de computadoras que normalmente no escriben programas, pueden hacer esto con facilidad.

Es un lenguaje de programación completo

En su nivel más básico, se puede codificar una aplicación completamente a mano, utilizando el lenguaje Clarion de cuarta generación. Clarion tiene un alto nivel de abstracción, lo que lo hace muy “legible”, y una sintaxis de base de datos compacta, de manera que fácilmente puede manipularse un registro, o grupos de éstos. No necesita *saber* Clarion para crear una aplicación... pero sí conocer como trabaja, le ayudará a comprender lo que hacen las aplicaciones, lo que mejorará su calidad. El cursillo sobre el **Generador de aplicaciones** en la sección *Aprendiendo Clarion*, lo estimula a escribir una pequeña porción de código Clarion dentro de la fuente generada por el *template*, y el cursillo sobre el **Lenguaje Clarion** (en la misma sección) lo introduce plenamente a la codificación manual.

Los desarrolladores profesionales apreciarán realmente el lenguaje Clarion. Fue diseñado desde el vamos para programadores de aplicaciones de negocios. Su facilidad de aprendizaje (en comparación a otros lenguajes de bajo nivel) no le impide tener una performance deslumbrante. El compilador de TopSpeed convierte todo el código, sea escrito a mano o mediante el Generador de aplicaciones, en código binario altamente optimizado.

NOTA: No importa en qué nivel intente trabajar finalmente: lo hará mucho mejor si practica todos los cursillo hasta el final.

Qué hay en esta sección

Esta es la introducción al entorno Clarion. La siguiente es una lista de partes de esta sección, y resume su contenido:

Introducción

Capítulo uno: presenta Clarion y las instrucciones para cargarlo en la computadora.

Cursillo para un rápido comienzo

Capítulo dos: unos pocos pasos fáciles con el Asistente (*Wizard*) de Comienzo Rápido le permiten crear una aplicación completa en *cinco minutos*. Luego, *en menos de una hora*, este capítulo le enseñará a usar el *Wizard de Aplicaciones* y los *wizards* de procedimientos para crear una aplicación simple, y luego extender su funcionalidad a un programa elaborado que trabaja con dos archivos. Esto se logrará *sin escribir ni una sola línea de código*.

Flujo de desarrollo

Capítulo tres: un repaso del concepto Clarion de desarrollo de aplicaciones. Describe todas las herramientas del ambiente de desarrollo.

Dónde encontrar más información

El primer lugar para buscar es la sección **How do I ... ?** (“Cómo hago para ...?”). Esos temas responden muchas de las preguntas comunes que se hacen los recién llegados a Clarion. Oprima el botón **How do I ... ?** en la sección de Help Contents para ir a esa sección.

Clarion tiene un muy amplio sistema de ayuda. La ayuda hipertextual aparece cuando oprime la tecla F1, o elige una de las ordenes del menú Help. La ayuda en línea se maneja mediante la caja de diálogo, para ofrecer ayuda sensible al contexto cuando la necesita.

También hay un número de libros que se incluyen con Clarion.

Primeros pasos con Clarion

Es este volumen, que suma dos volúmenes separados del original en inglés: *Getting Started* (esta sección) y

Learning Clarion (Sección: Aprendiendo Clarion)

Esta última sección contiene *dos* cursillo paso a paso. El primero es sobre el Generador de aplicaciones, que ofrece una introducción amplia a todas las herramientas del entorno de desarrollo. La segunda presenta el lenguaje de programación Clarion en sí.

Application Handbook (Manual de las aplicaciones)

Es el manual de las herramientas Clarion que utilizará más durante el desarrollo de aplicaciones (en el formato .PDF en el CD de la edición estándar). Describe los *templates* que se incluyen con el producto, brinda una documentación completa de la biblioteca de clases de construcción de aplicaciones (*ABC Library*) que utiliza el código generado por los *templates*, y ofrece información profunda sobre los controladores (*drives*) de archivos de Clarion.

Language Reference (Referencia del lenguaje)

La guía completa al lenguaje Clarion (en el formato .PDF en el CD de la edición estándar). Este libro ofrece descripciones de la sintaxis completa de Clarion, y todas las órdenes disponibles, con ejemplos de cada una. El *Language Reference* se organiza mediante tópicos funcionales. El texto completo de este libro también está disponible en la ayuda en línea. Cuando se trabaja en el Editor de texto, coloque el punto de inserción sobre cualquier función o sentencia Clarion, luego oprima F1 para obtener ayuda sobre ese tema.

User's Guide (Guía del usuario)

Provee una descripción clasificada por distintas tareas del entorno de desarrollo, ordenada por los componentes principales (en formato .PDF en el CD de las ediciones estándar y profesional).

Programmer's Guide (Guía del programador)

Colección de artículos que tratan en profundidad varios aspectos de la programación en lenguaje Clarion y artículos sobre la personalización del entorno de desarrollo (disponible solamente en las ediciones profesional y empresarial, en formato .PDF en el CD de la edición profesional). También contiene la guía completa al lenguaje de *templates* de

Clarion, lo que ofrece descripciones de todas las sentencias y funciones, con ejemplos de cada una, con demostraciones claras de cómo escribir *templates* propios.

ReportrWriter User's Guide (Guía del usuario del ReportWriter)

Documenta el ReportWriter for Windows™ de Clarion. Esta es una herramienta independiente para la confección de informes (incluída solamente en las ediciones profesional y empresarial -en formato .PDF en el CD de la edición profesional), que también está disponible como producto separado.

Enterprise Tools (Herramientas empresariales)

Documenta las herramientas disponibles para desarrollos cliente/servidor (Data Dictionary y Synchronizer), desarrollo en grupo (Version Control / Team Developer) y las herramientas Data Modeller. Documenta además la *Business Math Library* (Biblioteca de funciones de matemática empresarial) y *Wise for Clarion* (herramienta para usuario final que permite crear instalaciones). Este manual está incluido solamente en la versión empresarial (algunas componentes también están disponibles como producto separado).

Wizatron Handbook.

Documenta los wizatrons, generación avanzada de la tecnología de asistentes Clarion, quienes “aprenden” cómo usted trabaja, y se convierten en su asistente de programación automatizado (sólo disponible con la edición empresarial).

Master Index (Índice general)

Un listado completo de índice para toda la documentación impresa y tipo .PDF que viene con cada edición de Clarion (impresa solamente en la edición empresarial, en formato .PDF del CD en todas las demás ediciones).

Todos los libros que están en formato .PDF en el CD (o en disquetes separados, en el caso de las versiones en español) son accesibles mediante el programa *Adobe Acrobat Reader*, que también puede instalarse desde el CD de Clarion. Pueden hacerse búsquedas por tema o palabra en estos archivos .PDF, y también pueden imprimirse, si se desea.

Importante: si cualquier parte del texto on-line contradice la documentación impresa, la documentación en línea tiene precedencia. TopSpeed Corporation hace el mayor esfuerzo por asegurar que la documentación impresa esté al día. Sin embargo, el tiempo requerido por los impresores puede causar algunas inexactitudes: podemos actualizar los archivos de ayuda hasta el lanzamiento mismo de una revisión del producto; pero los materiales impresos deben “alcanzarlos” después.

Convenciones en la documentación

Convenciones tipográficas:

Bastardillas

Indican qué debe tipearse en el teclado, como por ejemplo *Escriba esto*. También, ocasionalmente, se utilizan para destacar algunas palabras o frases, o indicar expresiones en idioma extranjero.

VERSALITAS

Indican pulsaciones de teclas, como ENTER o ESCAPE, o dar un CLIC con el *mouse*.

Negritas

Indican órdenes u opciones de un menú, o del texto en una ventana de diálogo. Note: este estilo también puede usar una familia tipográfica diferente, para igualarse a las negritas en Helvética que utiliza Windows como fuente del sistema.

LETTER GOTHIC

Se utiliza para diagramas, listados de código fuente, para comentar ejemplos, y para ejemplos del uso de sentencias de código.

Convenciones del teclado

Fl

Indica una única pulsación de tecla. En este caso, oprima y suelte la tecla Fl.

ALT + X

Indica una pulsación combinada. En este caso, oprima la tecla ALT y seguidamente pulse la tecla X, luego suelte ambas teclas.

Información sobre el producto

Registro

Antes de comenzar a usar Clarion, llene y envíe la tarjeta de registro que se encuentra en la caja a UNISOFT, Uruguay 263 piso 5 (1015), Buenos Aires, Argentina ó a su Representante local. Esta tarjeta le permite gozar de varios beneficios importantes. Una vez registrado, puede usar los servicios de Apoyo Técnico, y recibirá automáticamente anuncios y notificaciones de mejoras al producto.

Apoyo técnico

En la Argentina

- ◆ Los usuarios registrados en la **Argentina** y países de habla hispana disponen de apoyo técnico gratuito vía Internet en la dirección: <http://www.clarion.com.ar> (o también, www.unisoft.com.ar) mediante Servicios de Newsgroup y FTP. Un equipo de experimentados técnicos locales (CST: Clarion Support Team Argentina) responderá a sus dudas. También pueden obtener asistencia por correo electrónico a soporte@unisoft.com.ar, o por fax al teléfono (+54-11) 4374-9469.
- ◆ Existen también planes alternativos de soporte técnico por abono anual u horas de consultoría, como el “Corporate Support” que incluye asistencia telefónica ilimitada, seguimiento de problemas, asesoría y otros servicios. **Para más información consulte al (+54-11) 4372-7243.**

Apoyo internacional

Puede recibir apoyo técnico ilimitado para Clarion en CompuServe Information Service y en Internet. Los empleados de TopSpeed, así como los Asociados de Apoyo Técnico Certificados TopSpeed (un grupo voluntario de usuarios Clarion conocidos como *Team TopSpeed*), responderán a sus preguntas oportunamente. Además, podrá recibir consejos y respuestas de otros usuarios Clarion. Recomendamos ampliamente que nuestros clientes hagan uso de estos servicios.

1. En CompuServe escriba GO TOPSPEED para ingresar al foro de apoyo técnico de TopSpeed.
2. En Internet, ingrese al grupo de noticias *comp.lang.clarion*, mediante cualquier servidor de Usenet server.
3. En Internet, ingrese al grupo de noticias *topspeed.products* en *tsnews.clarion.com* server.
4. También en Internet, visite la página Web de TopSpeed en <http://www.topspeed.com>. para más recursos de Clarion.

También puede obtenerse servicio telefónico pago de la Coporación TopSpeed al número (+1) (954) 785-4556.

Sistema de respuestas por fax

TopSpeed ofrece también acceso telefónico mediante fax para acceder a los documentos técnicos y comerciales más comunes. Los documentos en línea incluyen folletos de productos, documentos técnicos, reimpressiones de artículos, listas de precios, e incluso una actualización de los últimos productos TopSpeed (“What’sHot”).

Para pedir documentos específicos, disque (+1- 954) 4785-4555, oprima 53 y escuche las instrucciones del sistema. El menú es interactivo y accesible. Quienes llaman por primera vez pueden pedir una lista de documentos disponibles para hacer su elección. Luego puede ingresar el número de código del documento, y el material le será transmitido inmediatamente. Puede acceder al sistema desde su máquina de fax o desde cualquier teléfono multifrecuente.

El programa de instalación: Setup

El programa Setup descomprime y copia los archivos Clarion a su disco rígido.

- ◆ En todos los sistemas operativos, le ofrece opciones para instalar los varios componentes, tales como los archivos de ejemplo.
- ◆ En todas las versiones de Windows, *pregunta* antes de actualizar la vía de acceso (PATH) en el archivo AUTOEXEC.BAT, para incluir el directorio Clarion.
- ◆ En Windows 3.x pregunta si debe instalar los íconos del Program Manager para el entorno de desarrollo Clarion, y los archivos del Debugger (depurador), de ayuda y ReadMe.
- ◆ En Windows 95 y Windows NT, sugiere instalar todos los íconos del entorno de desarrollo entre los programas del menú de inicio (“Start”).

Requerimientos del sistema

Puede ejecutarse el entorno de desarrollo Clarion en cualquier sistema que cumpla con los requerimientos mínimos para Microsoft Windows 3.x, Windows 95™, o Windows NT. Estos son nuestros mínimos recomendados:

- ◆ Windows 3.x, 8 Megabytes de RAM.
- ◆ Windows 95, 16 Megabytes de RAM.
- ◆ Windows NT, 32 Megabytes de RAM.
- ◆ Mínimo de 12 a 35 Megabytes de espacio de disco libre, según las opciones de configuración (“Setup”) elegidas.

Las aplicaciones que usted desarrolle con Clarion se ejecutarán confortablemente en computadoras que cubran solamente los requerimientos mínimos para esos sistemas operativos.

Iniciando el Setup

Para empezar el programa de Setup de Clarion en Windows 3.x:

1. Inserte el CD de instalación de Clarion en el dispositivo de CD-ROM.
2. Desde el Administrador de Programas, o el Administrador de Archivos u otro programa capaz de ejecutar un programa, elija: **File > Run**.

Escriba D:\SETUP en el diálogo (Donde *D*: es la letra asignada a su unidad de CD-ROM), y oprima el botón de **OK**.

El programa Setup presenta una pantalla introductoria y algunos datos adicionales. Le pide indique las opciones de configuración deseadas.

Para iniciar el programa Clarion Setup en Windows 95:

1. Inserte el CD de instalación en el dispositivo de CD-ROM.
2. Desde el menú de Inicio, escoja **Configuración** ã **Panel de Control**.
3. Elija **Añadir/Quitar Programas**, y oprima el botón de **Instalar**.

El Asistente de Windows 95 lo guiará a través del proceso de instalación.

Opciones de instalación

Después de iniciar Setup, se mostrará un grupo de pantallas del Asistente que muestran una variedad de opciones:

1. Escoja las opciones de configuración ofrecidas por las pantallas (como el disco y directorio de destino) oprimiendo el botón **Next** para desplazarse por las pantallas de opciones.

Setup instalará los componentes principales del entorno de desarrollo Clarion a un subdirectorio BIN ubicado a un nivel más bajo del directorio de destino que usted ha indicado.

El programa de instalación instala *todos* sus archivos al directorio de destino, y a los subdirectorios que están por debajo de éste. No instala archivo alguno en otros directorios.

Durante la instalación, aparecerán barras de avance para mostrar la copia de los archivos.

2. Elija **Yes** o **No** cuando Setup pregunta si puede modificar el PATH.

En Windows 3.x y Windows 95, el entorno de desarrollo Clarion necesita que el subdirectorio BIN esté listado en la variable de entorno PATH. Si elije **No**, deberá modificar manualmente el archivo AUTOEXEC.BAT.

El único cambio adicional que hace Clarion a sus archivos de sistema es añadir su propia sección al archivo de inicialización de Windows (WIN.INI), cuando se lo ejecuta por primera vez.

3. Elija **Yes** o **No** cuando Setup pregunta si instalará los íconos en un grupo de programas Clarion o en el menú de Inicio.
4. Elija **Yes** o **No** cuando Setup pregunta si debe mostrar el archivo ReadMe.
Si no desea leerlo inmediatamente, encontrará el ícono correspondiente en el grupo del Administrador de Programas (o en el menú de Inicio) que crea Setup. Recomendamos leerlo tan pronto como Setup haya copiado todos los archivos.
5. Oprima el botón **OK** cuando Setup haya terminado.

Ejecución de Clarion

Para ejecutar Clarion, ubique el ícono correspondiente en el grupo de programas Clarion y dé DOBLE CLIC sobre él, o simplemente un CLIC en el ícono del menú Inicio.

Aparece el entorno de desarrollo Clarion, listo para trabajar.

- ◆ Encontrará una guía rápida para las partes del entorno de desarrollo, como un diagrama de los íconos de la barra de tareas, en el capítulo tres de esta sección.
- ◆ Siga ahora con el capítulo dos de este libro para crear *dos* aplicaciones impresionantes (desde el diccionario de datos, a las ventanas del *browse*, a las ventanas de *forms*) todo en *menos de una hora*.

2 - INICIO RÁPIDO: CURSILLO

En Clarion, puede crear un diccionario de datos y una aplicación funcional, sin necesidad de codificar. Todo lo que necesita es definir los campos del archivo de datos y el Asistente de Inicio Rápido (*Quick Start Wizard*) crea una aplicación Windows completa... en cinco minutos, si usted digita rápidamente. La aplicación incluye un procedimiento de ficha de actualización para mantener el archivo, una visualización (*browse*) sobre todas las claves de ordenamiento, y tantos informes como claves.

Usando el Asistente de Inicio Rápido, solamente necesita definir el nombre de cada campo del archivo de datos, su formato de visualización (*display picture*) e información sobre las claves. Todo lo demás está previsto en los valores por omisión. ¡El *Quick Start Wizard* se encarga de lo demás!

En este capítulo:

- Usará el **Quick Start Wizard** para crear un archivo de Clientes y una aplicación para mantener el archivo, luego compilará y ejecutará el programa.
- Utilizará Quick Load (“Carga rápida”). Este es otro atajo de Clarion que le permite añadir rápidamente un archivo a un diccionario de datos.
- Definirá una relación entre dos archivos en el **Dictionary Editor**, incluyendo las reglas de integridad referencial.
- Usará el Generador de Aplicaciones para añadir más funcionalidad a la aplicación, usando uno de los **Procedure Wizards** (Asistentes de Procedimiento) de Clarion.
- Finalmente utilizará el **Application Wizard** (Asistente para Aplicaciones) para crear una segunda base de datos relacional completa, a partir del mismo diccionario de datos.

Todo esto llevará unos quince minutos, sin necesidad de “codificación” alguna por su parte. Hacia el fin del capítulo, tendrá dos aplicaciones completas para una base de datos que contiene dos archivos.

¡Bienvenido! ¡A comenzar!

Asistente de Inicio Rápido (Quick Start Wizard)

Creación de un diccionario y de una aplicación

Punto de comienzo:

Debe tener abierto solamente el entorno de desarrollo Clarion. El diálogo Pic debe estar abierto.

En primer lugar, debemos crear un directorio C:\CLARION5\TUTORIAL para las aplicaciones que creará este libro. Este cursillo asume que usted ha instalado Clarion en C:\CLARION5. Si usted usa otro directorio, debe modificar las instrucciones correspondientes.

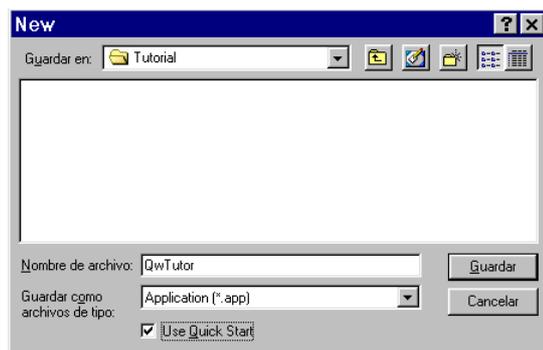
Crear un directorio de trabajo

1. Cambie a la herramienta apropiada del sistema operativo que utilice (Administrador de Archivos, Explorador, etc.) y cree un nuevo directorio llamado TUTORIAL bajo el subdirectorio **Clarion5**.
2. Regrese a Clarion.

Cree su primera aplicación Clarion

1. En la caja de diálogo **Pick** (“Escoger”) oprima el botón **New...** (“Nuevo”).

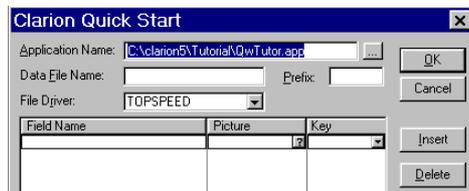
Aparece el diálogo **New**. Es una caja de diálogo estándar *Abrir Archivo* de Windows, que le permite cambiar el directorio y escribir el nombre de archivo.



2. Elija *Aplicacion (*.app)* de la lista desplegable **Save as type** (“Guardar como...”).
3. Elija el directorio C:\CLARION5\TUTUORIAL.
4. Tipee QWKTUTOR en el campo **File name** (“Nombre de archivo”)

El Quick Start Wizard usa esto tanto como nombre de aplicación y diccionario para crear QWKTUTOR.DCT (el archivo del diccionario de datos) y QWKTUTOR.APP (el archivo de la aplicación). Se pueden usar nombres de archivo largos, pero no lo haremos en este cursillo en consideración a los usuarios de Windows 3.1.

5. Tilde la caja **Use Quick Start**, luego oprima el botón **Save** (“Guardar”). Aparece el diálogo **Clarion Quick Start** (“Comienzo rápido”).



Definición del archivo de datos

1. Oprima TAB dos veces, escriba *Cliente* en el campo **Data File Name** y oprima TAB nuevamente.

El Asistente de Inicio Rápido utiliza este nombre como el nombre del archivo de datos. Adviértase que toma las tres primeras letras, *CLI*, y las coloca en el campo **Prefix** (“prefijo”). El prefijo de un archivo identifica inequívocamente los campos que pueden ser iguales (salvo por el prefijo) en múltiples archivos.

2. Oprima TAB para aceptar *CLI* como el prefijo.

A continuación, debe elegir el controlador de archivos (**File Driver**). Este campo indica, por omisión, **TOPSPEED** (uno de los dos sistemas propios de Clarion).

3. Oprima TAB para aceptar **TOPSPEED** como controlador de archivos.

Esto lo lleva a la caja de listado donde se definen los campos.

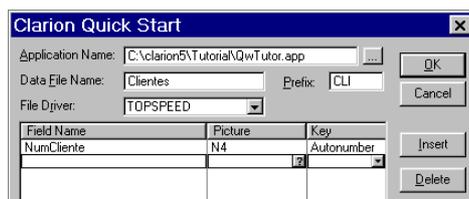
NOTA: Bajo Windows 3.1, debe tener cargado SHARE.EXE o el controlador de Windows VSHARE.386 para utilizar el controlador de archivos TopSpeed. Las versiones más recientes de Windows hacen esto automáticamente.

4. Escriba *NumCliente* en la primera fila de la columna **Field**, y oprima TAB.

Esto crea un campo llamado *NumCliente*. El Quick Start Wizard también utiliza este nombre para los encabezamientos de columna y de indicadores. El indicador (*prompt*) se utiliza siempre que coloque el campo en una ventana. Los encabezamientos de columna se utilizan en los informes y en las cajas de listados.

5. Escriba *N4* en la columna **Picture**, y oprima TAB.

Esto especifica el formato de visualización para los controles de la ventana y de los informes, lo que declara implícitamente el tipo de datos para el Quick Start Wizard.



6. En la columna **Key**, oprima la FLECHA-ABAJO para mostrar las opciones. Destaque *Autonumber* (“Autonumeración”), y oprima TAB.

De este modo se creará una clave exclusiva con este campo como componente. Al especificar *Autonumber*, la aplicación asegura que cada cliente tenga un número distintivo, que automáticamente se incrementará cada vez que se añada un cliente. El Asistente de Inicio Rápido crea *browsers* e informes basados en cada clave que se crea. Esta clave permitirá el desplazamiento entre registros ordenados por *NumCliente*.

El cursor se ubica automáticamente en la fila siguiente, permitiéndole definir el siguiente campo.

7. Escriba *Empresa* en la columna **Field**, y dé TAB.

Esto crea un campo llamado *Empresa* para utilizar como el nombre de la empresa del cliente.

8. Escriba S20 en la columna **Picture**, y pulse TAB.

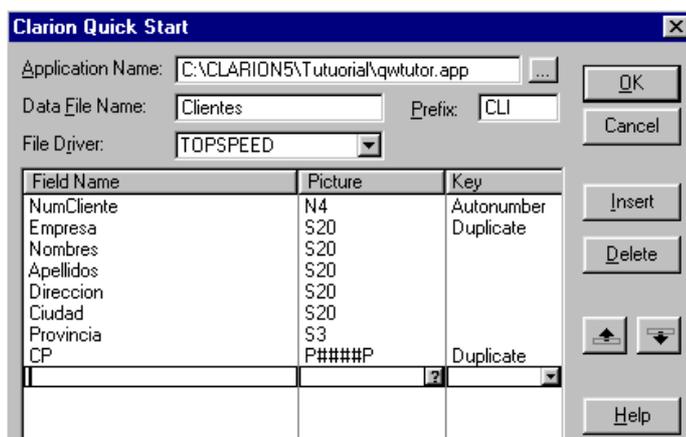
Esto especifica el formato de números mostrado en los controles de ventana y de informe. En este caso, el tipo de datos es un campo STRING de 20 caracteres.

9. En la columna **Key**, oprima la FLECHA-ABAJO para mostrar las opciones. Destaque *Duplicate*, y dé TAB.

10. Termine de definir el archivo creando los campos restantes, según la siguiente tabla:

FIELD	PICTURE	KEY
Nombres	S20	(sin clave)
Apellidos	S20	(sin clave)
Direccion	S20	(sin clave)
Ciudad	S20	(sin clave)
Provincia	S3	(sin clave)
CP	P####P	Duplicate

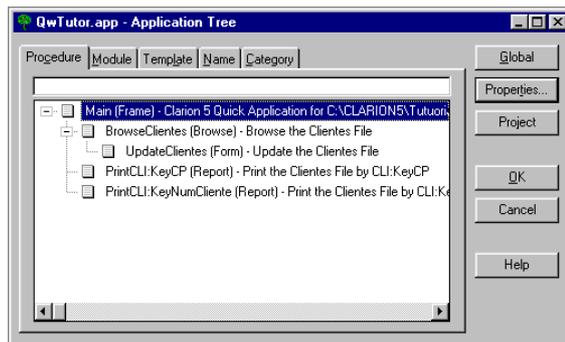
Adviértase el formato (P####P) especificado para el campo CP (Código Postal). Es un “patrón de visualización y archivo” que formatea los dígitos para el código postal.



Cierre del Asistente y creación de la aplicación

1. Cuando haya definido todos los campos, oprima el botón **OK**.

El Asistente le preguntará si ha terminado de definir los campos. Al oprimir el botón **OK**, se crea la aplicación y aparece la caja de diálogo **Application Tree**.



Advierta la estructura de la aplicación. En la parte superior del “árbol” se ubica el procedimiento *Main* (“Principal”), que crea un marco de aplicación MDI (Multiple Document Interface: Interfaz de documentos múltiples). Este procedimiento contiene el menú que llama a los demás procedimientos.

Debajo del procedimiento *Main* se ubica un procedimiento *BrowseClientes* (Browse) con un procedimiento *UpdateCliente* (Form) anexo. El visualizador (*browse*) muestra los datos basado en las tres claves que se especificaron al definir los campos. También puede ver tres informes (*reports*) basados en esas claves.

La aplicación está completa, sin necesidad de que usted suministre más información.

Los íconos de la barra de tareas son, de izquierda a derecha: Elegir, Nuevo, Abrir, Guardar, Hacer, Ejecutar v Depurar.



El botón Ejecutar

2. Elija **Project** ➤ **Run** (u oprima el botón *Ejecutar* en la barra de tareas).

El Generador de aplicaciones crea el código fuente para sus aplicaciones, realiza la compilación, el linkeado y ejecuta el programa resultante.

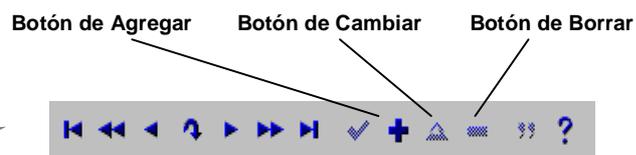
¡Felicitaciones! ¡Su aplicación ya está en funcionamiento!

Explorando la aplicación

En primer lugar añadamos algunos registros.

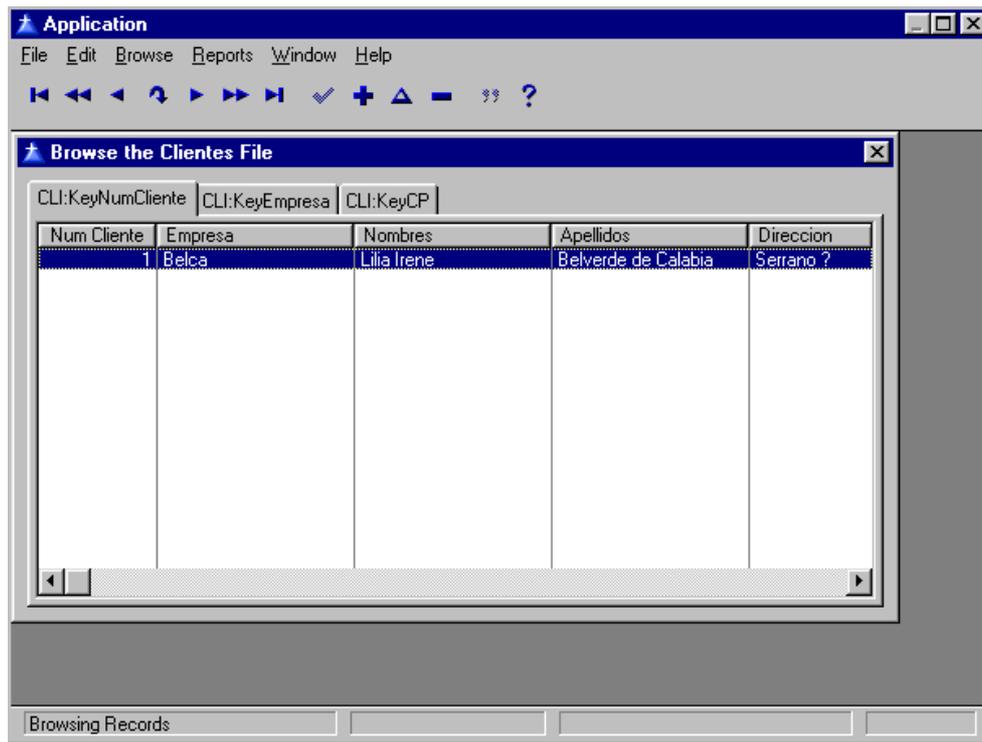
1. Elija **Browse** ➤ **Browse the Clientes file**
2. Oprima el botón **Insert** en la barra de tareas.

Los íconos de la barra de tareas son, de izquierda a derecha: Inicio del archivo, Página arriba, Arriba, Encontrar, Abajo, Página abajo, Fin del archivo, Elegir, Aregar. Cambiar, Borrar. Ídem v Avuda.



1. Una vez que haya tipeado los datos del primer cliente, pulse el botón **OK** para cerrar la ventana de ficha y guarde el registro.

Con ésto demostramos el concepto de las aplicaciones *Browse-Form* de Clarion. La lista de visualización *Browse* muestra los registros de la base de datos, y el procedimiento *Form* se usa para actualizar (agregar, cambiar o borrar) los registros individuales del archivo. Clarion también admite el concepto de *Scrolling Form* (“Ficha deslizante”), en que los registros pueden modificarse directamente sobre la lista de visualización (trataremos ésto más adelante).



2. Agregue algunos registros más (también puede dar CLIC DERECHO [con el botón derecho del *mouse*] sobre la lista deslizante y escoger **Insert** del menú contextual, para añadir registros). Vea cómo cada vez que añada un nuevo registro al archivo de Clientes, el valor del campo NumCliente aumenta automáticamente de a uno. Observe los distintos ordenamientos para la consulta en el procedimiento *browse*, luego varíe el tamaño de las columnas y utilice la barra de desplazamiento horizontal para examinar los datos en los campos que están ocultos a la derecha de la pantalla. Si lo desea, también puede imprimir informes.

3. Elija **File > Exit** para salir de la aplicación.

También puede intentar ejecutar el programa (QWKTUTOR.EXE) desde el Administrador de Archivos o el Explorador de Windows: es un verdadero ejecutable de Windows que no necesita el entorno Clarion para ejecutarse.

Ahora salvaremos el trabajo y saldremos del Generador de Aplicaciones para modificar el diccionario de datos que hemos creado. A continuación añadiremos un segundo archivo relacionado, al diccionario.

4. Elija **File > Close**, o pulse el botón **OK** para cerrar el Generador de Aplicaciones.

Asistente de Carga Rápida (Quick Load Wizard)

Puede añadir rápidamente un archivo al diccionario de datos utilizando la opción Quick Load en el Editor del Diccionario. Este Asistente trabaja de una manera similar al Asistente de Inicio Rápido: se introduce el archivo (nombres de los campos, formatos y claves), y se crea una definición del archivo en el diccionario. Esta opción está disponible siempre que se añade un nuevo archivo al diccionario.

Añadir un archivo

En esta sección añadiremos un archivo para guardar los números telefónicos de los clientes. Esto permite que cada cliente tenga tantos números telefónicos como sea necesario (lo que crea una relación de uno-a-muchos).

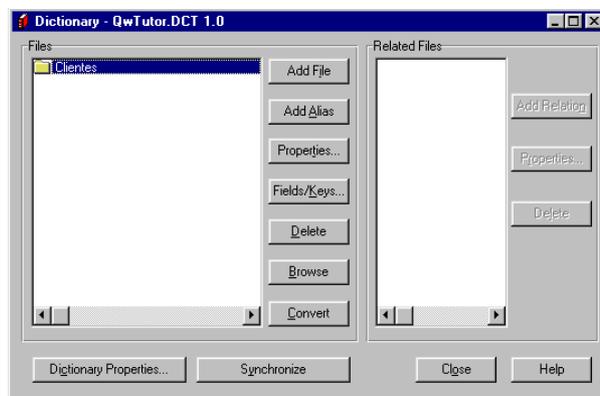
Uso de Quick Load en el Editor del Diccionario

1. Elija **File** ➤ **Open** (u oprima el botón *Open* en la barra de tareas).

Aparece la caja de diálogo **Open**.

2. Elija el tipo *Dictionary (*.dct)* de la lista desplegable **Files of type**, seleccione *QWKTUTOR.DCT*, y oprima el botón **Open**.

Aparece la caja de diálogo **Dictionary Editor**.



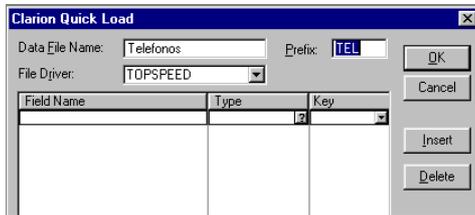
3. Oprima el botón **Add File**.

Aparece otra caja que pregunta: "Do you want to use Quick Load?" ("Desea utilizar la carga rápida?").



Oprima el botón **YES**.

4. Aparece la caja de diálogo **Clarion Quick Load**. Advierta que es muy similar al diálogo del Asistente de Inicio Rápido.



5. Seleccione el campo **Data File Name** (“Nombre del archivo de datos”), y escriba *Teléfono* y pulse TAB.

6. Oprima nuevamente TAB para acepta *TEL* como prefijo.

A continuación, se le pide elegir un controlador de archivos (*File Driver*).

7. Oprima TAB para aceptar **TOPSPEED** como controlador.

Esto lo conducirá a la caja de los listados en que se definen los campos.

8. Escriba *NumCliente* en la primera línea de la columna **Field Name**, y oprima TAB.

Esto crea un campo llamado *NumCliente*. Este campo es el vínculo (*link*) al campo *NumCliente* en el archivo *Cientes*. Usar los mismos nombres de campos facilita vincular los dos archivos. En el código que Clarion genera, el prefijo del archivo se añade a los nombres de campo (con dos puntos [:] como separación), lo que produce nombres exclusivos para los campos que comparten el mismo nombre en archivos separados. Por lo tanto, este campo se llamará en realidad, *TEL:NumCliente*, mientras que el campo en el archivo de *Cientes* se llama *CLI:NumCliente*.

9. Escriba *N4* en la columna **Picture**, y oprima TAB.

10. En la columna **Key**, oprima la FLECHA-ABAJO para mostrar las opciones. Elija *Duplicate* y dé TAB.

Esto especifica una clave que permite asientos duplicados, lo que permite tener más de un registro para cada cliente. Así puede crearse una relación de uno-a-muchos entre los dos archivos (*un* cliente puede tener *muchos* teléfonos).

El cursor se ubica en la próxima fila, lo que permite definir el campo siguiente.

11. En la columna **Field Name** de la fila siguiente, escriba *DDN*, y oprima TAB.

12. En la columna **Picture**, escriba *P(0###)P*, y oprima TAB dos veces (no se define una clave) para continuar con el campo siguiente.

Esta acción formatea el campo *DDN* (Discado Directo Nacional) como un cero más tres dígitos, rodeado por paréntesis, que es la forma estándar para los códigos de telediscado en Argentina. La norma es muy similar en todos los países, pero puede adaptarla localmente, si fuera necesario.

13. Complete el archivo creando los campos de la tabla:

FIELD	PICTURE	KEY
Telefono	P###-####P	(sin clave)
Descripcion	S20	(sin clave)

14. Cuando termine de definir los campos, oprima el botón **OK**.

El Asistente le preguntará si ha terminado de definir los campos.

15. Oprima el botón **OK**.

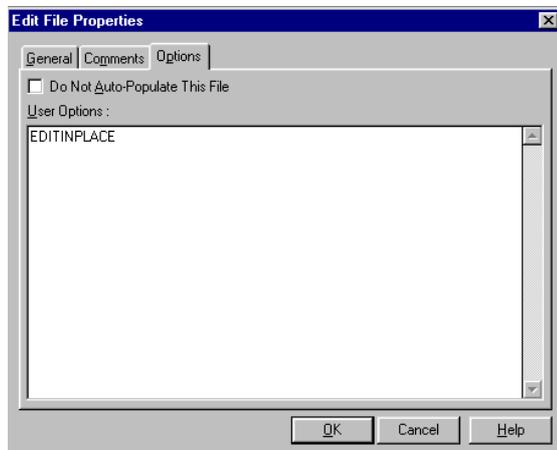
El Asistente crea las definiciones del archivo, y lo añade al diccionario.

Configure el archivo *Telefonos* para usar el concepto de Ficha deslizable (Scrolling Form)

Dado que este archivo tiene sólo cuatro campos, resulta un buen ejemplo del tipo de archivo adecuado para modificar datos sobre la lista deslizable.

1. Destaque el archivo *Teléfono* de la lista **Files**, y oprima el botón **Properties...** (“Propiedades”)
2. Elija la lengüeta **Options**.
3. En el control **User Options** (“Opciones del Usuario”), escriba *EDITINPLACE*.

Así advertirá a los Asistentes (“Wizards”) de Clarion que generen código que le permita modificar los datos del archivo directamente en la caja de visualización deslizable (*browse*).



4. Oprima el botón **OK**.

Añadir una relación

Obviamente, deseamos que el archivo *Telefonos* contenga los números telefónicos de los clientes. Esto significa que los dos archivos deben estar relacionados. En este caso, un cliente puede tener muchos teléfonos, lo que la convierte en una relación “uno-a-muchos”. Para definir esta relación, debemos vincular los archivos en el diccionario de datos, para proveer al Generador de Aplicaciones con la información necesaria para acceder a los registros relacionados.

Establecer una relación entre dos archivos

1. Seleccione *Cliente* en la caja de diálogo **Files** y oprima el botón **Add Relation**.

Aparece el diálogo **New Relationship Properties** (“Nuevas propiedades de la relación”). Aquí se definen las relaciones.



2. Asegúrese de que el campo **Type** indique 1:MANY (uno-a-muchos).
3. En el campo **Primary Key** (“Clave Primaria”), oprima la tecla FLECHA-ABAJO para desplegar las opciones, seleccione *KeyNumCliente*, y oprima TAB.

Esta es la clave del archivo Clientes (el lado “Uno” de la relación), que se usará para vincular los archivos.

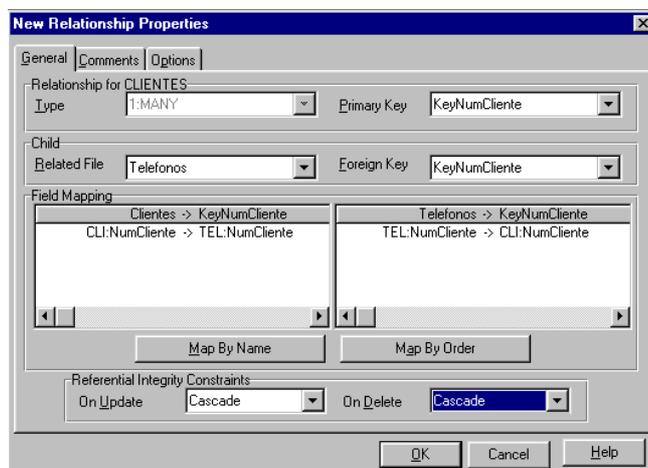
4. En el campo **Related File** (“Archivo relacionado”), oprima FLECHA-ABAJO para mostrar las opciones, detaque *Telefono* y dé TAB.
5. En el campo **Foreign Key** (“Clave foránea”), oprima FLECHA-ABAJO para desplegar las opciones, seleccione *KeyNumCliente* y dé TAB.

Esta es la clave del archivo Teléfono (el lado “Muchos” de la relación) que se usará para vincular los dos archivos.

A continuación, los campos relacionados en las claves deben ser “mapeados”, de manera que el Generador de Aplicaciones sepa exactamente cuáles son los campos que se relacionan entre sí. Dado que usamos nombres de campo idénticos, esto resulta fácil.

6. Oprima el botón **Map By Name** (“Vincular por el nombre”).

Los campos vinculantes entre ambos archivos aparecen en las dos cajas **Field Mapping**.



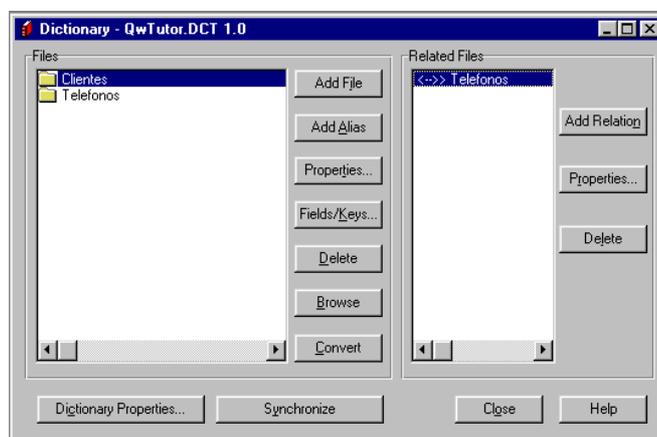
Configurar las restricciones de integridad referencial

1. Elija **Cascade** (“En cascada”) de la lista **On Update** (“Cuando se modifica”) en la caja **Referential Integrity Constrains** (“Restricciones de integridad referencial”).
2. Elija **Cascade** en las lista **On Delete** en la caja **Referential Integrity Constrains**.

El código fuente generado mantendrá en forma automática la integridad referencial entre los dos archivos. El cursillo de la sección *Aprendiendo Clarion* lo explica más ampliamente. Vea también *Using the Dictionary Editor* en la *User's Guide* para un breve tratamiento de la teoría de las bases de datos relacionales.

3. Oprima **OK** para cerrar la caja de diálogo **New Relationship Properties**.

El Diccionario ahora aparecerá así:



4. Elija **File** ➤ **Save**, u oprima el botón **Save** en la barra de tareas para guardar el diccionario de datos.
5. Seleccione **File** ➤ **Close**, u oprima el botón **Close** en la caja de diálogo de **Dictionary** para cerrar el diccionario.

Asistentes para Procedimientos (Procedures Wizards)

Además del Asistente de Inicio Rápido, Clarion tiene también Asistentes que crean procedimientos de visualización de archivos, fichas o informes (Browse, Form o Report), con tanta facilidad como el Asistente de inicio Rápido crea un diccionario de datos y una aplicación. Estos asistentes están disponibles siempre que se crea un nuevo procedimiento en una aplicación ya existente.

Ahora usaremos el *Browse Wizard* para crear un procedimiento de visualización para el archivo Teléfono. También puede crear el procedimiento de actualización (*Form*), pero no esta vez, dado que las modificaciones podrán realizarse directamente sobre la visualizada (“*edit in place*”)

Carga del Generador de aplicaciones

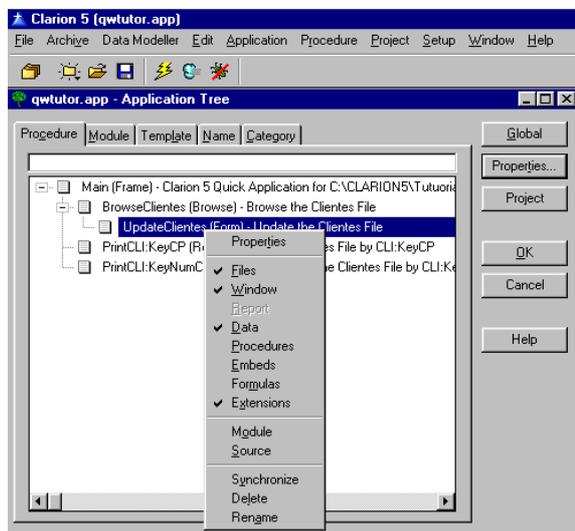
1. Elija **File** ➤ **Pick** (u oprima el botón respectivo en la barra de tareas).
2. Seleccione la lengüeta **Application**, destaque *C:\CLARION5\TUTORIAL\QWKTUTOR.APP*, y luego oprima el botón **Select**.

Aparece la caja de diálogo **Application Tree** (“árbol de la aplicación”).

Modificación del procedimiento de ficha

1. Destaque *UpdateCliente (Form)* en el árbol de la aplicación, y dé CLIC DERECHO para activar el menú contextual.

Este menú le permite el acceso directo a las herramientas del Generador de Aplicaciones, sin necesidad de abrir primeramente las Propiedades del procedimiento.



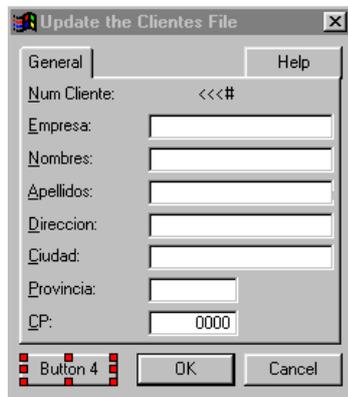
El menú contextual muestra una tilde (✓) junto a las herramientas que contienen datos.

2. Dé un CLIC sobre **Window**.

Aparece el Diseñador de Ventanas (*Window Formatter*). Aquí puede modificar visualmente la ventana y sus controles.

3. Elija **Control** ➤ **Push Button**, o dé CLIC sobre la herramienta Botón en la caja de herramientas de Controles (la que tiene un ícono que se parece a un botón **OK**).

- Dé CLIC cerca del ángulo inferior izquierdo de la ventana para colocar el nuevo control (botón) debajo de la hoja con lengüeta (*tab sheet*).

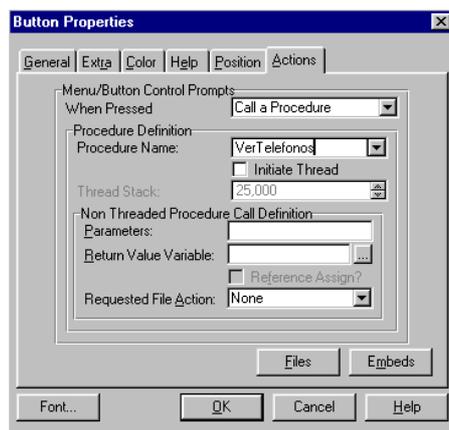


- Con el nuevo botón seleccionado, dé CLIC DERECHO para activar el diálogo llamado **Button Properties**.
- Escriba *Teléfonos* en el campo Text.
Esto cambia el texto que aparecerá sobre el botón.
- Seleccione la lengüeta **Actions** (“Acciones”).

Es en esta lengüeta donde usted especifica lo que hace el control. En este caso, deseamos que llame a un procedimiento *browse* para mostrar todos los registros del archivo Teléfono que se vinculan al registro actual del archivo Clientes.

- Elija **Call a Procedure** de la lista desplegable **When Pressed**.

Aparece la caja de definición de procedimientos (*Procedure Definition*) sobre la ficha de Acciones.



- Escriba *VerTelefonos* en el campo **ProcedureName**, y oprima el botón **OK**.
Así se le da nombre al procedimiento que se llamará cuando el usuario imprima el botón.
- Elija **Exit!** del menú para regresar al Árbol de Aplicaciones y oprima el botón **Yes** cuando se le pregunta si guarda los cambios en la ventana.
Aparecerá el procedimiento *VerTeléfonos* en el árbol como un ítem para elaborar (ToDo).

Uso del Asistente para *browse*

Los Asistentes de procedimientos generan procedimientos completos basados en la información mínima que provee el usuario en respuesta a una serie de ventanas de diálogo del Asistente que hacen una pregunta por vez.

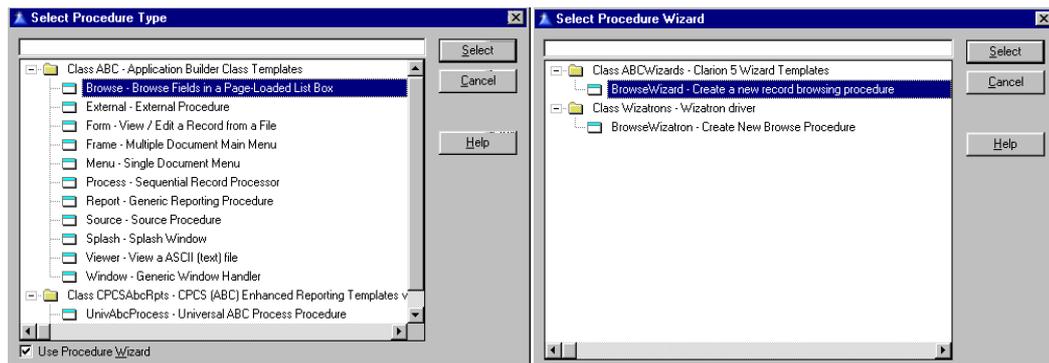
Ejecución del Asistente para browse

1. Dé DOBLE CLIC sobre el procedimiento *VerTeléfonos (ToDo)*.

Aparece la ventana **Select Procedure Type**. Esto le permite elegir la plantilla (*template*) que el Generador de Aplicaciones utilizará para crear el código fuente.

2. Destaque la *template* **Browse**, marque el casillero **Use Procedure Wizard**, luego oprima el botón **Select**.

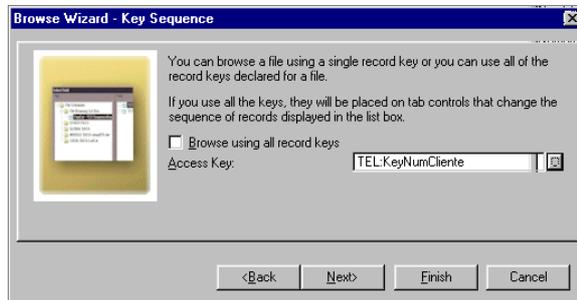
El secreto para poder usar los Asistentes de Procedimiento es marcar con una tilde el casillero **Use Procedure Wizard**. En la pantalla siguiente destaque **BrowseWizards** y oprima **Select**.



3. Después de leer la primera información de la caja de diálogo del Asistente, oprima el botón **Next** para empezar a crear el procedimiento.



4. Escriba *Teléfono* en el casillero de ingreso de datos (u oprima el botón ... y elíjalo de la lista), respondiendo a la pregunta “¿Which file do you want to browse?”(¿ Qué archivo desea visualizar?), y oprima el botón **Next**.



5. Despeje el casillero **Browse using all record keys** (“Visualizar utilizando todas las claves de búsqueda”), oprima el botón ... y elija *TEL:KeyNumCliente* de la lista, y oprima el botón **Next**.

Esto especifica la clave que se utilizará para ordenar los registros para su visualización en el listado.

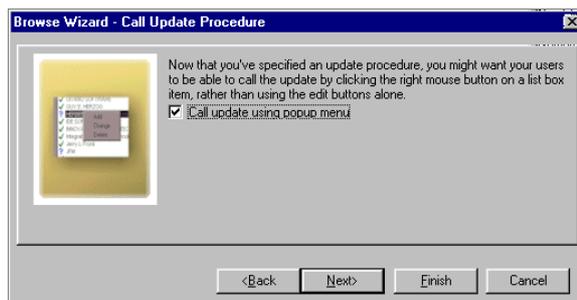
6. Deje en blanco el casillero **Update Procedure** (“Procedimiento de actualización”) y oprima el botón **Next**.

Este casillero de ingreso de datos le permite indicar el procedimiento que el Asistente para *Browse* creará para actualizar los registros del archivo Teléfono. En este caso, esto es innecesario, ya que el Asistente creará el código apropiado para poder modificar el archivo sobre el listado mismo (*in place*), lo que elimina la necesidad de un procedimiento de actualización (Ficha). Sin embargo, si no hubiéramos especificado EDITINPLACE en las opciones del usuario del archivo, el Asistente para *Browse* llamaría al asistente para Fichas para crear el Procedimiento Ficha. Nombrando aquí un “*Update procedure*” evitamos el trabajo de tener que activar luego el Asistente para Fichas.



7. Marque el casillero **Call update using popup menu**, y oprima el botón **Next**.

Esto permite que los usuarios utilicen un menú contextual (que aparece cuando dan CLIC DERECHO sobre el listado) para llamar al procedimiento de actualización (o la capacidad de realizar modificaciones en el listado mismo).



8. Despeje el casillero **Provide buttons for child files** (“Hacer botones para archivos hijos”), y oprima el botón **Next**.



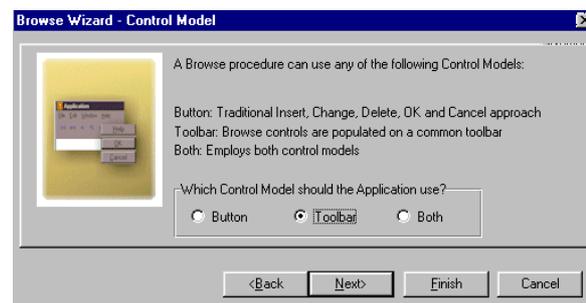
9. Seleccione el botón de radio **Assume that the parent record is active** (“Presumir que el registro «padre» está activo”), luego oprima el botón **Next**.

Esto asegura que el procedimiento se configura para mostrar solamente los registros *Teléfono* que están vinculados al registro del archivo *Cientes* activo en la memoria cuando se llama al procedimiento.



10. Despeje el casillero **Provide a “Select” button** (“Crear un botón «Select»”) y oprima el botón **Next**.

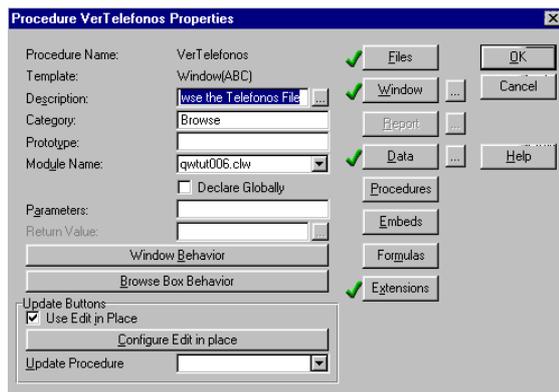
Dado que este procedimiento no se utilizará para consultar datos que el operador deba ingresar en una ficha, el botón *Select* resulta innecesario.



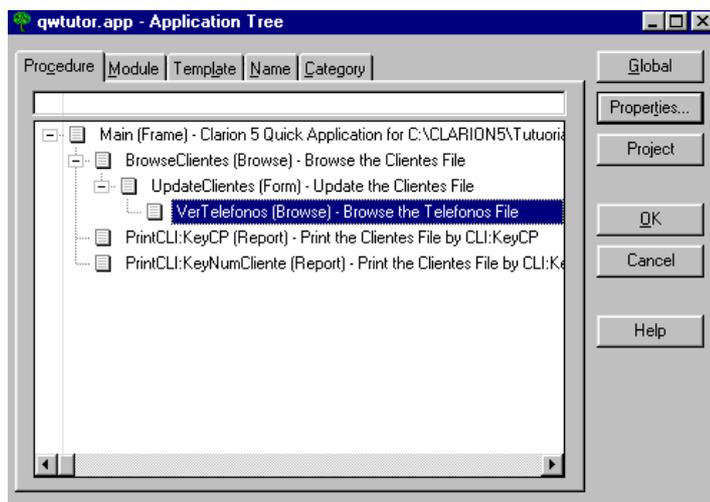
11. Deje configurado el botón de radio **Which Control model should the application Use** (“¿Qué tipo de controles usará la aplicación?”) en *Toolbar* (“Barra de herramientas”), y oprima el botón **Next**.
12. Pulse el botón **Finish** para aceptar el valor por omisión del casillero **Overwrite existing procedures** (“Sobreescribir los procedimientos existentes”).

El Asistente para *Browse* crea ahora el procedimiento *Browse* para el archivo Teléfono. Si no hubiéramos especificado EDITINPLACE en las opciones del usuario del archivo Teléfono, el Asistente para *Browse* también crearía un procedimiento de actualización (*Form*) para mantener los registros del archivo Teléfono. Cuando termina, deja activa la ventana **Procedure Properties** del nuevo procedimiento *Browse*.

13. Oprima el botón **OK** para regresar al Árbol de Aplicaciones.



Observe el nuevo procedimiento que se ha creado para usted en el Arbol.



14. Escoja **Project** ➤ **Run** (u oprima el botón Ejecutar sobre la barra de herramientas).

Tiene ahora una aplicación relacional completa, con múltiples ordenamientos para la visualización de los listados del archivo padre, una exploración del archivo hijo limitado a los registros vinculados a un registro del archivo padre, y código de integridad referencial para asegurar que la base de datos no se corrompa con registros hijos “huérfanos”.

Oprima el nuevo botón Teléfonos y añada registros al archivo Telefono para cada uno de sus Clientes.

15. Cuando haya terminado de explorar su aplicación, escoja **File** ➤ **Exit** para retornar a Clarion.
 16. Elija **File** ➤ **Close** (“Archivo: Cerrar”), o pulse el botón **OK** para cerrar el Generador de Aplicaciones.

¡Y, finalmente, el Asistente para Aplicaciones!

Todos los asistentes que hemos visto: *Quick Start*, *Browse*, *Form*, y *Report* son herramientas poderosas. Sin embargo, Clarion tiene otro Asistente, aún más potente en su arsenal de herramientas para la superproductividad.

El Asistente para Aplicaciones (*Application Wizard*) crea una aplicación completa a partir de un diccionario de datos. Este diccionario puede tener tantos archivos como se quiera; pero deben estar completamente definidos, incluyendo relaciones entre los archivos, reglas de integridad referencial y de integridad de los datos, que pueden definirse en el *Dictionary Editor*.

Para facilitar el uso del Asistente para aplicaciones, el *Dictionary Editor* tiene opciones de propiedades de cada archivo, campo y clave; todas expresamente diseñadas para interactuar con los Asistentes, para permitirle personalizar las aplicaciones que crea para usted.

Usar el Dictionary Editor para configurar las opciones del Asistente

1. Elija **File** ► **Pick** (u oprima el botón *Elegir* de la barra de herramientas).
2. Seleccione la lengüeta **Dictionary**, destaque *C:\CLARION5\TUTORIAL\QWKTUTOR.DCT*, y oprima el botón **Select**.
3. Destaque el archivo *Cientes* y oprima el botón **Fields/Keys...**
4. Elija la lengüeta **Keys**, dé CLIC sobre *KeyNumCliente* para activarla, y oprima el botón **Properties**.

Aparece la caja de diálogo **Edit Key Properties**.

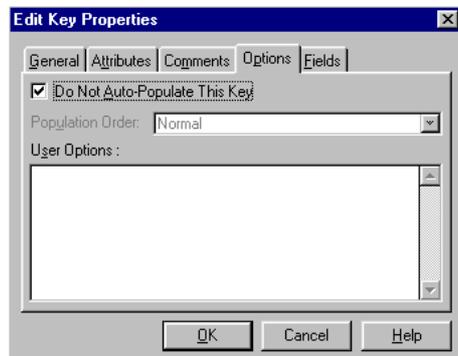
5. Elija la lengüeta **Options**.

Las configuraciones de esta lengüeta se utilizan para especificar el comportamiento de los Asistentes. Las opciones en la mitad superior de la ventana se utilizan con el conjunto estándar de plantillas incluidas con Clarion. La caja **User Options** es para el uso de plantillas adicionales provistas por terceros, lo que permite especificar restricciones u opciones para esos Asistentes que no estén cubiertas por las opciones estándar.

6. Marque el casillero **Do Not Auto-Populate This Key**.

Cuando este casillero está marcado, el Asistente para Aplicaciones no utilizará esta clave al crear un procedimiento *Browse* para el archivo. Esto eliminará la lengüeta *CLI:KeyNumCliente* en el procedimiento *Browse* del archivo *Cientes*.

Hay mucho más que esto que puede hacerse en el *Dictionary Editor* para personalizar el trabajo de los Asistentes. Al finalizar este cursillo, consulte la Sección *Optimizing the Wizards* del *Application Handbook* para más información sobre este importante tema.



7. Oprima el botón **OK**.
8. En la caja de diálogo **Field / Key Definition**, oprima el botón **Close**.
9. Pulse el botón **Close** en el diálogo **Dictionary**. Cuando se le indique, oprima el botón **Yes** para guardar el diccionario.

Eso es todo. A continuación, haremos que el Asistente para aplicaciones genere la aplicación completa.

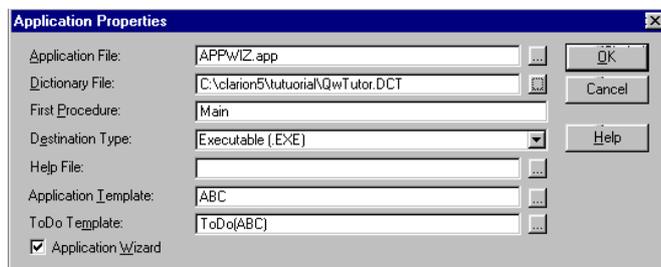
Uso del Asistente para Aplicaciones

El Asistente para Aplicaciones produce aplicaciones completas sobre la información que usted suministra en respuesta a una serie de ventanas del Wizard que van pidiendo un dato por vez.

Crear una nueva aplicación

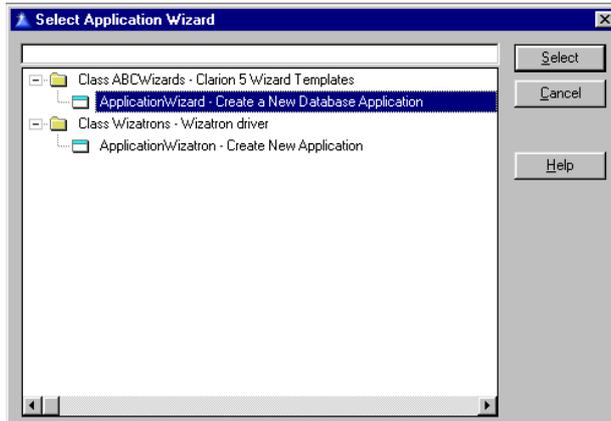
1. Elija **File** ➤ **New** ➤ **Application**
Aparece el diálogo **New**.
2. Elija el directorio `C:\CLARION5\TUTORIAL`
3. Escriba `APPWIZ` en el campo **File Name**.
4. Despeje el casillero **Use Quick Start**, y oprima el botón **Save**.

Aparece la caja de diálogo **Application Properties**.



5. Oprima el botón... a la derecha del casillero de ingreso de datos **Dictionary File**.
Aparece la caja de diálogo **Select Dictionary**.

6. Destaque el archivo `C:\CLARION5\TUTORIAL\QWKTUTOR.DCT` y oprima el botón **Open**.
7. Marque el casillero **Application Wizard** (“Asistente para Aplicaciones”), y oprima el botón **OK**. Seleccione luego **Application Wizard** pulse el botón **Select**.



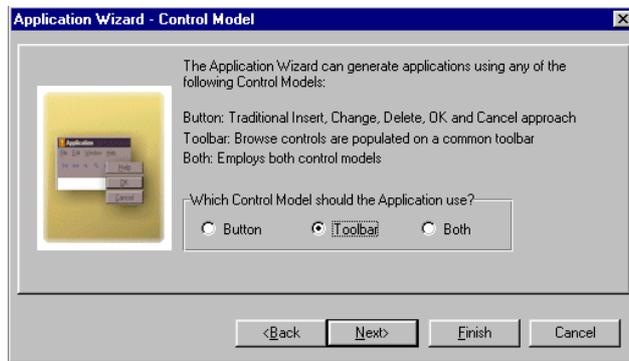
8. Oprima el botón **Next**.



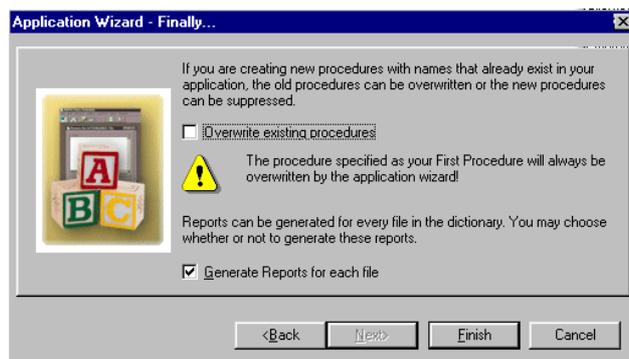
9. Oprima el botón **Next**.



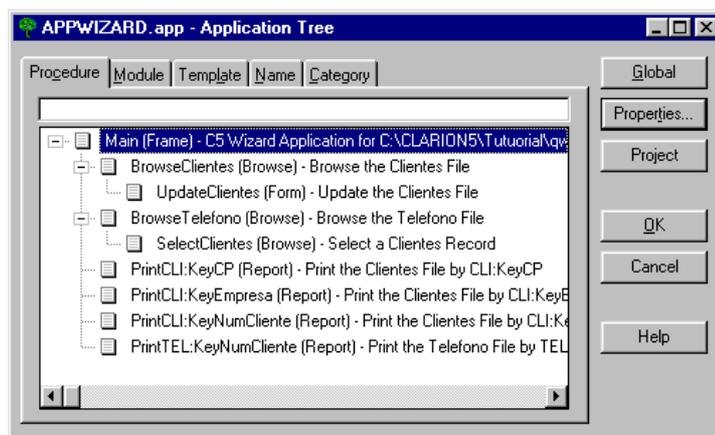
10. Oprima el botón **Next**.



11. Oprima el botón **Finish**.



12. Elija **Project > Run** (u oprima el botón Ejecutar en la barra de herramientas)



¡Felicitaciones!. Su segunda aplicación completa ya está ejecutándose.

Ahora puede explorar esta nueva aplicación, y compararla con la que ya creó. Advertirá que hay algunas diferencias menores entre las dos. Estas diferencias se deben simplemente a que construimos la primera aplicación de una pieza por vez, en lugar de utilizar el Asistente para aplicaciones.

Qué sigue

Si puede hacerse tanto con los “atajos” de Clarion, ¿qué más podrá hacerse con todas las demás herramientas que ofrece?

Además de los Asistentes, Clarion tiene un conjunto de herramientas de productividad en el Generador de Aplicaciones, empezando por los *templates* de Control, Extensión y Código que permiten añadir más funcionalidad a los procedimientos. Estas herramientas facilitan el modificar y personalizar los procedimientos tanto como los Asistentes, ¡y *ninguna* le exige escribir nada de código!

El próximo capítulo de este libro trata la filosofía de desarrollo subyacente a Clarion. Le sugerimos leerlo con cuidado. Se trata de un material de apoyo importante porque, como hemos visto, Clarion es una herramienta de desarrollo distinta a otras. *Comprender la filosofía de Clarion le hará más fácil remontar la curva de aprendizaje.*

El cursillo sobre el Generador de Aplicaciones se encuentra en la sección *Aprendiendo Clarion*. Allí se presentan todas las herramientas de Clarion, y también le demostrará cuánto poder estará a su disposición cuando finalmente escriba algo de código fuente usted mismo, para obtener mayor funcionalidad.

Por favor, siga adelante. Apenas hemos rasguñado la superficie, ¡y *aún hay mucho más!*

3 - PANORAMA DEL PROCESO DE DESARROLLO

El Tao de Clarion

“Tao” significa “el Camino” en chino. Clarion es un *camino completamente nuevo* para crear aplicaciones de Windows. ¡Así lo demuestran las aplicaciones que acaba de crear en unos 30 minutos! Sin embargo, Clarion va mucho más allá que sus Asistentes, de manera que ahora dedicaremos un tiempo a explicar los conceptos centrales detrás de estas herramientas.

Clarion es un lenguaje de programación

A la base de Clarion (como herramienta de desarrollo) se encuentra el lenguaje de programación Clarion. Se trata de un lenguaje de cuarta generación, de propósitos generales, totalmente orientado a objetos, que ha sido altamente optimizado desde su creación para el desarrollo de aplicaciones comerciales de base de datos. El lenguaje Clarion es el poderoso instrumento que tiene ahora en sus manos. Pero Clarion es mucho más que el lenguaje solamente: es también un completo juego de herramientas diseñadas específicamente para la creación y el mantenimiento de aplicaciones de bases de datos.

En el nivel superior del lenguaje Clarion se encuentra el entorno de desarrollo. Es un conjunto de herramientas completamente integrado que han sido especialmente diseñadas para el Desarrollo, Mantenimiento y Mejoramiento Rápido de Aplicaciones. (Estas actividades son conocidas en inglés como: *Rapid Application Development*, RAD; *Rapid Application Maintenance*, RAM; *Rapid Application Enhancement*, RAE). Estas herramientas se orientan hacia la generación automática del código fuente Clarion, de manera que usted no tenga que escribirlo. Es éste el poder de Clarion: aunque usted *puede* escribir su propio código fuente, el conjunto de herramientas puede generar (y regenerar) la mayor parte del código “estándar”, dándole libertad para que usted se concentre sobre los aspectos creativos de la aplicación.

Niveles de abstracción

Hay un principio en computación que indica que, cuanto más uno se aleja de los “fierros” (del código binario), más productivo se puede ser.

Algo de historia

Las primeras computadoras se programaban en binario, lo que hacía muy difícil el trabajo del programador. Luego apareció el lenguaje Assembly (o “ensamblador”) que facilitó la tarea,

haciendo que el código pudiera ser leído en términos “humanos”. Cada sentencia del lenguaje Assembly producía sólo una instrucción para la máquina.

Los primigenios programadores de Assembly advirtieron que escribían en forma repetitiva las mismas secuencias de instrucciones para ejecutar funciones computacionales habituales. Debido a esto, crearon los lenguajes de Tercera Generación (3GL, como Fortran, C, PL/I, etc.) de manera que cada sentencia de uno de estos lenguajes produjera múltiples instrucciones de máquina para esas tareas habituales, lo que hacía aún más productivos a los programadores. Con esto, se elevaron a un nivel de abstracción más alto, más alejado de los “fierros” del computador. La desventaja era que sólo podían generarse secuencias estandarizadas de código de máquina, lo que hizo perder algo de la flexibilidad de trabajar directamente en Assembly.

En la actualidad: lenguajes 4GL

El mismo proceso se ha repetido para la creación de los lenguajes de Cuarta Generación (4GL) como Clarion. Por ejemplo, el lenguaje Clarion maneja automáticamente todas las tareas comunes de Windows (como el redibujo de la pantalla), que los programadores deben escribir personalmente en los lenguajes 3GL. En Clarion, un clásico programa que diga solamente “¡Hola, mundo!” lleva apenas nueve líneas de código fuente, ¡mientras que en C/C++ se deben escribir más de ochenta!

En términos generales, cuanto más alto sea el nivel de abstracción, más productivo puede ser el programador. A medida que sube el nivel de abstracción, el trabajo del programador rinde más, ya que la computadora se encarga de los aspectos rutinarios, generando código normalizado para tareas comunes. Este es principio fundamental detrás del juego de herramientas que componen Clarion.

El secreto de la productividad.

El “problema” con operar a niveles más altos de abstracción es perder la flexibilidad para hacer las cosas exactamente del modo que uno quiere. Con Clarion, este problema está resuelto.

El secreto de la verdadera productividad sin pérdida de flexibilidad es éste: *siempre hay que operar al mayor nivel de abstracción que se pueda* para que el trabajo se haga, a la manera que uno quiere que se haga. El juego de herramientas Clarion contiene múltiples niveles de abstracción, lo que le facilita trabajar al nivel mejor para hacer exactamente lo que se necesita hacer. Uno nunca se encuentra “atrapado” en un nivel alto o forzado a “arreglárselas” a bajo nivel. El juego de herramientas genera código fuente Clarion, orientado a objetos, utilizando la biblioteca de construcción de aplicaciones de alto nivel (*Application Builder Class [ABC] Library*) y, cuando hay necesidad, se puede añadir código propio a bajo nivel, para incrementar la funcionalidad de los templates.

En otras palabras, el secreto de la productividad es *trabajar con más inteligencia, no más arduamente*. Esto es lo que Clarion le permite hacer: escribe el código estándar de manera que usted no deba escribirlo personalmente. Pero tiene la suficiente flexibilidad como para que usted añada todos los “chiches” que desee (lo que resulta mucho más entretenido).

Niveles de abstracción de Clarion

En este punto, usted ha trabajado con los niveles de abstracción más altos de Clarion: los Asistentes. Más adelante, usted interactuará con la mayor parte de los elementos siguientes (dejamos los títulos en inglés, porque es como los encontrará en los menús del programa):

Dictionary Editor (“Editor del Diccionario”)

El Diccionario de datos es el punto central de almacenamiento de todas las definiciones de los archivos de datos y de las configuraciones preestablecidas para el ingreso de datos. Si hay necesidad de tomar estas definiciones de un archivo de datos ya existente, el Diccionario puede importarlas consultando el archivo de datos mismo.

Application Wizard

El Asistente para Aplicaciones genera una aplicación “estándar” completa en sólo minutos, basado en las definiciones de datos y configuraciones del Diccionario.

Procedure Wizards

Los Asistentes para Procedimientos generan procedimientos comunes (como listados para visualización *-browse-*, o fichas de ingreso de datos) basados en las configuraciones establecidas en el Diccionario de datos.

Procedure Templates (“Plantillas para Procedimientos”)

Generan procedimientos comunes (como el menú del marco general de la aplicación, o los informes) basados en el diseño de ventana o de informe que usted crea en el Diseñador de Ventanas y el Diseñador de Informes, y en las opciones que usted elija de las propuestas por el *template* para crear el procedimiento.

Control Templates (“Plantillas de Control”)

Añaden controles “estándar” a los diseños de ventana o de informe (como un control de exploración de árbol de archivos) y generan todo el código necesario para controlar su desempeño.

Extension Templates (“Plantillas de Extensión”)

Generar código adicional en un procedimiento, que añade funcionalidad sin relación a ningún control específico (como el display de fecha y hora en la barra de estado principal de la aplicación).

Code Templates (“Plantillas de Código”)

Generan código en un procedimiento que usualmente realiza una tarea única, relacionada a un control específico (como, por ejemplo, validación del ingreso de datos).

Embedded Source Code (Incorporación de código fuente)

Es posible escribir código propio en cualquiera de los muy numerosos puntos de inserción (Embed Points), para personalizar o modificar el comportamiento del código generado por los Asistentes.

Source Template (Plantilla para código fuente)

Este Asistente le permite escribir enteramente a mano los procedimientos en lenguaje Clarion, conservando la aplicación dentro del Generador de Aplicaciones.

Bibliotecas: Windows API y C

En cualquier punto del código Clarion que usted escriba (procedimientos completos o puntos de inserción), puede llamar directamente a funciones de la API de Windows o de una biblioteca de funciones comunes en C, si fuese realmente necesario (los prototipos de funciones del lenguaje C se incluyen en las ediciones profesional y empresarial de Clarion).

Código C/C++ y Modula-2

Usted puede escribir las funciones C/C++ o Modula-2 que necesite, compilarlas con los compiladores TopSpeed incluidos en las ediciones profesional y empresarial (también disponibles separadamente) y linkearlas a la aplicación Clarion.

Como puede verse, los niveles de abstracción disponibles en Clarion cubren toda la gama, desde el Generador de Aplicaciones hasta escribir código C (¡o incluso Assembly, si quiere trabajar tanto!).

La clave de Clarion para la máxima productividad

La clave es trabajar siempre **al máximo nivel posible de abstracción:**

- ◆ Deje que el Asistente para aplicaciones cree una aplicación “inicial” para usted.
- ◆ Utilice los Asistentes para Procedimientos para añadir procedimientos nuevos, si son necesarios.
- ◆ Modifique lo que sea necesario en cada Asistente para Procedimiento utilizando las herramientas interactivas de Clarion (como los diseñadores de Ventanas y de Informes).
- ◆ Añada Plantillas de Control, Extensión y Código, si son necesarias para aumentar la funcionalidad de la aplicación.
- ◆ Personalícela incorporando algo de código fuente en los puntos de inserción.
- ◆ Cuando haya algo que ningún *template* pueda hacer, utilice la plantilla de Código Fuente (*Source Template*) y escriba en el lenguaje Clarion (u obtenga una plantilla de terceros que haga lo que necesita).
- ◆ “Tírese a lo profundo” recurriendo a los niveles de la API de Windows y la programación en C/C++ cuando sea absolutamente necesario.

He ahí la filosofía Clarion de *trabajar con más inteligencia, no más arduamente*, dejando que el conjunto de herramientas haga el trabajo repetitivo. Así su productividad llegará muy alto, en comparación con otras herramientas que haya usado.

Programación controlada por templates

El Generador de Aplicaciones es “controlado por *templates*”. Esto significa que es una herramienta que cambia según las configuraciones de la plantilla que se use para generar el código. Las plantillas de procedimiento, control, extensión y código todas escriben lenguaje Clarion por usted, dándole un tremendo impulso a su productividad. Los *templates* de Clarion ofrecen muchos de los beneficios de la programación orientada a objetos, especialmente la capacidad de reutilización, y el conjunto de plantillas estándar genera verdadero código orientado a objetos utilizando la

biblioteca de construcción de aplicaciones de alto nivel (*ABC Library*). Esto convierte a las plantillas en la verdadera clave del **Desarrollo Rápido de Aplicaciones** que ofrece Clarion.

¿Qué es un template?

En Clarion, un *template* no es una herramienta de generación de código “de un solo uso”, que después usted deba encargarse de mantener (como las llamadas “plantillas” o “templates” de otros lenguajes), sino que es **una herramienta permanentemente interactiva** que pide la información específica necesaria para generar el código. Si usted cambia la información, el *template* escribe código diferente la próxima vez que genera la aplicación. Es esta interactividad lo que permite el **Mantenimiento Rápido de aplicaciones**.

Los *templates* de Clarion le permiten insertar su propio código fuente Clarion en cualquiera de los numerosos puntos de inserción disponibles dentro de cada plantilla. Esto hace innecesario mantener el código generado para modificar el comportamiento de la aplicación. También garantiza que las personalizaciones especiales no serán sobreescritas la próxima vez que se genera el código fuente: el *template* respeta el código insertado dentro de su propio código. Esta facilidad de personalización y adaptación hace de los *templates* la clave para el **Mejoramiento Rápido de Aplicaciones**.

Todos los *templates* se guardan en el archivo de registro (REGISTRY.TRF). Este archivo contiene código ejecutable preescrito y estructura de datos que pueden modificarse y reutilizarse. Se puede usar el *Registry* para modificar el diseño de la ventana estándar de cada *template* para que aparezcan del modo que usted prefiere al crear un procedimiento.

Las plantillas de Clarion son modificables. También pueden añadirse *templates* de terceros, o usted puede escribir los propios, y usarlos *además de, y conjuntamente con* los *templates* originales. Esto hace del Generador de Aplicaciones una herramienta infinitamente extensible.

¿Cómo se usan los templates?

Al crear un procedimiento nuevo, usted debe identificar la plantilla de procedimiento que genera el código apropiado a la tarea que debe desarrollar, y luego puede personalizarlo con las herramientas de desarrollo. Estas plantillas incluyen elementos como “ventanas de visualización de listados” (*browsers*), para ver grupo de registros, y “ventanas de fichas o formularios” (*forms*) para modificar un registro por vez. Si el procedimiento es para una ventana que incluye un menú, las acciones del menú se añaden automáticamente al árbol de procedimientos, y se las marca inicialmente como “ToDo” (por realizar).

El modo habitual de personalizar o modificar un procedimiento es ejecutar una de las herramientas de diseño. Los Diseñadores de Ventana y de Informe son herramientas de diseño visuales que permiten trabajar mediante el método de “apuntar y dar CLIC”. Usted sólo debe tomar un control de una caja de herramientas, dar CLIC para ubicar el control, y luego un CLIC DERECHO para modificarle las propiedades.

Una vez que se ha creado un procedimiento, también se pueden usar las plantillas de Control, Extensión y Código para añadirle aún más funcionalidad. Las plantillas de Control contienen controles y generan todo el código ejecutable necesario para usarlas y mantenerlas. Todos los controles preconfigurados que aparecen en los diseños de ventanas estándar en los *templates* de procedimientos son en realidad plantillas de control que realizan todas las funciones necesarias.

Las plantillas de extensión añaden código ejecutable que incrementa la funcionalidad del procedimiento. Cada uno ofrece instrucciones en pantalla sobre qué datos debe proveer el programador para incorporar la funcionalidad del *template* a la aplicación.

