Cuadernos Didácticos

Ingeniería Informática

Guía de referencia de Multibase Cosmos

B. Cristina Pelayo García-Bustelo
Universidad de Oviedo

Juan Manuel Cueva Lovelle
Universidad de Oviedo

Oviedo, Julio 1998

Cuadernos Didácticos
Ingeniería Informática

Cuaderno Nº 7

Guía de referencia de Multibase Cosmos

Autores:

B. Cristina Pelayo García-Bustelo
Universidad de Oviedo - España

Juan Manuel Cueva Lovelle
Universidad de Oviedo - España

Editorial:

SERVITEC

ISBN: 84-8416-001-7

Oviedo, Julio 1998

1ª Edición

Consultor Editorial
Juan Manuel Cueva Lovelle
cueva@lsi.uniovi.es

MULTIBASE

Guía de Referencia

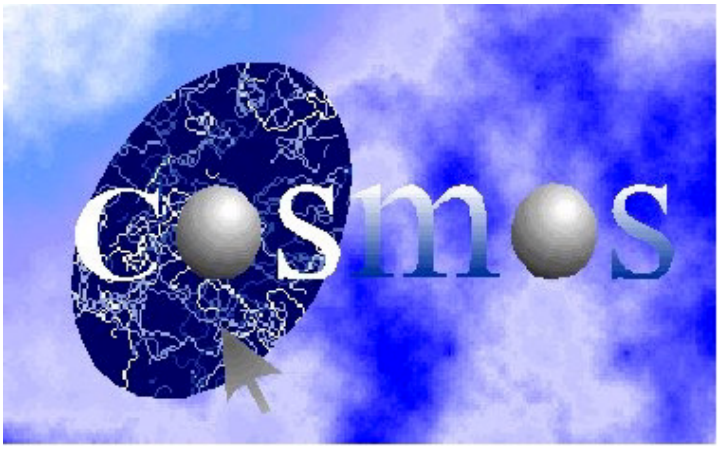


Tabla de Contenidos

Capítulo 1
El Entorno Windows

Contenidos

1. Introducción

Actualmente la mayoría de las aplicaciones que se desarrollan tienen una interfaz gráfica con ventanas. Así lo demuestra el éxito alcanzado por los entornos Macintosh, Windows95, X-Windows y Windows NT entre los usuarios. Esto provoca que la mayor parte de las aplicaciones que se desarrollan sean en este tipo de entornos.

En este capítulo se contemplaran algunos de los aspectos más relevantes del entorno Windows95.

2. Interfaces gráficas de usuario

Las interfaces gráficas de usuario o GUI (Graphics User Interface), también llamados "interfaz visual" o "entorno gráfico de ventanas", permiten el manejo de los programas de una forma más intuitiva y cómoda a los usuarios, por medio de gráficos, iconos y menús de ayuda.

Los conceptos básicos de los GUI fueron desarrollados a mediados de los años 70 en el centro de Investigación de Xerox en Palo Alto, en el proyecto liderado por Alan Kay, y denominado Dynabook, dentro del cual también se desarrollo el ratón y el lenguaje orientado a objetos puro Smalltalk.

Estas ideas no se popularizaron hasta el año 1984 con el lanzamiento al mercado del ordenador Macintosh.

Actualmente el interfaz gráfico de usuario es el consenso más importante en la industria de los ordenadores. Aunque los distintos entornos gráficos difieren en detalles, casi todos tienen características similares, habitualmente dadas por las especificaciones IBM SAA/CUA(*Systemes Application Architecture/Common User Access*), que dictan distintas normas sobre diseño de menús, aceleradores de teclado, cajas de diálogo, y otros aspectos de las GUI.

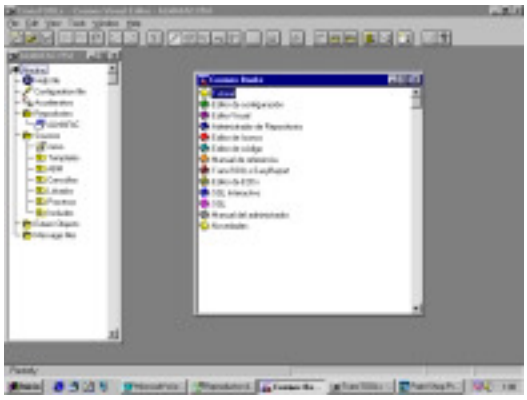


Figura 1.1 Interfaz gráfico de Usuario de Cosmos

Elementos principales

Los marcos de aplicación son tipos de librerías de clases que facilitan la tediosa, monótona y engorrosa tarea de desarrollar aplicaciones que requieren una interface de usuario sofisticada. Los marcos de aplicación ponen a disposición de las aplicaciones un soporte en tiempo de ejecución.

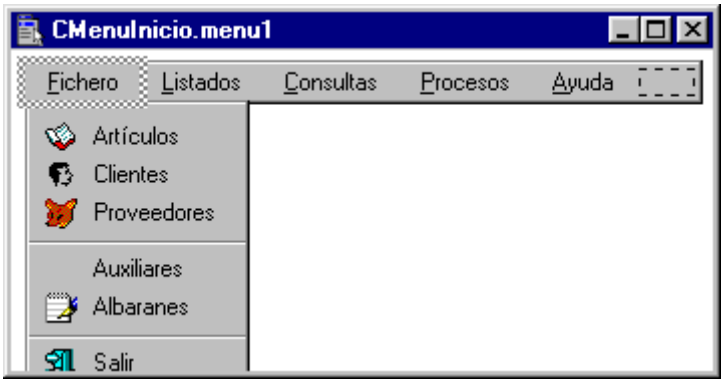
Los componentes básicos de un programa aportados por un marco de aplicación son las **vistas.** Las vistas son objetos visibles utilizados para comunicar información al usuario a través de la pantalla.

❑ Ventanas

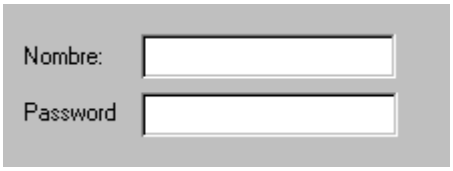
Una ventana es la forma más general de presentar un objeto, y pueden ser desde una región rectangular estática de la pantalla donde se presenta texto o gráficos, a objetos movibles con bordes, barras de desplazamiento, e iconos para mover, cerrar y redimensionar el marco de la ventana

❑ Menús

Un menú es un tipo especial de vista para presentar listas de elementos de menú. Un elemento de menú contiene a su ver otros objetos como subcomponentes. Un ejemplo típicode componente de menú contiene una cadena de caracteres o etiqueta de la opción de menú, una especificación de un acelerador de teclado, y una acción que será invocada al seleccionar el elemento del menú.



Cuadros de diálogo



Ofrecen un modo sencillo de especificar los parámetros de una aplicación controlador por el usuario.

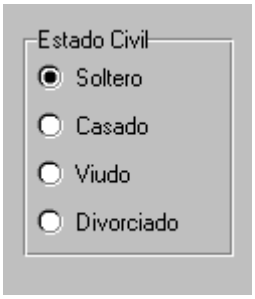
Botones

Un botón es un tipo de vista de control. Las vistas de control no se pueden mover, están fijadas en una posición dentro de una ventana movible. Cuando se activa envia mensajes de control al gestor de eventos. Para que se pueda activar un botón debe estar visible en la pantalla.

Existen tres tipos de botones: botones individuales, casillas de verificación y botones de radio.

- Botones individuales: permiten activar directamente comandos o acciones de la aplicación.

- Casillas de verificación: permiten activar o desactivar parámetros de valor biestado.
- Botones de radio: permiten especificar una opción de un conjunto de opciones excluyentes entre sí.



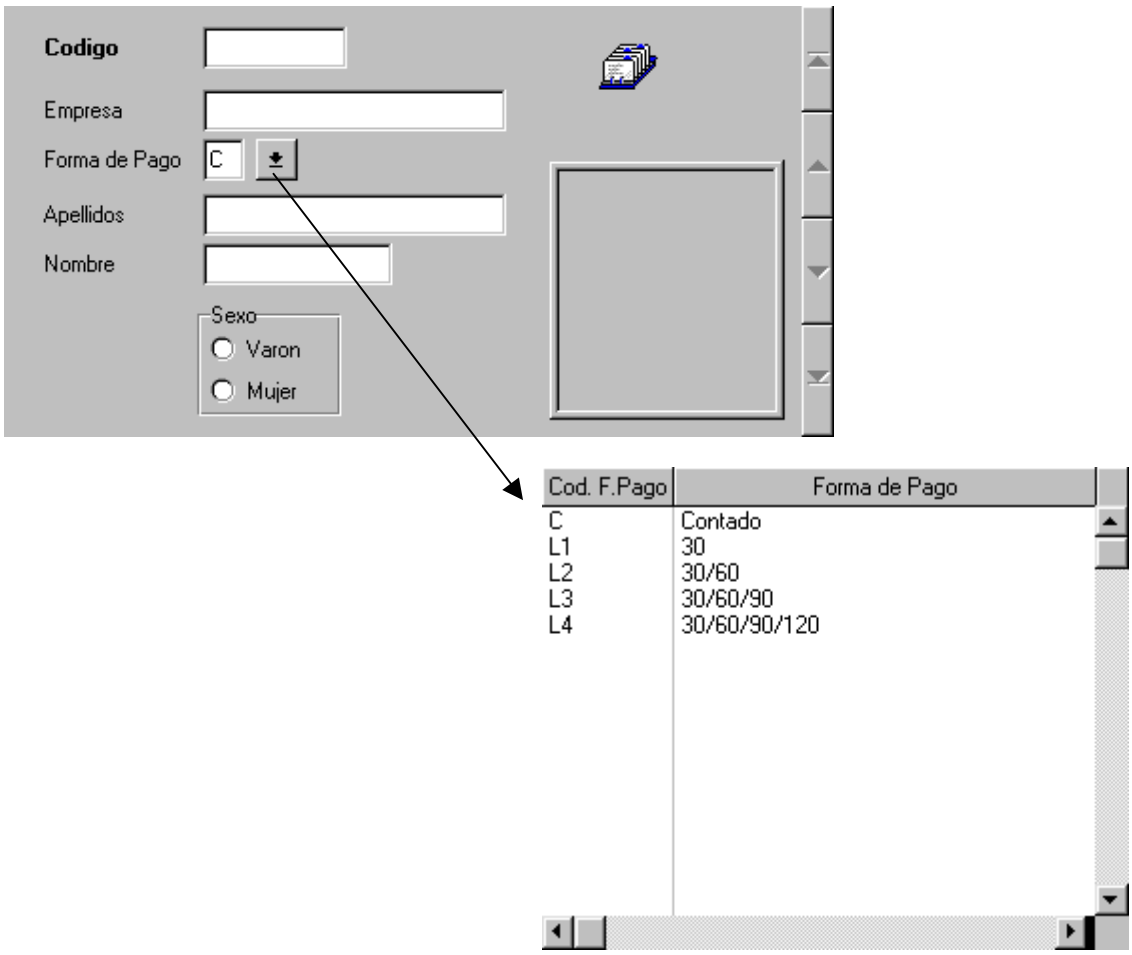
Los botones se reunen en grupos, a los cuales se les asigna un nombre de grupo. De esta forma se facilita al usuario la tarea de elegir las opciones.

Barras

Las barras de desplazamiento permiten especificar un porcentaje de un rango de posibles valores. Pueden ser horizontales y verticales.

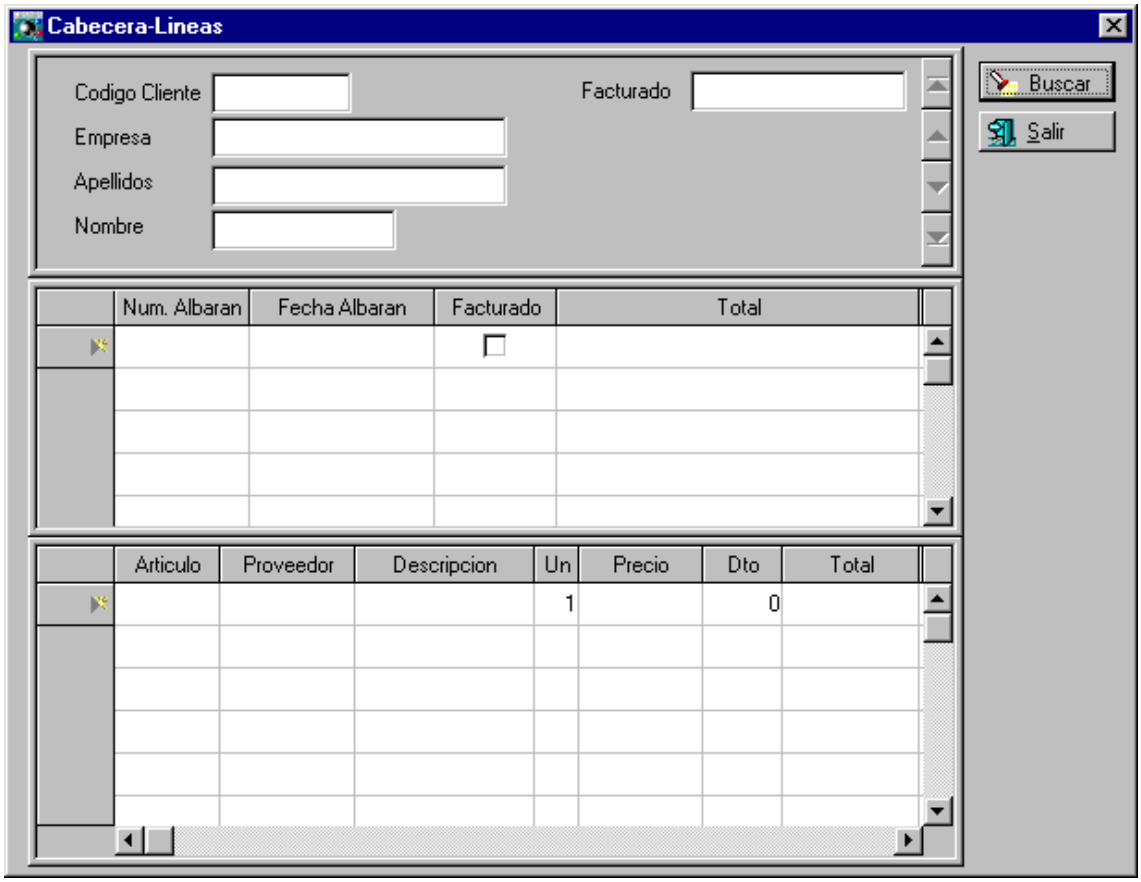
Listas de selección

Permiten al usuario escoger una cadena de caracteres de una lista de cadenas. Cada cadena está asociada con un objeto, que de esta forma puede ser seleccionado. Cuando la lista de selección tiene más elementos que el número de líneas del cuadro de diálogo que se utiliza para presentarla, aparecerá una barra de desplazamiento vertical que se usará para desplazarse a lo largo de la lista.



❑ *Vistas de textos*

Permiten visualizar y editar texto. Normalmente tienen barras de desplazamientos horizontal y vertical, iconos o símbolos de cerrar, mover, ampliar/reducir o redimensionar la vista. Las vistas de texto tienen métodos para visualizar o leer datos de la vista, trabajando con objetos de tipo string que pueden ocupar toda la memoria disponible para mantener la información de la lista si así se requiere.



3. Programación dirigida por eventos

Este paradigma de programación supone un cambio radical en la forma de desarrollar software. El usuario es quien dirige el programa. El programador ofrece al usuario un conjunto de acciones a realizar, y este las realiza en el orden que desee o incluso no las realiza. El programa se divide en un conjunto de pequeños módulos cada uno de los cuales implementa un servicio que se ofrece al usuario. La programación no es lineal, y es más complejo saber el estado del proceso.

La programación dirigida por eventos y la programación orientada a objetos suelen ir muy ligadas. Utilizando terminología de la POO, podríamos decir que en el espacio de trabajo de un sistema dirigido por eventos residen objetos que reciben eventos, o mensajes, y la respuesta a estos puede implicar el envío de otros eventos a otros objetos de este espacio

4. Utilización de multitarea

La multitarea se refiere a la capacidad que un sistema operativo tiene para compartir la CPU entre distintos programas. Una tarea es un programa cargado en memoria y que realmente este haciendo algo. Windows95 habla de procesos en lugar de tareas, aunque ambos términos son sinónimos en este aspecto.

El componente del sistema operativo que gestiona la multitarea en Windows95 es el planificador. El planificador trata principalmente con el tiempo y los sucesos. Un proceso(tarea) en Windows95 consigue un lapso de tiempo que determina durante cuanto tiempo puede usar la CPU. Al final del lapso del proceso, el planificador decide si permite que un proceso diferente utilice la CPU.

Los sucesos influyen en las decisiones del planificador. Para éste, una pulsación del ratón es un suceso que puede significar que deba asignar la CPU al proceso al que pertenece la ventana sobre la que se produjo la pulsación del ratón.

Windows95 utiliza multitarea cooperativa y multitarea con derecho preferente.

La planificación con derecho preferente pone a disposición del sistema operativo un control completo sobre qué proceso se ejecuta a continuación y por cuánto tiempo.

Con la multitarea cooperativa, el planificador puede conmutar entre procesos sólo cuando el proceso actualmente en ejecución entrega la CPU. Las aplicaciones deben devolver regularmente la CPU al sistema operativo.

5. Administración de memoria

La gestión de memoria en Windows tiene lugar a dos niveles diferentes: un nivel que ve el programador de la aplicación y otra visión totalmente diferente para el sistema operativo.

En la actualidad Windows95 tiene diversas capacidades nuevas de gestión de memoria a nivel de aplicación. Todas las funciones se refieren a la gestión de memoria dentro del espacio de direcciones de la aplicación, la memoria virtual privada asignada al proceso.

La gestión de memoria a lo largo de todo el sistema es responsabilidad del sistema base y la API (Application Program Interface, interface de programa de aplicación) de Windows intenta ocultar muchos de los pormenores de las funciones del sistema.

La mejora mas importante realizada en Windows95 es la introducción de una lista amplia de controladores de 32 bits en modo protegido, que se cargan cuando se carga Windows95. Algunos de ellos son: controladores de CD-ROM, de tarjetas de sonido, de red,...

También se proporcionan mayores recursos a Windows. Los recursos son esencialmente listas de memoria, que se llaman montones. Las listas apuntan a las áreas de memoria en las que están almacenados los elementos del interfaz de usuario (y otros elementos).

Windows95 ha sustituido los tres montones de 16 bits del recurso de Usuario por un único montón de 32 bits, que administra la parte de Interfaz de Usuario de Windows. Se ha mantenido el montón de 16 bits del recurso GDI (Graphic Device Interface, interfaz de dispositivos gráficos), que gestiona el dibujo de objetos en la pantalla por motivos de compatibilidad.

6. Desarrollo visual de aplicaciones

Por una parte utilizaremos el modelo vista controlador MVC. Es un marco para construir aplicaciones interactivas que esta basado en tres caminos. Fue diseñado como una parte de Smalltalk-80 pero tiene influencias de muchas otras arquitecturas orientadas a objetos.

Los tres componentes del MVC se corresponden con los tres componentes externos del interface de usuario.

El modelo representa la semántica de las entidades en el dominio de la aplicación.

Las instancias de la vista presentan el estado del modelo. Una vista y su modelo se conectan por mecanismos de dependencia.

Las instancias del controlador manejan la interacción entre el usuario final y el modelo o sus vistas. Los controladores implementan la acción de trasladar la pulsación de una tecla o el click del ratón en acciones sobre el modelo de objetos. Cada vista que permite iteracionar tiene asociado un controlador. Cuando se tiene una herencia de vistas también existe una herencia de controladores asociada.

7. Referencias Bibliográficas

[Collins95]	Collins, Dave *Designing Object-Oriented user interfaces*
[Cueva93]	Cueva Lovelle J.M., García Fuente MªP.A., López Pérez B., Luengo Díez Mª C., Alonso Requejo M. *Introducción a la programación estructurada y orientada a objetos con Pascal.* Distribuido por Ciencia-3, 1993.
[King94]	King, Adrian. *Inside Windows 95*
[Laurel90]	Laurel B., Mountford J. *The Art of Human-Computer Interface Design.*
[Livingston95]	Livingston B., Straub D. *Windows 95 Secrets*
[Microsoft95]	Microsoft Corporation, *Microsoft Windows 95. Resource Kit.*

Capítulo 2

Elementos de COSMOS

Contenidos

1. Introducción

Cosmos (COrporate Support MultiBase Object System) es una herramienta de programación visual orientada a objetos indicada para el desarrollo de aplicaciones de gestión, tanto en ámbitos corporativos de tipo cliente-servidor como para instalaciones departamentales y personales.

Cosmos incluye un entorno «RAD» (Rapid Application Development) de gran productividad, un lenguaje de cuarta generación orientado a objetos (COOL) y un gestor de base de datos relacional con interfaz SQL (CTSQL), todo ello complementado con un conjunto de herramientas auxiliares (Editor de Configuración, Editor de Código, Editor de Iconos, EasyReport, etc.).

En este capítulo se van a conocer los principales componentes de esta herramienta.

2. El Editor Visual

Cosmos se apoya en un Editor Visual altamente intuitivo y de manejo muy sencillo. Todas las herramientas necesarias para la realización de una aplicación se encuentran embebidas en el Editor Visual.

El Editor Visual proporciona una serie de asistentes (Wizards) que facilitan la programación.

El Editor Visual proporciona un conjunto de paletas que permiten la programación de la aplicación utilizando la técnica de arrastrar y soltar componentes a un lugar determinado del programa.

Otra de las características del Editor Visual es su capacidad para clasificar los objetos de un proyecto en función de las necesidades del programador. Esto permite añadir nuevas categorías de clasificación al proyecto y redefinir las ya existentes de forma dinámica.

El Editor Visual se invoca de dos formas diferentes:

- Si se hace doble click en el icono de Cosmos

- Por medio del comando Cosmos, cuya sintaxis es la siguiente:

```
Cosmos[-v][nombreProyecto]
```

Opciones	Significado
-h	Muestra en una ventana los parámetros de la línea de comandos.
-v	Muestra la versión del editor de Repositorio
nombreRepositorio.crf	Nombre del Repositorio con el que se desea trabajar

3. El Editor de Repositorios

Cosmos dispone de un Editor de Repositorios que automatiza el diseño de estructuras de datos. Estos repositorios, una vez definidos, pueden ser utilizados como paletas dentro del Editor Visual.

Un repositorio esta formado por el conjunto de datos necesarios para definir el esquema de una base de datos. También contiene información válida para el desarrollo de aplicaciones y para su generación automática.