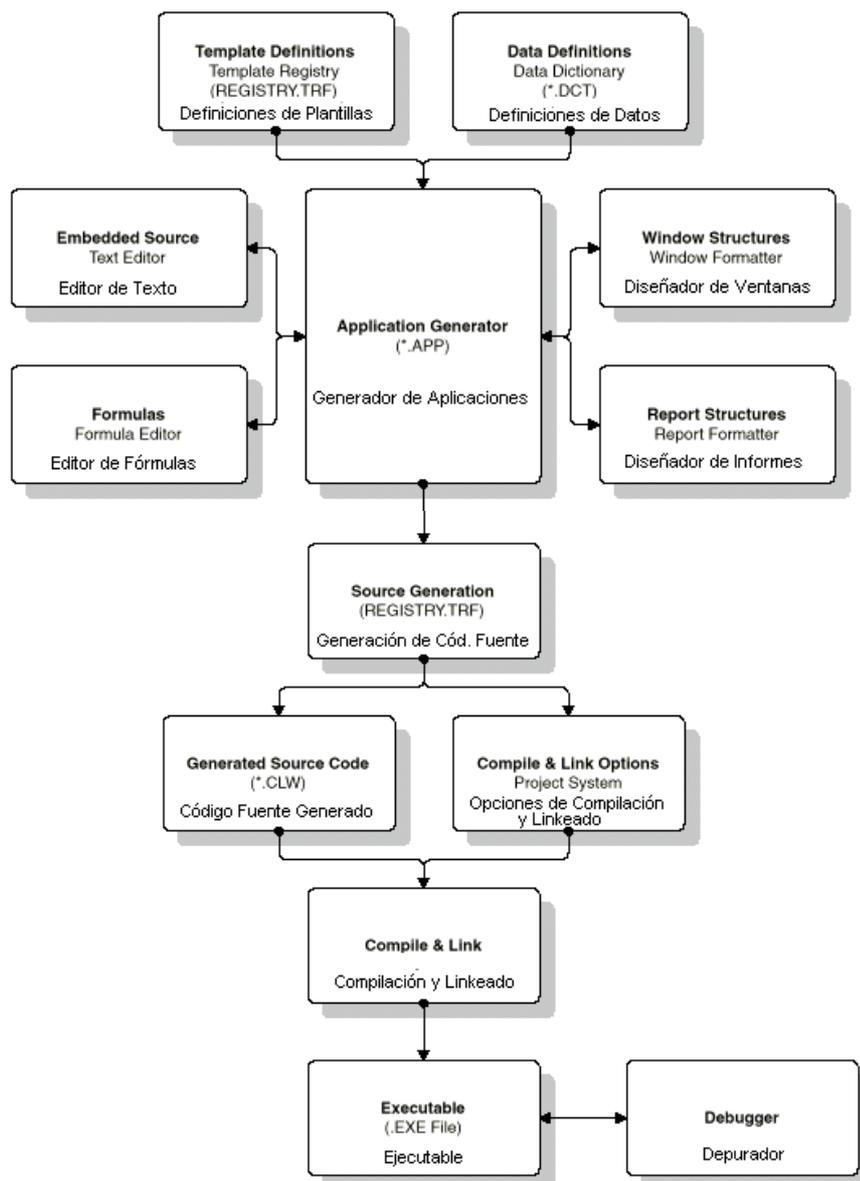
Otra forma de adaptar un procedimiento es añadir código incorporado propio. El Generador de Aplicaciones muestra un árbol que indica todos los puntos de inserción de que se dispone: antes, durante o después de la lógica principal del procedimiento, y para cada “evento” que puedan generar la ventana o los controles incorporados. También se puede escribir código incorporándolo dentro del código fuente generado. Así uno puede elegir el punto lógico exacto en que debe ejecutarse el código, para luego escribirlo “a mano”, o activar una plantilla de Código para que lo haga en su lugar. El Generador de Aplicaciones produce todo el código fuente a partir de los *templates* y de las modificaciones personales que usted haya realizado (incluyendo el código fuente incorporado).

Clarion tiene una amplia variedad de *templates* estándar para ayudarlo a desarrollar aplicaciones rápidamente. Así como el capítulo *Cursillo para un rápido comienzo* lo introdujo a los Asistentes, el cursillo sobre el Generador de Aplicaciones en la Segunda Sección de este manual lo introduce al uso de las plantillas de Clarion para producir cualquier aplicación de Windows que necesite.

El entorno de desarrollo Clarion

El entorno de desarrollo contiene varias herramientas principales, todas las cuales son accesibles entre sí. Al utilizar el Generador de Aplicaciones, hay botones y menús que llevan a otras herramientas.

El siguiente esquema de flujo muestra cómo todas las herramientas interactúan entre sí y con el registro de los *templates*, ubicando al Generador de Aplicaciones al centro de todo el proceso:



Esta sección ofrece una descripción de cada herramienta, en el orden que un programador típico utilizando el Generador de Aplicaciones podría encontrarlas. Cada herramienta contiene cajas de diálogo que el programador rellena para indicar las funcionalidades de la aplicación al Generador de Aplicaciones. Cuando usted lo desea, el Generador de Aplicaciones genera el código fuente especificado, y el Project System lo compila y linkea para convertirlo en un programa ejecutable.

Programar en Clarion es, de variadas maneras, un viaje personal a través de una serie de cajas de diálogo. No hay una secuencia obligatoria que deba seguirse; aunque haber llenado algunos es un prerequisite para alcanzar otros.

El Editor del Diccionario

El Diccionario de Datos (un archivo .DCT), se mantiene mediante el Editor del Diccionario. Aquí se guarda una descripción de la base de datos, que incluye los archivos, claves, índices, controladores, relaciones entre archivos, campos, reglas para la validación de campos, restricciones para la integridad referencial, y más. *Este es el primer archivo que uno crea cuando diseña una aplicación.*

Se pueden crear las definiciones de archivo “desde cero”, o pueden importarse las definiciones de archivos de datos existentes. Si usted posee la edición empresarial, puede utilizar el Sincronizador de Diccionario para coordinar un diccionario de datos Clarion con un diccionario del motor SQL. Esto permite tomar de una vez todas las definiciones de archivo de una base de datos SQL (lo que elimina muchísimo trabajo) y mantener el diccionario de datos Clarion coordinado permanentemente con el ambiente SQL RDBMS, sin importar los cambios que haga el Administrador de la Base de Datos SQL.

Las otras herramientas del entorno de desarrollo utilizan la información en el Diccionario para permitirle, por ejemplo, colocar fácilmente campos de datos en una caja de diálogo diseñada para el usuario final. El Generador de Aplicaciones crea código para todas las sentencias que acceden a los archivos de datos, basándose en cómo está construido el Diccionario. De hecho, ¡el Asistente de Aplicaciones puede generar una aplicación totalmente funcional basada exclusivamente en su Diccionario de Datos!

El Generador de Aplicaciones

El Generador de Aplicaciones genera el código fuente de la aplicación, un procedimiento a la vez, basado en las plantillas que se han elegido del *template registry*. Permite añadir variables de memoria globales y locales, y adaptar los procedimientos con herramientas de diseño visual y código fuente insertado.

El Generador de Aplicaciones da acceso a las demás herramientas del ambiente de desarrollo para poder personalizar el aspecto y la funcionalidad de las ventanas, menús, informes y otros elementos de la interfaz con el usuario. Los *templates* proveen sus propios controles (que aparecen dentro del Generador de Aplicaciones), que le permiten al programador dar la información necesaria para que el *template* personalice la funcionalidad deseada.

El Diseñador de Ventanas

Aquí se realiza el diseño visual de todas las ventanas y controles de la aplicación (todo lo que ve el usuario final). Genera automáticamente el código fuente para los elementos que usted diseña visualmente sobre la pantalla.

El Diseñador de Informes

El Diseñador de Informes trabaja con el Generador de Aplicaciones de una manera muy similar al Diseñador de Ventanas. El programador se limita a poner controles en una página de informe. En tiempo de ejecución, el motor de informes procesa los registros, encargándose de todo el manejo de las páginas impresas, división de datos en grupos, encabezamientos y pies de página, según se especifique.

El Editor de Textos

El Editor de Textos es un editor de programación plenamente funcional que se puede usar para escribir el código fuente que su aplicación necesite. Lo más probable es que, al usar el Generador de Aplicaciones, utilice el Editor de Textos para crear código fuente insertado, personalizando la operación del procedimiento. O, también, se pueden crear aplicaciones completas “desde cero” (si realmente quiere tomarse todo es trabajo).

El Editor de Textos resalta mediante colores las distintas partes de la sintaxis, lo que facilita identificar las distintas partes del lenguaje Clarion, al leerlo y modificarlo. También tiene capacidad plena de búsqueda y reemplazo, además de las herramientas habituales para la edición.

El Editor de Fórmulas

El Editor de Fórmulas le ayuda a generar y manejar fórmulas simples o complejas, basadas en todo tipo de expresiones matemáticas o alfanuméricas. El Editor de Fórmulas ofrece comprobación de la sintaxis, y acceso inmediato a todas las variables, funciones y operadores, de manera que las fórmulas generadas son siempre sintácticamente correctas.

El Sistema de Proyecto

El Generador de Aplicaciones crea automáticamente el archivo de proyecto (.PRJ) de las aplicaciones. Este archivo contiene opciones para la compilación y el linkeado, como si incluir código para depuración, opciones de optimización, archivos externos, archivos de código fuente, bibliotecas y otros archivos que deban incluirse en los procesos de compilación y linkeado.

Los Depuradores

La depuración de un programa generalmente requiere ejecutar un programa e irse deteniendo repetidamente en varios puntos de su ejecución para examinar el valor de distintas variables, para determinar la causa de los errores lógicos que pueda tener. Los Depuradores (*Debuggers*) Clarion (tanto de 16 como de 32 bits) tienen varias ventanas que muestran el código fuente, los contenidos de las variables, procedimientos activos, y mucho más.

Hay tres cursillos sobre el uso de los Depuradores contenidos en el *User's Guide*. Le recomendamos estudiarlos, después de terminar todos los cursillos de este libro.

Ayuda en línea

Clarion ofrece una amplia ayuda en línea sensible al contexto, desde casi cualquier pantalla. Dé un CLIC sobre el botón **Help**, u oprima F1.

La sección *Common Questions* (“Preguntas habituales”) de la ayuda en línea ofrece instrucciones y ejemplos para realizar tareas comunes en las aplicaciones. Usted puede copiar y pegar ejemplos de código directamente desde el sistema de ayuda a su programa. Elija **Help** ➤ **Contents**, y luego oprima el botón **How do I...?** (“¿Cómo hago para...?”)

El Sistema de ayuda en línea de Clarion incluye un archivo principal de ayuda, además de archivos auxiliares. Puede buscarse en los archivos en línea mediante palabras clave y mediante un índice. Para realizar búsquedas en los archivos auxiliares, ábralos directamente con el Explorador de Windows o el Administrador de Programas.

¿Qué sigue ahora?

¿A dónde ir para conocer más sobre la programación con Clarion?

- ◆ Continúe con la sección *Aprendiendo Clarion*. Los dos cursillos que siguen están diseñados para introducirlo a todas las herramientas de desarrollo que Clarion ofrece.
- ◆ Unisoft ofrece diversos cursillos.
- ◆ **Llame al 372-7243/9850 o 374-2998/9469** para informarse.
- ◆ Los usuarios registrados en la lengua hispana, pueden consultar a la siguiente página de Internet: <http://www.unisoft.com.ar>.
- ◆ Unase (o forme) un Grupo de Usuarios Clarion local, y participe en proyectos de estudio conjunto con otros desarrolladores Clarion.
- ◆ Participe en el foro de TopSpeed en CompuServe (GO TOPSPEED) o en el grupo de noticias de Internet *comp.lang.clarion*, para formar una red con los programadores Clarion de todo el mundo (**¡Muy recomendado!**)

Buena suerte, y ¡adelante!. La potencia de programación que Clarion pone en sus dedos sigue creciendo a medida que usted más la conoce.

CLARION 5™

**Aprendiendo
Clarion**

COPYRIGHT 1994, 1995, 1996; 1997 by TopSpeed Corporation

COPYRIGHT 1998, © Unisoft S. R. L.

Todos los derechos reservados

Esta obra está protegida por el derecho de autor y todos los derechos están reservados. No puede ser copiada, fotocopiada, reproducida, traducida o pasada a medios electrónicos o informáticos sin consentimiento, por escrito, de los titulares de los derechos.

Esta publicación acompaña a Clarion 5. Puede contener errores técnicos o tipográficos. TopSpeed y Unisoft la entregan “como está”, sin garantías, explícitas o implícitas.

TopSpeed Corporation
150 East Sample Road
Pompano Beach, Florida 33064
(+1-954)785-4555

Unisoft S.R.L.
Uruguay 263, piso 5
1015 - Buenos Aires
Tel./Fax: (54-11) 4372-7243/9850; 4374-2298/9469

Traducción: Héctor Daniel Calabia, Belca
Armado: Lilia G. B. de Calabia, Belca. Tel. 4774-6845. Fax: 4777-6683

Revisión y actualización: RC Sistemas. Salta – Argentina. Año 1.999.

Reconocimiento de marcas:

TopSpeed[®] es marca registrada de TopSpeed Corporation.

Clarion5[™] es marca comercial de TopSpeed Corporation.

Btrieve[®] es marca registrada de Btrieve Technologies.

Microsoft[®], Windows[®], y Visual Basic[®] son marcas registradas de Microsoft Corporation.

Todos los demás productos y nombres de empresas son marcas comerciales o registradas de sus respectivos propietarios.

Impreso en la Argentina - Printed in Argentina.

ÍNDICE

INTRODUCCIÓN	59
¡Bienvenido nuevamente!	59
Qué hay en esta Sección	59
Convenciones en la documentación	60
Convenciones tipográficas	60
Convenciones del teclado	60
Anatomía de una base de datos	61
Sistemas y controladores de archivos	62
Tipos de datos	62
Ordenamiento de los datos: claves e índices	64
Ordenamientos ascendentes y descendentes	65
Uso de las claves como limitadores	66
Relaciones entre archivos	66
Base de datos: Resumen	67
1 - PLANEAMIENTO DE LA APLICACIÓN	69
Definición de las tareas de la aplicación	69
Diseño de la base de datos	70
Archivo de clientes	71
Archivo de Teléfonos	71
Archivo de Pedidos	72
El Archivo de Detalle	72
El Archivo de Productos	72
Integridad referencial	72
Esquema completo de la Base de Datos	73
Interfaz de la aplicación	74

2 - CREACIÓN DE UN DICCIONARIO DE DATOS	75
Archivos del Cursillo	75
Creación del Diccionario	76
Copiado de archivos de uno a otro diccionario	78
Copia de la definición del archivo Clientes	78
Copia de la definición del Archivo de Teléfonos	79
Vinculación de los archivos y configuración de las opciones de integridad referencial	80
Configuración de las restricciones de integridad referencial	81
Pre-definición de los formatos de control de ventanas	82
3 - INCORPORACIÓN DE ARCHIVOS Y CAMPOS	85
Definición de archivos de datos nuevos	85
Creación del archivo de Pedidos	85
Dar nombre a los archivos de Detalle y Productos	86
Definición de los campos	88
Definición de los campos del archivo de Pedidos	88
Definición de los campos del archivo de Detalle	91
Definición de los campos del archivo de Productos	93
4 - INCORPORACIÓN DE CLAVES Y RELACIONES	95
Definición de claves del archivo de pedidos	95
Creación de la Clave Primaria	96
Designar una clave foránea	97
Definición de claves del archivo de Detalle	99
Definición de la primera clave foránea	99
Definición de la segunda clave foránea	100
Definición de claves del archivo de Productos	101
Crear la clave primaria	101
Definición de una clave alfabética	102
Definición de relaciones entre archivos	103
Definición de las relaciones de "Pedidos"	103
Definición de las relaciones del archivo Detalle	105

Definición de comprobaciones de validez dependientes de la relación	107
Definición de la comprobación de validez de los registros de pedidos	107
Definición de comprobaciones de validez para los registros de Detalle	108
5 - IMPORTACIÓN DE DATOS PREEXISTENTES	109
Conversión de archivos de datos	109
Importación de una definición de archivo. CSV	109
Conversión de un archivo de datos	111
6 - EMPEZANDO LA PROGRAMACIÓN	115
Uso del Generador de Aplicaciones	115
Creación del archivo .APP	115
Creación del procedimiento principal (Main)	116
Edición del menú	118
Creación del procedimiento SplashScreen	121
Adición de una barra de herramientas de la Aplicación	122
Prueba de una aplicación en desarrollo	127
Estudio del código fuente generado	128
7 - CREACIÓN DE UN BROWSE	131
Creación de una Ventana Browse	131
Creación de la visualización del listado de clientes	131
Llenado y diseño de una caja de listado	132
Elija el archivo y los campos para colocar en el listado	133
Adición de las lengüetas	137
Esconder los botones	138
Prueba del browse de clientes	139
Ordenamientos de visualización	140
Cierre del Browse de Clientes	142
8 - CREACIÓN DE UNA FICHA O FORMULARIO	143
Creación de un procedimiento de actualización	143
Incorporación de un procedimiento "ToDo"	143
Creación del procedimiento de Ficha o Formulario	144
Llenado de los campos	145

Mover y alinear los campos	147
Adición de un template de Control BrowseBox	151
Adición del template de control BrowseUpdateButtons	154

9 - COPIADO DE PROCEDIMIENTOS 157

Los procedimientos de “Productos”	157
Copia de los procedimientos	157
Trabajando en los puntos de inserción	158
Modificación del Browse	162
Creación del procedimiento Ficha	165

10 - USO DE CONTROL DE TEMPLATES 167

Creación del procedimiento	167
Selección del tipo de procedimiento	167
Colocación de la caja de listados (BrowseBox Control Template)	168
Adición de la plantilla de botones de actualización	169
Colocación de la segunda caja de listados	170
Adición del template del botón “Cerrar”	172
Especificación de las estrategias de ajuste	173

11 - TEMAS AVANZADOS 175

Actualización del archivo de Pedidos	175
Creación del formulario de Ingreso de Pedidos	175
Colocación de las plantillas de control del archivo Detalle	179
Hacerlo funcionar	184
Uso del Editor de Fórmulas	184
Configuración de la “modificación in situ”	185

12 - CREACIÓN DE INFORMES 197

Un simple listado de clientes	197
Actualización del menú principal	197
Creación del Informe	198
Diseño del Detalle	200
Informe Pedidos	203
Creación del Informe	203
Llenado de la banda de página base	205

Poblando la banda de detalle	206
Adición de saltos de grupo	207
Llenado de la banda cabecera de grupo	209
Relleno de la banda de pie de grupo	212
Llenado de la banda de pie de grupo “Clientes”	214
Adición de una fórmula	217
Adición de un filtro de registros	217
Informe de alcance limitado	
(Range Limited Report)	219
Creación del informe	219
Informe: Pedido individual	221
Creación del Informe	221
¿Qué sigue?	225
13 - CURSILLO SOBRE EL LENGUAJE CLARION	227
Clarion como lenguaje de programación	227
Programación en base a eventos	227
Hola Windows	228
Hola Windows con controles	233
“Hola Windows” con manejo de eventos	235
Adición de un PROCEDIMIENTO	236
Adición de un procedimiento	238
Hacia el mundo real: adición de un menú	239
Más hacia el mundo real: adición de un listado y un formulario	241
Código POO generado por los templates ABC	256
Empezar una aplicación con Quick Start	256
El Program Module	257
Qué sigue después	262
VOCABULARIO	265

INTRODUCCIÓN

¡Bienvenido nuevamente!

La sección anterior presentó el entorno Clarion y los Asistentes. Ahora estudiaremos cómo usar el resto de las herramientas que ofrece Clarion para crear aplicaciones efectivas. Esta sección contiene dos cursillos, a niveles muy diferentes.

- ◆ Cursillo sobre el **Generador de Aplicaciones** que lo familiariza con todas las herramientas del Entorno de Desarrollo Clarion.
- ◆ Cursillo sobre el **Lenguaje Clarion** que presenta este lenguaje y lo pone en contacto con el tipo de código que genera el entorno de desarrollo.

Qué hay en esta Sección

Cursillo sobre el Generador de Aplicaciones

Capítulos 1 al 12: presentación de todas las principales herramientas de desarrollo Clarion. Comienza en la etapa de planificación de la aplicación, lo que lleva hacia la creación del diccionario de datos con el Editor del Diccionario, y lo conduce para crear una aplicación completa con el Generador de Aplicaciones. Al terminar el cursillo, usted habrá creado una aplicación completa de ingreso de pedidos y emisión de informes.

Utilizará el Generador de Aplicaciones y trabajará con las plantillas de Procedimiento, Control y Código para producir esa aplicación. Trabjará con el Diseñador de Ventanas y de informes. Usará el Editor de Textos para incorporar código fuente Clarion en el código fuente generado por las plantillas.

Cursillo sobre el lenguaje Clarion

Capítulo 13: Presenta el lenguaje a nivel de programación manual. Comienza con el simple "¡Hola mundo!", y después lo conduce hacia la creación de ejemplos sencillos de los procedimientos más comunes de las aplicaciones Clarion; todo mientras se explica la funcionalidad del código que usted escriba, y su vinculación con el código que produce el Generador de Aplicaciones.

Convenciones en la documentación

Convenciones tipográficas:

Bastardillas

Indican qué debe tipearse en el teclado, como por ejemplo: *Escriba esto*. También, ocasionalmente, se utilizan para destacar algunas palabras o frases, o indicar expresiones en idioma extranjero.

VERSALITAS

Indican pulsaciones de teclas, como ENTER o ESCAPE, o dar un CLIC con el mouse.

Negritas

Indican órdenes u opciones de un menú o texto en una ventana de diálogo. Note: este estilo también puede usar una familia tipográfica diferente, para igualarse a las negritas en Helvética que utiliza Windows como fuente del sistema.

L E T T E R G O T H I C

Se utiliza para diagramas, listados de código fuente, para comentar ejemplos, y para ejemplos del uso de sentencias de código.

Convenciones del teclado

F1

Indica oprimir y soltar una tecla; en este caso, por ejemplo F1.

ALT+X

Combinación de teclas, en este caso, oprima la tecla ALT y mientras lo hace, pulse X, y luego suelte ambas teclas.

Anatomía de una base de datos

Describiremos aquí brevemente los elementos básicos del diseño de bases de datos. Solamente nos proponemos ofrecer un repaso del tema para quienes no están plenamente familiarizados con los conceptos comunes del diseño de bases de datos. Los desarrolladores experimentados pueden omitir esta sección y avanzar directamente al próximo capítulo.

Definiciones

Una **base de datos** (*database*) es un conjunto de información (*datos*) organizada en un sistema de archivos, registros y campos. La base de datos se mantiene mediante uno o más programas o aplicaciones informáticas.

La unidad básica de almacenamiento de datos es un **campo** (*field*). Un campo es un sitio donde se guarda información de un mismo tipo. Por ejemplo, un campo puede almacenar un nombre y otro campo puede guardar un número de teléfono.

Un grupo de campos diferentes que se vinculan lógicamente conforman un **registro** (*record*). Un registro contiene toda la información vinculada a un tema. por ejemplo, todos los campos con información sobre un estudiante (nombre, dirección, número de teléfono, número de estudiante, etc.), están reunidos en el registro del alumno. Este se asemeja al expediente o ficha que la escuela tiene de cada estudiante.

Un conjunto de registros vinculados lógicamente conforman un **archivo** (*file*). Siguiendo la analogía, una colección de todas las fichas de alumnos producen el archivo general de estudiantes. Esto se asemeja a los archivadores dónde se guardan los expedientes de los alumnos.

Otra forma de visualizarlo es mediante una planilla:

Number	First Name	Last Name	Major
206-65-7223	Dan	Headman	English
168-91-7542	Gary	Pack	English
105-22-0863	Diane	Quinn	English
141-02-7461	Alvin L.	Bailey	Computer Science
123-44-9999	Deb	Brown	Computer Science
101-71-0630	Jerry	Cade	Computer Science
180-43-9184	Nina	Robb	Sociology
229-87-5138	Steve	Stone	Sociology
188-67-7396	Robin	Bailey	Business
164-10-8264	John	Uber	Business
207-95-3939	Brian	Wilcox	Business
116-62-4660	Robert	Macdonald	Law

En este formato, la tabla completa es un archivo, cada fila es un registro, y las columnas representan campos.

Una **base de datos** es una colección de archivos vinculados, también llamados tablas (*tables*). Esto se asemeja a un grupo de ficheros en que se guardan todos los registros de la escuela. Un fichero guarda los archivos con los datos personales de los alumnos; otro, con datos sobre los cursos; y un tercero, con información sobre el personal.

Una base de datos es un conjunto de tablas que guardan entre sí relaciones definidas. El buen diseño de bases de datos separa la información en archivos relacionados que se vinculan mediante campos de enlace. Pronto veremos más sobre esto.

Resumen:

Uno o más campos se combinan para formar un registro.

Uno o más registros se combinan para formar un archivo.

Un conjunto de archivos vinculados es una base de datos.

Sistemas y controladores de archivos

Hay numerosos formatos de archivos utilizados en las PCs. Se trata de los formatos físicos de almacenamiento en el disco que utilizan los programas que mantienen los archivos de datos. Clarion admite numerosos formatos, gracias a la tecnología de controladores (*drivers*) de TopSpeed. Estos controladores permiten a los programas Clarion leer y escribir esos distintos formatos.

La edición estándar trae controladores para los formatos TopSpeed, Clarion, ASCII, BASIC y DOS. Las ediciones profesional y empresarial incluyen Btrieve, ODBC, Clipper, dBase III y IV, y FoxPro. Si necesita utilizar datos de otro sistema de archivos, puede adquirir los controladores apropiados y añadirlos a las ediciones profesional y empresarial. Póngase en contacto con su distribuidor para conocer si el *driver* que necesita está disponible.

Cada sistema de archivos tiene sus particularidades y limitaciones. Para más información, consulte el apéndice *Database Drivers* en la *User's Guide*.

Tipos de datos

Los campos pueden guardar muchos tipos diferentes de datos, pero cada campo individual puede guardar solamente un tipo. Cuando se define el campo, se especifica el tipo de datos. Por ejemplo, para guardar un número de 0 a 100, se usa un campo definido como un único BYTE ocupa menos espacio que uno definido como número decimal (un byte puede guardar cualquier número entero, sin signo, entre 0 y 225).

Clarion admite los siguientes tipos de datos (documentados plenamente en el capítulo 3 de la Referencia del Lenguaje):

Alfanuméricos (Alphanumeric)

STRING Campo que guarda un número específico de caracteres alfanuméricos y otros caracteres ASCII.

PSTRING Campo que guarda una cadena de caracteres con un byte inicial de longitud que indica el número de bytes en la cadena. Este es el tipo de datos utilizado en el lenguaje Pascal y el tipo de datos "LSTRING" del Administrador de Registros Btrieve.

CSTRING Campo que guarda una cadena de caracteres y que termina en un carácter nulo; ASCII cero (0). Este tipo de datos usado en el lenguaje "C" y el tipo de datos "ZSTRING" del Administrador de Registros Btrieve.

Entero (Integers)

BYTE Campo de un byte que guarda un entero sin signo de 0 a 225 (positivo).

SHORT Campo de dos bytes que guarda un entero con signo, de -32.768 a 32.767.

USHORT Campo de dos bytes que guarda un entero sin signo de 0 a 65.535 (positivo).

LONG Campo de cuatro bytes que guarda un entero con signo desde -2.147.483.648 a 2.147.483.647.

ULONG Campo de cuatro bytes que guarda un entero sin signo desde 0 a 4.294.967.295.

Números de coma flotante (reales) (números con fracciones)

REAL Campo que guarda un número de ocho bytes con signo y coma flotante, con 15 dígitos significativos.

SREAL *Campo que guarda un número de cuatro bytes, con signo, con coma flotante y seis dígitos significativos. El SREAL utiliza el formato de short real (precisión simple) de los procesadores Intel 8087.*

BFLOAT8 Variación del REAL, este tipo almacena un número de ocho bytes con signo, coma flotante y ocho bytes según el formato de doble precisión del BASIC de Microsoft.

BFLOAT4 Variación del tipo de datos SREAL, un campo BFLOAT4 guarda un número de cuatro bytes, con signo y coma flotante, que utiliza el formato de precisión simple del BASIC de Microsoft.

Números decimales empaquetados (packed decimal)

DECIMAL Campo que guarda un valor decimal empaquetado, con signo, que va desde -9.999.999.999.999.999.999.999.999.999 a 9.999.999.999.999.999.999.999.999.999.

PDECIMAL Campo que guarda un número decimal con signo en el formato decimal empaquetado de Btrieve e IBM/EBCDIC. La única diferencia entre un DECIMAL y un PDECIMAL es el lugar donde guarda el signo.

Otros tipos de datos

DATE	Campo de cuatro bytes que guarda una variable de fecha en el formato del Administrador de Registros Btrieve.
TIME	Campo de cuatro bytes que guarda una variable de hora en el formato del Administrador de Registros Btrieve.
GROUP	<i>Campo lógico que contiene múltiples campos de datos. Por ejemplo, un campo GROUP llamado NumTeléfono podría contener dos campos: DDN y Teléfono. Un campo de grupo (Group) puede usarse en forma conjunta o cada uno de sus componentes individuales.</i>
PICTURE	Aunque PICTURE no es un tipo de datos, cuando se lo selecciona, declara un tipo de datos STRING con un formato de visualización o máscara (como @P###P) que define el número de caracteres de la cadena. El formato de visualización utilizado para la declaración del campo se ingresa en el campo "Record Picture". Esto resulta práctico para campos cuyo formatos de almacenamiento y de visualización sean diferentes.

Ordenamiento de los datos: claves e índices

Uno de los aspectos más poderosos de una base de datos informatizada es la capacidad de ordenar datos de muchos modos diferentes. Hacer esto manualmente requiere múltiples copias de las fichas, muchas carpetas y múltiples archiveros. También llevaría mucho tiempo archivar cada copia en los distintos lugares, según su ordenamiento.

Por el contrario, en una base de datos informatizada, el ordenamiento requiere solamente definir una serie de "claves" o "índices". Estos declaran ordenamientos adicionales al orden físico de los registros dentro del archivo de datos. En algunos sistemas de archivos las claves se mantienen en archivos separados, en otras, comparten el mismo archivo que los datos. La tecnología de controladores de archivos de TopSpeed maneja transparentemente esas diferencias.

Las claves e índices son funcionalmente equivalentes. Sólo difieren en la forma en que se mantienen en cada aplicación:

- ◆ Una **clave** es mantenida en forma dinámica. Cada vez que se añade, modifica o borra un registro, se actualiza el ordenamiento. Las claves sirven para ordenamiento de uso frecuente.
- ◆ Un **índice** no se mantiene dinámicamente, ya que se lo construye (*build*) sólo cuando es necesario. Los índices son útiles para crear ordenamientos de uso ocasional, como para los procesos que se corren a fin de mes, o a fin de año.

Siguiendo con nuestro ejemplo de un archivo escolar, supongamos que queremos ordenar los registros de alumnos de dos modos: alfabéticamente por nombre, y por nombre dentro de cada materia. Esto produce dos ordenamientos alternados.

Number	First Name	Last Name	Major
141-02-7461	Alvin L	Bailey	Computer Science
188-67-7396	Robin	Bailey	Business
123-44-9999	Deb	Brown	Computer Science
101-71-0630	Jerry	Cade	Computer Science
206-65-7223	Dan	Headman	English
116-62-4660	Robert	Macdonald	Law
168-91-7542	Gary	Pack	English
105-22-0863	Diane	Quinn	English
180-43-9184	Nina	Robb	Sociology
229-87-5138	Steve	Stone	Sociology
164-10-8264	John	Uber	Business
207-95-3939	Brian	Wilcox	Business

Este ejemplo utiliza una clave de un componente sobre el nombre de los estudiantes.

El siguiente ejemplo tiene dos componentes en la clave: curso y nombre del estudiante. **Una clave puede contener más de un campo, lo que permite ordenamientos dentro de otros ordenamientos.**

Number	First Name	Last Name	Major
206-65-7223	Dan	Headman	English
168-91-7542	Gary	Pack	English
105-22-0863	Diane	Quinn	English
141-02-7461	Alvin L	Bailey	Computer Science
123-44-9999	Deb	Brown	Computer Science
101-71-0630	Jerry	Cade	Computer Science
180-43-9184	Nina	Robb	Sociology
229-87-5138	Steve	Stone	Sociology
188-67-7396	Robin	Bailey	Business
164-10-8264	John	Uber	Business
207-95-3939	Brian	Wilcox	Business
116-62-4660	Robert	Macdonald	Law

Ordenamientos ascendentes y descendentes

Algunos sistemas de archivo permiten ambos tipos de ordenamiento para las claves o sus componente (como, por ejemplo, el sistema de archivos TopSpeed). Otros sistemas solamente sustentan el orden ascendente, lo que significa que los datos sólo pueden ordenarse de menor a mayor (por ejemplo, el sistema de archivos Clarion).

El siguiente ejemplo tiene dos componentes en la clave: Año de graduación (descendiente), y nombre del estudiante).

Grad Year	Last Name	First Name
1999	Babbitt	Jim
1999	Headman	Dan
1998	Bailey	Alvin L
1998	Bailey	Robin
1998	Brown	Deb
1998	Quinn	Diane
1998	Stone	Steve
1998	Wilcox	Brian
1997	Cade	Jerry
1997	Macdonald	Robert
1997	Pack	Gary
1997	Robb	Nina
1997	Uber	John

Uso de las claves como limitadores

Supóngase que desea crear un informe de inscripción a cursos de una base de datos que contiene información de los últimos quince años. Para este informe, sólo nos interesan los últimos tres años. Se puede reducir drásticamente el tiempo de procesamiento si se usa una porción del archivo de datos que contenga solamente los registros de los últimos tres años. Esto se hace en dos etapas:

- ◆ En primer lugar, debe definirse una clave para ordenar los datos por la fecha de cada curso.
- ◆ Después, hay que definir los límites que nos interesan. Un límite de alcance (range limit), también llamado “límite de búsqueda” especifica un subconjunto (*subset*) a procesar, tomada del archivo completo. Solamente se considerarán los registros que caen dentro del límite.

En este ejemplo, sólo se procesará un quinto de los registros (suponiendo que cada año tiene similares características). Al reducir el número de los registros en un 80 por ciento, el tiempo de procesamiento disminuye en la misma proporción.

Relaciones entre archivos

Uno de los fines del diseño de bases de datos relacionales es reducir los datos redundantes. La regla básica es que los datos deben ubicarse solamente en un lugar. Esto apareja dos clases de beneficios: en primer lugar, reduce las exigencias de espacio de almacenamiento. En segundo lugar, facilita el mantenimiento de la base de datos. Para alcanzar este objetivo, las bases se dividen en archivos separados, pero relacionados, mediante un proceso llamado *normalización de los datos*.

El primer paso es mover los datos que se repiten a distintos archivos. Por ejemplo, si un estudiante podía tomar un máximo de seis materias, hay que diseñar el fichero de estudiantes para contener seis campos de materias (mat1, mat2, mat3, etc.). Pero no se usarán todos esos campos en cada registro. Si un estudiante toma las seis materias, su registro estará completo; pero si otro estudiante toma solamente una, habrá seis campos vacíos en su registro. Por esta razón, los campos de materias se mueven a un archivo separado, lo que elimina la necesidad de reservar espacio para campos vacíos. Esto crea una relación *uno-a-muchos*. (*Un* estudiante puede tomar *Muchas* clases) entre el archivo de estudiantes y el archivo de materias.

El siguiente paso es mover los datos redundantes a archivos separados. Cada campo en el archivo de estudiantes debe depender de la clave primaria (Número de estudiante). El nombre, dirección y teléfono permanecen en el archivo de estudiantes. Pero la descripción del curso de cada estudiante podría moverse a un archivo separado. Esto elimina la necesidad de repetir la descripción del curso para cada alumno dentro de ese Curso. Para hacerlo, hay que añadir un campo de número de Curso al archivo de estudiantes y al archivo de cursos. Esto crea una relación de *muchos-a-uno* (*Muchos* estudiantes siguen *Un* curso) entre el archivo de estudiantes y el archivo de cursos.

Una vez que se ha “normalizado” el almacenamiento de información, los datos relacionados se vinculan haciendo que cada archivo tenga un campo idéntico a otro en el archivo relacionado. Un

campo de enlace podría ser el número de estudiante, el número de curso, o un número de aula. Cualquier campo (o grupo de campos) que identifica en forma unívoca un registro en el archivo primario, puede usarse como enlace.

Se pueden encontrar ejemplos de esas relaciones en el archivo de datos.

- ◆ Un profesor enseña **muchas** (varias) materias(uno-a-muchos).

Number	First Name	Last Name	Description	Scheduled Time	Room Number
143-79-3932	Ben	Noble	Microcomputers	M-W 900	381
			Microcomputers	M-W 1100	381
			Logic & Algorithms	M-W 1300	301
			Logic & Algorithms	T-Th 1300	301

- ◆ **Muchos** estudiantes compartirán **un** curso (muchos-a-uno).

Number	First Name	Last Name	Major
168-90-3748	Jim	Babbitt	English
101-71-0630	Jerry	Cade	English
206-65-7223	Dan	Headman	English
168-91-7542	Gary	Pack	English
105-22-0863	Diane	Quinn	English
141-02-7461	Alvin L	Bailey	Computer Science
123-44-9999	Deb	Brown	Computer Science
180-43-9184	Nina	Robb	Sociology
229-87-5138	Steve	Stone	Sociology
188-67-7396	Robin	Bailey	Business
164-10-6264	John	Uber	Business
207-95-3939	Brian	Wilcox	Business
116-62-4660	Robert	Macdonald	Law

Bases de datos: resumen

- ◆ Una base de datos es un conjunto de información (datos organizado en un sistema de campos, registros y archivos).
- ◆ Los campos pueden guardar muchos tipos diferentes de datos, pero cada campo individual sólo almacena de una sola clase.
- ◆ Cada porción de información debe guardarse en un solo lugar.
- ◆ Uno o más campos componen un registro. Uno o más registros componen un archivo. Un grupo de archivos relacionados componen una base de datos.
- ◆ Los programas Clarion pueden acceder a numerosos sistemas de archivo mediante el uso de controladores de archivos.
- ◆ Las claves e índices declaran ordenamientos distintos del orden físico de los registros dentro del archivo, y pueden contener más de un campo, lo que permite ordenamientos secundarios dentro de un ordenamiento.
- ◆ Los límites de búsqueda (*range limits*) permiten procesar un subconjunto de registros, lo que reduce el tiempo de procesamiento.
- ◆ Los archivos se relacionan a través de campos comunes, que contienen datos idénticos, lo que ayuda a eliminar información redundante.

1 - PLANEAMIENTO DE LA APLICACIÓN

Por regla general, cada minuto invertido en planear la aplicación, ahorra diez minutos después. Este capítulo describe informalmente el proceso de planeamiento de la aplicación que crearemos en los capítulos subsiguientes. En el mundo real, probablemente habría que crear una especificación funcional previa, por escrito, de las aplicaciones a realizar. Esta descripción informal define:

- ◆ Las tareas que realiza la aplicación.
- ◆ Los datos que la aplicación mantiene, y cómo los guarda.

Como punto de inicio, la aplicación del cursillo del **Generador de Aplicaciones** usa el diccionario de datos de la aplicación que usted ya creó usando los Asistentes en la sección anterior de este manual. Extiende el concepto a un sistema simple de Pedidos, usando el diccionario de datos para el seguimiento de los clientes.

Definición de las tareas de la aplicación

La aplicación mantendrá los archivos de clientes y de pedidos para una empresa industrial. La primera tarea para planear lo que la aplicación hará es evaluar lo que la empresa espera que haga.

Para los fines de este cursillo, la aplicación que crearemos es un sistema simple de ingreso de pedidos. Es común que los clientes hagan pedidos telefónicos para uno o más productos por vez. A la noche, el departamento administrativo imprime la orden para depósito.

Por lo tanto, la aplicación debe proveer:

- ◆ Cajas de diálogo de ingreso de datos, para tomar el pedido, o modificarlo después.
- ◆ Acceso a la lista de clientes desde dentro de los formularios de pedido. La lista de clientes es la que ya se creó con el Asistente para un Rápido Comienzo, guardado en el archivo de clientes.
- ◆ Acceso a la lista de números de repuestos (artículos) que fabrica la compañía desde los formularios de ingreso de pedidos.
- ◆ Ventanas de visualización de transacciones de ventas.
- ◆ Procedimientos que mantengan la lista de Productos e información sobre Clientes.
- ◆ Informes impresos.

Diseño de la base de datos

La primera tarea al planear la estructura de archivos es evaluar qué datos necesita la aplicación, y cómo guardarlos con un mínimo de duplicaciones.

El buen manejo de las bases de datos requiere mantener archivos de datos separados para cada "entidad" o grupo de datos. Las "entidades de datos" que mantendrá esta aplicación son:

Cliente	Nombre y dirección de los clientes (ésta última sólo cambia si el cliente se muda). Creado en el "Cursillo para un rápido comienzo", conjuntamente con el archivo de Teléfonos.
Pedidos	Información básica necesaria para ensamblar los datos necesarios para imprimir una nota de pedido. Hace "consultas" a la información de otros archivos, como nombre y dirección de los clientes. Cuando una persona hace un pedido nuevo, se añade un registro a este archivo.
Detalle	Producto, precio y cantidad de un artículo dado en un pedido: la información variable de cada pedido. Aunque esto produce un duplicado de la información de precios en el archivo de Productos, debe mantenerse aquí el precio del momento de la venta. De lo contrario, el monto variaría cuando cambien los precios en archivo Productos.
Productos	Información sobre los productos vendidos por la empresa, incluyendo su número, descripción y precio. Estos datos cambian solamente cuando hay un cambio de precios, o se añade un producto. *

***Nota del Traductor:** En la versión original, el archivo de Productos se llama "Products" y, por ejemplo, el de Teléfonos, "Phones". Como se ve, ambos nombres de archivos no superan las ocho letras permitidas tradicionalmente por el DOS. En castellano, tanto "Productos" como "Teléfonos" tienen más de ocho letras. Si usted usa Windows 3.x deberá llamar a estos archivos "Producto" y "Teléfono". Quienes usen Windows 95 y superiores podrán usar cualquier denominación. En adelante, para facilitar la lectura en estos casos y otros similares usaremos el nombre de archivo largo.

Nota 2: El diccionario de datos permite ingresar nombre de archivo y de campos con acentos, como "Teléfonos", y éstos son procesados correctamente por el Generador de Aplicaciones. Al verificar la traducción de este manual descubrimos, sin embargo, que algunas partes de Clarion no funcionaban bien

con acentos; y que causan dificultades difíciles de diagnosticar. Aconsejamos no usar acentos en nombres de archivos, campos o claves. Sí pueden usarse (y es aconsejable hacerlo) en la interfaz de usuario: etiquetas de campo, encabezamientos de columnas, leyendas explicativas, etc. Estos elementos pueden predefinirse sin problemas en el Diccionario de Datos(HDC).

Archivo de clientes

El archivo de Clientes guarda datos "constantes" como nombres y direcciones. Resulta eficiente guardar esta información en un solo lugar, lo que facilita realizar una única actualización cuando hay un cambio. También ahorra espacio, evitando duplicados de información en el archivo de Pedidos. De lo contrario, si hubiera mil pedidos de la compañía XYZ, la dirección se repetiría 1000 veces. La reducción de las necesidades de almacenamiento mediante el recurso de guardar los datos sólo una vez se llama *normalización*.

Los datos de clientes requieren un campo que identifique unívocamente al cliente. El nombre de la empresa no sirve, porque podría haber duplicados. Por ejemplo, podrían existir registros múltiples para un cliente llamado "Ferreterías <<La Manguera>>" si tiene varias sucursales. La aplicación que ya hemos realizado especificaba que el campo *NumCliente* sería una clave autonumerada, que crea y guarda números exclusivos para cada cliente.

El campo *NumCliente* sirve como "clave primaria"(*primary key*) para el archivo de datos. Cualesquiera otros archivos que se vinculen al archivo de Clientes deben declarar el campo *NumCliente* como "clave foránea"(*foreign key*). Una clave primaria es un campo, o una combinación de campos, que singulariza unívocamente cada registro de un archivo. Una clave "foránea" es un campo, o una combinación de éstos, en un archivo (o tabla) cuya clave debe ser igual a una clave primaria en otro archivo vinculado.

Debido a que hay muchos pedidos para cada número de cliente, la relación entre el archivo Clientes y el archivo Pedidos será de *uno-a-muchos* (1:Many). Decimos que el archivo de clientes es el archivo *padre* y que el archivo pedidos el *hijo*.

Archivo de Teléfonos

El archivo Teléfonos guarda los números de cada cliente, y pueden haber muchos números por cada cliente.

El archivo de Teléfonos también incluye un campo de texto en el que indicamos si el número de teléfono pertenece a la oficina, al fax, al celular, o al domicilio particular. Usando el diccionario de datos, especificaremos que el control de ingreso de datos sea una lista desplegable, con las opciones precargadas.

Archivo de Pedidos

El archivo de pedidos toma información de cada transacción desde todos los demás archivos (como los datos del cliente). Debido a que gran parte de la información básica de este archivo se imprime en la "cabecera" de la Nota de Pedido, suele llamárselo "Cabecera de Pedidos".

Cada transacción de ventas requiere un registro en el archivo de Pedidos. El registro se vincula a la información del cliente gracias al número de cliente. Debido a que algunos pedidos pueden tener un solo artículo, y otros más de diez, habrá que crear un archivo separado de "Detalle" que se vincule con el número único del pedido. Esto crea una relación de *uno-a-muchos*, con el archivo de Pedidos como padre y el archivo de "Detalle" como hijo. Los productos pedidos se identifican por sus códigos de producto, en el archivo de "Detalle".

De modo que el registro de Pedidos tiene un número de cliente para vincularlo a los datos del cliente (clave foránea) y un número exclusivo de pedido, para vincularlo al "Detalle". Crearemos una clave primaria multicomponente sobre ambos campos, para crear fácilmente un procedimiento de visualización (*browse*) ordenado por cliente y por número de pedido.

El Archivo de Detalle

Este archivo guarda los artículos ordenados por sus códigos de producto (clave foránea que referencia el archivo Productos), sus precios, cantidades, y tasa imponible. Un campo adicional guarda el número de pedido, que se vincula al archivo de Pedidos en un relación de *muchos-a-uno*.

El archivo de Detalle duplica la información de precios ya guardada en el archivo de productos. Esto se debe a que los precios pueden variar. Es importante guardar el precio dentro del archivo de detalle, porque de lo contrario, si el precio se modificara, el módulo de facturación tomaría precios diferentes a los acordados con el cliente al momento del pedido.

El archivo de Productos

Este archivo guarda los números exclusivos de cada producto, descripciones y precios. Cuando el vendedor busca un producto por su nombre, la aplicación inserta el número de producto en el registro de Detalle. El código de producto es la clave primaria: ningún artículo puede tener un número repetido, y cada producto debe tener un número. Un campo adicional contiene la tasa imponible de ese producto.

Integridad Referencial

La integridad referencial se refiere al proceso de controlar todas las modificaciones al campo clave de un archivo determinado, para asegurarse que se mantiene la validez de la relación padre-hijo.

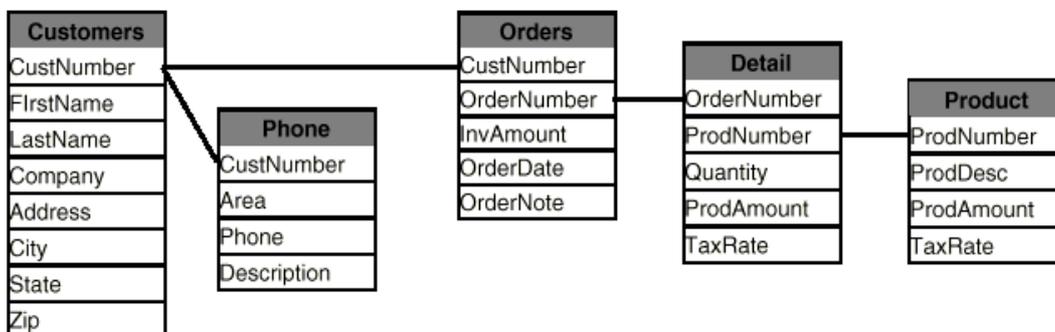
También significa vigilar que todos los registros "hijos" tengan "padres" válidos; de manera que no queden registros "huérfanos" en la base de datos.

Debido a que los datos para una transacción dada residen en varios archivos, esta aplicación debe mantener la integridad referencial. Esto es crítico, sin embargo, muchas otras herramientas de desarrollo de bases de datos exigen que uno escriba procedimientos manuales para asegurarlo. El Generador de Aplicaciones lo implementa automáticamente en el código fuente generado, cuando usted elige unas pocas opciones del Diccionario de Datos.

Es esencial que la aplicación no permita modificar un registro de modo que deje un valor vacío o duplicado en un campo clave primario. Por ejemplo, es necesario limitar la capacidad del usuario final de crear un número de Cliente duplicado. Si dos compañías distintas compartiesen su número de cliente, correríamos el riesgo de registrar un pedido a la empresa equivocada.

Esquema completo de la Base de Datos

El esquema siguiente ofrece un panorama de la base de datos completa. Si se lo mira desde el punto de vista de un vendedor que toma un pedido telefónico, el archivo de pedidos graba de quién es el pedido, el Detalle guarda los artículos pedidos, y los archivos de Clientes y Productos guardan información constante sobre ambos temas.



El código de artículo consulta la descripción y el precio. El código de cliente busca el nombre y la dirección del cliente. Otros datos, como la fecha de la operación, se llenan automáticamente (consultando la fecha del sistema, por ejemplo).

Finalmente, el cursillo creará un diccionario de datos nuevo, al que usted copiará los archivos que se definieron anteriormente, al nuevo diccionario.

En cuanto a la aplicación creada, debido a que el cursillo es un instrumento pedagógico que intenta mostrar lo que Clarion puede ayudarle a hacer, no creará un sistema de pedidos a escala completa. Sin embargo, ciertas partes de la aplicación serán muy reveladoras, para que aprenda cómo aplicar procedimientos similares en sus propias aplicaciones.

Interfaz de la aplicación

La tarea mayor que sigue antes de empezar a codificar es planear la interfaz con el usuario. En una aplicación de negocios como esta, es importante que el vendedor ubique rápidamente los datos que necesita para que pueda registrar la venta y atender la próxima llamada. Por lo tanto, la aplicación debería poner todos los datos importantes a la vista, desde el principio, de manera que ningún ingreso o modificación de datos esté a más de un botón u orden del menú de distancia.

Además, la empresa ya usa otras aplicaciones Windows; por lo que es importante que la aplicación tenga una apariencia y comportamiento común con las demás. Los usuarios finales aprenden más rápidamente una interfaz familiar.

Para implementar las tareas, la aplicación deberá ejecutarse de un modo consistente con los lineamientos que indicamos más abajo. Aunque lo que sigue no sustituye a una real especificación del programa, deberá servirnos a los fines del Cursillo.

- ◆ Dado que la aplicación manejará el mantenimiento de los archivos de los clientes, artículos, y pedidos en distintos formularios, es necesaria la Interfaz para Documentos Múltiples (MDI).
- ◆ La aplicación deberá tener una barra de tareas con botones para cargar formularios y ventanas de visualización (*browse*) para controlar su funcionamiento.
- ◆ Para mantener coherencia en la interfaz, las principales opciones del menú serán Archivo, Editar, Ver, Ventana y Ayuda. El menú de archivo accede a los procedimientos de impresión y salida. Los botones de la barra de herramientas llaman a las cajas de diálogo para modificar un registro existente (si está seleccionado en un *browse*) o para añadir /borrar registros, y para desplazarse por los archivos. El menú Ver llama los procedimientos para examinar los archivos de datos. Ventana y Ayuda realizan las acciones habituales.
- ◆ Cuando se añaden nuevos pedidos, la gente de ventas debería poder elegir clientes y productos de listas deslizantes. Muchos datos del pedido (direcciones, descripciones y precios) deberían irse "llenando" a medida que fuese necesario.

Ahora que hemos descrito en modo bastante completo la aplicación, estamos listos para empezar a trabajar. El primer paso es crear el diccionario de datos.

2 - CREACION DE UN DICCIONARIO DE DATOS

Este capítulo enseña cómo:

- ◆ Crear un nuevo diccionario de datos.
- ◆ Copiar y adaptar las definiciones tomadas del Diccionario creado por Quick Start.
- ◆ Vincular los archivos y especificar las restricciones de integridad referencial.
- ◆ Pre-formatear los controles de ventana para los campos.

Este cursillo presume que usted ha finalizado el Cursillo para un Inicio Rápido de la primera parte de este libro.

Archivos del Cursillo

Le recomendamos realizar el cursillo completo, especialmente si el entorno de desarrollo Clarion le resulta totalmente nuevo(e incluso si ya ha usado versiones anteriores de Clarion). Como habrá visto en el cursillo anterior, el enfoque de Clarion con su Generador de Aplicaciones manejado por *templates* es muy diferente de otros entornos de desarrollo en lenguajes de tercera y cuarta generación. Si hace el curso enteramente, obtendrá los mejores resultados de esta nueva herramienta.

Los archivos completos del cursillo residen en el directorio

`\CLARION5\EXAMPLES\TUTOR`. Los ofrecemos para que pueda ver el resultado final del cursillo y compararlo con su aplicación. Solamente deberían presentarse diferencias superficiales.

Si usted ya es un programador Clarion experto, puede examinar los archivos terminados, más que hacer el cursillo por sí mismo. Sin embargo, le recomendamos que al menos *lea* completamente este cursillo, dado que hay métodos de trabajar en el entorno que no pueden adquirirse simplemente " tirándose a la piqueta" y trabajando con las herramientas. Este cursillo demuestra ciertas características del entorno de desarrollo que no aparecen en primera vista.

Nota: Cuando hay múltiples maneras de hacer la misma tarea, este cursillo mostrará varias, para demostrar la flexibilidad del entorno de desarrollo Clarion.

Creación del Diccionario

Siempre que se crea una nueva aplicación, primero hay que definir el diccionario de datos (archivo .DCT). El Generador de Aplicaciones toma toda su información del diccionario: los archivos de datos que usa la aplicación, sus relaciones mutuas, además de información adicional como el formateo predefinido de los controles.

Por regla general, cuanto más previsión y trabajo se ponga en diseñar el diccionario de datos, más podrá hacer el Generador de Aplicaciones. **Y, cuanto más trabajo haga el Generador de Aplicaciones por usted, ¡menos tendrá que hacer usted mismo !**

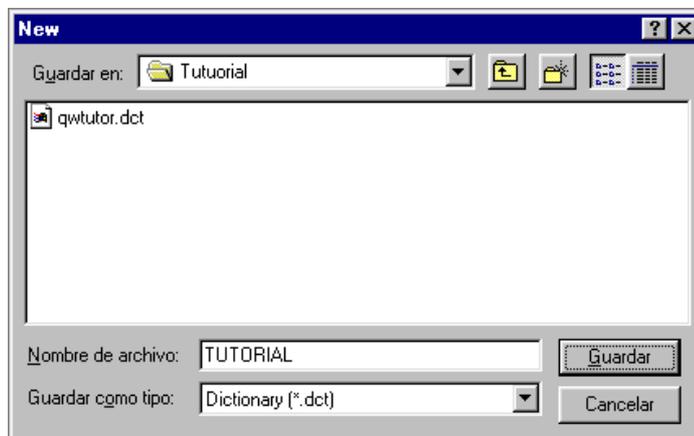
Cuando en el cursillo anterior, ejecutamos el Asistente para un Rápido Comienzo, simplemente definimos el archivo de datos. Ahora, presentamos el Editor del Diccionario.

Punto Inicial :

Debe tener abierto el entorno de desarrollo Clarion, y cerrada la caja de diálogo Pick (Escoger).

Indicar el nombre del nuevo diccionario

1. Elija **File** ➤ **New** del menú.
2. Dé **CLIC** sobre la opción **Dictionary**.



Aparece el diálogo **New**.

Dar nombre al nuevo archivo de diccionario

1. Seleccione el directorio o carpeta \CLARION5\ TUTORIAL
2. Escriba *TUTORIAL* en el campo **File Name**.

Clarion agrega la extensión: el archivo de diccionario tiene por nombre completo TUTORIAL.DCT. Puede usar nombres de archivos largos si utiliza un sistema operativo de 32 bits (Windows 95 o NT). La versión en inglés de este cursillo no usa nombres largos, para

hacerlo válido para todos los usuarios, incluyendo los de Windows 3.x. Como hemos explicado, en esta versión castellana ocasionalmente permitimos que la longitud exceda los ocho caracteres, para facilitar la lectura. Los usuarios de sistemas operativos que no admitan esta mayor longitud, deberán truncar los nombres respectivos.

Nota: El Diccionario de Datos permite ingresar nombres de archivos y de campos con acentos, como "**Teléfonos**", y éstos son procesados correctamente por el Generador de Aplicaciones. Al verificar la traducción de este manual descubrimos, sin embargo, que algunas partes de Clarion no funcionan bien con acentos; y que causan problemas difíciles de diagnosticar. Aconsejamos no usar acentos, en nombres de archivos, campos o claves. Sí pueden utilizarse (y es aconsejable hacerlo) en la interfaz de usuario: etiquetas de campo, encabezamientos de columnas, leyendas explicativas, etc. Estos elementos pueden predefinirse sin dificultades en el diccionario de datos (HDC).

3. Oprima el botón **Save** para crear el archivo.

Esto crea un archivo de diccionario de datos vacío. La barra de título muestra el nombre de archivo.

Especificar una descripción del diccionario

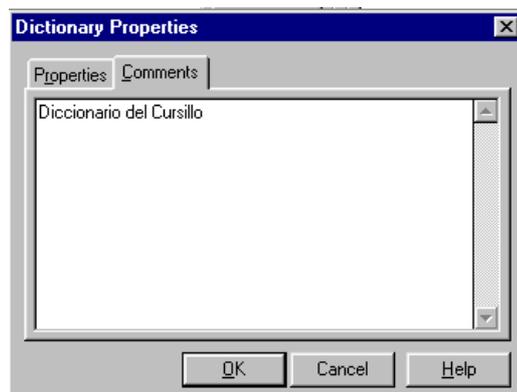
1. Oprima el botón **Dictionary Properties** (Propiedades del Diccionario).

Aparece la caja de diálogo respectiva.

2. *Seleccione la lengüeta **Comments**, y escriba* Diccionario del Cursillo.

La lengüeta **Comments** permite escribir notas de texto libre sobre el diccionario. Es opcional, pero extremadamente útil para programadores que quizás deban volver a revisar un proyecto después de varios meses de intervalo.

Este diálogo también ofrece un botón de contraseña (**Password**), que evita que otros usen este diccionario. No hay necesidad de llenarlo durante el Cursillo, pero es una característica útil para tener en cuenta.



3. Cierre la caja de diálogo **Dictionary Properties** oprimiendo el botón **OK**.

Copiado de archivos de uno a otro diccionario

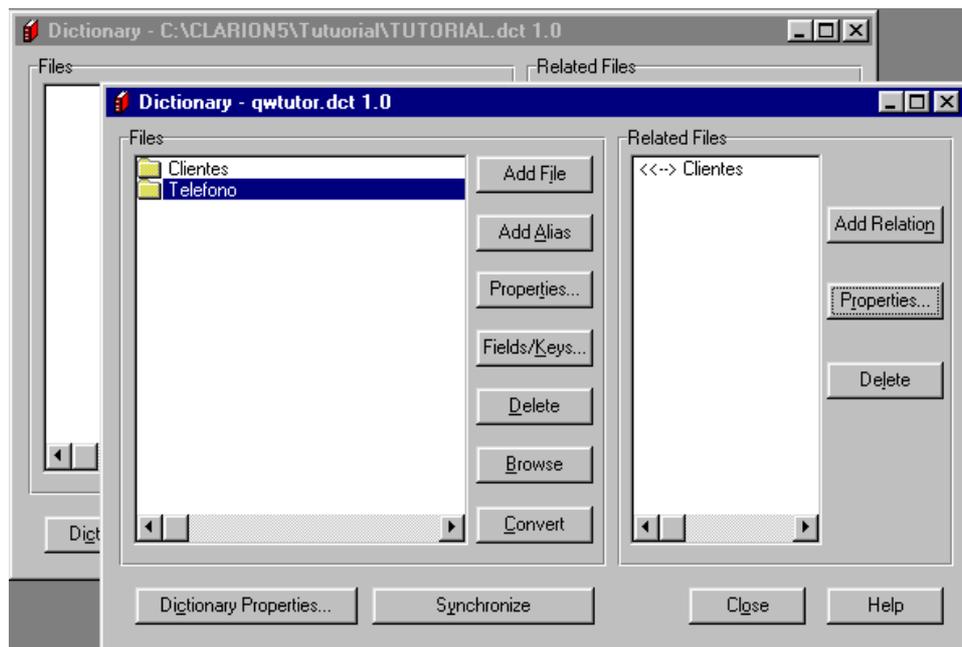
Puede usar las órdenes normales de copiar y pegar de Windows para copiar las definiciones de archivos de otro diccionario (o para copiar campos de un archivo a otro). **En otras palabras, una vez que está definido... Para que preocúpese por redefinirlo si basta con copiar lo que ya se ha hecho!**

Copia de la definición del archivo Clientes

Abra el otro diccionario de datos, elija un archivo, y cópielo

1. Elija **File** ➤ **Pick** del menú, luego elija la lengüeta **Dictionary**.
2. Seleccione el archivo *QWTUTOR.DCT* del listado de archivo, y oprima el botón **Select**.

Se abre otra caja de diálogo de Diccionario, que contiene las definiciones de archivo de la aplicación realizada con Quick Start. Muévela, y verá que ambos archivos .DCT están abiertos.



3. Elija el archivo **Clientes** de la lista **Files**.
4. Elija **Edit** ➤ **Copy** (u oprima CTRL+C)
5. Dé CLIC sobre la caja de diálogo del diccionario *TUTORIAL.DCT*.
Esto convierte al diccionario Tutorial en activo.

6. Elija **Edit** ➤ **Paste** (o pulse CTRL+V).

Aparece la caja de diálogo **Edit File Properties**.

Esta operación no solamente copia la definición del archivo, también copia los campos y claves.

7. Oprima el botón **OK** para cerrar la caja de diálogo **Edit File Properties**.

Copia de la definición del archivo de Teléfonos

Utilicemos ahora las órdenes de copiar y pegar para copiar la otra definición de archivos.

Elija el archivo y cópielo

1. Dé CLIC sobre la caja de dialogo del diccionario *QWKTUTOR.DCT*.
2. Elija el archivo **Teléfonos** de la lista **Files**.
3. Elija **Edit** ➤ **Copy** (u oprima CTRL+C).
4. Pulse CTRL+F6, o de CLIC sobre la caja de diálogos del diccionario *TUTORIAL.DCT*.
5. Elija **Edit** ➤ **Paste** (u oprima CTRL+V).
Aparece la caja de diálogo **Edit File Properties**.
6. Pulse el botón **OK** para cerrar la caja de diálogo **Edit File Properties**.

Cierre el archivo de diccionario creado por Quick Start

1. Dé CLIC sobre la caja de diálogo *QWKTUTOR.DCT*.
2. Oprima el botón **Close**, o elija **File** ➤ **Close**.

Vinculación de los archivos y configuración de las opciones de integridad referencial

Se pueden copiar las definiciones (incluyendo sus claves), pero Clarion no puede copiar las relaciones entre archivos de otros diccionarios. Por lo tanto, debe redefinir las relaciones entre esos dos archivos.

Definir el primer lado de la relación

Definiremos la relación de modo diferente que lo hicimos en el ejercicio anterior. Esta vez lo haremos desde el lado de Muchos (“hijo”) en lugar del lado Uno (“Padre”), sólo para demostrar que puede hacerse.

1. Destaque el archivo *Teléfonos*, y añada el botón **Add Relation**.

Aparece la caja de dialogo **New Relationship Properties**. Dado que el último archivo seleccionado fue *Teléfonos*, estableceremos la relación desde su perspectiva.

Cada Cliente puede tener más de un teléfono. “Cliente” es el Padre, en una relación *padre-hijo*. Por lo tanto, es una relación de muchos a uno (**MANY:1**), vista desde Teléfonos a Cliente.

2. Elija **MANY:1** de la lista desplegable **Type** en el grupo **Relationship for Teléfonos**.
3. Elija *KeyClienteNum* de la lista desplegable **Foreign Key**.



Esta es la clave que se vincula a una clave primaria en el archivo de Clientes; por lo tanto, se trata de una clave foránea.

Definición del otro lado de la relación

1. Elija *Clientes* de la lista desplegable **Related File** en el grupo **Parent**.
2. Elija *KeyClienteNum* de la lista desplegable **Primary Key**.

En la relación *padre-hijo*, la clave foránea del hijo debe relacionarse con una clave primaria en el padre. Véase *Anatomía de una base de datos*, en el Capítulo 4 de esta sección, y el artículo *Databasabe Design* en la *Programmer's Guide*, para más información sobre la teoría de las bases de datos.

3. Pulse el botón **Map by Name** para vincular los campos.

Puede usar este botón porque dio el mismo nombre a los campos que se vinculan en ambos archivos. Esta es una muy buena práctica para adoptar dado que, a largo plazo, hace la aplicación mucho más fácil para mantener. Cuando uno regresa para hacer cambios a un proyecto que terminó tiempo atrás, es mucho más fácil reconocer los campos vinculados, si tienen el mismo nombre a ambos lados de la relación.

Configuración de las restricciones de integridad referencial

Al indicar estas restricciones, se puede especificar cómo el Generador de Aplicaciones escribe el código fuente que maneja lo que ocurre si un usuario final intenta modificar un valor en una clave primaria, o si intenta borrar un registro “padre” con “hijos”. Si no se establecen las restricciones, el usuario podría comprometer la integridad de la base de datos creando registros “huérfanos”, de dos maneras: borrando un registro “padre” o cambiando el valor del campo relacional. En las aplicaciones que se generan con las plantilla de Clarion Application Builder Class, el código necesario para mantener esta integridad está incorporado a la biblioteca de construcción de aplicaciones (ABC).

Para este cursillo, especifique que la aplicación deberá *actualizar* la clave foránea, si se cambia el valor de la clave primaria. También, especificaremos que *no podrá permitirse* al usuario borrar un padre que tenga hijos.

1. Elija **Cascade** (“En cascada”) de la lista **On Update** (“Cuando se modifica”) en el grupo **Referential Integrity Constraints**.

Hacer un cambio en *cascada* significa que la aplicación extiende el cambio y actualiza la clave foránea (del archivo hijo), en todos los registros hijos vinculados a ese registro padre.

2. Elija **Restrict** (“Restringir”) de la lista **On Delete** (“Cuando se borra”)

Restringir el borrado quiere decir que la aplicación no permite eliminar un padre con hijos (el usuario debe borrar primero a los hijos).

3. Presione **OK** para cerrar la caja de diálogo **New Relationship Properties**.



En este punto, el Diccionario se ve así:



Observe las pequeñas flechas a la derecha del nombre de archivo vinculado en la lista **Related Files**. Indica la naturaleza de la relación entre los dos archivos.

Dos signos (>>o<<) apuntan al lado “muchos” desde el lado “uno”. Un signo (>o<) apuntan al lado “uno” desde el lado “muchos”. Por lo tanto, **Teléfonos <<-> Clientes** indica que *muchos* registros de teléfonos pueden estar vinculados a *un* registro de Cliente.

Guarde su trabajo

*Es un buen hábito guardar el trabajo frecuentemente, mientras desarrolla las aplicaciones (las fallas y cortes de luz vienen sin previo aviso). Para hacerlo, elija **File** ➤ **Save**, u oprima el botón Guardar en la barra de herramientas. Esto escribe el archivo del diccionario al disco.*

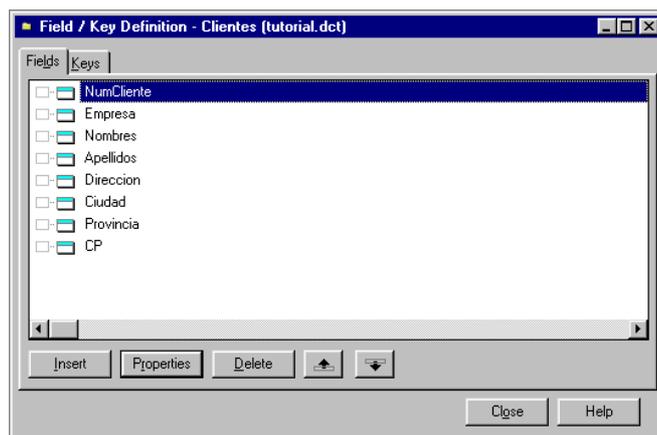
Pre – definición de los formatos de control de ventanas

Dentro del diccionario de datos, se pueden especificar las propiedades por omisión de los controles de las ventanas que servirán para modificar los campos que usted defina. También se pueden especificar ciertas reglas de integridad de datos, configurando las Comprobaciones de Validez (*Validity Checks*) y el valor inicial del campo. **Estas opciones son factores clave para ayudar al Generador de Aplicaciones a hacer la mayor parte del trabajo para usted.**

Acceso a la caja de diálogo Field Properties

1. Destaque el archivo *Cientes* en la lista **Files**.
2. Pulse el botón **Fields/Keys**.

La caja de diálogo **Field/Key Definition** le permite modificar las propiedades de cualquier campo o clave en el archivo.



3. Elija el campo *Provincia*, y oprima el botón de **Properties**.

Aparece la caja de diálogo **Edit Field Properties**, con la opción Quick Start preseleccionada en este campo.

Configurar las Comprobaciones de Validez

1. Elija la lengüeta **Validity Checks**.

Esta ficha le permite configurar límites para los campos numéricos, especificar que el valor de un campo debe ser igual al mismo campo en otro archivo relacionado, que debe ser verdadero o falso o, como en este caso, que el valor del campo debe estar en una lista que usted especifica en esta caja de dialogo.

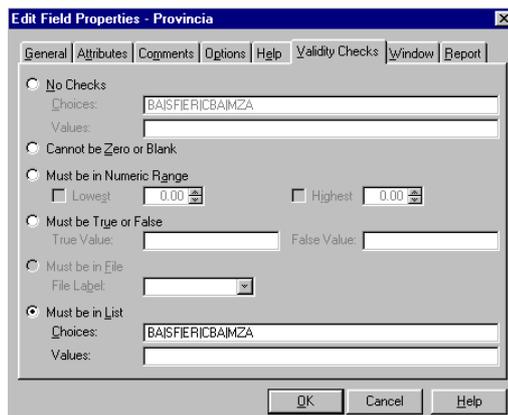
2. Elija el botón de radio **Must be in List** (“debe figurar en la lista”)
3. Escriba lo siguiente en la caja de opciones (**Choices**):

BA | SF | ER | CBA | MZA

Cada opción debe estar separada por una barra vertical (|).

Así se define la lista de opciones permisibles. En este caso, el diccionario especifica que solamente las abreviaturas de esas cinco provincias argentinas son aceptables. Usted especificará que el control por omisión para este campo es una lista desplegable.

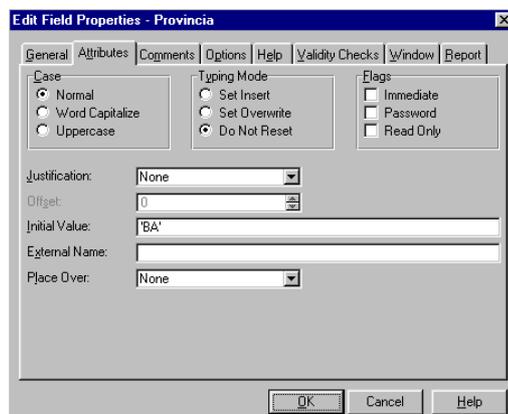
En el teclado en español, la barra vertical se obtiene oprimiendo ALT (botón de la derecha) + 1 (del teclado normal, no numérico).



Configurar un valor por omisión

1. Seleccione la lengüeta **Attributes**.
2. Escriba 'BA' el campo **Initial Value** (“Valor inicial”), incluyendo las comillas simples (en el teclado Windows en español es la tecla a la derecha del número 0)

Así se especifica que, siempre que aparezca el control, el valor por omisión será “BA”. Los valores iniciales pueden ahorrar mucho tiempo para el usuario final; en este caso, si la mayoría de los clientes están ubicados en “BA”, evita que deban buscarlo en la lista cada vez que se añade un nuevo cliente. Las comillas simples son necesarias porque también es posible nombrar una variable o función como valor inicial de un campo, que irían sin comillas (advertida que hay reglas algo distintas para los valores iniciales de variables en memoria). Como en este caso, el valor inicial es una cadena de caracteres, las comillas simples la identifican como tal.



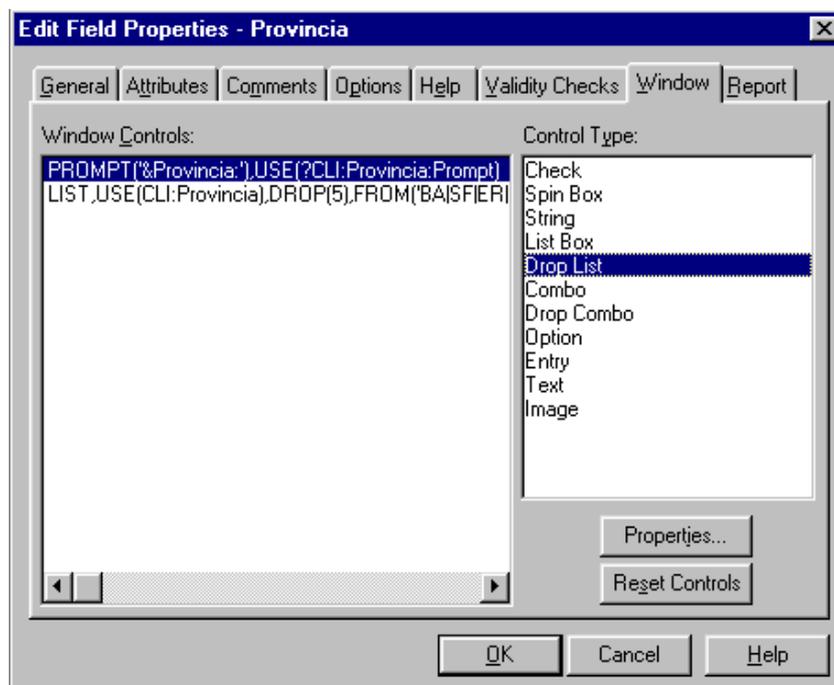
Especifique un control de ventana por omisión

1. Elija la lengüeta **Window**.

Cuando se especifica **Must be in List**, la ventana por omisión del campo es una estructura OPTION con botones de RADIO. Estos aparecen como default en la lista **Windows Controls**.

2. Elija **Drop List** (“Lista desplegable”) en la caja **Control Type**.

La lista de **Window Controls** muestra ahora solamente un PROMPT (“Indicador”) y un control LIST (“lista”) con el atributo DROP (“desplegable”).



3. Oprima el botón **OK** para cerrar la caja de diálogo **Edit Field Properties**.
4. Oprima el botón **Close** para cerrar la caja de diálogo **Field/Key Definition**.
5. Elija **File** ➤ **Save**, u oprima el botón *Guardar* en la barra de herramientas.

En el capítulo siguiente, aprenderá cómo añadir un archivo al diccionario de datos, comenzando completamente “desde cero”, sin usar Quick Start o Quick Load. Comprobará lo fácil y rápido que es, aún sin usar los Asistentes.

3 - INCORPORACION DE ARCHIVOS Y CAMPOS

Definición de archivos de datos nuevos

Después de copiar y modificar los dos archivos definidos en la aplicación de Inicio Rápido, está listo para añadir un nuevo archivo desde cero.

Punto de Inicio:

El archivo TUTORIAL.DCT (diccionario) debe estar abierto.

Creación del archivo de Pedido

Especificación del rótulo, el prefijo y la descripción

1. Pulse el botón **Add File**, en la caja de diálogo **Dictionary**.
2. Cuando aparece la caja de dialogo **Add File**, con la pregunta si desea usar Quick Load, oprima el botón **No**.

Aparece la caja de diálogo **New File Properties**. Clarion utiliza la información que se incorpora aquí para declarar la estructura del archivo(FILE).

3. Escriba *Pedidos* en el campo **Name**, y oprima TAB.

Este campo solamente acepta un rótulo válido para Clarion, que identifique unívocamente la estructura de datos. Un rótulo puede contener solamente, letras, números, y el carácter de subrayado(_) o los dos puntos y debe empezar con una letra o subrayado. Las sentencias del código ejecutable usan este rótulo para referirse al archivo.

Después de oprimir TAB, aparece automáticamente "PED" en el campo **Prefix**. El prefijo es un modo de identificar inequívocamente los campos que tienen el mismo nombre en distintos archivos de datos. Por ejemplo, *PED:NumCliente* es el campo *NumCliente* en el archivo *Pedidos*, mientras que *CLI:NumCliente* es el campo del mismo nombre en el archivo *Clientes*. También puede identificar de forma unívoca los campos utilizando la sintaxis de Calificación de Campos (tratada en *Language Reference*).

4. Escriba *Cabecera de los pedidos* en el campo **Description**.

Esta descripción aparece junto al rótulo del archivo de datos, en la lista del Diccionario. Si elige la lengüeta **Comments**, puede escribir una descripción más larga. Esta descripción puede resultar muy útil cuando se vuelve al archivo para realizar mantenimiento del programa.

Elección del controlador de archivo

1. Elija TopSpeed de la lista desplegable **File Driver**.

Esta acción declara el formato de archivo para los datos como TopSpeed. Este es el mas nuevo de los dos formatos propios que la Corporación TopSpeed a desarrollado (el otro es Clarion).

La sección *Database Driver* del *Application Handbook* provee información sobre que clase de datos admite cada controlador, además de otra información útil como las extensiones de archivo por omisión para los archivos de datos y/o índices. También ofrece sugerencias y trucos para elegir el controlador adecuado para la tarea, cuáles controladores son mejores cuando su aplicación debe manejar una gran base de datos que se actualiza frecuentemente, o qué controladores son mejores cuando lo que más importa es el menor tiempo de búsqueda.

2. Oprima el botón **OK**.

Puede aceptar los valores por omisión de todas las demás opciones. Se cierra la caja de diálogo, y la caja **Dictionary** muestra el archivo Pedidos, con la aclaración “*Cabecera de los pedidos*” a su lado.

Dar nombre a los archivos Detalle y Productos

Creación del archivo Detalle

1. Oprima el botón **Add File** en la caja de diálogo **Dictionary**.

Elija **No** cuando se le pregunta si desea usar Quik Load (“Carga Rápida”).

2. Escriba *Detalle* en el campo **Name**.
3. Escriba *Detalle (líneas) de pedidos* en el campo **Description**.
4. Escriba *DETAL* en el campo **Prefix**.

Esta personalización del prefijo por omisión (cambiarlo de “DET” a “DETAL”) hace más legible el código. Aunque lo convencional para los prefijos son tres caracteres, también puede ser más, como vemos aquí.

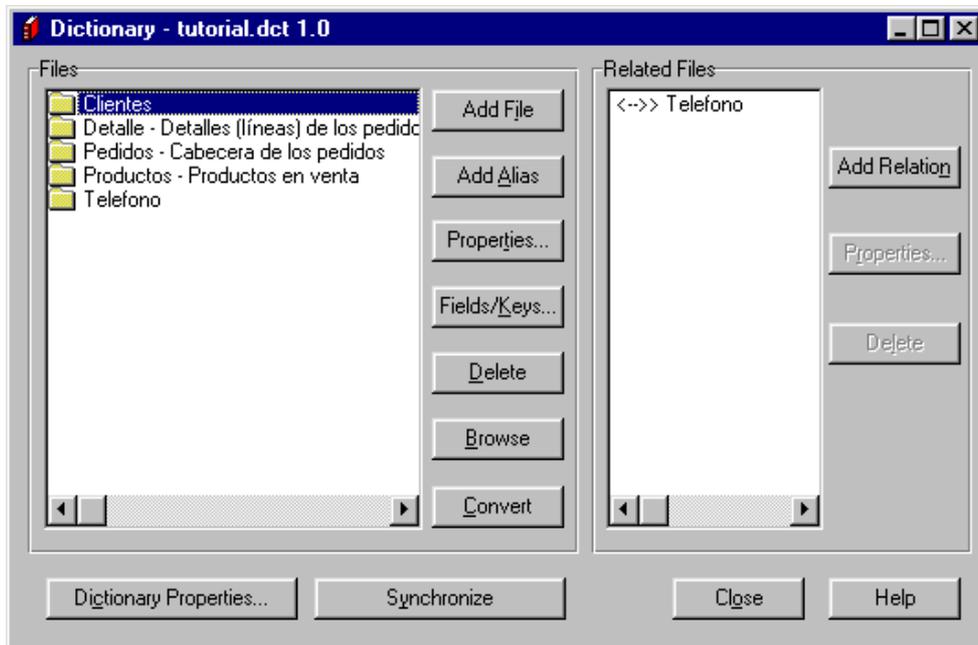
5. Elija **TOPSPEED** de la lista desplegable **FileDriver**.

Acepte todas las demás opciones preconfiguradas.