Para invocar al editor de Repositorios existen dos formas:

- Haciendo doble click en el icono correspondiente:

- Mediante el comando:

```
Cosrep[-h][-v][nombreRepositorio.crf]
```

-h	Muestra en una ventana los parámetros de la línea de comandos.
-v	Muestra la versión del editor de Repositorio
nombreRepositorio.crf	Nombre del Repositorio con el que se desea trabajar.

El Editor de Repositorios permite la definición del tipo de interfaz gráfico para cada una de las columnas de la base de datos: en el momento de arrastrar una columna desde una paleta de repositorio hasta un diálogo, esta columna aparecerá en el diálogo con el interfaz gráfico que se haya definido.

El Editor de Repositorios permite tanto la importación de esquemas de bases de datos desde una base de datos ya creada como el proceso inverso, esto es, la generación de un fichero SQL que incluya todas las instrucciones necesarias para la creación de una base de datos a partir de un repositorio determinado.

Asimismo, permite directamente la creación o actualización de una base de datos a partir de la información contenida en el repositorio y viceversa.

4. El Lenguaje de Programación: COOL

COOL (Cosmos Object Oriented Lenguage), es un lenguaje de programación de cuarta generación orientado a objetos (OO4GL). Dicho lenguaje es el utilizado en la generación automática de código y el que se deberá utilizar para completar el código escrito por el sistema.

En las versiones de desarrollo se dispone de un compilador para el COOL llamado Cosmake.

5. Gestor de Base de Datos: CTSQL

Cosmos dispone de un gestor de base de datos con interfaz SQL que le proporciona una total autonomía.

Entre otras características del CTSQL cabe destacar su potencia, su gran rendimiento y su sencillez de manejo. No requiere de la constante atención de un administrador de base de datos.

CTSQL puede funcionar tanto en modo local como en modo cliente-servidor, y en ambos casos tiene capacidad multi-servidor, es decir, una misma instancia del lenguaje de cuarta generación COOL puede mantener abiertas simultáneamente varias bases de datos a través de una única instancia de CTSQL.

Estas características permiten a Cosmos el desarrollo de aplicaciones en modo local, independientemente de la arquitectura del sistema en el que la aplicación será finalmente ejecutada.

Entre las extensiones más significativas del CTSQL podemos citar, entre otras, las siguientes:

- Integridad de diccionario
- Integridad referencial
- Secuencia de ordenación
- Cursores bidireccionales
- «Views» multitabla actualizables

El CTSQL incluye los siguientes sublenguajes:

- Lenguaje de Definición de Datos (DDL)
- Lenguaje de Control de Acceso a Datos (DCL)
- Lenguaje de Consulta (QL)
- Lenguaje de Manipulación de Datos (DML).

6. Generación de Informes

EasyReport es una herramienta que permite al usuario final definir sus propios informes a partir de la información contenida en una base de datos sin necesidad de conocer su estructura interna.

La conexión de EasyReport a través de MultiWay le permite el acceso en modo cliente-servidor tanto a bases de datos Cosmos como a las bases de datos más extendidas del mercado.
EasyReport está especialmente diseñado para realizar las siguientes tareas:

- Obtención dinámica de informes de la base de datos.
- Facilitar el acceso a esta información a un usuario no experto.
- Ocultar la estructura interna de la base de datos al usuario final (nombres de tablas, columnas, etc.)
- Generar y catalogar informes parametrizables para su posterior ejecución.

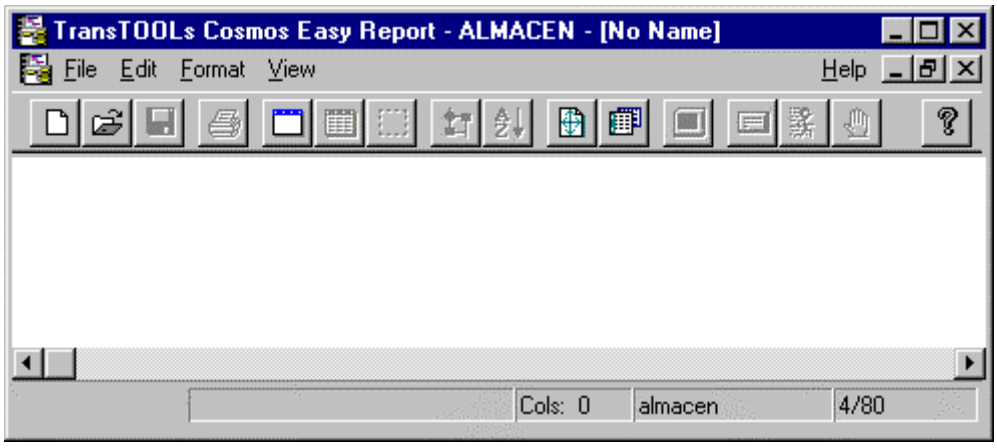


Figura 2.1. Aspecto del EasyReport

7. Herramientas auxiliares

Editor de Código Fuente

Cosmos incluye un Editor de Código fuente, de nombre «CEdit», que permite la edición de ficheros de texto. Si bien su manejo es similar al de otros editores existentes para Windows, «CEdit» incorpora, además, algunas técnicas de edición propias.

Editor de Iconos

El Editor de Iconos permite añadir, borrar, modificar y editar ficheros de iconos. Estos ficheros podrán ser usados como paletas dentro del Editor Visual. Se pueden arrastrar iconos sobre las opciones de un menú, sobre controles de tipo «botón» y sobre algunos otros controles de una «screen» de un «Form» o una página de impresión.

El Editor de Iconos permite crear dibujos sencillos o elaborados a color.

Sql Interactivo

El SQL Interactivo permite generar y ejecutar ficheros con extensión «.sql» que contienen instrucciones pertenecientes al CTSQL.

El SQL Interactivo es una herramienta que permite un interfaz directo con una base de datos tipo MultiWay u ODBC. Se puede utilizar para crear, modificar o consultar bases de datos tanto locales como remotas (en red local o cliente-servidor). Asimismo, a través del «gateway» apropiado, podrá acceder a otras bases de datos de distintos fabricantes.

Por otra parte, el SQL Interactivo permite también la realización de ciertas operaciones relativas al tratamiento de ficheros SQL, como por ejemplo:

- Crear ficheros SQL.
- Trabajar con ficheros SQL existentes.
- Ejecutar ficheros SQL.
- Seleccionar la base de datos.
- Establecer conexiones.

8. Conexiones

Cosmos dispone de dos mecanismos diferentes de conectividad. El primero de ellos, a través del estándar ODBC de Microsoft, asegura un intercambio de datos con otras aplicaciones en los entornos gráficos más actuales. El segundo, MultiWay, permite la conexión en cliente-servidor tanto con bases de datos propias de TransTOOLs (TransTOOL 3.x, MultiBase y Cosmos) como con las bases de datos con interfaz SQL más extendidas del mercado. Este tipo de conexión a través de MultiWay presenta la ventaja de no necesitar modificación alguna en los programas fuentes.

MultiWay es una tecnología desarrollada por TransTOOLs que dota a Cosmos de unas extraordinarias prestaciones en instalaciones con arquitectura cliente-servidor.

MultiWay es una capa de software que permite al lenguaje de cuarta generación de Cosmos (COOL), y por tanto a cualquier aplicación desarrollada con él, el acceso transparente a diferentes gestores de bases de datos. Dicho acceso puede hacerse en modo local (COOL y servidor de base de datos en una misma máquina) o en modo cliente-servidor (COOL y servidor de base de datos en máquinas distintas).

Para ello, MultiWay se apoya en diferentes estándares del mercado, el SQL como lenguaje de interfaz con bases de datos relacionales, el TCP/IP como protocolo de comunicaciones y la librería «WINSOCK.DLL» como estándar para TCP/IP en el caso del Windows (máquina «cliente»).

La arquitectura del sistema es sencilla y muy ligera, no exigiendo, entre otras cosas, el soporte de red proporcionado por el correspondiente fabricante del gestor de base de datos.

Esta capa de software, que se sitúa al lado del servidor de la base de datos, hace a Cosmos independiente de otras capas de software de red de otros fabricantes de bases de datos, permitiendo la conexión remota y el enlace lógico entre el lenguaje de cuarta generación COOL y el SQL de distintos fabricantes.

MultiWay se encarga de proporcionar una visión homogénea de cualquier SQL que se conecte al lenguaje de cuarta generación de Cosmos, ofreciendo un único «set» de tipos de datos entre los diferentes «sets» de los distintos gestores de base de datos y una única gramática, al tiempo que se encarga de que cualquier conversión de tipo sea realizada de forma ortogonal.

Por último, MultiWay se encarga de la traducción de sentencias de SQL entre los distintos gestores de bases de datos soportados.

Los gestores de bases de datos soportados por MultiWay son los siguientes:

- CTSQL de TransTOOLs.
- Oracle (versiones 6 y 7).
- Informix-OnLine e Informix-SE (versiones 4 y 5 y 7).
- Ingres (versión 6).
- DB2/6000 de IBM (versión 2.1).

Capítulo 3
Instalación de COSMOS

Contenidos

1. Introducción

Antes de proceder a la instalación es aconsejable comprobar que ha recibido todos los elementos que constituyen la Licencia de Cosmos. Salvo casos excepcionales, una Licencia de Desarrollo deberá incluir los siguientes elementos:

- Soporte magnético de la Licencia, que podrá ser en discos flexibles de 3,5 pulgadas o en un solo disco compacto (CD).
- Contrato de Licencia de Uso y Garantía de Licencia. Si su copia es en CD encontrará estos documentos en el interior de la carátula de portada de la caja que contiene el CD.
- Tarjeta de Usuario. Esta tarjeta le permitirá registrarse como usuario de Cosmos. Su cumplimentación es obligatoria, debiendo devolverla a TransTOOLs, S.A. para que el Contrato de Licencia de Uso y la Garantía de Licencia entren en vigor.

2. Requisitos de Hardware y Software

Los requisitos de software y hardware necesarios para el correcto funcionamiento de Cosmos son los siguientes:

Instalaciones en Monopuesto y Red Local (Cosmos y Cosmos/WG)

- Ordenador personal compatible con procesador 80486 o superior.
- RAM mínima de 8 Mbytes (recomendables 16 Mbytes).
- Microsoft Windows 3.11, Windows 95 o Windows NT.
- 20 Mbytes libres en el disco fijo (instalación completa).
- Unidad de discos flexibles de 3,5 pulgadas o lector de discos compactos (CDs) de doble velocidad como mínimo.
- Monitor VGA o superior.
- Ratón u otro dispositivo de puntero compatible.
- Asimismo, será preciso disponer de espacio suficiente en disco duro para la creación de los ficheros temporales de Cosmos.
- Las Licencias para red local precisarán, además, del software de red y de su correspondiente tarjeta.

Instalaciones en Cliente-Servidor

En este tipo de instalaciones deberá disponer, además, de los siguientes elementos:

- Una Licencia de Cosmos instalada en cada una de las máquinas clientes.
- Tarjeta de red cuyo driver esté soportado por la versión del paquete de comunicaciones que se vaya a emplear (consulte a su distribuidor del paquete de comunicaciones sobre cuáles son los tipos de tarjetas soportadas).
- Software de red que contenga la librería estándar WINSOCK.DLL para comunicar Windows con el servidor UNIX o Windows NT (por ejemplo, el paquete PC/TCP Network software for Windows, versión 2.3).
- TCP/IP para comunicaciones UNIX o Windows NT con otros sistemas. Este paquete deberá estar instalado en la máquina servidor.
- Las máquinas clientes Windows deberán cumplir asimismo los requisitos indicados anteriormente para la versión local (tipo de máquina, espacio en disco fijo, versión de Windows, etc.).

Instalaciones con Gateways

Este módulo es instalable únicamente en arquitecturas cliente-servidor. La máquina servidor deberá ser necesariamente UNIX, mientras que las máquinas cliente podrán ser DOS, Windows o también UNIX.

Además de los requisitos expuestos anteriormente para la versión en modo cliente-servidor, será preciso disponer también del gestor de base de datos sobre el que se desee trabajar:

- **INFORMIX:**

  - La máquina servidor precisará una Licencia Run-Time con el correspondiente gateway.
  - Servidor de la base de datos Informix: Informix-OnLine, versión 4, 5, 7 o SE.
  - Informix-ESQL Embedded Languages Runtime Facility.

- **ORACLE:**

  - La máquina servidor precisará una Licencia Run-Time con el correspondiente gateway.
  - Servidor de la base de datos Oracle, versión 6.0 ó 7.0.

- **INGRES:**

  - La máquina servidor precisará una Licencia Run-Time con el correspondiente gateway.
  - Servidor de la base de datos Ingres, versión 6.0 ó 7.0.

- **DB2/6000:**

  - La máquina servidor precisará una Licencia Run-Time con el correspondiente gateway.
  - Servidor de la base de datos DB2/6000, versión 2.1.

3. Diferencias entre Versiones

Cosmos está disponible en las siguientes versiones:

- Cosmos: Licencia monousuario de Desarrollo.
- Cosmos/WG: Licencia de Desarrollo hasta 3 usuarios.
- Cosmos/SQL-Desk: Licencia Run-Time.
- Cosmos/SQL-WG: Licencia Run-Time a partir de 3 usuarios.
- Cosmos/SQL-Enterprise: Licencia Run-Time para instalaciones en cliente-servidor a partir de 3 usuarios.

Las diferencias fundamentales entre las versiones de Desarrollo y las de Run-Time son las siguientes:

- Las versiones Run-Time no incorporan los compiladores ni el entorno de desarrollo..

- Las versiones Run-Time no incluyen el compilador cosmsg. En las versiones de Desarrollo este comando se encuentra en el subdirectorio bin de Cosmos. Su función es compilar los ficheros de mensajes.

- Las versiones Run-Time incluyen únicamente el programa cosrun para permitir la ejecución de aplicaciones desarrolladas con Cosmos, así como el SQL Interactivo, el Editor de Iconos, el Editor de Configuración, el Editor de Código y el EasyReport.

4. Instalación paso a paso

1. Al insertar el CD-ROM en la lectora se ejecuta automáticamente el programa de instalación. En ese momento aparece la pantalla de la Figura 3.1. solicitando al usuario los datos sobre la licencia que se encuentra impresos en el CD-ROM.

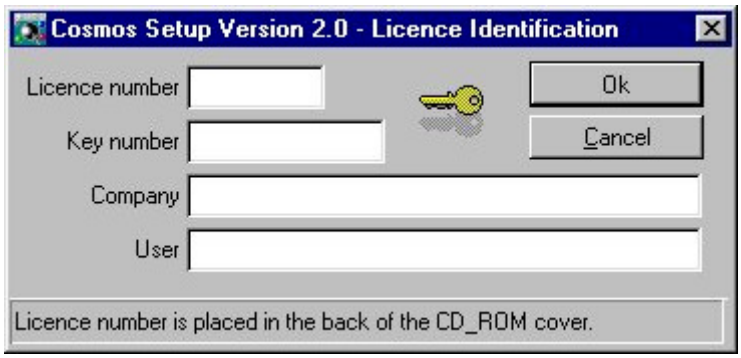


Figura 3.1. Solicitud de datos de la Licencia

2. El programa de instalación le permite elegir entre las diferentes opciones de instalación (Figura 3.2.).

   - Instalación de un entorno de desarrollo
   - Instalación de un Run-Time para distribuir aplicaciones
   - Instalación de Multway.
   - Instalación de los drivers ODBC para sistemas operativos de 16 o 32 bits.
   - Instalación del lector de fichero en formato PDF. Se utiliza para poder leer documentación incluida en el CD-ROM.

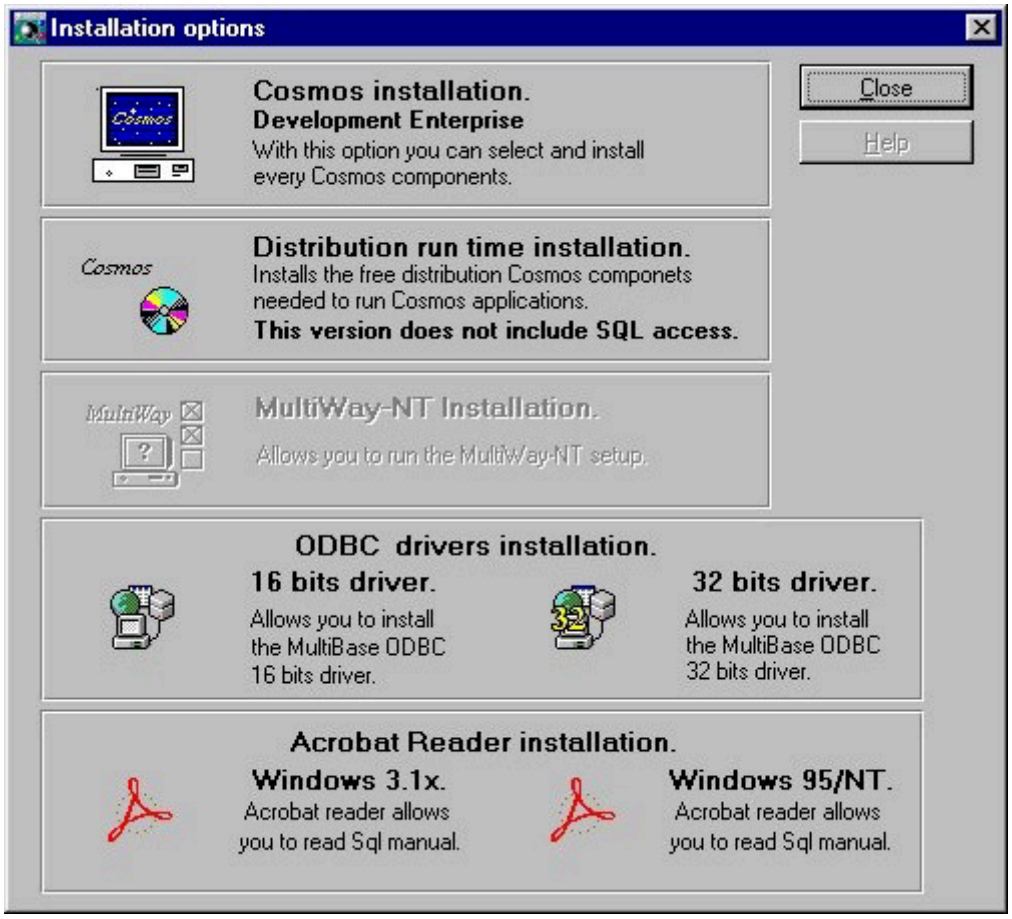


Figura 3.2. Opciones de Instalación

3. Instalación del entorno de desarrollo de COSMOS. Solicita el directorio destino donde se procederá a la instalación de los directorios y subdirectorios que contendrán los ficheros del entorno de desarrollo (Figura 3.4.).

Figura 3.4. Seleccionar directorio destino

4. Solicitud de opciones de instalación: se podrá elegir entre una instalación típica, personalizada o mínima (Figura 3.5.).

Figura 3.5. Seleccionar la forma de instalación

5. Indica si se desea instalar las aplicaciones de demostración y se permite seleccionar el directorio donde se almacenarán (Figura 3.6.).

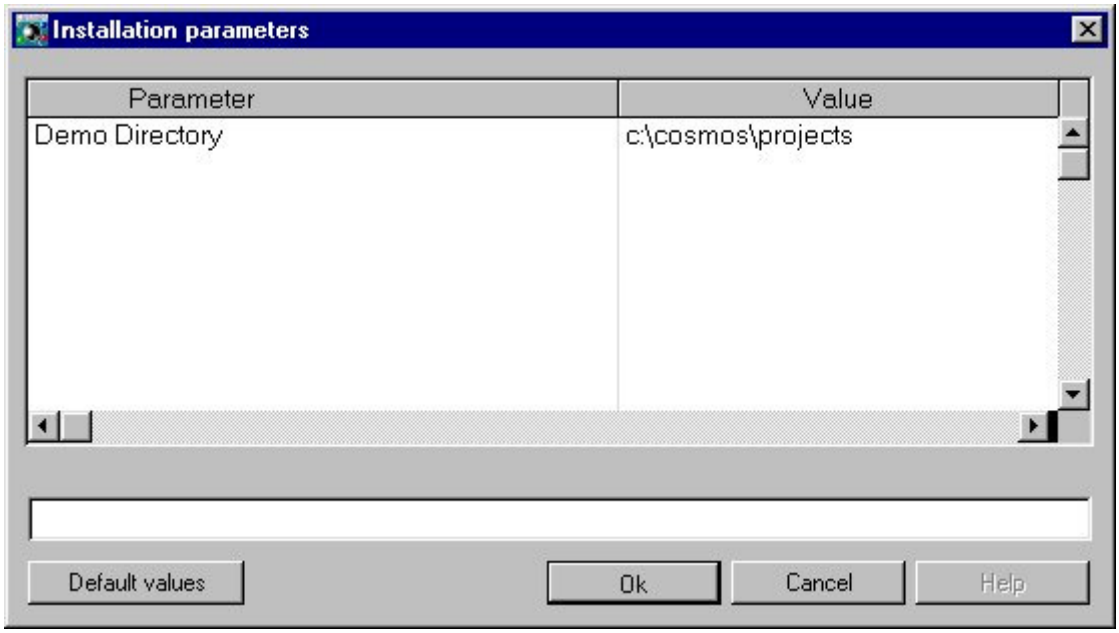


Figura 3.6. Instalación de aplicaciones de demostración.

6. Se solicita el grupo al que se desea que pertenezca el entorno de desarrollo COSMOS (Figura 3.7.).

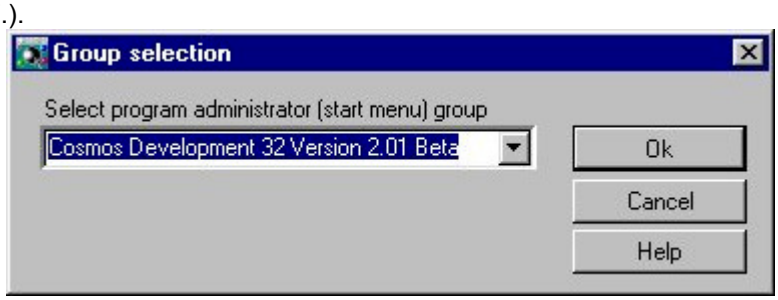


Figura 3.7. Selección de Grupo

5. Organización de los directorios

Cosmos se organiza en distintos directorios y subdirectorios dentro de la partición elegida para su instalación. Estos directorios y subdirectorios son:

- Bin
- Drw
- Etc
- Msg
- Projects
- Samples

Bin

Dentro de este directorio se encuentran todos los ficheros ejecutables correspondientes a los distintos comandos de Cosmos. Estos ficheros ejecutables son:

- Cedit	Editor de código fuente.
- Ceasyrep	Para la realización de informes a partir de la información contenida en una base de datos.
- Cosmos	Editor Visual.
- Cosmake	Compilador del lenguaje orientado a objetos de cuarta generación COOL (disponible únicamente en versiones de Desarrollo).
- Cosicons	Editor de Iconos.
- Cosinfo	Genera un listado con las propiedades, los métodos y eventos predefinidos en Cosmos.
- Coscds	Editor de Esquemas Conceptuales de Datos.
- Cosconf	Editor de Configuración.
- Cosrep	Editor de Repositorios.
- Cosrun	Ejecutor de programas Cosmos.
- Cosbooks	Este programa es un visualizador de ficheros de ayuda
- Csql	Sql Interactivo.
- Tchkidx	Chequeador de tablas de la base de datos.
- Tcollcom	Compilador de ficheros de secuencias de ordenación para el gestor de bases de datos (disponible únicamente en Versiones de Desarrollo).
- cosmsg	Compilador de ficheros de mensajes (disponible unicamente en versiones de Desarrollo).
- Tlisterr	Monitorización de posibles errores en las tablas de la base de datos.
- trepidx	Reparador de tablas defectuosas de la base de datos.