6. Oprima el botón **OK**.

Creación del archivo Productos

1. Oprima el botón **AddFile** (no utilice Quick Load).
2. Tipee *Productos* en el campo **Name**.



3. Típee *Productos en venta* en el campo **Description**.
4. Escriba **PROD** en el campo **Prefix**.
5. Elija **TOPSPEED** de la lista desplegable **File Driver**.
6. Oprima el botón **OK**.

Guarde su trabajo

1. Elija **File** ➤ **Save**, u oprima el botón *Guardar* de la barra de herramientas.
En este punto, la caja del Diccionario se ve así:

Definición de los campos

Definición de un Pool de Campos

En este momento definiremos varios campos que serán campos de vínculo entre los archivos de la base de datos. Usaremos una facilidad del Diccionario de Datos llamada Pool de Campos (Field Pool) para asegurarnos que éstos están definidos exactamente igual en los archivos que los requieran.

1. Presione el botón **Add File**.

Un Pool de Campos es tratado en el Diccionario de Datos como un archivo, aún cuando no general código alguno en las aplicaciones. Usted puede definir en el Diccionario de Datos tantos pools de campos como desee, sin embargo no es usualmente necesario emplear más de uno.

2. Elija la opción **Pool**.
3. Tipee *CamposPool* en el campo **Name**.
4. Escriba *POOL* en el campo **Prefix**.
5. Oprima el botón **Ok**.

Abra la venta de Definición de Campos y Claves

1. Ilumine el archivo **CamposPool** en la lista **Files**.
2. Oprima el botón **Field/Keys...**

Esta ventana muestra todos los campos y las claves definidas en un archivo. Ya que se trata de un archivo nuevo está en blanco. Podríamos empezar añadiendo campos, pero en su lugar comenzaremos copiando un campo del archivo de clientes.
3. Dé CLIC sobre la barra de título de la ventana **Field/Key Definition**, y arrastre para poder ver la caja de diálogo del Diccionario que está abajo.
4. Destaque el archivo **Clientes** en la lista **Files** del Diccionario. Advierta que el foco se desplaza a esta ventana
5. Presione el botón **Fields/Keys...**

Ahora aparece otra venta **Field/Key Definition** con los campos definidos en el archivo **Clientes**.

Eleccion y copia de un campo

1. Elija el campo **NumCliente** de la lista **Fields**.
2. Elija **Edit > Copy** (u oprima CTRL + C).
3. Elija **Window > Field/Key Definition – Campos Pool** (o dé CLIC sobre la ventana abierta para ponerla en foco).
4. Elija **Edit > Paste** (u oprima CTRL + V).

Aparece la Caja de Diálogo **Edit Field Properties**.

5. Oprima **OK** para cerrar la Caja de Diálogo **Edit Field Properties**.

Este es el campo que oficiará de vínculo para la relación entre los archivos Clientes y Pedidos. Los campos de vínculo entre distintos archivos siempre se definen iguales, de manera que copiar la definición es una forma de ingresar la definición existente en el Pool de Campos.

Derivando un campo existente

1. Seleccione **Window > Field/Key Definition – Clientes** (o dé CLIC sobre la ventana abierta para ponerla en foco).
2. Destaque el campo **NumCliente** de la lista **Fields** y presione el botón **Properties**.

Aparece la caja de diálogo **Edit Field Properties**.
3. Escriba **POOL:Cliente** en el campo de **Derived From**.

Esto significa que el campo CLI:NumCliente es ahora derivado del campo POOL:NumCliente. El término “derivado de” significa que la definición del campo POOL:NumCliente es el “padre” y todos los campos “hijos” que son derivados de él, automáticamente comparten todos sus atributos.

El derivar la definición de los campos de campos existentes, le brinda la posibilidad de hacer los cambios en un solo lugar y aplicarlos en cascada a todos los campos derivados. Por ejemplo, si necesitaríamos cambiar la definición del campo NumCliente en todos los archivos que lo emplean, simplemente haríamos los cambios en la definición de POOL:NumCliente y los aplicaríamos en cascada a todos los campos derivados. Esto puede hacerse eligiendo **Edit > Distribute Field** luego de efectuar los cambios en POOL:NumCliente.

4. Oprima **OK** en la caja de diálogo **Edit Field Properties**.

Añadir el resto de los campos al Pool de Campos

1. Seleccione **Window > Field/Key Definition – Campos Pool** (o simplemente dé CLIC sobre la ventana abierta para darle foco).

2. Presione Insert para abrir la caja de diálogo New Field Properties.

Una vez que se ha comenzado a definir campos nuevos aparece automáticamente una caja de diálogo **New Properties** después de añadir cada campo. Esto acelera el proceso de añadir campos múltiples. Después de añadir el último, simplemente debe apretarse el botón **Cancel** sobre la caja de diálogo en blanco para regresar a la caja **Field/Key Definition**.

3. Escriba *NumPedido* en el campo **FieldName**.

Este campo será el vínculo entre los archivos Pedidos Y Detalle.

4. Elija **SHORT** en la caja **Data Type**.

Esto especifica un entero corto (-32.769 a 32.767).

5. Presione el botón **OK**.

6. Escriba *NumProducto* en el campo **Field Name**.

Este vinculará el archivo Detalle con el de Productos.

7. Seleccione **SHORT** de la lista desplegable para el campo **Data Type**.

8. Presione el botón **OK**.

9. Presione el botón **Cancel**.

Reaparece la caja de diálogo **Field/Key Definition**.

10. Presione el botón **Close**.

Definición de los campos del archivo de Pedidos

En este punto, regrese al archivo Pedidos y prepárese a definir los campos.

Abrir las ventanas Field/Key Definition.

1. Destaque el archivo *Pedidos* de la lista **Files**.
2. Oprima el botón **Fields/Keys...**

3. Oprima el botón **Insert** para abrir la caja de diálogo **New Field Properties**.
4. Escriba *NumCliente* en el campo **Field Name**.
5. Presione el botón elipsis (...) a la derecha del campo **Derived From**.

Aparece un ventana de selección que contiene una lista, en forma de árbol, con todos los campos ya definidos en el diccionario. Usted puede derivar un campo nuevo de otros existente, ya sea en un archivo, global data o pool de campos.

6. Ilumine el campo **NumCliente** en CamposPool y presione el botón **Select**.

El nuevo campo será automáticamente una copia perfecta del campo del cual fue derivado. El botón con un ícono igual a una flecha circular, a la derecha del botón elipsis (...), permite refrescar el campo derivado según la definición del campo “padre” si fuera necesario (pero es mucho más simple elegir **Edit > Distribute Field** luego de modificar el campo “padre”).

7. Presione el botón **OK** para cerrar la caja de diálogo **New Field Properties**.

Este campo oficiará de vínculo entre el archivo de Pedidos y de Clientes.

Derivar el campo NumPedido

Este proporciona una identificación única para cada pedido.

1. Escriba *NumPedido* en el campo **Field Name**.
2. Presione el botón elipsis (...) a la derecha del campo **Derived From**.
3. Destaque el campo NumPedido de CamposPool y oprima el botón **Select**.
4. Presione el botón **OK** para cerrar la caja de diálogo **New Field Properties**.

Definir el campo Monto Factura

Este campo guarda el valor total del pedido.

1. Escriba *MontoFactura* en el campo **Field Name**.
2. Elija **DECIMAL** de la lista desplegable **Data Type**.
3. Escriba 7 en el campo **Characters**.

Esto especifica el número total de dígitos en el número (a ambos lados de la coma decimal).

4. Escriba 2 en el campo **Places** (“Lugares decimales”).

The screenshot shows the 'New Field Properties' dialog box with the following fields and values:

- Field Name:** MontoFactura
- Derived From:** (empty)
- Description:** (empty)
- Data Type:** DECIMAL
- Base Type:** (empty)
- Characters:** 7
- Places:** 2
- Dimensions:** 0, 0, 0, 0
- Record Picture:** (empty)
- Screen Picture:** @n-10.2
- Prompt Text:** Monto Factura:
- Column Heading:** Monto Factura
- Freeze:**

5. Oprima el botón **OK**.

Definición del campo Fecha

El campo guarda la fecha en que se realizó el pedido.

1. Escriba *FechaPedido* en el campo **FieldName**.

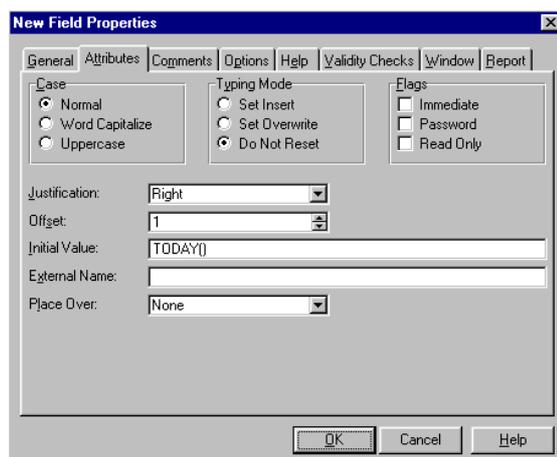
2. Escoja **LONG** de la lista desplegable **Data Type**.

LONG es el tipo preferido para guardar fechas con el controlador de TopSpeed. Allí se contiene un valor de Fecha Estándart Clarion (véase *Language Reference* para más información sobre las Fechas y Horas Estándar de Clarion).

3. Tipee @d5 en el campo **Screen Picture** (“patrón de visualización”).

Este patrón de visualización especifica el formateo por omisión de los “caracteres” del campo. En este caso @d5 significa formato de fecha DD/MM/YY. La caja de diálogo muestra una representación del texto formateado junto al campo.

4. Seleccione la lengüeta de Atributos (*Attributes*) y escriba TODAY() en el campo **Initial Value**.



El código generado coloca la fecha de hoy en cualquier control, cuando adiciona un registro, utilizando la función incorporada TODAY().

5. Pulse el botón **OK**.

Definición del campo Nota

Esto permite incluir una nota breve para instrucciones especiales del pedido.

1. Tipee *Nota* en el campo **FieldName**.

2. Elija **STRING** de la lista desplegable **Data Type**.

3. Escriba 80 en el campo **Characters**.

Esto especifica 80 caracteres.

4. Oprima el botón **OK**.

Cierre la caja de diálogo New Field Properties

Todos los campos están definidos, y estaría abierta ahora una caja de **diálogo New Field Properties**.

1. Oprima el botón **Cancel** para cerrar la caja de diálogo **New Field Properties**.
2. Presione el botón **Close** para cerrar la caja de diálogo **Field/Key Definition**.

Definición de los campos del archivo Detalle

En este punto, definiremos los campos del archivo Detalle.

Derivar la definición del campo vínculo

1. Destaque el archivo Detalle en la lista **Files** y oprima el botón **Insert** para abrir la caja de diálogo **New Field Properties**.
2. Escriba *NumPedido* en el campo **FieldName**.
3. Presione el botón elipsis (...) a la derecha del campo **Derived From**.
4. Destaque el campo NumPedido en CamposPool y oprima el botón **Select**.
5. Presione el botón **OK** para cerrar la caja de diálogo **New Field Properties**.

Este campo será el vínculo entre los archivos Pedidos y Detalle.

Derivando el campo NumProducto

Este campo permite relacionar este archivo y el de Productos.

1. Tipee *NumProducto* en el campo **Field Name**.
2. Presione el botón elipsis (...) a la derecha del campo **Derived From**.
3. Destaque el campo NumProducto en CamposPool y oprima el botón **Select**.
4. Presione el botón **OK** para cerrar la caja de diálogo **New Field Properties**.

Definición del campo Cantidad

Este campo guarda la cantidad pedida.

1. Escriba *Cantidad* el campo **Field Name**.
2. Elija **SHORT** como el tipo de dato (**Data Type**)
3. Oprima el botón **OK**.

Definición del campo ProdPrecio

Esto guarda el precio unitario del producto al momento del pedido.

1. Tipee *ProdPrecio* en el campo **Field Name**.
2. Elija **DECIMAL** como el tipo de dato.
3. Escriba 5 en el campo **Characters**.
4. Escriba 2 en el campo **Places**.

5. Oprima el botón **OK**.

Definición del campo *TasaImponible*

6. Típee *TasaImponible* en el campo **Field Name**.

7. Elija **DECIMAL** como el tipo de dato.

8. Escriba 4 en el campo **Characters**.

9. Escriba 2 en el campo **Places**.

10. Oprima el botón **OK**.

Cierre la caja de diálogo *New Field Properties*

Todos los campos están definidos, y estaría abierta ahora una caja de **diálogo *New Field Properties***.

1. Oprima el botón **Cancel** para cerrar la caja de diálogo ***New Field Properties***.

2. Presione el botón **Close** para cerrar la caja de diálogo ***Field/Key Definition***.

Definición de los campos del archivo de Productos

En este punto, definiremos los campos del archivo de Productos. Este es el último archivo.

Derivar la definición del campo vínculo

1. Destaque el archivo **Productos** en la lista **Files** de la caja de diálogo **Dictionary**.

Puede que sea necesario mover algunas ventanas para hacerlo. El foco se desplaza a la caja de diálogo del diccionario.

2. Oprima el botón **Field/Keys...**

Ahora se abre una nueva ventana **Field/Key Definition** con los campos definidos para el archivo de Productos.

3. Presione el botón **Insert** para abrir la caja de diálogo ***New Field Properties***.

4. Escriba *NumProducto* en el campo **Field Name**.

5. Presione el botón elipsis (...) a la derecha del campo **Derived From**.

6. Destaque el campo *NumProducto* en **CamposPool** y oprima el botón **Select**.

7. Presione el botón **OK** para cerrar la caja de diálogo ***New Field Properties***.

Este campo será el vínculo entre los archivos *Detalle* y *Productos*.

Definición del campo Descripción

Este campo lleva la descripción del producto.

1. Escriba *Descripción* en el campo **FieldName**.

2. Elija **STRING** de la lista desplegable **Data Type**.

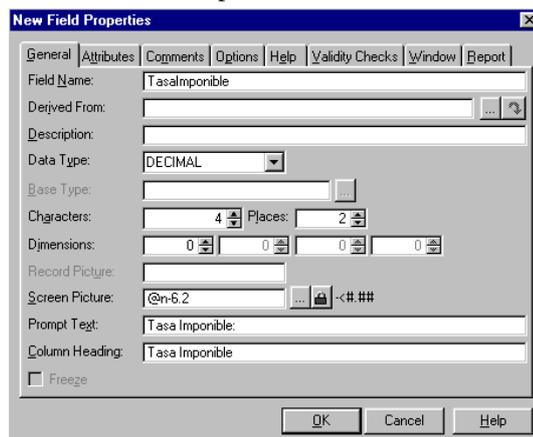
3. Escriba 25 en el campo de caracteres (**Characters**).
4. Oprima el botón **OK**.

Definición del campo Precio

1. Tipee *Precio* en el campo **Field Name**.
2. Elija **DECIMAL** como el tipo de dato.
3. Escriba 5 en el campo **Characters**.
4. Escriba 2 en el campo **Places**.
5. Oprima el botón **OK**.

Definición del campo TasalImponible

1. Tipee *TasalImponible* en el campo **Field Name**.
2. Elija **DECIMAL** como el tipo de dato.
3. Escriba 4 en el campo **Characters**.
4. Escriba 2 en el campo **Places**.



5. Oprima el botón **OK**.

Cierre de la caja de diálogo New Field Properties y guardar el trabajo

Todos los campos han sido definidos, y debería estar activo una nueva caja de diálogo **New Field Properties**.

1. Oprima el botón **Cancel** para cerrar esa caja de diálogo.
2. Oprima el botón **Close** en cada caja de diálogo **Field/ KeyDefinition** abierta.
3. Escoja **File** ➤ **Save** u oprima el botón *Guardar* de la barra de herramientas.

4 - INCORPORACION DE CLAVES Y RELACIONES

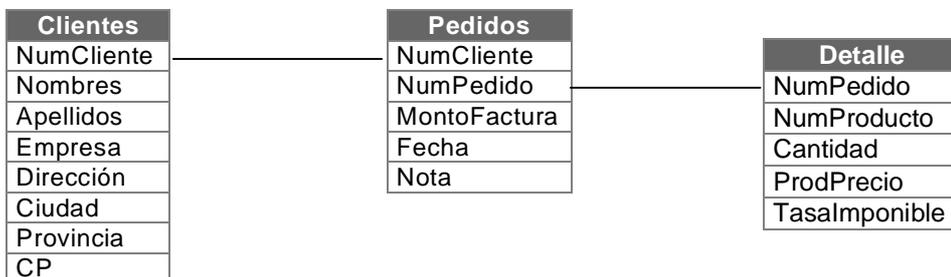
Ahora que tenemos definidos todos los archivos, podemos añadir claves y, después, especificar las relaciones entre los archivos. El Asistente de Inicio Rápido ya definió las claves de los dos archivos creados para la aplicación del primer cursillo. En este capítulo, definiremos las claves del resto de los archivos

Punto de Inicio:

El archivo TUTORIAL.DCT (diccionario) debe estar abierto.

Definición de claves del archivo *Pedidos*

Los campos en el archivo de Pedidos que se vinculan a otros archivos en la base de datos son los campos NumPedido y NumCliente.



- ◆ El campo NumPedido se vincula al archivo Detalle.

No deben existir duplicados o valores nulos (vacíos) de pedidos en ese archivo: se trata de su clave primaria.

Hay muchos registros de Detalle para un único número de Pedido. Por lo tanto, esta es una relación de *uno-a-muchos*: el archivo de Pedidos es el “Padre” del archivo de Detalle.

- ◆ El campo NumCiente se vincula al archivo de Clientes.

Habrà valores duplicados en el campo NumCliente. La clave que definimos en el archivo de Pedidos es *forànea*. La clave del archivo de Pedidos no permite duplicados y valores nulos, y fue definida como la clave principal de ese archivo.

Puede haber múltiples Pedidos de cada Cliente, lo que lo hace una relación de *muchos-a-uno*: el archivo de Pedidos es el “hijo” del archivo de Clientes.

Creación de la Clave Primaria

Dar nombre a la clave

1. Destaque el archivo *Pedidos* en la lista **Files**.
2. Oprima el botón **Field/Key...**
3. Elija la lengüeta **Keys**.
4. Oprima el botón **Insert**.

Aparece la caja de diálogo **New Key Properties**.

5. Escriba *KeyNumPedido* en el campo **Key Name**.

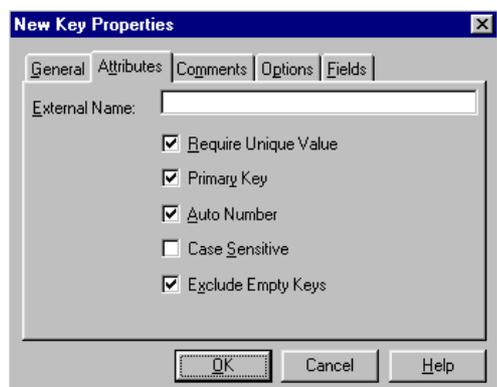
Como convención mnemotécnica, le sugerimos incorporar la palabra “key” y el nombre del campo en el nombre de la clave (como lo hace el Asistente para un Rápido Comienzo).

6. Seleccione la lengüeta **Attributes**, marque el casillero **Require Unique Value** (“Se requiere un valor exclusivo”), y también marque el casillero **Primary Key**.

Esto especifica que la clave es primaria. El código fuente generado automáticamente evita que el usuario ingrese valores duplicados o nulos.

7. Marque la caja **Auto Number**.

El código generado por los *templates* incrementará el número de clave a cada nuevo registro.



8. Seleccione la lengüeta **Fields**.

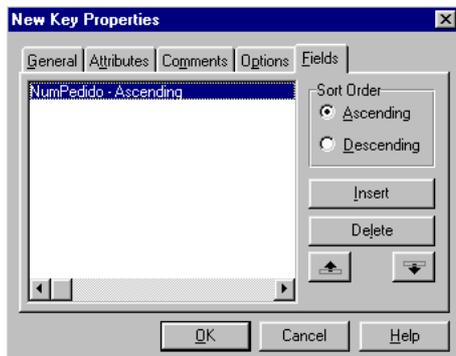
Designar el campo clave

1. Oprima el botón **Insert**.

Aparece la caja de diálogo **Insert Key Component** (“Indique componente de la clave”), para que usted especifique el campo o los campos que componen la clave.

2. Dé DOBLE CLIC sobre *NumPedido*.

Esto añade el campo a la lista de los campos componentes para esta clave.



3. Oprima el botón **OK**.

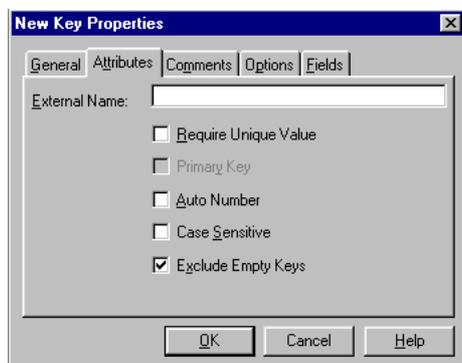
Aparece una nueva caja de diálogo **New Key Properties**, para especificar una nueva clave.

Designar una clave foránea

Ahora podemos definir la clave del NumCliente. Los duplicados son permisibles en este archivo. Se vincula a la clave primaria en el archivo Clientes, por lo que es una clave foránea.

1. Escriba *KeyNumCliente* en el campo **Key Name**.
2. Seleccione la lengüeta **Attributes**.

La clave permite duplicados, de manera que deje las configuraciones preestablecidas.



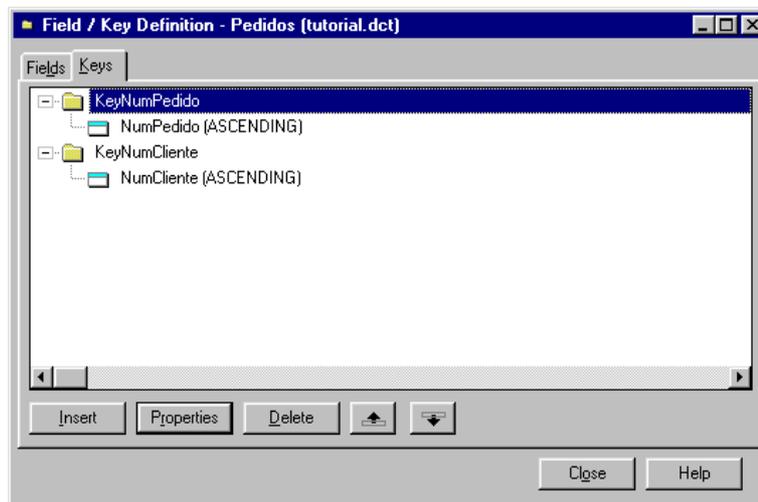
3. Seleccione la lengüeta **Fields**.
 4. Oprima el botón **Insert**.
5. Seleccione *NumCliente* y oprima el botón **Select**.
 6. Oprima el botón **OK**.

Aparece la caja de diálogo **Insert Key Component**, para que especifique un campo o campos para la clave.

Cada vez que termina de definir una clave, aparece una caja de diálogo **New Key Properties**, para que incorpore la próxima.

7. Oprima el botón **Cancel** para cerrar la caja de diálogo **New Key Properties**.

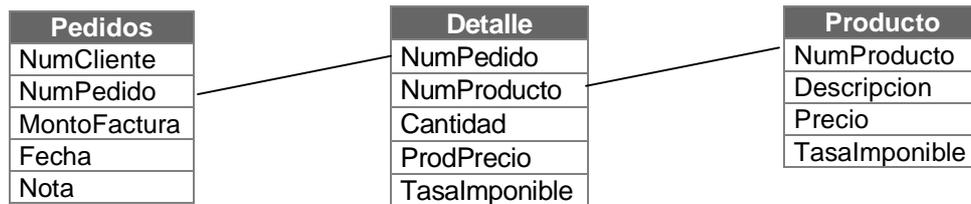
La caja de diálogo del archivo Pedidos ahora aparece así:



8. Oprima el botón **Close** para cerrar la caja de diálogo **Field/key Definition**.
9. Elija **File** ➤ **Save**, u oprima el botón *Guardar* de la barra de herramientas.

Definición de Claves del archivo de Detalle

Los campos en el archivo de Detalle que se vinculan a otros archivos en la base de datos son NumProductos y NumPedido.



- ◆ El campo NumPedido se vincula al archivo Pedidos.

Habrá valores duplicados en el campo NumPedido que se vincula a registros en el archivo de Pedidos. La clave que definimos en el archivo de Detalle es otra clave *foránea*. La clave del archivo Pedidos no permite duplicados ni valores nulos, y fue definida como clave primaria.

Puede haber mas de un registro de Detalle para un determinado Número de Pedido. Por lo tanto, se trata de una relación de *muchos-a-uno*, siendo el archivo Detalle el “hijo” del archivo Pedidos.

- ◆ El campo NumProducto se vincula al archivo Productos.

Habrá valores duplicados en el campo NumProducto en los registros del archivo Detalle. Puede haber más de un registro Detalle (línea del pedido) referenciado a un único Número de Producto. Por lo tanto, tenemos aquí otra relación de *muchos-a-uno*, siendo el archivo Detalle el “hijo” del archivo “Productos”.

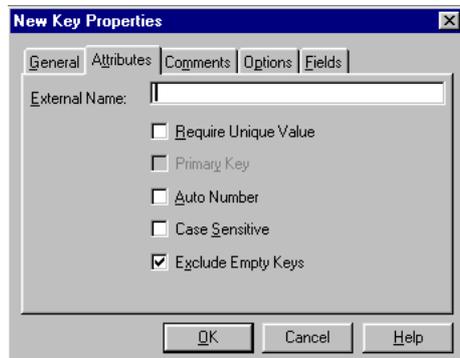
Definición de la primera clave foránea

Defina KeyNumProducto de manera que pueda haber duplicados valor NumProducto en este archivo.

1. Destaque el archivo *Detalle* en la lista **Files**.
2. Oprima el botón **Fiel/Keys...**
3. Seleccione la lengüeta **Keys**.
4. Oprima el botón **Insert**.
5. Tipee *KeyNumProducto* en el campo **Key Name**.
6. Seleccione la lengüeta **Attributes**.

Esta clave permite duplicados, de manera que deje todos los valores preestablecidos.

7. Seleccione la lengüeta **Fields**.



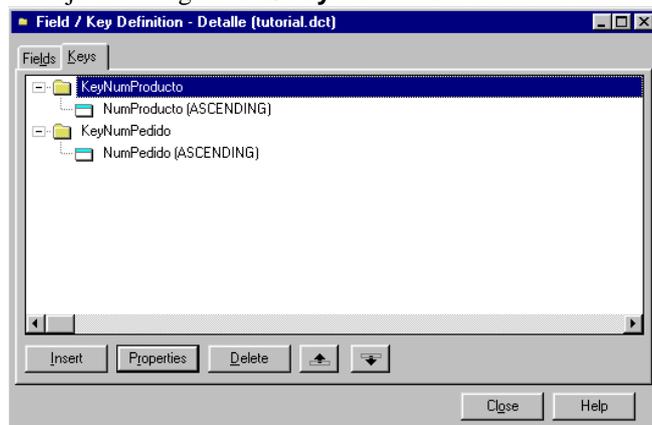
8. Oprima el botón **Insert**.
9. Elija *NumProducto* y haga DOBLE CLIC.
10. Oprima el botón **OK**.

Aparece una caja de diálogo **Key Properties** en blanco, esperando que usted especifique otra clave.

Definición de la segunda clave foránea

1. Tipee *KeyNumPedido* en el campo **KeyName**.
2. Seleccione la lengüeta **Fields**.
3. Oprima el botón **Insert**.
4. Seleccione *NumPedido* y haga DOBLE CLIC.
5. Oprima el botón **OK**.
6. Oprima el botón **Cancel** para cerrar la caja de diálogo **New Key Properties** en blanco.

La caja de diálogo **Field/Key Definition** del archivo de Detalle se ve ahora así:

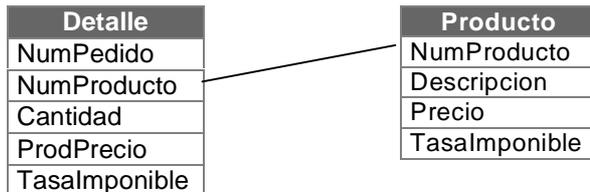


7. Oprima el botón **Close** para cerrar la caja de diálogo **Field/Key Definition**.
8. Elija **File** ➤ **Save**, u oprima el botón *Guardar* de la barra de herramientas.

Definición de claves para el archivo de productos

Solamente un campo en el archivo de Productos se vincula a otro archivo en la base de datos: el campo NumProducto.

- ◆ NumProducto se vincula al archivo Detalle.



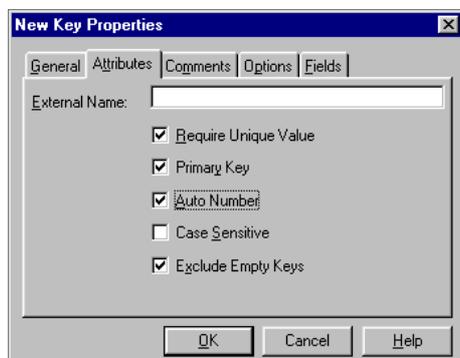
No debe haber números de pedido duplicados o nulos en el archivo de Productos, ya que es una clave *primaria*.

Para cada NumProducto en el registro puede haber muchos registros Detalle. Esta es una relación *uno-a-muchos*, en que el archivo Productos es “Padre” del archivo Detalle.

Crear la clave primaria

Nombrar la clave

1. Destaque el archivo *Productos* en la lista **Files**.
2. Oprima el botón **Field/Keys...**
3. Elija la lengüeta **Keys**.
4. Oprima el botón **Insert**.
5. Escriba *KeyNumProducto* en el campo **KeyName**.
6. Elija la lengüeta **Attributes**.



7. Marque el casillero **Requiere Unique Value**, y también el casillero **Primary Key**.
8. Marque el casillero **Auto Number**.

9. Elija la lengüeta **Fields**.
10. Oprima el botón **Insert**.
11. Elija *NumProducto* y haga DOBLE CLIC.
12. Presione el botón **OK**.

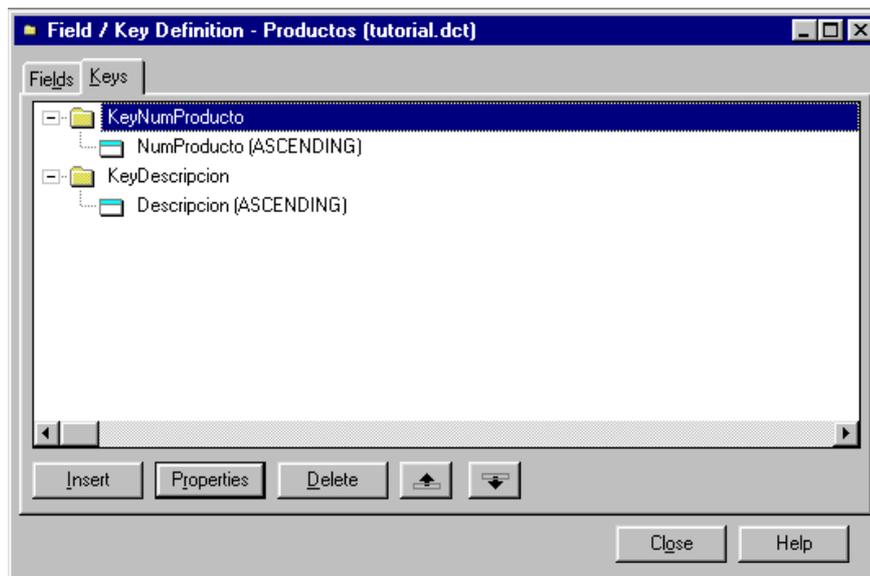
Aparece una caja de diálogo **Key Properties** en blanco, lista para la definición de otra clave.

Definición de una clave alfabética

Los usuarios probablemente querrán ver la lista de Productos en orden alfabético, de manera que añadiremos la clave correspondiente.

1. Escriba *KeyDescripcion* en el campo **Key Name**.
2. Elija la lengüeta **Fields**.
3. Oprima el botón **Insert**.
4. Elija Descripción, y haga DOBLE CLIC.
5. Oprima el botón **OK**.
6. Oprima el botón **Cancel** para cerrar la caja de diálogo **Key Properties** en blanco.

La caja de diálogo **Field / Key Definition** del archivo Productos se ve así:



7. Oprima el botón **Close** para cerrar la caja de diálogo **Field/Key Definition**.
8. Escoja **File** ➤ **Save**, u oprima el botón *Guardar* de la barra de herramientas.

Definición de relaciones entre archivos

Definición de las relaciones de "Pedidos"

Ahora que tenemos definidas todas las claves, podemos añadir relaciones. Una vez definidas éstas, se pueden añadir las Comprobaciones de Validez de los campos que sólo deben contener valores ya existentes en otro archivo. Estas son las últimas etapas para completar el diccionario de datos.

- ◆ `KeyNumPedido` vincula el archivo de `Pedidos` al archivo de `Detalle` en una relación de *uno-a-muchos*.
- ◆ `KeyNumCliente` vincula el archivo `Pedidos` al archivo de `Cientes` en una relación *muchos-a-uno*.

Definición de la primera relación

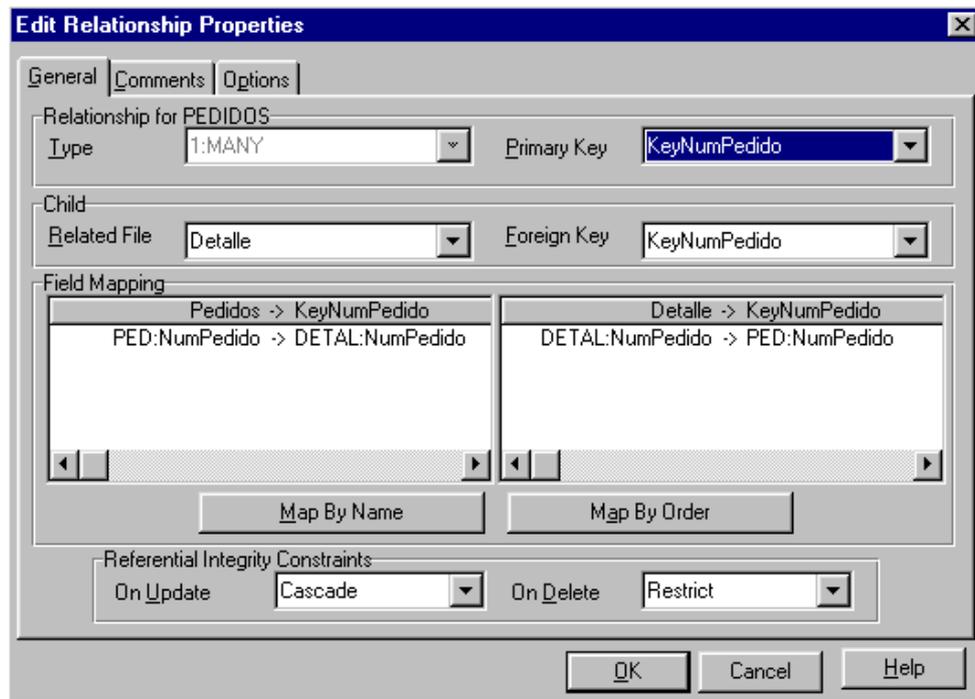
1. Destaque el archivo `Pedidos` de la lista **Files**.
2. Oprima el botón **Add Relation**.
El tipo preconfigurado es *1:MANY*, que es el correcto en este caso.
3. Elija `KeyNumPedido` de la lista desplegable **Primary Key**.
4. Escoja `Detalle` de la lista deaplegable **Related File**.
5. Elija `KeyNumPedido` de la lista desplegable **Foreign Key**.
6. Oprima el botón **Map by Name** ("Correlacionar por nombre").
Esto establece la relación vinculando todos los campos en las dos claves que comparten el mismo nombre.

Configuración de las restricciones de integridad referencial

1. Elija *Cascade* de la lista **On Update**.
Esto indica a los *templates* que deben generar código para actualizar automáticamente todos los registros "hijos" vinculados, cuando cambia el valor de la clave padre.
2. Elija *Restrict* de la lista **On Delete**.
Esto impide que el usuario borre un archivo "padre" que tenga archivos "hijos" vinculados.
3. Oprima el botón **OK**.

Definiendo la segunda relación

1. Destaque el archivo `Pedidos` de la lista **Files**.
2. Oprima el botón **Add Relation**.



3. Escoja *MANY:1* de la lista desplegable **Type**.

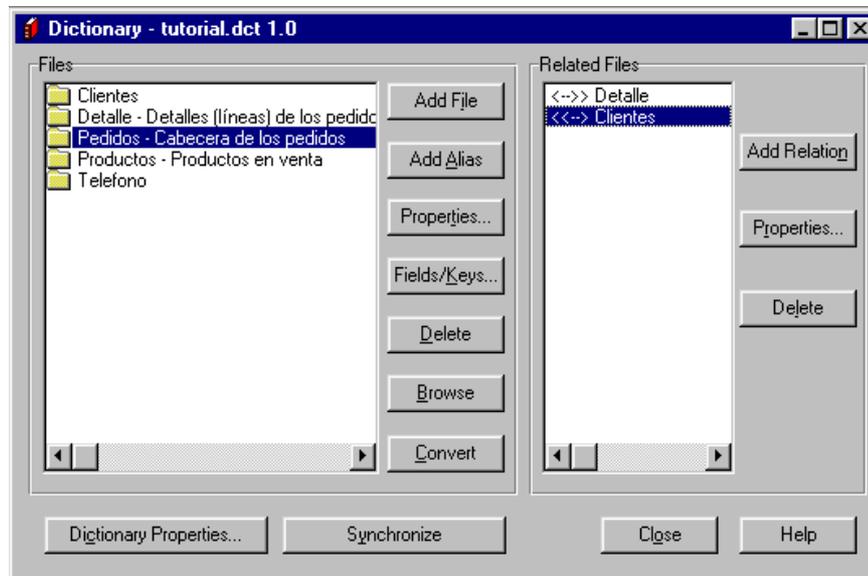
Advierta que cuando elige *MANY:1*, los indicadores de los campos **Primary Key** y **Foreign Key** intercambian lugares. Esto ocurre porque estamos definiendo las relaciones desde el punto de vista del archivo "hijo"; es el lado contrario de la relación que hicimos previamente. Una clave primaria siempre está en el archivo "padre", mientras que una clave foránea siempre pertenece al archivo hijo.

4. Elija *KeyNumCliente* de la lista desplegable **Foreign Key**.
5. Elija *Clientes* de la lista **Related File**.
Esto establece el archivo de Clientes como el "Padre" en esta relación
6. Escoja *KeyNumCliente* del listado desplegable **Primary Key**.
7. Oprima el botón **Map by Name**.

Configurar las restricciones de integridad referencial

1. Elija *Cascade* de la lista **On Update**.
Aunque estamos definiendo esta relación desde el lado del archivo "hijo", las restricciones de integridad referencial se mantienen sobre las acciones del archivo "padre".
2. Elija *Restrict* de la lista **On Delete**.
3. Oprima el botón **OK**.

La caja de diálogo del Diccionario debería verse así:



Elija **File** > **Save**, u oprima el botón *Guardar* de la barra de herramientas.

Definición de las relaciones del archivo Detalle

Cada vez que se define una relación en el Editor del Diccionario, es válida para ambos archivos a la vez. Por lo tanto, dado que ya ha definido todas las relaciones del archivo Pedidos, solamente queda una relación por definir.

Esta última relación pertenece al archivo Detalle.

- ◆ La clave *KeyNumPedido* vincula el archivo de Pedidos al de Detalle en una relación de *uno-a-muchos*. Esto ya se ha definido.
- ◆ La clave *KeyNumProducto* vincula el archivo de Detalle al archivo de Productos en una relación de *muchos-a-uno*.

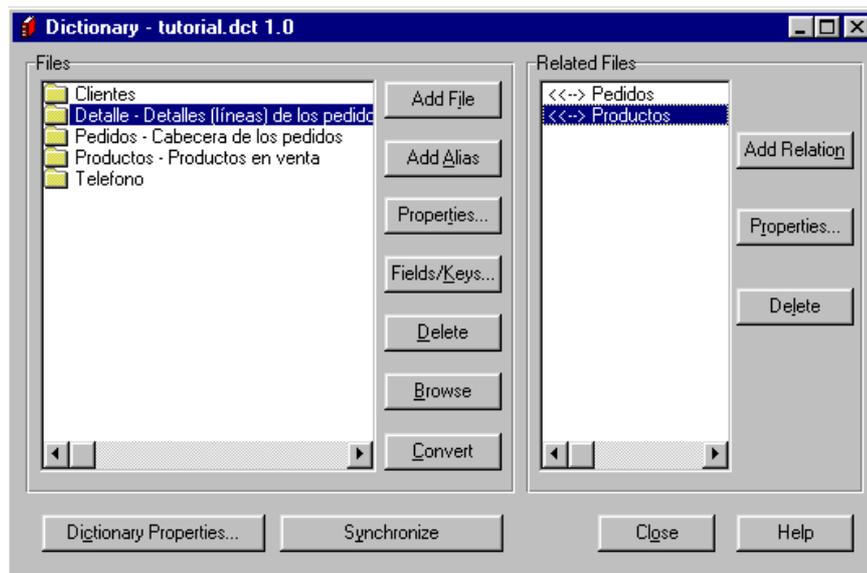
Definir la relación

1. Destaque el archivo *Detalle* de la lista **Files**.
2. Oprima el botón **Add Relation**.
3. Elija *MANY:1* de la lista desplegable **Type**.
4. Elija *KeyNumProducto* de la lista desplegable **Foreign Key**.
5. Escoja *Productos* de la lista deaplegable **Related File**.
6. Elija *KeyNumProducto* de la lista desplegable **Primary Key**.
7. Oprima el botón **Map by Name** ("Correlacionar por nombre").

Configuración de las restricciones de integridad referencial

1. Elija *Restrict* de la lista desplegable **On Update**.
Esto impedirá que se cambien los números de producto.
2. Elija *Restrict* de la lista desplegable **On Delete**.
3. Oprima el botón **OK**.

La caja de diálogo del diccionario ahora debería verse así:



4. Elija **File** ➤ **Save**, u oprima el botón *Guardar* de la barra de herramientas.

Definición de comprobaciones de validez dependientes de la relación

Ahora que hemos definido todas las relaciones entre archivos, podemos configurar las comprobaciones de validez de los dos campos que esperamos poner en las fichas o formularios.

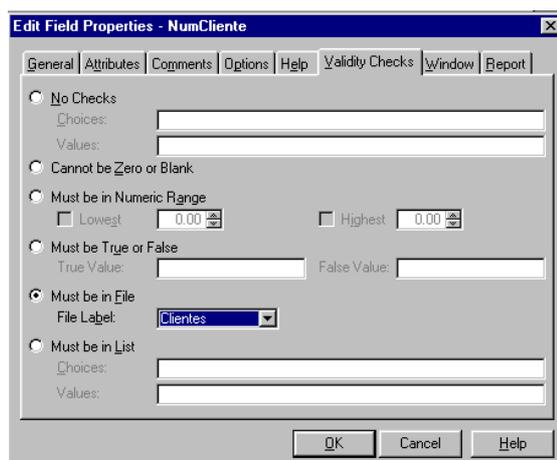
- ◆ Cuando se ingresa un nuevo registro de Pedidos, podemos especificar que el NumCliente corresponda a un registro ya existente en el archivo de Clientes.
- ◆ Cuando se entra un nuevo registro de detalle, podemos especificar que el NumProducto debe coincidir con un registro ya existente en el archivo de Productos.

Definición de la comprobación de validez de los registros de pedidos

1. Ilumine el archivo *Pedidos* en la lista **Files**.
2. Oprima el botón **Fields/Keys...**
3. Ilumine (destaque) *NumCliente* y oprima el botón de **Properties**.
4. Seleccione la lengüeta **Validity Checks**.
5. Seleccione el botón de radio **Must Be In File** ("Debe existir en el archivo...")
6. Elija *Clientes* de la lista desplegable **File Label**.

Esta acción requiere que el campo contenga solamente valores verificados, mediante la consulta al registro correspondiente en el archivo *Clientes*.

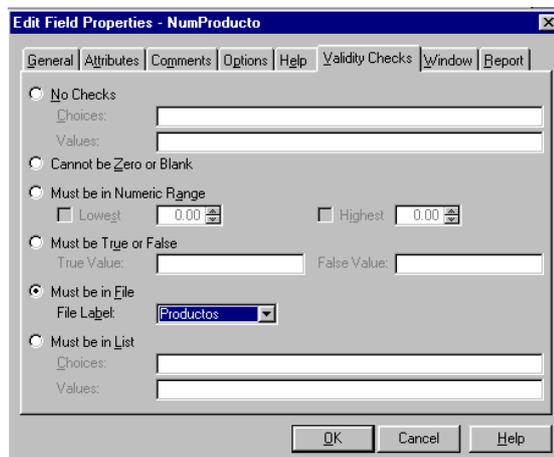
Como esto se valida utilizando la información sobre relaciones del archivo, no es posible configurar esta comprobación de validez hasta que hayan sido definidas las relaciones.



7. Oprima el botón **OK**.
8. Oprima el botón **Close** para cerrar la caja de diálogo **Field/Key Definition**.

Definición de comprobaciones de validez para los registros de Detalle

1. Destaque el archivo *Detalle* de la lista **Files**.
2. Oprima el botón **Field/Keys...** .
3. Destaque *NumProducto* y oprima el botón **Properties**.
4. Elija la lengüeta **Validity Checks**.
5. Elija el botón de radio **Must Be In File**.
6. Elija *Productos* de la lista **File Label**.



7. Oprima el botón **OK**.
8. Oprima el botón **Close** para cerrar la caja de diálogo **Field / Key Definition**.

El diccionario de datos ya está completo. En el siguiente capítulo, importaremos datos existentes desde otra aplicación, para mostrarle lo fácil que es de lograr.

5 - IMPORTACION DE DATOS PREEXISTENTES

Conversión de archivos de datos

Usted puede tener datos de aplicaciones heredadas que quiere conservar y usar en las aplicaciones Clarion . Para esto, este capítulo le enseñará a:

- ◆ Importar una definición de archivo a partir de datos preexistentes.
- ◆ Recorrer y modificar un archivo usando el Administrador de Base de Datos (" Database Manager")
- ◆ Cómo convertir datos de un formato de archivo a otro.

Punto de inicio:

El archivo TUTORIAL.DCT debe estar abierto.

Importación de una definición de archivo .CSV

Una forma sencilla de convertir archivos de datos es exportar los datos existentes de la aplicación anterior a archivos de valores separados por comas(.CSV). Se trata del formato de archivo utilizado originalmente por el lenguaje Basic, en que los datos se contienen entre comillas dobles, los campos están separados por comas y los registros se separan mediante un retorno de carro y avance de línea. El controlador Clarion de archivos BASIC lee y escribe estos archivos .CSV.

Importaremos la definición de un archivo .CSV existente que contiene datos de Clientes, luego generaremos un programa simple de conversión de datos (para demostrar lo fácil que es), que los colocará en un archivo TopSpeed.

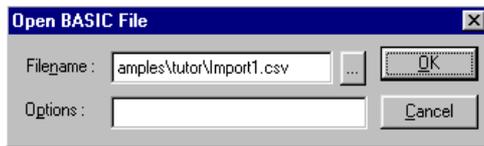
Importación de la definición de archivo

1. Elija **File** > **Import File**.



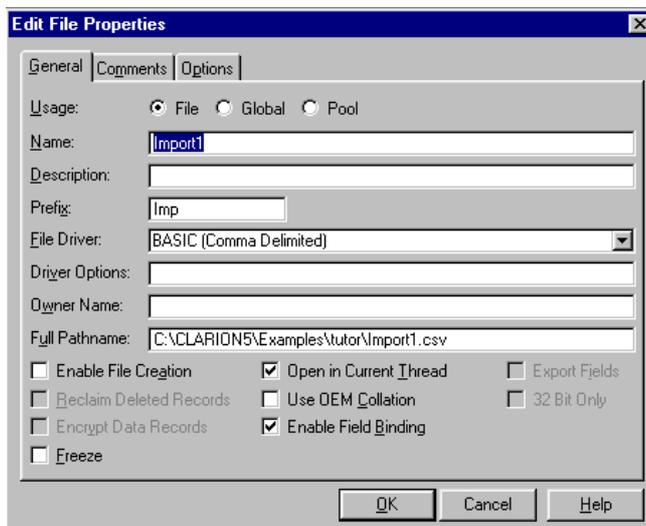
2. Seleccione **BASIC** de la lista desplegable y luego oprima el botón **OK**.
Aparece la caja de diálogo **Open BASIC File**.

3. En el campo **Filename**, escriba `c:\clarion5\examples\tutor\import1.csv` y luego oprima el botón **OK**.



Aparece la caja de diálogo **Edit File Properties**.

4. Oprima el botón **OK**.



Ahora tiene la definición del archivo IMPORT1.CSV. El siguiente paso será echar un vistazo a los datos en el Administrador de Bases de Datos.

Modificación de los datos

Dé CLIC DERECHO sobre el archivo IMPORT1.CSV, ilumine **Browse IMPORT1** y luego dé CLIC para llamar al Administrador de Bases de Datos.



El Administrador de Bases de Datos de Clarion le permite modificar en forma directa los datos de los archivos. Se trata de una herramienta para el programador, diseñada para permitirle hacer todos los cambios necesarios en los archivos. Esto significa que no hay salvaguardas contra violaciones de la integridad referencial o de la integridad de los datos. Por lo tanto, esta herramienta debe usarse con mucho cuidado.

CustNumber	Company
CustNumber	Company
1	Technology Today
2	Acme Fastners
3	A & A Exports
4	K Processing
5	Fidelity America
6	Easi Everything
7	Have Everything

Advierta que el primer registro contiene los nombres de campos, no datos. Este es el modo estándar de construir archivos .CSV. También, esos nombres de campo son exactamente los mismos que los nombres de campo de las definiciones del archivo *Cientes* (lo que facilitará mucho la conversión)

Conversión de un archivo de datos

En este punto, usted está observando a los datos del archivo .CSV en el Administrador de Bases de Datos Clarion. Lo siguiente es llevar esos datos a un archivo TopSpeed, para que los programas Clarion puedan usarlos.

Generación de un programa convertidor de archivos

1. Elija **File** ➤ **Convert File** (u oprima CTRL+V)
Aparece la caja de diálogo **File Convert**.
2. En el campo **Target Filename**, escriba *Cientes.TPS* como nombre del nuevo archivo (eliminando todo el texto preconfigurado que había en el campo).
3. Oprima el botón de tres puntos suspensivos (...) junto al campo **Target Structure**.
4. Destaque *Cientes*, y luego oprima el botón **Select**.

La caja de diálogo **File Convert** debería aparecer ahora así :

File Convert

Source Filename: C:\CLARION5\examples\tutor\Import1.csv

Source Dictionary: C:\CLARION5\TUTORIAL\TUT

Source Structure: Import1

Target Filename: C:\CLARION5\examples\tutor\Cientes.tps

Target Dictionary: C:\CLARION5\TUTORIAL\TUT

Target Structure: Cientes

Generated Source: C:\CLARION5\TUTORIAL\CO

Cuando oprima el botón **OK**, esto generará todo el código Clarion necesario para tomar los datos del archivo fuente (**Source Filename**), y copiarlos en un nuevo archivo de destino (**Target Filename**), utilizando el formato de archivo especificado en la estructura de destino (**Target Structure**).

La mejor razón para generar código fuente Clarion para la conversión de datos es darle la oportunidad de modificar el código antes de su compilación y ejecución, lo que da la oportunidad de manejar cualquier necesidad especial que uno pueda tener referente a la conversión. Esto hace completamente flexible el proceso de conversión para manejar cualquier situación que pudiera presentarse.

5. Oprima el botón **OK**.
Aparecerá un mensaje que anuncia la generación del código fuente.
6. Oprima el botón **OK** para regresar al Administrador de Bases de Datos.
7. Oprima el botón **Exit** para salir del Administrador de Base de Datos.

Borrado de la definición del archivo IMPORT1

El único fin de esta definición era permitir que el Administrador de Bases de Datos generara el código de conversión. Por lo tanto, podemos eliminarlo ahora del Diccionario de Datos.

1. Estando iluminado IMPORT1, oprima el botón **Delete**.
2. Oprima el botón **Yes** cuando se le pregunte si confirma el borrado.
3. Oprima el botón **Close** para salir del Editor del Diccionario de Datos, y oprima el botón **Yes** para guardar los cambios al salir.

Compilación y ejecución del programa convertidor

1. Elija **File** ➤ **Open** (u oprima CTRL+O).
2. Seleccione *Clarion source (*.clw)* de la lista desplegable **File of type**.
3. Seleccione la carpeta *C:\CLARION5\TUTORIAL*.
4. Destaque el archivo *CONVERT.CLW* y oprima el botón **Open**.

```

PROGRAM
MAP
  CheckError(),LONG
END

SourceName  STRING(260)
TargetName  STRING(260)
Count       ULONG

Import1     FILE, DRIVER('BASIC'), NAME(SourceName), PRE(IN)
RECORD
  CustNumber  STRING(32)
  Company     STRING(32)
  FirstName   STRING(32)
  LastName    STRING(32)
  Address     STRING(32)
  City        STRING(32)
  State       STRING(32)
  Zipcode     STRING(32)
END

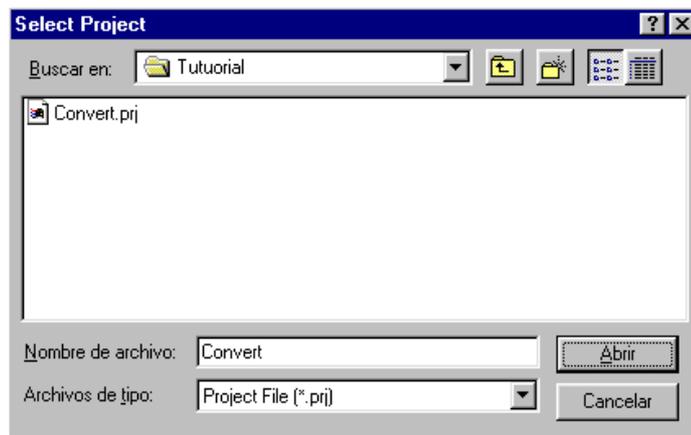
Clientes    FILE, DRIVER('TOPSPEED'), NAME(TargetName), CREATE, PRE(CLI)
KEY(+CLI:NumCliente), OPT, NOCASE
KEY(+CLI:Empresa), DUP, NOCASE
KEY(+CLI:CP), DUP, NOCASE
RECORD
  NumCliente  LONG
  Empresa     STRING(20)
  Nombres     STRING(20)
  Apellidos   STRING(20)

```

Aparece el Editor de Texto Clarion, con el archivo cargado para modificar.

El Administrador de Bases de Datos creó el código fuente de conversión en este archivo, que contiene todo lo necesario para leer los datos del archivo BASIC y copiarlo al archivo TopSpeed.

5. Elija **Project** ➤ **Set ...** .



Aparece la caja de diálogo **Select Project**.

6. Elija *Project file (*.prj)* de la lista desplegable **File of type**.
7. Seleccione el directorio *C:\CLARION5\TUTORIAL*.
8. Destaque el archivo *CONVERT.PRJ*, y luego oprima el botón **Open**.

El Administrador de Bases de Datos generó también el archivo CONVERT.PRJ al mismo tiempo que el archivo fuente CONVERT.CLW.

Cada programa Clarion tiene un Proyecto ("*Project*") que controla las opciones para la compilación del código fuente y el linkeado para producir el archivo .EXE resultante. En el caso de programas codificados a mano (y conversiones de archivos generadas por el Administrador de Bases de Datos), estas configuraciones se contienen en un archivo .PRJ. No hay necesidad de un .PRJ cuando se usa el Generador de Aplicaciones, dado que el archivo .APP mismo contiene todas las configuraciones del Proyecto.

En este punto, podríamos modificar el código fuente para realizar cualquier conversión de datos que se necesite (véase los temas en la ayuda en línea sobre: *How to Convert a File-Generate Source* y *How to Make a Field Assignment* para más información sobre cómo adaptar el código de conversión de datos). Sin embargo, no hay nada que debamos hacer en este proyecto, así que basta con compilar y ejecutar el programa.

9. Elija **Project** ➤ **Run** (u oprima CTRL+R)

Esto compila el programa, lo linkea a un .EXE, y corre el ejecutable resultante para realizar la conversión de archivos. Mientras corre el programa, aparece una ventana de estado, que indica el estado de la conversión. Dado que hay sólo unos pocos registros que convertir en este caso, desaparecerá muy pronto y probablemente no podrá verla.

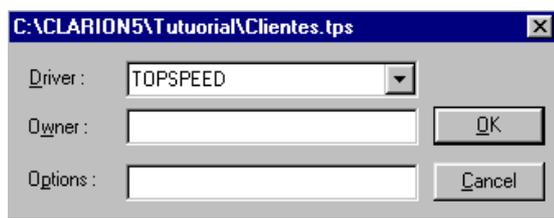
10. Elija **File** ➤ **Close** para cerrar el archivo y salir del Editor de Texto.

Prueba

Ahora puede comprobar los datos en el nuevo archivo, abriéndolo con el Administrador de Bases de Datos y desplazándose por los registros.

1. Elija **File** ➤ **Open** (u oprima CTRL+O)
2. Elija *Database* file de la lista desplegable **File of type**, ilumine el archivo *CLIENTES.TPS*, y oprima el botón **Open**.

Se abre una caja de diálogo que pregunta por el Controlador (*driver*) del archivo, la contraseña (password), y opciones necesarias para abrir el archivo.



3. Seleccione *TOPSPEED* de la lista desplegable **Driver**, y oprima el botón **OK**.

Esto demuestra otra forma de abrir el Administrador de Bases de Datos, distinta que desde el Editor del Diccionario.

Adviértase que el primer registro contiene los nombres de campo y no datos, así como ocurría en el archivo en el archivo *IMPORT1.CSV*. No se necesitan tales nombres en este archivo, de modo que puede borrar este registro inútil.

4. Con la barra deslizable ubicada sobre el primer registro, oprima **DELETE**, luego oprima el botón **Yes** cuando se le pregunte si confirma el borrado.
5. Oprima el botón **Yes** cuando se le pregunte si desea hacer un archivo de respaldo(*backup*).

El Administrador de Bases de Datos siempre pregunta si desea hacer una copia de seguridad antes de realizar cualquier tipo de modificación de los datos. Siempre es buena idea permitir que lo haga (¡por las dudas!).

6. Oprima el botón **Exit**.

Aparece un mensaje que pregunta si se desea guardar los cambios realizados en el archivo *CLIENTES.TPS*. Este diálogo ofrece una oportunidad adicional para deshacer los cambios realizados, si se da cuenta que fue un error efectuar modificaciones. *Yes* guarda los cambios y sale; *No* revierte el archivo al estado en que estaba cuando usted ingresó al Administrador de Bases de Datos y sale, y *Cancel* lo devuelve al Administrador.

7. Oprima el botón **Yes** para guardar los cambios realizados.

Ahora usted ha convertido algunos datos existentes importantes al formato TopSpeed para que las aplicaciones Clarion puedan usarlos. En el siguiente capítulo, comenzaremos a construir una aplicación "desde cero" usando el Generador de Aplicaciones.

6 - EMPEZANDO LA PROGRAMACIÓN

Uso del generador de Aplicaciones

Habiendo completado el Diccionario de Datos, ahora puede utilizarse el Generador de Aplicaciones para crear la aplicación. Este capítulo muestra cómo:

- ♦ Crear el archivo .APP, que guarda todo el trabajo del proyecto.
- ♦ Definir el primer procedimiento (Main) para crear un marco de aplicación MDI, y cómo llamar a los procedimientos desde el menú de la aplicación.

Punto Inicial:

Debe estar abierto solamente el entorno de desarrollo Clarion.

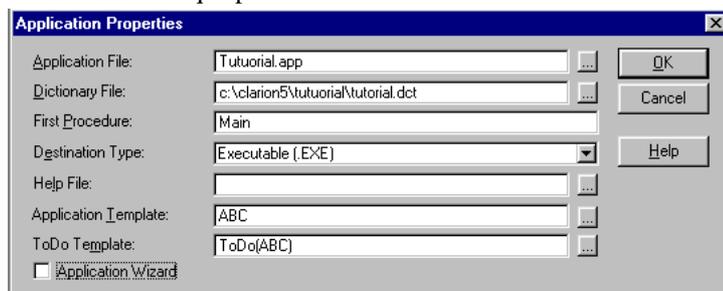
Creación del archivo .APP

1. Elija **File** > **New** > **Application**.
2. Seleccione *Application (*.app)* de la lista desplegable **Save as type**.
3. Seleccione el directorio C:\CLARION5\TUTORIAL.
4. Escriba *TUTORIAL* en el campo File Name.
5. Despeje el casillero Use Quick Start, y oprima el botón **Save**.

Aparece la caja de diálogo **Application Properties**.

6. Tipee *TUTORIAL.DCT* en el campo Dictionary.
7. Despeje el casillero **Application Wizard**.

Este cursillo del generador de Aplicaciones no utilizará los Wizards, para demostrar como usar las herramientas que provee Clarion



8. Oprima **OK** para cerrar la caja de diálogo **Application Properties**.

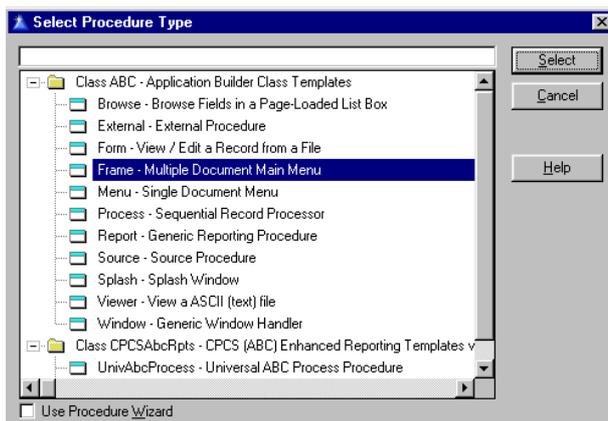
Creación del procedimiento principal (Main)

Aparece la caja de diálogo con el árbol de la aplicación (Application Tree). Este presenta un listado de todos los procedimientos de la aplicación en un árbol lógico que ofrece una guía visual que muestra el orden en que un procedimiento llama a otro. Lo vimos por primera vez en el Cursillo de Inicio Rápido.

El procedimiento *Main* es el punto de inicio. La aplicación del cursillo será un programa con interfaz de documentos múltiples (MDI). Por lo tanto, el lugar natural donde empezar es definir el procedimiento *Main* utilizando el template **Frame** para crear un marco de la aplicación.

Selección del tipo de procedimiento para Main

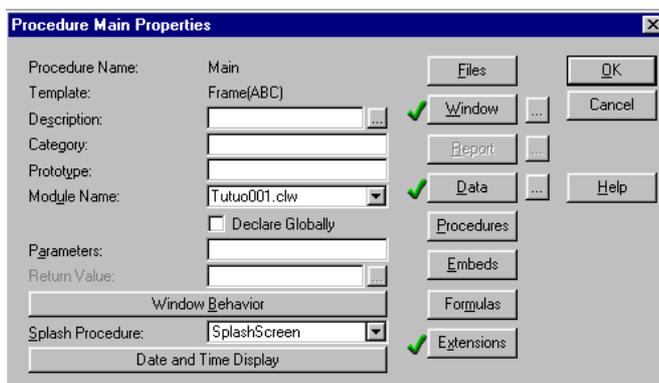
1. Teniendo a *Main* destacado en la caja de diálogo **Application Tree**, oprima el botón **Properties**.
2. Destaque *Frame* en la caja de diálogo **Select Procedure Type**, despeje el **casillero Use Procedure Wizard**, y oprima **Select**.



Aparece la caja de diálogo **Procedure Properties**. Define la funcionalidad y las estructuras de datos del procedimiento.

3. Escriba *SplashScreen* en el campo **Splash Procedure**.

Esto da nombre al procedimiento que contiene una pantalla de apertura que aparecerá brevemente mientras el usuario abre la aplicación.



Generalmente la primera tarea cuando se crea un procedimiento es diseñar la ventana principal. Se pueden colocar controles, o si el *template* ya tiene controles, puede personalizarlos.

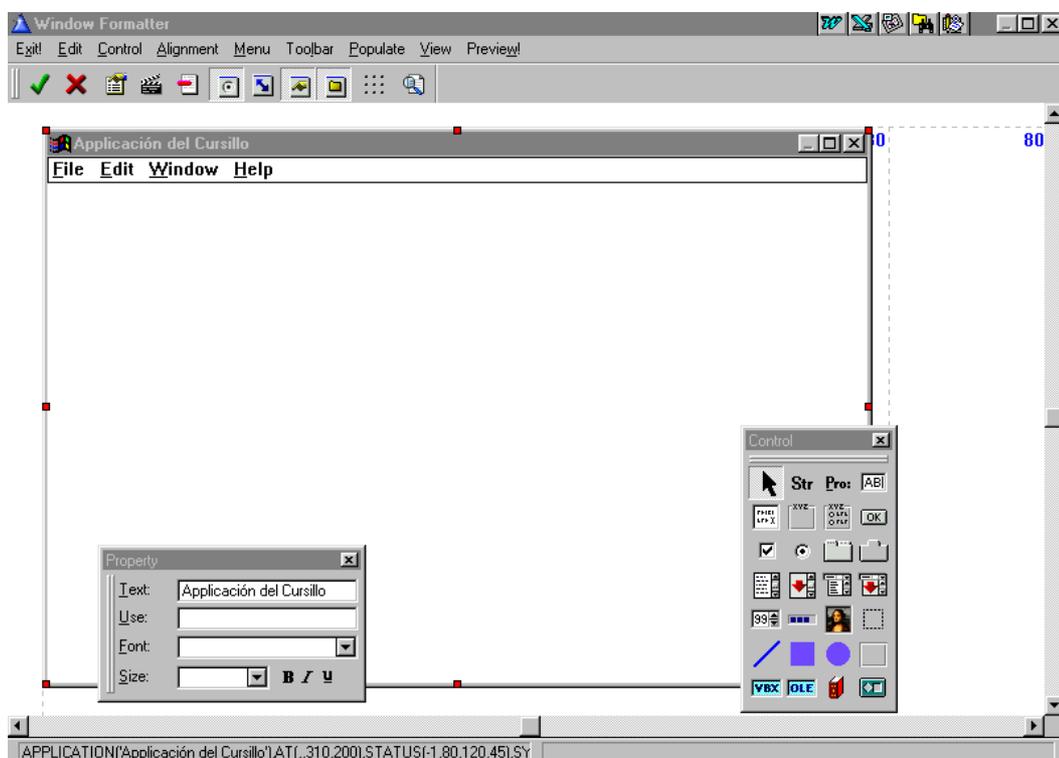
El marco de la aplicación *jamás* lleva controles sobre sí. Windows no lo permite. Sin embargo, personalizaremos el título de la ventana (el texto que aparece en la barra de título). Luego añadiremos ítems al menú predefinido, que también está preconfigurado en la plantilla del procedimiento *Frame*, y crearemos una barra de herramientas para la aplicación (una barra de herramientas si puede incluir controles).

Modificación de la ventana Main

1. Oprima el botón **Window**.

Aparece el Diseñador de Ventanas. Aquí están todas las herramientas necesarias para diseñar visualmente la ventana y sus controles.

2. Elija **View** > **Show Propertybox** para mostrar la barra de herramientas flotante **Property** (si no está ya presente).
3. Dé un CLIC sobre la ventana de título para darle foco, con lo que aparecerán asas (“manijitas” o *handles*) coloradas en los bordes de la ventana.
4. Escriba “*Aplicación del cursillo*” en el campo **Text** en la barra de herramientas flotante **Property**, luego oprima TAB.



Esto modifica el texto del título en la ventana de muestra. Asegúrese de que las asas estén dentro del marco de la aplicación de muestra al ejecutar este paso.

Edición del menú

Desde el menú del Diseñador de Ventanas, puede llamar al Editor de Menús, que permite añadir o modificar los ítems del menú de la ventana del marco de la aplicación. Al agregar cada ítem, se puede seleccionar la lengüeta TAB para nombrar el procedimiento a ejecutar cuando el usuario escoge ese ítem.

Por cada nuevo procedimiento que se añade al menú, el Generador de Aplicaciones agrega un procedimiento “*ToDo*” (“por hacer”) al árbol de la aplicación. Después puede definirse la funcionalidad de ese procedimiento, de la misma manera que ahora se está definiendo la funcionalidad del marco de la aplicación.

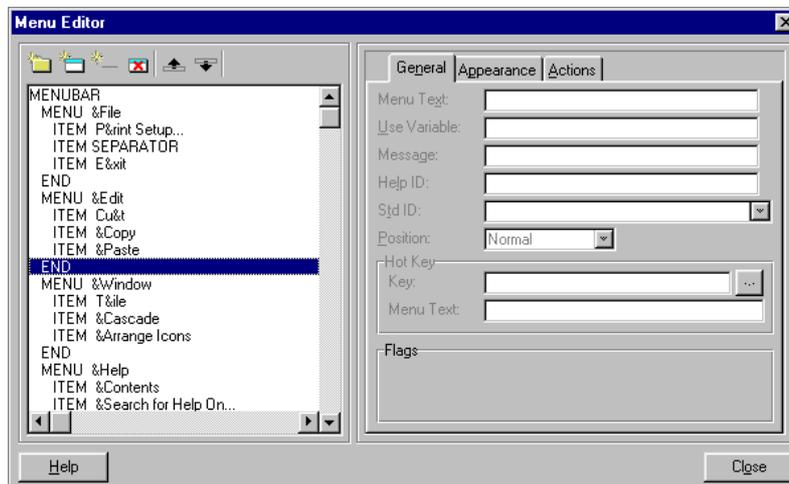
Cuando el Generador de Aplicaciones genera el código fuente para la aplicación, automáticamente abre un nuevo “hilo de ejecución” para cada procedimiento llamado desde el menú principal (lo que es necesario en toda aplicación MDI).

Agregado de ítems al menú

1. Desde el menú del Diseñador de Ventanas, elija **Menu** ➤ **Edit Menu** (o simplemente dé DOBLE CLIC sobre la barra de acciones del menú en el diseño de ventana).

Aparece el Editor de Menús. Muestra el menú en forma jerárquica en un listado a la izquierda. Los campos a la derecha permiten nombrar y personalizar los menús desplegados y sus ítems.

Esta plantilla viene ya provista con el menú “estandar”. Contiene órdenes básicas como **Exit** en el menú **File**, las órdenes comunes **Cut**, **Copy**, y **Paste**, y las órdenes de manejo básico de ventanas que se encuentran comúnmente en una aplicación MDI.



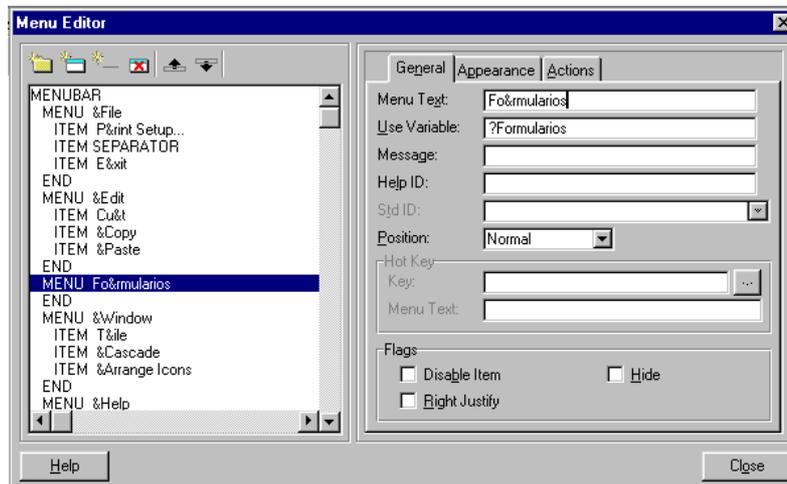
2. Ilumine la segunda sentencia END (véase la ilustración).

El Editor de Menú inserta nuevos ítem inmediatamente *debajo* de la selección iluminada. El menú a añadir se llamará **Formularios**. Contendrá tres ítems: **Productos**, **Clientes** y **Pedidos**. Aparecerá en la barra de menú justo antes del menú **Window**.

3. Oprima el botón **New Menu** del extremo superior izquierdo (o presione SHIFT+INSERT).

Esto inserta una nueva sentencia MENU, y su sentencia END correspondiente.

4. Escriba *Fo&rmularios* en el campo Menu Text y oprima TAB.



Esto define el texto que aparecerá en el menú. El ampersand (&) señala que el carácter siguiente (r) estará subrayado y proveerá un “atajo del teclado” (el usuario podrá oprimir ALT+R para activar este menú).

Agregar el primer ítem del menú

1. Oprima el botón **NewItem** (o presione INSERT).

Esto actualiza la lista a la izquierda de la caja de diálogo, cambiando el texto del menú que acaba de añadir a “Fo&rmularios”. Agrega un nuevo ITEM de menú -una orden en el menú desplegable- bajo Fo&rmularios, y antes de la sentencia END que va con el menú &Formularios.

2. Escriba *&Clientes* en el campo **Menu Text** y oprima TAB.

Aparece *?FormulariosClientes* en el campo **Use**. Esto es una “equivalencia” (*equate*) del ítem de menú, al que se referirán las sentencias del código fuente. El signo de interrogación antepuesto (?) indica que es un rótulo de campo equivalente (*field equate label*) (véase el libro *Language Reference*).

3. Elija la lengüeta **Actions**.

Los indicadores permiten nombrar un procedimiento a ejecutarse cuando el usuario seleccione el ítem del menú **Formularios** ➤ **Clientes**.

4. Elija **Call a Procedure** de la lista desplegable **When Pressed** (“Cuando se oprima”).

Aparecen nuevo indicadores para que nombre el procedimiento a llamar y otras opciones.

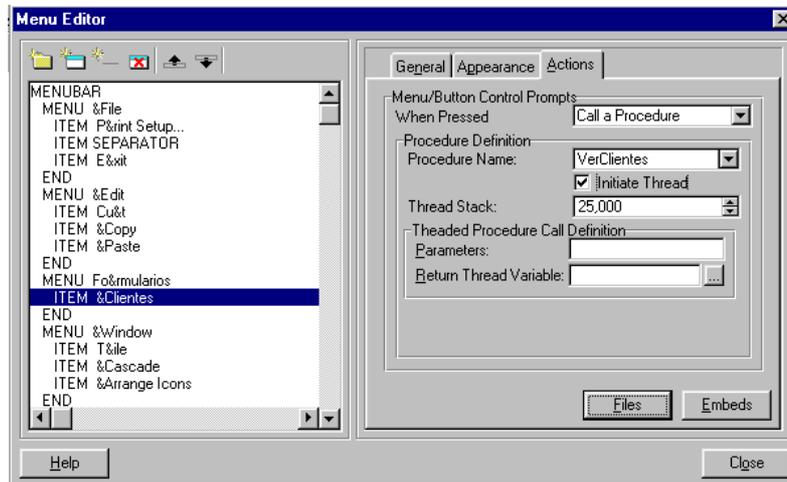
5. Escriba *VerClientes* en el campo **Procedure Name**.

Esto crea un procedimiento “ToDo” (por hacer) en el árbol de la aplicación.

6. Marque la casilla **Initiate Thread** (“Abrir un hilo de ejecución”).

El procedimiento *VerClientes* mostrará una ventana “hija” MDI, siempre debe comenzarse un nuevo hilo de ejecución para cada ventana MDI que se llame directamente desde el marco de la

aplicación. El campo **Thread Stack** (“Pila del hilo de ejecución”) muestra el valor mínimo recomendado.



Adición del segundo ítem del menú

1. Oprima el botón **NewItem**.

Esto actualiza la lista a la izquierda, cambiando el texto del texto que acaba de agregar a “&Clientes”.

2. Escriba *&Productos* en el campo **Menu Text** y oprima TAB.

Aparece *?FormulariosProductos* en el campo **Use**.

Normalmente, la siguiente etapa es definir la acción que realizará este ítem del menú (lo que se ejecutará cuando el usuario final escoge esta opción). Esta vez, saltaremos esta acción, solamente en este ítem. Más tarde, usted creará un procedimiento copiándolo, y luego adjuntándolo al menú, sólo para demostrar esta capacidad del entorno Clarion.

Adición del tercer ítem del menú

1. Oprima el botón **NewItem**.
2. Escriba *&Pedidos* en el campo **Menu Field** y oprima TAB.
Aparece *?FormulariosPedidos* en el campo **Use**.
3. Elija la lengüeta **Actions**.
4. Elija **Call a Procedure** desde la lista desplegable **When Pressed**.
5. Escriba *VerPedidos* en el campo **Procedure Name**.
6. Marque el casillero **Initiate Thread**.

Cierre del Editor de Menú y del Diseñador de Ventanas

1. Oprima el botón **Close** para cerrar el Editor de Menú.

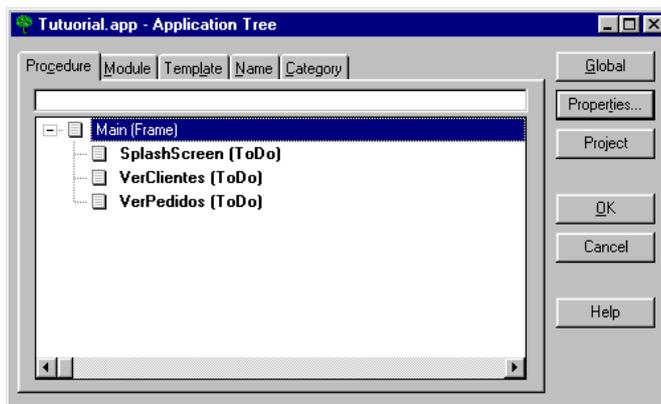
Esto lo regresa al Diseñador de Ventanas.

2. Elija la opción **Exit!** y responda **Yes** cuando se le pregunte si guardará los cambios a la ventana.

Esto lo regresa a la caja de diálogo **Procedure Properties**.

3. Oprima el botón **OK** para cerrar caja de diálogo **Procedure Properties**.

Esto lo regresa a la caja diálogo **Application Tree** (“Árbol de la Aplicación”). Hay ahora tres nuevos procedimientos marcador como “ToDo”: VerCliente, VerPedidos, y SplashScreen. Estos fueron los procedimientos que usted nombró en Editor de Menús.



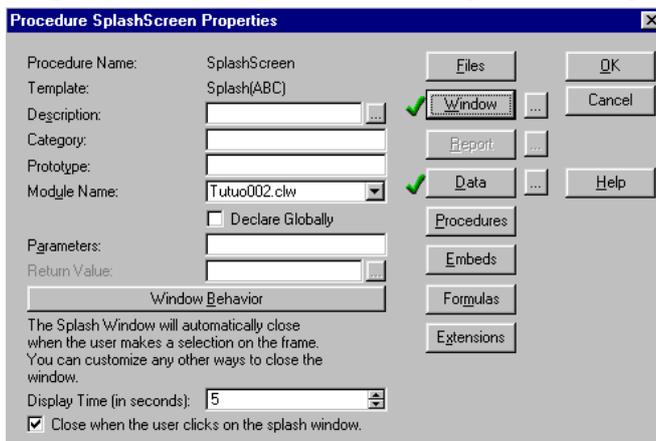
4. Elija **File** > **Save**, u oprima el botón Guardar en la barra de herramientas.

Creación del procedimiento SpalshScreen

Nombramos un procediemiento Splash (“pantalla de presentación”), y ahora lo crearemos.

1. Destaque el procedimiento *SpalshScreen* (ToDo) y oprima el botón **Properties**.
2. Seleccione *Splash* en la caja de diálogo **Select Procedure Type**, despeje el **casillero Use Procedure Wizard**, y oprima el botón **Select**.

Aparece la caja de diálogo **Procedure Properties**. No hay nada que tengamos que hacer con este procedimiento en el cursillo, salvo aceptar todas las opciones preconfiguradas.

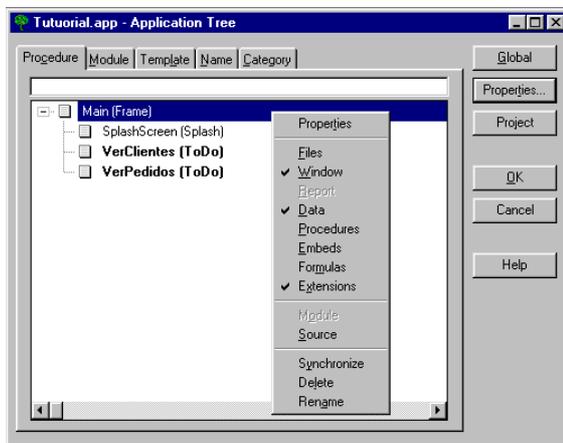


3. Oprima el botón **OK**.

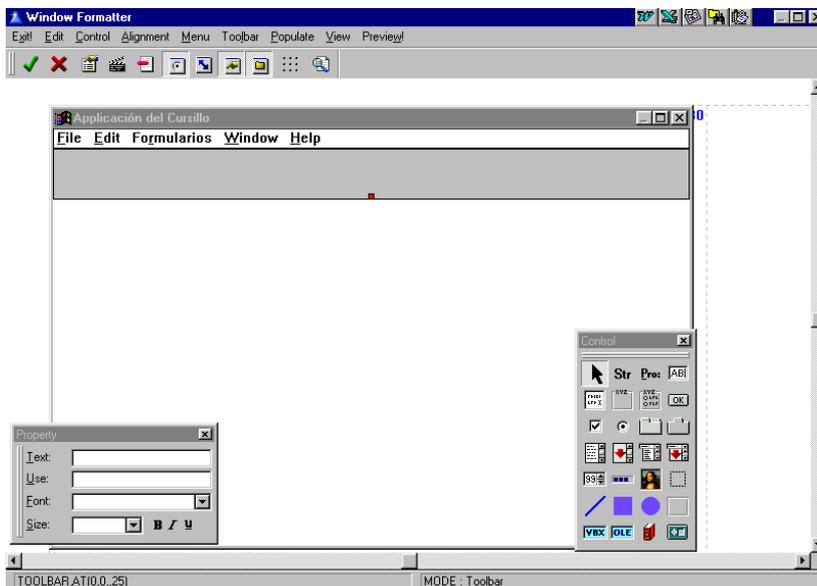
Adición de una barra de herramientas de la Aplicación

Llamar al Diseñador de Ventanas y crear la barra de herramientas

1. Destaque el procedimiento **Main**.
2. Dé CLIC DERECHO para activar el menú contextual.
Adviértase que este menú contextual tiene un grupo de opciones que equivale a la columna de botones ubicada a la derecha de cada ventana de **Procedure Properties**. Este menú contextual da acceso directo a todas las herramientas Clarion que se usan para modificar procedimientos existentes, para que no tenga que pasar por la ventana **Procedure Properties** cada vez.



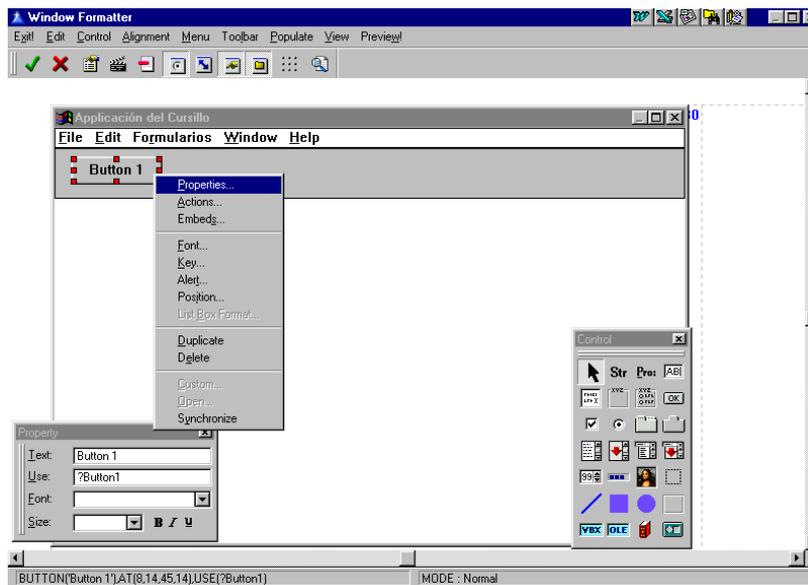
3. Elija **Window**.
4. En el menú del Diseñador de Ventanas, elija **Toolbar** ➤ **NewToolbar** (“Barra de herramientas ➤ Nueva”).
Esto añade la barra de herramientas, que siempre aparece inmediatamente debajo del menú, a la ventana de muestra. Puede añadir cualquier control a la barra de herramientas dando CLIC sobre un icono de herramienta en la caja flotante de herramientas **Controls**, y luego dando CLIC sobre su barra de herramientas.



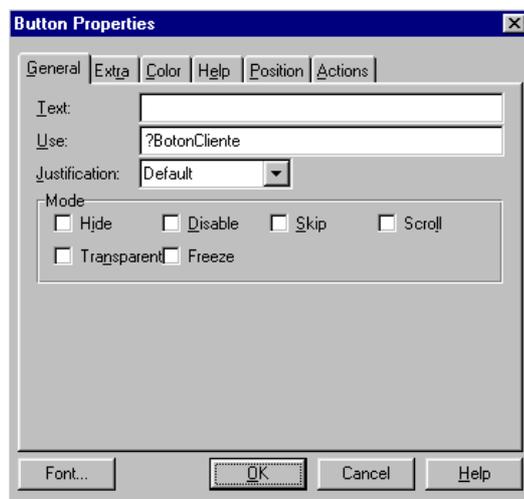
Colocar el primer botón

1. Dé CLIC sobre la herramienta de botón (el que se ve como botón “OK”).
2. Dé CLIC en el área de la barra de herramientas de la ventana, justo debajo del ángulo superior izquierdo.
3. Dé CLIC DERECHO sobre el botón que acaba de colocar, luego elija **Properties** del menú contextual.

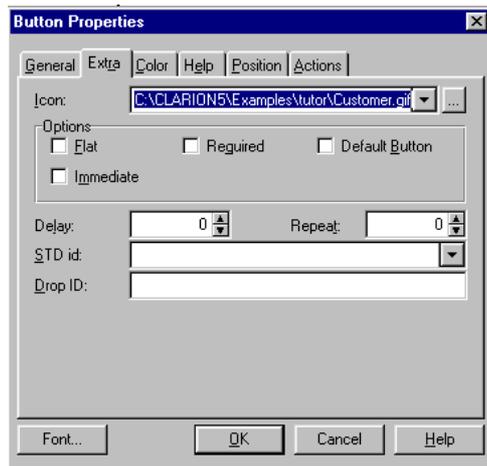
Aparece la caja de diálogo **Button Properties**.



4. Borre el contenido del campo **Text**.
Colocaremos imágenes en esos botones, en lugar de texto, para dar una apariencia más contemporánea al diseño.
5. Escriba *?BotonCliente* en el campo **Use**.
Este es el rótulo de campo equivalente para referenciar el botón en el código. Incluimos la palabra “botón” para la legibilidad del código.



6. Elija la lengüeta **Extra**.
7. Marque el casillero **Flat**.
8. Despliegue la lista **Icon**, desplácese hasta el fondo y dé CLIC sobre **Select File ...** Aparece la caja de diálogo **Select Image File**.



9. Elija el archivo `C:\CLARION5\EXAMPLES\TUTOR\CUSTOMER.GIF`, y oprima el botón **Open**.
10. Seleccione la lengüeta **Help**.
11. Escriba *Ver listado* de clientes en el campo **Tip**.
Esto añade una acotación, que aparecerá cada vez que el cursor del *mouse* se detenga sobre el botón.
12. Elija la lengüeta **Position**.
13. Seleccione los botones de radio fijos (**Fixed**) para tanto el largo(**Width**) como la altura (**Height**).
14. Configure el largo (**Width**) a 16 y la altura (**Height**) a 14.
15. Elija la lengüeta **Actions**.
16. Elija **Call a Procedure** de la caja desplegable **When Pressed**.
17. Elija **VerClientes** de la caja desplegable **Procedure Name**.
Este es el procedimiento que usted nombró en el ítem **Formularios** > **Clientes**. Oprimir el botón llamará al mismo procedimiento. A menudo, un botón de órdenes en una barra de herramientas sirve de atajo para ejecutar una opción del menú.
18. Marque el casillero **Initiate Thread**.
19. Oprima el botón **OK**.

Colocación del segundo botón

1. De CLIC sobre la herramienta “botón”.
2. De CLIC sobre el área de la barra de herramientas de la ventana en que estamos trabajando, justo a la derecha del primer botón. Aparece otro botón, etiquetado “Button2”.

3. De CLIC DERECHO sobre este botón, y elija **Properties** del menú contextual.
4. Borre el campo **Text**.
5. Escriba *?BotonProductos* en el campo **Use**.
6. Elija la lengüeta **Extra**.
7. Marque la casilla **Flat**.
8. Despliegue la lista de iconos (**Icon**), desplácese hasta el fondo y dé CLIC sobre **Select File...**
9. Seleccione el archivo de ejemplo *C:\CLARION5\EXAMPLES\TUTOR\PRODUCTS.GIF*, y luego oprima el botón **Open**.
10. Seleccione la lengüeta **Help**.
11. Escriba *Ver listado de productos* en el campo **Tip**.
12. Oprima el botón **OK** para cerrar la caja de diálogo **Button Properties**.

Normalmente, en este punto se adjunta una acción al botón. Por ahora, saltaremos esa acción. Más tarde, para demostrar el uso de los puntos de inserción de código, copiaremos un procedimiento, y lo llamaremos en el punto de código fuente que maneja qué debe hacerse cuando el usuario final oprime el botón.

Colocación del tercer botón

1. Dé CLIC sobre la herramienta “botón”.
2. Dé CLIC en el área de la barra de herramientas, al lado del segundo botón.
3. De CLIC DERECHO sobre el botón que acaba de ubicar, luego elija **Properties** del menú contextual.
4. Borre el contenido del campo **Text**.
5. Tipee *?BotonPedidos* en el campo **Use**.
6. Seleccione la lengüeta **Extra**.
7. Marque la casilla **Flat**.
8. Despliegue la lista **Icon**, desplácese hasta el fondo y dé CLIC sobre **Select File...** .
9. Elija el archivo *C:\CLARION5\EXAMPLES\TUTOR\ORDERS.GIF*, y oprima el botón **Open**.
10. Elija la lengüeta **Help**.
11. Tipee *Ver listado de pedidos* en el campo **Tip**.
12. Seleccione la lengüeta **Actions**.
13. Elija **Call a Procedure** de la lista desplegable **When Pressed**.
14. Elija *VerPedidos* de la lista desplegable **Procedure Name**.
Este es el nombre del procedimiento que escribió en el ítem de menú **Formularios Pedidos** ➤
15. Marque con una tilde el casillero **Initiate Thread**.
16. Oprima el botón **Ok**.

Cambiar el tamaño de los botones y alinearlos

El diseñador de Ventanas trae un juego de herramientas de alineamiento que facilitan el ajuste y alineamiento de los controles.

1. Con el botón *Pedidos* todavía seleccionado, de CTRL+CLIC en el botón *Clientes*.

Esto pone “manijitas” o “asas” alrededor de ambos botones, y el botón *Clientes* tiene asas rojas, lo que indica que posee el foco.

Dar CTRL+CLIC es la acción de “selección múltiple” que permite realizar acciones en varios controles a la vez. Una vez que se han seleccionado controles múltiples, también puede moverlos ARRASTRANDO cualquiera de los controles seccionados, o pueden usarse cualquiera de las herramientas de alineación (**Alignment**), que actúan sobre todo el grupo seleccionado.

2. Con ambos botones todavía seleccionados, de CTRL+CLIC sobre el botón *Productos*.

Ahora los tres botones tienen asas y el último botón seleccionado, *Productos*, las tiene rojas, lo que indica que posee el foco, y es el “control maestro” para el alineamiento.

3. Dé CLIC DERECHO y elija “Alineamiento superior” (**Align Top**) del menú contextual.

Cuando tiene seleccionados controles múltiples, un CLIC DERECHO muestra un menú contextual de alineamiento, en lugar del menú contextual del control aislado. Esta acción alineará los tres botones a la altura de *Productos*.

4. Dé CLIC DERECHO y elija “Espaciamiento horizontal” (**Spread Horizontally**) del menú contextual.

Esto distancia en forma proporcional los tres botones. Que no estén demasiado separados, ¡para dejar lugar para lo que viene!

Agregar los botones de control del Browse

1. Elija **Populate** ➤ **Control Template...** (o dé CLIC sobre la herramienta Control Template: la que se ubica en el ángulo inferior derecho de la caja flotante de herramientas **Controls**).

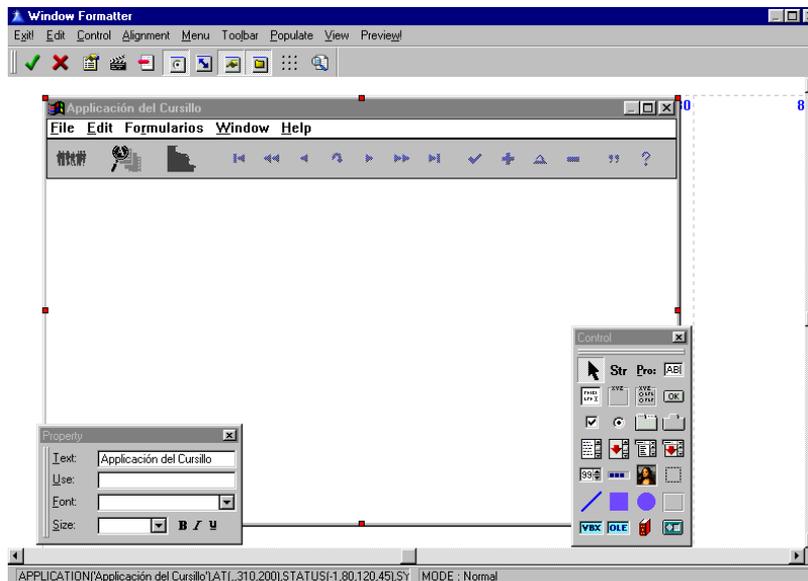
Aparece la caja de diálogo Select Control **Template**.

2. Ilumine el template *FrameBrowseControl*, y oprima el botón **Select**.

3. Dé CLIC en la barra de herramientas de la ventana, a la derecha del botón *Pedidos*.

Sobre la barra de herramientas aparecen los trece botones de control del *browse*. Ya conoce esos botones, por la aplicación que creó en la sección anterior. Estos botones controlan el desplazamiento del listado y la llamada al procedimiento de actualización que creará más adelante.

El diseño de pantalla deberá verse ahora muy parecido a esto:



Cierre el Diseñador de Ventanas y guarde el trabajo

1. Escoja la selección del menú **Exit!** y responda **Yes** cuando se le pregunte si desea guardar los cambios en la ventana.

Esto lo devuelve directamente al árbol de la aplicación, que contiene los mismos dos procedimientos, marcados como “por hacer” (ToDo): *VerClientes* y *VerPedidos*.

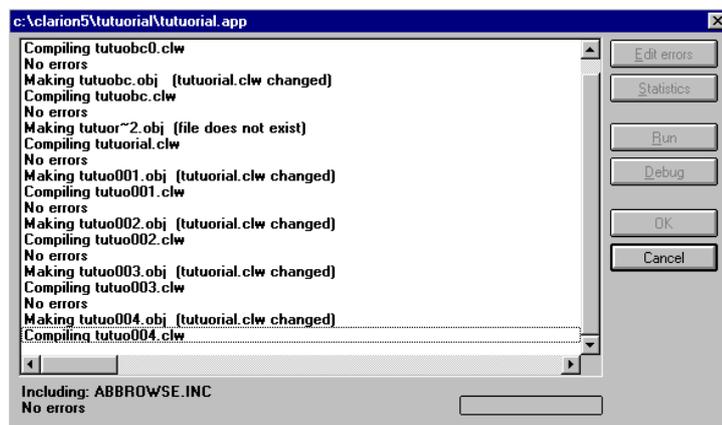
2. Elija **File** ➤ **Save**, u oprima el botón Guardar de la barra de herramientas.

Prueba de una aplicación en desarrollo

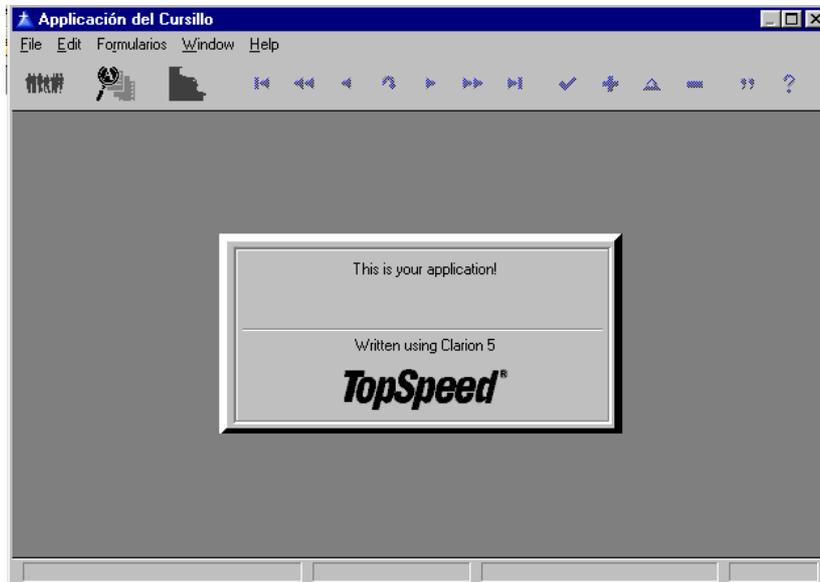
1. Con el árbol de la aplicación abierto, elija **Project** ➤ **Run**, u oprima el botón *Ejecutar* de la barra de tareas.

El Generador de Aplicaciones genera el código fuente, mostrando el avance del proceso en una ventana de mensaje, procedimiento por procedimiento.

A continuación aparece la ventana Make, mostrando el progreso del trabajo del compilador y linkeador.



Luego aparece la ventana inicial de la Aplicación, que debería ser aproximadamente así:

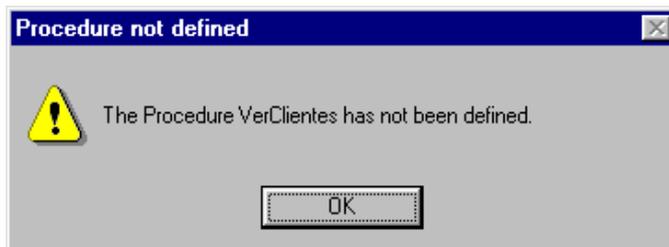


2. Oprima uno de los botones de la barra de herramientas, o escoja uno de los ítems del menú **Formularios**.

Aparece el siguiente mensaje:

Esta capacidad le permite probar la aplicación en modo progresivo, sea que haya diseñado todos los procedimientos o no.

En el próximo capítulo lo completaremos.



3. Oprima el botón **OK** para cerrar la caja de mensajes.
4. Elija **File** > **Exit** para cerrar la nueva aplicación.

Durante todo el resto de este cursillo, no tenga reparos en generar y ejecutar (*Make and Run*) la aplicación en desarrollo, en cualquier punto en que el cursillo le indique guardar el archivo.

Estudio del código fuente generado

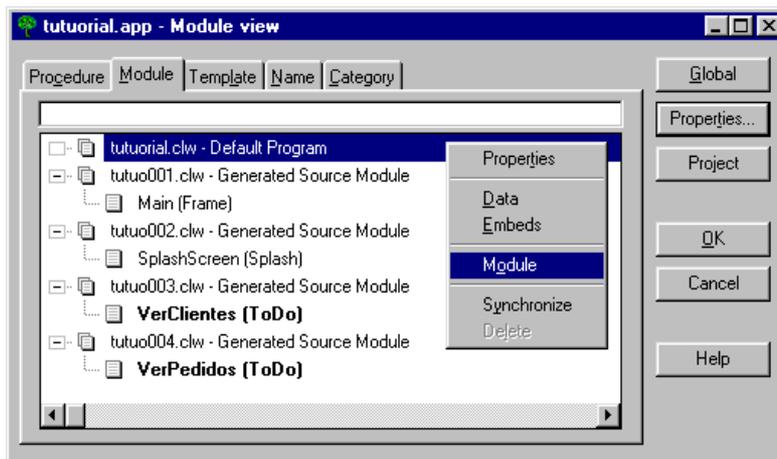
Veamos ahora lo que el Generador de Aplicaciones ha hecho. El propósito del generador de Aplicaciones (y sus Templates) es escribir el código fuente Clarion por usted. No hay “magia” en lo que hace este conjunto de herramientas para crear aplicaciones: todo se resume en el lenguaje de programación Clarion.

1. Teniendo abierto el árbol de la aplicación, dé CLIC sobre la lengüeta **Module**.

Esto cambia su visión de la aplicación del árbol que muestra la lógica con que se van llamando los procedimientos a los módulos de código fuente creados para la aplicación.

2. Ilumine el módulo *TUTORIAL.CLW*, dé CLIC DERECHO para mostrar el menú contextual y luego dé CLIC sobre **Module**.

Esto lo lleva directamente al Editor de Texto, para observar el último código fuente que generó (la última vez que oprimió el botón *Ejecutar*). Cualesquiera cambios que haya hecho desde la última vez que generó el código no serán visibles aquí.



El archivo TUTORIAL.CLW es el módulo de programa principal de esta aplicación, donde se contiene todo el código y las declaraciones de datos globales. No se deje asustar por todo este código. Una vez que haya terminado con el cursillo, puede consultar la “*Introducción al lenguaje Clarion*” al final de este libro y familiarizarse más con el lenguaje Clarion (que, de hecho, es muy simple).

```

PROGRAM
  _ABCD11Mode_ EQUATE(0)
  _ABCLinkMode_ EQUATE(1)

  INCLUDE('ABERROR.INC '),ONCE
  INCLUDE('ABFILE.INC '),ONCE
  INCLUDE('ABUTIL.INC '),ONCE
  INCLUDE('ABWINDOW.INC '),ONCE
  INCLUDE('EQUATES.CLW'),ONCE
  INCLUDE('ERRORS.CLW'),ONCE
  INCLUDE('KEYCODES.CLW'),ONCE

  MAP
    MODULE('TUTU0BC.CLW')
  DctInit PROCEDURE
  DctKill PROCEDURE
  END
  ?--- Application Global and Exported Procedure Definitions ---
  MODULE('TUTU001.CLW')
  Main PROCEDURE ?
  END
  END
  
```

The screenshot shows the Clarion 5 text editor window titled 'Clarion 5 [tutorial.app] - [C:\clarion5\tutorial\tutorial.clw]'. The menu bar includes File, Archive, Data Modeller, Edit, Search, Project, Setup, Window, and Help. The toolbar contains various icons for file operations and editing. The main text area displays the source code of the 'tutorial.clw' file, which is a program module. The code starts with 'PROGRAM' and includes several global declarations and include statements. It then defines a 'MAP' section with a 'MODULE' definition for 'TUTU0BC.CLW'. Below this, there are procedure definitions for 'DctInit' and 'DctKill', followed by an 'END' statement. A comment line indicates 'Application Global and Exported Procedure Definitions' with a dashed line. The code then defines a 'MODULE' for 'TUTU001.CLW' and a procedure 'Main' with a question mark as a parameter, followed by 'END' and 'END' statements. The status bar at the bottom shows 'Line: 23', 'Col: 7', and 'Insert' mode.

3. Cuando haya terminado de observar el código, elija **File** ➤ **Close** para cerrar el Editor de Texto y regresar al Generador de Aplicaciones.

Pero NO vaya a elegir **File** ➤ **Exit**, o saldrá completamente de Clarion for Windows.

4. Ahora dé CLIC sobre la lengüeta **Procedure**.

Hacerlo, cambia su vista de la aplicación en el árbol de procedimientos.

5. Destaque el procedimiento *Main (Frame)*, de CLIC DERECHO para activar el menú contextual, y dé CLIC sobre **Module**.

Esto lo lleva de nuevo al Editor de Texto, a observar el último código fuente que generó para el procedimiento *Main*. De nuevo, los cambios que haya realizado en el generador de Aplicaciones desde la última vez que generó código no aparecerán en este código.

Quizás advierta que justo debajo de la selección **Module** había otra llamada **Source**. No las confunda, ya que hacen cosas muy diferentes. Más adelante demostraremos la selección **Source**.

```

MEMBER('tutorial.clw')                                ? This is a MEMBER mod

INCLUDE('ABTOOLBA.INC'),ONCE
INCLUDE('ABWINDOW.INC'),ONCE

MAP
  INCLUDE('TUTU0001.INC'),ONCE                        ?Local module pro
  INCLUDE('TUTU0002.INC'),ONCE                        ?Req'd for module
  INCLUDE('TUTU0003.INC'),ONCE                        ?Req'd for module
  INCLUDE('TUTU0004.INC'),ONCE                        ?Req'd for module
END

Main PROCEDURE                                       ?Generated from proced

FilesOpened      BYTE
SplashProcedureThread LONG
AppFrame         APPLICATION('Aplicación del Cursillo'),AT(,310,190),S
                MENUBAR
                MENU('&File'),USE(?FileMenu)

```

Si hace cambios en el código, puede compilar y ejecutar el programa para ver que efectos causan los cambios. Sin embargo, **los cambios realizados aquí se perderán la próxima vez que genere el código fuente**. Por lo tanto, no es buena práctica realizar cambios aquí.

Esta capacidad de ver código fuente es muy útil para estudiar código generado para determinar el contexto dentro del cual aparecerá el código que desee añadir en los puntos de inserción (más adelante trataremos esto).

6. Cuando haya terminado de observar el código, elija **File** ➤ **Close** para cerrar el Editor de Texto y regresar al Generador de Aplicaciones.

7 - CREACIÓN DE UN BROWSE

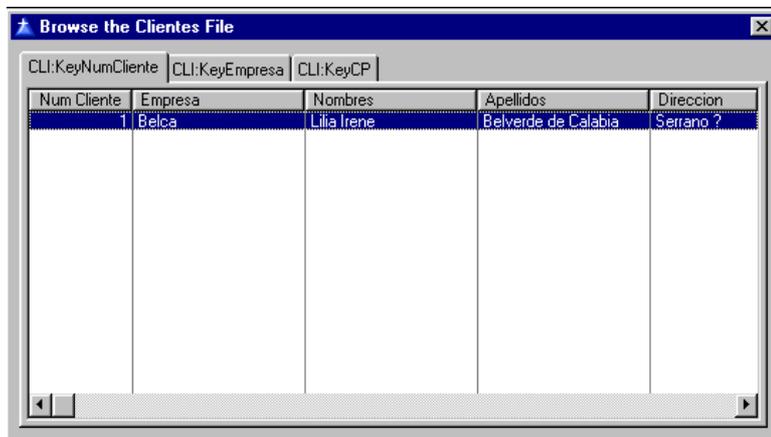
Creación una ventana Browse

En este capítulo, crearemos una ventana de visualización de listados(browse) parecida a la que creó para usted el Asistente para Inicio Rápido. El Generador de Aplicaciones usa las mismas plantillas, que generan el mismo código fuente básico; pero haciendo así tendrá la oportunidad de "hacerlo desde cero". Esto demuestra todo lo que los Asistentes pueden hacer por usted, y cuánto puede hacer usted también. Empezaremos la ventana *browse* de Clientes.

Punto de inicio:
El archivo TUTORIAL.APP debe estar abierto.

Creación de la visualización del listado de clientes

Usted recordará que el Asistente para Inicio Rápido creó una ventana para el procedimiento *browse* de Clientes, que se veía así:



Ahora, crearemos una similar utilizando el template de procedimiento *Browse*.

Selección del tipo de procedimiento para VerClientes

1. Dé DOBLE CLIC sobre *VerClientes* en el árbol de la aplicación.
2. Destaque (ilumine) la plantilla *Browse* en la caja de diálogo **Select Procedure Type**, verifique que la casilla **Use Procedure Wizard** no esté marcada y oprima el botón **Select**.

Se abre la caja de diálogo **Procedure Properties**.

Hacer ajustable el tamaño de la ventana

1. En la caja de diálogo **Procedure Properties**, oprima el botón **Extensions**.
2. En la caja de diálogo **Extensions and Control Templates**, oprima el botón **Insert**.
3. Ilumine *WindowResize* en la caja de diálogo **Select Extension**, y oprima el botón **Select**.
Esta Plantilla de Extensión genera código para manejar automáticamente el ajuste de tamaño y reubicación de todos los controles de la ventana cuando el usuario cambia el tamaño de la ventana, o cuando oprime los botones de Maximizar o Restaurar.
4. Oprima el botón **OK** para cerrar la caja de diálogo **Extension and Control Template**.

Modificación del procedimiento Browse

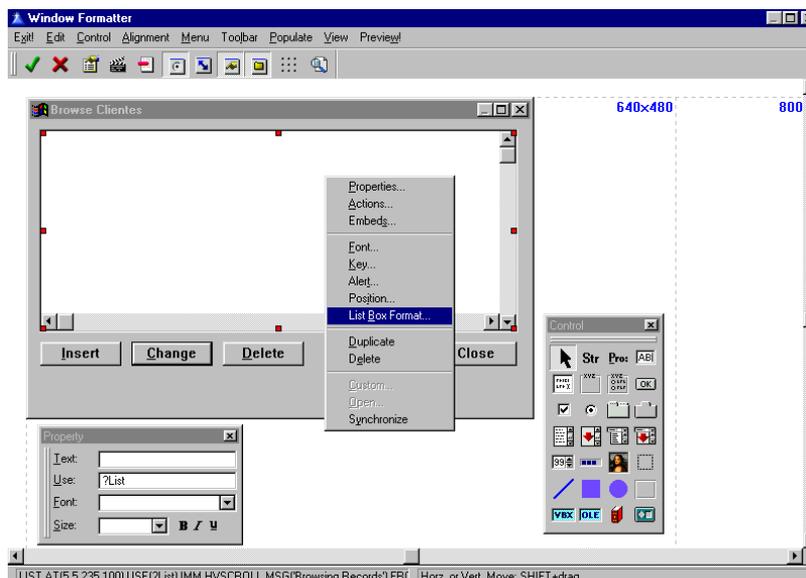
1. En la caja de diálogo **Procedure Properties**, oprima el botón **Window**.
2. Dé CLIC DERECHO sobre la ventana de título de la ventana, y elija **Properties** del menú contextual.
3. Tipee *Browse Clientes* en el campo Text.
4. Elija *Resizable* de la lista desplegable **Frame Type**.
5. Seleccione la lengüeta **Extra** y marque el casillero **Maximize Box**.
Estos dos últimos pasos le permiten al usuario cambiar el tamaño de la ventana durante la ejecución.
6. Oprima el botón **OK** para cerrar la caja de diálogo **Window Properties**.

Llenado y diseño de una caja de listado

Utilizando el Diseñador de Cajas de Listado, puede *llenar y dar formato* a los campos del diccionario de datos que aparecen en las columnas de la lista.

Prepararse a dar formato a la ventana de listado

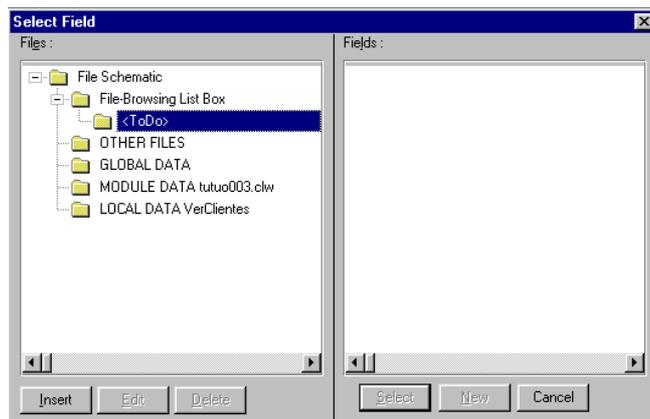
Dé CLIC DERECHO sobre la caja de listados en la ventana, y elija **List Box Format...** del menú contextual.



Aparece la caja de diálogo **Select Field**, que da acceso a los archivos definidos en el diccionario. El listado **Files** muestra todos los archivos seleccionados para usar en este procedimiento en un orden jerárquico, o árbol de archivos (*File Schematic*) que incluye el control de cajas de listado.

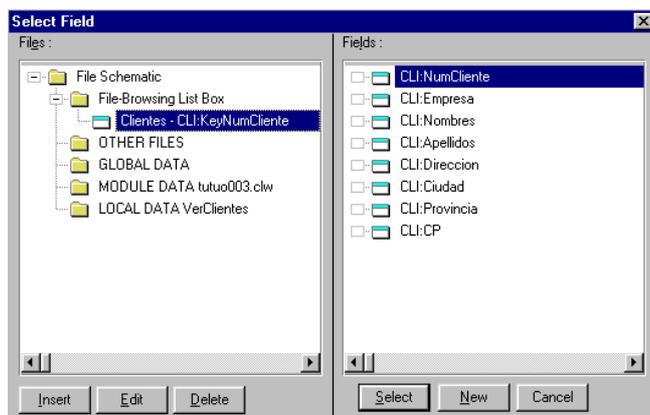
Elija el archivo y los campos para colocar en el listado

1. Ilumine el ítem "ToDo" debajo de **FileBrowsing List Box** y oprima el botón **Insert**.

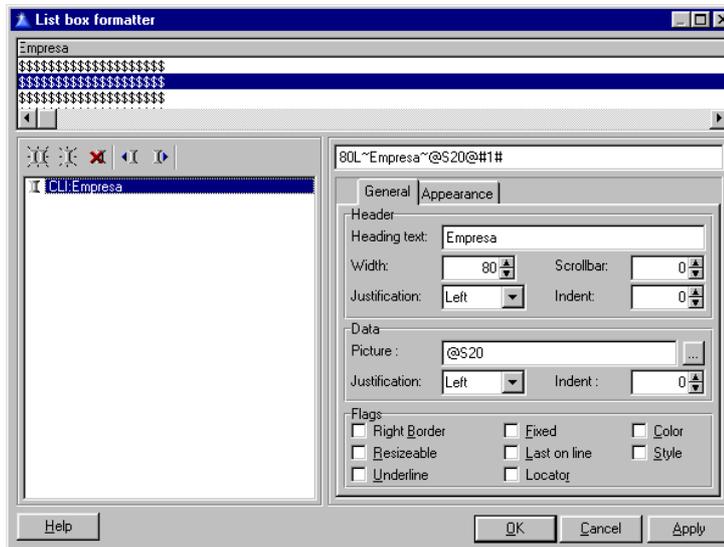


2. Destaque el archivo *Cientes* en la caja de diálogo **Insert File**, y oprima el botón **Select**.
Esto añade el archivo al árbol en la caja de diálogo **Select Field**, que ahora muestra el archivo y sus campos.
3. Oprima el botón **Edit**.
4. Destaque *KeyNumCliente* en la caja de diálogo **Change Access Key** y oprima el botón **Select**.

Esto es importante, porque indica el orden de visualización de los registros. Si no se especifica una clave, los registros aparecen en el orden en que fueron incorporados al archivo (también llamado "orden de los registros")

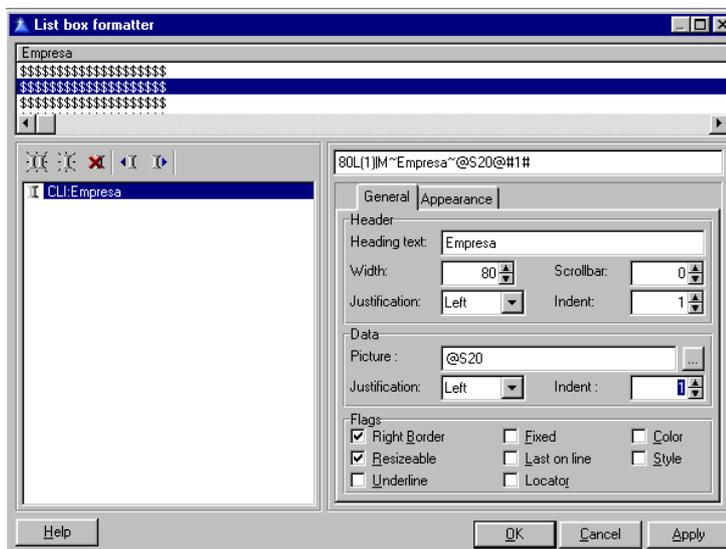


5. Ilumine CLI: Empresa en la lista **Fields**, y oprima el botón **Select**. Esto lo regresa al Diseñador de Cajas de Listado, con el campo seleccionado añadido a la lista. La lengüeta de la derecha le permiten diseñar la apariencia del campo.



Aplicación de un formato especial al primer campo

1. Elija la lengüeta **General** marque los casilleros de "Borde Derecho" (**Right Border**) y "Permitir cambio de tamaño" (**Resizable**). En tiempo de ejecución, esto añadirá un borde vertical reconfigurable a la derecha del campo.
2. Dé un CLIC sobre las flechas hacia arriba de las cajas de "Sangría" (**Indent**) para dar una ligera sangría al texto de la cabecera y al contenido del campo.



Llenado del segundo campo

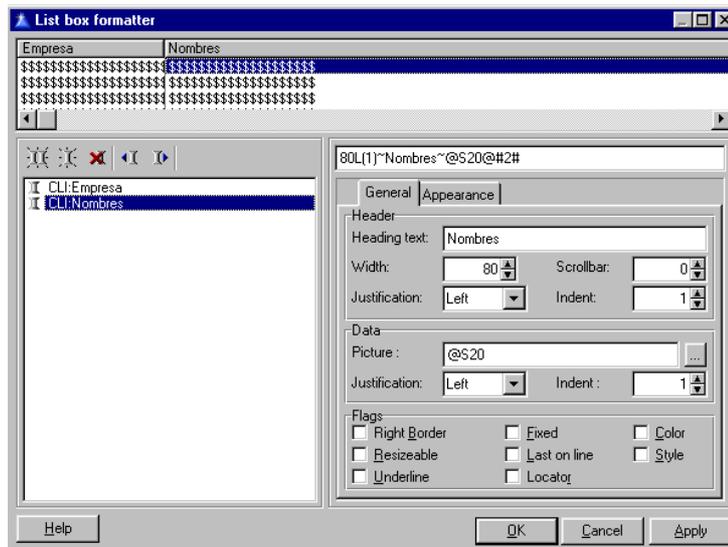
1. Oprima el botón **New Column** (Insertar columna nueva).



New Column (Insert)

2. Destaque *CLI:Nombres* en la lista **Fields**, y oprima el botón **Select**.
3. Despeje los casilleros de "Borde Derecho" (**Right Border**) y "Permitir cambio de tamaño" (**Resizable**).

El Diseñador de Cajas de Listado automáticamente "arrastra" esas opciones del último campo añadido a los campos que se sigan añadiendo, lo que hace muy fácil incorporar campos múltiples con similares opciones de formato. En este caso, el borrar las marcas de los casilleros borra el divisor de columnas entre esta columna y la siguiente, que será el campo *Apellidos*.



Llenado del tercer campo

1. Oprima el botón de **New Column**.
2. Ilumine *CLI:Apellidos* en la lista de campos (**Fields**), y oprima el botón **Select**.
3. Marque los casilleros **Right Border** y **Resizable**.

Esto nuevamente añade el divisor de columnas ajustable entre esta columna y la próxima.

Agrupamiento de algunos campos

1. Presione el botón **Nuevo Grupo** (Agrega un Grupo).
2. Presione UP ARROW para iluminar el nuevo grupo.

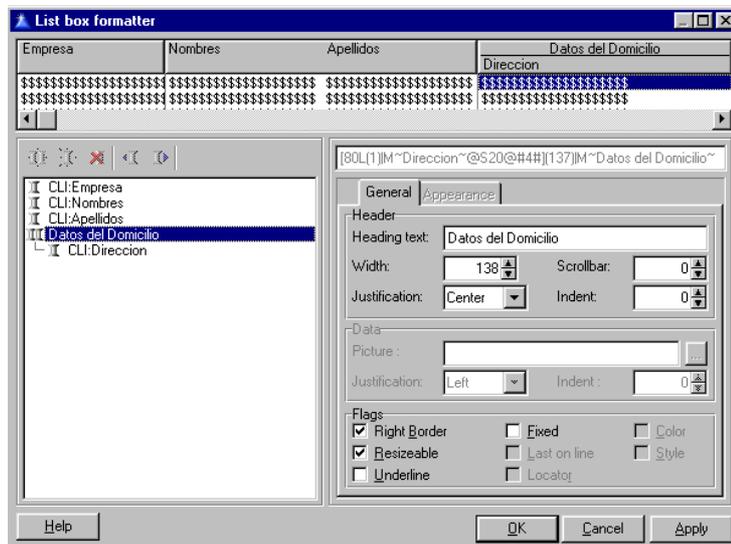


New Group (Shift+Insert)

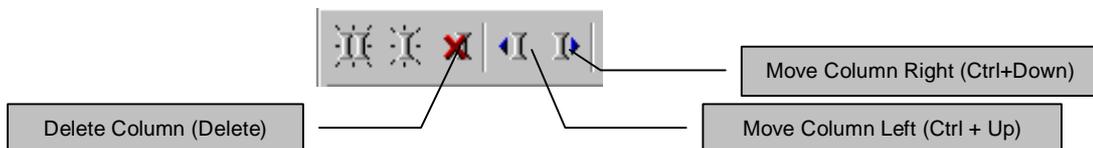
Esto le permitirá agregar una cabecera de grupo para la información referida al domicilio. Esta última aparece "sobre" las cabeceras de los campos, y visualmente une los datos contenidos en las columnas que abarca.

3. Tipee Datos del Domicilio en el campo encabezado (**Heading Text**).

Esto proporciona el texto a la cabecera de grupo. Todo campo que se agregue a la derecha quedará incluido en el grupo, hasta que se defina un grupo nuevo.



Note que los efectos que se producen por agregar campos y efectuar modificaciones se reflejan en la caja de listado de muestra que aparece en la parte superior del Diseñador de Cajas de Listado. Además puede emplear los tres botones ubicados a la derecha del de *New Column* para eliminar la columna correspondiente al campo iluminado (**Delete Column**), mover la columna a la izquierda (**Move Column Left**) o mover la columna a la derecha (**Move Column Right**), respectivamente.



4. Ilumine *CLI:Direccion* en el Diseñador de Cajas de Listado.

Llenado del quinto campo

1. Oprima el botón **New Column**.
2. Ilumine CLI: *Ciudad* en la lista **Fields**, y oprima el botón **Select**.

Llenado del sexto campo

1. Oprima el botón **New Column**.
2. Destaque *CLI:Provincia* en el listado de **Fields**, y oprima el botón **Select**.

Llenado del séptimo campo y salida del Diseñador

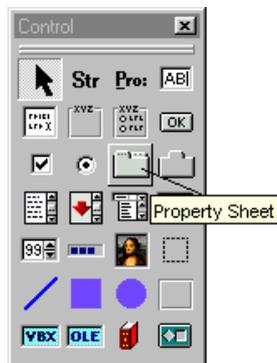
1. Oprima el **Populate**.
2. Destaque *CLI:CP* en el listado de **Fields**, y oprima el botón **Select**.
3. Oprima el botón **OK** para cerrar el Diseñador de Cajas de Listado.

Adición de las lengüetas

Cuando el Quick Start Wizard creó este procedimiento, tenía controles de lengüetas que cambiaban el orden de visualización de la lista según la lengüeta seleccionada. Por lo tanto, añadiremos esta funcionalidad para mostrar lo fácil que es hacerlo.

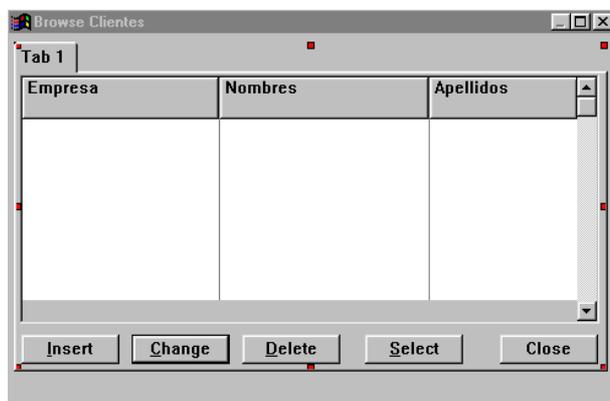
Añadir la Hoja de Propiedades y la primera lengüeta

1. Dé CLIC sobre la ventana de título para que aparezcan las asas rojas en los bordes de la ventana.
2. Coloque el cursor del *mouse* directamente sobre el asa central de la parte superior, y ARRASTRE para aumentar el espacio en la parte de arriba.



3. Dé CLIC sobre el control de propiedades de la hoja (*Property Sheet*) en la caja flotante de herramientas **Controls** (la que parece varias carpetas de archivo y está a la derecha del círculo).
4. Dé un CLIC sobre y a la derecha de la caja de listado para colocar la hoja de propiedades y un control de lengüeta .
5. ARRASTRE el "asa roja" del ángulo inferior *izquierdo* de manera que aparezca justo abajo y a la izquierda del botón Insert.
6. ARRASTRE el "asa roja" del ángulo inferior *derecho* de manera que aparezca justo abajo y a la izquierda del botón Close.

Esto modifica el tamaño de la hoja de propiedades de modo que *parezca* que la caja de listado y los botones están sobre la lengüeta. De hecho, no lo están, y no queremos que estén, dado que deseamos que estos controles sean visibles, sin importar la lengüeta que el usuario haya seleccionado. La ventana se veía ahora más o menos así:



7. Dé CLIC sobre el texto "Tab 1", en el campo **Text** de la caja flotante **Property**.
8. Escriba *KeyNumCliente* en el campo **Text** en la caja flotante de herramientas **Property** y pulse TAB.

Esto cambia el texto de la lengüeta. El texto es libre, pero si se pone el nombre de la clave, también indica el orden de visualización .

Adición de las demás lengüetas

1. Dé CLIC sobre el *Tab control* en la caja flotante de herramientas **Controls** (es la que se parece a una sola carpeta).



2. Dé CLIC inmediatamente a la derecha *KeyNumCliente* para colocar la nueva lengüeta .
3. Escriba *KeyEmpresa* en el campo **Text** de la caja flotante de herramientas **Property**, y oprima TAB.
4. Dé CLIC sobre el *Tab control* en la caja flotante de herramientas **Controls**.
5. Dé CLIC inmediatamente a la derecha de la lengüeta *KeyEmpresa* para colocar la siguiente lengüeta.
6. Escriba *KeyCP* en el campo **Text** de la caja flotante de herramientas **Property**, y oprima TAB.

Esconder los botones

Cuando el Asistente de Inicio Rápido creó este procedimiento no tenía botones de Agregar, Cambiar, Borrar y Cerrar; o al menos, ¡usted no pudo verlos al ejecutar el programa! En realidad, los botones están allí, pero están escondidos para que el usuario utilice solamente los botones de la barra de herramientas para modificar el archivo.

El "secreto" es que los botones de la barra de herramientas simplemente ordenaban a los botones escondidos del procedimiento *Browse* que hagan lo que siempre hacen. Por lo tanto, cuando usted diseña un procedimiento *Browse* sin utilizar los Asistentes, necesita tener los botones de actualización sobre la pantalla; pero no es necesario que el usuario los vea el ejecutar el programa.

1. Dé CLIC DERECHO sobre el botón Close de la ventana de muestra, y luego dé CLIC sobre **Properties...** en el menú contextual.

2. Marque el casillero de "Esconder" (**Hide**), y oprima el botón **OK**.
Esto añade el atributo HIDE al control, de manera que no sea visible en pantalla durante la ejecución. Claro que siempre puede vérselo en el Diseñador de Ventanas.
3. Dé CLIC DERECHO sobre el botón *Select*, y luego dé CLIC sobre **Properties...** .
4. Marque el casillero **Hide**, y oprima el botón **OK**.
5. Dé CLIC DERECHO sobre el botón *Delete* y dé CLIC sobre **Properties....**
6. Marque el casillero **Hide**, y oprima el botón **OK**.
7. Dé CLIC DERECHO sobre el botón *Change*.
8. Marque el casillero **Hide**, y oprima el botón **OK**.
9. Dé CLIC DERECHO sobre el botón *Insert* y dé CLIC sobre **Properties...** .
10. Marque el casillero **Hide**, y oprima el botón **OK**.

Mover los botones y cambiar el tamaño de la lista

No hay necesidad de desperdiciar el espacio que ocupan esos botones (que el usuario no puede ver), de manera que los sacaremos de en medio.

1. Dé CLIC sobre el botón **Close** y luego oprima MAYÚS+CLIC y ARRASTRE el botón hacia arriba, dentro del listado.

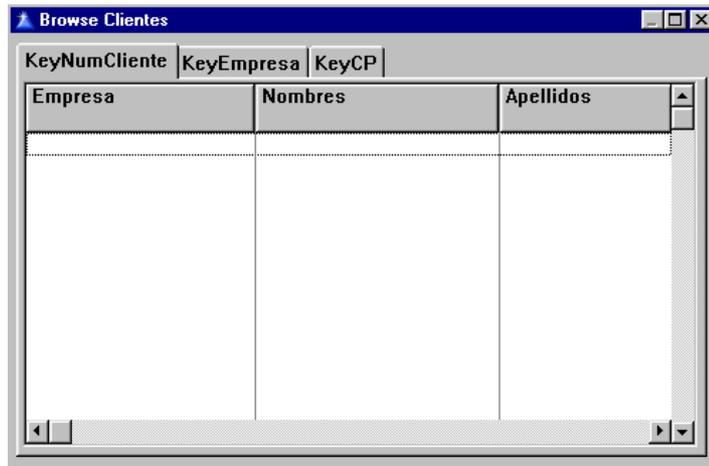
ARRASTRAR un botón con la tecla MAYÚS (SHIFT) oprimida permite mover el control en la línea recta: si empieza a moverlo hacia abajo sólo irá hacia arriba y hacia abajo; si empieza a moverlo hacia un lado, sólo se desplazará horizontalmente.
2. Oprima el botón **NO** cuando se le pregunte si desea moverlo dentro de la lengüeta (Tab).
3. Dé CLIC sobre el botón *Select* y luego dé CTRL+CLIC sobre los botones *Insert*, *Change* y *Delete* para seleccionarlos todos, y luego dé CLIC y arrastre los botones hacia la caja de listado.

ARRASTRAR controles múltiples simultáneamente permite moverlos manteniendo sus posiciones relativas dentro del grupo.
4. Oprima el botón **NO** cuando se le pregunte si desea moverlos dentro de la lengüeta (Tab).
5. Ahora, usaremos el espacio ganado para extender el listado.
6. Dé CLIC sobre la caja de listado, y ARRASTRE el asa inferior del centro hacia abajo, para prolongar la lista.

Prueba del Browse de clientes

El Diseñador de Ventanas tiene un modo de prueba, que ofrece una vista preliminar de cómo aparece la ventana sobre el escritorio. Dado que se trata de una ventana MDI, aparecerá dentro del marco del Diseñador de Ventanas.

1. Elija **Preview!** en la barra del menú del Diseñador de Ventanas.



2. Oprima Esc (o el botón **X**) para regresar al Diseñador de Ventanas.

Ordenamientos de visualización

Ahora que han sido colocadas las lengüetas, debemos indicar a la caja de listado qué ordenamientos utilizar.

1. Dé CLIC DERECHO sobre la caja de listado, y elija **Actions...** de menú contextual.

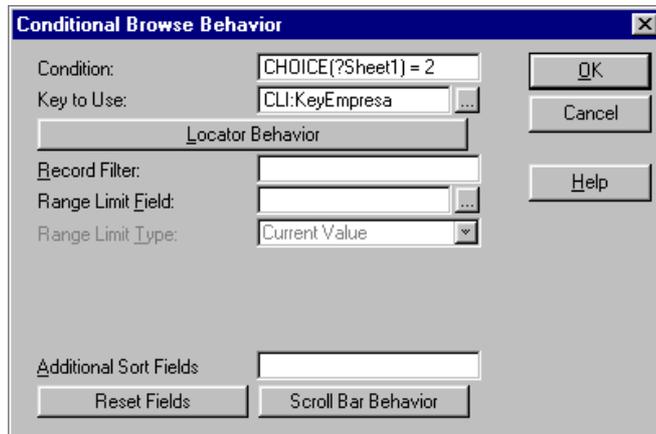
La caja de listado es en realidad una plantilla de control *BrowseBox* que ha sido colocada en la ventana preconfigurada de la plantilla de procedimiento *Browse*, en el Registro de Plantillas (*Template Registry*) (véase la *User's Guide* para más información sobre el *Template Registry*). Esto significa que tiene indicadores asociados que le indicarán cómo llenar el listado y las acciones a realizar.

Los indicadores que aparecen en la lengüeta **Actions** vienen directamente de los *templates* (en este caso, la plantilla *BrowseBox*). De esta manera, puede comunicarse a las plantillas exactamente qué código deben generar para producir exactamente la funcionalidad que usted necesita. Esos indicadores, sus usos y significados, están cubiertos en la *Guía del usuario* y en la ayuda en línea de cada ventana en que aparecen.

2. Seleccione la lengüeta de funcionamiento condicional (**Conditional Behavior**).
3. Oprima el botón **Insert**.
4. Escriba `CHOICE(?Sheet1) = 2` en el campo **Condition**.

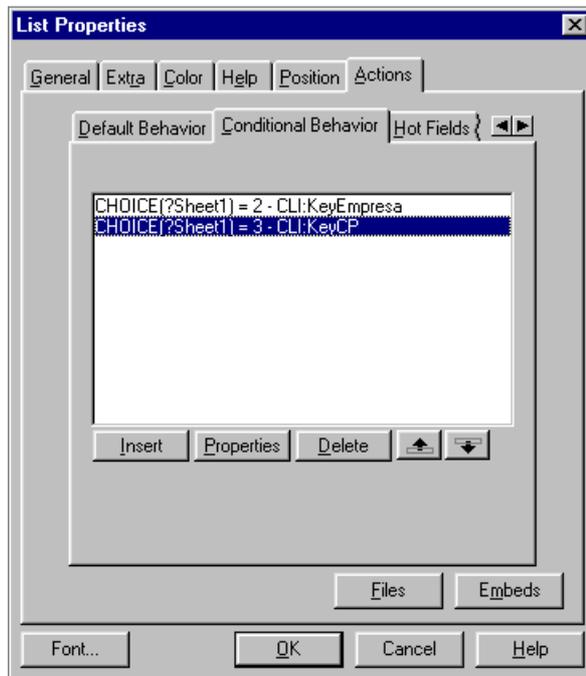
Esto indica la condición bajo la cual se utilizará el orden alterno de visualización. Esta expresión utiliza la función `CHOICE` del lenguaje Clarion (véase *Language Reference*) para detectar cuando el usuario ha elegido la segunda lengüeta de la hoja. El código generado usará esa expresión en una sentencia condicional que cambiará el orden de visualización cuando el programa se ejecute.

5. Oprima el botón con tres puntos suspensivos(...) junto al campo **Key to Use**.
6. Destaque `CLI: KeyEmpresa` y oprima el botón **Select** en la caja de diálogo **Select Key**.



Ahora, cuando el usuario elija la segunda lengüeta, la plantilla de control *BrowseBox* generará código para elegir la clave del campo *Empresa*. No necesita saber qué hacer con la primer lengüeta, dado que siempre utiliza la clave de acceso que configuramos en el árbol de archivos (*File Schematic*).

7. Oprima el botón **OK**.
8. Oprima el botón **Insert**.
9. Escriba `CHOICE(?Sheet1) = 3` en el campo **Condition**.
10. Oprima el botón con tres puntos suspensivos(...) junto al campo **Key to Use**.
11. Destaque `CLI: KeyCP` y oprima el botón **Select** en la caja de diálogo **Select Key**.
12. Oprima el botón **OK**.



13. Oprima el botón **OK** para cerrar la caja de diálogo **List Properties**.

Cierre el Browse de Clientes

Elija **Exit!** en la barra del menú del Diseñador de Ventanas, y guarde los cambios en la ventana, cuando se le indique.

Oprima el botón **OK** en la caja de diálogo **Procedure Properties** para cerrarla.

Elija **File** ➤ **Save**, u oprima el botón Guardar para grabar el trabajo realizado.

8 - CREACIÓN DE UNA FICHA O FORMULARIO

Creación de un procedimiento de actualización

En el capítulo anterior, diseñamos la caja de listados del procedimiento *browse* de Clientes, y añadimos lengüetas para cambiar el orden de visualización. Para terminar con el procedimiento básico, nombraremos el procedimiento de actualización (*update procedure*). Este es el procedimiento que maneja la acción de los botones de Agregar, Cambiar y Borrar (*Insert, Change y Delete*).

Punto de inicio:

El archivo TUTORIAL.APP debe estar abierto.

Incorporación de un procedimiento “ToDo”

1. Destaque *VerClientes* en la caja de diálogo del árbol de la aplicación, y oprima el botón **Properties**.

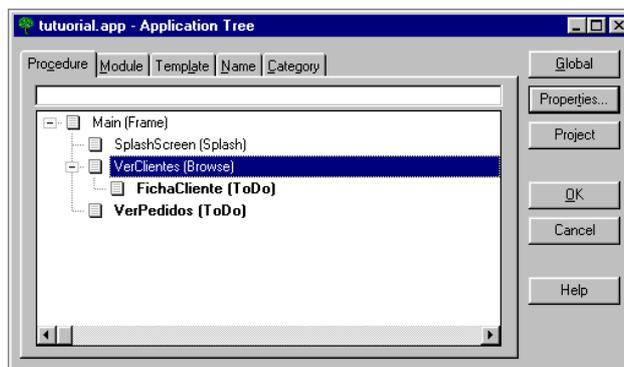
Aparece la caja de diálogo de propiedades del procedimiento (**Procedure Properties**). Hay tres modos de llegar a esta caja de diálogo, para que usted use el que más le convenga:

- ◆ Destaque el procedimiento y luego oprima el botón **Properties**.
- ◆ Dé DOBLE CLICK en la caja de diálogo del árbol de la aplicación.
- ◆ Dé CLIC DERECHO sobre el procedimiento y elija **Properties** del menú contextual.

2. Escriba *FichaCliente* en la caja de entrada de datos **Update Procedure** ubicada al pie de la caja de diálogo **Procedure Properties**.

Esto da nombre al procedimiento que permite añadir y modificar(actualizar) los registros que muestra el *browse*. El nuevo procedimiento aparece en el árbol de la aplicación como “Por Hacer” (ToDo).

3. Oprima el botón **OK** para cerrar la caja de diálogo **Procedure Properties**.



Advierta que no tuvo que iniciar un nuevo hilo de ejecución para este procedimiento. Debe correr sobre el mismo hilo que el *browse*, para que el usuario no pueda abrir una ventana de ficha o formulario para cambiar un registro, después active la ventana de *browse* de nuevo, y abra otra ficha sobre el mismo registro. En otras palabras, ¡no sería bueno dejar que el usuario intente cambiar el mismo registro dos veces a la vez!

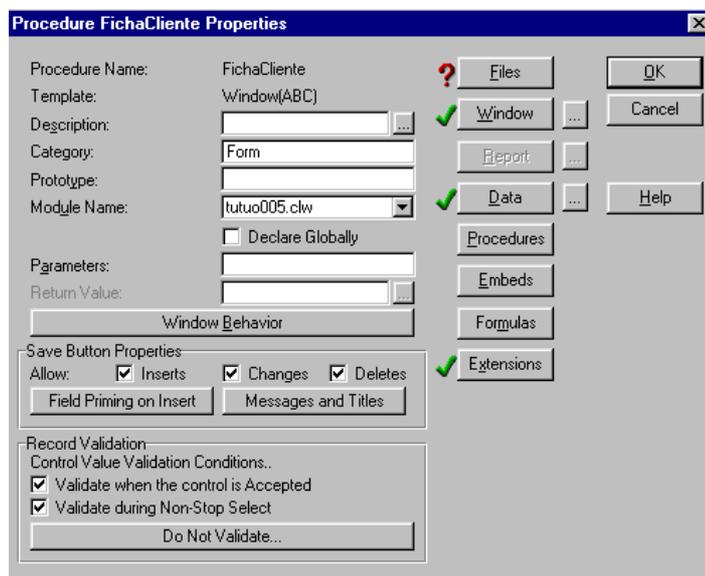
Creación del procedimiento de Ficha o Formulario

El procedimiento de actualización debe utilizar el template *Form Procedure* para crear un procedimiento que el usuario final pueda utilizar para mantener los registros.

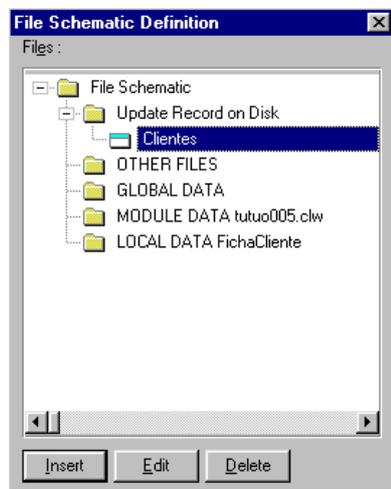
Selección del tipo de procedimiento para FichaCliente

1. Dé DOBLE CLIC sobre *FichaCliente* en el árbol de la aplicación.
2. Ilumine la plantilla de procedimientos *Form*, despeje el casillero **Use Procedure Wizard**, y oprima el botón **Select**.

Aparecerá la ventana **Procedure Properties**. Advierta que esta caja dialogo luce diferente a las de los demás procedimientos con que hemos trabajado, porque los indicadores a la izquierda varían según cada tipo de Procedure template. El *Application Handbook* y la ayuda en línea describen las opciones de personalización disponibles en cada caja de diálogo **Procedure Properties**.



3. Oprima el botón **Files** para nombrar el archivo sobre el que trabajará el formulario. Aparece la caja de diálogo **File Schematic Definition**.
4. Destaque el ítem “ToDo” debajo de la expresión “Actualización del archivo en el disco” (**Update Record on Disk**) y oprima el botón **Insert**.
5. Destaque el archivo *Clientes* en la caja de diálogo **Select File**, y oprima el botón **Select**.



6. Oprima el botón **OK** para regresar a la ventana **Procedure Properties**.

Hacer ajustable el tamaño de la ventana

1. En la caja de diálogo **Procedure Properties**, oprima el botón **Extensions**.
2. En la caja de diálogo **Extensions and Control Templates**, oprima el botón **Insert**.
3. Ilumine *WindowResize* en la caja de diálogo **Select Extension**, y oprima el botón **Select**.

Esta plantilla de Extensión genera código para manejar automáticamente el ajuste de tamaño y reubicación de todos los controles de la ventana cuando el usuario cambia el tamaño de la ventana, o cuando oprime los botones de Maximizar o Restaurar.

4. Oprima el botón **OK** para cerrar la caja de diálogo **Extensions and Control Templates**.

Modificación de las propiedades de la ventana

1. En la caja de diálogo **Procedure Properties**, oprima el botón **Window**.
2. Dé CLIC DERECHO sobre la ventana de título, y elija **Properties** del menú contextual.
3. Elija *Resizable* (“tamaño ajustable”) de la lista desplegable **Frame Type**.
4. Seleccione la lengüeta **Extra** y marque el casillero **Maximize Box**.

Estos dos últimos ítem permiten que el usuario modifique el tamaño de la ventana durante la ejecución.

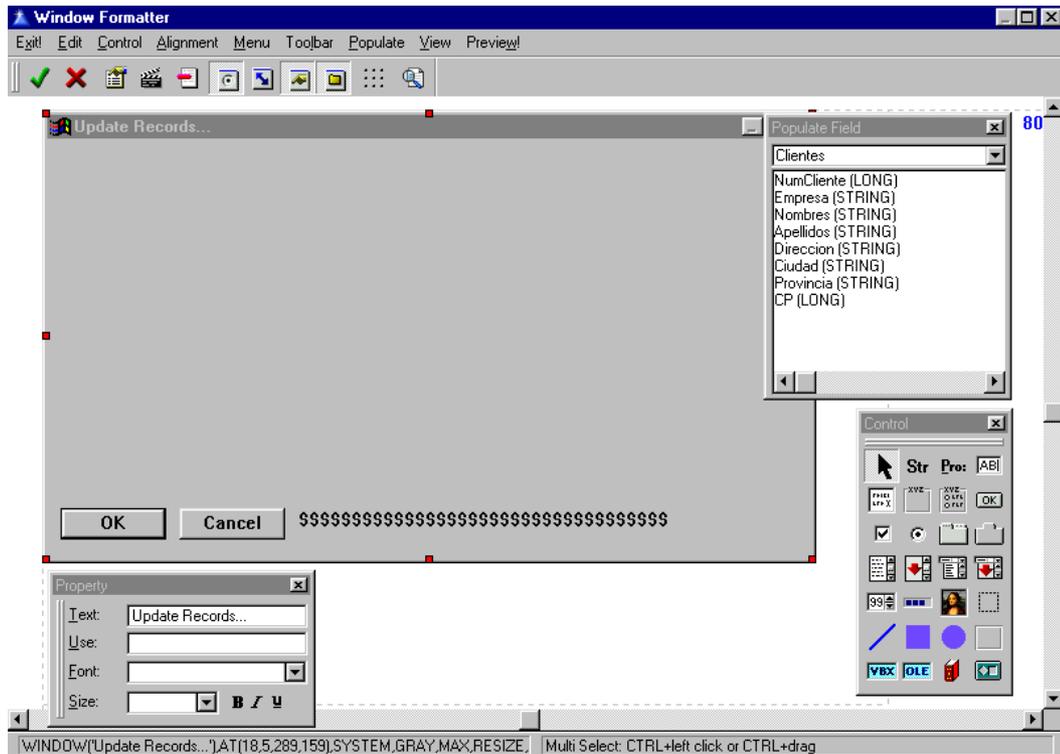
5. Oprima el botón **OK** para cerrar la caja de diálogo **Window Properties**.

Llenado de los campos

El diseño proconfigurado de la ventana ya contiene tres campos. El botón *OK* que cierra la caja de diálogo, aceptando lo ingresado por el usuario y graba la información en el disco. El botón *Cancel* que cierra el formulario sin grabar cambios. Y el campo textual provee un mensaje informativo

para el usuario sobre la acción que está realizando sobre el registro. Esta acción se llama en inglés *to populate*, cuya traducción literal es *poblar* (la ventana).

1. Elija **Options** ➤ **Show Fieldbox**



Esto muestra una caja flotante de herramientas que contiene todos los campos especificados en el árbol de archivos de la aplicación.

2. Dé CLIC sobre *NumCliente* en la caja flotante de herramientas **Populate Field**, y mueva el cursor sobre el diseño de la ventana.

El cursor cambia a una cruz y un “librito” que indica que el campo proviene del diccionario de datos.

3. Dé CLIC cerca del ángulo superior izquierdo de la ventana.

Esto coloca tanto el campo como su etiqueta asociada. Estos controles traen todas las configuraciones especificadas anteriormente en el diccionario.

4. Dé DOBLE CLIC sobre *Empresa* en la caja flotante de herramienta **Populate Field**.

El DOBLE CLIC coloca inmediatamente tanto el campo como su etiqueta justo debajo de los primeros campos que acaba de colocar. Podría seleccionar y colocar cada campo como hicimos con el primero, pero es mucho más rápido con DOBLE CLIC.

5. Dé DOBLE CLIC sobre *Nombres* en la caja flotante de herramientas **Populate Field**.

6. Dé DOBLE CLIC sobre *Apellidos* en la caja flotante de herramientas **Populate Field**.

7. Dé DOBLE CLIC sobre *Dirección* en la caja flotante de herramientas **Populate Field**.

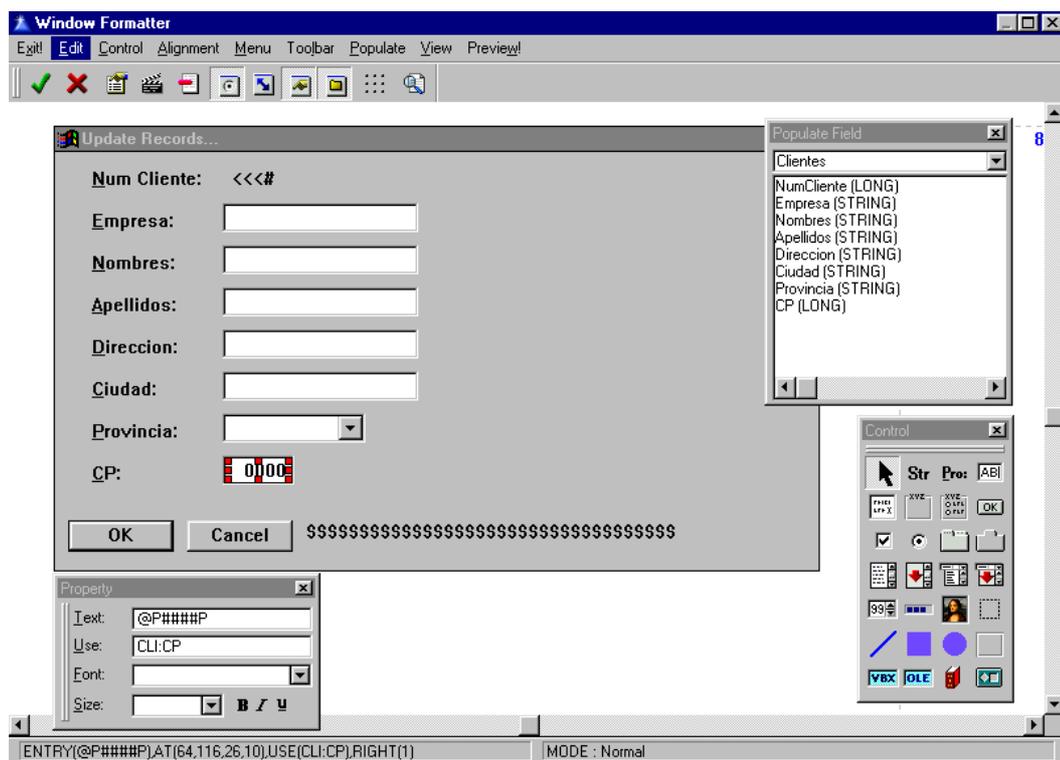
8. Dé DOBLE CLIC sobre *Ciudad* en la caja flotante de herramientas **Populate Field**.

9. Dé DOBLE CLIC sobre *Provincia* en la caja flotante de herramientas **Populate Field**.

Esto coloca la etiqueta y la lista desplegable. En el diccionario de datos predefinimos este campo como un control de Lista con el atributo DROP. Dado que los valores válidos (BA|SF|ER|CBA|MZA) que contiene ya están predefinidos, no hay necesidad de modificar su formato.

10. Dé DOBLE CLIC sobre *CP* en la caja flotante de herramientas **Populate Field**.

La ventana de la ficha o formulario se ve ahora así:



Mover y alinear los campos

Para una apariencia más profesional, debemos mover los campos y alinear los márgenes de todos los controles.

Ubicación de los campos en sus posiciones aproximadas

1. Dé CLIC sobre el listado *Provincia*.
2. ARRASTRE mientras mantiene oprimida la tecla de mayúsculas (MAYÚS+ARRASTRE) la lista *Provincia* a la derecha, más cerca de su etiqueta.

MAYÚS+ARRASTRE limita el movimiento del control al plano en que comenzó a moverse (sea vertical u horizontal).

3. Dé CTRL+CLIC sobre la etiqueta *Provincia*.

Cuando se da CTRL+CLIC sobre un control, las asas del control seleccionado previamente se vuelven azules, mientras que las del control recién seleccionado aparecen en rojo. Ahora hay dos controles seleccionados.

- Mueva la lista desplegable *Provincia* y su etiqueta (dé CLIC y ARRASTRE sobre cualquiera de los controles seleccionados), arriba y a la derecha de los controles *Ciudad*.

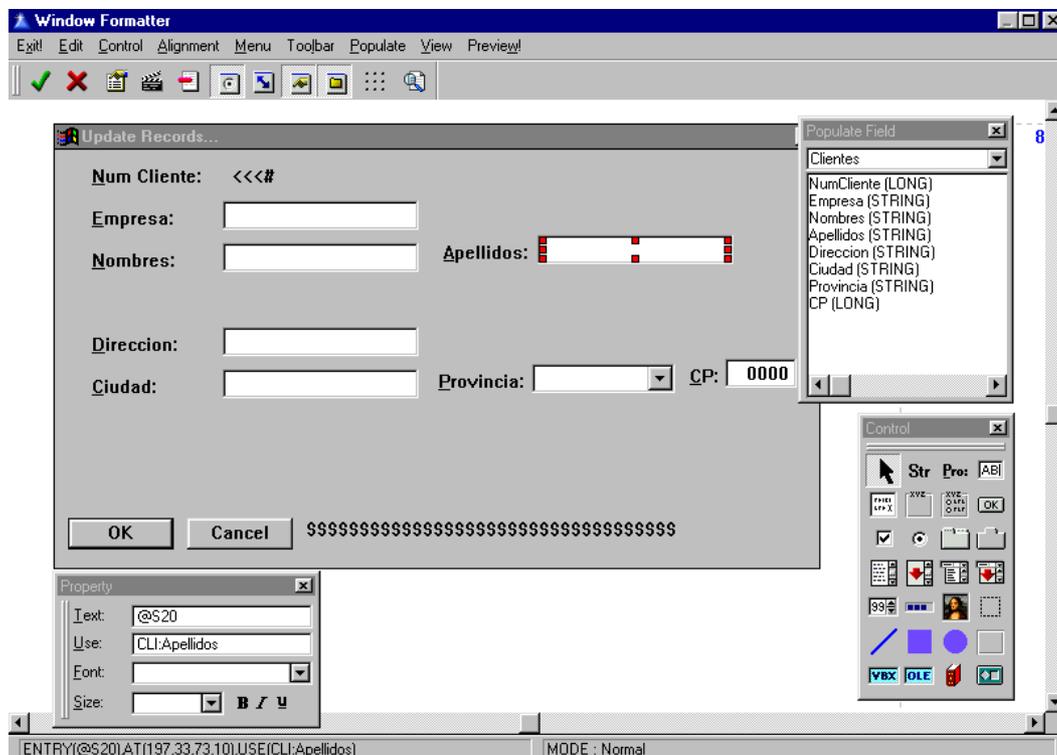
Cuando se han seleccionado controles múltiples, se los puede mover en grupo, de igual manera que se desplazaría un control individual.

- Dé CLIC sobre el control de ingreso de datos *CP*.
- Oprima MAYÚS+ARRASTRE sobre la caja de *CP*, a la izquierda, más cercano a su etiqueta.
- Dé CTRL+CLIC sobre la etiqueta *CP*.
- Mueva la caja de ingreso de datos *CP* y su etiqueta arriba y a la derecha de los controles *Ciudad* y *Provincia*.

Quizá deba ensanchar un poco la ventana para lograrlo.

- Dé CLIC sobre la caja de ingreso de datos *Apellido*, a la derecha, más cerca de su etiqueta.
- Dé SHIFT+ARRASTRAR sobre la caja *Apellido* hacia la izquierda, más cerca de su etiqueta.
- Dé CTRL+CLIC sobre la etiqueta *Apellido*.
- Mueva la caja de ingreso de datos *Apellido* más arriba y a la derecha de *Nombres*.

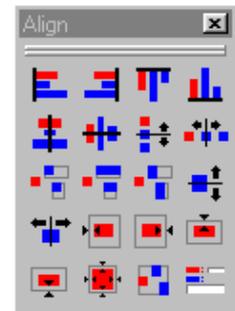
Ahora la ventana debería aparecer aproximadamente así:



Alineamiento final de los controles

1. Escoja **Option** ➤ **Show Alignbox** (“Mostrar caja de alineación”)

El diseñador de Ventanas tiene una caja flotante de herramientas que contiene el mismo juego de herramientas de alineación que están disponibles a través del menú **Alignment**. La caja flotante de herramientas **Alignbox** puede no verse exactamente igual que la mostramos aquí, debido a que es posible cambiar la forma de cualquiera de las barras flotantes ARRASTRANDO sus bordes. Inténtelo y comprobará su flexibilidad. Gracias a esta técnica podrá configurar su espacio de trabajo como lo desee.



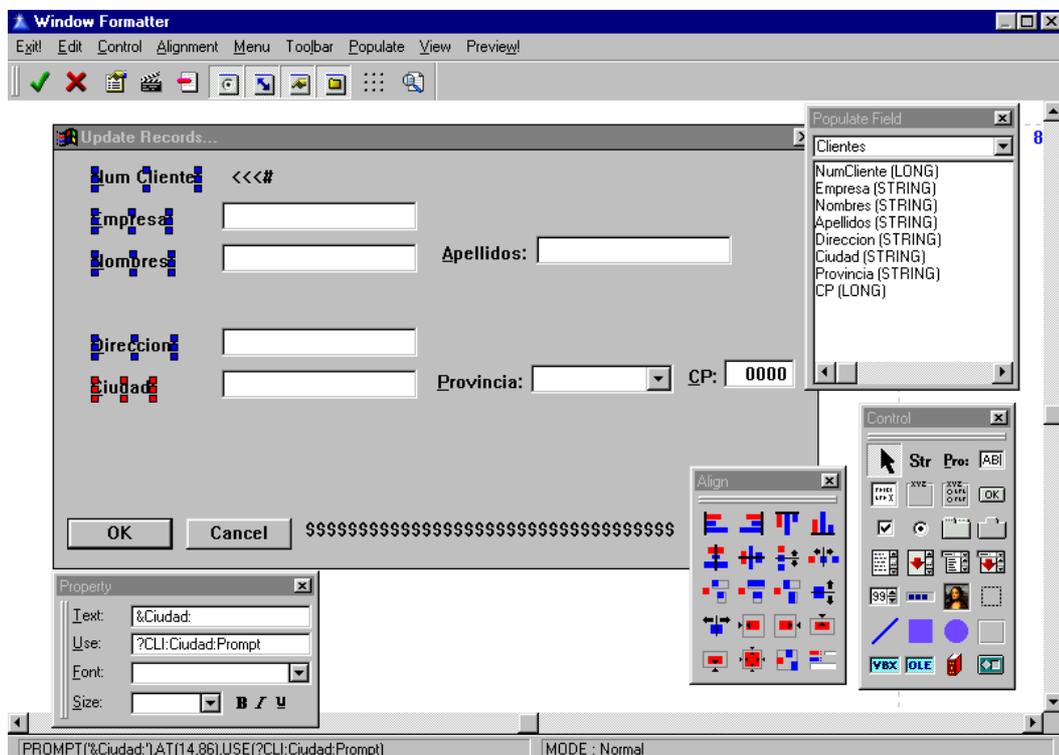
2. Dé CLIC sobre la primera etiqueta en el ángulo superior izquierdo.

Este debe ser la etiqueta *NumCliente*. Cuando le da CLIC deben aparecer las asas alrededor.

3. Dé CTRL+CLIC en las cuatro etiquetas ubicadas inmediatamente debajo de la primera.

A medida que da CTRL+CLIC sobre cada control, las asas del que estaba seleccionado previamente pasan a azul, y las del control recién seleccionado aparecen rojas. El control con las asas rojas es el “punto de referencia” o “ancla” para la operación de alineamiento. Todos los demás controles se alinean en relación al que tiene las asas rojas.

Con las cinco etiquetas seleccionadas debería verse así:



4. Oprima el botón **Align Left** (el botón superior a la izquierda) en la caja flotante de herramientas **Align**.

Los controles se alinean por el lado izquierdo, con respecto al último ítem seleccionado (con las asas rojas).

Para identificar los controles en la caja de herramientas **Align**, simplemente coloque el cursor del *mouse* sobre el control y espere medio segundo hasta que aparezca la acotación identificatoria.

5. Oprima el botón **Spread Vertically** en la caja flotante de herramientas **Align**.

Los controles se distribuyen en forma pareja entre el primer y el último botón seleccionados.

6. Dé CLIC sobre el primer control de ingreso de datos (*NumCliente*)
7. Dé CTRL+CLIC sobre los controles de ingreso inmediatamente abajo, para seleccionarlos a todos.
8. Dé CLIC DERECHO y elija “alineación izquierda” (**Align Left**) del menú contextual que aparecerá.

Esta es otra forma de activar las herramientas de alineación. El menú contextual de alineamiento aparece solamente cuando ha seleccionado múltiples controles.

9. Dé CLIC DERECHO y elija **Spread Vertically** del menú contextual.

Esto debería alinear todos los controles de ingreso de datos con las etiquetas respectivas, que ya hemos sometido a este alineamiento.

10. Dé CLIC sobre la caja de ingreso de datos *Apellidos*.
11. Dé CTRL+CLIC sobre los tres controles a la izquierda (la etiqueta, el campo *Nombres* y la etiqueta).
12. Elija alineamiento horizontal (**Align Horizontally**) de la caja flotante de herramientas **Align** o del menú contextual (CLIC DERECHO).

Hay otra forma de elegir controles múltiples en el Diseñador de Ventanas: utilizar el lazo.