Drw

Dentro de este directorio se encuentras todos los ficheros con extensión .drw que contienen los cuadros de diálogo utilizados por los distintos comandos de Cosmos.

Etc

En este directorio se incluyen ficheros de iconos y el fichero de configuración de Cosmos (COSMOS.INI).

Msg

Incluye los ficheros de mensajes de Cosmos y la ayuda en línea.

Projects

Incluyen los programas fuentes de la aplicación de demostración y la base de datos utilizada por la misma.

Samples

Incluye otros ejemplos de demostración utilizados por la ayuda en línea.

6. Tipos y extensiones de los ficheros

Los tipos de ficheros que tienen contiene Cosmos son los siguientes:

Extensión	Tipo	Contenido
prj(1)	Bin	Estructura y configuración del proyecto
Pws	Bin	Estado de ejecución del proyecto abierto. Se crea cuando se guarda el proyecto por primera vez
Smd	Ascii	Código fuente de un programa, una librería o un include.
Omd(1)	Bin	Ficheros objeto generados por el compilador de Cosmos (cosmake). Durante el desarrollo de una aplicación estos ficheros tienen que encontrarse en el mismo directorio que los módulos fuente.
Shl	Ascii	Ficheros fuente que contienen los mensajes del usuario. Estos ficheros son de tipo ASCII, y por tanto editables con cualquier editor de textos. Durante el desarrollo de una aplicación estos ficheros tienen que encontrarse en el mismo directorio que los módulos fuente
Ohl(1)	Bin	Ficheros de mensajes compilados con cosmsg. Durante el desarrollo de una aplicación estos ficheros tienen que encontrarse en el mismo directorio que los ficheros fuente (ficheros con extensión .shl).
oms(1)	Bin	Similar a los anteriores, pero empleando la opción -s en el comando cosmsg.
Crf	Bin	Estructura de los repositorios. Estos ficheros son Generados por el Editor de Repositorios (cosrep.exe).
Bmp(1)	bin	Ficheros de iconos generados con el Editor de Iconos cosicons.exe.
Scs	ascii	Estos ficheros se encargan de componer el orden de Caracteres en la tabla ASCII para utilizarla en las operaciones de comparación y ordenación, y son asignados a la base de datos cuando ésta se genera con la instrucción CREATE DATABASE y la cláusula COLLATING. Por defecto, Cosmos incluye un fichero spanish.scs como ejemplo.
Ocs(1)	Bin	Ficheros objetos generados por el compilador de ficheros de ordenación (tcollcom).

Extensión	Tipo	Contenido
Sql[1]	Ascii	Ficheros con instrucciones del SQL. La forma de ejecutar estos ficheros es mediante el SQL interactivo (csql.exe), o bien con los métodos SqlFile y SqlToFile de la clase SqlServer predefinida en el lenguaje de Cosmos.
Unl[1]	Ascii	Ficheros generados al descargar una tabla. Estos ficheros contienen todas las filas de la tablas descargadas en formato ASCII, y su estructura es la generada por la instrucción UNLOAD: Cada línea del fichero corresponde con una fila de la tabla, diferenciando los campos mediante un separador que, por defecto, es la barra vertical (carácter ASCII 124 |).
dbs[1]	Bin	Directorio que representa la base de datos creada por el CTSQL. Las tablas de la base de datos se almacenarán en este directorio.
dat[1]	Bin	Fichero que contendrá los datos de una base de datos. La única forma de acceso a este fichero es mediante instrucciones del CTSQL. Este fichero tiene que estar Siempre en el mismo directorio que su respectivo *.idx.
idx[1]	Bin	F ichero que contendrá la estructura de todos los índices relativos a una tabla de la base de datos. Este fichero tiene que estar siempre en el mismo directorio que su respectivo *.dat. La elección del índice por el cual se desea acceder a los datos depende de la optimización de la instrucción SELECT del CTSQL (cláusulas WHERE y ORDER BY).
Trw[1]	Ascii	Ficheros que contienen la especificación completa de un Esquema Conceptual de Datos.
Orw[1]	Bin	Ficheros generados por EasyReport cada vez que el usuario guarda un informe, siendo informe el nombre asignado por el usuario.
Crw[1]	Bin	Fichero que contiene el catálogo correspondiente a un Esquema Conceptual de Datos. Este fichero incluirá los nombres y las descripciones de los informes guardados por el usuario.

[1] Los ficheros con estas extensiones son necesarios para la ejecución de una aplicación.

Los ficheros de extensión orw guardan (en formato no editable) no sólo la información correspondiente al informe generado, sino también la estructura completa del Esquema Conceptual de Datos (ECD) con el que se creó. Por lo tanto, una modificación en un ECD posterior a la creación de un informe no afecta en absoluto a éste. Al modificar el informe le aparecerá al usuario la estructura del ECD con la que se creó por primera vez el informe.

7. Instalación Cliente-Servidor

El funcionamiento en este tipo de arquitecturas consiste básicamente en lo siguiente: El lenguaje de programación de Cosmos (COOL) junto con los programas de la aplicación residen en la máquina cliente Windows, mientras que el gestor de la base de datos (CTSQL de MultiBase, Oracle, Informix, Ingres, DB2/6000, etc.), junto a la propia base de datos, se encuentran en el servidor UNIX o Windows NT (solo para el CTSQL).

El procedimiento de instalación se realiza de acuerdo a las siguientes fases:

- Puesta a punto de TCP/IP en el servidor UNIX o Windows NT.
- Puesta a punto de PC/TCP en la máquina cliente Windows.
- Comprobación de comunicación entre ambas máquinas.
- Instalación de un Run-Time para el gestor de base de datos en el servidor.
- Instalación de Cosmos en la máquina cliente.

Todos los ejemplos que se exponen en estas fases se refieren al gestor de base de datos de MultiBase (CTSQL). La forma de operar con otros gestores se explica en la siguiente sección de este mismo capítulo (Gateways).

El servidor podrá ser una máquina UNIX o Windows NT para el caso del gestor CTSQL y solo una máquina UNIX para el caso de los Gateways.

Puesta a punto de TCP/IP en el servidor:

Una vez instalado el paquete TCP/IP en la máquina servidor habrá que modificar los siguientes ficheros del servidor: hosts, services y inetd.conf (sólo para UNIX):

- **Fichero hosts**

Este fichero se encuentra en el directorio /etc para UNIX o winnt/system32/drivers/etc para Windows NT. Se deberán incluir todos los nombres y direcciones tanto del servidor como de las máquinas cliente. Una vez realizada esta operación, el fichero deberá presentar un aspecto similar al siguiente:

```
127.0.0.0       local     localhosts
125.0.0.1       nombre_servidor       sinónimo_servidor
125.0.0.2       nombre_cliente1       sinónimo_cliente1
125.0.0.3       nombre_cliente2       sinónimo_cliente2
125.0.0.4       nombre_cliente3       sinónimo_cliente3
```

- **Fichero inetd.conf (solo UNIX)**

Este fichero se encuentra en el directorio /etc y es utilizado por el proceso inetd, que es el encargado de invocar al gestor de base de datos cuando éste es requerido por algún cliente de la red.

En este fichero se deberá añadir la siguiente línea:

```
ctsql stream tcp nowait root $TRANSDIR/lib/ctsql ctsql system 2.0 0.0 NET
```

Debiendo sustituir $TRANSDIR por el nombre del directorio donde esté instalado el Run-Time para el gestor de base de datos en este servidor.

- **Fichero services**

Este fichero se encuentra en el directorio /etc para UNIX o winnt/system32/drivers/etc para Windows NT. En él se indica el nombre del gestor de la base de datos (por ejemplo CTSQL) junto al número de servicio que será común a servidor y cliente. En este fichero habrá que añadir la siguiente línea:

ctsql número/tcp ctsql

El número debe ser único en todo el fichero services, y deberá coincidir necesariamente con el de la máquina cliente que solicita el servicio.

Además si el servidor es Windows NT se realizarán también los siguientes pasos:

1. La versión del Run-Time para el gestor CTSQL en este servidor trae consigo el comando MBLISTENER.EXE (equivalente al comando inetd de UNIX) que es el encargado de invocar al gestor de base de datos cuando éste sea requerido por algún cliente de la red. Va acompañado del fichero de texto MBLISTENER.INI que debe encontrarse siempre en el mismo lugar que el primero (.EXE). El contenido de ese fichero de texto especifica por un lado, el directorio de instalación del gestor CTSQL en este servidor:

   TRANSDIR=c:/directorioraiz/directoriouno/directoriodos o

   TRANSDIR=c:\directorioraiz\directoriouno\directoriodos

   y por otro, si las bases de datos que cree el CTSQL junto con las tablas del catálogo tendrán la limitación de 8 caracteres en su nombre (como en MSDOS) o no:

   DOSNAME=ON o

   DOSNAME=OFF

   siendo éste último el valor por defecto.

   Si en algún momento se renombra el comando 'mblistener.exe', no hay que olvidar hacer lo mismo con el fichero 'mblistener.ini' ya que ambos van asociados.

2. Este comando que invoca al gestor CTSQL se instala en el servidor Windows NT como un servicio más con el nombre MBLISTENER. La línea de ejecución siguiente realiza esa labor:

   MBlistener -install [dominioNT\usuarioNT [password]]

   Si no se especifica "dominioNT\usuarioNT" se toma por defecto "Local System" y NULL como su password. Estos dos parámetros definen el entorno de seguridad en el que se ejecutará el servicio y los procesos CTSQL que éste ponga en marcha, dentro de la máquina NT.

   Después de la instalación y desde el panel de control de servicios del servidor lo veremos como muestra la figura 3.8.