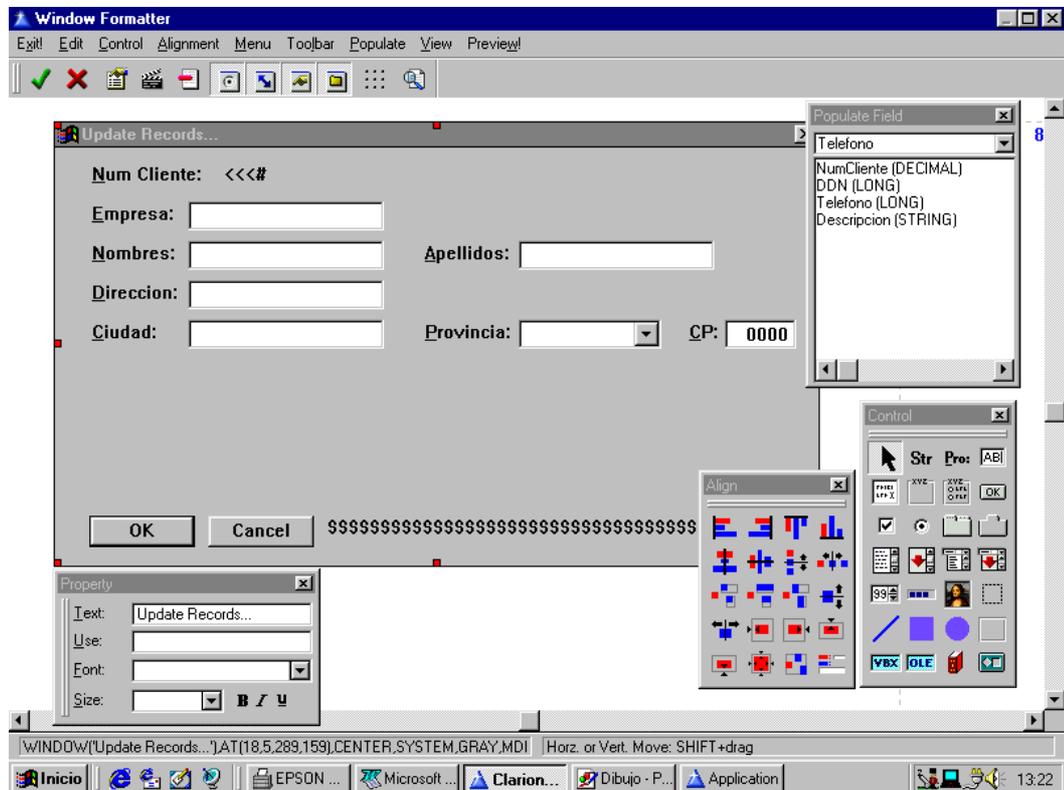
13. Coloque el cursor del *mouse* ligeramente sobre y a la derecha del primer control en la fila inferior (que sería *Ciudad*).
14. Dé CTRL+CLIC y ARRASTRE ligeramente hacia abajo y a la derecha hasta que el dibujo de la caja rodea los cinco controles de la derecha (las etiquetas y campos de *Ciudad*, *Provincia* y *CP*), y suelte el botón del *mouse*.

Las asas rojas aparecen en el campo CP y las azules en los demás controles.

Esta es la técnica del “lazo”.

15. Elija el alineamiento horizontal (**Align Horizontally**) sea de la caja de herramientas flotante o del menú contextual de alineamiento (CLIC DERECHO).
16. Use esa misma herramienta para alinear los campos *NumCliente*, *Empresa* y *Dirección* con sus etiquetas respectivas.

La ventana debería aparecer ahora así:



La ventana está casi lista. Ahora añadiremos una caja de listado para los registros de Teléfonos vinculados.

Adición de un template de Control BrowseBox

Los templates (plantillas) de control generan todo el código fuente requerido para crear y mantener un tipo de control, o grupo de controles, específico en la ventana. Todos los controles de ingreso de datos que acabamos de colocar no son simples controles, ni plantillas de control, porque no necesitan código extra para cumplir sus funciones normales. Las plantillas de control se usan solamente cuando un control específico necesita funcionalidad extra no provista por el control mismo. Por ejemplo, los botones OK y Cancel son ambos *template* de control: el botón OK guarda el archivo al disco, mientras que la plantilla de control Cancel realiza toda la “limpieza” necesaria para cancelar la operación en curso.

Ahora vamos a colocar un *template* de control *BrowseBox* que muestra todos los registros del archivo *Teléfonos* vinculados al registro actual de Clientes.

Colocación del template de control

1. Elija **Populate** > **Control Template**, o dé CLIC sobre la herramienta *Control Template* en la caja flotante de herramientas **Controls** (último icono de herramientas a la derecha, en la fila inferior).

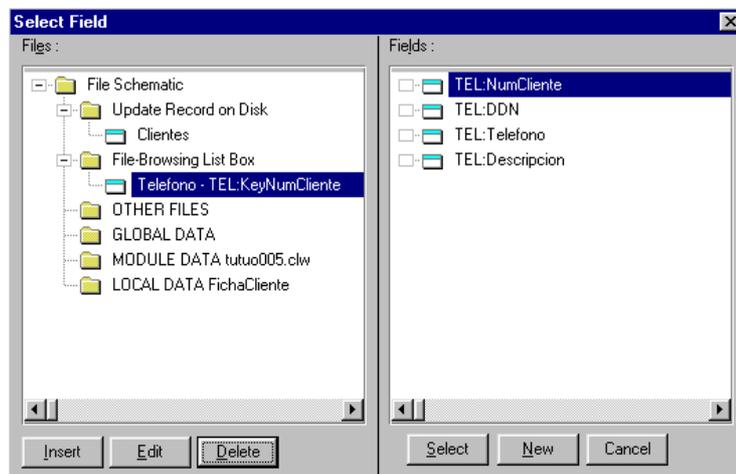
2. En la caja de diálogo **Select Control Template** destaque *BrowseBox*, y oprima el botón **Select**.

El cursor cambia a una cruz y un librito.

3. Dé CLIC justo debajo de la caja de ingreso de datos *Ciudad* para colocar el control.

Aparece la caja de selección de campos para que usted elija el archivo que se mostrará en este *BrowseBox*.

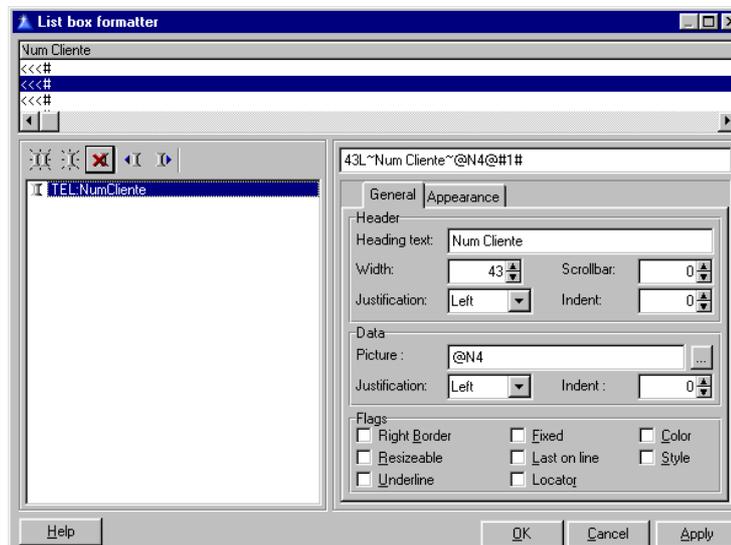
4. Elija el ítem "ToDo" debajo de la caja **File-Browsing List Box** y oprima el botón **Insert**.
5. Destaque el archivo *Teléfonos* en la caja de diálogo **Insert File**, y oprima el botón **Select**.
6. Destaque el archivo *Teléfonos* en la lista **Files**, y oprima el botón **Edit**.
7. Destaque *KeyNumCliente* en la caja de diálogo **Change Access Key**, y oprima el botón **Select**.



Colocación de los campos del archivo Telefonos

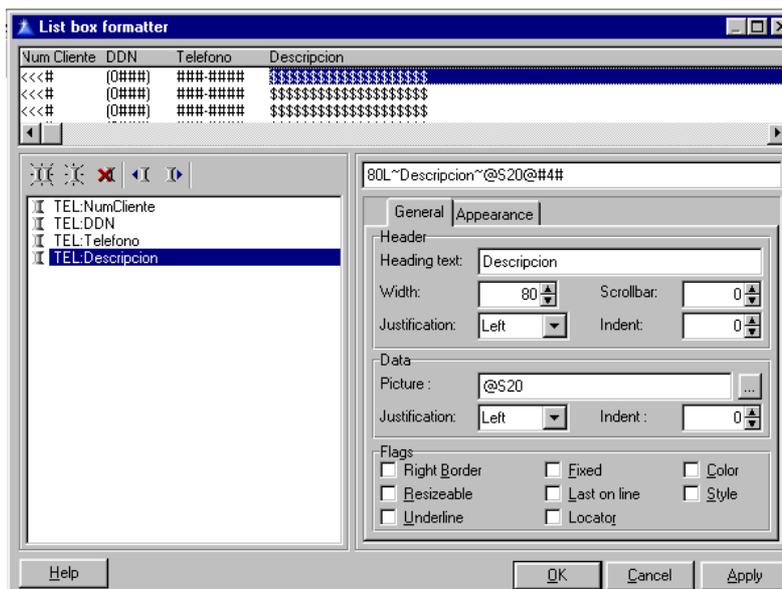
1. Destaque *Tel:NumCliente* en la lista **Fields**, y oprima el botón **Select**.

Aparece el Diseñador de Cajas de Listado para agregar el resto de los campos que desea mostrar.



2. Seleccione justificación izquierda (**Left**) de la lista desplegable **Justification**.
Esto cambia la justificación predeterminada (derecha para valores numéricos).
3. Oprima el botón **New Column**.
4. Ilumine (destaque) *TEL:DDN* en la lista **Fields**, y oprima el botón **Select**.
5. Seleccione **Left** de la lista **Justification**.
6. Oprima el botón **New Column**.
7. Destaque *TEL:Teléfono* en la lista **Fields**, y luego oprima el botón **Select**.
8. Seleccione **Left** de la lista **Justification**.
9. Oprima el botón **New Column**.
10. Destaque *TEL:Descripcion* en la lista **Fields**, y luego oprima el botón **Select**.

Como paso opcional, modifique las columnas de la ventana para hacerlas suficientemente anchas para los respectivos encabezamientos. Simplemente, ARRÁSTRELAS con el *mouse*.



11. Oprima el botón **OK** para cerrar el Diseñador de Listados.

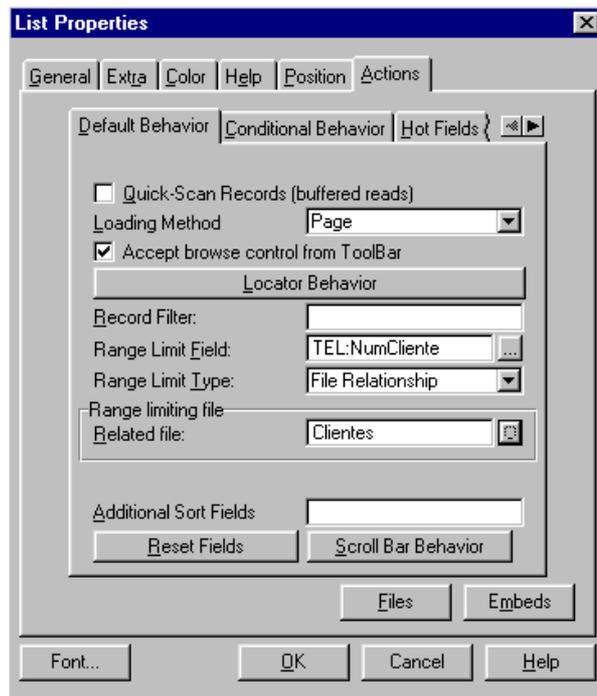
Esto coloca la caja de listados en la ventana, en la posición que hemos especificado. Esto puede expandir la ventana. Si es así, puede modificar el tamaño de la ventana ARRASTRANDO las asas, y moviendo los controles **OK**, **Cancel** y Mensaje hacia el nuevo pie de la ventana.

Configuración de los límites de alcance del template

1. Dé CLIC DERECHO en la caja que acaba de colocar, y elija **Actions** del menú contextual.
2. Oprima los puntos suspensivos(...)ubicados junto a **Range Limit Field**.
3. Destaque el campo *TEL:NumCliente* en la lista de componentes de las claves (**key Components**), y oprima el botón **Select**.

4. Elija “relación entre archivos” (**File Relationship**) de la lista desplegable de límite de búsqueda (**Range Limit Type**).
5. Oprima los puntos suspensivos (...) junto al campo **Related File** y elija *Clientes* de la caja de diálogo **Select File**.

Esto identifica al archivo de *Clientes* como el archivo relacionado. Estas etapas limitan los registros que se muestran en la caja de listados a solamente los registros vinculados al registro de Clientes que se esté mostrando en el momento de la consulta.



6. Oprima el botón **OK** de la caja de diálogo **List Properties**.

Adición del template de control BrowseUpdateButtons

A continuación añadiremos los botones estándar Insert, Change y Delete para el control de la caja de listados.

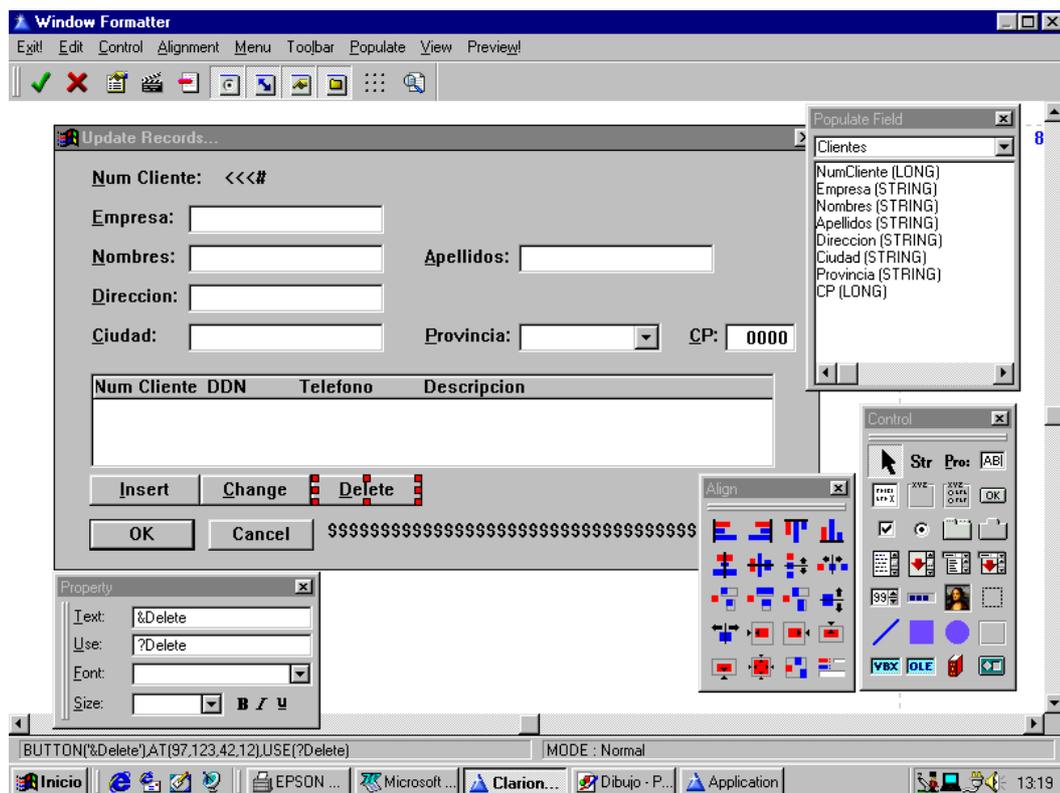
Colocación de otro tipo de template de control

1. Escoja **Populate > Control Template**, o dé CLIC sobre la herramienta *Control Template* en la caja de herramientas **Controls** (el último icono de la lista, en la fila de abajo).
2. Destaque la plantilla de control *BrowseUpdateButton*, y oprima el botón **Select**.
El cursor cambia a una cruz y un librito.
3. Dé CLIC debajo del extremo inferior izquierdo de la caja de listado.

Aparecen los botones Insert, Change y Delete. Recuérdese: esos son los botones que permitirán el funcionamiento de los botones de la barra de tareas, y por eso deben estar presente en el diseño de la ventana. No es necesario que estén visibles para el usuario, de

manera que puede esconderlos si lo desea. Sin embargo, dado que esta caja de listados (*BrowseBox*) se ubica sobre un procedimiento de actualización (*Form*), en esta aplicación dejaremos visibles este grupo de botones. Esto permitirá usar cualquiera de los dos grupos de botones. Los botones de la barra de herramientas sólo funcionarán sobre esta lista cuando la caja de listados tiene el foco (no cuando el usuario ingresa datos en ningún otro control), de manera que dejar visibles a esos botones asegurará que el usuario pueda mantener fácilmente los registros de Teléfonos.

En este punto, la ventana debería verse aproximadamente así:



Especificar “Modificar in situ” para actualizar los Teléfonos

1. Dé CLIC DERECHO sobre el botón de “Borrar” (**Delete**) y elija **Actions** del menú contextual.
2. Marque el casillero “Modificación in situ” (**Use Edit in place**).

Configurar las acciones (**Actions**) para un botón sirve para los tres botones del grupo, dado que los tres pertenecen a la misma plantilla de control. Dado que el archivo *Teléfonos* tiene sólo un par de campos, no hay necesidad de un procedimiento de actualización separado.

3. Oprima el botón **OK**.
4. Elija **Exit!** para cerrar el Diseñador de Ventanas.
5. Oprima el botón **OK** para cerrar la caja de diálogo **Procedure Properties**.
6. Elija **File** ➤ **Save**, u oprima el botón *Guardar* de la barra de herramientas.

9 - COPIADO DE PROCEDIMIENTOS

Los Procedimientos de "Productos"

Ahora que hemos creado el *browse* de Clientes, podemos reutilizar ese trabajo para el siguiente procedimiento, copiándolo y cambiando después los campos. En este capítulo, copiaremos el procedimiento *VerClientes* para crear *VerProductos*.

También usaremos los "Puntos de inserción" (Embeds points) para escribir "código fuente insertado", que llame al procedimiento *VerProductos* desde el menú y barra de herramientas de la aplicación. Así presentaremos los numerosos puntos en que se pueden agregar algunas (o muchas) líneas de código fuente propio para añadir funcionalidad a cualquier procedimiento.

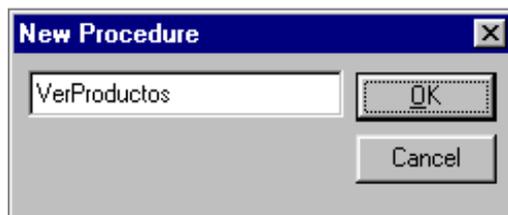
Punto de inicio :

El archivo TUTORIAL.APP debe estar abierto.

Copia de los procedimientos

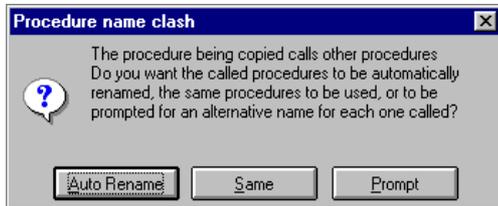
Como se recordará cuando se creó el ítem de menú **Ver > Productos**, y el botón de la barra de herramientas titulado "Productos", no especificamos un procedimiento a llamar cuando el usuario final los ejecutara. Empezaremos creando el procedimiento a llamar.

1. Destaque el procedimiento *VerClientes* en el árbol de la aplicación .
Este es el procedimiento que copiaremos.
2. Escoja **Procedure > Copy...**
Aparece la caja de diálogo **New Procedure**.
3. Escriba *VerProductos* en la caja de entrada, y oprima el botón **OK**.



Debido a que el procedimiento *FichaCliente* está enlazado a *VerClientes* (el que está copiando), aparece una caja de diálogo advirtiendo sobre el conflicto de nombres (*Procedure name clash*), con opciones para manejar la situación. La traducción del mensaje es: "El procedimiento que está copiando llama a otros procedimientos. ¿Desea que les dé nuevo nombre automáticamente a los procedimientos llamados, utilizar los mismos nombres, o

indicar nombres alternativos en cada caso?". Hay tres botones al pie: " Auto Rename" (dar nuevo nombre automáticamente),"Same"(dejar el mismo), y "Prompt" (preguntar).



4. Oprima el botón **Prompt**.

Al oprimir este botón, usted le pide al Generador de Aplicaciones que pregunte antes de dar el nuevo nombre a los procedimientos conflictivos.

Aparece otra caja de mensaje que informa sobre un duplicado de nombre en particular:



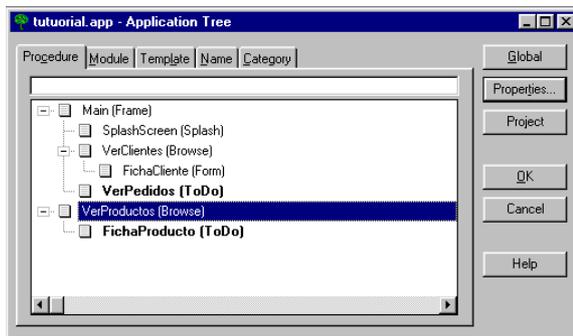
5. Oprima el botón **Rename**.

Aparece la caja de diálogo **Alternate Procedure** (procedimiento alternativo).

6. Escriba *FichaProducto* en la caja de ingreso, y oprima el botón **OK**.



Aparecerán los procedimientos *VerProductos* y *FichaProducto* en el árbol de la aplicación. Se ven "desconectados" del resto de los procedimientos, porque no son llamados por ningún otro procedimiento (todavía). Ahora nos encargaremos de eso.



Trabajando en los puntos de inserción

Los templates de Clarion le permiten añadir su propio código fuente a muchos puntos predefinidos dentro del código generado. Es un modo muy eficiente de lograr la máxima capacidad de

reutilización y flexibilidad del código. El punto en que se introduce el código se llama " Punto de inserción " (*Embed Point*). Estos puntos se hallan disponibles para todos los eventos comunes de la ventana y de cada control, y en muchas otras posiciones lógicas dentro del código generado.

En este ejemplo, añadiremos código fuente insertado (utilizando una plantilla de Código que escribirá el código fuente por usted) en los puntos donde el usuario final elige la opción **Ver > Productos** del menú, y en el punto en que ese usuario oprime el botón **Productos** sobre la barra de herramientas de la aplicación.

Dar nombre a un procedimiento

1. Dé CLIC DERECHO al procedimiento *Main* del árbol de la aplicación.

Hay varias formas de ingresar a los puntos de inserción dentro de un procedimiento. Dos aparecen en el menú contextual que acaba de aparecer..

La primera opción es **Embeds**, que llama a la caja de diálogo **Embedded Source**, que muestra una lista de todos los puntos de inserción dentro del procedimiento.

La segunda forma es la opción **Source**, que en realidad genera código fuente para el procedimiento y llama al "Embeditor" (El editor de texto en modo de inserción de código) para permitirle modificar todos los puntos de inserción dentro del contexto del código fuente generado. Este último aparece "grisado" para indicarle que no es posible modificarlo, y cada punto de inserción posible del procedimiento se identifica mediante comentarios, tras los cuales usted puede insertar su propio código.

Hay ventajas en cada método de trabajo con los puntos de inserción, así que cubriremos ambos durante este cursillo. Primero utilizaremos la caja de diálogo **Embedded Source**.

2. En el menú contextual, elija **Embeds**.

Aparece la caja de diálogo **Embedded Source**, que le da acceso a todos los puntos de inserción del procedimiento. También se puede llegar aquí con el botón **Embeds** en la ventana **Procedure Properties**, pero es más rápido mediante el menú contextual. La lista está ordenada alfabéticamente o en el orden en que los puntos aparecen en el código fuente generado, según sea la configuración de **Sort Embeds Alphabetically** en las opciones del entorno (**Setup > Application Options**).

3. Oprima el botón **Contract All**.

Esto facilitará ubicar el punto de inserción específico que necesite.

4. Ubique la carpeta **Control Event Handling**, y dé CLIC sobre el signo + para expandir el contenido.

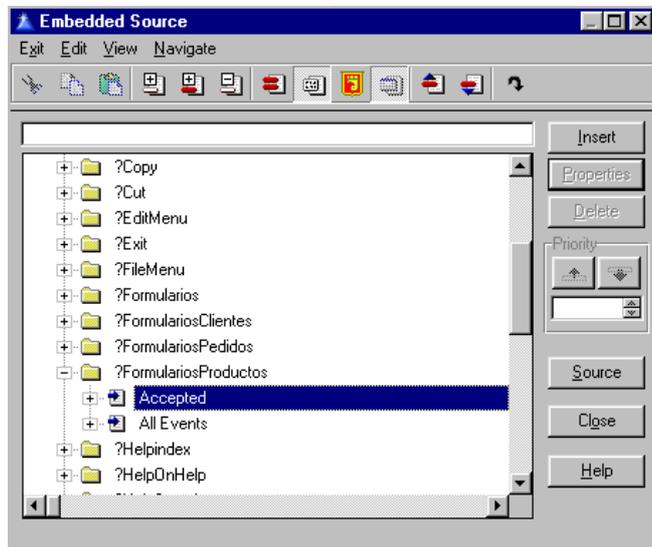
La selección del menú es un control, tal como una caja de ingreso de datos, o la ventana misma.

En el lado derecho de la ventana, advertirá los botones con las flechas hacia arriba y hacia abajo y un spin box que le permiten establecer una prioridad (**Priority**); éstos son importantes. A veces, el código que usted quiere escribir debe ejecutarse **antes** del código generado por el template, y otras veces debe ejecutarse **después**, y en ocasiones debe ejecutarse en algún punto intermedio **dentro** del código generado. La prioridad establece la ubicación exacta de su código embebido. Esto le proporcionan tanta flexibilidad para ubicar su código embebido como es posible. El valor de la prioridad en sí mismo, no es importante, pero sí su posición lógica dentro del código generado. Pronto aclararemos más sobre este tema.

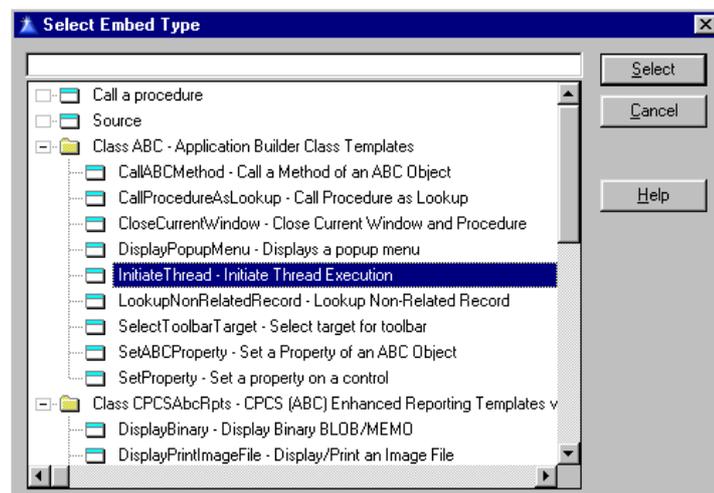
5. Ubique la carpeta **?FormulariosProducto**, y dé CLIC sobre ella para expandirla.

6. Destaque **Accepted**, y oprima el botón **Insert**.

El evento "aceptado" para la selección del menú marca el punto del código generado que se ejecuta mientras el usuario elije la orden del menú.



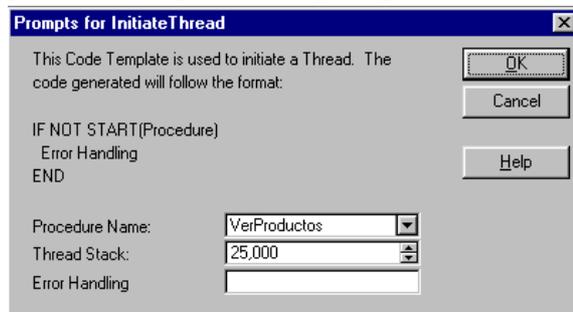
Aparece la caja de diálogo **Select embed type** con una lista de todas las opciones para incorporar código. Se puede, simplemente, llamar a un procedimiento (**Call a Procedure**), escribir su propio código fuente (**Source**) en lenguaje Clarion, o utilizar una plantilla de código (**Code Template**) que escriba el código necesario. Esta es una ventaja de editar los puntos de inserción dentro de la caja de diálogo **Embedded Source**: puede utilizar las plantillas de código para que le escriban código, en lugar de tener que hacerlo por sí mismo.



7. Seleccione la plantilla de código **Initiate Thread**, y oprima el botón **Select**.

Una plantilla de código generalmente viene con algunas indicaciones e instrucciones para su uso. Obtiene de usted la información que necesita para escribir el código ejecutable, que a su vez inserta dentro del código generado estándar producido por el *Procedure template* directamente en este punto de inserción. Esta plantilla de código en particular está diseñada para comenzar un nuevo hilo de ejecución llamando, mediante el procedimiento **START**, al procedimiento que usted indique.

8. Elija *VerProductos* de la lista desplegable **Procedure Name**. Esto indica el procedimiento a llamar cuando el usuario elige el ítem del menú. Se trata del procedimiento que hemos copiado previamente.

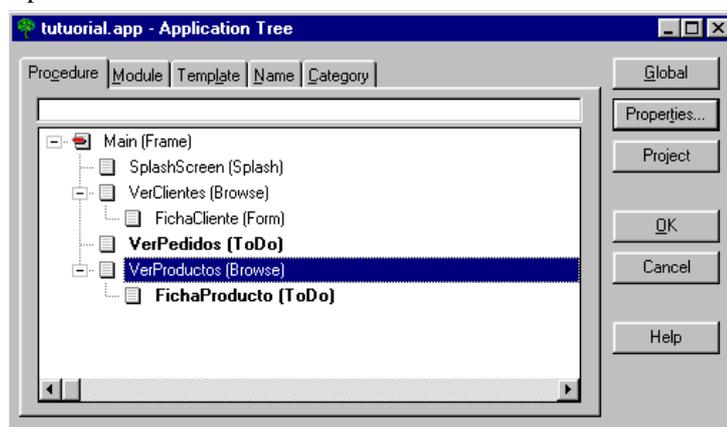


9. Oprima el botón **OK**.

Dar nombre al procedimiento a llamar mediante el botón Productos de la barra de herramientas

En este punto, usted podría hacer lo mismo para llamar el procedimiento *VerProductos* desde el botón Productos. Sin embargo, hay un modo más fácil de escribir de nuevo este código: ¡simplemente Cópíelo y Péguelo desde un punto de inserción al otro!

- Dé CLIC sobre el botón **Copy** (el botón central del grupo ubicado a la derecha del botón **Contract All**).
El *Code template* que acaba de añadir todavía debería estar destacado, así que esta acción lo copiará al portapeles de Windows.
- Ubique la carpeta *?BotonProductos* (en la misma carpeta **Control Even Handling** en que ya usted se halla ubicado), y dé CLIC sobre ella para abrirla.
- Ilumine *Accepted* (“Aceptado”) y el botón **Paste** (“Pegar”) (el botón ubicado a la extrema derecha del grupo de botones a la derecha del botón **Contract All**).
Aparece nuevamente la caja de diálogo **Procedure name class** para advertirle que ya ha llamado una vez este procedimiento.
- Oprima el botón “El mismo” (**Same**).
Ahora este punto de inserción generará el mismo código que el anterior.
- Oprima el botón **Close**.



El procedimiento *VerProductos* ahora “se conecta” debajo del procedimiento *Main*. Ahora puede personalizar los procedimientos copiados para que hagan referencia al archivo *Productos*.

Modificación del *browse*

Cambiar el archivo del nuevo control de listados

1. Dé CLIC DERECHO sobre el procedimiento *VerProductos* y elija **Window** del menú contextual.
2. Dé CLIC DERECHO sobre el control de lista elija **List Box Format...** del menú contextual.
3. En el Diseñador de Listados oprima el botón de borrar (**Delete**) repetidamente, hasta haber eliminado todos los campos.
Cuando haya eliminado el último campo, aparece automáticamente la caja de diálogo de Selección de Campos.
4. Destaque el archivo *Cientes* de la lista de archivos (**Files**) y oprima el botón **Delete**.
5. Destaque “**To Do**” que reemplaza al archivo de clientes y presione el botón **Insert**.
6. Destaque el archivo *Productos* y presione el botón **Select**.
7. Oprima el botón **Edit** y seleccione *KeyDescription* del la caja de diálogo **Change Access Key**.

La caja de diálogo **Select Field** muestra ahora el archivo y los campos correctos.

Re poblado de los campos

1. Ilumine *PROD:NumProducto* en la lista **Fields**, y oprima el botón **Select**.
2. Marque las casillas **Right Border** y **Resizable**, e incremente la caja **Ident** a uno (1).
3. Oprima el botón **New Column**.
4. Marque *PROD:Descripción* en la lista de campos (**Field**), y oprima el botón **Select**.
5. Seleccione *Left* para la justificación del dato.
6. Oprima el botón **New Column**.
7. Ilumine *PROD:Precio* en la lista **Fields**, y oprima el botón **Select**.
8. Incremente la indentación en el grupo *Data* a 12. Como la justificación es decimal se requiere de este paso para asegurarnos que dejamos lugar para que aparezca la parte fraccionaria
9. Oprima el botón **New Column**.
10. Ilumine *PROD:TasaImponible* en la lista **Fields**, y oprima el botón **Select**.

Note que la justificación por defecto es decimal, y que el spin que indica la indentación del grupo *Data* se mantiene en doce (12) tal como lo indicamos en el campo anterior.

11. Oprima el botón **OK** para cerrar el Diseñador de Listados.

No se preocupe por los botones que aparecen sobre la caja del listado. Recuerde que estos están ocultos, y son llamados por los botones de la barra de herramientas.

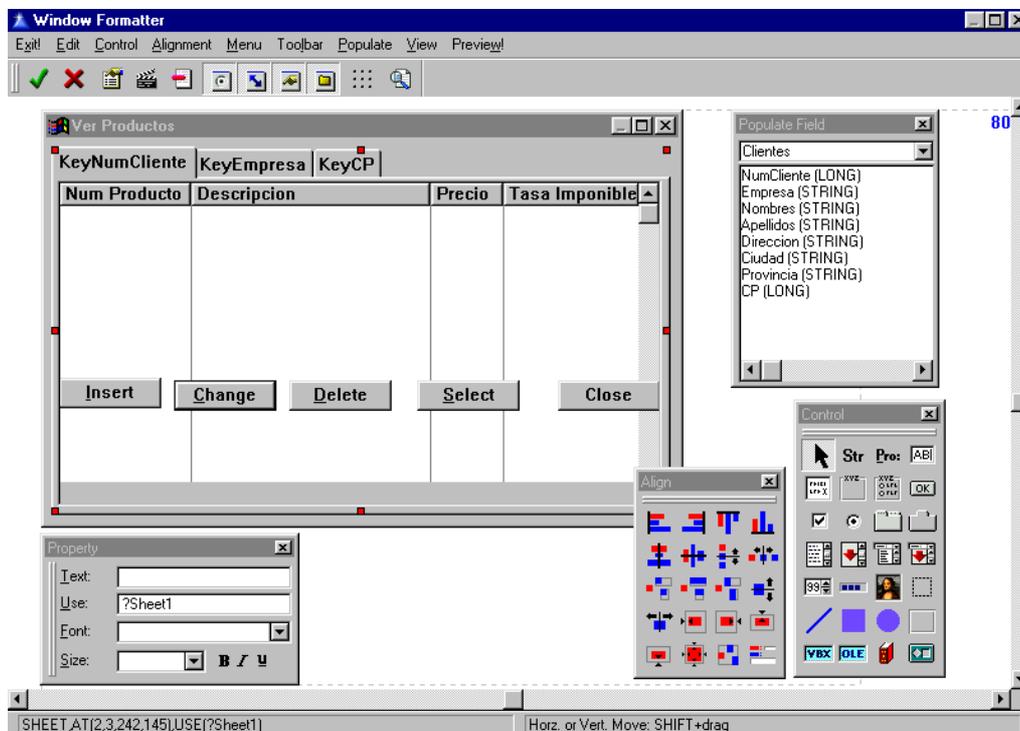
Cambiar el nombre de la ventana

1. Dé CLIC sobre la barra de título de la ventana.
2. Escriba *VerProductos* en el campo de texto (**Text**) de la caja flotante de herramientas **Property**, y oprima TAB.

Quitar todas las lengüetas

1. Dé CLIC justo a la derecha de la lengüeta *KeyCP* para seleccionar toda la hoja de propiedades.

Para estar seguro de haber dado CLIC sobre el lugar correcto, observe la caja flotante de herramientas **Property**, y observe si el campo **Use** muestra *?Sheet1*. Si no es así, pruebe de nuevo, hasta lograrlo.



2. Oprima la tecla DELETE.

Todas las lengüetas desaparecen.

Adición de un localizador

Si la lista de Productos a mostrar es muy larga, el usuario tendrá que hacer una extensa búsqueda antes de encontrar el producto requerido. Inicialmente, todas las cajas de visualización de listados traen un localizador por “pasos” (*Step*) que va al primer registro que comienza con la letra que el operador ingrese en el teclado.

Pero, con bases de datos muy grandes, el usuario necesita ingresar varias letras iniciales para llegar al registro buscado. Un localizador de ingreso (*Entry locator*) es un campo en el que el usuario puede escribir. Al oprimir TAB, la lista se mueve hacia el primer registro que coincida con lo tipeado. Esta característica funciona solamente con claves alfanuméricas (STRING).

1. Si no está ya presente, elija **Options** ➤ **Show Fielbox** para mostrar la caja flotante de herramientas **Populate Field**.

Si la caja de herramientas **Populate Field** todavía muestra los campos del archivo Clientes, simplemente cierre la caja de herramientas (con DOBLE CLIC sobre la barra de título) y reábralo para refrescar la caja de herramientas con los campos del archivo Productos.

2. Dé DOBLE CLIC en *Descripcion* sobre la caja de herramientas **Populate Field**.

Esto coloca tanto la etiqueta como la caja de ingreso para el campo ubicado cerca del ángulo superior izquierdo de la ventana. Este es el campo localizador.

3. Elija **Exit!** del menú para cerrar el Diseñador de Ventanas, y guarde los cambios.

Limpieza de los ordenamientos alternativos

1. Dé CLIC DERECHO sobre el procedimiento *VerProductos* y elija **Extensions** en el menú contextual.

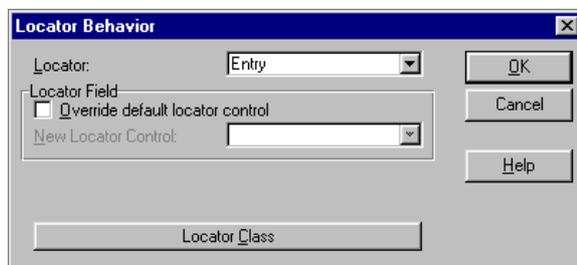
Aparece la caja de diálogo **Extension and Control Template**. Esta caja lista todos los templates de control que hay en el procedimiento y su etiquetas de acciones (**Actions**).

Esta caja de diálogo también permite añadir y mantener templates o plantillas de extensión en el procedimiento. Las plantillas de extensión son muy similares a las de control, en que añaden funcionalidad específica al procedimiento, pero la funcionalidad de una plantilla de extensión no se vincula directamente con ningún control que aparezca en la ventana. En otras palabras, se trata de plantillas que añaden funcionalidad “tras bambalinas”, sin afectar directamente la interfaz con el usuario.

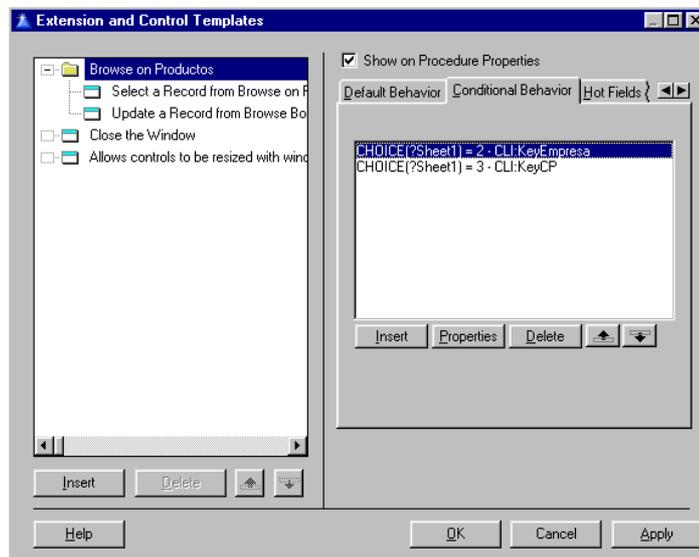
Un muy buen ejemplo de los Extension Templates viene en el producto Clarion Internet Connect. Éste contiene (entre otras herramientas) un grupo de plantillas de extensión que automáticamente “traducen” una aplicación Clarion para Windows en páginas dinámicas HTML. Esto permite que la aplicación Clarion funcione por Internet. El usuario solamente necesita un visualizador con Java habilitado para acceder al programa Clarion. Si esta interesado, puede obtener más información de TopSpeed o Unisoft sobre Internet Connect.

2. Ilumine *Browse on Productos* y presione el botón **Locator Behavior** (Comportamiento del localizador).
3. Seleccione *Entry* de la lista desplegable u oprima **OK**.

Esto completa los requisitos para el localizador. El campo clave del orden de búsqueda (en este caso *PROD:Descripción*) es el control de localización predeterminado.



4. Seleccione la lengüeta **Conditional Behavior**.



5. Oprima el botón **Delete** dos veces.
Esto elimina las dos expresiones condicionales que ingresamos para el procedimiento *VerClientes*.
6. Oprima el botón **OK**.
7. Elija **File** ➤ **Save**, u oprima el botón *Guardar* de la barra de herramientas.

Creación del procedimiento Ficha

Cuando dio nuevo nombre a la referencia del *Update Procedure* mientras copiaba *VerClientes* a *VerProductos*, convirtió a *FichaProducto* en un procedimiento "ToDo". Por lo tanto, es necesario crear una ficha o formulario para actualizar el archivo *Productos*.

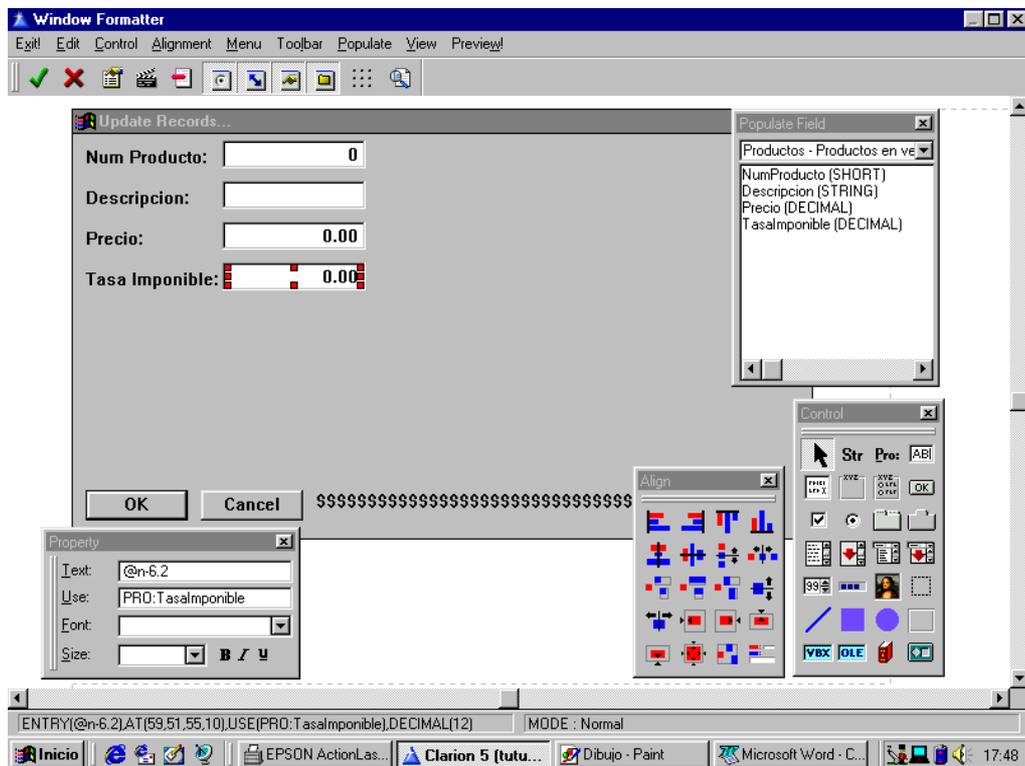
Seleccione el tipo de procedimiento para FichaProducto

1. Destaque *FichaProducto*, y luego oprima el botón **Properties**.
2. Ilumine *Form*, despeje el casillero **Use Procedure Wizard**, y oprima el botón **Select**.
3. Oprima el botón **Files** en la caja de diálogo **Procedure Properties**.
4. Ilumine el archivo "ToDo", y oprima el botón **Insert**.
5. Ilumine el archivo *Productos*, y oprima el botón **Select**.
6. Oprima el botón **OK** para volver a la caja de diálogo **Procedure Properties**.
7. Oprima el botón **Window** para diseñar el formulario.

Poblar los campos

1. Si no está ya presente, elija **Options** ➤ **Show Fieldbox** para mostrar la caja de herramientas **PopulateField**.

2. De DOBLE CLIC sobre *NumProducto* en la caja de herramientas **Populate Field**.
Esto coloca automáticamente tanto la etiqueta y la caja de ingreso del campo cerca del extremo superior izquierdo de la ventana.
3. De DOBLE CLIC sobre *Descripcion* en la caja de herramientas **Populate Field**.
Esto coloca automáticamente tanto la etiqueta y la caja de ingreso abajo del campo ubicado anteriormente.
4. De DOBLE CLIC sobre *Precio* en la caja de herramientas **Populate Field**.
5. De DOBLE CLIC sobre *TasaImponible* en la caja de herramientas **Populate Field**.
El formulario aparece así:



Cambio del título de la ventana

1. Dé CLIC sobre la barra de título de la ventana del formulario.
2. Escriba *Ficha de productos* en el campo de texto (**Text**) de la caja flotante de herramientas **Property** y oprima TAB.

Salida del Diseñador de Ventanas y grabación del trabajo

1. Elija **Exit!** Del menú, para cerrar el Diseñador de Ventanas (guarde los cambios).
2. Oprima el botón **OK** en la caja de diálogo **Procedure Properties** para cerrarlo.
3. Escoja **File > Save**, u oprima el botón *Guardar* en la barra de herramientas para guardar el trabajo.

El formulario de actualización de Productos está ya listo.

10 - USO DE CONTROL TEMPLATES

Para el procedimiento *VerPedidos*, creará una ventana con dos cajas de listados que se deslizan sincronizadamente. Uno mostrará el contenido del archivo de Pedidos, y el otro mostrará los registros vinculados en el archivo de Detalle. Usaremos un procedimiento Window genérico, y lo "poblaremos" con control templates. La única razón para hacerlo así en lugar de comenzar con una plantilla *Browse* normal es para demostrar otro modo de crear un procedimiento : usando templates de control colocados en una ventana genérica (de la misma manera en que fué creado el mismo *template Browse*).

Los control templates generan todo el código fuente para crear y *mantener* un control único o un grupo vinculado de controles. En este caso, la colocación de un template de control *BrowseBox* permite que el Generador de Aplicaciones produzca el código que abre los archivos y pone los datos necesarios en las estructuras que los visualizan.

Punto de inicio:

El archivo TUTORIAL.APP debe estar abierto.

Creación del procedimiento

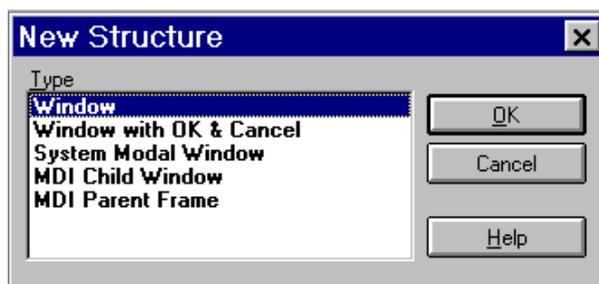
Selección del tipo de procedimiento

1. Destaque *VerPedidos* y oprima el botón **Properties**.
2. Destaque *Window* en la caja de diálogo **Select Procedure Type** y oprima el botón **Select**.

Modificación de la ventana

1. En la caja de diálogo **Procedure Properties**, oprima el botón **Window**.

Aparece la caja de diálogo **New Structure**. El procedimiento genérico de ventana es como un papel en blanco donde se define la ventana que uno desea. Dado que no hay un tipo predefinido de ventana, hay que elegir uno. En este caso, lo que necesitamos es una ventana "hija" MDI.



2. Destaque *MDI Child Window*, y oprima el botón **OK**.
Aparece el Diseñador de Ventanas.
3. Agrande la ventana, haciéndola aproximadamente el doble que el tamaño original.
4. Dé CLIC DERECHO sobre el título, y elija **Properties** del menú contextual.
5. Escriba *Pedidos* en el campo **Text**.
6. Elija *Resizable* de la lista desplegable **Frame Type**.
7. Seleccione la lengüeta **Extra** y marque el casillero **Maximize Box**.
8. Oprima el botón **OK** para cerrar la caja de diálogo **Window Properties**.

Colocación de la caja de listados(**BrowseBox Control Template**)

1. Elija **Populate** ➤ Control Template ó dé CLIC sobre la herramienta *Control Template* en la caja flotante de herramientas **Controls** (última herramienta a la derecha, fila inferior).
Aparece la caja de diálogo **Select Control Template**.
2. Destaque la plantilla de control *BrowseBox*, y oprima el botón **Select**.
El cursor cambia a una cruz y un "librito".
3. Dé CLIC sobre el ángulo superior izquierdo de la ventana de muestra.

Colocar los campos del archivo Pedidos en el Diseñador de Listados

1. Destaque el ítem "ToDo" ubicado debajo de **File-Browsing List Box** y oprima el botón **Insert**.
2. Ilumine el archivo *Pedidos* en la caja de diálogo **Fields**, y oprima el botón **Select**.
3. Oprima el botón **Edit** y seleccione *KeyNumPedido* de la caja de diálogo **Change Access Key**.
4. Destaque *PED:NumCliente* en la lista de campos (**Fields**) y oprima el botón **Select**.
5. Oprima el botón **Populate**.
6. Destaque *PED:NumPedido* en la lista de campos (**Fields**) y oprima el botón **Select**.
7. Oprima el botón **New Column**.
8. Destaque *PED:MontoFactura* en la lista de campos (**Fields**) y oprima el botón **Select**.
9. Incremente la casilla **Indent** del campo a doce (12).
10. Oprima el botón **New Column**.
11. Destaque *PED:FechaPedido* en la lista de campos (**Fields**) y oprima el botón **Select**.
12. Oprima el botón **New Column**.
13. Destaque *PED:NotaPedido* en la lista de campos (**Fields**) y oprima el botón **Select**.
14. Seleccione *Left* de la lista desplegable para la justificación del campo.
15. Ajuste el tamaño de columnas en la caja de listado de muestra.
16. Oprima el botón **OK** para cerrar el Diseñador de Listados.

17. Ajuste el tamaño de la caja de listados, **ARRASTRANDO** el asa del centro del lado derecho hacia la derecha, haciendo la caja casi tan ancha como la ventana.

Mejora del aspecto de la caja de listados

1. Dé **CLIC DERECHO** sobre la caja de listados, y elija **Properties** del menú contextual.
2. Seleccione la lengüeta **Extra** y marque los casilleros **Vertical** y **Horizontal** en la caja de diálogo **List Properties**.

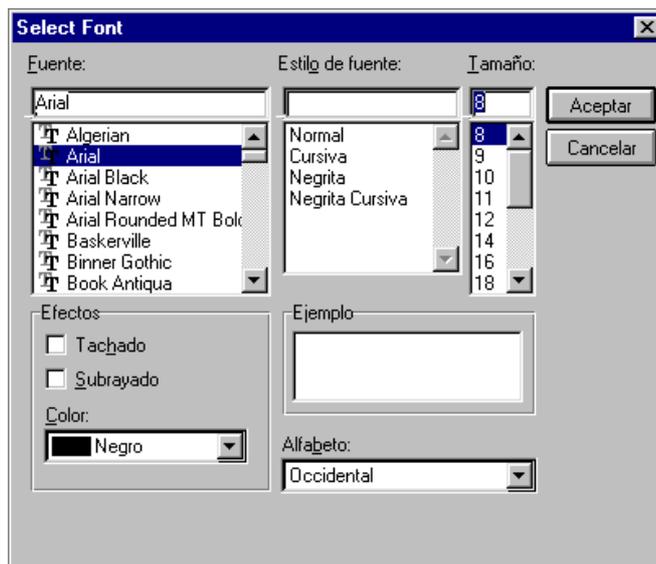
Esto añade barras de desplazamiento horizontal y vertical a la lista.

3. Oprima el botón de fuente (**Font**)

Debido a que un campo (Descripción) es largo, puede especificar que la caja de listados presente una fuente más pequeña, lo que le permite mostrar más información sin que el usuario deba desplazar el cursor.

4. Elija una fuente y configure el tamaño en 8 puntos.

Consulte *User's Guide* para algunas sugerencias sobre temas como escoger las fuentes adecuadas para los controles. En general, es bueno limitarse a las fuentes que vienen con Windows; porque no puede asegurarse que el usuario tenga otro tipo de fuente en su sistema. La ilustración muestra la fuente configurada como Arial, que es común en Windows.



5. Oprima el botón **OK** para cerrar la caja de diálogo **Select Font**.
6. Oprima el botón **OK** para cerrar la caja de diálogo **List Properties**.

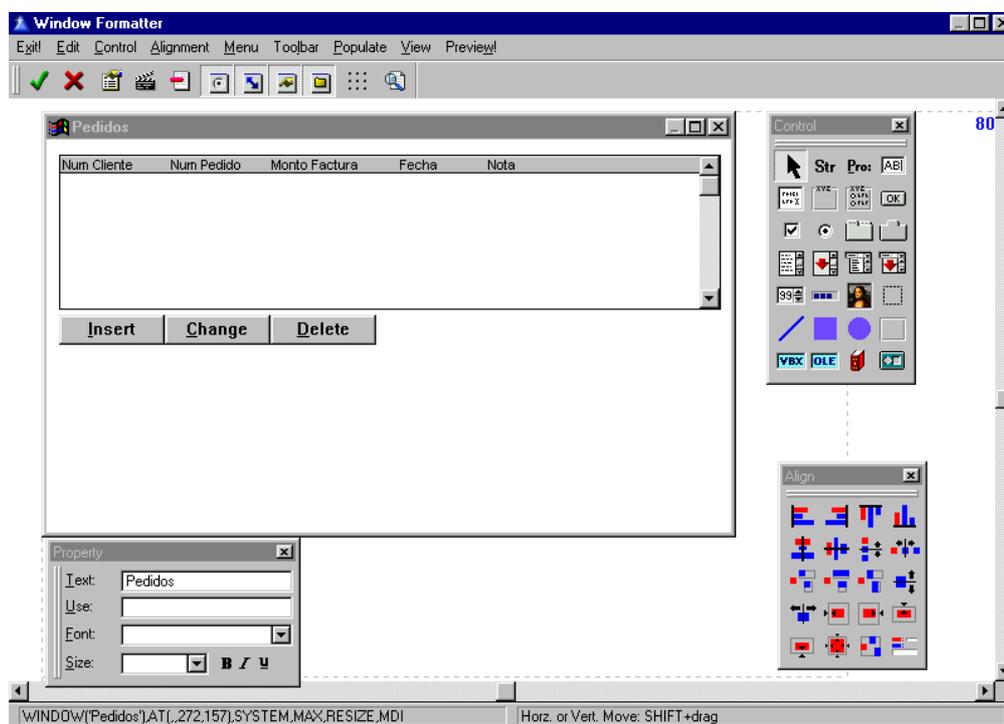
Adición de la plantilla de botones de actualización

A continuación añadiremos los botones **“Agrega”, “Cambia” y “Borra”** (Insert, Change y Delete) para el control de listados superior. Dado que habrá dos listados en esta ventana, dejaremos esos botones visibles para el usuario. Después, añadiremos un procedimiento Form para añadir o editar un pedido.

1. Elija **Populate** ➤ **Control Template**, o dé CLIC sobre la herramienta Control Template en la caja de herramientas Controls (el último icono a la derecha, fila inferior).
2. En la caja de diálogo **Select Control Template**, destaque la plantilla de control *BrowseUpdateButtons*, y oprima el botón **Select**.
3. Dé CLIC debajo del borde izquierdo de la caja de listado

Los tres botones aparecen juntos. Si lo desea, puede dar CLIC DERECHO sobre ellos, y cambiar el Texto de la superficie de cada botón (solamente) por las expresiones traducidas “Agrega”, “Cambia”, y “Borra”.

En este punto, la ventana debería verse así:



Designar el procedimiento de actualización

1. Dé CLIC DERECHO sobre el botón **Delete**, y elija **Actions** del menú contextual.
2. Escriba *FichaPedido* en la caja **Update Procedure**.
Esto da nombre al procedimiento. Designar el procedimiento de actualización para un botón de este Control template vale para los tres botones.
3. Oprima el botón **OK**.

Colocación de la segunda caja de listados

A continuación, coloque una caja de listados con el contenido del archivo de Detalle. Esta se actualizará automáticamente cuando el usuario cambie la selección en la lista superior.

1. Elija **Populate** ➤ **Control Template**, o dé CLIC sobre la herramienta Control Template en la caja de herramientas **Controls** (el último icono a la derecha, fila de abajo).

2. Destaque la plantilla de control *BrowseBox*, y oprima el botón **Select**.
3. Dé CLIC inmediatamente debajo del botón **Insert** que colocó anteriormente.

Colocar los campos del archivo Detalle en el Diseñador de Listados

1. Destaque el ítem “ToDo” ubicado debajo de **File-Browsing List Box** y oprima el botón **Insert**.
2. Ilumine el archivo *Detalle* en la caja de diálogo **Fields**, y oprima el botón **Select**.
3. Oprima el botón **Edit**.
4. Seleccione *KeyNumPedido* de la caja de diálogo **Change Acces Key**, y oprima el botón **Select**.
5. Destaque *DETAL:NumPedido* en la lista de campos (**Fields**), y oprima el botón **Select**.
6. Oprima el botón **New Column**.
7. Destaque *DETAL:NumProducto* en la lista de campos (**Fields**) y oprima el botón **Select**.
8. Oprima el botón **New Column**.
9. Destaque *DETAL:Cantidad* en la lista de campos (**Fields**) y oprima el botón **Select**.
10. Oprima el botón **NewC Column**.
11. Destaque *DETAL:ProdPrecio* en la lista de campos (**Fields**) y oprima el botón **Select**.
12. Incremente la casilla **Indent** del campo a doce (12).
13. Ajuste el tamaño de las columnas de la caja de listados de muestra.
14. Oprima el botón **OK** para cerrar el Diseñador de Listados.
15. Ajuste el tamaño de la caja de listados, ARRASTRANDO las asas, y dándole un tamaño conveniente para la visualización (deje algo de espacio a la derecha para un botón que agregaremos después).

Configuración de los límites de búsqueda

1. Dé CLIC DERECHO sobre la caja de listados que acaba de colocar y seleccione **Actions** del menú contextual.
2. Oprima los puntos suspensivos en “Límites de búsqueda” (**Range Limit Field**).
3. Destaque el campo *DETAL:NumPedido* en el listado de componentes (**Components**), y oprima el botón **Select**.
4. Escoja “Relación entre archivos” (**File Relationship**) de la lista desplegable **Range Limit Type**.
5. Oprima los puntos suspensivos junto a “ Archivo Relacionado” (**Related File**).
6. Destaque el archivo *Pedidos* en la lista **Select File** y oprima el botón **Select**.

Estas últimas cuatro etapas limitan los registros que aparecerán en la segunda caja de listados a solamente aquellos registros de Detalle que se vinculan al registro actual en la caja de listados de Pedidos.

De esta manera la segunda plantilla de control utiliza la relación entre archivos definida en el diccionario de datos para sincronizar la lista de abajo con la de arriba.

Mejora del aspecto de la caja de listados

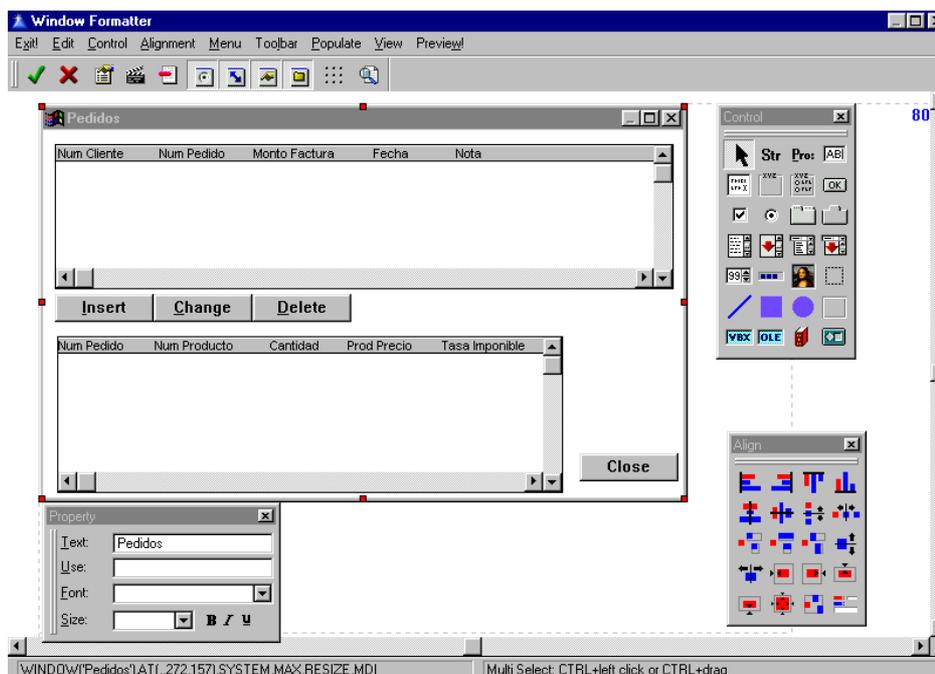
1. Dé CLIC DERECHO sobre la caja de listados, y elija **Properties** del menú contextual.
2. Seleccione la lengüeta **Extra** y marque los casilleros **Vertical** y **Horizontal** en la caja de diálogo **List Properties**.
Esto añade barras de desplazamiento vertical y horizontal al listado.
3. Oprima el botón de fuente (**Font**).
Aunque no hay campos “largos” en este listado, se verá mejor si la fuente es igual a la del listado superior.
4. Elija una fuente, y configure el tamaño en 8 puntos.
5. Oprima el botón **OK** para cerrar la caja de diálogo **Select Font**.
6. Oprima el botón **OK** para cerrar la caja de diálogo **List Properties**.

Adición del template del botón “Cerrar”

Finalmente, puede añadir un botón **Close** que cierre la ventana.

1. Elija **Populate** > **Control Template**, o dé un CLIC sobre la herramienta de plantilla de control en la caja de herramientas **Controls** (último icono a la derecha, lista de abajo).
2. Seleccione la plantilla de control *CloseButton*, y oprima el botón **Select**.
3. Dé CLIC sobre el ángulo inferior derecho de la ventana.

En este punto, la ventana debería verse similar a la siguiente ilustración. Puede verse más grande que el área de muestra en el Diseñador de Ventanas, pero no deberá ser tan grande como el escritorio:



4. Elija **Exit!** del Diseñador de Ventanas para cerrarlo.

Hacer ajustable el tamaño de la ventana

1. En la caja de diálogo **Procedure Properties**, oprima el botón **Extensions**.
2. En la caja de diálogo **Extension and Control Templates**, oprima el botón **Insert**.
3. Destaque *WindowResize* en la caja de diálogo **Select Extension**, y oprima el botón **Select**.

Ya hemos usado varias veces este template de extensión, pero esta vez modificaremos su comportamiento, en lugar de aceptar las opciones prefijadas.

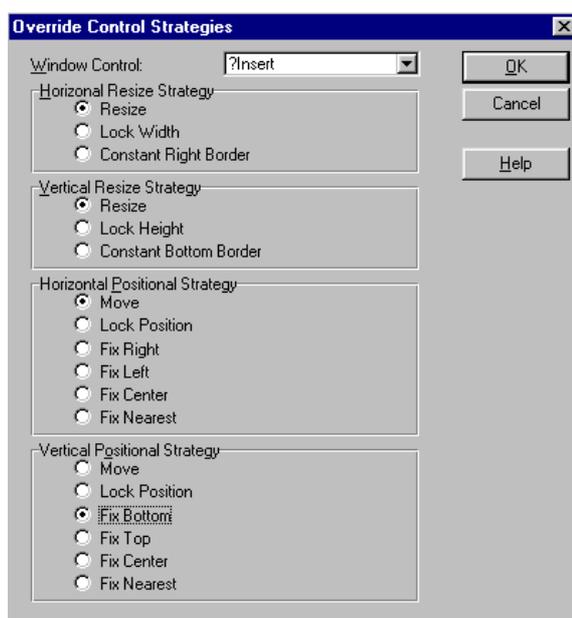
Especificación de las estrategias de ajuste

1. Marque el casillero de restricción del tamaño mínimo de la ventana (**Restrict Minimum Window Size**).

Al marcar este casillero y dejando los campos de anchura y alto mínimos (**Minimum Width – Minimum Height**) en cero, el template asegura que los usuarios no puedan reducir la ventana más allá del tamaño en que fue diseñada.

2. Oprima el botón de cancelación de las estrategias de control (**Override Control Strategies**). Aparece la caja de diálogo correspondiente. Aquí se puede especificar la estrategia de ajuste de tamaño para los controles individuales.
3. Oprima el botón **Insert**.
4. Elija *?Insert* de la lista desplegable **Window Control**.
5. Escoja el botón de radio “Fijar la base” (*Fix Bottom*) en la configuración de opciones **Vertical Positional Strategy**.

Esto determina la estrategia de ajuste del botón **Insert** para mantenerlo a una distancia fija de la base de la ventana. Ahora, haremos lo mismo para los otros dos botones de actualización y la lista de detalle.

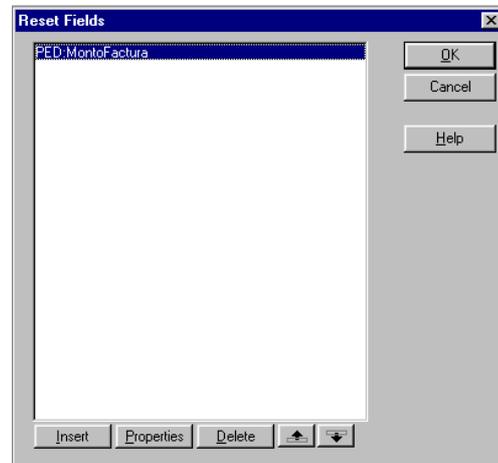


6. Oprima el botón **OK**.
7. Oprima el botón **Insert**, y elija *?Change* de la lista desplegable **Window Control**.
8. Elija el botón de radio “Fijar la base “ (*Fix Bottom*) en la configuración de opciones **Vertical Positional Strategy**, y oprima el botón **OK**.
9. Oprima el botón **Insert**, y elija *?Delete* de la lista desplegable **Window Control**.
10. Elija el botón de radio “Fijar la base “ (*Fix Bottom*) en la configuración de opciones **Vertical Positional Strategy**, y oprima el botón **OK**.
11. Oprima el botón **Insert**, y elija *?List:2* de la lista desplegable **Window Control**.
12. Elija el botón de radio “Fijar la base “ (*Fix Bottom*) en la configuración de opciones **Vertical Positional Strategy**, y oprima el botón **OK**.
13. Oprima el botón **Insert**, y elija *?List* de la lista desplegable **Window Control**.
14. Elija el botón de radio “Borde inferior Constante “ (*Constant Bottom Border*) en la configuración de opciones **Vertical Resize Strategy**

Esto configura la estrategia de ajuste del listado de Pedidos para que el borde inferior de la lista se mantenga a una distancia fija del borde de la ventana. Por lo tanto, la lista se extenderá como sea necesario para llenar el espacio a medida que la ventana se agranda.
15. Oprima el botón **OK** (tres veces) para volver a la caja de diálogo **Procedure Properties**.

Establecer un campo para refrescar (Reset Field)

1. En la caja de diálogo **Extensions and Control Templates**, destaque **Browse on Detail**.
2. Oprima el botón **Resets Fields**, y luego presione **Insert**.
3. Escriba *PED:MontoPedido* en el campo **Reset Field**, luego presione **OK**.
4. Esto especifica que la caja de Listado de Detalle se actualizará cada vez que el valor del campo
5. *PED:MontoPedido* cambie. Esto asegura que al retornar de la modificación de un pedido existente, el cambio efectuado se reflejará en la caja de listado.



Cierre de la caja de diálogo Procedure Properties y guardado de la aplicación

1. Oprima el botón **OK** en la caja de diálogo **Procedure Properties** para cerrarla.
2. Escoja **File** ➤ **Save**, u oprima el botón *Guardar* en la barra de herramientas para preservar el trabajo.

11 - TEMAS AVANZADOS

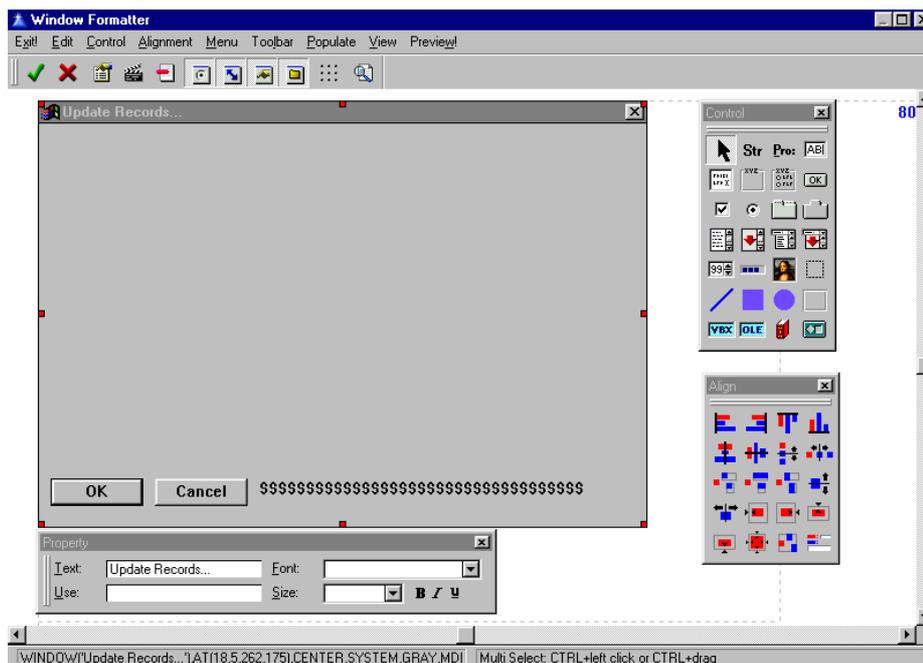
Actualización del archivo de Pedidos

Para el formulario de Pedidos, colocaremos los campos en una ficha de actualización de archivos, realizaremos una consulta automática al archivo Clientes, añadiremos un template de control *BrowseBox* para mostrar y modificar todos los ítems de detalle, calcularemos el monto de cada línea, y luego el total del pedido.

Punto de inicio:
El archivo TUTORIAL.APP debe estar abierto.

Creación del formulario de ingreso de Pedidos

1. Ilumine *FichaPedido* en el árbol de la aplicación, y oprima el botón **Properties**.
2. Destaque *Form* en la caja de diálogo **Select Procedure Type**, despeje el casillero que invoca al Asistente (**Use Procedure Wizard**), y oprima el botón **Select**.
3. Oprima el botón **Window** para abrir el Diseñador de Ventanas.
4. Haga más alta la ventana ARRASTRANDO el asa central superior (utilice la barra de desplazamiento vertical para mover el área de visualización, si fuera necesario).



Colocación de controles de ingreso de datos

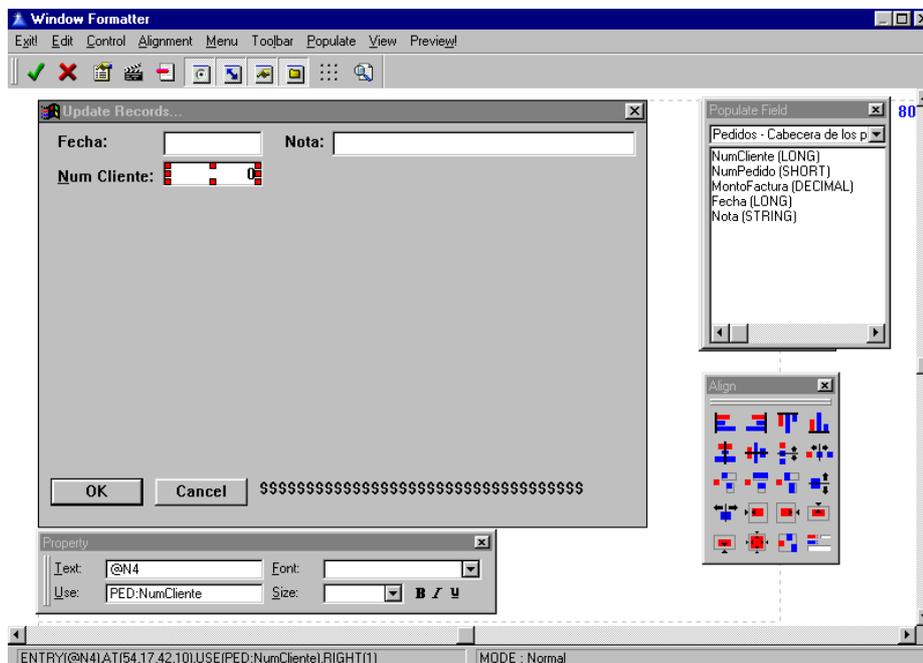
En lugar de utilizar la caja flotante de herramientas **Fieldbox** para cargar los controles, esta vez utilizaremos la caja de diálogo **Select Field** para poner numerosos controles.

1. Elija **Populate** ➤ **Multiple Fields**, o elija la herramienta *Dictionary Field* de la caja de herramientas flotante **Controls**.
2. En la caja de diálogo **Select Field**, destaque el ítem “ToDo” bajo **Update Record on Disk**, y oprima el botón **Insert**.
3. Destaque el archivo *Pedidos* de la lista **Insert File**.
4. Destaque *PED:Fecha* y oprima el botón **Select**.
5. Dé CLIC cerca del extremo superior izquierdo de la ventana.

Esto coloca tanto el campo como su etiqueta. La caja de diálogo **Select Field** reaparece inmediatamente, listo para el campo siguiente, lo que hace este método de colocación de controles casi tan rápido como la caja flotante de herramientas **Fieldbox**.

6. Destaque *PED:NotaPedido*, y oprima el botón **Select**.
7. Dé CLIC justo a la derecha de la caja de ingreso utilizada para la fecha.
8. Marque *PED:NumCliente*, y oprima el botón **Select**.
9. Dé CLIC debajo de la etiqueta del campo de fecha.
10. Oprima el botón **Cancel** en la caja de diálogo **Select Field** para salir de este modo de trabajo.

La ventana de la ficha se ve ahora aproximadamente así:



Añadir una llamada a un procedimiento de consulta

1. Dé CLIC DERECHO sobre el control *PED:NumCliente* y elija **Actions** del menú contextual.

Las acciones comunes de cada control de ingreso le permiten realizar la validación de los datos de ingreso contra un registro en otro archivo, sea cuando el control es seleccionado (justo antes de que el usuario ingrese datos) o cuando es aceptado (inmediatamente después del ingreso de datos).

2. En el grupo **When the Control is Accepted**, oprima el botón de tres puntos suspensivos (...) para la caja de ingreso **Lookup Key**.
3. Destaque el archivo *Pedidos* en la caja de diálogo **Select Key**, y oprima el botón **Insert**.
4. Destaque el archivo *Cientes* en la lista **Files**, y oprima el botón **Select**.

Estas últimas acciones añaden el archivo *Cientes* al árbol de archivos del procedimiento como una consulta automática desde el archivo *Pedidos*. Esto generará una “consulta” en el archivo vinculado *Cientes*, basado en la relación de archivos configurada en el diccionario.

5. Destaque *CLI:KeyNumCliente* en la caja de diálogo **Select Key**, y oprima el botón **Select**.

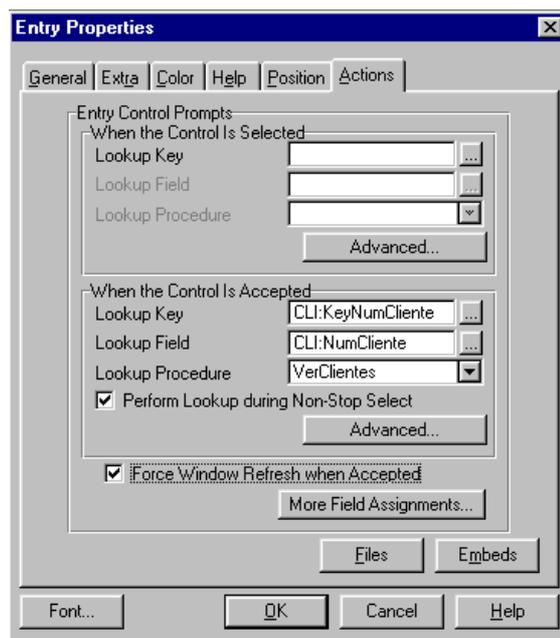
Esto indica la clave que se utilizará para intentar obtener un registro válido vinculado en el archivo de *Cientes* para el valor que el usuario ingrese en este control.

6. Oprima el botón de puntos suspensivos (...) de la caja **Lookup Field**.
7. Destaque *CLI:NumCliente* en la caja de diálogo **Select Component**, y oprima el botón **Select**.

Esto hace de *CLI:NumCliente* el campo que debe contener el valor correspondiente al que el usuario ingrese en el control.

8. Elija el procedimiento *VerClientes* de la lista desplegable **Lookup Procedure**.

Esto llama al procedimiento nombrado cuando el usuario ingresa un número de cliente inválido, para que pueda elegir de una lista de clientes.



9. Marque los casilleros **Perform Lookup during Non-Stop Select** (“Realizar consulta durante la revisión automática de la ficha”) y **Force Window Refresh when Accepted** (“Redibujar la pantalla al aceptar”), para asegurarse que los datos mostrados sean siempre válidos y actualizados.

10. Oprima el botón **Embeds** en la lengüeta **Actions**.

Esto muestra una lista de los puntos de inserción vinculados a este control, únicamente. Esta es la forma más rápida de llegar a los puntos de inserción de un control determinado, y es el segundo modo que hemos visto para llegar a un punto de inserción. Hay todavía un tercer método, que ya trataremos.

11. Destaque el evento **Selected** bajo **Control Event Handling**, y oprima el botón **Insert**.

El “evento seleccionado” ocurre justo antes de que el control obtenga foco. Este punto de inserción permite hacer cualquier “configuración específica” para este control.

12. Destaque **Source** y oprima el botón **Select**.

Esto invoca al editor de textos para que pueda escribir código Clarion. Adviértase que está presente la caja flotante de herramientas **Populate Field**. Siempre que da DOBLE CLIC sobre un campo en esta caja de herramientas, coloca el nombre del campo (incluyendo el prefijo) en su código en el punto de inserción. Esto no solamente ayuda a la productividad (menos tipeo), sino que asegura la inexistencia de errores tipográficos (que provocarían errores en la compilación).

13. Escriba el siguiente código:

```
?PED:NumCliente{PROP:Touched} = TRUE
```

Es parte del “comportamiento estándar de Windows” que, si el usuario no ingresa datos a un control y se limita a oprimir TAB(o da un CLIC con el ratón) para seguir a otro control, no se genera la aceptación de un evento (esto es muy diferente a como trabajan los programas de DOS). Así los clientes pueden pasar con TAB sobre el control sin provocar la validación de los datos en cada control. Sin embargo, a veces es necesario evitar este comportamiento estándar para asegurar la integridad de la base de datos.

La sentencia que ha escrito utiliza la sintaxis de asignación de propiedad de Clarion (véase el Apéndice C de *Language Reference*). Configurando PROP:Touched a TRUE (verdadero) en el evento *selected* de este control, siempre se genera un evento de aceptación, sea que el usuario haya ingresado datos o no. Esto fuerza la ejecución del código de consulta (ingresado en la lengüeta **Actions** de la página previa). Esto asegura que el usuario ingrese un código de cliente válido; de lo contrario, aparecerá la lista de clientes para permitir elegir un número válido.

14. Elija **Exit!** (y guarde) para regresar a la caja de diálogo **Embedded Source**.

15. Oprima el botón **Close** para regresar a la caja de diálogo **Entry Properties**, y oprima el botón **OK** para cerrarla.

Añadir un control “para visualización únicamente”

1. Elija **Control** ➤ **String**, u oprima el botón String en la caja flotante de herramientas **Controls** (el icono situado a la derecha de la “gran flecha” en la fila superior).
2. Dé CLIC a la derecha de la caja de ingreso de número de cliente que colocó anteriormente.
3. Dé CLIC DERECHO al control de cadena de texto que acaba de colocar, y elija **Properties** del menú contextual.

4. Marque la caja **Variable String**.
Esto especifica que el control mostrará datos provenientes de una variable, no solamente una cadena alfanumérica. Aparece la caja de diálogo **Select Field**.
5. Destaque el archivo *Cliente* en la lista **Files**, y elija *CLI:Empresa* de la lista de campos (**Fields**) y oprima el botón **Select**.
6. Oprima el botón **OK** y cierre la caja de diálogo **String Properties**.

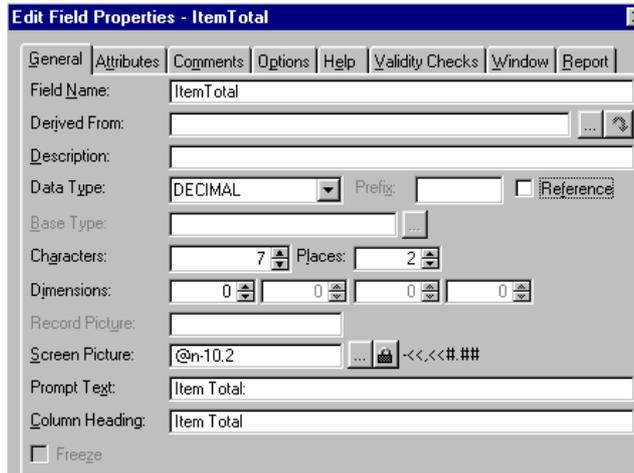
Colocación de las plantillas de control del archivo Detalle

El siguiente elemento clave en esta ventana es un control de listados (*browse*), sincronizado al Número de Pedido de esta ficha. Esto mostrará todos los registros del archivo Detalle vinculados al registro de *Pedidos*.

Adición de una lista Detalle

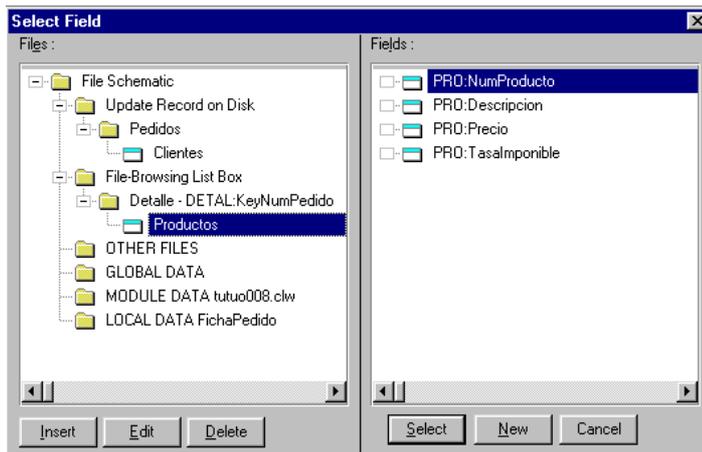
1. Elija **Populate** ➤ **Control Template** o dé CLIC sobre la herramienta Control Template en la caja flotante de herramientas **Controls** (último icono a la derecha, fila inferior).
2. Destaque *BrowseBox*, y oprima el botón **Select**.
3. Dé CLIC debajo de la caja de ingreso de datos de cliente.
4. Destaque el ítem “ToDo” ubicado debajo de **File-Browsing List Box** y oprima el botón **Insert**.
5. Seleccione el archivo *Detalle* de la caja de diálogo **Insert File**, y oprima el botón **Insert**.
6. Oprima el botón **Edit**.
7. Destaque *keyNumPedido* en la caja de diálogo **Change Access Key**, y oprima el botón **Select**.
8. Destaque *DETAL:NumProducto* en la en la lista de campos (**Fields**), y oprima el botón **Select**.
9. Seleccione *Center* en la lista desplegable para la justificación del dato (**Data group**).
10. Presione el botón **New Column**.
11. Destaque *DETAL:Cantidad* en la en la lista de campos (**Fields**), y oprima el botón **Select**.
12. Seleccione *Center* en la lista desplegable para la justificación del dato.
13. Presione el botón **New Column**.
14. Destaque *DETAL:ProdPrecio* en la en la lista de campos (**Fields**), y oprima el botón **Select**.
15. Seleccione *Center* en la lista desplegable para la justificación del dato.
16. Presione el botón **New Column**.
17. Destaque *LOCAL DATA Ficha Pedido* en la lista **Files**, y oprima el botón “Nuevo” (**New**).
Este botón **New** habilita la adición de una variable local sin necesidad de ir a la ventana de **Procedure Properties** y oprima el botón **Data**. Usaremos esta nueva variable para mostrar el precio total para cada ítem (la cantidad multiplicada por el precio unitario).
18. Tipee *ItemTotal* en el campo **Name**.

19. Seleccione *DECIMAL* de la lista desplegable **Type**.
20. Escriba 7 como longitud del campo (**Characters**).
21. Escriba 2 para la cantidad de decimales (**Places**) y oprima el botón **OK**.



22. Seleccione *Center* en la lista desplegable para la justificación del dato.
23. Presione el botón **New Column**.
24. Destaque el archivo *Detalle* ubicado debajo de **file-Browsing List Box** y oprima el botón **Insert**.
25. Seleccione el archivo *Productos* de la caja de diálogo **Insert File**, y oprima el botón **Select**.

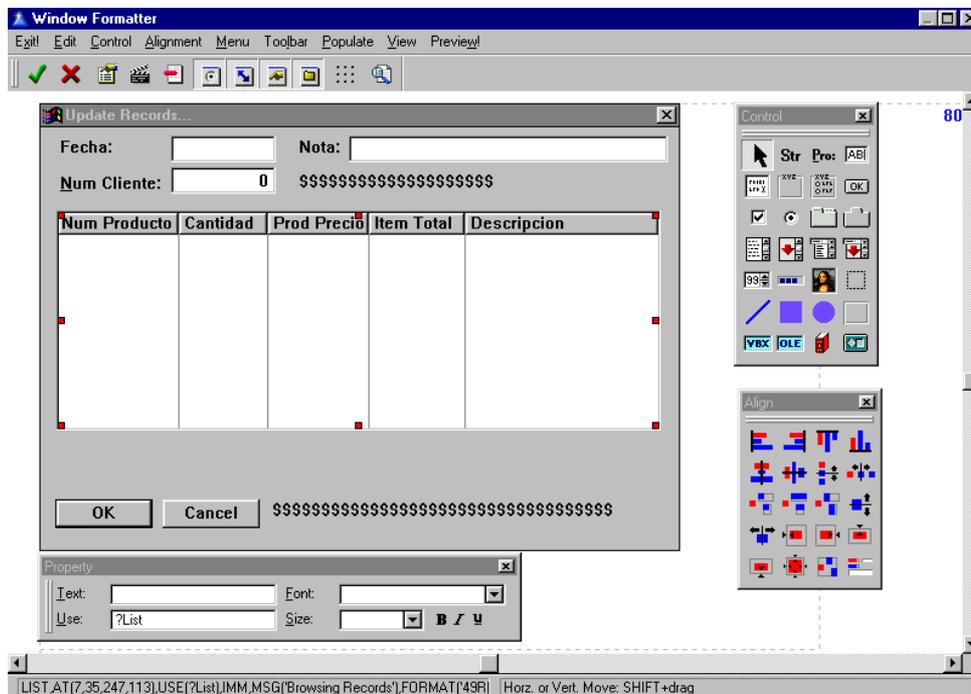
Esto añade el archivo *Productos* al árbol de archivos del template de control como un archivo de consulta. El registro vinculado se busca automáticamente, para que pueda verse la descripción del producto en la lista.



26. Destaque *PROD:Descripcion* en la lista **Fields**, y oprima el botón **Select**.
27. Seleccione *Left* en la lista desplegable para la justificación del dato.
28. Modifique el tamaño de las columnas y ajuste el diseño general (como ya lo ha hecho antes).
29. Oprima el botón **OK** para cerrar el Diseñador de Listados.

30. Ajuste el tamaño de la caja para mostrar todos los campos cargados.

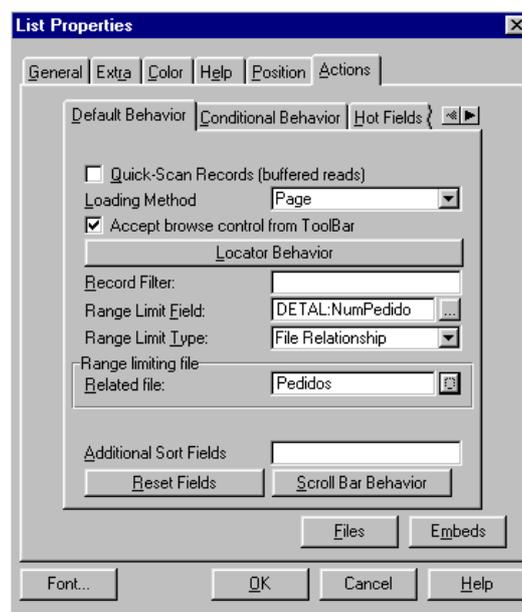
Ahora debería aparecer aproximadamente así:



Sincronización de las cajas de listado

Queremos que esta lista muestre solamente los registros que están vinculados al registro de Cliente que es sobre el que se está trabajando. Por lo tanto, debemos especificar un límite de búsqueda (*range limit*).

1. Dé CLIC DERECHO a la caja de listado que acaba de colocar, y elija **Actions** del menú.
2. Oprima el botón con puntos suspensivos (...) en **Range Limit Field**.
3. Destaque el campo *DETAL:NumPedido* en la lista **Components**, y oprima el botón **Select**.
4. Elija "Relación de archivos" (**File Relationship**) de la lista desplegable **Range Limit Type**.
5. Escriba *Pedidos* en el campo **Related File**.



Adición de un cálculo del monto total del pedido

Ahora deseamos calcular el total del pedido que luego guardaremos en el archivo de Pedidos.

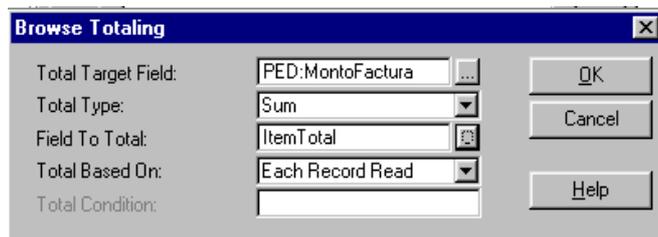
1. Oprima el botón pequeño de flecha, ubicado a la derecha de la lengüeta **Hot Fields** para desplazarse por las lengüetas, y elija la lengüeta **Totaling** cuando aparezca.
2. Oprima el botón **Insert**.
3. Oprima el botón de puntos suspensivos del campo **Total Target Field**.
4. Destaque el archivo *Pedidos* en la lista de archivos (**Files**), seleccione *PED:MontoFactura*, y oprima el botón **Select**.

Este es el campo que recibirá el resultado del cálculo total.

5. Elija “Suma” (**Sum**) de la lista desplegable **Total Type**.
6. Oprima el botón de puntos suspensivos (...) del campo a totalizar (**Field to Total**).
7. Destaque la opción *LOCAL DATA FichaPedidos* en la lista de archivos (**Files**), seleccione *ItemTotal* de la lista **Fields**, y oprima el botón **Select**.

Este es el campo cuyo contenido se usará en el cálculo total. Hasta ahora, solamente hemos declarado este campo y no hemos hecho nada para introducirle valor alguno, pero pronto lo haremos.

8. Elija “Para cada registro leído” (*Each Record Read*) de la lista desplegable **Total Based On**.



9. Oprima el botón **OK**, para cerrar la caja de diálogo **Browse Totaling**.

Añadir barras de desplazamiento horizontal y vertical

1. Elegir la lengüeta **Extra**.
2. Marcar los casilleros **Horizontal y Vertical**.
3. Oprima **OK** para cerrar la caja de diálogo **List Properties**.

Adición de los botones de actualización de archivo

1. Elija **Populate** > **Control Template** o dé CLIC sobre la herramienta *Control Template* en la caja de herramientas **Controls** (último icono a la derecha, fila inferior).
2. En la caja de diálogo **Select Control Template**, elija la plantilla de control *BrowseUpdateButtons* y oprima el botón **Select**.

El cursor cambia a una cruz y un “librito”.

3. Dé CLIC debajo del listado.

Aparecen juntos los botones de **Insert**, **Change** y **Delete**.

4. Dé CLIC DERECHO sobre el botón **Delete**, y elija **Actions** del menú contextual.

5. Marque el casillero de “Modificación in situ” (**Use Edit in Place**).

Basta con indicar esta opción en un solo botón, para que la plantilla de control lo marque en los tres. Utilizaremos la técnica de “modificación in situ” (*edit-in-place*) para actualizar los registros de Detalle, en lugar de un formulario. Esto nos permitirá demostrar algunas técnicas de programación bastante avanzadas y ver qué fáciles son de realizar dentro del Generador de Aplicaciones.

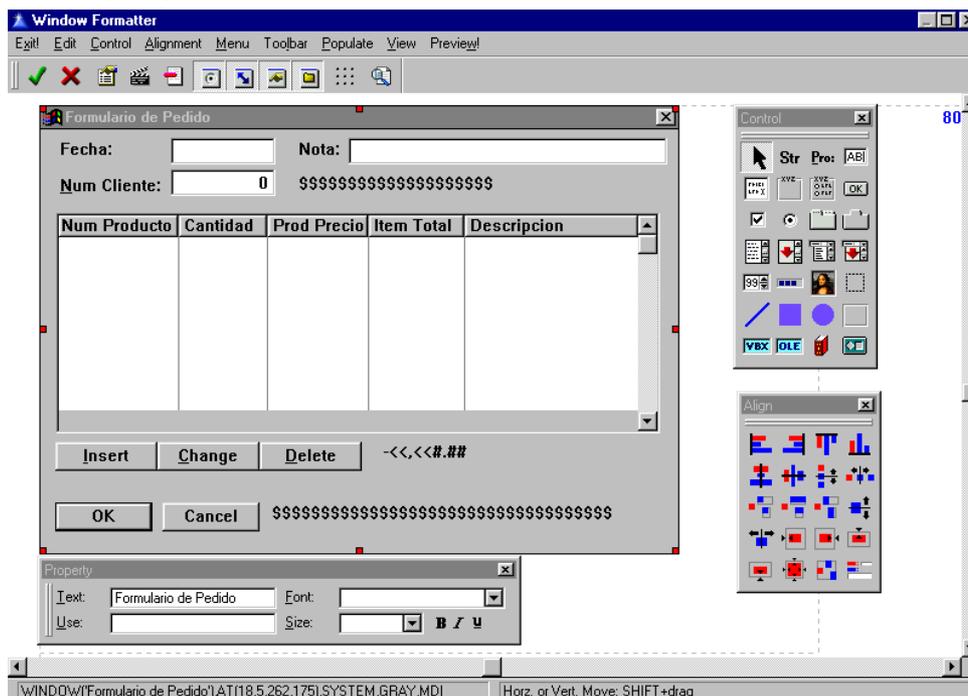
6. Oprima el botón **OK**.

Adición de un control de solo visualización

1. Elija **Control** > **String**, o dé CLIC sobre la herramienta *String* en la caja de herramientas **Controls**.
2. Dé CLIC debajo del ángulo inferior derecho de la caja de listados.
3. Dé CLIC DERECHO sobre el control que acaba de colocar, y elija **Properties** del menú contextual.
4. Marque el casillero **Variable String**.
Esto especifica que el control mostrará datos provenientes de una variable, no una mera cadena alfanumérica fija. Aparece automáticamente la caja de diálogo **Select Field**.
5. Marque el archivo *Pedidos* en la lista **Files**, y elija *PED:MontoFactura* de la lista de campos (**Field**) y oprima el botón **Select**.
6. Oprima el botón **OK**.

Cambio del título de la ventana y salida del Diseñador

1. Dé CLIC sobre la barra de título de la ventana sobre la que está trabajando.
2. Escriba *Formulario de pedido* en el campo de texto (**Text**) en la caja de herramienta flotante **Property**, y oprima TAB.



3. Elija **Exit!** para cerrar el Diseñador de Ventanas.

Hacerlo funcionar

Hay un par de cosas que debemos hacer para que este procedimiento adquiera plena funcionalidad: añadir una fórmula, y configurar la “modificación in situ”.

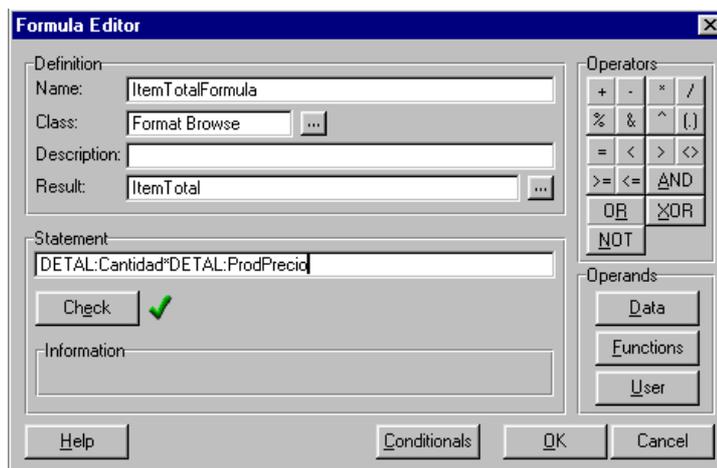
Uso del Editor de Fórmulas

Para hacer que *ItemTotal* calcule el monto de cada registro de Detalle en la caja de listados, es necesario añadir una fórmula al procedimiento. Esto permitirá también realizar la suma total en PED:MontoFactura.

1. Oprima el botón **Formulas** en la caja de diálogo **Procedure Properties**.
Aparece la caja de diálogo **Formulas**. Aquí se listan todas las fórmulas del procedimiento.
2. Oprima el botón **New** para añadir una nueva fórmula.
Aparece la caja diálogo **Formula Editor**.
3. Escriba *Item Total Formula* en el campo **Name**.
4. Oprima el botón de puntos suspensivos (...) en el campo **Class**.
5. Destaque *Format Browse* en la lista **Template Classes**, y luego oprima el botón **OK**.
El campo **Class** simplemente especifica la posición lógica dentro del código fuente generado en que se calcula fórmula. La clase *Format Browse* indica a la plantilla de control *BrowseBox* que efectúe el cálculo cada vez que formatea un registro para mostrarlo en la caja de listado.
6. Oprima el botón de puntos suspensivos (...) en el campo **Result**.
7. Destaque *LOCAL DATA FichaPedidos* en la lista **Files**, seleccione *ItemTotal* de la lista **Fields**, y oprima el botón **Select**.
Esto da nombre al campo que recibirá el resultado del cálculo. Este es el campo que definimos anteriormente en el Diseñador de Listados.
8. Oprima el botón **Data** en el grupo de **operandos** (Operands).
9. Destaque el archivo Detalle en la lista **Files**, seleccione *DETAL:Cantidad* de la lista **Fields**, y oprima el botón **Select**.
Esto lo coloca automáticamente en el campo **Statement**. Este campo contiene la expresión que se construye. Si se desea, también se puede escribir directamente sobre él para construir la expresión.
10. Oprima el botón ***** en el grupo de operadores (**Operators**).
Este es el operador de multiplicación.
11. Oprima el botón **Data** en el grupo **Operands**.
12. Destaque el archivo *Detalle* en la lista **Files**, seleccione *DETAL:ProdPrecio* de la lista **Fields**, y oprima el botón **Select**.

13. Oprima el botón de comprobación (**Check**) para verificar la sintaxis de la expresión.

Aparece una tilde ✓ verde a la izquierda de la sentencia, indicando que la sintaxis es correcta. Si aparece una “X” roja, la sintaxis es incorrecta, y aparece destacada la porción de la sentencia que debe cambiarse.



14. Oprima el botón **OK** para cerrar el Editor de Fórmulas.

Reaparece la caja de diálogo **Formulas**.

15. Oprima el botón **OK** para cerrarla.

Configuración de la “modificación in situ”

Ahora debemos configurar las características de la “modificación in situ” (*Edit in Place*). Ya hemos usado esta característica en el archivo de Teléfonos, y sencillamente aceptamos el funcionamiento por omisión, debido a que se trataba de un archivo muy simple. Sin embargo, ahora estamos trabajando sobre una línea del registro de Detalle de un sistema de ingreso de pedidos, lo que significa que necesitamos validar los datos ingresados por el operador. Para hacerlo, necesitamos extender la funcionalidad simple que ofrece la Biblioteca de clases ABC.

Configurar clases específicas para las columnas

1. Oprima el botón **Extensions** en la caja de diálogo **Procedure Properties**.
2. Destaque *Update a Record form Browse Box on Detalle* y oprima el botón **Configure Edit in place**.

Aparece la caja de diálogo correspondiente. Las opciones de esta caja le permite especificar el comportamiento del control cuando el usuario ingresa o modifica datos y oprime ENTER o una tecla de flecha, junto con varias opciones de “guardar”. Aceptaremos todos los valores preconfigurados.

3. Oprima el botón **Column Specific**.
4. Oprima el botón **Insert**.
5. Oprima el botón de puntos suspensivos (...) del campo **Field**.

6. Destaque *DETAL:NumProducto*, y oprima el botón **Select**.

La biblioteca ABC contiene una clase de objetos llamada *EditEntryClass* que es la clase por omisión de *Edit in Place*. "Pasaremos por alto" algunos de los métodos de la clase, para esta columna, para poder modificar el comportamiento por omisión. La adición de este campo a la lista de campos **Column Specific** es lo que nos permite "hacer a un lado" los métodos preconfigurados para este caso. Lo haremos para poder realizar la validación de ingreso de datos: queremos asegurarnos que los usuarios solamente puedan ingresar números de producto válidos.

7. Oprima el botón **OK**.
8. Oprima el botón **Insert**.
9. Oprima el botón de puntos suspensivos (...) del campo **Field**.
10. Destaque *DETAL:Cantidad*, y oprima el botón **Select**.

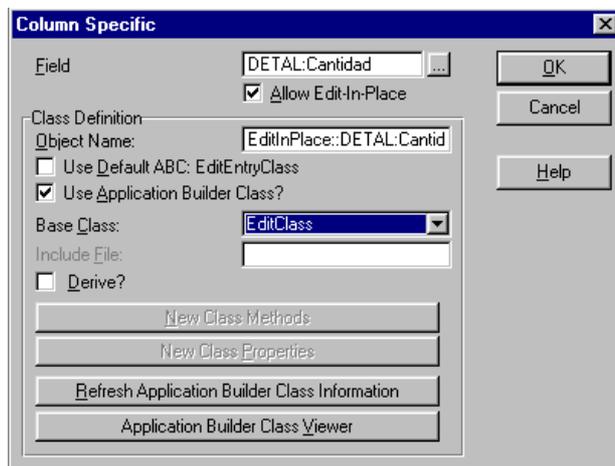
La clase *EditEntryClass* solamente ofrece controles de ingreso de datos (ENTRY). Para este campo, deseamos utilizar un control SPIN, de manera que el usuario lo haga girar hasta la cantidad que desea. Por lo tanto, necesitamos pasar por alto algunos de los métodos de la clase preconfigurada también para esta columna, para tener un control SPIN en lugar de un control ENTRY.

11. Limpie la casilla **Use Default ABC**.

Esto le permite indicar la clase exacta a partir de la que derivará.

12. Seleccione *EditClass* de la lista desplegable **Base Class**.

La *EditEntryClass* hace mucho más de lo que necesitamos y de todas maneras vamos a pasar por alto los métodos. Entonces vamos a derivar y sobre-escribir la clase a partir de la cual se derivan todas las clases *Edit in Place*: la *EditClass*.



13. Oprima el botón **OK**.
14. Oprima el botón **Insert**.
15. Oprima el botón de puntos suspensivos (...) para campo **Field**.
16. Destaque *DETAL:ProdPrecio*, y oprima el botón **Select**.
17. Quite la marca del casillero que permite la modificación in situ (**Allow Edit in Place**).

Esto evita que esta columna pueda modificarse.

Solamente los campos del archivo primario del *Browse* pueden modificarse in situ y la opción preconfigurada es que *todos* los archivos primarios sean modificables. En este caso, significa que se pueden modificar los campos de *Detalle*, pero no los de *Productos*. Sin embargo, en este caso no queremos que el usuario pueda modificar el precio, porque lo obtendremos directamente del archivo de *Productos*, y no deseamos que se lo cambie. Es por esto que inhabilitaremos el casillero **Allow Edit in Place**.

18. Oprima el botón **OK** cinco veces para volver al árbol de la aplicación.

Utilización del Editor de Puntos de Inserción (Embeditor)

Dé CLIC DERECHO sobre *FichaPedido* y elija **Source** del menú contextual.

Esta opción del menú abre el Editor de Puntos de Inserción : el tercer método de acceder a esos puntos. Se trata del mismo editor de textos que ya ha utilizado, pero abierto en un modo especial que le permite no solamente modificar los puntos de inserción en el procedimiento, sino también hacerlo dentro del contexto del código generado por las plantillas.

El código fuente se verá así :

```

FichaPedido PROCEDURE
? Start of "Data for the procedure"
? [Priority 1300]

FilesOpened      BYTE
ActionMessage    CSTRING(40)
ItemTotal        DECIMAL(7,2)
? [Priority 3000]

? Views & Queues
BRW5::View:Browse  VIEW(Detalle)
                   PROJECT(DETAL:NumProducto)
                   PROJECT(DETAL:Cantidad)
                   PROJECT(DETAL:ProdPrecio)
                   PROJECT(DETAL:NumPedido)
                   JOIN(PRO:KeyNumProducto,DETAL:NumProducto)
                   PROJECT(PRO:Descripcion)
                   PROJECT(PRO:NumProducto)
                   END
                   END

Queue:Browse      QUEUE
DETAL:NumProducto LIKE(DETAL:NumProducto)
DETAL:Cantidad    LIKE(DETAL:Cantidad)
DETAL:ProdPrecio  LIKE(DETAL:ProdPrecio)
ItemTotal        LIKE(ItemTotal)
PRO:Descripcion  LIKE(PRO:Descripcion)
DETAL:NumPedido  LIKE(DETAL:NumPedido)
PRO:NumProducto  LIKE(PRO:NumProducto)
Mark             BYTE
ViewPosition     STRING(1024)
  
```

!Generated from procedure template - Window

Populate Field

Detalle - Detalles (líneas) de

KeyNumProducto (KEY)

KeyNumPedido (KEY)

NumPedido (SHORT)

NumProducto (SHORT)

Cantidad (SHORT)

ProdPrecio (DECIMAL)

TasalImponible (DECIMAL)

Line: 1 Col: 1 Insert

Adviértase que la mayor parte del código se ubica sobre un fondo gris, y que los puntos en que usted escribió código tienen fondo blanco. Hay además comentarios identificatorios para cada punto de inserción. Se pueden ver o no esos comentarios según sea su opción en **Setup > Application Options**. Una vez que se hayan familiarizado con ellos, posiblemente desee quitarlos para ver más del código propiamente dicho.

Usted advertirá que apareció brevemente un mensaje : "Generating tutorial".

Este editor muestra *todos los puntos de inserción posibles que podrían generarse* para el procedimiento. Adviértase la distinción: **el editor no muestra el código que será generado, sino todo el código que podría ser generado, si usted eligiese cada opción posible y colocara código en cada punto disponible.** Es muy improbable que usted alguna vez haga esto. Por lo tanto, se muestra muchísimo más código en este editor que cuando se ha compilado la aplicación. Una vez que terminemos aquí, daremos una mirada al código generado para apreciar la diferencia.

Al lado derecho de la barra de herramientas hay cuatro botones que son muy importantes para conocer cuando se trabaja en este editor. Son (de izquierda a derecha): **Punto de inserción anterior, Punto siguiente, Punto utilizado anterior, y Punto utilizado siguiente** (si detiene el puntero del *mouse* sobre ellos verá las acotaciones en inglés para cada uno de los botones). Permiten ir rápidamente de un punto de inserción a otro, especialmente cuando ya ha escrito código en algunos.

Detección de Pedidos modificados

Una de las cosas que deseamos que haga este procedimiento es detectar cambios a los pedidos existentes, y asegurarse que no se provoquen desajustes de datos entre los archivos *Pedidos* y *Detalle*. Este sistema guarda el monto total de un pedido en el campo PED:MontoFactura, de manera que cuando el usuario cambia un ítem del Detalle en un Pedido existente, debe hacerse que el registro se actualice. Hay una forma muy simple de hacerlo, lo que nos permitirá demostrar el flexible manejo de errores que realiza la biblioteca ABC.

1. Dé CLIC sobre el botón **Punto de inserción siguiente** (unas cinco veces) hasta que llegue al punto de inserción inmediatamente anterior a la línea de código en la que se lee ThisWindow CLASS(WindowManager), (esto debería ser en **[Priority 7500]**).

Cada punto de inserción tiene potencialmente 10.000 niveles de prioridad dentro. Este sistema de prioridades está diseñado para permitir insertar el código antes o después del código generado (sea mediante los Templates ABC de Clarion o de terceras partes). Esto hace que el sistema de inserción sea completamente flexible, permitiéndole añadir su código en cualquier punto lógico que se necesite, antes o después de cualquier "trozo" de código generado.

2. Escriba el código siguiente:

```
LocalErrGroup GROUP                !Esta línea debe ir exactamente sobre la columna1
                                     !(aparecerá en color rojo), ya que se trata de una
                                     !etiqueta de procedimiento.
                                     USHORT (1)
                                     USHORT(99)
                                     BYTE (Level:Notify)
                                     PSTRING( 'Guardar el pedido')
                                     PSTRING('Ha cambiado un ítem. Oprima el botón OK')
                                     END
SaveTotal LIKE (PED:MontoFactura)
```

Los templates ABC de Clarion generan código orientado a objetos utilizando la biblioteca ABC. Esta biblioteca contiene una clase de manejo de errores llamada ErrorClass. La porción de código que hemos escrito declara un **LocalErrGroup GROUP** (exactamente en la manera que requiere la ErrorClass: véase el Application Handbook) que contiene un número de error "personalizado" y un mensaje que estamos definiendo para el uso del objeto ErrorClass de nuestra aplicación. La declaración **SaveTotal** es una variable local que se define COMO ('LIKE', siempre con el mismo tipo de datos) el campo *PED:MontoFactura*. Utilizaremos esta variable para guardar el total del pedido cuando el usuario actualice un pedido existente.

```

! [Priority 7500]
LocalErrGroup GROUP
    USHORT(1)
    USHORT(99)
    BYTE(Level:Notify)
    PSTRING('Guardar Pedido')
    PSTRING('Ha cambiado un ítem. Oprima el botón OK')
END
SaveTotal LIKE(PED:MontoFactura)
! End of "Data for the procedure"
ThisWindow CLASS(WindowManager)
AddHistoryField PROCEDURE(SIGNED Control,SIGNED FieldNumber)
AddHistoryFile PROCEDURE(*GROUP RecBuf,*GROUP HistBuf)
AddItem PROCEDURE(BrowseClass BC)
AddItem PROCEDURE(FileDropClass FD)
AddItem PROCEDURE(SIGNED Control,BYTE Action)
AddItem PROCEDURE(ToolBarClass TC)
AddItem PROCEDURE(ToolBarUpdateClass TF)
AddItem PROCEDURE(TranslatorClass Translator)
AddItem PROCEDURE(WindowResizeClass RC)
AddUpdateFile PROCEDURE(FileManager FM)
Ask PROCEDURE(),DERIVED
Init PROCEDURE(),BYTE,PROC,DERIVED
Kill PROCEDURE(),BYTE,PROC,DERIVED
Open PROCEDURE(),DERIVED
PostCompleted PROCEDURE()
PrimeFields PROCEDURE(),PROC,DERIVED
PrimeUpdate PROCEDURE(),BYTE,PROC,DERIVED
Reset PROCEDURE(BYTE Force=0),DERIVED
RestoreField PROCEDURE(SIGNED Control),DERIVED

```

Populate Field

Detalle - Detalles (líneas) de

- KeyNumProducto (KEY)
- KeyNumPedido (KEY)
- NumPedido (SHORT)
- NumProducto (SHORT)
- Cantidad (SHORT)
- ProdPrecio (DECIMAL)
- Tasalmpoible (DECIMAL)

Embed Data for the procedure [7500] Line: 66 Col: 39 Insert Modified

- Elija **Search** ➤ **Find** para traer la caja de diálogo **Find**.
- Escriba *ThisWindow.Init* en el campo **Find what**, y oprima el botón **Find next**.
Esto nos lleva directamente al método *ThisWindow.Init*
- En el punto embebido inmediatamente siguiente a la línea `SELF.Errors &= GlobalErrors` (debería ser **[Priority 5600]**), escriba el siguiente código:

```

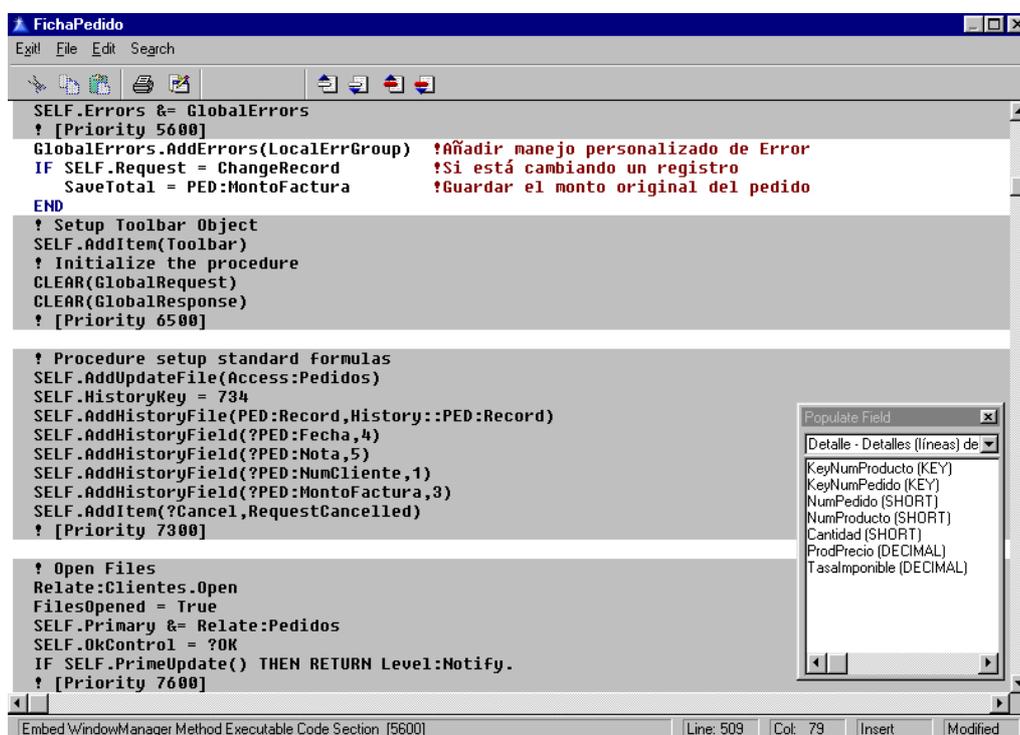
GlobalErrors.AddErrors(LocalErrGroup) !Añadir manejo personalizado del error
IF SELF.Request = ChangeRecord !Si se está cambiando un registro
    SaveTotal = PED:MontoFactura !Guardar el monto original del pedido
END

```

Este código llama al Método **AddErrors** del objeto **GlobalErrors** para añadir el **LocalErrGroup** a la lista de errores disponibles que maneja el objeto. El objeto **GlobalErrors** es una instancia de la *ErrorClass* que los templates ABC declaran globalmente para manejar todas las condiciones de error en la aplicación. La adición de nuestro **LocalErrGroup** habilita al objeto **GlobalErrors** a manejar nuestra condición "personalizada" de error. Esto demuestra la flexibilidad de la biblioteca ABC de Clarion. La sentencia **IF** detecta cuando el usuario está modificando un pedido existente y guarda el total original.

Este punto de inserción se ubica en **ThisWindow.Init PROCEDURE** que realiza algunas tareas necesarias de inicialización. Este es un método virtual del objeto **ThisWindow**. Este es el objeto que dirige todo el código de ventanas y de manejo de controles.

Quizás usted no lo haya percibido, pero las plantillas ABC generan exactamente una línea de código fuente ejecutable dentro del **FichaPedido PROCEDURE** en sí (`GlobalResponse = ThisWindow.Run`) de manera que *toda* la funcionalidad del **FichaPedido PROCEDURE** se efectúa en métodos objeto; sean métodos virtuales específicos al **FichaPedido PROCEDURE** mismo o a métodos estándar de la biblioteca ABC. Esto se aplica a cada procedimiento generado por los templates ABC. La generación de código plenamente Orientado a Objetos hace que éste sea muy compacto y eficiente: sólo el código que necesariamente debe ser diferente en un procedimiento individual se maneja diferentemente. Todo lo demás es código estándar que reside en solamente un lugar, y que ha sido comprobado y depurado para asegurar un funcionamiento coherente y confiable.



```

SELF.Errors &= GlobalErrors
! [Priority 5600]
GlobalErrors.AddErrors(LocalErrGroup) !Añadir manejo personalizado de Error
IF SELF.Request = ChangeRecord !Si está cambiando un registro
SaveTotal = PED:MontoFactura !Guardar el monto original del pedido
END
! Setup Toolbar Object
SELF.AddItem(Toolbar)
! Initialize the procedure
CLEAR(GlobalRequest)
CLEAR(GlobalResponse)
! [Priority 6500]

! Procedure setup standard formulas
SELF.AddUpdateFile(Access:Pedidos)
SELF.HistoryKey = 734
SELF.AddHistoryFile(PED:Record,History::PED:Record)
SELF.AddHistoryField(?PED:Fecha,4)
SELF.AddHistoryField(?PED:Nota,5)
SELF.AddHistoryField(?PED:NumCliente,1)
SELF.AddHistoryField(?PED:MontoFactura,3)
SELF.AddItem(?Cancel,RequestCancelled)
! [Priority 7300]

! Open Files
Relate:Clientes.Open
FilesOpened = True
SELF.Primary &= Relate:Pedidos
SELF.OkControl = ?OK
IF SELF.PrimeUpdate() THEN RETURN Level:Notify.
! [Priority 7600]

```

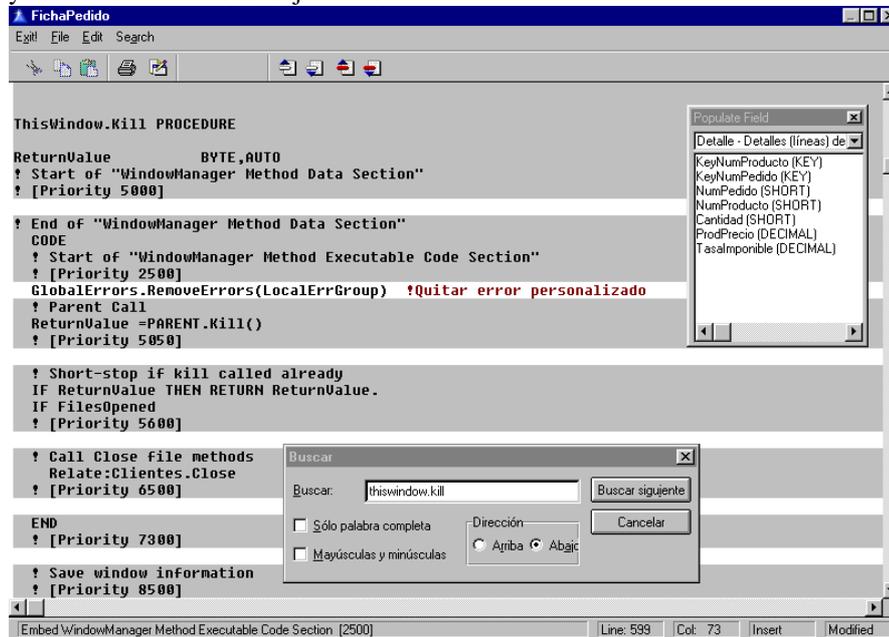
La Programación Orientada a Objetos (POO) en Clarion comienza con la estructura CLASS. Vea CLASS en *Language Reference* para el tratamiento de la sintaxis POO. La guía del programador (*Programmer's Guide*) contiene varios artículos que tratan la POO en profundidad, y hay una documentación completa de la Biblioteca ABC de Clarion.

6. Escriba `ThisWindow.Kill` en el campo "¿Encontrar qué ?" (**Find what**), y oprima el botón "Encontrar el próximo" (**Find next**) .
7. En el punto embebido inmediatamente siguiente a la línea CODE (debería ser en **[Priority 2500]**), escriba el siguiente código:

```
GlobalErrors.RemoveErrors(LocalErrGroup) !Quitar error personalizado
```

Esto llama al método de la Biblioteca ABC para quitar nuestro error "personalizado". El método **ThisWindow.Kill** es un procedimiento de "limpieza" (realiza tareas a la salida) que se

ejecuta cuando el usuario termina de trabajar en el procedimiento FichaPedido, de manera que ya no se necesita el manejo de error.

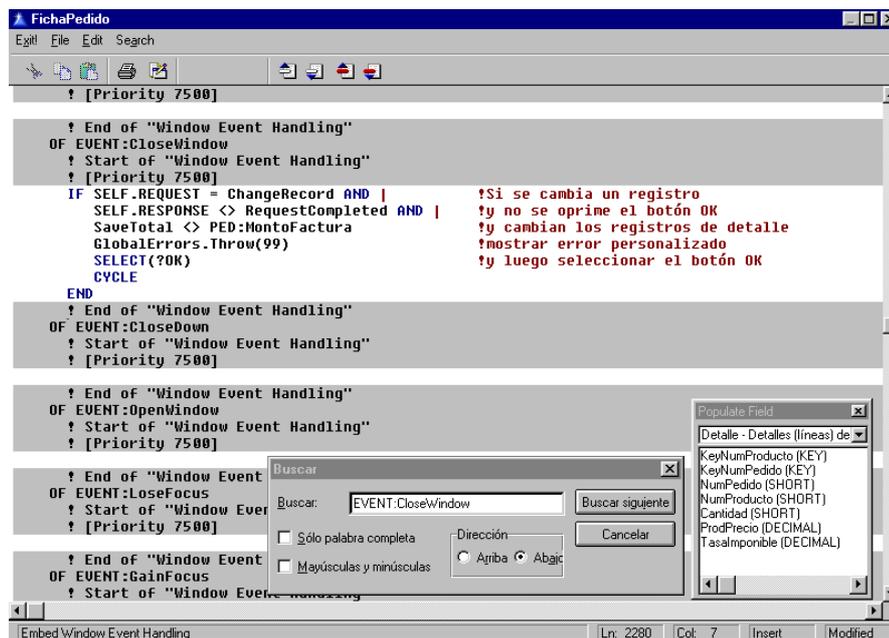


8. Escriba *EVENT:CloseWindow* en el campo **Find what**, y oprima el botón **Find next**.
9. En el punto embebido inmediatamente siguiente a *EVENT:CloseWindow* (debe ser en **[Priority 2500]**), escriba:

```

IF SELF.REQUEST = ChangeRecord AND |           !Si se cambia un registro
  SELF.RESPONSE <> RequestCompleted AND | !y no se oprime el botón OK
  SaveTotal <> PED:MontoFactura           !y cambian los registros de detalle
  GlobalErrors.Throw(99)                 !mostrar "error personalizado"
  SELECT(?OK)                            !y luego seleccionar el botón OK
  CYCLE
END

```



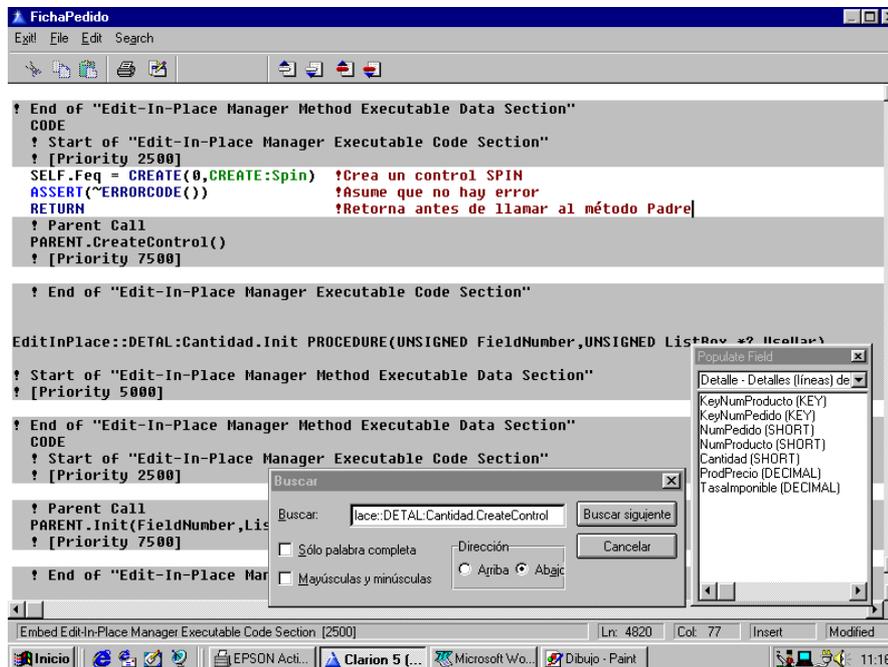
Este código detectará cualquier intento del usuario de salir del procedimiento sin guardar el registro de *Pedidos* después de hacer cambios a un pedido existente. Advierta la barra vertical(|) al final de las primeras líneas de código. Son absolutamente necesarias. La barra vertical es el carácter de continuación del lenguaje Clarion. Esto significa que las tres primeras líneas de continuación de este código forman una única sentencia lógica, que evalúa tres condiciones separadas y solamente ejecutará la sentencia **GlobalErrors.Throw(99)** si las tres condiciones son verdaderas.

Pasar por alto las clases de "Modificación in situ"

Bien, acabamos de ver un ejemplo de cómo usted puede *usar* la biblioteca ABC en su propio código insertado. Ahora le mostraremos cómo *pasar por alto* una clase para proveer funcionalidad especial, no provista por la biblioteca ABC. Las declaraciones CLASS para los objetos que hemos nombrado a través de la caja de diálogo **Configure Edit in Place** son generados para usted por los templates ABC. Estas clases se derivan de la clase **EditClass** de la biblioteca ABC.

1. Escriba `EditInPlace::DETAL:Cantidad.CreateControl` en el campo **Find what**, y oprima el botón **Find next**.
2. En el punto embebido inmediatamente siguiente la a línea CODE (esto debería ser [**Priority 2500**]), escriba el siguiente código:

```
SELF.Feq = CREATE (0, CREATE:Spin) !Crea un control SPIN
ASSERT(~ERRORCODE()) !Asume que no hay error
RETURN !Retorna antes de llamar al método padre
```

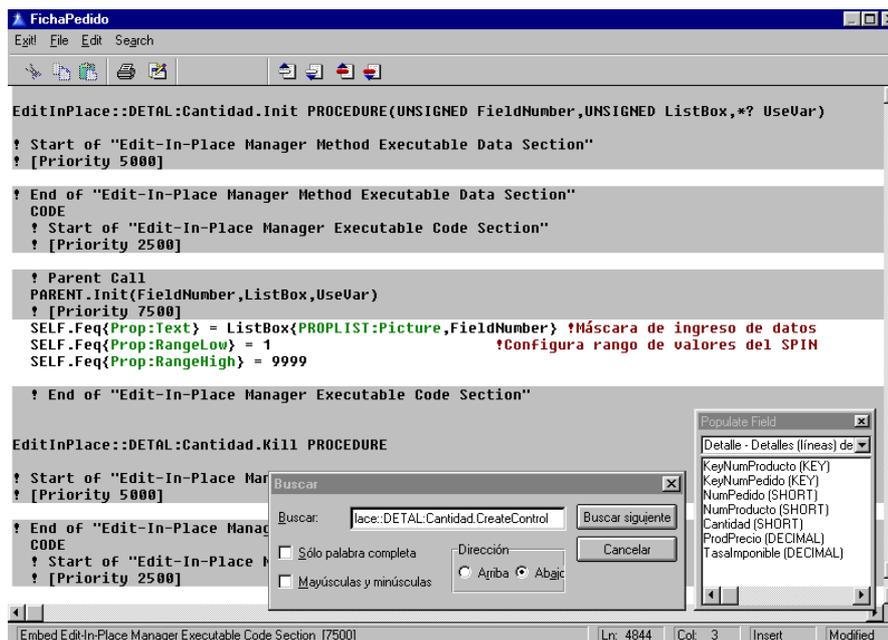


Este código crea un control SPIN y asume que no habrá errores. La sentencia RETURN es necesaria ya que necesitamos estar seguros que no ocurrirá la llamada al método padre.

3. Desplácese hasta el método `EditInPlace::DETAL:Cantidad.Init` (éste debería estar inmediatamente debajo del método `EditInPlace::DETAL:Cantidad.CreateControl`).

4. En el punto embebido inmediatamente siguiente a la línea `PARENT.Init(FieldNumber,ListBox,UseVar)`, (que debería ser [**Priority 7500**], escriba el siguiente código:

```
SELF.Feq{PROP:Text} = ListBox{PROPLIST:Picture,FieldNumber}           !Máscara de ingreso de datos
SELF.Feq{PROP:RangeLow} = 1                                           !Configurara el rango de valores
SELF.Feq{PROP:RangeHigh} = 9999                                       !del SPIN
```



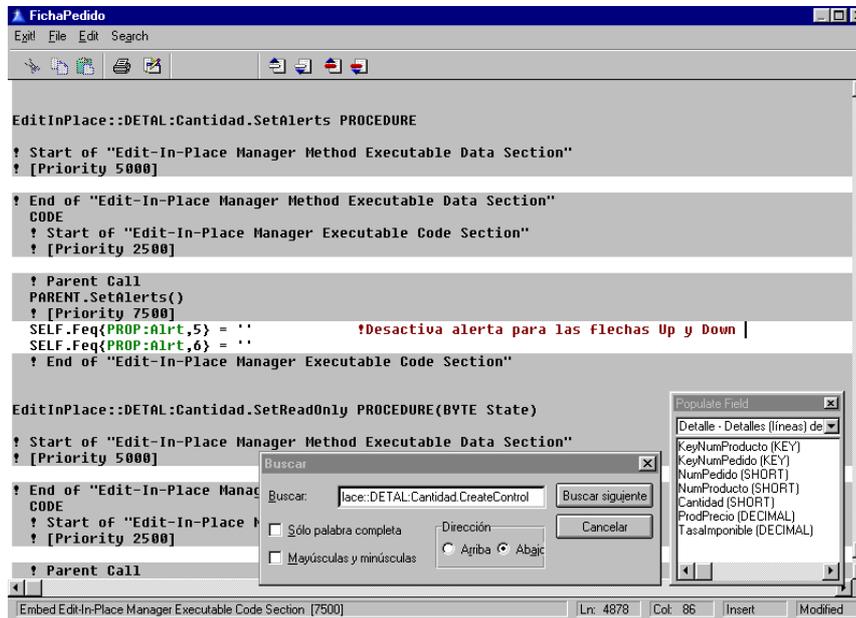
Este código establece la máscara de ingreso de datos y el rango de valores válido para el control SPIN.

5. Desplácese hasta el método `EditInPlace::DETAL:Cantidad.SetAlerts` (éste debería estar inmediatamente debajo del método `EditInPlace::DETAL:Cantidad.Kill`).
6. En el punto embebido inmediatamente siguiente a la línea `PARENT.SetAlerts()` (esto debería ser [**Priority 7500**], escriba el siguiente código:

```
SELF.Feq{PROP:Alt,5} = "
SELF.Feq{PROP:Alt,6} = "
```

Este código “desalerta” dos de las teclas estándar que alerta el método **SetAlerts** de la `EditClass`, que son las flechas Up y Down. Necesitamos hacer esto porque el control SPIN las emplea para operar, aumentando y disminuyendo el valor del campo respectivamente, cuando son presionadas.

Para ver el código que vamos a sobre-escribir, busque la `EditClass` en el archivo `\CLARION5\LIBSRC\ABEIP.CLW`.



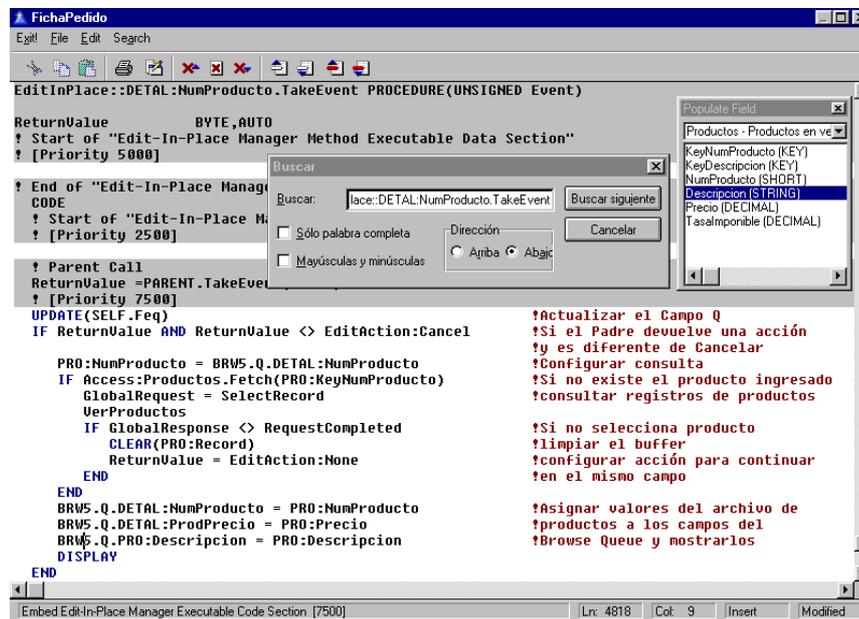
7. Escriba `EditInPlace::DETAL:NumProducto.TakeEvent` en el campo **Find what**, y oprima el botón **Find next**.
8. En el punto embebido inmediatamente siguiente la a línea de código que dice `ReturnValue = Parent.TakeEvent()` (esto debería ser [**Priority 7500**], escriba lo siguiente:

```

UPDATE(SELF.Feq)                ! Actualizar el campo Q
IF ReturnValue and ReturnValue <> EditAction:Cancel ! Si el Padre devuelve una acción
                                     ! y es diferente de Cancelar
    PRO:NumProducto = BRW5.Q.DETAL:NumProducto ! Configurar consulta
    IF Access:Productos.Fetch(PRO:KeyNumProducto) ! Si no existe el producto ingresado
        GlobalRequest = SelectRecord           ! consultar registros de productos
        VerProductos
        IF GlobalResponse <> RequestCompleted ! Si no selecciona producto
            CLEAR(PRO:Record)                 ! limpiar el buffer
            ReturnValue = EditAction:None      ! configurar acción para continuar
        END                                   ! en el mismo campo
    END
    BRW5.Q.DETAL:NumProducto=PRO:NumProducto ! Asignar valores del archivo de
    BRW5.Q.DETAL:ProdPrecio = PRO:Precio     ! productos a los campos del
    BRW5.Q.DETAL:Descipcion = PRO:Descripcion ! Browse Queue y mostrarlos
    DISPLAY
END

```

Este código es realmente interesante. Advierta que la primera sentencia ejecutable (generada para usted) es una llamada al método `PARENT.TakeEvent`. Esto llama al método **EditClass.TakeEvent** que estamos “pasando por alto” para que pueda hacer lo que habitualmente hace (código muy similar al de `TakeEvent` que usted ya escribió para el control `SPIN` de `DETAL:Cantidad`).



Todo el resto del código es para dar a esta clase derivada una funcionalidad extra no presente en su clase padre. Este es el poder real de la POO: si uno quiere todo lo que hace el padre, más algo más, no es necesario duplicar todo el código del padre, sino que basta con llamarlo. En este caso, todo el código extra es para realizar algunas tareas estándar de validación de datos. Este código verificará si el usuario tipió un número de producto válido y, si no lo hizo, llamará al procedimiento VerProductos para que escoja de una lista.

“Pasar por alto” los métodos

Hay un punto muy importante a comprender sobre el trabajo en el Editor de puntos de inserción, cuando se “pasan por alto” los métodos: *apenas escribe algo en un punto de inserción de un método que puede pasarse por alto, ¡ya lo ha hecho!*. Incluso una simple línea de comentario (!) puede hacerlo, porque el Generador de Aplicaciones advierte que usted ha escrito algo de código, y genera el correspondiente prototipo del método dentro de la declaración local de CLASS. Para evitar que usted accidentalmente escriba un comentario que provoque “pasar por alto” el método y que destruya su funcionalidad, los templates ABC generan automáticamente llamadas al método PADRE (PARENT) y retornan (RETURN) sentencias para usted, según se necesite.

Advertirá que todos nuestros métodos “pasados por alto” contenían llamadas generadas a su método PARENT, y que los métodos TakeEvent también tenían sentencias RETURN generadas. A veces, usted desea que esas sentencias se ejecuten; aunque a veces, no. Para estos últimos casos, simplemente escriba el código en el punto de inserción que está antes de la llamada al método PARENT y escriba su propia sentencia RETURN al final del código (como lo hicimos al final del código que escribimos para los métodos **EditInPlace::DETAL:Cantidad.CreateControl**). Esto significa que el método PARENT generado y la sentencia RETURN nunca se ejecutarán. El compilador de Clarion tiene la suficiente inteligencia para reconocer que esas sentencias nunca se ejecutarán, y optimiza el código objeto, quitándolas de él.

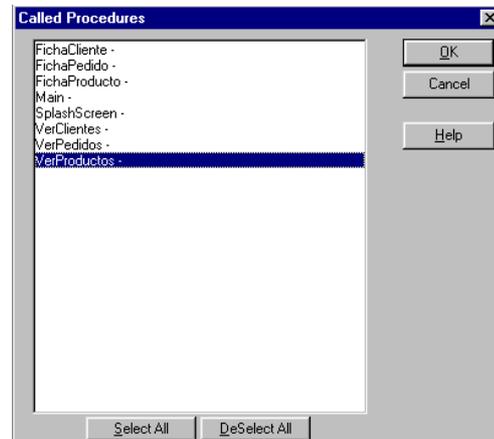
Elija **Exit!** para cerrar el Editor de puntos de inserción.

Actualización del árbol de procedimientos

El método `EditInPlace::Detal:NumProducto.TakeEvent` llama al procedimiento `VerProductos` de dentro de su código.

Dado que este es código insertado, el Generador de Aplicaciones ignora que usted ha llamado a este procedimiento, y necesita ser informado (o se generarán errores de compilación), para que pueda generar la estructura MAP correcta para el módulo que contiene este procedimiento.

1. Dé CLIC DERECHO sobre *FichaPedido* y elija **Procedures** del menú contextual.
2. Destaque *VerProductos* y oprima el botón **OK**.
3. Elija **File** > **Save**, u oprima el botón *Guardar* para guardar su trabajo.



Generación de código

1. Elija **Project** > **Generate** para generar el código fuente de su aplicación.

```

MAP
  INCLUDE('TUTU0008.INC'),ONCE      !Local module procedure declarations
  INCLUDE('TUTU0003.INC'),ONCE      !Req'd for module callout resolution
  INCLUDE('TUTU0006.INC'),ONCE      !Req'd for module callout resolution
END

FichaPedido PROCEDURE              !Generated from procedure template - Window

FilesOpened      BYTE
ActionMessage    CSTRING(40)
ItemTotal        DECIMAL(7,2)
BRWS::View:Browse VIEW(Detalle)
  PROJECT(DETAL:NumProducto)
  PROJECT(DETAL:Cantidad)
  PROJECT(DETAL:ProdPrecio)
  PROJECT(DETAL:NumPedido)
  JOIN(PRO:KeyNumProducto,DETAL:NumProducto)
  PROJECT(PRO:Descripcion)
  PROJECT(PRO:NumProducto)
END
Queue:Browse     QUEUE              !Queue declaration for browse/combo box usin
DETAL:NumProducto LIKE(DETAL:NumProducto) !List box control field - type derived from
DETAL:Cantidad   LIKE(DETAL:Cantidad)  !List box control field - type derived from
DETAL:ProdPrecio LIKE(DETAL:ProdPrecio) !List box control field - type derived from
ItemTotal       LIKE(ItemTotal)        !List box control field - type derived from
PRO:Descripcion LIKE(PRO:Descripcion)  !List box control field - type derived from
DETAL:NumPedido LIKE(DETAL:NumPedido)  !Browse key field - type derived from field
PRO:NumProducto LIKE(PRO:NumProducto)  !Related join file key field - type derived
Mark            BYTE                 !Stores entry's marked status
  
```

2. Dé CLIC DERECHO sobre *FichaPedido* y seleccione **Module** del menú contextual. Aparece el Editor de Texto que contiene el código fuente generado para el procedimiento *FichaPedido*. Adviértase que hay mucho menos código aquí que lo que había en el Editor de Puntos de Inserción. Allí todo el código estaba presente para que usted pudiera apreciar el contexto, y para ofrecerle puntos de inserción mediante los cuales “pasar por alto” los métodos, si hubiere necesidad. Sin embargo, el Generador de Aplicaciones y los Template ABC de Clarion tienen la suficiente inteligencia para generar solamente el código necesario, cuando es necesario.
3. Elija **Exit!** para cerrar el Editor de Texto. Ahora es un buen momento para probar la aplicación. Están listas todas las partes de ingreso de datos, y lo único que queda por hacer son los informes, que trataremos en la próxima lección.

12 - CREACIÓN DE INFORMES

Un simple listado de clientes

El último punto a cubrir en este cursillo son los informes. Primero, crearemos un simple listado de clientes para presentarle el Diseñador de Informes.

Después crearemos un Informe de Pedidos para demostrar lo fácil que es crear informes relacionales, con segmentos multinivel, totales de grupo, y diseño de páginas. Luego, copiaremos el Informe de Pedidos y limitaremos la impresión a los Pedidos de un cliente y, por último a un solo Pedido.

Punto de inicio:

Debe estar abierto el archivo TUTORIAL. APP

Actualización del menú principal

Primero, necesitamos añadir selecciones al menú, de manera que el usuario pueda llamar los informes, y así el Generador de Aplicaciones podrá llamar a los procedimientos "ToDo" que correspondan.

Nuevo ítem de menú

1. Dé CLIC DERECHO sobre el procedimiento *Main* en el árbol de la aplicación y elija **Window** del menú contextual.
2. Elija **Menu** ➤ **Edit Menu** del menú del Diseñador de Ventanas.
3. Destaque el ítem **P&rint Setup** en la lista del Editor de Menús. Si lo desea, puede editar el campo **Menu Text** y traducirlo a: *Configuración de &Impresora*.
4. Oprima el botón **Item**.
5. Escriba *Imprimir Lista &Clientes* en el campo **Menu Text**, y oprima TAB.

Especificar la acción del nuevo ítem

1. Elija la lengüeta **Actions**.
2. Elija *Call a Procedure* de la lista desplegable **When Pressed**.
3. Escriba *InformeClientes*.
4. Marque la casilla **Initiate Thread**.

Añición de un segundo menú

1. Oprima el botón **Item**.

2. Escriba *Imprimir & Pedidos* en el campo **Menu Text**.
3. Elija la lengüeta **Actions**.
4. Elija *Call a Procedure* de la lista desplegable **When Pressed**.
5. Escriba *InformePedidos* en el campo **Procedure Name**.
6. Marque el casillero **Initiate Tread**.

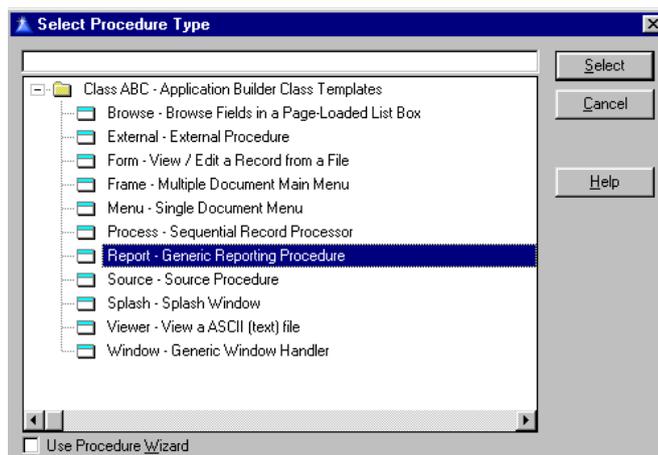
Adición de un tercer ítem al menú

1. Oprima el botón **Item**.
2. Escriba *Imprimir Pedidos por & Cliente* en el campo **Menu Text**.
3. Elija la lengüeta **Actions**.
4. Elija *Call a Procedure* de la lista desplegable **When Pressed**.
5. Escriba *InformePedidosCliente* en el campo **Procedure Name**.
6. Marque el casillero **Initiate Thread**.
7. Oprima el botón **Close** para cerrar el Editor de Menús.
8. Elija **Exit!** para cerrar el Diseñador de Ventanas (y guarde su trabajo).

Creación del Informe

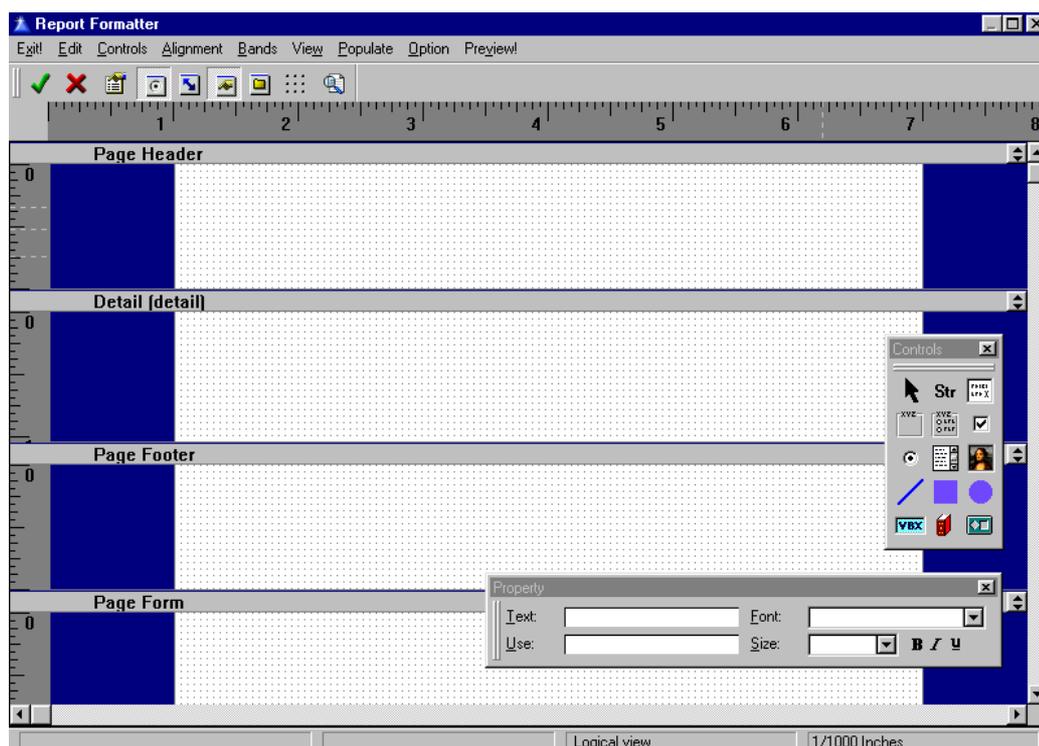
Ahora podemos crear el primer informe, utilizando el Diseñador de Informes.

1. Destaque el procedimiento *InformeClientes* en el árbol de aplicaciones.
2. Oprima el botón **Properties**.
3. Destaque *Report* en la caja de diálogo **Select Procedure Type**, despeje el casillero **Use Procedure Wizard** (utilizar el Asistente de Procedimientos), y oprima el botón **Select**.



4. Oprima el botón **Report** en la caja de diálogo **Procedure Properties**.

Aparece el Diseñador de Informes. Aquí puede editar visualmente el informe y sus controles. El diseñador de Informe representa las cuatro partes básicas de la estructura de datos REPORT mostrando cuatro bandas: *Page Header* (Encabezado de página), *Detail* (Detalle), *Page Footer* (Pie de página), y *Form* (Formulario). Cada banda es una unidad funcional, que se imprime toda junta. Vea el capítulo de la *User's Guide* titulado *Using the Report Formatter* para más información sobre las partes del informe y el modo en que las genera el motor de impresión.



Para este informe, colocará controles de número de página en el encabezamiento, y colocará los campos del archivo *Cientes* en la banda de Detalle.

Colocación de una cadena

1. Elija **Controls** > **String**, o tome la herramienta *String* de la caja de herramientas **Controls**.
2. Dé CLIC sobre la banda **Page Header**.
3. Dé CLIC DERECHO sobre el control, y elija **Properties** desde el menú contextual.

Aparece la caja de diálogo **String Properties**.

4. Escriba *Pág. N°*: en el campo **Text**, y oprima el botón **OK**.

Colocación de un template de control para imprimir el número de página

1. Elija **Populate** > **Control Template**, o “pique” sobre la herramienta *Control Template* de la caja de herramientas **Controls**.

2. Destaque **ReportPageNumber** y oprima el botón **Select**.
3. Dé CLIC a la derecha de la cadena ubicada previamente.

Diseño del Detalle

Se imprime una banda de Detalle (Detail) por cada registro del informe. Para este procedimiento, colocará los campos en un arreglo de bloque, que produce un informe de etiquetas de correo.

1. Elija **Populate > Multiple Fields**, o dé CLIC sobre el control *Dictionary Field* en la caja de herramientas **Controls**.
2. En la caja de diálogo **Select Field**, destaque la carpeta “ToDo”, y oprima el botón **Insert**.
3. Seleccione el archivo *Clientes* de la caja de diálogo **Insert File**, y oprima el botón **Select**.
4. Oprima el botón **Edit**.
5. Destaque *KeyNumCliente* en la caja de diálogo **Change Access Key**, y oprima el botón **Select**.
6. Destaque *CLI:Empresa* en la lista **Fields** y oprima el botón **Select**.
7. Dé CLIC dentro de la banda **Detail**, cerca del ángulo superior izquierdo.
8. Destaque *CLI:Nombres* en la lista **Fields** y oprima el botón **Select**.
9. Dé CLIC dentro de la banda **Detail**, justo debajo del primer control.
10. Destaque *CLI:Apellidos* en la lista **Fields** y oprima el botón **Select**.
11. Dé CLIC dentro de la banda **Detail**, a la derecha del control que acaba de colocar.
12. Destaque *CLI:Dirección* en la lista **Fields**, y oprima el botón **Select**.
13. Dé CLIC dentro de la banda **Detail**, debajo del segundo control que acaba de colocar.

Ajuste de la banda de detalle

En este punto, probablemente quede muy poco lugar en la banda, y haya que agrandarla.

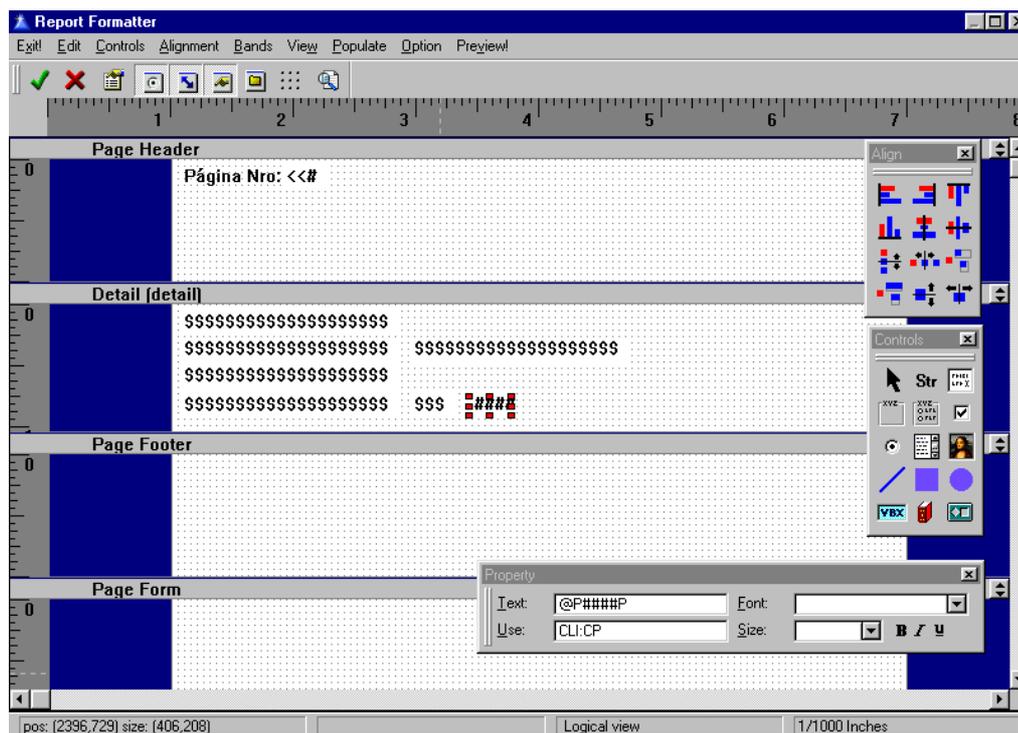
1. Oprima el botón **Cancel** para salir del modo de llenado múltiple.
2. Dé CLIC dentro de la banda **Detail**, pero no en uno de los controles de cadena. Aparecen las asas del área de detalle.
3. Ajuste el tamaño, ARRASTRANDO el asa central inferior, deje lugar para una o más líneas.

Coloque el resto de los campos

1. Elija **Populate > Multiple Fields**, o dé CLIC sobre la herramienta *Dictionary Field* en la caja de herramientas **Controls**.
2. Destaque *CLI:CP* en el campo **Fields**, y oprima el botón **Select**.
3. Dé CLIC sobre la banda **Detail**, debajo del último control que ha colocado.
4. Destaque *CLI:Ciudad* en el campo **Fields**, y oprima el botón **Select**.
5. Dé CLIC sobre la banda **Detail**, a la derecha del último control que ha colocado.

6. Destaque *CLI:Provincia* en el campo **Fields**, y oprima el botón **Select**.
7. Dé CLIC sobre la banda **Detail**, a la derecha del último control que ha colocado.
8. Oprima el botón **Cancel** para salir del modo de llenado múltiple.

Advierta que tiene el mismo grupo de herramientas de alineación en el Diseñador de Informes que en el Diseñador de Ventanas (elija **Option** > **Show Alignbox**, para mostrar esa caja flotante de herramientas).



Selección de una fuente base para el informe

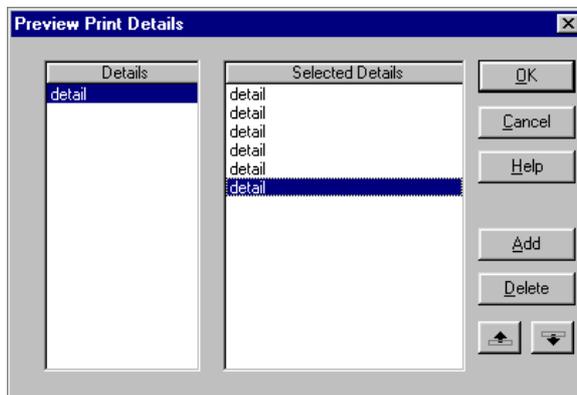
1. Elija **Edit** > **Report Properties** para configurar los atributos por omisión.
2. Oprima el botón **Font**.
3. Seleccione una fuente, estilo y tamaño para usar como base del informe.
Si no elige una fuente, se usa la fuente por omisión de la impresora.
4. Oprima el Botón **OK** para cerrar la caja de diálogo **Select Font**.
5. Oprima el botón **OK** para cerrar la caja de diálogo **Report Properties**.

Vista preliminar del informe

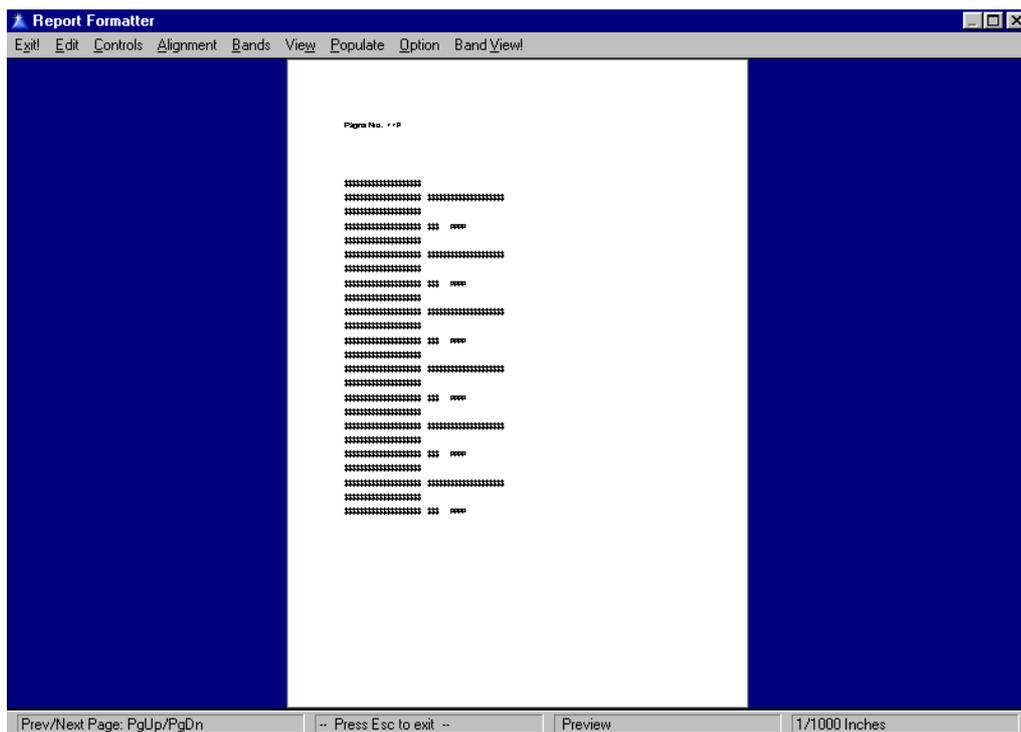
1. Elija **Preview!** para visualizar cómo quedará la página impresa.
2. Destaque *Detail* en la lista **Details** y oprima el botón **Add** varias veces.

Esto llena la vista preliminar con varias bandas “impresas” para poder ver cómo quedan. Debido a que cada informe puede contener varias bandas de distinto tipo, hay que seleccionar

cuál desea ver antes de la previsualización. De este modo, el Diseñador de Informes sabe qué debe componer sobre la pantalla.



3. Oprima el botón **OK**.



4. Cuando ha terminado de “previsualizar”, elija **Band View!**
5. Elija **Exit!** para regresar a la caja de diálogo **Procedure Properties** (y guarde su trabajo).
6. Oprima el botón **OK** para cerrar la caja de diálogo **Procedure Properties**.
7. Elija **File** > **Save**, u oprima el botón *Guardar* en la barra de herramientas para guardar su trabajo.

Informe Pedidos

A continuación, crearemos uno de los tipos de informe más comunes. Un pedido usa la mayor parte de los archivos del diccionario de datos, y demuestra cómo crear saltos de grupo y totales. También le mostrará cómo controlar la paginación basada en los saltos de grupo.

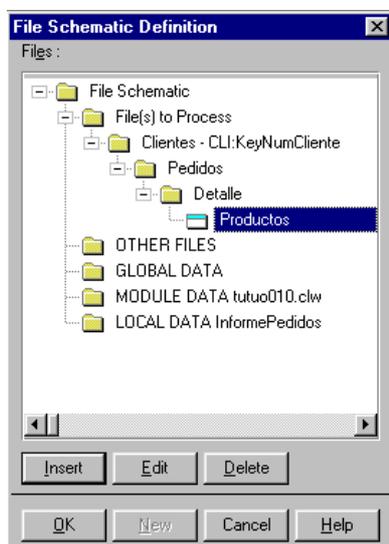
Creación del Informe

1. Destaque el procedimiento *InformePedidos*.
2. Oprima el botón **Properties**.
3. Destaque *Report* en la caja de diálogo **Select Procedure Type**, evite el uso del Asistente de Procedimiento (despeje el casillero **Use Procedure Wizard**), y oprima el botón **Select**.
Aparece la caja de diálogo **Procedure Properties**.

Archivos del informe

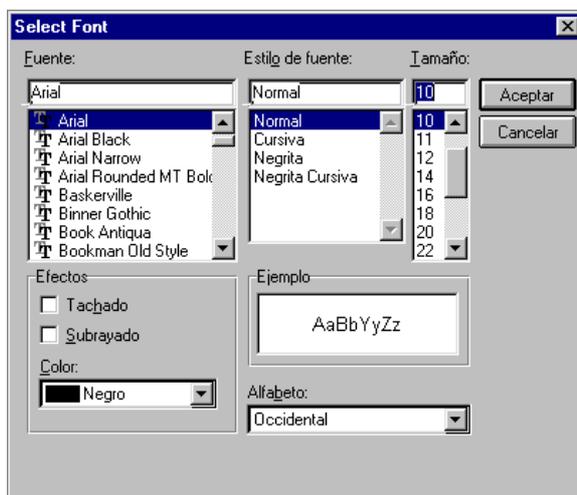
1. Oprima el botón **Files** en la caja de diálogo **Procedure Properties**.
Aparece la caja de diálogo **File Schematic Definition**.
2. Destaque la carpeta "ToDo", y oprima el botón **Insert**.
3. Seleccione el archivo *Clientes* de la caja de diálogo **Insert File**, y oprima el botón **Select**.
4. Oprima el botón **Edit**.
5. Destaque *CLI:KeyNumCliente* en la caja diálogo **Change Access Key**, y oprima el botón **Select**.
Este informe procesará todos los registros del archivo *Clientes* ordenados por *NumCliente*.
6. Destaque el archivo *Clientes*, y oprima el botón **Insert**.
7. Seleccione el archivo *Pedidos* de la caja de diálogo **Insert File**, y oprima el botón **Select**.
Esto procesará todos los *Pedidos* de cada *Cliente*.
8. Destaque el archivo *Pedidos*, y oprima el botón **Insert**.
9. Seleccione el archivo *Detalle* de la caja de diálogo **Insert File**, y oprima el botón **Select**.
Cada Pedido se imprimirá con todas las líneas de Detalle que le corresponden.
10. Destaque el archivo *Detalle*, y oprima el botón **Insert**.
11. Seleccione el archivo *Productos* de la caja de diálogo **Insert File**, y oprima el botón **Select**.
Cada registro de Detalle buscará el registro vinculado del archivo Productos. En este punto, el árbol de archivos se verá así:

- Oprima el botón **OK** para volver a la caja de diálogo **Procedure Properties**.