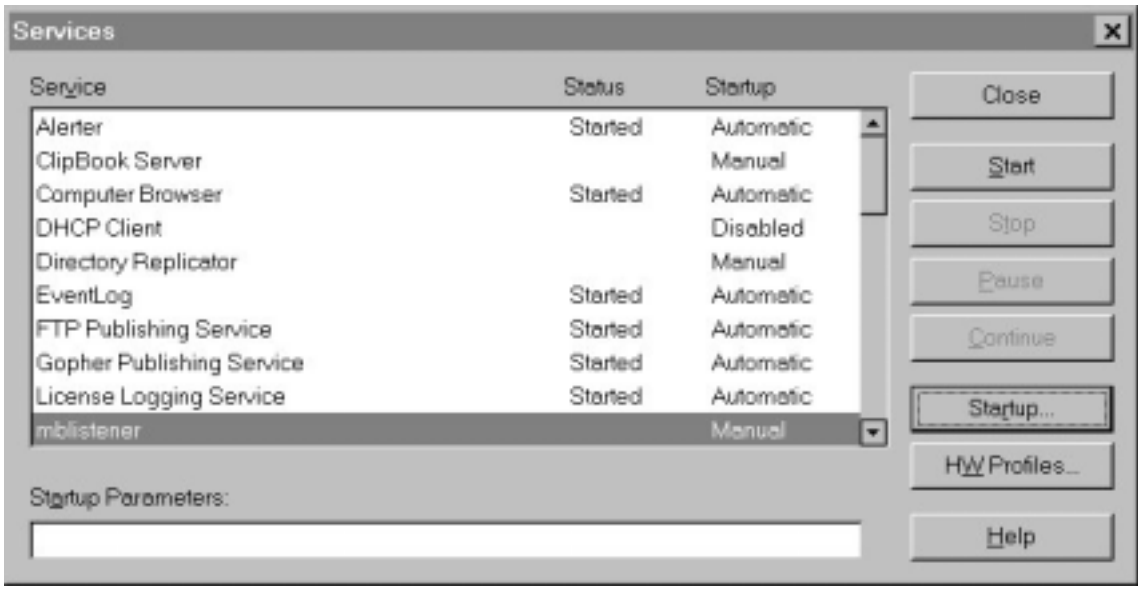
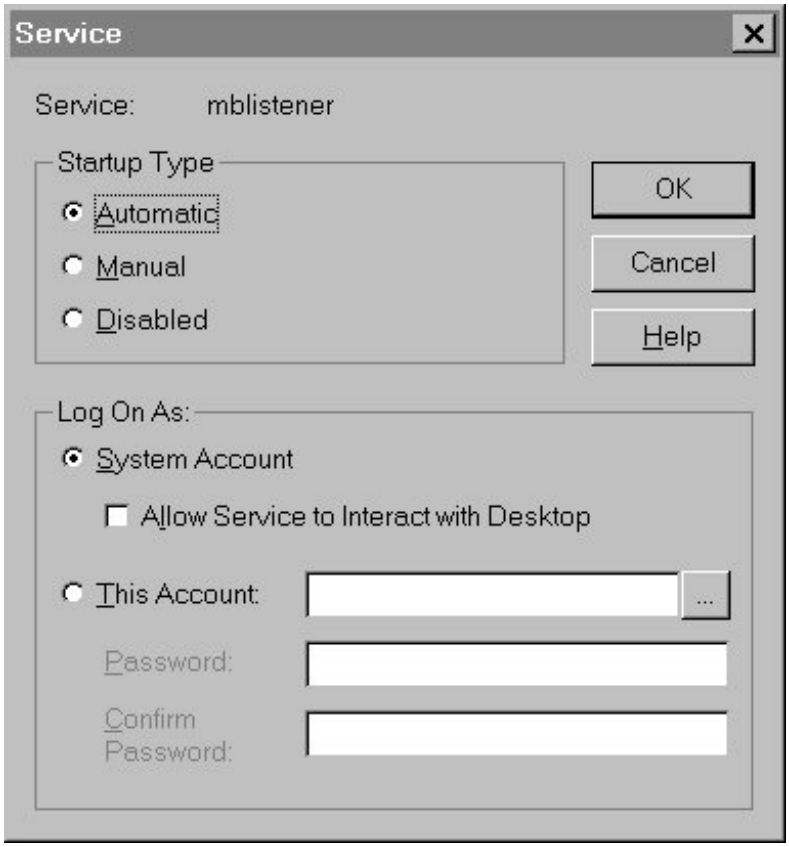


Figura 3.8. Servicios instalados

3. Como puede verse, queda instalado como un servicio que ha iniciarse MANUALMENTE ejecutando la opción "Start" del menú. Sin embargo, en cualquier momento puede cambiarse esta característica, si así se desea en el futuro, desde la opción "Startup".

   Si se configura como "Automatic", como muestra la figura 3.9., el servicio lo inicia el sistema Windows-NT cuando la máquina se pone en marcha.



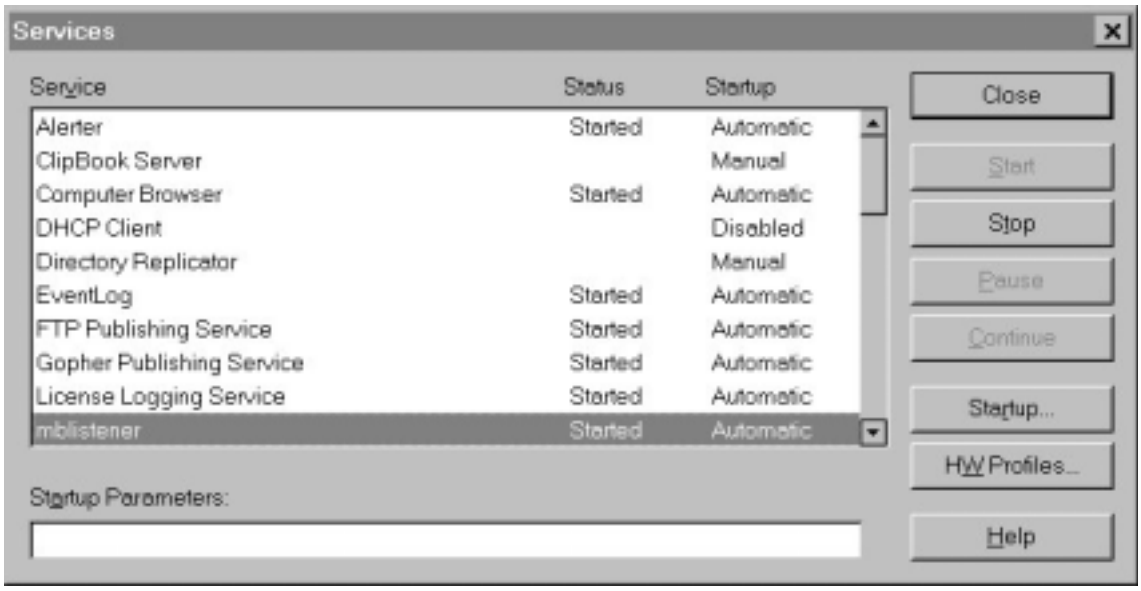


Figura 3.9. Servicio configurado para ser detectado automáticamente

4. Si en algún momento se cambia la localización ($TRANSDIR) del CTSQL.EXE se ha de parar el servicio que atiende peticiones (MBLISTENER) e iniciarlo de nuevo.

5. El servicio MBLISTENER se desinstala en el servidor NT a través de la línea de ejecución siguiente:

   MBlistener -remove

6. El fichero MBLISTENER.LOG, en el mismo lugar que los anteriores, recoge cualquier mensaje que se produzca durante la instalación, puesta en marcha y desinstalación del servicio.

   También guarda la identificación de las máquinas clientes que se conectan y cuando lo hicieron, así como si tuvieron éxito en comunicarse con un CTSQL.

   En caso de error proporciona alguna sugerencia sobre la causa del mismo.

Puesta a punto en la máquina cliente Windows:

Una vez instalado el paquete de comunicaciones (PC/TCP para Windows o similar —siempre que incluya la librería WINSOCK.DLL—) se deberán modificar los ficheros: \pctcp\hosts y \pctcp\services o los homónimos en el paquete de comunicaciones sobre Windows.

- **Fichero hosts**

En este fichero se deberán incluir las direcciones y los nombres de cada una de las máquinas que intervienen en la red (servidores y clientes), y su ubicación será normalmente el directorio \pctcp de la partición del disco donde se haya instalado el paquete PC/TCP.

El aspecto del fichero será prácticamente idéntico al comentado para UNIX:

```
125.0.0.1       nombre_servidor       sinónimo_servidor
125.0.0.2       nombre_cliente1       sinónimo_cliente1
125.0.0.3       nombre_cliente2       sinónimo_cliente2
125.0.0.4       nombre_cliente3       sinónimo_cliente3
```

Las direcciones y nombres asignados deberán coincidir en los ficheros equivalentes de todas las máquinas que conforman la red.

- **Fichero services**

Este fichero se encuentra normalmente en el directorio \pctcp. La modificación que habrá que realizar es similar a la descrita para este mismo fichero en la fase de instalación en el servidor, consistente en añadir la siguiente línea:

ctsql número/tcp ctsql

El número asignado a este servicio debe ser único en todo el fichero, debiendo coincidir asimismo con el número asignado al mismo servicio en la máquina servidor.

Comprobación de comunicación entre ambas máquinas:

Para comprobar que la instalación del paquete de comunicaciones (TCP/IP en UNIX o Windows NT y PC/TCP o similar en Windows) se ha realizado de manera satisfactoria, puede ejecutar el comando ping seguido del nombre de la máquina (el que se haya definido en el fichero hosts). Para mayor seguridad, ejecute dicho comando tanto en el servidor (UNIX) como en el cliente (Windows).

A continuación, compruebe si existe comunicación entre ambas máquinas (servidor y cliente). Para ello podrá utilizar también el comando ping seguido del nombre de la máquina (servidor si lo ejecuta desde la máquina cliente o viceversa).

En la máquina cliente Windows se podrá utilizar también el comando rloginvt para comprobar si la comunicación es correcta o no. Este comando convierte la máquina cliente en un terminal del servidor UNIX.

Hasta que la ejecución de cualquiera de los comandos anteriores no sea correcta habrá que administrar los paquetes de comunicaciones TCP/IP y PC/TCP o similar en las respectivas máquinas.

En el caso de que se produzca algún error tanto en la instalación como en el funcionamiento consulte a su distribuidor del paquete de comunicaciones.

Instalación de Cosmos en la máquina cliente:

Para instalar Cosmos en Windows consulte el apartado número 4 de este capítulo.

En este caso habrá que copiar el fichero WINSOCK.DLL del paquete de comunicaciones al subdirectorio bin de Cosmos. Dicho fichero deberá reemplazar al de igual nombre existente en el mencionado subdirectorio.

8. Instalación de Gateways (MultiWay).

Este módulo es instalable únicamente en arquitecturas del tipo cliente-servidor combinando diferentes sistemas operativos (Windows y UNIX).

Antes de comenzar la instalación compruebe si las máquinas cumplen los requisitos mínimos especificados.

La instalación de este módulo deberá seguir las siguientes fases:

- Instalación del gateway en cliente-servidor (UNIX).
- Puesta a punto en cliente-servidor:
  Configuración del servidor.
  Configuración del cliente.

Instalación del gateway:

La instalación de este módulo afecta únicamente al servidor UNIX.

Sólo en aquellos casos en los que el gateway se suministra en soporte separado al que contiene el Run-Time para el gestor de base de datos, el procedimiento para su instalación es el siguiente:

1. Acceda al sistema con el nombre del usuario donde haya instalado el Run-Time para el gestor de base de datos. Por ejemplo:

   ```
   Login: ctl
   Password:

   $
   ```

2. Cuando aparezca el prompt de la Bourne Shell ($), ejecute el comando su para pasar a superusuario:

   ```
   $ su
   Password:
   #
   ```

   Aparecerá un nuevo prompt de shell: #.

3. A continuación, copie el contenido del soporte magnético que contiene el gateway al disco fijo. Para ello, utilice el comando cpio con la siguiente sintaxis:

   ```
   # cpio -iBdcuv < NOMBRE_DISPOSITIVO
   ```

   El nombre del dispositivo dependerá del sistema operativo y de la versión que se esté utilizando.