Configuración de las características predeterminadas

- Oprima el botón **Report**.
- Elija **Edit** ➤ **Report Properties** para configurar las características predeterminadas del informe.
- Oprima el botón **Font**.
- Elija una fuente, estilo y tamaño para usar como fuente básica de este informe.



Si no elige una fuente para el informe, se imprimirá utilizando la fuente por omisión de la impresora. Debe elegir una fuente que usted sepa que el usuario tendrá instalada (en general, pueden considerarse seguras las fuentes que vienen con Windows).

- Oprima **OK** para cerrar la caja de diálogo **Select Font**.
- Oprima **OK** para cerrar la caja de diálogo **Report Properties**.

Llenado de la banda de página base

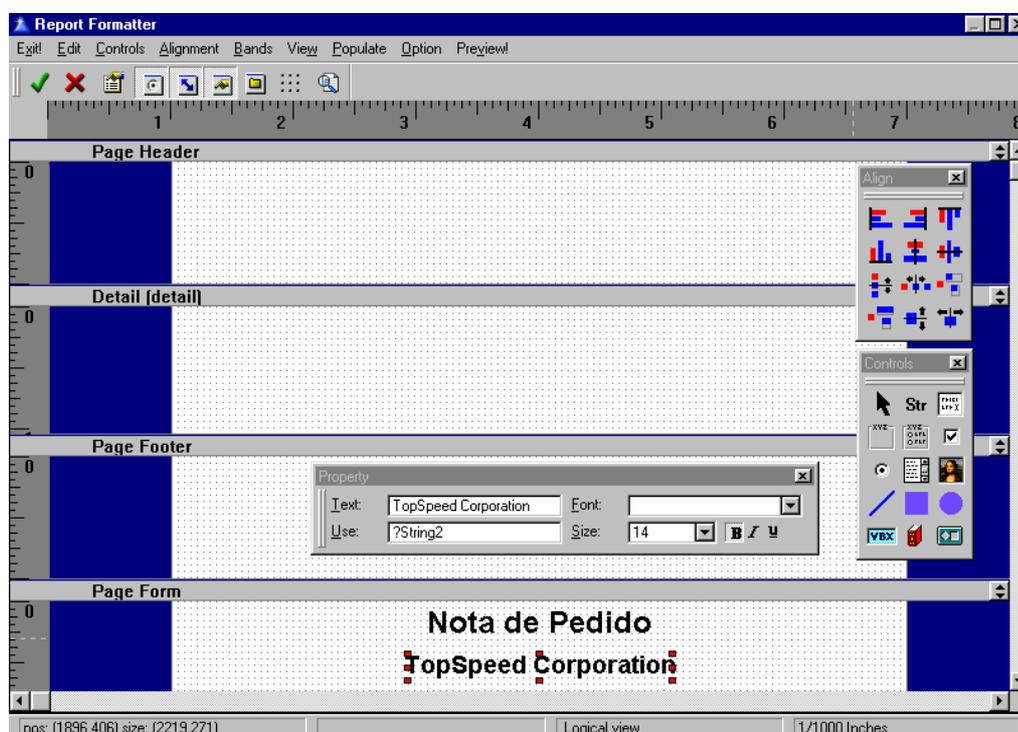
La banda “página base” (*Page Form*) se imprime una vez por cada página del informe; pero su contenido se compone sólo una vez, al abrir el informe. Esto la hace útil para información constante, que estará presente en cada página.

Colocación de una cadena de caracteres

1. Elija **Controls** ➤ **String**, o tome la herramienta *String* de la caja de herramientas **Controls**.
2. Dé CLIC sobre la banda **Page Form**.
3. Dé DOBLE CLIC sobre el control.
Aparece la caja de diálogo **String Properties**.
4. Escriba *Nota de Pedido* en el campo **Text**.
5. Oprima el botón **Font**.
6. Elija una fuente, estilo y tamaño para el texto (aquí sería adecuada una tipografía grande, en negritas).
7. Oprima el botón **OK** para cerrar la caja de diálogo **Select Font**.
8. Oprima el botón **OK** para cerrar la caja de diálogo **String Properties**.
9. Ajuste el tamaño del control para que pueda contener el texto, ARRASTRANDO las asas.

Colocación de la siguiente cadena de caracteres

1. Elija **Controls** ➤ **String**, o elija la herramienta **String** de la caja de herramientas **Controls**.
2. Dé CLIC en la parte superior de la banda **Page Form**, justo debajo de la última cadena de caracteres que ha colocado.
3. Dé CLIC DERECHO sobre el control, y elija **Properties** del menú contextual.
4. Escriba el nombre de su empresa en el campo **Text**.
5. Oprima el botón **Font** y elija una fuente, estilo y tamaño adecuados (algo ligeramente más pequeño que el campo anterior estaría bien aquí).
6. Oprima el botón **OK** para cerrar la caja de diálogo **Select Font**.
7. Oprima el botón **OK** para cerrar la caja de diálogo **String Properties**.
8. Ajuste el tamaño del control, para que pueda contener el texto, ARRASTRANDO las asas.



Poblando la banda de detalle

La banda de detalle (**Detail**) se imprimirá cada vez que se lea nueva información del archivo “Hijo” en el nivel más bajo del árbol de archivos. Para este informe de Pedidos, el nivel inferior es el archivo “Detalle” (recuerde que *Productos* es un archivo de “consulta” de muchos-a-uno desde el archivo Detalle).

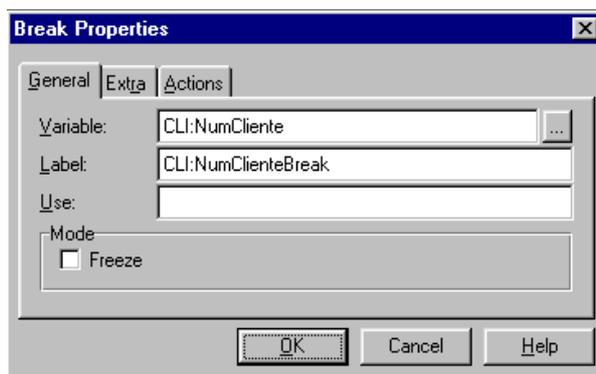
1. Elija **Populate** > **Multiple Fields**, o dé CLIC sobre la herramienta *Dictionary Field* en la caja de herramientas **Controls**.
2. Destaque *Detalle* en la lista **Files** y elija *DETAL:Cantidad* en la lista **Fields** y oprima el botón **Select**.
3. Dé CLIC dentro de la banda **Detail**, cerca del ángulo superior izquierdo.
4. Destaque *DETAL:NumProducto* en la lista de **Fields**, y oprima el botón **Select**.
5. Dé CLIC dentro de la banda **Detail**, a la derecha del primer control.
6. Destaque *Productos* en la lista **Files**, y elija *PROD:Descripción* en la lista **Fields** y oprima el botón **Select**.
7. Dé CLIC dentro de la banda **Detail**, a la derecha del control anterior.
8. Destaque *Detalle* en la lista **Files**, y elija *DETAL:ProdPrecio* en la lista **Fields** y oprima el botón **Select**.
9. Dé CLIC dentro de la banda **Detail**, a la derecha del control anterior.

10. Destaque *LOCAL DATA InformePedidos* en la lista **Files**, y oprima el botón **New**.
Así podrá añadir una variable local sin tener que ir hasta la ventana **Procedure Properties**, y oprimir el botón **Data**. Esta variable se usará para mostrar el precio total para cada línea del pedido.
11. Escriba *ItemTotal* en el campo **Field Name**.
12. Elija *DECIMAL* de la lista desplegable **Data Type**.
13. Escriba 7 en el campo de caracteres (**Characters**).
14. Escriba 2 en el campo de decimales (**Places**) y oprima el botón **OK**.
15. Dé CLIC dentro de la banda **Detail**, a la derecha del último control colocado.
16. Oprima el botón **Cancel** para salir de este modo de operación.
17. Mueva todos los controles a la parte superior de la banda **Detail**, alíneelos horizontalmente, y ajuste el tamaño de la banda, para que sea apenas más alta que los controles mismos.

Adición de saltos de grupo

Necesitamos imprimir distintos tipos de información en cada pedido. Por lo tanto, necesitamos crear estructuras de salto (BREAK) para que se imprima algo cada vez que cambia la información en el archivo *Pedidos* y cada vez que cambia la información del archivo *Cientes*.

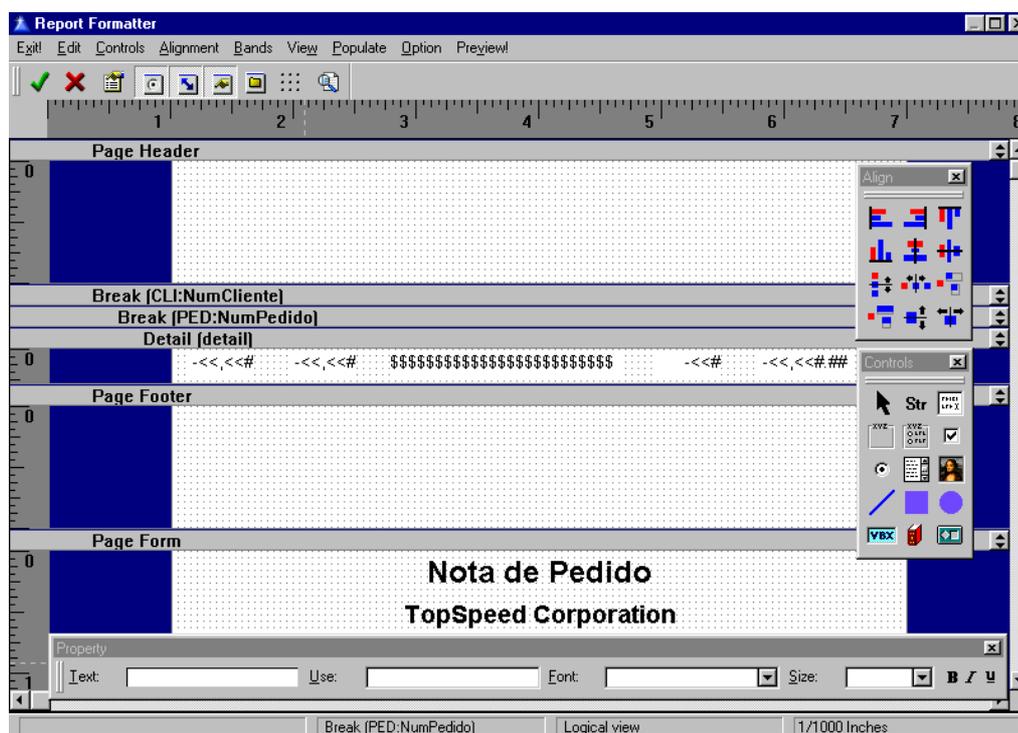
1. Elija **Bands** > **Surrounding Break**, y luego dé CLIC sobre la banda **Detail**.
Aparece la caja de diálogo **Break Properties**.
2. Oprima el botón de puntos suspensivos (...) junto al campo **Variable**.
3. Destaque *Cientes* en la lista **Files** y elija *CLI:NumCliente* en la lista **Fields** y oprima el botón **Select**.
4. Escriba *CLI:NumClienteBreak* en la etiqueta **Label**, y oprima el botón **OK**.



Aparece una banda de salto (**Break** [CLI:NumCliente]) sobre la banda Detalle, con una sangría a la izquierda, lo que está dentro de la estructura **Break**.

5. Elija **Bands** > **Sorrounding Break**, y dé CLIC sobre la banda **Detail**.
6. Aparece la caja de diálogo **Break Properties**.
7. Oprima el botón de puntos suspensivos (...) junto al campo **Variable**.
8. Destaque *Pedidos* en la lista **Files** y elija *PED:NumPedido* en la lista **Fields** y oprima el botón **Select**.
9. Escriba *PED:NumPedidoBreak* en el campo **Label** y oprima el botón **OK**.

Ahora el informe se ve así:



Creación de las cabeceras y pies de grupo

1. Elija **Bands** > **Group Header**, y dé CLIC sobre la banda **Break (PED:NumPedido)**.

La banda **Group Header (PED:NumPedido)** aparece encima de la banda **Detalle**. Esta banda se imprimirá cada vez que el valor del campo **PED:NumPedido** cambie, al principio de cada nuevo grupo de registros. Lo usaremos para imprimir el nombre y dirección de la empresa, junto con el número de pedido y su fecha.

2. Elija **Bands** > **Group Footer**, y dé CLIC sobre la banda **Break (PED:NumPedido)**.

Aparecerá la banda **Group Footer (PED:NumPedido)** debajo de la banda **Detail**. Esta banda se imprimirá cada vez que el valor de **PED:NumPedido** cambie, al final de cada grupo de registro. La usaremos para imprimir el total del pedido.

3. Dé CLIC **DERECHO** sobre la banda **Group Footer (PED:NumPedido)** y elija **Properties** del menú contextual.

Aparece la caja de diálogo **Page/Group Footer Properties**.

4. Marque el casillero **Page after**.

Esto hará que el motor de impresión imprima esta banda, y luego inicie el proceso de desbordamiento de página (Page Overflow). Se compondrá la banda de pie de página (**Page Footer**), se enviará una orden de “página nueva” a la impresora, y luego se compondrá la banda de encabezado (**Page Header**) de la página siguiente.

5. Oprima el botón **OK**.

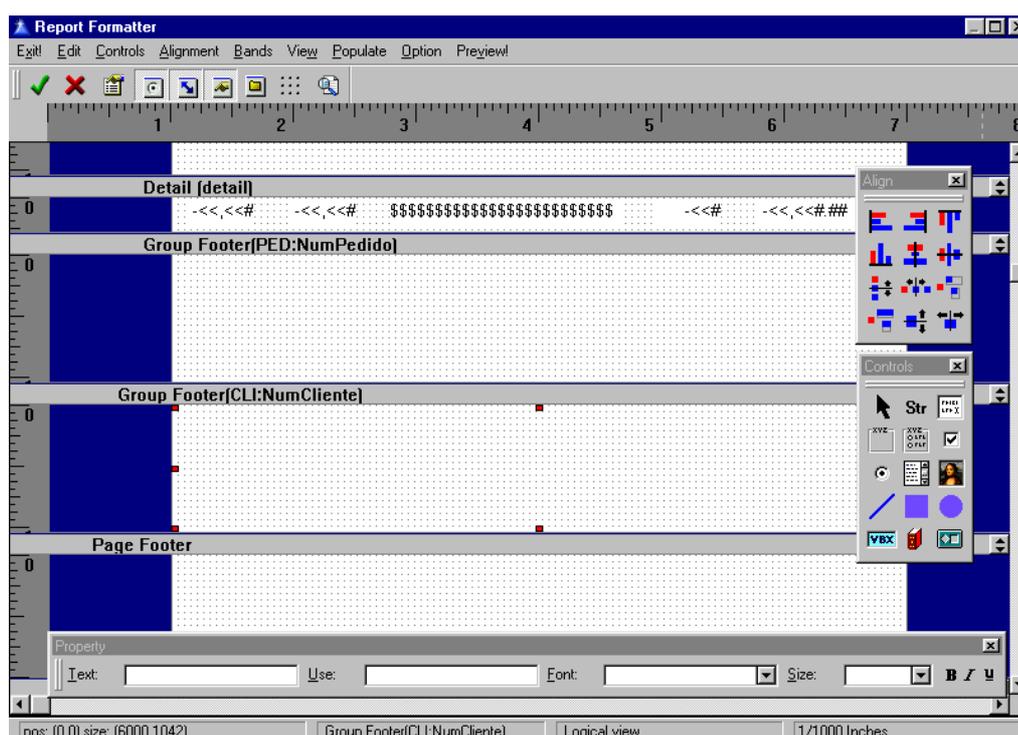
6. Elija **Bands** > **Group Footer**, y dé CLIC sobre la banda **Break (CLI:NumCliente)**.

Esta banda aparece debajo de la del **Group Footer (Ped:NumPedido)**. Se imprimirá cada vez que el valor del campo **CLI:NumCliente** cambie, al final de cada grupo de registros. Usaremos esto para imprimir la información resumida sobre pedidos de cada empresa cliente.

7. Dé CLIC DERECHO sobre la banda **Group Footer (CLI:NumCliente)** y elija **Properties** del menú contextual.

8. Marque el casillero **Page after**.

9. Oprima el botón **OK**.



Llenado de la banda cabecera de grupo

Colocación de los campos de Clientes

1. Elija **Populate** > **Multiple Fields**, o dé CLIC sobre la herramienta *Dictionary Field* en la caja de herramientas **Controls**.

2. Destaque *Clientes* en la lista **Files** y elija *CLI:Empresa* en la lista **Fields** y oprima el botón **Select**.
3. Dé CLIC dentro de la banda **Group Header (PED:NumPedido)** cerca del ángulo superior izquierdo.
4. Destaque *CLI:Nombres* en la lista **Fields** y oprima el botón **Select**.
5. Dé CLIC dentro de la banda **Group Header (PED:NumPedido)**, debajo del primer control.
6. Destaque *CLI:Apellidos* en la lista **Fields** y oprima el botón **Select**.
7. Dé CLIC dentro de **Group Header (PED:NumPedido)**, a la derecha del control ya colocado.
8. Destaque *CLI:Direccion* en la lista **Fields** y oprima el botón **Select**.
9. Dé CLIC dentro de **Group Header (PED: NumPedido)**, debajo del *segundo* control que ha colocado.
10. Destaque *CLI:CP* en la lista **Fields** y oprima el botón **Select**.
11. Dé CLIC dentro de **Group Header (PED:NumPedido)**, debajo de la dirección.
12. Destaque *CLI:Ciudad* en la lista **Fields**, y oprima el botón **Select**.
13. Dé CLIC dentro de **Group Header (PED:NumPedido)**, junto al último control que colocó.
14. Destaque *CLI:Provincia* en la lista **Fields**, y oprima el botón **Select**.
15. Dé CLIC dentro de **Group Header (PED:NumPedido)**, a la derecha de *CLI:Ciudad*.

Coloque los campos del archivo Pedidos

1. Marque *Pedidos* en la lista **Files** y elija *PED:NumPedido* en la lista **Fields** y oprima el botón **Select**.
2. Dé CLIC dentro de **Group Header (PED:NumPedido)**, cerca del ángulo superior derecho.
3. Destaque *PED:FechaPedido* en la lista **Fields**, y oprima el botón **Select**.
4. Dé CLIC dentro de la banda **Group Header (PED:NumPedido)**, debajo del último control colocado.
5. Oprima el botón **Cancel** para cerrar la caja de diálogo **Select Field** y salir de este modo de trabajo.

Colocación del texto permanente y de las cabeceras de columna

1. Elija **Controls** > **String**, o “pique” sobre la herramienta **String** de la caja de herramientas **Controls**.
2. Dé CLIC dentro de la banda **Group Header (PED:NumPedido)**, a la izquierda del control de número de pedido que acaba de colocar.
3. Escriba *Pedido N°:* en el campo **Text** de la caja de herramientas **Property**, y oprima TAB.

4. Elija **Controls** > **String**, o “pique” sobre la herramienta **String** de la caja de herramientas **Controls**.
5. Dé CLIC dentro de la banda **Group Header (PED:NumPedido)**, a la izquierda del control de fecha.
6. Escriba *Fecha*: en el campo **Text** de la caja de herramientas **Property**, y oprima TAB.
7. Elija **Controls** > **String**, o “pique” sobre la herramienta **String** de la caja de herramientas **Controls**.
8. Dé CLIC dentro de la banda **Group Header (PED:NumPedido)**, a la izquierda debajo de *Cliente*.
9. Escriba *Cantidad* en el campo **Text** de la caja de herramientas **Property**, y oprima TAB.
10. Elija **Controls** > **String**, o “pique” sobre la herramienta **String** de la caja de herramientas **Controls**.
11. Dé CLIC dentro de la banda **Group Header (PED:NumPedido)**, a la derecha del último control colocado.
12. Escriba *Producto* en el campo **Text** de la caja de herramientas **Property**, y oprima TAB.
13. Elija **Controls** > **String**, o “pique” sobre la herramienta **String** de la caja de herramientas **Controls**.
14. Dé CLIC dentro de la banda **Group Header (PED:NumPedido)**, a la derecha de la última cadena de caracteres colocada, encima del control *DETAL:Precio* en la banda **Detail**.
15. Escriba *Precio Unitario* en el campo **Text** de la caja de herramientas **Property**, y oprima TAB.
16. Elija **Controls** > **String**, o “pique” sobre la herramienta **String** de la caja de herramientas **Controls**.
17. Dé CLIC dentro de la banda **Group Header (PED:NumPedido)**, a la derecha de la última cadena de caracteres ubicada, directamente encima del control *ItemTotal* en la banda **Detail**.
18. Escriba *Total Item* en el campo **Text** de la caja de herramientas **Property**, y oprima TAB.

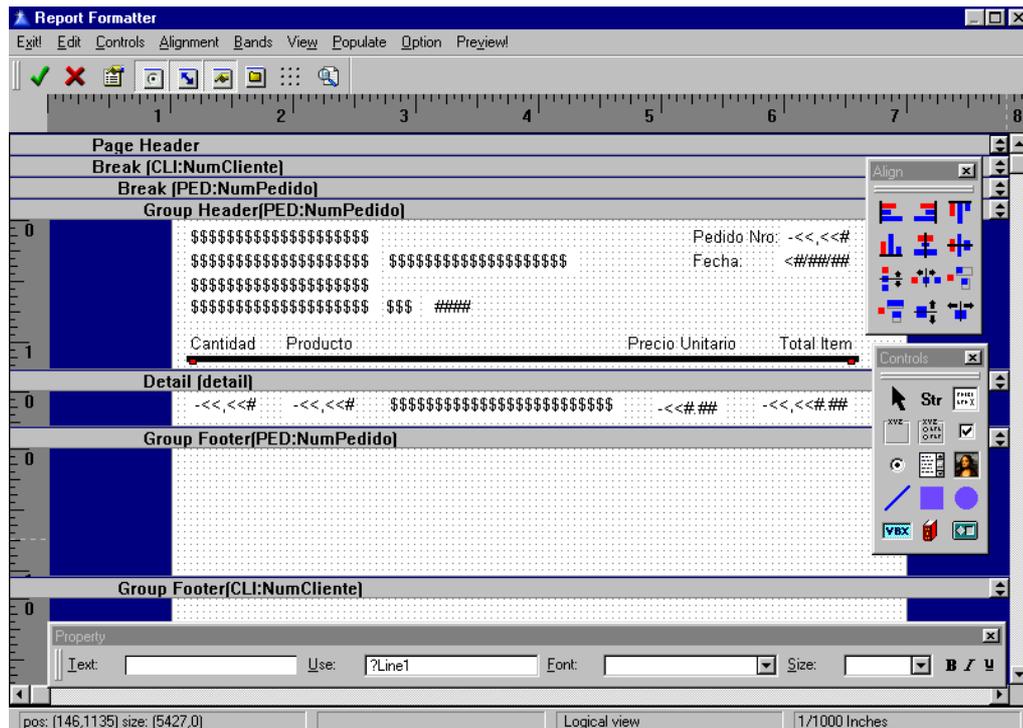
Colocación de una línea gruesa bajo los encabezamientos

1. Elija **Controls** > **Line**, o pique sobre la herramienta *Line* de la caja de herramientas **Controls**.
2. Dé CLIC dentro de la banda **Group Header (PED:NumPedido)**, bajo la cadena *Cantidad*.
3. Ajuste la línea **ARRASTRANDO** las asas hasta que se aparezca a lo ancho de todos los encabezados de columna.
4. Dé CLIC **DERECHO** y elija **PROPERTIES** del menú contextual.
5. Escriba 50 en el campo de ancho de línea (**Line Width**).

Así se engrosa la línea.

- Oprima el botón **OK** para cerrar la caja de diálogo **Line Properties**.

El informe debería verse ahora aproximadamente así:

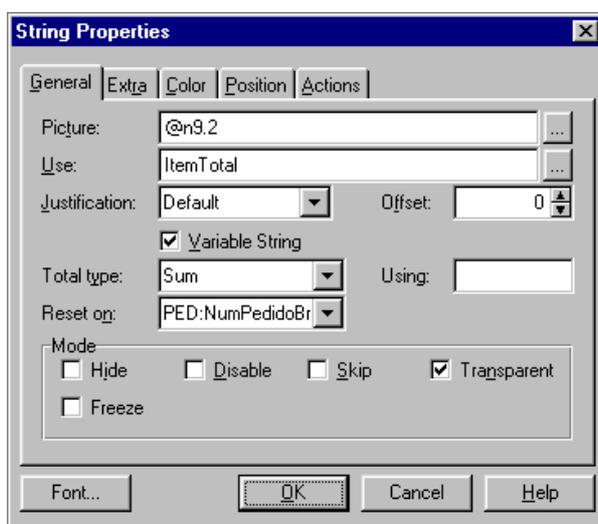


Relleno de la banda de pie de grupo

Colocación del texto constante y del campo de total

- Elija **Controls** > **String**, o "pique" sobre la herramienta **String** de la caja de herramientas **Controls**.
- Dé CLIC dentro de la banda **Group Footer (PED: NumPedido)**, en el medio de la banda.
- Escriba *Total del Pedido:* en el campo **Text** de la caja de herramientas **Property**, y oprima TAB.
- Elija **Controls** > **String**, o "pique" sobre la herramienta **String** de la caja de herramientas **Controls**.
- Dé CLIC dentro de la banda **Group Footer (PED:NumPedido)**, a la derecha de la cadena que acaba de colocar.
- Dé CLIC DERECHO y elija **Properties** del menú contextual.
- Marque el casillero **Variable String**.

8. Oprima el botón de puntos suspensivos (...) del campo **Use**.
9. Destaque *LOCAL DATA InformePedidos* en la lista **Files**, elija *ItemTotal* del campo **Fields**, y oprima el botón **Select**.
10. Escriba @n9.2 en el campo de formato de visualización (**Picture**). Esto dará formato "en español" al total: con puntos en las separaciones de miles, y una coma como separador decimal.
11. Elija *Sum* de la lista desplegable **Total Type**.
12. Elija *PED:NumPedidoBreak* de la lista desplegable **Reset On**.

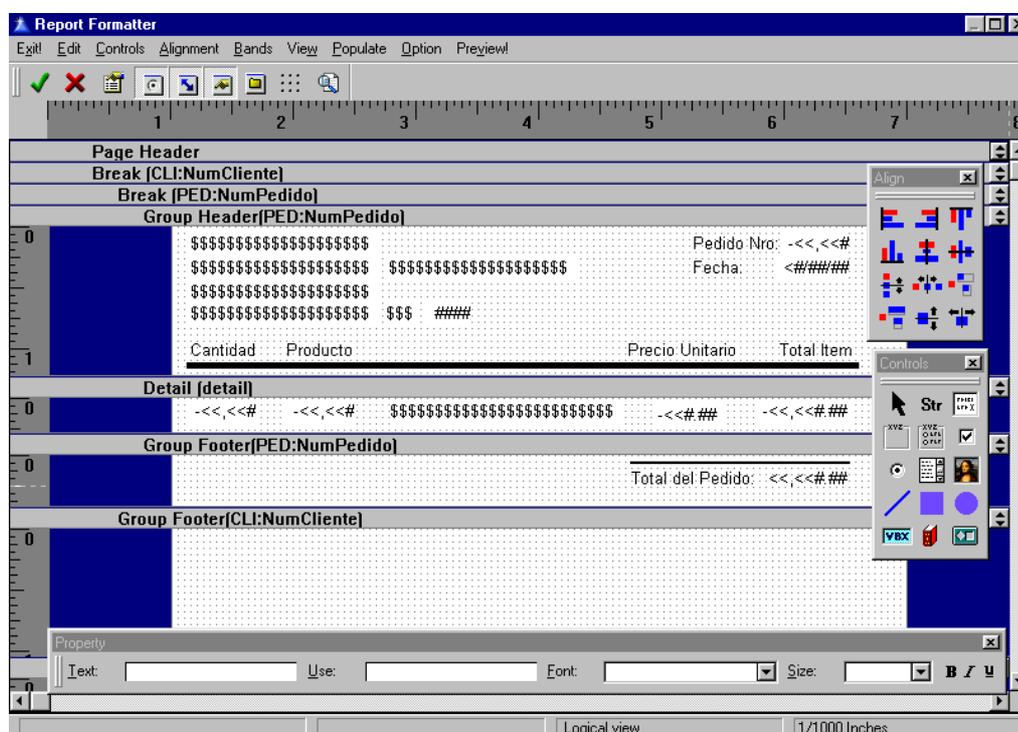


13. Oprima el botón **OK**.

Así se sumarán todos los contenidos del *ItemTotal* del pedido, y volverá a cero cuando cambie el valor en *PED:NumPedido*.

Colocación de una línea sobre el total

1. Elija **Controls** ➤ **Line**, o pique sobre la herramienta *Line* de la caja de herramientas **Controls**.
2. Dé CLIC dentro de la banda **Group Footer (PED:NumPedido)**, sobre los controles que acaba de colocar.
3. Ajuste la línea ARRASTRANDO las asas hasta que aparezca a lo ancho de todos los controles.
4. Dé CLIC DERECHO y elija **Properties** del menú contextual.
5. Escriba 20 en el campo de ancho de línea (**Line Width**).
Así se engrosa un poco la línea.
6. Oprima el botón **OK** para cerrar la caja de diálogo **Line Properties**.



Llenado de la banda de pie de grupo “Clientes”

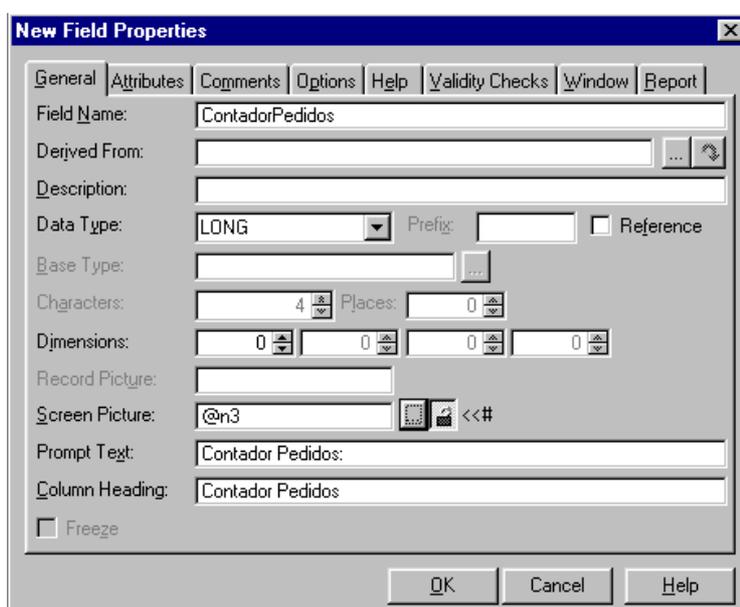
Colocación del texto constante.

1. Elija **Controls** ➤ **String**, o "pique" sobre las herramientas **String** de la caja de herramientas **Controls**.
2. Dé CLIC dentro de la banda **Group Footer (CLI:NumCliente)**, a mitad de la banda.
3. Escriba *Resumen de Pedidos para:* en el campo **Text** de la caja de herramientas **PropertyBox**.
4. Elija **Controls** ➤ **String**, o "pique" sobre la herramienta **String** de la caja de herramientas **Controls**.
5. Dé CLIC dentro de la banda **Group Footer (CLI:NumCliente)**, debajo del texto que acaba de colocar.
6. Escriba *Total Notas de Pedido:* en el campo **Text** de la caja de herramientas **PropertyBox**, y oprima TAB.

Colocación de los campos de total

1. Elija **Controls** ➤ **String**, o "pique" sobre la herramienta **String** de la caja de herramientas **Controls**.
2. Dé CLIC dentro de la banda **Group Footer (CLI:NumCliente)**, a la derecha del texto que acaba de colocar.

3. Dé CLIC DERECHO y elija **Properties** del menú contextual.
4. Marque la casilla **Variable String**.
5. Oprima el botón de puntos suspensivos (...) del campo **Use**.
6. Destaque *LOCAL DATA InformePedidos* en la lista de **Files**, y oprima el botón **New**.
7. Escriba *ContadorPedidos* en el campo **Name**.
Este campo imprimirá el número de pedidos impresos para cada cliente.
8. Elija *LONG* de la lista desplegable **Data Type**.
9. Escriba *@N3* en el campo **Picture**, y oprima el botón **OK**.



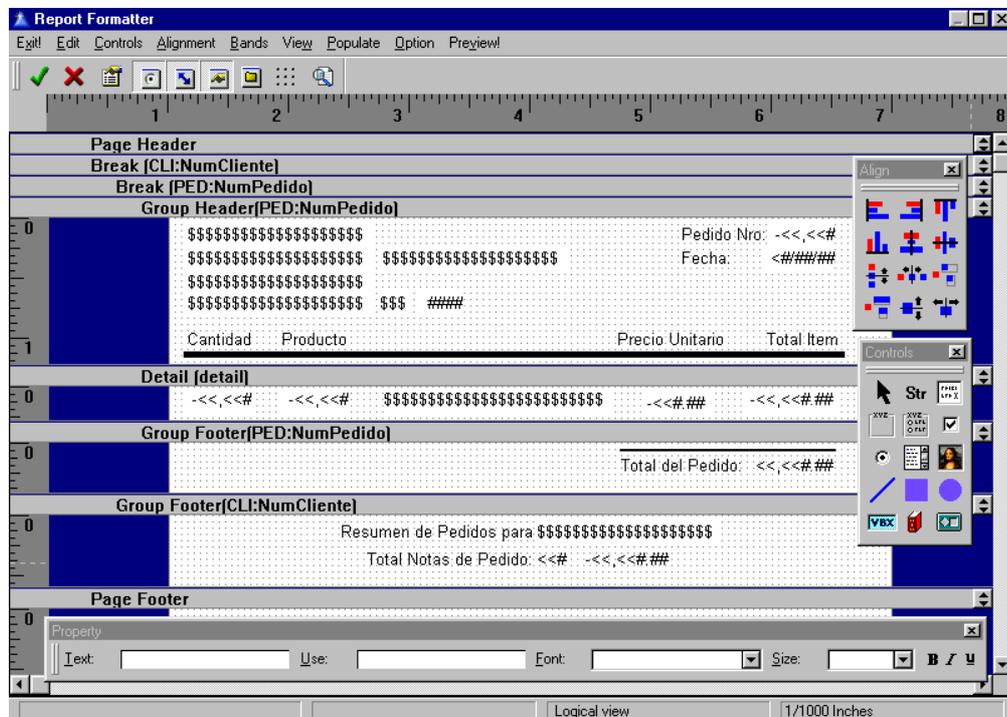
10. Seleccione *Count* (contar) de la lista desplegable **Total type**.
11. Seleccione *CLI:NumClienteBreak* de la lista desplegable **Reset On**.
Este es el mismo tipo de campo totalizador que colocamos en el pie del grupo *PED:NumPedido*, pero solamente se restaurará cuando cambie *CLI:NumCliente*.
12. Seleccione la lengüeta **Extra**.
13. Destaque *PED:NumPedidoBreak* de la lista de marcadores (**Tallies**), y oprima el botón **OK**.
Este campo totalizador contará el número de pedidos que se imprimen de cada cliente. La lista de marcadores (Tallies) le permite seleccionar los puntos en que se incrementa el total. Seleccionado *PED:NumPedidoBreak* de la lista, la cuenta solo se incrementará al comenzar cada nuevo pedido.
14. Elija **Controls** ➤ **String**, o "pique" sobre la herramienta **String** de la caja de herramientas **Controls**.
15. Dé CLIC dentro de la banda **Group Footer (CLI:NumCliente)**, a la derecha de la cadena de caracteres que acaba de colocar.

16. Dé CLIC DERECHO y elija **Properties** del menú contextual.
17. Marque el casillero **Variable String**.
18. Oprima el botón de puntos suspensivos (...) del campo **Use**.
19. Destaque *LOCAL DATA InformePedidos* en la lista **Files**, y elija *ItemTotal* en la lista **Fields** y oprima el botón **Select**.
20. Elija *Sum* de la lista desplegable **Total Type**.
21. Seleccione *CLI:NumClienteBreak* de la lista desplegable **Reset On**.
Este es el mismo tipo de campo totalizador que ubicamos en el pie de grupo *PED:NumPedido*, pero solamente se restaurará cuando cambie el número *CLI:NumCliente*.
22. Oprima el botón **OK**.

Colocación del campo y salida

1. Elija **Populate** > **Multiple Fields**, o dé CLIC sobre la herramienta *Dictionary Field* en la caja de herramientas **Controls**.
2. Destaque *Cientes* en la lista **Files** y seleccione *CLI:Empresa* en la lista **Fields** y oprima el botón **Select**.
3. Dé CLIC dentro de la banda **Group Footer (CLI: NumCliente)**, a la derecha del texto *Resumen de Pedidos para:* que se ha colocado.
4. Oprima el botón **Cancel** para salir de este modo de operación.

El diseño del informe está completo.



5. Elija **Exit!** para volver a la caja de diálogo **Procedure Properties** (asegúrese de guardar el diseño).

Adición de una fórmula

Para hacer que el campo *ItemTotal* contenga el monto adecuado de cada registro de *Detalle* del pedido, debe añadir una fórmula al procedimiento.

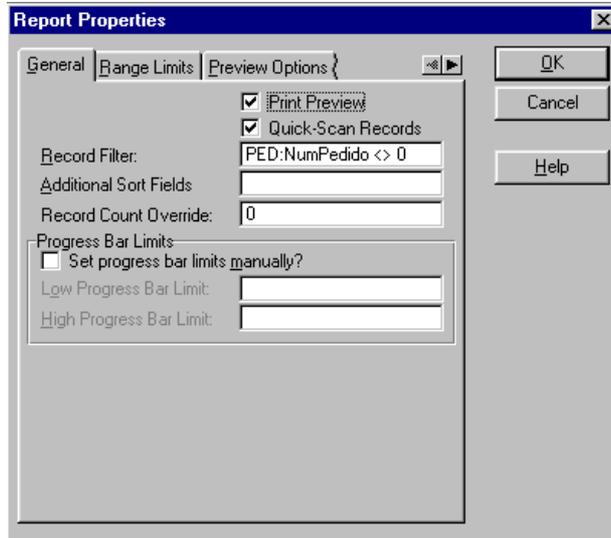
1. Oprima el botón **Formulas** en la caja de diálogo **Procedure Properties**.
2. Oprima el botón **New** en la caja de diálogo **Formulas**.
Aparece el Editor de Fórmulas.
3. Escriba *Item Total Formula* (u otra expresión similar) en el campo **Name**.
4. Oprima el botón de puntos suspensivos (...) en el campo **Class**.
5. Destaque *Before Print Detail* ("Antes de imprimir el detalle") en la lista **Template Classes**.
La clase *Before Print Detail* le indica a la plantilla de Informe que realice el cálculo cada vez que está por imprimir un Detalle.
6. Oprima el botón de puntos suspensivos (...) en el campo **Result**.
7. Destaque LOCAL DATA *InformePedidos* en la lista **Files**, seleccione *ItemTotal* de la lista **Fields**, y oprima el botón **Select**.
8. Oprima el botón **Data** en el grupo **Operands**.
9. Destaque el archivo *Detalle* en la lista **Files**, seleccione *DETAL:Cantidad* de lista **Fields**, y oprima el botón **Select**.
Esto coloca el campo seleccionado dentro de **Statement**. Este último contiene la expresión que se está construyendo. También es posible escribir directamente dentro de este campo, si se desea.
10. Oprima el botón * en el grupo **Operators**.
11. Oprima el botón **Data** en el grupo **Operands**.
12. Destaque el archivo *Detalle* en el listado **Files**, seleccione *DETAL:ProdPrecio* de lista **Fields**, y oprima el botón **Select**.
13. Oprima el botón **Check** para verificar la sintaxis de la expresión.
14. Oprima el botón **OK** para cerrar el Editor de Fórmulas.
15. Oprima el botón **OK** para cerrar la caja de diálogo **Formulas** y volver a **Procedure Properties**.

Adición de un filtro de registros

Para asegurarnos que el informe solamente imprime Notas de Pedido para las empresas que tienen pedidos, incluiremos un filtro de registros(*record filter*).

1. Oprima el botón **Report Properties** en la caja de diálogo **Procedure Properties**.
Aparece la caja de diálogo **Report Properties**.

2. Escriba *PED:NumPedido* <> 0 en el campo **Record Filter**.



Esto eliminará a todos los clientes que no hayan hecho pedidos. Internamente, el Report Template genera una estructura VIEW (véase la Language Reference). Esta estructura VIEW realiza un “Join externo” de los archivos del árbol de archivos. “Join externo” (outer join) es una expresión común en la teoría de bases de datos relacionales: significa que la VIEW invocará todos los registros del archivo Padre, sea que hayan registros Hijos vinculados o no. Si encuentra un registro Padre sin Hijo, los campos de éste último se consideran en blanco o “cero”, aunque hay datos válidos en el archivo Padre. Por lo tanto, esta es la condición que verificaremos.

La expresión *PED:NumPedido* <> 0 verifica si el campo del mismo nombre tiene un valor distinto a cero. Dado que éste es el campo clave del archivo de Pedidos que crea el vínculo al registro relacionado de *Clientes*, debe contener un valor, si existen Pedidos para el Cliente actual. Si *PED:NumPedido* está en cero, el actual registro de Cliente se saltea (es “filtrado”). Esto elimina la impresión de registros Padre sin Hijos vinculados (en este caso, Clientes sin Pedidos).

**Nota: si tiene dificultad para poner los signos < y > (ALT+60 y ALT+62), puede utilizar la expresión equivalente “NOT = ”. Quedaría así:
PED:NumPedido NOT = 0.**

3. Oprima el botón **OK**.

Cambio de la Ventana indicadora

1. Oprima el botón **Window** en la caja de diálogo **Procedure Properties**.
2. Escriba *Cargando los pedido...* en el campo **Text** de la caja de herramientas **Property**.
3. Elija **Exit!** para regresar a la caja de diálogo **Procedure Properties**.

Salida y guardado

1. Oprima el botón **OK** en la caja de diálogo **Procedure Properties** para cerrarla.
2. Elija **File > Save**, u oprima el botón *Guardar* en la caja de herramientas.

Informe de alcance limitado (Range Limited Report)

Ahora, limitaremos el alcance de los registros para imprimir.

Creación del informe

1. Destaque el informe *InformePedidos*.
2. Copiaremos el procedimiento: Elija **Procedure** > **Copy...**
Aparece la caja de diálogo **New Procedure**.
3. Escriba *InformePedidosCliente* y oprima el botón **OK**.
El procedimiento copiado aparece en el árbol de la aplicación, reemplazando el procedimiento “por hacer” (“ToDo”) que estaba en su lugar.

Modificación del nuevo informe

1. Destaque el procedimiento *InformePedidosCliente*.
2. Dé CLIC DERECHO y elija **Properties** del menú contextual.
3. Oprima el botón **Embeds**.
4. Presione el botón **Contract All**.
Esto hará más fácil localizar el punto embebido deseado.
5. Ubique la carpeta **Local Objects**, y dé CLIC en el signo + para expandir su contenido.
Los templates ABC generan código orientado a objetos. Cada procedimiento instancia un conjunto de objetos que son derivados de la Biblioteca ABC. La carpeta de Objetos Locales (**Local Objects**) muestra todos los objetos y métodos que pueden ser sobre-escritos en el procedimiento, simplemente embebiendo código propio, para incrementar la funcionalidad de la Biblioteca ABC, Para mayor información sobre esta poderosa técnica, vea los artículos *Easy into OOP* y *Object Oriented Programming* en el manual *Programmer's Guide*.
6. Ubique la carpeta **ThisWindow**, dé CLIC para expandirla.
7. Ubique la carpeta **Init PROCEDURE(),BYTE,VIRTUAL**, y dé CLIC para expandirla.
8. Ubique la carpeta **CODE**, y de CLIC para expandirla.
9. Ilumine **Open Files** y oprima el botón **Insert**.
Este punto embebido está al principio del procedimiento, antes que el informe haya comenzado a procesar la información. Es importante que los archivos para el informe estén ya abiertos porque llamaremos otro procedimiento para que el usuario seleccione un registro de Clientes. Si los archivos del informe no estuvieran ya abiertos, el procedimiento que llamamos abriría el archivo Clientes para su propio uso, y luego lo cerraría, con lo que perderíamos la información que necesitamos para el informe. Esto tiene que ver con los múltiples hilos de ejecución y la Interfaz de Múltiples Documentos (MDI); para más datos, consulte THREAD en *Language Reference*.
10. Destaque *Source* (“Fuente”) y oprima el botón **Select** para llamar al Editor de Texto.

11. Escriba el siguiente código:

```
Global Request = SelectRecord
```

Este código configura un procedimiento *Browse* para elegir un registro (habilita el botón **Select** del procedimiento *Browse*).

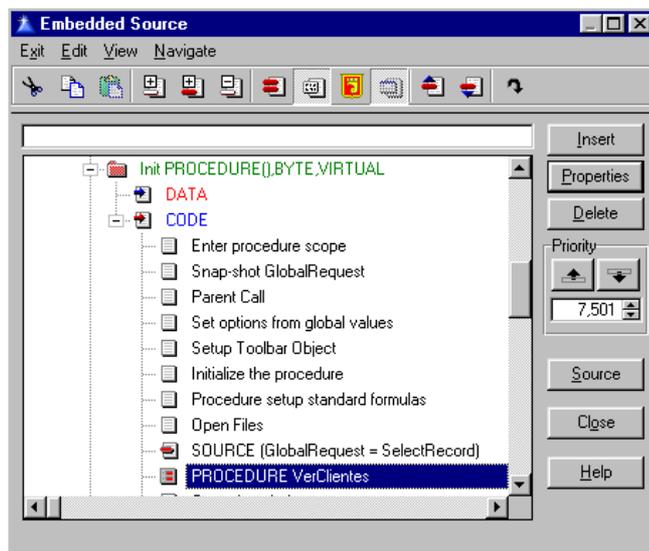
12. Elija **EXIT!** para volver a la caja de diálogo **Embedded Source**.

13. Destaque el código fuente (SOURCE) que acaba de escribir y oprima el botón **Insert**.

14. Destaque *Call a procedure* y oprima el botón **Select**.

15. Elija *VerClientes* de la Lista desplegable y oprima el botón **OK**.

Así se generará una llamada al procedimiento *VerClientes* (*browse*) para permitir que el usuario seleccione el Cliente cuyos pedidos va a imprimir



Adviértase que aparecen ahora dos entradas bajo el punto de inserción. En cada punto, puede poner tantos ítems como desee, mezclando plantillas de código (Code Templates) con sus propias llamadas al código fuente (SOURCE) o a procedimientos (PROCEDURE). También puede mover los ítems individuales en torno al punto de inserción, cambiando el orden lógico de la ejecución (el que se ve primero es el primero en ejecutarse). Tenga en cuenta que moverlos cambiará la opción de Prioridad (**Priority**) asignada al ítem movido si intenta mover un ítem de más alta prioridad frente a otro con una prioridad menor.

16. Oprima el botón **Close** para regresar a la caja de diálogo **Procedure Properties**.

Configuración del límite de alcance

1. Oprima el botón **Report Properties**.

Aparece la caja de diálogo **Report Properties**. Esta caja le permite configurar filtros de registros o límites de alcance (además de los Hot Fields y filtros de detalle): véase la *User's Guide*.

Los filtros de registros (*record filters*) y los límites de alcance o de búsqueda (*range limites*) son muy similares. Un filtro de registros es una expresión condicional para eliminar registros indeseados del informe, mientras que un límite de alcance limita los registros impresos a solamente aquellos que coinciden en sus claves. Ambos pueden usarse para crear informes

sobre un subconjunto de los archivos. La diferencia es que el límite de alcance opera sobre una clave, mientras que los filtros, no. Así, éstos últimos son muy flexibles; y los límites de alcance, muy rápidos. Se pueden usar simultáneamente, seleccionando un límite de alcance y luego filtrando los registros innecesarios dentro de ese límite.

2. Seleccione la lengüeta **Range Limits**.
3. Oprima el botón de puntos suspensivos (...) de **Range Limit Field**.
4. Destaque *CLI: NumCliente*, y oprima el botón **Select**.
5. Deje el **Range Limit Type** (tipo de límite de alcance) como “valor actual” (Current Value), y oprima el botón **OK**.

Esto indica que cualquier valor (Current Value) presente en el campo al iniciarse el informe, será su límite de alcance. Dado que el usuario elegirá un registro de Clientes en el procedimiento *VerClientes*, el valor correcto estará en el campo *CLI:NumCliente* al comenzar.

Salir y Guardar

1. Oprima el botón **OK** en la caja de diálogo **Procedure Properties** para cerrarlo.
2. Elija **File > Save**, u oprima el botón *Guardar* en la barra de herramientas para grabar su trabajo.

Informe Pedido Individual

A continuación, imprimiremos un pedido individual elegido desde la lisa de visualización (browse) de pedidos.

Creación del informe

1. Destaque el procedimiento *InformePedidosCliente*.
2. Elija **Procedure > Copy...**
Aparece la caja de diálogo **New Procedure**.
3. Escriba *InformeUnPedido* en la caja de ingreso de datos, y oprima el botón **OK**.
4. Oprima el botón **Same** (el mismo nombre) en la caja de diálogo **Procedure Name clash**.
El procedimiento copiado aparece en el árbol de aplicaciones, aislado. Al terminar el informe, “conectaremos las líneas”.

Borrado del código insertado

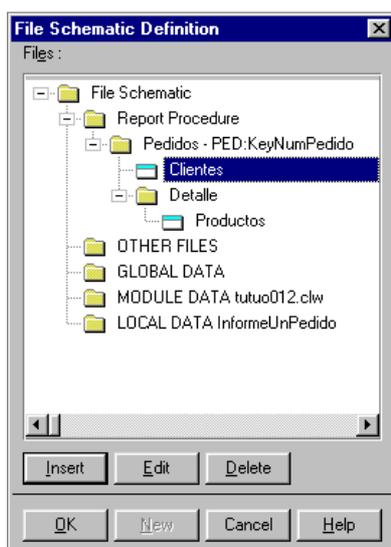
1. Destaque el procedimiento *InformeUnPedido*.
2. Dé CLIC DERECHO y elija **Properties** del menú contextual.
3. Oprima el botón **Embeds**.
4. Oprima el botón **Next Filled** (en el extremo derecho de la barra de herramientas).
5. Presione el botón **Delete**.
6. Responda **Yes** (o “Sí”) a la pregunta sobre si está seguro (**Are you sure?**)

7. Presione nuevamente **Delete** y responda **Yes** (o “Sí”) a la pregunta sobre si está seguro.
8. Oprima el botón **Close**.

Cambio del árbol de archivos

En primer lugar, necesitamos cambiar el orden del árbol de archivos. Terminaremos con los mismos archivos, pero en lugar del archivo *Clientes* como archivo Primario (el primero del árbol), necesitamos que el archivo *Pedidos* sea el archivo Primario para este procedimiento, para poder limitar fácilmente el alcance a un único pedido.

1. Oprima el botón **Files**.
2. Destaque el archivo *Clientes*, y oprima el botón **Delete**.
Esto hace que todos los archivos desaparezcan.
3. Ilumine la carpeta “ToDo”, y oprima el botón **Insert**.
4. Seleccione el archivo *Pedidos* de la caja de diálogo **Insert File**, y oprima el botón **Select**.
5. Oprima el botón **Edit**.
6. Ilumine *KeyNumPedido* en la caja de diálogo **Change Access Key**, y oprima el botón **Select**.
7. Destaque el archivo *Pedidos*, y oprima el botón **Insert**.
8. Seleccione el archivo *Detalle* de la caja de diálogo **Insert File**, y oprima el botón **Select**.
9. Destaque el archivo *Detalle*, y oprima el botón **Insert**.
10. Seleccione el archivo *Productos* de la caja de diálogo **Insert File**, y oprima el botón **Select**.
11. Ilumine de nuevo el archivo *Pedido*, y oprima el botón **Insert**.
12. Seleccione el archivo *Clientes* de la caja de diálogo **Insert File**, y oprima el botón **Select**.
En este punto, el árbol de archivos debería verse así:



Hemos seleccionado los mismos archivos, pero ahora el archivo primario es *Pedidos*, y se hará una consulta al registro vinculado de *Clientes*. Esto es importante, porque debemos

limitar este informe a un único pedido y eso sería mucho más difícil de hacer si el archivo primario fuese *Cientes*.

13. Oprima el botón **OK**.

Configuración del límite de alcance

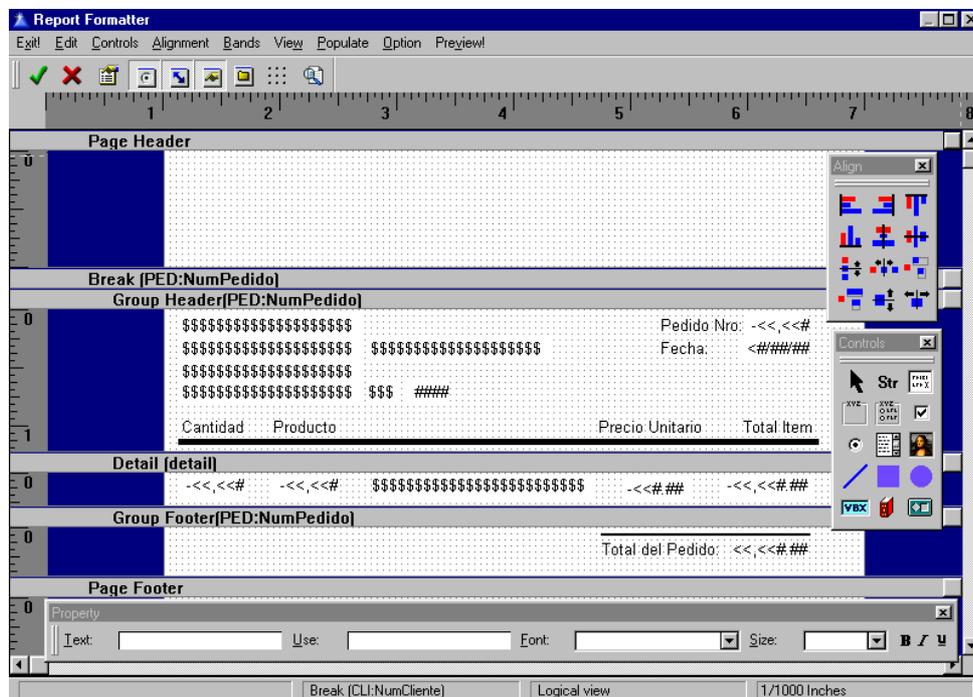
1. Oprima el botón **Report Properties**.
2. Seleccione la lengüeta **Range Limits**.
3. Oprima el botón de puntos suspensivos (...) junto a **Range Limit Field**.
4. Destaque *PED: NumPedido* y oprima el botón **Select**.
5. Deje “Valor Actual” (*Current value*) como **Range Limit Type** y oprima el botón **OK**.

Current Value indica que el valor que esté en este campo cuando empieza el informe es el valor al que este se limitará. Dado que el usuario lo ejecutará desde el procedimiento VerPedido, el valor correcto estará en el campo PED:NumPedido cuando empieza el informe.

Modificación del nuevo informe

Ahora tenemos que cambiar el informe mismo, para que imprima un único pedido.

1. Oprima el botón **Report**.
2. Dé CLIC DERECHO sobre la banda **Break(CLI:NumCliente)** y elija **Delete** del menú contextual.



Esto quita no solamente el salto de grupo (*Group Break*), sino también el Pie de Grupo correspondiente, lo que nos deja con un informe parecido a éste:

3. Elija **Exit!** para regresar a la caja de diálogo **Procedure Properties**.

Salida y guardado del trabajo

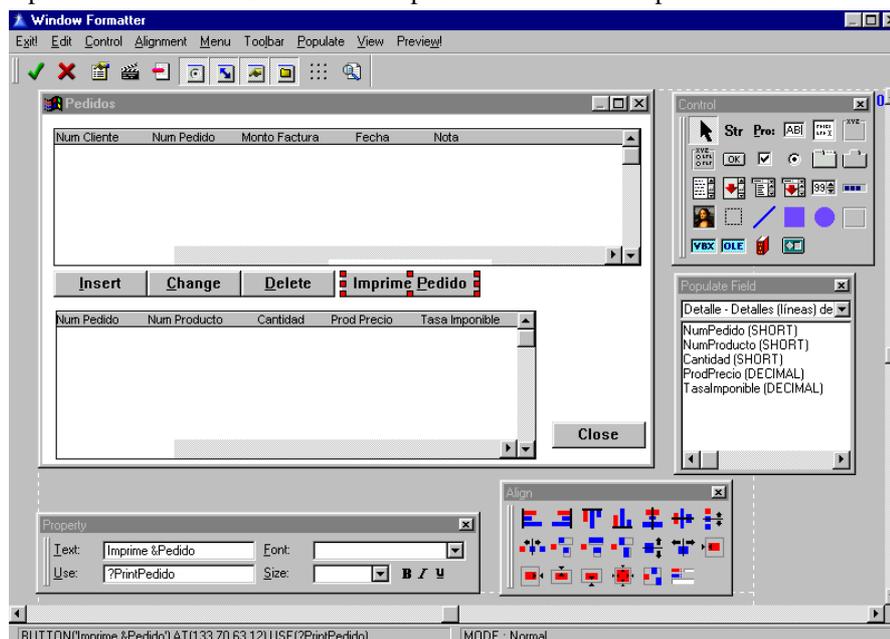
1. Oprima el botón **OK** en la caja de diálogo **Procedure Properties** para cerrarla.
2. Elija **File > Save**, u oprima el botón *Guardar* de la barra de herramientas para guardar su trabajo.

Conectando las líneas

1. Destaque el procedimiento *VerPedidos*.
2. Dé CLIC DERECHO y elija **Window** del menú contextual.
3. Elija **Populate > Control Template**, o de CLIC sobre la herramienta *Control Template* de la caja de herramientas **Control** (en el extremo inferior izquierdo).
4. Ilumine **BrowsePrintButton** y presione el botón **Select**.
5. Ilumine **BrowsePedidos** y presione el botón **Select**.
6. Dé CLIC a la derecha del botón *Delete* para ubicar el nuevo botón control.
7. Dé CLIC DERECHO sobre el nuevo botón y escoja **Properties** del menú contextual.
8. Escriba *Imprime &Pedido* en el campo **Field**.
9. Escriba *?ImprimePedido* en el campo **Use**.
10. Elija la lengüeta **Actions**.
11. Seleccione *InformeUnPedido* de la lista desplegable **Print Button**.

Este Control Template está específicamente diseñado para llamar a un reporte de alcance limitado según el registro iluminado en la caja de listado seleccionada (Browse on Pedidos). El buffer del registro del archivo Pedidos contendrá el valor correcto para que funcione el alcance limitado al *Current Value* en el informe para un pedido. Además, la acción queda automáticamente añadida al menú contextual del Browse.

12. Oprima el botón **OK**. El diseño de la pantalla deberá verse aproximadamente así:



13. Elija **Exit!** para volver al Arbol de la Aplicación.
14. Elija **File > Save**, u oprima el botón *Guardar* de la barra de herramientas para guardar su trabajo.

¿Qué sigue?

Felicitaciones, usted ha llegado al final del cursillo sobre el Generador de Aplicaciones. ¡Bienvenido o bienvenida a la creciente comunidad de desarrolladores Clarion!.

Aunque la aplicación que hemos realizado no tiene la amplitud de un "programa comercial", ha servido para demostrar el proceso normal de usar el Generador de Aplicaciones y todas sus herramientas para crear una aplicación que realiza algunas tareas bastante elaboradas. En el camino, usted ha usado la mayor parte del conjunto de herramientas de alto nivel de Clarion, y ha visto cuánto puede éste hacer por usted, sin necesidad de escribir código. También ha visto cómo *un poquito* de código fuente insertado puede añadir funcionalidad extra al código generado por los templates, y qué fácilmente pueden pasarse por alto las clases preconfiguradas de la biblioteca ABC.

Una breve recorrida por los recursos a su disposición

Usted tiene muchos recursos a mano para ayudarle a programar con Clarion. He aquí una breve recorrida por dos de los más importantes de que se dispone:

1. Elija **Help > Contents**.

Esta es la página de Contenidos del extenso sistema de ayuda en línea.

2. Oprima el botón **How do I ...?** ("¿Cómo hago para ...?").

Esto abre el archivo de ayuda de Clarion, y lo lleva a una sección de preguntas comunes, con sus respuestas. Esta lista de temas es el primer lugar donde buscar cuando usted se haga cualquier pregunta sobre la programación en Clarion, que comienza con "¿Cómo hago para...?". Estos temas resuelven muchas de las preguntas comunes que tienen los recién llegados a Clarion. A menudo, las respuestas las encontrará allí.

3. Oprima el botón **Back**, y luego el botón **Guide to Examples**.

Este tema ofrece saltos hipertextuales al tratamiento de todos los programas de ejemplo que vienen con Clarion. Allí encontrará variadas sugerencias, trucos, y técnicas demostradas en los ejemplos, para adaptarlos a sus propios programas.

4. Oprima el botón **Back**, y luego el botón **Late Breaking News**.

Este tema le ofrece la información más reciente y actualizada sobre la versión de Clarion más reciente que haya instalado. Siempre debe consultar esta sección cuando haya una actualización del programa. Hay generalmente detalles de último minuto que se documentan solamente en esta sección. Esto hace muy recomendable la lectura.

5. Cierre la ayuda en línea y, sin salir del programa, active el administrador de archivos del sistema operativo (el Explorador de Window 95,98, o NT).

6. Ponga el CD de Clarion en el dispositivo de CD, y navegue hasta el subdirectorio \DOCS.

7. Dé DOBLE CLIC sobre el archivo C5-UG.PDF (tiene que haber instalado Acrobat Reader del CD Clarion para leer este archivo).

Esto llamará a la *User's Guide* en el Acrobat Reader. Este es el libro íntegro, en línea, y disponible desde el CD (o disco rígido, si lo copia allí). Esto significa que por el tamaño y el peso de un CD puede tener disponible toda la documentación de Clarion en cualquier parte (Aprovechando las búsquedas de texto de Acrobat). Advierta que a la izquierda hay un grupo de "saltos" disponibles desde la Tabla de Contenidos. También puede ir directamente a cualquier página individual que elija (y ahora le enseñaremos cómo).

8. Oprima el control **VCR** de "fin de documento"(el triángulo que apunta a la derecha con una línea vertical en el extremo), y oprima un par de veces el botón de control de "página previa"(se ve como un triángulo que apunta a la izquierda).

Ahora está mirando a la posición del *Índice* de la *User's Guide*.

9. Seleccione la herramienta "zoom in" (una lupa con signo +), y dé CLIC en cualquier parte de la página (si el cursor cambia a un "dedo apuntando" entonces irá directamente al ítem al que apunta, en lugar de hacer un acercamiento).

Puede hacer "zoom in" a cualquier texto que elija utilizando esta herramienta. Anote el número de página de cualquier ítem que desee.

10. Dé CTRL+CLIC para volver a la vista anterior.

11. Oprima el CTRL+5 para ir a la caja de diálogo **Go To Page**, ingrese el número de página que anotó, y oprima el botón **OK**.

Ahora puede ver lo fácil que es ir a cualquier lugar que desee.

12. Cierre el archivo y salga de Acrobat Reader.

¿Dónde aprender más?

¿ A dónde debería ir desde aquí para aprender más? Los mejores lugares (aparte de crear una aplicación propia) son:

- ♦ El *Application HandBook* explica los Clarion ABC Templates y la biblioteca ABC; todas las herramientas que le ofrece el Generador de Aplicaciones.
- ♦ El *Cursillo sobre el Lenguaje Clarion* en el capítulo siguiente es el mejor lugar para empezar a aprender el lenguaje.
- ♦ Los usuarios registrados de la Argentina y otros países de habla hispana tienen acceso a apoyo técnico vía Internet en la dirección <http://www.unisoft.com.ar> (o, también, www.clarion.com.ar. Correo electrónico: soporte@unisoft.com.ar) mediante Servicios de Newsgroup y FTP. Un equipo de experimentados técnicos locales (CST: Clarion Support Team Argentina) responderá a sus dudas. Pueden también dirigirse al fax: (+541) 374-9469.
- ♦ TopSpeed ofrece seminarios educativos en varias localidades. Llame a Servicio al Cliente a los números (800)354-5444 o (954)785-4555 para informarse.
- ♦ En CompuServe escriba GO TOPSPEED para ingresar al foro de apoyo técnico de TopSpeed, o - En Internet, ingrese al grupo de noticias **comp.lang.clarion**, mediante cualquier servidor de Usenet (¡ Muy recomendado!).

Muy buena suerte, y continúe trabajando: ¡el poder de programación que Clarion pone en sus manos va creciendo a medida que aprende más sobre él!

13 - CURSILLO SOBRE LENGUAJE CLARION

Clarion como lenguaje de programación

La base del entorno de desarrollo de aplicaciones Clarion es el lenguaje de programación Clarion. Se trata de un lenguaje de cuarta generación (4GL) que se orienta a aplicaciones empresariales y a propósitos generales. La primera orientación significa que contiene estructuras de datos y órdenes altamente optimizadas para la programación y el mantenimiento de bases de datos. También es de propósito general porque es un lenguaje compilado (no interpretado) y tiene un conjunto de órdenes funcionalmente similar a otros lenguajes de tercera generación (tales como c/c++, Modula-2, Pascal, etc.). Tratamos más ampliamente esta distinción en las primeras páginas de la *Language Reference*.

A esta altura, usted debe de haber terminado todos los cursillos sobre el Generador de Aplicaciones en los capítulos precedentes. El propósito del presente cursillo es presentarle los aspectos fundamentales del lenguaje Clarion, particularmente en su orientación a la programación empresarial utilizando el paradigma de manejo de eventos propios de Windows. Las palabras claves (language keywords) del lenguaje se muestran EN MAYUSCULAS, y este cursillo se concentra en el uso de cada palabra clave y su interacción con otros elementos del lenguaje solamente en el contexto específico en la que se está utilizando. Debe consultarse siempre la *Language Reference para una explicación más completa de cada palabra clave individual y sus capacidades*.

Cuando complete este breve cursillo, se habrá familiarizado con:

- ◆ La estructura básica de un programa Clarion.
- ◆ La estructura de código más común para el manejo de eventos.
- ◆ Cómo compilar y linkear programas escritos a mano.

Programación en base a eventos

Los programas de Windows se manejan mediante eventos. El usuario causa un evento dando un CLIC con el *mouse* sobre un control de la pantalla, u oprimiendo una tecla. Cada acción del usuario hace que Windows envíe un mensaje (o *evento*) al programa propietario de la ventana, indicándole lo que ha hecho el usuario.

Una vez que Windows ha enviado el mensaje que indica un evento, el programa tiene la oportunidad de manejarlo de modo apropiado. Esto significa que el paradigma de programación de Windows es exactamente lo opuesto a lo que ocurre en DOS. En Windows el sistema operativo le dice al programa lo que el usuario ha hecho, en lugar de que el programa le indique al sistema operativo lo que debe hacer.

Este es el concepto más importante de la programación en Windows: que el usuario está siempre al control (o debería estarlo). Por lo tanto, estos programas son reactivos, más que proactivos; siempre tratan con lo que el operador ha hecho, en lugar de indicarle qué hacer después. El ejemplo más común de esto es la caja de diálogo de ingreso de datos. En la mayor parte de los programas DOS el usuario debe seguir un camino de campo en campo para ingresar datos. Siempre deben ingresar datos en un campo antes de seguir al próximo. Esto hace que la validación de datos sea simple: basta con ejecutarla apenas el usuario ha dejado el campo.

En los programas Windows el operador puede utilizar el *mouse* o un atajo de teclado para ir de un control a otro, en cualquier momento, sin seguir un orden determinado, y omitiendo pasar por los controles que le parezca. Por lo tanto, se debe ejecutar dos veces la validación de los ingresos de datos, para estar seguros de que se ejecuta al menos de una vez: cuando el usuario acaba de ingresar datos a un control, y nuevamente cuando oprime OK para salir de la caja de diálogo. Si no se ejecutara entonces, podrían omitirse datos importantes. Esto hace que los programas Windows sean reactivos, antes que proactivos.

Hola Windows

Tradicionalmente, todos los cursillos sobre un lenguaje de programación empiezan por la creación de un programa tipo: “Hola, mundo”, y éste también.

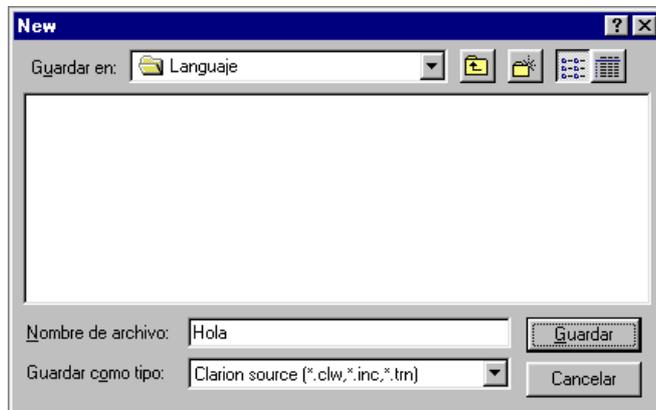
Punto inicial:

El entorno Clarion debe estar abierto, y la aplicación TUTORIAL.APP, cerrada.

Creación del archivo fuente

1. Mediante la herramienta apropiada de su sistema operativo (Administrador de Archivos, Explorador, etc.), cree un nuevo directorio llamado *Language* bajo el subdirectorio **Clarion5**. (En Windows 95, puede oprimir TECLA-DE-VENTANA+E para acceder al Explorador).
2. Regrese a Clarion
3. Elija **File > New > Source**.

Aparece la caja de diálogo **New**. Es una caja de diálogo Windows estándar *Open File*, que le permite cambiar el directorio y escribir el nombre del archivo.



4. Seleccione el directorio `\CLARION5\LANGUAGE`.
5. Escriba *Hola* en el campo **File Name**.
6. Oprima el botón **Save**.

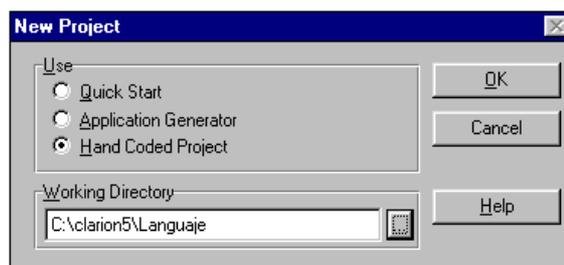
Esto crea un archivo *HOLA.CLW* vacío (.CLW es la extensión estándar para los archivos de código fuente), y abre el Editor de texto.

Creación del archivo de Proyecto

1. Elija **Project** ➤ **New**.

Aparece la nueva caja de diálogo **New Project**. Cuando se codifica completamente a mano, también se necesita un archivo de proyecto (*HOLA.PRJ*) que controle el proceso de compilación y linkeado.

2. Elija el botón de radio de proyecto escrito a mano (**Hand Coded Project**).



3. Escriba `C:\CLARION5\LANGUAGE` en la casilla del directorio de trabajo (**Working Directory**) u oprima el botón de puntos suspensivos para elegir el directorio en la caja de diálogo Change Working Directory.

Cuando se escribe un programa a mano, el directorio que contiene el archivo de proyecto(.PRJ) se convierte en el directorio de trabajo del proyecto. Cuando se selecciona, Clarion cambia automáticamente a ese directorio.

4. Oprima el botón **OK**.

Aparece la caja de diálogo **New Project File**.

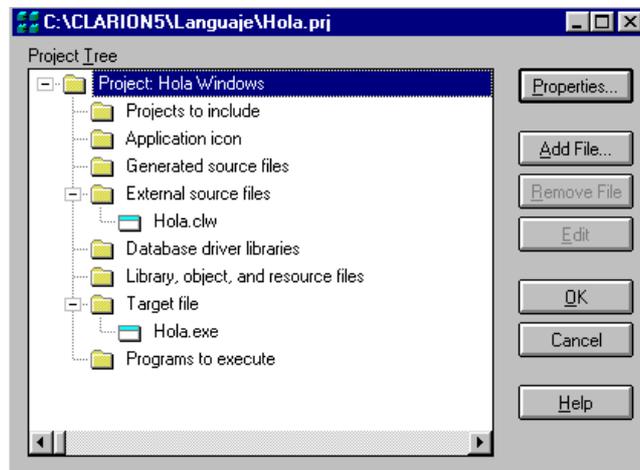
5. Escriba *Hola Windows* en la caja **Project Title**, y oprima TAB.
6. Escriba *HOLA.CLW* en la caja **Main File**, y oprima TAB.

Una vez que haya nombrado el módulo fuente principal del proyecto, el resto de los controles se llenan automáticamente con los valores preconfigurados.



7. Oprima el botón **OK**.

Aparece la caja de diálogo Project Editor. Esto controla los módulos fuente que el compilador incluye en el proyecto y las bibliotecas que se linkean para crear el programa ejecutable. Cuando se escribe una aplicación completa a mano, debe también actualizar el archivo de Proyecto para incluir todos los módulos de código fuente, controladores de archivo, iconos y bibliotecas externas que utiliza el programa. Vea el capítulo *Using the Project System* en la *User's Guide* para más información sobre cómo mantener el Proyecto.



8. Oprima el botón **OK**.

El foco pertenece ahora al archivo *HOLA.CLW*

.Advierta que la barra de título del entorno de programación dice ahora:

Clarion5 (HOLA.PRJ) – (C:\CLARION5\LANGUAGE\HOLA.CLW). Esta línea siempre indica el proyecto y el directorio de trabajo actuales. Es importante atender a esto cuando se codifica a mano, porque el mero hecho de abrir un archivo fuente en el Editor de texto *no cambia* el proyecto actual para el compilador.

Escriba el programa

1. Escriba el siguiente código:

```
PROGRAM
MAP
END
MyWin      WINDOW('Hola Windows'), SYSTEM
           END
CODE
OPEN(MyWin)
ACCEPT
END
```

Este código comienza con la sentencia **PROGRAM**. Esta debe ser la primera sentencia (que no sea un comentario) en todo programa Clarion. Advierta que la palabra clave **PROGRAM** está sangrada con respecto a **MyWin**. En Clarion, las únicas sentencias que comienzan en la columna uno (1) del código fuente son las sentencias-etiqueta. Por definición, una etiqueta debe empezar en la columna uno (1). La sentencia **PROGRAM** da comienzo con la sección de declaración de datos globales (Global).

Ahora tenemos una estructura **MAP** vacía. La sentencia **END** es una terminación necesaria de la estructura de datos **MAP**. Este tipo de estructura contiene los prototipos que definen tipos de datos de parámetros, tipos de datos de respuesta, y otras opciones que indican al compilador cómo debe manejar las llamadas a procedimientos (tratamos todo esto más adelante). Se necesita una estructura **MAP** cuando se divide el código del programa en procedimientos (**PROCEDURES**). No lo hemos hecho aún, pero igualmente necesitamos la sección ya que hay una sentencia **OPEN** en el código ejecutable.

Cuando el compilador procesa una estructura **MAP**, automáticamente incluye los prototipos en el archivo `\CLARION5\LIBSRC\BUILTINS.CLW`. Este archivo contiene prototipos de casi todos los procedimientos incorporados del lenguaje Clarion (incluyendo la sentencia **OPEN**). Si la estructura **MAP** vacía no se incluyera en el código, el compilador generaría un error en la sentencia **OPEN**.

MyWin es la etiqueta de la estructura de datos **Window** (la "M" debe estar sobre la columna 1). En Clarion, las ventanas se declaran como estructuras de datos, y no se construyen dinámicamente mediante sentencias de código ejecutable, como en otros lenguajes. Este es uno de los aspectos que hacen de Clarion un lenguaje de cuarta generación. Aunque Clarion puede construir cajas de diálogo en forma dinámica durante la ejecución, esto es innecesario. Usando una estructura de datos, el compilador crea el recurso de Windows para cada diálogo, permitiendo mejor *performance* al ejecutarse.

El parámetro ('Hola Windows') sobre la sentencia **WINDOW** define la barra de texto del título de la ventana. La sentencia **SYSTEM** añade un sistema de menú estándar a la ventana. La sentencia **END** es una terminación requerida de la estructura de datos **WINDOW**. En Clarion, todas las estructuras complejas (tanto código ejecutable como datos) deben terminar con un **END** o con un punto (.). Esto significa que el código que sigue es funcionalmente equivalente al anterior:

```

PROGRAM
MAP
END
MyWin    WINDOW('Hola Windows'), SYSTEM
        END
CODE
OPEN(MyWin)
ACCEPT.

```

Aunque funcionalmente son equivalentes, este código se vuelve mucho más difícil de leer en cuanto se añade algo a las estructuras MAP, WINDOW, o ACCEPT. Convencionalmente, utilizamos END para terminar esas sentencias complejas que abarcan varias líneas, colocando el END en la misma columna que la palabra clave a la que está dando terminación, y sangrando todo lo demás dentro de la estructura. Usamos el punto solamente para terminar estructuras de una sola línea, como sentencias IF con una sola cláusula THEN. Esta convención hace que el código sea mucho más fácil de leer, y también que los “terminadores” faltantes sean mucho más fáciles de ubicar.

La sentencia **CODE** se necesita para identificar el principio de la sección de código ejecutable. Los datos (variables de memoria, archivos de datos, estructuras de ventana, informes, etc.) se declaran en una sección de datos (anterior a la sentencia CODE), y las sentencias ejecutables sólo pueden venir después de la sentencia CODE.

Dado que este programa no contiene ningún PROCEDURE (los trataremos más adelante), solamente tiene una sección Global Data seguida por tres líneas de código ejecutable. Las variables declaradas en esta sección son visibles y disponibles para su uso en cualquier parte del programa.

La sentencia **OPEN(MyWin)** abre la ventana, pero no la muestra. La ventana solamente aparecerá en la pantalla cuando se ejecute una sentencia DISPLAY o ACCEPT. Esta característica permite cambiar dinámicamente las propiedades de la ventana, o cualquier control sobre ella, antes de que aparezca en la pantalla.

ACCEPT es el procesador de eventos. La mayoría de los mensajes (eventos) provenientes de Windows son manejados internamente por ACCEPT. Estos son los eventos comunes manejados por la biblioteca de tiempo de ejecución (redibujo de la pantalla, etc.). Solamente aquellos eventos que requieren acción del programa son pasados por ACCEPT al código Clarion. Esto simplifica la programación, permitiendo que el programador se concentre solamente en los aspectos de alto nivel.

La sentencia ACCEPT tiene una sentencia **END** como terminación, lo que significa que es una estructura de código compleja. ACCEPT es una estructura reentrante, o en lazo (loop), que “pasa a través” de todos los eventos que el programador podría querer manejar (ninguno, en este caso; aunque luego veremos algunos), y luego volviendo para manejar el evento siguiente.

Es necesario un lazo ACCEPT para cada ventana abierta en un programa Clarion. Una ventana abierta “se pega” al siguiente lazo ACCEPT que encuentra en el código, y lo toma como su procesador de eventos.

Para este programa, ACCEPT maneja internamente todo lo que hace el menú del sistema (ubicado en la ventana mediante el atributo SYSTEM). Por lo tanto, cuando el usuario utiliza el menú del sistema para cerrar la ventana, ACCEPT pasa automáticamente el control a cualquier sentencia que siga a su terminador END. Dado que no existe ninguna otra sentencia Clarion que ejecutar, el programa termina. Cuando cualquier programa Clarion llega al fin del código ejecutable, se ejecuta un RETURN implícito que, a su vez, vuelve al usuario al sistema operativo.

2. Dé CLIC sobre el botón **Ejecutar (Run)**.

El programa se compila, linkea y ejecuta. La ventana del título muestra el mensaje “Hola Windows”, y debe cerrar la ventana con el menú del sistema.

Hola Windows con controles

Este programa es el más pequeño que puede crearse en Clarion. Ahora lo expandiremos un poquito para demostrar cómo se añaden algunos controles a la ventana y cómo se manejan los eventos generados por esos controles.

Cambio del código fuente

1. Modifique el código para que diga:

```
PROGRAM
MAP
END
MyWin      WINDOW('Hola Windows'), AT(,100,100),SYSTEM           !Cambiado
            STRING('Hola Windows'),AT(26,23),USE(?String1)      !Añadido
            BUTTON('OK'),AT(34,60),USE(?Ok),DEFAULT             !Añadido
            END
CODE
OPEN(MyWin)
ACCEPT
    IF ACCEPTED() = ?Ok THEN BREAK.                             !Añadido
END
```

Nota: El Diseñador de Ventanas está disponible en el Editor de Texto, igual que en el Generador de Aplicaciones. Para llamar al Diseñador de Ventanas, coloque el punto de inserción (el cursor) en cualquier punto dentro de la estructura WINDOW y oprima CTRL+F. Las únicas restricciones son que las herramientas Control Template y Dictionary Field no están disponibles (ya que son específicas del Generador de Aplicaciones).