4. Una vez copiada toda la información al disco fijo, ejecute el comando installgw desde superusuario. Este comando se encuentra en el subdirectorio /home del directorio donde haya instalado el Run-Time para el gateway, y su sintaxis es la siguiente:

   ```
   Login: ctl
   Password:

   $ su
   Password:
   # ./installgw
   ```

La ejecución del comando installgw tendrá que realizarse igualmente si la licencia de MultiBase ya incluye el gateway.

Puesta a punto:

El procedimiento de instalación es prácticamente idéntico al descrito para el gestor de base de datos CTSQL de MultiBase. La única diferencia estriba en que en este caso habrá que modificar los ficheros de comunicaciones /etc/services y /etc/inetd.conf (este último sólo en el servidor).

- **Configuración del servidor**

A continuación se comenta la configuración de los ficheros /etc/services y /etc/inetd.conf dependiendo del gestor de base de datos con el que se vaya a trabajar.

• INFORMIX:

En este caso el gateway tendrá que serializarse con el número de serie y la clave de activación de Informix. Para serializar el gateway hay que emplear el comando brand de Informix con la siguiente sintaxis:

```
$ INFORMIXDIR/etc/brand -s serie activ. gwinformix
```

- Fichero /etc/inetd.conf: Editarlo y añadir la siguiente línea (una sola):

```
gwinformix stream tcp nowait root $TRANSDIR/lib/gwinformix
gwinformix system 3.0 $TRANSDIR/etc/gwinformix.env NET
```

  para indicar el nombre del gestor de la base de datos. Se deberá sustituir $TRANSDIR por el nombre del directorio donde se haya instalado el Run-Time para el gateway.

- Fichero /etc/services: Se deberá indicar el nombre del gestor de la base de datos gwinformix. Para ello, editar el fichero y añadir la siguiente línea:

```
gwinformix  número/tcp gwinformix
```

  El número deberá ser único en todo el fichero y deberá coincidir con el de la máquina cliente [ver punto b)].

• ORACLE:

- Fichero /etc/inetd.conf: Se deberá añadir la siguiente línea (una sola):

```
gworacle stream tcp nowait root $TRANSDIR/lib/gworacle
gworacle system 3.0 $TRANSDIR/etc/gworacle.env NET
```

  para indicar el nombre del gestor de la base de datos. Se deberá sustituir $TRANSDIR por el nombre del directorio donde se haya instalado el Run-Time para el gateway.

- Fichero /etc/services: Se deberá indicar el nombre del gestor de la base de datos gworacle. Para ello, editar el fichero y añadir la siguiente línea:

```
gworacle número/tcp gworacle
```

  El número deberá ser único en todo el fichero y deberá coincidir con el de la máquina cliente [ver punto b) más adelante].

• INGRES:

- Fichero /etc/inetd.conf: Se deberá añadir la siguiente línea (una sola):

  gwingres stream tcp nowait root $TRANSDIR/lib/gwingres
  gwingres system 3.0 $TRANSDIR/etc/gwingres.env NET

  para indicar cuál será el gestor de la base de datos. $TRANSDIR hay que sustituirlo por el nombre del directorio donde se haya instalado el Run-Time para el gateway.

- Fichero /etc/services: Se deberá indicar el nombre del gestor de la base de datos gwingres. Para ello, editar el fichero y añadir la siguiente línea:

  gwingres número/tcp gwingres

  El número deberá ser único en todo el fichero y deberá coincidir con el de la máquina cliente [ver punto b) más adelante].

• DB2/6000:

- Fichero /etc/inetd.conf: Se deberá añadir la siguiente línea (una sola):

  gwdb2 stream tcp nowait root $TRANSDIR/lib/gwdb2
  gwdb2 system 3.0 $TRANSDIR/etc/gwdb2.env NET

  para indicar cuál será el gestor de la base de datos. $TRANSDIR hay que sustituirlo por el nombre del directorio donde se haya instalado el Run-Time para el gateway.

- Fichero /etc/services: Se deberá indicar el nombre del gestor de la base de datos gwdb2. Para ello, editar el fichero y añadir la siguiente línea:

  gwdb2 número/tcp gwingres

  El número deberá ser único en todo el fichero y deberá coincidir con el de la máquina cliente [ver punto b) más adelante].

• CTSQL de MultiBase:

La administración de los ficheros /etc/inetd.conf y /etc/services con el gestor de base de datos de MultiBase (CTSQL) se ha comentado en el epígrafe anterior de este mismo capítulo.

- **Configuración de la máquina cliente**

A continuación se comenta la configuración del fichero \etc\services en Windows dependiendo del gestor de base de datos que se vaya a emplear.

• INFORMIX:

Se deberá indicar el nombre del gestor de la base de datos gwinformix.

Para ello, editar el fichero /etc/services y añadirle la siguiente línea:

gwinformix número/tcp gwinformix

El número debe ser único en todo el fichero y tiene que coincidir con el del servidor UNIX.

• ORACLE:

Se deberá indicar el nombre del gestor de la base de datos gworacle. Para ello, editar el fichero /etc/services y añadirle la siguiente línea:

gworacle número/tcp gworacle

El número debe ser único en todo el fichero y tiene que coincidir con el del servidor UNIX.

• INGRES:

Se deberá indicar el nombre del gestor de la base de datos gwingres. Para ello, editar el fichero /etc/services y añadirle la siguiente línea:

gwingres número/tcp gwingres

El número debe ser único en todo el fichero y tiene que coincidir con el del servidor UNIX.

• DB2/6000:

Se deberá indicar el nombre del gestor de la base de datos gwdb2. Para ello, editar el fichero /etc/services y añadirle la siguiente línea:

gwdb2 número/tcp gwdb2

El número debe ser único en todo el fichero y tiene que coincidir con el del servidor UNIX.

Capítulo 4
Entorno COSMOS

Contenidos

1. Nociones básicas del Editor de Configuración

Este editor permite modificar el fichero de configuración COSMOS.INI sin necesidad de recurrir a un editor de textos.

El Editor de Configuración permite editar entornos para ver y modificar las variables que se tienen definidas.

El editor de ficheros de configuración se invoca por medio del comando cosconf, cuya sintaxis es la siguiente :

```
cosconf [-v] [-h] [fichero]
```

-h	Muestra esta lista de opciones.
-v	Muestra la versión del comando, así como su upgrade.
fichero	Nombre del fichero de configuración que se desea editar. Este nombre se indicará con su path completo y su extensión.

Véase Capítulo 7 para ampliar el conocimiento sobre el Editor de Configuración y su uso.

2. Fichero de Configuración COSMOS.INI

El fichero de configuración contiene algunas de las características con las que deberán arrancar las aplicaciones Cosmos. Este fichero se encuentra en el subdirectorio etc del directorio donde se encuentre instalado Cosmos.

Este fichero está especialmente diseñado para:

- Evitar un número excesivo de parámetros en la línea de comando de las aplicaciones Cosmos.

- Facilitar el paso de variables de entorno entre aplicaciones.

- Facilitar el manejo de las aplicaciones Cosmos al evitar al usuario el tener que introducir un número excesivo de datos para poder acceder a ellas.

3. Variables de Entorno

Tanto el sistema operativo como Cosmos poseen unas variables de entorno, que se utilizan para personalizar el sistema.

En Cosmos estas variables sólo se pueden definir en el fichero de configuración "COSMOS.INI" mediante el editor de configuración.

Todas las variables de entorno, se tomarán del fichero de inicialización y no de las variables de entorno MS-DOS.

El siguiente cuadro recoge las variables de entorno atendiendo a su función.

- **De Conexión**.

DBCHARSET	DBNAME	DBPASSWD	DBUSER
DBHOST	DBPATH	DBSERVICE	STRANSDIR
XDBTEMP			

- **De Formato**

DBDATE	DBMONEY	DBTIME	

- **De EasyReport**

CRWPATH	GRWPATH	TRWPATH	TRWPRIV

- **CTSQL**

ISAMBUFS	MBISFILES	OUTOPT	OUTOPT2
OUTOPT3	SORTMEM		

- **Gateways**

DBEMBED	DBLONGCHAR	DBSERVER	DBSQL
DBPATH	DBSYN	II_SYSTEM	INFORMIXDIR
MBCOMMIT	MBLCKTIMEOUT	DB2INSTANCE	DB2_UID
INGRES_UID	INGRES_DBA	MBLOADCOMMIT	ORACLE_DECIMAL
ORACLE_HOME	ORACLE_PROC	ORACLE_SID	ORACLE_UID

- **Diversas**

DBDELIM	DBEDIT	DBQUOTED	DBTEMP
LANG			

Véase Anexo D para ampliar el conocimiento sobre las Variables de Entorno.

4. Conectividad con Sistemas de Gestión de Bases de Datos

El Editor de Configuración permite definir conexiones locales, remotas (cliente-servidor) y ODBC para tener acceso a distintos servidores de base de datos.

Se da por lo tanto la posibilidad de establecer comunicaciones con otras bases de datos heterogéneas. Esto es posible gracias a la tecnología MultiWay soportada por el producto, que hace que el gestor SQL sea transparente a cualquier aplicación, ya sea ésta del propio Cosmos o bien del de otros fabricantes (Oracle, Informix, Ingres, Sybase, etc.).

Véase Anexo E para ampliar el conocimiento sobre las Conexiones a Bases de Datos

5. Conexión Cliente-Servidor CTSQL

El gestor de base de datos de Cosmos (CTSQL) puede funcionar tanto en modo local como en cliente-servidor, disponiendo en ambos casos de capacidad multiservidor. Es decir, una misma instancia del lenguaje de cuarta generación COOL puede mantener abiertas de manera simultánea varias bases de datos a través de una única instancia del CTSQL.

Véase Capítulo 6 .

6. Cliente-Servidor MultiWay

El segundo, MultiWay, permite la conexión en cliente-servidor tanto con bases de datos propias de TransTOOLs (TransTOOL 3.x, MultiBase y Cosmos) como con las bases de datos con interfaz SQL más extendidas del mercado. Este tipo de conexión a través de MultiWay presenta la ventaja de no necesitar modificación alguna en los programas fuentes.

MultiWay es una tecnología desarrollada por TransTOOLs que dota a Cosmos de unas extraordinarias prestaciones en instalaciones con arquitectura cliente-servidor.

MultiWay es una capa de software que permite al lenguaje de cuarta generación de Cosmos (COOL), y por tanto a cualquier aplicación desarrollada con él, el acceso transparente a diferentes gestores de bases de datos. Dicho acceso puede hacerse en modo local (COOL y servidor de base de datos en una misma máquina) o en modo cliente-servidor (COOL y servidor de base de datos en máquinas distintas).

Para ello, MultiWay se apoya en diferentes estándares del mercado, el SQL como lenguaje de interfaz con bases de datos relacionales, el TCP/IP como protocolo de comunicaciones y la librería WINSOCK.DLL como estándar para TCP/IP en el caso del Windows (máquina cliente).

La arquitectura del sistema es sencilla y muy ligera, no exigiendo, entre otras cosas, el soporte de red proporcionado por el correspondiente fabricante del gestor de base de datos.

Esta capa de software, que se sitúa al lado del servidor de la base de datos, hace a Cosmos independiente de otras capas de software de red de otros fabricantes de bases de datos, permitiendo la conexión remota y el enlace lógico entre el lenguaje de cuarta generación COOL y el SQL de distintos fabricantes.

MultiWay se encarga de proporcionar una visión homogénea de cualquier SQL que se conecte al lenguaje de cuarta generación de Cosmos, ofreciendo un único set de tipos de datos entre los diferentes sets de los distintos gestores de base de datos y una única gramática, al tiempo que se encarga de que cualquier conversión de tipo sea realizada de forma ortogonal.