El cambio es la adición de los controles **STRING** y **BUTTON** a la estructura WINDOW. “STRING” coloca texto constante en la ventana, y BUTTON añade un botón de órdenes.

La única otra adición es la sentencia **IF ACCEPTED() = ?Ok THEN BREAK**. Esta sentencia detecta cuando el usuario ha presionado el botón **OK** y rompe (BREAK) el lazo ACCEPT, terminando el programa. El procedimiento ACCEPTED devuelve el número de campo del control para el cual acaba de generarse el evento EVENT:Accepted. (EVENT:Accepted es una equivalencia o EQUATE, contenido en el archivo \CLARION5\LIBSRC\EQUATES.CLW, que el compilador incluye automáticamente en todos los programas).

?Ok es la equivalencia (*Field Equate Label*) del control BUTTON, definida por el atributo USE del control (consulte *Field Equate Labels* en la *Language Reference*). El compilador

automáticamente iguala ?Ok al número de campo que asigna al control (el uso de las equivalencias facilita la lectura del código).

Cuando el procedimiento ACCEPTED devuelve un valor igual al número de campo asignado por el compilador al botón OK, se ejecuta la sentencia BREAK y se termina el lazo ACCEPT.

2. Dé CLIC sobre el botón **Run**.

El programa se compila y ejecuta. El título de la ventana muestra aun el mensaje “Hola Windows” y , ahora, también lo hace el texto constante en medio de la ventana. Puede cerrar la ventana con el menú del sistema, o el botón OK.

Forma común del código fuente

Hay otras formas de escribir el código en el lazo ACCEPT para lograr lo mismo. Iremos a la forma más directa, debido a que se parece más al estilo de código que produce el Generador de Aplicaciones a partir de los Templates Clarion ABC.

1. Modifique el código de la siguiente manera:

```

PROGRAM
MAP
END
MyWin WINDOW('Hola Windows'), AT(.,100,100),SYSTEM
        STRING('Hola Windows'),AT(26,23),USE(?String1)
        BUTTON('OK'),AT(34,60),USE(?Ok),DEFAULT
END
CODE
OPEN(MyWin)
ACCEPT
    CASE FIELD()                                !Añadido
    OF ?Ok                                       !Añadido
        CASE EVENT()                             !Añadido
        OF EVENT:Accepted                       !Añadido
            BREAK                                 !Añadido
        END                                     !Añadido
    END                                         !Añadido
END
END

```

En este código hay una estructura CASE anidada en otra. Una estructura CASE busca una coincidencia exacta entre la expresión que sigue inmediatamente a la palabra clave CASE y otra expresión que sigue inmediatamente a la cláusula OF (aunque en este caso se muestra solamente una cláusula OF en cada caso, una estructura CASE puede tener tantas como sea necesario.)

La estructura **CASE FIELD()** determina a qué control se aplica el evento actual. Cuando el procedimiento FIELD() devuelve un valor igual al número de campo del botón OK (la equivalencia del campo ?Ok), ejecuta entonces la estructura CASE EVENT().

Esta estructura determina qué evento se generó. Cuando el procedimiento EVENT devuelve un valor igual a EVENT:Accepted (un EQUATE que se contiene en el archivo \CLARION\LIBSRC\EQUATES.CLW) ejecuta entonces la sentencia de romper el lazo (BREAK).

Anidar CASE EVENT() dentro de CASE FIELD() le permite poner todo el código vinculado a un control determinado en un lugar único. También podría anidarse una estructura CASE FIELD() dentro de una estructura CASE EVENT(), invirtiendo el código, pero esto desperdigaría el código asociado a un control por múltiples lugares.

2. Dé CLIC en el botón **Run**.

De nuevo, es posible cerrar la ventana tanto con el menú del sistema, o el botón **OK**, como ocurría con el código anterior, pero ahora el código está estructurado según el estilo común.

“Hola Windows” con manejo de eventos

Hay dos tipos de eventos pasados al programa por los eventos ACCEPT: **Field-specific** y **Field-independent**.

Un evento **Field-specific** (específico del campo) ocurre cuando el usuario hace cualquier cosa que requiera que el programa tome una acción específica relativa a un control en particular. Por ejemplo, cuando el operador oprime TAB después de entrar datos en un control, se genera el evento específico del campo EVENT:Accepted.

Un evento **Field-independent** (independiente del campo) no se vincula a ningún control en particular, pero puede requerir alguna acción del programa (por ejemplo, cerrar una ventana, o cambiar los hilos de ejecución).

Anidar dos estructuras CASE, como hemos visto, es el modo más común de manejar eventos específicos al campo. El método más habitual de tratar con los eventos independientes del campo es una estructura CASE EVENT() sin anidar, en general ubicada inmediatamente antes de la estructura CASE FIELD().

Cambio del código fuente

1. Modifique el código fuente del siguiente modo:

```

PROGRAM
MAP
END
MyWin WINDOW('Hola Windows'), AT(,,100,100),SYSTEM
        STRING('Hola Windows'),AT(26,23),USE(?String1)
        BUTTON('OK'),AT(34,60),USE(?Ok),DEFAULT
        END
CODE
OPEN(MyWin)
ACCEPT
CASE EVENT()
OF EVENT:OpenWindow
    MESSAGE('Opened Window')
OF EVENT:GainFocus
    MESSAGE('Gained Focus')
END
CASE FIELD()
OF ?Ok
    CASE EVENT()
    OF EVENT:Accepted
        BREAK
    END
END
END
END

```

!Añadido
!Añadido
!Añadido
!Añadido
!Añadido
!Añadido

La nueva estructura **CASE EVENT()** maneja dos eventos independientes del campo: `EVENT:OpenWindow` y `EVENT:GainFocus`. El procedimiento **MESSAGE** usado en este código es simplemente para mostrar visualmente, durante la ejecución que se disparó el evento. En lugar del procedimiento **MESSAGE**, podría añadirse aquí cualquier otro código que el programa necesite ejecutar cuando esos eventos son disparados por el usuario.

Esto demuestra el flujo lógico y la estructura de código para los procedimientos de ventana: un lazo **ACCEPT** que contiene una estructura **CASE EVENT()** que maneja todos los eventos independientes de campos, seguidos por una estructura **CASE FIELD()** con estructuras anidadas **CASE EVENT()** para manejar todos los eventos específicos de los campos.

2. Dé CLIC sobre el botón **Run**.

Advierta que se generan los eventos `EVENT:GainFocus` y `EVENT:OpenWindow` cuando la ventana se abre por primera vez (en ese orden). `EVENT:GainFocus` (entrar en foco) se regenerará cuando el usuario haga `ALT+TAB` hacia otra aplicación, y luego vuelva nuevamente a *Hola Windows!* mediante la misma combinación de teclas.

Adición de un PROCEDIMIENTO

En *Hola Windows* tenemos un ejemplo de un programa muy simple. La mayoría de los programas empresariales actuales no tienen esa simplicidad; requieren el uso de técnicas de programación estructurada. Esto significa que el código se divide en secciones funcionales que realizan cada una de ellas una única tarea lógica. En el lenguaje Clarion, esas secciones funcionales se llaman Procedimientos (PROCEDURES).

En primer lugar, añadiremos un Procedimiento al programa *Hola Windows*.

Cambio del código fuente

1. Modifique el código fuente de esta manera:

	PROGRAM	
	MAP	
Hola	PROCEDURE	!Añadido
	END	
	CODE	!Añadido
	Hola	!Añadido
Hola	PROCEDURE	!Añadido
MyWin	WINDOW('Hola Windows'), AT(,,100,100),SYSTEM	
	STRING('Hola Windows'),AT(26,23),USE(?String1)	
	BUTTON('OK'),AT(34,60),USE(?Ok),DEFAULT	
	END	
	CODE	
	OPEN(MyWin)	!Añadido
	ACCEPT	
	CASE EVENT()	

```
OF EVENT:OpenWindow
  MESSAGE('Opened Window')
OF EVENT:GainFocus
  MESSAGE('Gained Focus')
END
CASE FIELD()
OF ?Ok
  CASE EVENT()
  OF EVENT:Accepted
    BREAK
  END
END
END
```

Los únicos cambios están aquí al principio del programa. Dentro de la estructura MAP vemos ahora la sentencia **Hola PROCEDURE** que hace de prototipo al procedimiento *Hola*. Un prototipo es la declaración del procedimiento para el compilador, para indicarle qué esperar cuando el código llame al procedimiento. Este prototipo indica que el procedimiento no lleva parámetros y no devuelve un valor. Todos los PROCEDURES de su programa deben tener su prototipo en una estructura MAP. Consulte *PROCEDURE Prototypes* en la *Language Reference* para conocer los demás prototipos.

La palabra clave **CODE** inmediatamente a continuación de la estructura MAP termina la sección de datos globales (Global data) y marca el principio de la sección Global de código ejecutable, que solamente contiene la sentencia **Hola**, que es una llamada para ejecutar el procedimiento *Hola*. Un PROCEDURE que no devuelve un valor siempre se llama con una única sentencia con su nombre en el código ejecutable.

La segunda sentencia **Hola PROCEDURE** termina la sección Global de código ejecutable y marca el comienzo de la definición mediante código del procedimiento *Hola*.

Un PROCEDURE contiene una sección de declaración de datos del mismo modo que lo hace un programa (PROGRAM) y también requiere de la palabra clave **CODE** para definir el límite entre las declaraciones de datos y el código ejecutable. Es por esto que el resto del código no cambió en el ejemplo previo. Esta es la declaración de datos locales (Local data declaration) para el PROCEDURE.

La mayor diferencia entre las declaraciones de datos Global y Local está en el alcance de los datos declarados. Cualquier dato declarado en la sección Local es visible solamente al procedimiento que lo declara, mientras que cualquier dato declarado en la sección Global es visible en todas partes del programa. Consulte *Data Declarations and Memory Allocation* en la *Language Reference* para un tratamiento más completo de las diferencias.

2. Dé CLIC sobre el botón **Run**.

El programa se ejecuta exactamente como antes. La única diferencia es que el *Hola PROCEDURE* puede ser llamado ahora desde cualquier parte dentro del programa, incluso desde otro PROCEDURE. Esto significa que aunque el programa ejecute el procedimiento muchas veces, el código necesario se escribe una sola vez.

Adición de un Procedimiento

Un PROCEDURE Clarion que no devuelve (RETURN) un valor puede ser llamado solamente en una sentencia ejecutable separada; no puede usárselo en una expresión, lista de parámetros, o una asignación. Un Procedimiento que sí devuelve un valor debe siempre contener una sentencia RETURN y puede utilizarse en expresiones, listas de parámetros, y sentencias de asignación. Se puede llamar a un PROCEDURE que sí devuelve (RETURN) un valor como una sentencia separada si no desea el valor retornado, pero esto genera una advertencia del compilador (a menos que el prototipo del PROCEDURE tenga el atributo PROC).

Estructuralmente, ambos tipos de PROCEDURE son equivalentes; ambos tienen secciones locales de datos, seguidas por una sección de código ejecutable que comienza con la palabra clave CODE.

Cambio del Código Fuente

1. Modifique la estructura MAP del siguiente modo:

```
Hola      MAP
EventString  PROCEDURE
            PROCEDURE(LONG PassedEvent),STRING      ! Agregado
            END
```

Esto añade el prototipo para **EVENTString PROCEDURE**. EventString recibe un parámetro **LONG** llamado **PassedEvent** que no puede ser omitido, y devuelve una **STRING**. Los tipos de datos de todos los parámetros pasados a un PROCEDURE se especifican dentro de los paréntesis que siguen a PROCEDURE, cada uno separado por una coma (si hay parámetros múltiples). El tipo de dato del valor devuelto de un PROCEDURE se especifica siguiendo el paréntesis de cierre de la lista de parámetros. Ambos tipos de PROCEDUREs pueden recibir parámetros, consulte la sección *Prototype Parameter List* en la *Language Reference* para un tratamiento más completo del paso de parámetros.

2. Añada la definición del procedimiento EventString al final del archivo:

```
EventString  PROCEDURE(LONG PassedEvent)
ReturnString String(255)
            CODE
            CASE PassedEvent
            OF EVENT:OpenWindow
                ReturnString = 'Opened window'
            OF EVENT:GainFocus
                ReturnString = 'Gained Focus'
            ELSE
                ReturnString = 'Unknown Event: ' & PassedEvent
            END
            RETURN(ReturnString)
```

La etiqueta **EventString** (recuerde que por ser una etiqueta, *debe estar sobre la columna 1*) de la sentencia **PROCEDURE** da nombre a este procedimiento, mientras que la lista de parámetros unidos a la palabra clave **PROCEDURE** da nombre al parámetro **LONG**, llamado **PassedEvent**.

Debe existir siempre un número igual de nombres de parámetro listados en la sentencia **PROCEDURE** como existen tipos de datos listados en el prototipo para ese **PROCEDURE**.

ReturnString es una variable local declarada como un campo alfanumérico (**STRING**) de 255 caracteres, en la pila. La sentencia **CODE** termina la sección de datos locales (Local data) del procedimiento. La estructura **CASE PassedEvent** debe verse conocida, dado que es la misma que en una estructura **CASE EVENT()**, pero su condición **CASE** es el **PassedEvent**, en lugar del procedimiento **EVENT()**. Esta estructura **CASE** simplemente asigna el valor correspondiente a la variable **ReturnString** para cada evento que se pasa al procedimiento.

El código interesante es aquí la sentencia **RETURN(ReturnString)**. Un **PROCEDURE** sin un valor de devolución no requiere de un **RETURN** específico, dado que siempre ejecuta un **RETURN** implícito cuando no hay más código que ejecutar. Sin embargo, un **PROCEDURE** indicado como prototipo para retornar un valor siempre contiene una sentencia **RETURN** explícita que especifica el valor a retornar. En este caso, la sentencia **RETURN** devuelve cualquier valor que le fue asignado a la **ReturnString** en la estructura **CASE**.

3. Modifique el procedimiento **CASE EVENT()** para que diga:

```

CASE EVENT()
OF EVENT:OpenWindow
    MESSAGE(EventString(EVENT:OpenWindow))           ! Cambiado
OF EVENT:GainFocus
    MESSAGE(EventString(EVENT:GainFocus))             ! Cambiado
END

```

Esto cambia los procedimientos **MESSAGE** para que muestren el valor devuelto del procedimiento **EventString**. El número de evento es pasado a **EventString** como el equivalente del evento (*Event EQUATE*) para hacerlo más comprensible a la lectura.

4. Dé **CLIC** sobre el botón **Run**.

El programa aún se ejecuta exactamente como antes.

Hacia el mundo real: adición de un menú

Hola Windows es un lindo programita de demostración, pero en realidad no le enseña mucho sobre cómo realizar verdadera programación comercial. Por lo tanto, lo expandiremos para que incluya algo de funcionalidad útil, comenzando por un menú.

Cambio del Código Fuente

1. Modifique el principio del archivo de este modo:

```

                PROGRAM
                MAP
Main           PROCEDURE                ! Agregado
Hola          PROCEDURE
EventString   PROCEDURE(LONG PassedEvent), STRING
                END
                CODE
                Main                       ! Cambiado

```

Esto añade el prototipo del **Main PROCEDURE** a la estructura MAP y reemplaza la llamada a Hola con una llamada al procedimiento **Main**.

2. Añada la definición del PROCEDURE al final del archivo:

```

Main           PROCEDURE
AppFrame       APPLICATION('Hola Windows'), AT(, ,280,200), SYSTEM, RESIZE, MAX
                MENUBAR
                MENU(' &File'), USE(?File)
                ITEM('&Browse Telefono'), USE(?FileBrowseTelefono)
                ITEM, SEPARATOR
                ITEM('E&xit'), USE(?FileExit),STD(STD:close)
                END
                ITEM('&About!'), USE(?About)
                END
                END
                CODE
                OPEN(AppFrame)
                ACCEPT
                CASE ACCEPTED()
                OF ?About
                Hola
                END
                END
                END

```

El procedimiento **Main** no acepta parámetros. Contiene la estructura **AppFrame APPLICATION**. Una estructura APPLICATION es la clave para la creación de programas con Interfaz de Documentos Múltiples (MDI). Una aplicación MDI puede contener múltiples hilos de ejecución. Este es el marco “padre” MDI que se necesita para crear una aplicación MDI.

La estructura **MENUBAR** define los ítems del menú disponibles al usuario. La estructura **MENU('&File')** crea el menú File estándar que aparece en casi todos los programas Windows. El signo (&) que precede a la “F” especifica que éste es el “atajo del teclado”, y el sistema operativo la subrayará durante la ejecución.

La opción **ITEM('&BrowseTelefono')** crea un ítem de menú que usaremos para llamar a un procedimiento (como pronto veremos). La sentencia **ITEM,SEPARATOR** crea una línea divisoria en el menú a continuación de la selección *Browse Telefono*.

El **ITEM('E&xit')** crea un ítem del menú para salir del procedimiento (y del programa, dado que este procedimiento es el único llamado desde el código ejecutable Global). El atributo **STD(STD:Close)**, especifica la acción estándar de cerrar una ventana para salir del lazo ACCEPT. Esto esporque no se vé ningún código ejecutable asociado a este ítem de menú en el lazo ACCEPT: la biblioteca Clarion de tiempo de ejecución se encarga automáticamente.

Las dos nuevas sentencias **INCLUDE** añaden los EQUATEs estándar para los códigos de tecla y los números de error. El uso de las equivalencias (EQUATEs) en lugar de los números hace más comprensible el código y, por lo tanto, más fácil de mantener.

La estructura MAP ha adquirido otros dos prototipos de procedimientos: el **BrowseTelefono PROCEDURE y UpdateTelefono PROCEDURE(LONG Action),LONG**. El procedimiento BrowseTelefono mostrará una lista de los registros en el archivo y UpdateTelefono actualizará los registros individuales.

En interés de mantener la simplicidad del código (trataremos esto enseguida), el procedimiento BrowseTelefono se limitará a mostrar todos los registros del archivo en un control LIST. Esto no es exactamente lo mismo que un procedimiento Browse del código generado por los templates (que se carga de a páginas para poder manejar archivos muy grandes), pero servirá al mismo fin aquí.

La declaración **Telefono FILE** crea un archivo de datos simple utilizando el controlador TopSpeed. Hay dos campos de datos: **Nombre y Numero**, declarados ambos como alfanuméricos: **STRING(20)**. La declaración del campo de número de este modo le permite contener números telefónicos de cualquier país del mundo (pronto más sobre esto)

Las cinco sentencias **EQUATE** definen valores constantes que hacen más legible el código. **InsertRecord, ChangeRecord y DeleteRecord** definen acciones que deben pasarse como parámetros al procedimiento UpdateTelefono. Las equivalencias **ActionComplete y ActionAborted** definen los dos posibles valores devueltos por el procedimiento UpdateTelefono.

2. Modifique el código CASE ACCEPTED() del procedimiento *Main*, de este modo:

```

CASE ACCEPTED()
  OF ?FileBrowseTelefono                ! Agregado
    START(BrowseTelefono,25000)         ! Agregado
  OF ?About
    Hola
END

```

La sentencia **START(BrowseTelefono,25000)** se ejecuta cuando el usuario elige la selección del menú Browse Telefono. El procedimiento START crea un nuevo hilo de ejecución para el procedimiento BrowseTelefono y el segundo parámetro (25000) especifica el tamaño (en bytes) de la pila del nuevo hilo de ejecución. Puede utilizarse el procedimiento START cuando está llamando a un procedimiento que contiene una ventana MDI hija desde el marco de la APPLICATION, debido a que cada ventana hija MDI debe estar sobre un hilo de ejecución separado. Si usted no abre (START) la hija MDI, durante la ejecución aparecerá el error "Unable to open MDI window on APPLICATION's thread" cuando intente llamar el procedimiento BrowseTelefono y el programa se terminará inmediatamente.

Una vez que ha comenzado un nuevo hilo de ejecución, un hijo MDI puede simplemente llamar a otro hijo MDI en su mismo hilo sin necesidad de empezar uno nuevo. Esto significa que, aunque

BrowseTelefono y UpdateTelefono ambos contienen ventanas MDI, solamente BrowseTelefono debe empezar (START) un nuevo hilo, dado que es llamado desde la ventana de la aplicación.

Añadir el procedimiento BrowseTelefono

1. Añada la sección de datos de la definición BrowseTelefono PROCEDURE al final del archivo:

```

BrowseTelefono      PROCEDURE
TelefonoQue         QUEUE
Nombre              STRING(20)
Numero              STRING(20)
Position            STRING(512)
                    END

Window WINDOW('Browse Telefono'), AT(, , 185,156), SYSTEM, GRAY,RESIZE,MDI
LIST, AT(6,8,173,100),ALRT(MouseLeft2),USE(?List), |
    FORMAT('84L|M~Nombre~80L~Numero~'),FROM(TelefonoQue),VSCR
OLL
    BUTTON('&Insert'),AT(20,117),KEY(InsertKey),USE(?Insert)
    BUTTON('&Change'),AT(76,117,35,14),KEY(EnterKey),USE(?Change)
    BUTTON('&Delete'),AT(131,117,35,14),KEY(DeleteKey),USE(?Delete)
    BUTTON('Close'),AT(76,137,35,14),KEY(EscKey),USE(?Close)
END

```

La estructura de “cola” TelefonoQue QUEUE define la estructura de datos que contendrá todos los registros del archivo Telefono y los mostrará en el control LIST. Una “cola” (QUEUE) de Clarion es similar a un archivo de datos debido a que tiene un buffer de datos y permite un número indeterminado de registros, pero solamente existe en memoria durante la ejecución. Una cola también puede asimilarse a una matriz asignada dinámicamente en tiempo de ejecución. Consulte el capítulo *Memory Queues* en la *Language Reference* para más información sobre estas estructuras.

La cola TelefonoQue contiene tres campos. Los campos **Nombre y Numero** duplican los campos en la estructura del archivo Telefono, y se mostrarán en las dos columnas definidas por el atributo FORMAT del control LIST. El campo **Position** contendrá el valor de devolución del procedimiento POSITION de cada registro del archivo Telefono. Guardar la posición de cada registro nos permitirá reubicar inmediatamente el registro desde el archivo de datos, antes de llamar a UpdateTelefono para cambiar o borrar un registro.

La estructura **window WINDOW** contiene un control LIST y cuatro botones (BUTTON). El control **LIST** es la clave de este procedimiento. La alerta **ALRT(MouseLeft2)** sobre LIST alerta sobre un posible DOBLE CLIC para que la sentencia ACCEPT pase un EVENT:AlertKey al código Clarion. Esto nos permitirá escribir código para traer el procedimiento UpdateTelefono para cambiar el registro sobre el que el usuario da DOBLE CLIC.

La barra vertical (|) al final de la sentencia LIST es el carácter de continuación de Clarion, lo que significa que el control LIST continúa en la línea siguiente con el atributo **FORMAT('84L|M~Nombre~80L~Numero~')**, aunque si lo desea puede poner todo en una sola línea, sin la barra de separación. El parámetro del atributo FORMAT define la apariencia del control LIST en tiempo de ejecución.

Es mejor dejar la herramienta de Diseño de Listados en el Diseñador de Ventanas que escriba las cadenas de formato, dado que rápidamente se vuelven muy complejas. Este formato define dos columnas. La primera columna tiene 84 unidades de diálogo de ancho (84), está justificado a la izquierda (L), tiene un borde derecho (|) ajustable (M) y la palabra "Nombre" es su encabezamiento (~Nombre~). La segunda columna tiene 80 unidades de diálogo de ancho (80), con justificación a la izquierda (L), y su encabezamiento es "Numero" (~Numero~).

FROM(TelefonoQue) en la LIST especifica que la cola (QUEUE) que mostrará el control LIST, y **VSCROLL** añade la barra de deslizamiento vertical. La LIST desplegará los campos Nombre y Numero de todos los ítems en la TelefonoQue ignorando el campo Position, dado que el atributo FORMAT sólo especificó dos columnas.

La LIST automáticamente permite a los usuarios desplazarse por la lista sin necesidad de codificación adicional. La LIST hace esto porque no está presente el atributo IMM (immediate). Si éste existiera, deberíamos escribir código para manejar el desplazamiento (*scrolling*) de los registros, como ocurre en la plantilla Browse.

La sentencia **BUTTON('&Insert)** define un botón de órdenes que el usuario oprimirá para añadir un nuevo registro. El atributo **KEY(InsertKey)** especifica que el botón se oprime automáticamente para el usuario al pulsar la tecla INSERT. Advertida que los otros tres controles BUTTON tienen similares atributos KEY. Esto significa que no es necesario escribir código especial alguno para manejar el acceso mediante el teclado, si el usuario lo prefiere al *mouse*.

2. Añada el principio de la sección de código ejecutable de la definición del BrowseTelefono PROCEDURE al final del archivo:

```

CODE
DO OpenFile
DO FillQue
OPEN(window)
ACCEPT
  CASE EVENT()
  OF EVENT:OpenWindow
    DO QueRecordsCheck
  OF EVENT:AlertKey
    IF KEYCODE() = MouseLeft2
      POST(EVENT:Accepted, ?Change)
    END
  END
END
END

```

El principio de la sección **CODE** empieza con la sentencia **DO OpenFile**. **DO** ejecuta una **ROUTINE**, en este caso **OpenFile**, y cuando ha terminado este código, el control regresa a la siguiente línea de código después de la sentencia **DO** que llamó a **ROUTINE**.

El código que define una **ROUTINE** debe estar al final del procedimiento, luego de toda la lógica principal, debido a que la sentencia **ROUTINE** da fin a la sección de código ejecutable de un **PROCEDURE**.

Hay dos razones para usar una **ROUTINE** dentro de un **PROCEDURE**: para escribir un grupo de sentencias de código que deben ejecutarse en múltiples puntos lógicos dentro del procedimiento, o para hacer el código más comprensible al sustituir con una simple sentencia **DO** un grupo de sentencias que realizan una única labor lógica.

En este caso, la sentencia **DO OpenFile** sirve al segundo propósito, moviendo el código que abre o crea el archivo fuera de la lógica principal del procedimiento. A continuación, la sentencia **DO FillQue** ejecuta el código que llena la estructura **TelefonoQue QUEUE** con todos los registros tomados del archivo. Las dos sentencias **DO** hacen muy fácil seguir el flujo lógico del procedimiento.

Las estructuras **CASE EVENT()** de la cláusula lógica **OF EVENT:OpenWindow** ejecutan **DO QueRecordsCheck** para llamar a una **ROUTINE** que verifica para ver si hay registros en **TelefonoQue**.

A continuación, la cláusula **OF EVENT:AlertKey** contiene la estructura **IF KEYCODE() = MouseLeft2** para verificar si hay **DOBLE CLIC** en la **LIST**. Dado que el atributo **ALRT(MouseLeft2)** solamente aparece sobre el control **LIST**, sabemos que la sentencia **POST(EVENT:Accepted,?Change)** se ejecutará solamente cuando el usuario de **DOBLE CLIC** sobre la **LIST**.

La sentencia **POST** le indica a **ACCEPT** que “remita” el evento en su primer parámetro (**EVENT:Accepted**) al control en su segundo parámetro (**?Change**). El efecto es que se ejecute el código del botón **Change**, como si el usuario hubiese oprimido el botón con el *mouse* o con el teclado.

Esto ilustra una excelente práctica de programación: escribir código específico una vez y llamarlo desde muchos lugares. Este concepto de la programación estructurada nos brindó los Procedimientos y las Rutinas. Aunque el código **EVENT:Accepted** para el botón **Change** no está separado en un **PROCEDURE** o **ROUTINE**, el uso de la sentencia **POST** (“enviar”) significa que esa es la única sección del código que debe mantener: si quiere cambiar la lógica, basta hacerlo en un solo lugar.

3. Añada el resto de la sección de código ejecutable de la definición del procedimiento **BrowseTelefono** al final del archivo:

```

CASE FIELD()
OF ?Close
  CASE EVENT()
  OF EVENT:Accepted
    POST(EVENT:CloseWindow)
  END
OF ?Insert
  CASE EVENT()
  OF EVENT:Accepted
    IF UpdateTelefono(InsertRecord) = ActionComplete
      DO AssignToQue
      ADD(TelefonoQue)
      IF ERRORCODE() THEN STOP(ERROR()).
      SORT(TelefonoQue,TelefonoQue.Nombre)
      ENABLE(?Change, ?Delete)
    END
    SELECT(?List)
  END
END

OF ?Change
  CASE EVENT()
  OF EVENT:Accepted
    DO GetRecord
    IF UpdateTelefono(ChangeRecord) = ActionComplete
      DO AssignToQue
      PUT(TelefonoQue)
      IF ERRORCODE() THEN STOP(ERROR()).
      SORT(TelefonoQue,TelefonoQue.Nombre)
    END
    SELECT(?List)
  END
END

OF ?Delete
  CASE EVENT()
  OF EVENT:Accepted
    DO GetRecord
    IF UpdateTelefono(DeleteRecord) = ActionComplete
      DELETE(TelefonoQue)
      IF ERRORCODE() THEN STOP(ERROR()).
      DO QueRecordsCheck
      SORT(TelefonoQue,TelefonoQue.Nombre)
    END
    SELECT(?List)
  END
END
END
FREE(TelefonoQue)
CLOSE(Telefono)

```

Siguiendo la estructura CASE EVENT() se ubica la estructura **CASE FIELD()**. Advierta que cada cláusula OF contiene su propia estructura **CASE EVENT()**, y cada una solamente contiene una cláusula **OF EVENT:Accepted**. Debido a esto, podríamos haber reemplazado la estructura

CASE FIELD() con un case ACCEPTED() y eliminado los CASEs anidados. Esto hubiera mejorado ligeramente el desempeño; pero demasiado poco como para notarlo en la pantalla. No lo hicimos para mantener la coherencia; habrá muchas ocasiones en que se deban capturar más eventos específicos al campo que sólo EVENT:Accepted, y cuando esto ocurre, esta estructura anidada Case es la lógica a usar, de manera que resulta bueno habituarse desde ahora. También demuestra el tipo de estructura de código que generan los templates de Clarion en el Generador de Aplicaciones.

La cláusula **OF ?CLOSE** ejecuta el **POST(EVENT:CloseWindow)** cuando el usuario oprime el botón Close. Dado que no hay un segundo parámetro en la sentencia POST que nombra un control, el evento envía a la WINDOW (y debe ser un evento independiente del campo). EVENT:CloseWindow provoca la terminación del lazo ACCEPT y el control de la ejecución cae a la primera sentencia que sigue al END de terminación del lazo ACCEPT. En este caso, el control va a la sentencia **FREE(TelefonoQue)**, lo que libera toda la memoria utilizada por los registros de la cola, cerrándola, y cerrando el archivo. Dado que no hay otras sentencias que ejecutar, el procedimiento realiza un RETURN implícito y vuelve al procedimiento *Main* (desde donde fue llamado).

La cláusula **OF ?INSERT** contiene la estructura **IF UpdateTelefono(InsertRecord)=ActionComplete**. Esto llama al procedimiento UpdateTelefono, pasándole el valor constante InsertRecord que definimos en la sección de datos Global, y luego verifica la devolución del valor ActionComplete.

La sentencia **DO AssignToQue** (asignar a la cola) se ejecuta solamente cuando el usuario añade un registro nuevo. AssignToQue es una rutina que asigna datos del buffer de registro del archivo Telefono al buffer de la cola respectiva. Entonces, la sentencia **ADD(TelefonoQue)** añade una nueva entrada a la cola. La sentencia **IF ERROCODE() THEN STOP(ERROR())** es una comprobación estándar de errores que debería ejecutarse después de cada acción sobre un archivo o sobre una cola que tenga la posibilidad de devolver un error (para formarse otro buen hábito).

La sentencia **SORT(TelefonoQue,TelefonoQue.Nombre)** alfabetiza la cola por el campo Nombre. Dado que no hay un atributo PRE (prefijo) en la estructura TelefonoQue, debe referenciar sus campos utilizando la sintaxis de calificación de campos Clarion, anteponiendo el nombre de la estructura que contiene el campo (TelefonoQue) al nombre del campo (Nombre), conectándolos entre sí con un punto, de este modo:TelefonoQue.Nombre. Consulte *Field qualification* en la *Language Reference* para más información.

La sentencia **ENABLE(?Change, ?Delete)** asegura que los botones de cambio y borrado estén activos (si esta fue la primera entrada en la QUEUE, los botones están grisados por la QueRecodsCheck ROUTINE). La sentencia **SELECT(?List)** vuelve al usuario al control LIST.

El código en la cláusula **OF ?Change** es casi idéntico al código en la cláusula OF ?Insert. Hay una sentencia adicional **DO GetRecord** que llama una rutina para poner el registro iluminado de TelefonoQue dentro del buffer del registro del archivo Telefono. La única otra diferencia es la sentencia **PUT(TelefonoQue)** que incorpora los cambios del usuario en TelefonoQue.

El código en la cláusula **OF ?Delete** es casi idéntico al código en la cláusula **OF ?Change**. La diferencia es la sentencia **DELETE(TelefonoQue)** que quita el registro de TelefonoQue y la llamada al **DO QueRecordsCheck** para comprobar si el usuario ha borrado el último registro.

4. Añada las rutinas llamadas en la definición del procedimiento BrowseTelefono, al final del archivo:

```

AssignToQue  ROUTINE
              TelefonoQue.Nombre = Telefono.Rec.Nombre
              TelefonoQue.Numero = Telefono.Rec.Numero
              TelefonoQue.Position = POSITION(Telefono)

QueRecordsCheck  ROUTINE
                  IF NOT RECORDS(TelefonoQue)
                      DISABLE(?Change, ?Delete)
                      SELECT(?Insert)
                  ELSE
                      SELECT(?List,1)
                  END

GetRecord        ROUTINE
                  GET(TelefonoQue,CHOICE(?List))
                  IF ERRORCODE() THEN STOP(ERROR()).
                  REGET(Telefono,TelefonoQue,Position)
                  IF ERRORCODE() THEN STOP(ERROR()).

OpenFile         ROUTINE
                  OPEN(Telefono,42h)
                  CASE ERRORCODE()
                  OF NoError  OROF IsOpenErr
                      EXIT
                  OF NoFileErr
                      CREATE(Telefono)
                      IF ERRORCODE() THEN STOP(ERROR()).
                      OPEN(Telefono,42h)
                      IF ERRORCODE() THEN STOP(ERROR()).
                  ELSE
                      STOP(ERROR())
                      RETURN
                  END

FillQue          ROUTINE
                  SET(Telefono,KeyNombre)
                  LOOP
                      NEXT(Telefono)
                      IF ERRORCODE() THEN BREAK.
                      DO AssignToQue
                      ADD(TelefonoQue)
                      IF ERRORCODE() THEN STOP(ERROR()).
                  END

```

Como puede observar, hay cinco rutinas en este procedimiento. Adviértase que, aunque esas rutinas se ven similares a un procedimiento, *no* contienen sentencias CODE. Esto se debe a que una rutina comparte los datos locales del procedimiento y *no suele tener* una sección propia de declaración de datos (aunque *sí* puede tenerla: vea ROUTINE en la *Language Reference* para un tratamiento completo de este tema).

La rutina **AssignToQue** efectúa tres asignaciones. La sentencia **TelefonoQue.Nombre = Telefono.Rec.Nombre** copia los datos en el campo Nombre de buffer de registro y los coloca en el campo Nombre de la cola. Dado que no hay prefijo (atributo PRE) en la estructura Telefono FILE, debe también referenciar sus campos según la sintaxis de calificación de campos de Clarion, uniendo el nombre del archivo (Telefono), el nombre del registro (Rec), y el nombre del campo (Nombre), conectándolos con un punto (Telefono.Rec.Nombre). Consulte *Field Qualification* en la *Language Reference* para más información.

La sentencia **TelefonoQue.Numero = Telefono.Rec.Numero** asigna los datos en el campo de número del archivo de teléfono al campo de la cola TelefonoQue "Numero". La sentencia **TelefonoQue.Position = POSITION(Telefono)** asigna el valor devuelto por el procedimiento POSITION al campo TelefonoQue.Position. Este valor nos permite recuperar del disco el registro específico que esté actualmente en el buffer de teléfonos del archivo Telefono. El procedimiento POSITION lo hace comunicándose con el controlador de archivos Clarion, y es por lo tanto el método recomendado de toma de registros en todos los sistemas de archivos utilizados.

La **QueRecordsCheck ROUTINE** verifica para ver si hay registros en TelefonoQue. La estructura **IF NOT RECORDS(TelefonoQue)** utiliza el operador lógico NOT contra el valor devuelto del procedimiento RECORDS. Si RECORDS(TelefonoQue) devuelve cero, el NOT hace verdadera la condición y se ejecuta el código que sigue al IF (cero es siempre falso, y el NOT lo hace verdadero). Si RECORDS(TelefonoQue) devuelve algo distinto de cero, el código que sigue al ELSE se ejecutará (cualquier número que no sea cero es siempre verdadero, y el NOT lo hace falso). Por lo tanto, si no hay registros en la cola TelefonoQue, **DISABLE(?Change, ?Delete)**, se ejecuta para inhabilitar los botones respectivos, y la sentencia **SELECT(?Insert)** coloca al usuario sobre el botón Insert (la siguiente acción lógica). Si hay registros en TelefonoQue la sentencia **SELECT(?List)** coloca al usuario sobre la LIST.

La rutina **GetRecord ROUTINE** sincroniza el buffer del registro del archivo Telefono y el buffer de datos de la cola TelefonoQue con el registro actualmente iluminado en la LIST. La sentencia **GET(TelefonoQue,CHOICE(?List)** utiliza el procedimiento CHOICE para "apuntar" al "registro" iluminado en la lista y recuperar (GET) el "registro" vinculado al buffer de TelefonoQue (comprobando siempre si hay errores inesperados). Luego, la sentencia **REGET(Telefono,TelefonoQue.Position)** utiliza la información guardada sobre el registro grabado para recuperarlo y enviarlo al buffer del registro de la cola.

La rutina **OpenFile ROUTINE** abre o crea el archivo Telefono. La sentencia **OPEN(Telefono,42h)** intenta abrir el archivo Telefono para ser compartido. El segundo parámetro es un número hexadecimal (condición indicada por la "h"). Clarion admite los sistemas numéricos decimal, hexadecimal, binario y octal. Este número representa el acceso plenamente compartido al archivo (Read/Write, Deny None) (consulte la sentencia OPEN en la *Language Reference* para más sobre los modos de acceso a los archivos). Pedimos acceso compartido por que

este es un programa MDI y el usuario podría llamar a copias múltiples del mismo procedimiento en el mismo programa. Sin embargo, este programa no hace todas las comprobaciones de acceso concurrente que se requieren para una aplicación multiusuario real. Consulte la sección correspondiente de la *Programmer's Guide* para más información sobre el tema.

La estructura **CASE ERRORCODE()** verifica cualquier error en el OPEN. La cláusula **OF NoError OROF IsOpenErr** (ahora puede ver por qué hemos incluido el archivo ERRORS.CLW) ejecuta la sentencia **EXIT** para salir inmediatamente de esta rutina. Es muy importante no confundir **EXIT** con **RETURN**, dado que este último finaliza inmediatamente el procedimiento, mientras que **EXIT** sólo termina la *rutina*. Es válido utilizar **RETURN** dentro de una rutina, pero asegúrese de que quiere terminar todo el procedimiento, y no solamente la rutina misma.

La cláusula **OF NoFileErr** detecta que no hay un archivo de datos para abrir. La sentencia **CREATE(Telefono)** creará un nuevo archivo de datos vacío. Debe asegurarse de que, si intenta usar la sentencia **CREATE** que la declaración **FILE** también contenga el atributo **CREATE**, o la sentencia **CREATE** no podrá trabajar. Esta sentencia no abre el archivo, lo que explica la sentencia **OPEN(Telefono,42h)**. El código en la cláusula **ELSE** se ejecuta si ocurre cualquier error distinto de esos. La sentencia **STOP(ERROR())** muestra el mensaje de error del procedimiento al usuario en una ventana modal del sistema, lo que le da la oportunidad de cancelar el programa (regresando a Windows) o ignorar el error. La sentencia **RETURN** termina el procedimiento si el usuario elige esta última opción.

La rutina **FillQue ROUTINE** llena la cola TelefonoQue con todos los registros en el archivo Telefono. La sentencia **SET(Telefono.KeyNombre)** prepara el orden de procesamiento y el punto de comienzo del archivo Telefono. El parámetro Telefono.KeyNombre hace el procesamiento alfabético según el campo Nombre. La ausencia de un segundo parámetro en la sentencia **SET** hace del punto de comienzo el principio (o el fin) del archivo. La estructura **LOOP** no tiene condición, lo que significa que debe ponerse una sentencia **BREAK** en alguna parte del **LOOP**, o tendría un lazo infinito. La sentencia **NEXT(Telefono)** recupera el registro siguiente del archivo de datos Telefono, y luego la sentencia **IF ERRORCODE() THEN BREAK** asegura que salgamos (**BREAK**) del **LOOP**, cuando hayan sido leídos todos los registros. La sentencia **DO AssignToQue** llama a la rutina del mismo nombre que ya hemos tratado, y **ADD(TelefonoQue)** añade el nuevo registro a la cola.

Adición del procedimiento UpdateTelefono

1. Añada la sección de datos de la definición del UpdateTelefono PROCEDURE al final del archivo:

```

UpdateTelefono    PROCEDURE(LONG Action)
ReturnValue       LONG,AUTO
Window           WINDOW('Update Telefono'), AT( ,185,92),SYSTEM,GRAY,RESIZE,MDI,MASK
                 PROMPT('N&ombre:'),AT(14,14),USE(?Prompt1)
                 ENTRY(@s20),AT(68,13),USE(Telefono.Rec.Nombre),REQ
                 PROMPT('N&umero:'),AT(14,43),USE(?Prompt2)
                 ENTRY(@s20),AT(68,42),USE(Telefono.Rec.Numero)
                 BUTTON('OK'),AT(45,74),USE(?Ok),REQ,DEFAULT
                 BUTTON('Cancela'),AT(109,74,32,14),USE(?Cancel)
END

```

La sentencia **UpdateTelefono PROCEDURE(LONG Action)**, define un PROCEDURE que recibe un único parámetro LONG que se llamará "Action" dentro del procedimiento (sin importar qué variable o constante se pase).

ReturnValue LONG,AUTO declara una variable LONG que permanece sin inicializar por el compilador (debido al atributo AUTO). Por omisión, las variables de memoria son inicializadas a espacios en blanco o a cero (según el tipo de datos). Especificar AUTO ahorra algo de memoria, pero a condición que el programador tenga la certeza de que asignará un valor a la variable sin inicializar *antes* de verificar por primera vez su contenido, o se produciría un *bug* intermitente muy difícil de localizar.

La estructura **WINDOW** posee el atributo de máscara (**MASK**), lo que significa que los patrones de ingreso de datos son comprobados a medida que se ingresan datos, en lugar del comportamiento estándar de Windows de ingreso de datos "de formato libre".

Los dos controles **PROMPT** y **ENTRY** se combinan para el ingreso de datos. Los dos controles **BUTTON** permitirán al usuario completar o cancelar la acción actual.

Los controles **PROMPT** definen las etiquetas de campos en pantalla, y la tecla "aceleradora" (o atajo del teclado) para navegar hasta el control **ENTRY** que sigue al **PROMPT**. Los atajos del teclado se forman utilizando **ALT** más la letra que sigue al signo **&**. Por ejemplo ('N&ombre:') indica que **ALT+A** dará foco al control del mismo nombre.

Los atributos **USE** de los controles **ENTRY** nombran los campos de datos que automáticamente reciben el ingreso del usuario. La biblioteca de tiempo de ejecución asegura que el valor actual en la variable nombrada en el atributo **USE** muestra cuando el control tiene foco. Cuando el usuario ingresa datos en el control **ENTRY** y luego pasa a otro control, la biblioteca de tiempo de ejecución asegura que la variable nombrada en el atributo **USE** reciba el valor actual mostrado en el control.

El atributo **REQ** del primer control **ENTRY** significa que el usuario no puede dejarlo en blanco, mientras que el atributo **REQ** sobre el botón **OK** verifica que el usuario haya ingresado datos en todos los controles de ingreso de datos que lo requieran. Esta verificación sólo se hace cuando se oprime el botón con el atributo **REQ**, porque el usuario puede ni siquiera haber pasado por el control de ingreso con este atributo.

2. Añada la lógica principal del procedimiento UpdateTelefono al final del archivo:

```

CODE
OPEN(Window)
DO SetupScreen
ACCEPT
  CASE FIELD()
  OF ?Telefono:Rec:Numero
    CASE EVENT()
    OF EVENT:Selected
      DO SetupInsert
    END
  OF ?Ok
    CASE EVENT()
    OF EVENT:Accepted
      EXECUTE Action
      ADD(Telefono)
      PUT(Telefono)
      DELETE(Telefono)
    END
    IF ERRORCODE() THEN STOP(ERROR()).
    ReturnValue = ActionComplete
    POST(EVENT:CloseWindow)
  END
  OF ?Cancel
    CASE EVENT()
    OF EVENT:Accepted
      ReturnValue = ActionAborted
      POST(EVENT:CloseWindow)
    END
  END
END
RETURN(ReturnValue)

```

La sentencia **DO SetupScreen** llama a la SetupScreen ROUTINE para realizar algunas funciones de inicialización de la ventana. Advierta que sigue a la sentencia **OPEN(Window)**. Cuando se va a alterar dinámicamente la ventana en un procedimiento, debe ser abierta en primer lugar.

La cláusula **OF ?Telefono:Rec:Numero** en la estructura **CASE FIELD()** demuestra dos cosas importantes. La primera es la equivalencia de campo (Field Equate Label) misma. El atributo **USE(Telefono.Rec.Numero)** contiene puntos en el nombre del campo, y los puntos no son válidos para las etiquetas de código Clarion. Por lo tanto, para construir una equivalencia para Telefono.Rec.Numero, el compilador sustituye el punto por dos puntos (dado que los dos puntos son válidos en una etiqueta de Clarion).

El segundo punto a considerar es la cláusula **OF EVENT:Selected** en la estructura **CASE EVENT()**. EVENT:Selected genera cuando el control tiene foco para el ingreso de datos, pero antes de que el usuario empiece a hacerlo. La sentencia **DO SetupInsert** se ejecuta para ofrecer al operador una opción y configurar luego el formato de despliegue y de ingreso de datos para ese control ENTRY.

El código **OF EVENT:Accepted** on **OF ?Ok** es el código que escribe el registro al disco. La estructura **EXECUTE Action** ejecuta exactamente una de las sentencias **ADD(Telefono)**, **PUT(Telefono)**, o **DELETE(Telefono)**.

Una estructura EXECUTE es similar a las estructuras IF y CASE en que ejecuta código en forma condicional, según la evaluación de una condición. La condición EXECUTE debe evaluar a un entero en la gama de 1 a n (en que n es el número de sentencias de código dentro de la estructura EXECUTE), luego ejecuta la línea de código dentro de la estructura que corresponde al valor de la condición.

En este código, EXECUTE consulta el parámetro Action, y luego ejecuta ADD(Telefono) si el valor de Action es 1, PUT(Telefono), si es 2, o DELETE(Telefono) si es 3.

En general, cuando se evalúa cuál estructura de código Clarion debe usar para una cláusula condicional (IF/ELSIF, CASE o EXECUTE), la estructura IF/ELSIF es la más flexible, y la menos eficiente. CASE es menos flexible pero mucho más eficiente que IF/ELSIF, y EXECUTE no es flexible, pero muy eficiente. Por lo tanto cuando la condición a evaluar puede resolverse en un entero en el rango de 1 a n , utilice EXECUTE, de lo contrario, intente usar CASE. Si éste no resulta lo suficientemente flexible, recurra a IF/ELSIF.

La sentencia **IF ERRORCODE() THEN STOP(ERROR())** verificará que no haya errores, sin importar cuál sentencia ejecutó EXECUTE. La sentencia **ReturnValue = ActionComplete** configura el regreso al procedimiento que hizo la llamada, señalando que el usuario ha completado la acción del archivo, y luego el **POST(EVENT:CloseWindow)** termina el lazo ACCEPT, dejando caer el control a la sentencia **RETURN(ReturnValue)**.

3. Añada las ROUTINES llamadas en la definición del procedimiento UpdateTelefono, al final del archivo:

```
SetupScreen      ROUTINE
                  CASE Action
                  OF InsertRecord
                     CLEAR(Telefono.Rec)
                     TARGET{PROP:Text} = 'Agrega número nuevo'
                  OF ChangeRecord
                     TARGET{PROP:Text} = 'Modifica número telefónico de ' |
                                             & CLIP(Telefono.Rec.Nombre)
                     IF Telefono:Rec:Numero[1] <> '+'
                        ?Telefono:Rec:Numero[PROP:Text] = '@P###-###-####P'
                     END
                  OF DeleteRecord
                     TARGET{PROP:Text} = 'Elimina número telefónico de ' |
                                             & CLIP(Telefonos.Rec.Nombre)
                     DISABLE(FIRSTFIELD(),LASTFIELD())
                     ENABLE(?Ok, ?Cancel)
                  END
```

```

SetupInsert      ROUTINE
                  IF Action = InsertRecord
                    CASE MESSAGE('Internacional?', 'Format',ICON:Question, |
                                BUTTON:Si+BUTTON:No,BUTTON:No,1)
                    OF BUTTON:Si
                      TARGET{PROP:Text} = 'Agrega número internacional'
                      ?Telefono:Rec:Numero{PROP:Text} = '@S20'
                      Telefono:Rec:Numero[1] = '+'
                      DISPLAY
                      Select(?,2)
                    OF BUTTON:No
                      TARGET{PROP:Text} = 'Agrega número nacional'
                      ?Telefono:Rec:Numero{PROP:Text} = '@P###-###-####P'
                    END
                  END
                END

```

La **SetupScreen ROUTINE** comienza por evaluar **CASE Action**. Cuando el usuario está añadiendo un registro (**OF InsertRecord**) la sentencia **CLEAR(Telefono.Rec)** limpia el buffer de registro poniendo todos los campos en blanco o en cero. La sentencia **TARGET{PROP:Text} = 'Agrega número nuevo'** usa la sintaxis de propiedades en tiempo de ejecución de Clarion para cambiar dinámicamente la barra de título a "Agrega número nuevo". La sintaxis de propiedades permite cambiar dinámicamente cualquier propiedad (atributo) de una estructura APPLICATION, WINDOW o REPORT en código ejecutable. Vea *Appendix C – Property Assignments* en la *Language Reference* para más sobre esto.

TARGET es una variable incorporada que siempre "apunta" a la estructura de WINDOW abierta. Las llaves ({}) delimitan la propiedad misma, y PROP:Text es una equivalencia (contenida en EQUATES.CLW) que identifica el parámetro del elemento de datos (en este caso, la sentencia WINDOW).

El código **OF ChangeRecord** que indica **TARGET{PROP:Text} = 'Modifica número telefónico de ' & CLIP(Telefono.Rec.Nombre)** hace lo mismo, pero cambia la barra de título para que muestre: "Modifica número telefónico de alguien". El signo (&) es el operador de concatenación de cadenas alfanuméricas en Clarion, y el procedimiento CLIP (Telefono.Rec.Nombre) remueve los espacios a la derecha del nombre. La estructura **IF Telefono:Rec:Numero[1] <> '+'** verifica la presencia de un signo + como primer carácter del campo de Numero. Este signo más se usa aquí como una señal de que se trata de un número internacional.

Advierta que el Telefono:Rec:Numero[1] apunta al primer byte del campo como si fuera una matriz (*array*). Pero usted recordará que no había atributo DIM en la declaración (lo que hubiera declarado un *array*). Todos los tipos de datos STRING, CSTRING, y PSTRING en Clarion también son implícitamente una matriz de STRING(1),DIM(SIZE(StringField)). Esto significa que puede referenciar directamente cualquier carácter individual de la cadena, sea que haya sido declarada como una matriz, o no.

Si el número no es internacional, la sentencia **?Telefono:Rec:Numero{PROP:Text} = '@P###-###-####P'** utiliza el mismo tipo de sintaxis para cambiar la máscara de ingreso del control. Advierta que PROP:Text se refiere a cualquiera sea el parámetro del control. Por lo tanto, en una WINDOW('texto del título') se refiere al título, mientras que en ENTRY(@S20) se refiere a la máscara (@S20).

El código **OF DeleteRecord** es similar al código **ChangeRecord**. La sentencia **DISABLE(FIRSTFIELD(),LASTFIELD())**, utiliza los procedimientos FIRSTFIELD() y LASTFIELD() para inhabilitar todos los controles en la ventana, y luego **ENABLE(?Ok,?Cancel)**, habilita solamente los botones OK y Cancel.

La **ROUTINE SetupInsert** se ejecuta justo antes de que el usuario vaya al control Numero ENTRY. La sentencia **IF Action = InsertRecord** verifica el valor de Action y solamente ejecuta la estructura **CASE MESSAGE** cuando el usuario está añadiendo un nuevo registro. El procedimiento MESSAGE puede utilizarse para crear elecciones simples Si/No para los usuarios. En este caso, se le pregunta si el número es internacional.

El código **OF BUTTON:Si** se ejecuta cuando el usuario ha presionado el botón Yes de la caja de diálogo MESSAGE. La sentencia **TARGET{PROP:Text} = 'Agrega número internacional'** cambia el título de la barra de título, y luego **?Telefono:Rec:Numero{PROP:Text} = '@S20'** cambia la máscara de ingreso ENTRY. La sentencia **Telefono:Rec:Numero[1] = '+'** coloca un signo + en la posición del primer carácter, luego lo muestra (**DISPLAY**) y **SELECT(? ,2)** coloca el punto de inserción del usuario en la segunda posición del control actual.

El código **OF BUTTON:No** es similar cambiando el texto de la barra de título de la ventana, y la máscara de ingreso del control.

Actualización del archivo de proyecto

Dado que hemos añadido una estructura FILE al programa, debemos añadir su controlador de archivos (*file driver*) al proyecto, para que pueda ser "linkeado" al programa. Si no se hace, se provocará un error similar a: "link error: TOPSPEED is unresolved in file Hola.obj".

1. Elija **Project** ➤ **Edit...**

Aparece la caja de diálogo Project.

2. Destaque **Database Driver Libraries** y oprima el botón **ADD File...**
3. Destaque **TOPSPEED** y luego oprima el botón **OK**.
4. Oprima el botón **OK** para cerrar la caja de diálogo **Project**.
5. Dé CLIC sobre el botón **Run**.
Se ejecuta el programa.

Código POO generado por los templates ABC

Cuando examine el código generado para usted por el Generador de Aplicaciones, observará algunas diferencias fundamentales con respecto al código que acabamos de escribir. Esto se debe a que las plantillas utilizan la biblioteca de construcción de aplicaciones de alto nivel (*Application Builder Class [ABC] Library*), que es un conjunto de clases de Programación Orientada a Objetos (POO).

El código que acabamos de escribir es “procedural”, no POO. Ahora daremos una breve mirada al código POO generado para mostrar cómo lo que ha aprendido se vincula al código que generan los *templates*. Será solamente un vistazo rápido, para destacar las diferencias principales.

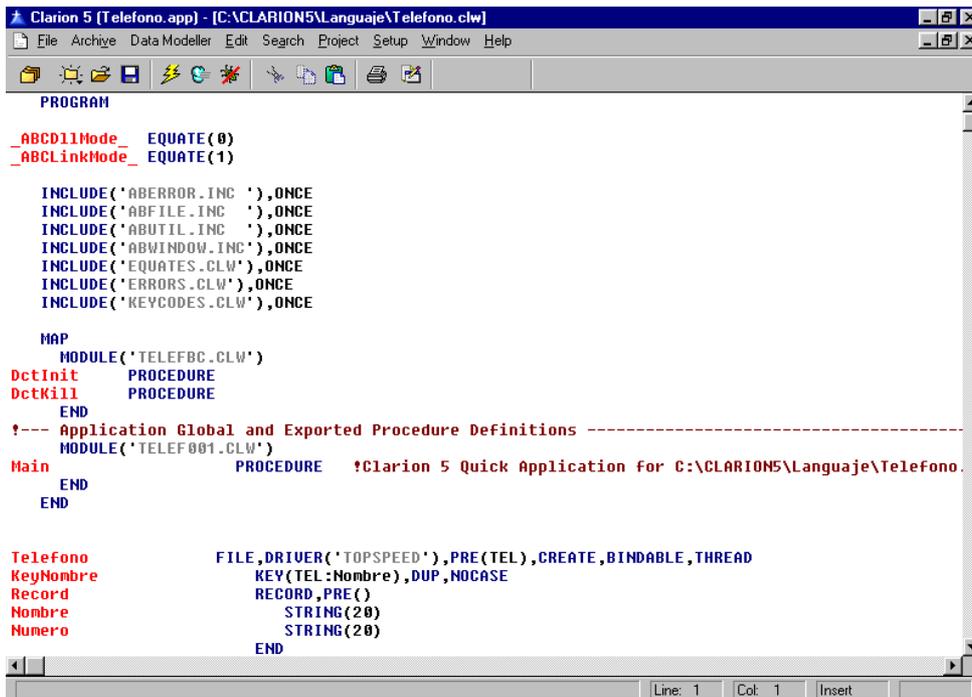
Empezar una aplicación con Quick Start

1. Elija **File** > **New** > **Application**.
2. Escriba *Telefono* en el campo **File Name** y marque el casillero **Quick Start**, y luego oprima el botón **Save**.
3. Nombre el archivo *Telefono*, defina los campos *Nombre* y *Numero* ambos con máscara @S20, e indique una clave *Duplicate*, y luego oprima el botón **OK**.
4. Elija **Project** > **Generate** para generar el código fuente.

Una mirada al programa fuente

Examinemos ahora el código fuente que se generó para usted.

1. Elija la lengüeta **Module** y destaque *Telefono.clw*
2. Dé CLIC DERECHO y elija **Module** del menú contextual.



```

Clarion 5 (Telefono.app) - [C:\CLARION5\Lenguaje\Telefono.clw]
File Archive Data Modeller Edit Search Project Setup Window Help

PROGRAM

_ABCD11Mode_ EQUATE(0)
_ABCLinkMode_ EQUATE(1)

INCLUDE('ABERROR.INC '),ONCE
INCLUDE('ABFILE.INC '),ONCE
INCLUDE('ABUTIL.INC '),ONCE
INCLUDE('ABWINDOW.INC '),ONCE
INCLUDE('EQUATES.CLV'),ONCE
INCLUDE('ERRORS.CLV'),ONCE
INCLUDE('KEYCODES.CLV'),ONCE

MAP
MODULE('TELEFBC.CLV')
DctInit PROCEDURE
DctKill PROCEDURE
END
!--- Application Global and Exported Procedure Definitions ---
MODULE('TELEF001.CLV')
Main PROCEDURE ?Clarion 5 Quick Application for C:\CLARION5\Lenguaje\Telefono.
END
END

Telefono FILE,DRIVER('TOPSPEED'),PRE(TEL),CREATE,BINDABLE,THREAD
KeyNombre KEY(TEL:Nombre),DUP,NOCASE
Record RECORD,PRE()
Nombre STRING(20)
Numero STRING(20)
END
Line: 1 Col: 1 Insert

```

El Program Module

Este es el módulo **PROGRAM** (el código debería verse parecido a este, pero podría no ser exactamente igual). La estructura básica de un programa Clarion POO es el mismo que utilizando el lenguaje procedural. Las primeras dos sentencias son **EQUATE**s que definen valores constantes que necesita la Biblioteca ABC. A continuación hay varias sentencias **INCLUDE**. Estas le indican al compilador que coloque el texto del archivo nombrado dentro del programa en el punto exacto de la sentencia **INCLUDE**.

```

PROGRAM
  _ABCD11Mode_    EQUATE(0)
  _ABCLinkMode_  EQUATE(1)
  INCLUDE('ABERROR.INC')
  INCLUDE('ABFILE.INC')
  INCLUDE('ABUTIL.INC')
  INCLUDE('ABWINDOW.INC')
  INCLUDE('EQUATES.CLW')
  INCLUDE('ERRORS.CLW')
  INCLUDE('KEYCODES.CLW')
```

Los primeros cuatro archivos **INCLUDE** (todos empiezan con "AB" y terminan con ".INC") contienen definiciones **CLASS** para algunas de las clases de la biblioteca ABC. Los siguientes tres archivos incluidos (todos con la extensión ".CLW") contienen un número de sentencias **EQUATE** utilizadas por los templates y la biblioteca.

```

MAP
  MODULE('TELEFOCB.CLW')
  DctInit    PROCEDURE
  DctKill    PROCEDURE
  END
!--- Application Global and Exported Procedure Definitions -----
  MODULE
  Main       PROCEDURE  ! Clarion 5 Quick Application
  END
END
```

La estructura **MAP** contiene dos estructuras **MODULE**. La primera declara dos procedimientos **DctInit** y **DctKill** que se definen en el archivo **TELEFOCB.CLW**. Estos dos procedimientos se generan para inicializar y desinicializar correctamente los archivos de datos para su uso por la biblioteca ABC. La segunda estructura **MODULE** simplemente nombra el primer procedimiento que llamará la aplicación (en este caso, *Main*).

Las siguientes dos líneas de código son las primeras sentencias POO:

```

Access:Telefono    &FileManager
Relate:Telefono    &RelationManager
```

La sentencia **Access:Telefono** declara una referencia a un objeto del **FileManager**, mientras que **Relate:Telefono** declara una referencia a un objeto del **RelationManager**. Esas dos referencias son inicializadas por el procedimiento **DctInit**, y desinicializadas por **DctKill**. Estas sentencias son muy importantes, porque definen la manera en que usted deberá dirigirse al archivo en el código POO.

Las siguientes dos líneas de código declaran un objeto **GlobalErrors** y otro llamado **INIMgr**.

```
GlobalErrors      ErrorClass
INIMgr           INIClass
```

Estos objetos manejan todos los errores y el archivo .INI de su programa (si lo hay), respectivamente. Estos objetos son usados extensamente por las demás clases de la biblioteca ABC, de manera que deben estar presentes (como analizaremos muy pronto).

```
GlobalRequest      BYTE(0),THREAD
GlobalResponse     BYTE(0),THREAD
VCRRequest        LONG(0),THREAD
Telefono          FILE,DRIVER('TOSPEED'),PRE(TEL),CREATE,BINDABLE,THREAD
KeyNombre         KEY(TEL:Nombre),DUP,NOCASE
Record            RECORD,PRE()
Nombre            STRING(20)
Numero           STRING(20)
                  END
                  END
```

Seguindo, hay tres declaraciones de variables globales que los templates ABC usan para comunicar los procedimientos, seguidas por la declaración **Telefono FILE**. Advierta que las variables globales tienen todas el atributo **THREAD**. Este se requiere, dado que, por omisión, los templates ABC generan una aplicación MDI, lo que hace necesario tener copias separadas de las variables globales para cada hilo de ejecución activo (que es lo que hace el atributo **THREAD**).

La sección global CODE tiene sólo seis líneas:

```
CODE
GlobalErrors.Init
DctInit
Main
DctKill
GlobalErrors.Kill
```

Las primeras dos sentencias llaman a métodos **Init** (en la jerga de la POO, un procedimiento en una clase es llamado "método": consulte **CLASS** en la *Language Reference* y los dos artículos sobre POO en la *Programmer's Guide*). Estos son los métodos constructores para los objetos **GlobalErrors** e **INIMgr**. Habrá notado que este último método lleva un parámetro. En la biblioteca ABC, todos los métodos constructores de objetos son llamados explícitamente y nombrados **Init**. Hay varias razones para esto. El lenguaje Clarion *admite* constructores (y destructores) automáticos de objetos, y es perfectamente "legal" usarlos en cualquier clase que usted escriba. Sin embargo, los constructores automáticos no pueden recibir parámetros y muchos de los métodos **Init** de la Biblioteca ABC deben recibir parámetros. Debido a esto, y para mantener

la coherencia, todos los métodos constructores de objetos ABC son llamados explícitamente y nombrados **Init**. Esta práctica añade el beneficio de una mayor legibilidad del código, dado que usted puede ver explícitamente que se está ejecutando un constructor, mientras que con constructores automáticos debe observar la declaración **CLASS** para ver si hay uno para ejecutar o no.

La llamada al procedimiento **DctInit** inicializa las variables de referencia **Access:Telefono** y **Relate:Telefono**, de manera que el código generado por los templates (y cualquier código que usted inserte), puede referirse a los métodos de archivo de datos utilizando la sintaxis **Access:Telefono.Methodname** o **Relate:Telefono.Methodname**. esto ofrece una forma coherente de referenciar cualquier archivo en un programa generado por las plantillas ABC: cada **FILE** tendrá sus objetos **Access:** y **Relate:** correspondientes.

La llamada al procedimiento **Main** empieza la ejecución del resto del programa. Una vez que el usuario vuelve al procedimiento **Main**, **DctKill** y **GlobalErrors.Kill** efectúan las operaciones de limpieza necesarias antes de regresar al sistema operativo.

El módulo UpdateTelefono

Examinemos ahora el código fuente que fue generado en uno de los procedimientos. Observaremos el procedimiento *UpdateTelefono*, que es representativo dado que todos los procedimientos generados por las plantillas ABC seguirán básicamente el mismo patrón (advertimos de nuevo, el código que usted encuentre será similar a este, pero no necesariamente igual)

1. Elija **File** ► **Close**
2. Destaque *UpdateTelefono*, y dé **CLIC DERECHO** y elija **Module** del menú contextual.

Lo primero que debe advertir es la sentencia **MEMBER** en la primera línea. Esta es una sentencia requerida que indica al compilador a qué módulo de programa “pertenece” este archivo fuente. También señala el comienzo de una “Module **Data** Section”: un área del código fuente en que pueden hacerse declaraciones de datos que son visibles a cualquier procedimiento en el mismo módulo, pero no fuera de él (vea *Data Declaration and Memory Allocation* en la *Language Reference*).

```

MEMBER('Telefono.clw')           !Este es un módulo miembro
INCLUDE('ABRESIZE.INC')
INCLUDE('ABTOOLBA.INC')
INCLUDE('ABWINDOW.INC')
MAP
  INCLUDE('TELEF004.INC')         !Declaración del procedimiento local del
módulo
END

```

Los tres archivos **INCLUDE** contienen definiciones **CLASS** para algunas de las clases de la biblioteca ABC. Adviértase que la lista de los archivos **INCLUDE** es distinta a la del nivel global. Solamente es necesario incluir las definiciones de clase que el compilador precisa conocer para compilar este código fuente únicamente. Es por ello que la lista de archivos **INCLUDE** sea algo diferente en cada módulo.

Advierta la estructura **MAP**. Por omisión, los templates ABC general “local MAPs” que contienen las sentencias **INCLUDE** para incluir los prototipos de los procedimientos definidos en el módulo y cualesquiera otros llamados desde el módulo. Esto permite una compilación más eficiente, dado que solamente se realizará una recompilación global cuando cambie algún dato global, y no

meramente por añadir un nuevo procedimiento a la aplicación. En este caso, no hay otros procedimientos llamados desde este módulo.

La sentencia **PROCEDURE** empieza el procedimiento **UpdateTelefono** (que también termina la Module Data Section).

```
UpdateTelefono  PROCEDURE                !Generado a partir del template – Window

CurrentTab      String(80)
FilesOpened     Byte
ActionMessage   Cstring(40)
History::TEL:Record  LIKE(TEL:RECORD),STATIC
```

A continuación de la sentencia **PROCEDURE** hay cuatro declaraciones. Las primeras dos son comunes a la mayor parte de los procedimientos generados. Proveen banderines locales usados internamente por el código del template. Las declaraciones **ActionMessage** y **History::TEL:Record** son específicas de un procedimiento Form. Declaran un mensaje para el usuario y un “área de guardado” para el uso de la tecla de historia del campo (*Field History Key*), también llamada tecla “ídem” ubicada en la barra de herramientas.

Después de la estructura WINDOW vienen las siguientes declaraciones de objetos:

```
ThisWindow      CLASS(WindowManager)
Ask              PROCEDURE(),VIRTUAL
Init            PROCEDURE(),BYTE,PROC,VIRTUAL
Kill            PROCEDURE(),BYTE,PROC,VIRTUAL
                END
Toolbar         ToolbarClass
ToolBarForn     ToolbarUpdateClass
Resizer         WindowResizeClass
```

Las tres últimas son simples declaraciones que crean los objetos locales que permitirán al usuario utilizar la barra de herramientas y ajustar el tamaño de la ventana. Lo interesante aquí es la declaración **ThisWindow CLASS**. Esta estructura de **CLASS** declara un objeto derivado de la clase **WindowManager** en la que los métodos **Ask**, **Init**, y **Kill** de la clase padre (**WindowManager**) son “pasados por alto” localmente. Se trata de métodos **VIRTUALES** lo que significa que todos los métodos heredados de la clase **WindowManager** podrán seguir llamando a los métodos pasados por alto. Este es un concepto muy importante en la POO, que se trata más profundamente en las referencias a **CLASS** y **OOP** en *Language Reference y Programmer's Guide*.

A continuación viene el código ejecutable de su procedimiento:

```
CODE
GlobalResponse = ThisWindow.Run()
```

Exactamente...¡una única sentencia! ¡La llamada a **ThisWindow.Run** es todo el código ejecutable del procedimiento! De esta manera que usted pregunta: “Pero, ¿dónde está todo el código que realiza todas las funciones que veo ejecutarse cuando ejecuto el programa?” La respuesta es: “¡En la biblioteca ABC!”, o al menos, la mayor parte. Lo bueno es que todo el código estándar para operar cualquier procedimiento está ya incorporado a la biblioteca, lo que disminuye notoriamente el “peso” total del programa, dado que todos los procedimientos comparten el mismo conjunto de código común que ha sido depurado (y que es improbable introduzca problema alguno a los programas que usted genere).

Toda la funcionalidad que específica a este procedimiento se genera en los métodos “pasados por alto”. En el caso de este procedimiento, hay solamente tres métodos que deben sufrir esa acción. Según sea la funcionalidad que usted solicita para el procedimiento, los templates ABC pasarán por alto los métodos que haga falta. Usted también tiene puntos de inserción disponibles en cada método que es posible “pasar por alto”, de manera que es fácil forzar a los templates a “pasar por alto” cualquier método que necesite funcionalidad algo diferente, simplemente añadiendo su propio código en esos puntos de inserción (usando el Editor de puntos de inserción del Generador de Aplicaciones).

Bien, veamos entonces los métodos “pasados por alto” en este procedimiento:

```
ThisWindow.Ask    PROCEDURE
                  CODE
                  CASE SELF.Request
                  OF InsertRecord
                     ActionMessage = 'Adding a Telefono Record'
                  OF Change Record
                     ActionMessage = 'Changing a Telefono Record'
                  END
                  QuickWindow{Prop:Text} = ActionMessage
                  PARENT.Ask
```

La línea verdaderamente interesante de este procedimiento es la última. **PARENT.Ask** llama al método padre que ha sido “pasado por alto” por este mismo procedimiento, para que ejerza su funcionalidad estándar. La palabra clave **PARENT** tiene mucho poder, ya que permite que un método pasado por alto en una clase derivada llame al método que reemplaza para que haga su trabajo, lo que permite al método pasado por alto incrementar la funcionalidad del método padre.

```
ThisWindow.Init  PROCEDURE()
                  CODE
                  SELF.Request = GlobalRequest
                  IF PARENT.Init() THEN RETURN Level:Notify.
                  SELF.FirstField = ?TEL:Nombre:Prompt
                  SELF.VCRRequest &= VCRRequest
                  SELF.Errors &= GlobalErrors
                  CLEAR(GlobalRequest)
                  CLEAR(GlobalResponse)
                  SELF.AddItem(ToolBar)
                  SELF.AddUpdateFile(Access:Telefono)
                  SELF:HistoryKey = 734
                  SELF.AddHistoryFile(TEL:Record,History::TEL:Record)
                  SELF.AddHistoryField(?TEL:Nombre,1)
                  SELF.AddHistoryField(?TEL:Numero,2)
                  SELF.AddItem(?Cancel,RequestCancelled)
                  Relate:Telefono.Open
                  FilesOpened = True
                  SELF.Primary &= Relate:Telefono
                  SELF.OkControl = ?Ok
                  IF SELF.PrimeUpdate() THEN RETURN Level:Notify.
                  OPEN(QuickWindow)
                  OPEN(QuickWindow)
                  SELF.Opened = True
                  Resizer.Init(AppStrategy:Surface,Resize:SetMinSize)
```

```

SELF.AddItem(Resizer)
Resizer.AutoTransparent = True
ToolBarForm.HelpButton = ?Help
SELF.AddItem(ToolBarForm)
SELF.SetAlerts()
RETURN Level:Benign

```

Hay varias líneas interesantes en **ThisWindow.Init PROCEDURE**. Este es el método constructor del objeto **ThisWindow**, de manera que todo el código que en él se incluye realiza las tareas de inicialización específicamente requeridas por el procedimiento **UpdateTelefono**.

La primera sentencia **SELF.Request = GlobalRequest**, recupera el valor de la variable global y la coloca en la propiedad **SEFT.Request**. **SELF** es otra poderosa palabra clave de la POO en Clarion, que siempre significa “el objeto actual”, o “yo”. **SELF** es el prefijo de objeto que permite que los métodos de clase se escriban genéricamente para referirse a cualquier objeto, como instancia de una clase, que se está ejecutando.

La segunda sentencia llama al método **PARENT.Init()** (el código del método padre para realizar todas sus funciones estándar) antes de que se ejecute el resto del código de inicialización específico del procedimiento. A continuación hay una serie de sentencias que inicializan varias propiedades necesarias. La sentencia **Relate:Telefono.Open** abre el archivo de datos Telefono para su procesamiento, y si hubo archivos hijos relacionados necesarios para el procesamiento de Integridad Referencia en este procedimiento, también los abriría (en este caso, no los hay).

Además de llamar al método **PARENT.Kill()** para realizar todas las funciones de cierre (como cerrar la ventana), **ThisWindow.Kill** cierra todos los archivos abiertos en el procedimiento, y configura la variable **GlobalResponse**.

Elija **File > Close**

Qué sigue después

Este cursillo ha sido solamente una breve introducción al lenguaje de programación Clarion y al código generado por los Templates ABC. Hay mucho más para aprender de lo que hemos cubierto aquí, de manera que ...¿dónde continuar?

- ◆ Los artículos en la *Programmer's Guide*. Estos cubren varios aspectos de la programación en este lenguaje. Aunque no son “cursillos”, proveen información que tratan con profundidad áreas específicas, y hay varios que tratan especialmente la POO en Clarion.
- ◆ El *Application Handbook* documenta completamente la biblioteca ABC y los templates. Este es el recurso fundamental para obtener lo máximo de la tecnología del Generador de Aplicaciones.
- ◆ La *Language Reference* es la “Biblia” del lenguaje Clarion. Siempre es buena idea leerlo por completo.
- ◆ Estudie el código fuente producido por el Generador de Aplicaciones. Después de haber hecho este cursillo, la estructura básica del código debería resultarle familiar, aunque no la lógica específica de determinadas secciones.

- ◆ La *User's Guide* contiene cursillos sobre el uso de los Depuradores (*Debuggers*). Esto le permitirá ir paso a paso por el código Clarion a medida que se ejecuta para ver exactamente qué efecto tiene cada sentencia en la ejecución de un programa.
- ◆ Los usuarios registrados de la Argentina y países de habla hispana tienen acceso a apoyo técnico vía Internet en la dirección <http://www.unisoft.com.ar> (o, también, www.clarion.com.ar) Correo electrónico: suporte@unisoft.com.ar
- ◆ TopSpeed ofrece seminarios educativos en varias localidades. Llame a Servicio al cliente a los números: (800) 354-5444 o (954) 785-4555 para informarse.
- ◆ En CompuServe escriba GO TOPSPEED para ingresar al foro de apoyo técnico de TopSpeed, o, en Internet, ingrese al grupo de noticias *comp.lang.clarion*, mediante cualquier servidor de Usenet (**¡Muy recomendado!**)

Muy buena suerte, y continúe trabajando. Como ya dijimos: ¡el poder de programación que Clarion pone en sus manos va creciendo a medida que aprende más sobre el!

VOCABULARIO

Inglés

Castellano

application builder class [ABC] library	= biblioteca de construcción de aplicaciones de alto nivel
application handbook	= manual de aplicaciones
application wizard	= asistente de aplicaciones
box	= casillero (para poner tildes), caja
break	= salto
browse	= visualización de listados
buffer	= área de almacenamiento temporal
business math library	= biblioteca de funciones de matemática empresarial
conditional behavior	= funcionamiento condicional
create	= crear
default	= por omisión, preconfigurado
Deployment kit user's guide	= Guía para, el kit de instalación
dialogue	= caja de diálogo
Dictionary Editor	= Editor del diccionario
drivers	= controladores
edit-in-place	= modificación in situ
embed points	= puntos de inserción
embedded code	= código de inserción
enterprise tools	= herramientas empresariales
entry box	= casillero de ingreso de datos
equate	= equivalencia
event-driven programming	= programación basada en eventos
field equate label	= rótulo de campo equivalente
file	= archivo
file schematic	= árbol de archivos
finish	= terminar, finalizar
forms	= fichas
handle	= asa, "manijita"
help	= ayuda
help contents	= contenido de la ayuda
incremental	= progresivo
incrementally	= progresivamente
initiate thread	= iniciar hilo de ejecución
keys	= claves

label	= rótulo; etiqueta
Lenguaje reference	= referencia del lenguaje
List Box Formatter	= Diseñador de Cajas de Listado
lookup	= consulta (a un archivo relacionado)
map	= mapa, mapeo, correlación, correlacionar
master index	= índice general
new	= nuevo
next	= siguiente
Objet Oriented Programming (OOP)	= Programación orientada a objetos (POO)
Page Form	= página base (en informes)
page overflow	= desbordamiento de página (en informes)
path	= vía o ruta de búsqueda
picture format	= formato de datos (P####P); patrón de visualización y archivo
picture token	= máscara de ingreso (o visualización) de datos
popup menu	= menú contextual
print engine	= motor de impresión (de informes)
procedure wizards	= asistentes de procedimiento
progress window	= ventana indicadora
quick load	= (asistente de) “carga rápida”
quick start	= (asistente de) “inicio rápido”
range limit	= límite de alcance (límite de búsqueda)
record filter	= filtro de registros
record order	= orden de los registros
report	= informe
report formatter	= diseñador de informes
reset on	= reinicializarse sobre
run	= ejecutar
save as type	= guardar como ...
setup	= instalación, configuración
tab	= tabulador, tecla de tabulación, lengüeta
ütally	= marcador
template	= plantilla
thred	= hilo de ejecución
tip	= acotación (leyenda aclaratoria)
total type	= tipo de total (suma, promedio, más alto, más bajo)
update	= modificación, actualización
user’s guide	= guía del usuario
Window Formatter	= Diseñador de Ventanas
wizards	= asistentes

Castellano

acotación
árbol de archivos
archivo
asa
asistente de aplicaciones
asistentes
asistentes de procedimientos
ayuda
biblioteca de construcción de aplicaciones de alto nivel
biblioteca de funciones de matemática empresarial
caja
caja de diálogo
casillero
casillero o caja de ingreso de datos
claves
código insertado
comprobaciones de validez
configuración
consulta
contenido de la ayuda
controladores
crear
desbordamiento de página
Diseñador de informes
Diseñador de Ventanas
Editor del Diccionario
ejecutar
fichas
filtro de registros
formato (P####P) o “patrón de visualización y archivo”
funcionamiento condicional
guardar como ...
Guía del programador
Guía del usuario
Guía para el kit de instalación
herramientas empresariales

Inglés

= tip
= File Schematic
= file
= handle
= application wizard
= wizards
= procedure wizards
= help
= Application Builder Class [ABC] Library
= business math library
= box
= dialog, dialog box
= box
= entry box
= keys
= embedded code
= validity checks
= setup
= lookup
= help contents
= drivers
= create
= page overflow
= Report Formatter
= Window Formatter
= Dictionary Editor
= run
= forms
= record filter
= picture format
= conditional behavior
= save as type
= Programmer's Guide
= User's Guide
= Deployment Kit User's Guide
= enterprise tools

hilo de ejecución	= thread
índice general	= master index
informe	= report
iniciar hilo de ejecución	= initiate thread
instalación	= setup
límite de alcance	= range limit
límite de búsqueda	= range limit
manual de aplicaciones	= application handbook
marcador	= tally
máscara de ingreso (o visualización) de datos	= picture token, picture format
modificación in situ	= edit-in-place
motor de impresión	= print page
nuevo	= new
orden de los registros	= record order
página base	= Page Form
plantilla	= template
programación en base a eventos	= event driven programming
Programación orientada a objetos (POO)	= Object Oriented Programming (OOP)
puntos de inserción	= embed points
referencia del lenguaje	= language reference
rótulo	= label
ruta (o vía) de búsqueda	= path
salto	= break
siguiente	= next
terminar, finalizar	= finish
ventana indicadora	= progress window
verificaciones de validez	= validity cheks
vía (o ruta) de búsqueda	= path
visualizador	= browse