Por último, MultiWay se encarga de la traducción de sentencias de SQL entre los distintos gestores de bases de datos soportados.

Los gestores de bases de datos soportados por MultiWay son los siguientes:

- CTSQL de TransTOOLs.
- Oracle (versiones 6 y 7).
- Informix-OnLine e Informix-SE (versiones 4 y 5 y 7).
- Ingres (versión 6).
- DB2/6000 de IBM (versión 2.1).

7. Conexión ODBC

Cosmos, a través del estándar ODBC de Microsoft, asegura un intercambio de datos con otras sistemas de gestión de bases de datos.

Capítulo 5
COOL: El lenguaje de programación de Cosmos

Contenidos

1. Introducción

El núcleo de Cosmos está constituido por un lenguaje de cuarta generación orientado a objetos, denominado COOL (Cosmos Object Oriented Language). Dicho lenguaje es el utilizado en la generación automática de código y el que se deberá utilizar para complementar el código escrito por el sistema.

COOL incluye una serie de clases predefinidas, además de proporcionar al programador las herramientas necesarias para definir sus propias clases y facilitar la programación de aplicaciones.

Al disponer de un conjunto de clases predefinidas, especializadas en las tareas más comunes de una aplicación Windows, COOL proporciona al programador toda la funcionalidad necesaria para el desarrollo de aplicaciones con una productividad elevada.

La sintaxis en COOL para ejecutar un método sobre un objeto sigue la secuencia objeto.método(parámetros), similar a otros lenguajes orientados a objetos.

La arquitectura modular de COOL permite la encapsulación de clases y objetos a los que se puede hacer referencia desde otros módulos.

Algunas de las características más importantes del COOL son las siguientes:

- Orientado a objetos.
- Modular: Una aplicación Cosmos estará compuesta por tantos módulos como se desee. Dichos módulos se cargarán dinámicamente en tiempo de ejecución.
- Estructurado.
- SQL con perfecto acoplamiento de tipos y estructuras.
- Moderno interfaz de usuario caracterizado por el empleo de distintos tipos de menús, cajas de diálogo, ayuda en línea sensible al contexto, colores, ventanas, ...
- Comunicación entre programas por medio de librerías e includes.
- Depurador interactivo.
- Funcionamiento transparente en arquitecturas cliente-servidor contra distintos SQL's.

2. Visión general del Lenguaje

Convenciones utlizadas

Las convenciones empleadas en Cosmos para presentar la sintaxis de cualquier elemento son las siguientes:

Elemento	Significado
[ ]	Aquellos elementos que se encuentren entre corchetes son opcionales
{ }	Indica la obligatoriedad de incluir uno de los elementos encerrados entre llaves
|	(carácter ASCII 124). Este carácter se utiliza para separar los diferentes elementos opcionales u obligatorios de la sintaxis, dependiendo de si éstos van entre corchetes o entre llaves
...	El elemento anterior a los puntos puede aparecer más veces en la sintaxis
,...	Como el caso anterior, pero cada nueva aparición del elemento debe ir precedida por una coma
palabra	Las palabras en minúsculas corresponden con identificadores COOL o expresiones, etc
PALABRA	Las palabras en mayúsculas y en negrita son reservadas del lenguaje COOL por tanto invariables
subrayado	Si no se especifica otra opción, la palabra o palabras subrayadas se toman por defecto
negrita	Cualquier signo de las convenciones que se encuentre en negrita es obligatorio en la sintaxis

El resto de signos se utilizan con su valor. Así, en aquellos lugares en los que se pueden especificar comillas, éstas podrán ser simples o dobles (", '), con la obligatoriedad de cerrar el texto con el mismo tipo con el que se abrió. Asimismo, se pueden incluir comillas como parte de un texto, en cuyo caso, dichas comillas deberán ser necesariamente del tipo contrario a las que sirven de delimitadores del texto. Es decir, en el caso de definir un literal entre comillas que incluya una parte también entrecomillada, habrá que combinar ambos tipos.

Módulos

Un módulo es un conjunto de constantes, clases, objetos y métodos que pueden ser compartidos por varias aplicaciones.

A medida que desarrolle proyectos mayores y más complicados puede ser difícil mantener todo el código en un solo archivo. En estos casos lo que se hace es distribuir el código en varios archivos, a los que se denomina módulos.

La división de un proyecto en pequeños módulos tiene las siguientes ventajas:

- El seguimiento de la lógica de su programa es más sencillo. Si todo el código de un proyecto está incluido en un solo fichero fuente, el tamaño de este hace que su programa sea difícil de seguir. Al dividir el programa en trozos más pequeños, de forma lógica, conseguirá que sea más fácil ver el diseño general del mismo.
- Los procesos de modificación y depuración del programa serán más sencillos. Podrá encontrar más fácilmente el código que está causando el problema.
- Puede desarrollar módulos compartidos (librerías o includes). A medida que se desarrolla una aplicación se crearán, probablemente, módulos que agrupan cierta funcionalidad. Esta funcionalidad vendrá definida por las clases y métodos definidos en cada módulo. Al crear un modulo que implementa una funcionalidad específica, éste será fácilmente utilizable por otro proyecto.

- Otra de las ventajas del uso de módulos, a parte de poder estructurar el código en múltiples archivos, es el hecho de que se puede compilar y usar posteriormente desde cualquier otro proyecto sin necesidad de disponer del código fuente. Esto le permite, por ejemplo, usar en múltiples proyectos el código que más utilice sin necesidad de tener que volver a escribirlo, y ni siquiera volver a compilarlo, tan solo tendrá que indicar que desea usar el módulo que lo contiene.

Al dividir un proyecto en módulos, deberá decidir la forma de a agrupar los métodos en los distintos ficheros. Para ello deberá considerar como hacer que el programa sea fácilmente depurable y modificable, y que métodos son las que va a poder reutilizar.

Se pueden agrupar los módulos de un proyecto basándonos en su funcionalidad.

Existen tres tipos de módulos Cosmos: Programas, librerías e includes.

Library	Una librería es un módulo que exporta (permite su uso desde otros módulos) sus métodos públicos. Sus métodos privados y protegidos, no se exportan y por tanto no pueden ser utilizados en el módulo que incluya la librería.
Include	Un Include es un módulo que exporta sus métodos públicos, clases públicas, objetos globales públicos y constantes. Así mismo un include exporta los métodos, clases, etc., públicos que incluye a su vez.
Program	Un programa no exporta nada de su contenido. Desde otro módulo sólo se puede acceder a él cargándolo en memoria por medio de las instrucciones Run y Load. La comunicación entre programas se establece a través de los parámetros pasados a la función Main del módulo llamado.

La gramática completa del lenguaje COOL se describe en el Anexo correspondiente.

Secciones de un módulo fuente COSMOS

Un módulo fuente Cosmos esta dividido en secciones, en cada una de las cuales se definen agrupados, los distintos componentes del módulo.

Si bien todas estas secciones son opcionales, su definición ha de realizarse obligatoriamente en el orden en que se describen.

- *Sección MESSAGES*

Sección cuya función es determinar el fichero de mensajes que utilizará el módulo por defecto

Sintaxis:

MESSAGES identifier

Parámetro	Significado
Identifier	Nombre de uno de los ficheros de mensajes registrados en el proyecto

- *Sección REPOSITORY*

Sección cuya función es determinar el repositorio de datos que se utilizará en la compilación del módulo. Su definición es obligatoria si existe un objeto o clase en el módulo que haga referencia a una tabla o columna contenida en el repositorio.

Sintaxis:

REPOSITORY identifier

Parámetro	Significado
Identifier	Nombre de uno de los repositorios registrados en el proyecto

- *Sección LIBRARIES*

Sección donde se declaran los módulos de tipo librería que necesita este módulo.

Sintaxis:

LIBRARIES BEGIN identifier ... END

Parámetro	Significado
Identifier	Nombre de uno de los módulos registrados como librería en el proyecto

- *Sección INCLUDES*

Sección donde se declaran los módulos de tipo include que necesita este módulo.

Sintaxis:

INCLUDES BEGIN identifier ... END

Parámetro	Significado
Identifier	Nombre de uno de los módulos registrados como include en el proyecto

- *Sección CONSTANTS*

Sección donde se definen las constantes del módulo.

Sintaxis:

CONSTANTS BEGIN constant_definition ... END

- *Sección CLASSES*

Sección donde se definen las clases de usuario en este módulo.

Sintaxis:

CLASSES BEGIN class_definition ... END

- *Sección OBJECTS*

Sección donde se definen los objetos globales (atributos) del módulo.

Sintaxis:

OBJECTS BEGIN object_definition ... END

- *Sección CODE CLASS identifier*

Sección donde se definen los métodos de la clase *identifier*. Dicha clase ha de estar definida en la sección CLASSES de este módulo. Por cada una de las clases definidas en el módulo puede haber una sección de este tipo asociada a ella.

Sintaxis:

CODE CLASS identifier BEGIN method_definition ... END

- *Sección CODE*

Sección donde estarán definidos los métodos del módulo.

Sintaxis:

CODE BEGIN method_definition ... END

3. Token, operadores y expresiones

El elemento básico reconocido por el compilador en un programa fuente es el token. Un token es una parte de un programa fuente tal que el compilador no necesita dividirla en más partes.

Caracteres tales como espacios, tabuladores, saltos de línea, retorno de carro, etc... se descartan salvo que formen parte de un string. Un token está limitado por estos caracteres y por otros token.

Los comentarios son una secuencia de caracteres que se descartan por el compilador. Pueden ser de una línea o de varias. El comentario de una línea comienza con los caracteres // y abarca hasta el final de dicha línea. El comentario multilínea comprende todos los caracteres entre { y }, pudiendo comprender más de una línea. Se utilizan para documentar el código y hacerlo más inteligible. Es posible anidar comentarios.

token ::= identifier | keyword | literal | string | operator | punctuator

Identificadores (identifier)

Un identificador (identifier) COOL es el nombre simbólico asociado a un componente Cosmos: objeto, clase, constante, método. Un identificador se diferencia de otro por los caracteres que lo componen. No se puede utilizar como identificador una palabra reservada.

Su longitud no puede exceder de 18 caracteres. Ha de ser único un su ámbito. Se distingue entre mayúsculas y minúsculas.

Palabras reservadas (keywords)

El lenguaje COOL hace distinción entre mayúsculas y minúsculas exceptuando las palabras reservadas del lenguaje (keyword). Véase anexo Palabras Reservadas del Lenguaje

Se consideran también palabras reservadas los identificadores de las clases predefinidas

Literales (literal)

Un literal es un objeto tal que su valor puede ser consultado pero no modifcado. Como objeto que es, pertenece a una clase y sobre él se pueden aplicar las operaciones de su clase. Hay dos tipos de literales:

- Un literal estático es aquel cuyo valor permanece siempre constante
- Un literal dinámico es aquel cuyo valor puede variar, por lo que no coincide con su nombre.

Ejemplos:

5 es un literal estático, pues su valor no varía. Además es un objeto de clase Smallint por lo que se le pueden aplicar las operaciones de esta clase.

Now es un literal dinámico de clase Time. Su valor varía continuamente.

Cadenas (string)

Las cadenas en COOL son objetos de la clase char. Las constantes de tipo cadena se representan entre comillas dobles. Ejemplo: “Hola a todos”

Operadores (operator)

Los tipos de operadores que existen son:

- Operadores lógicos
- Operadores de comparación
- Operadores aritméticos
- Operadores unarios

Separadores (punctuator)

El separador de instrucciones en COOL es el punto y coma.

Expresiones

Una expresión es una combinación de operadores y operandos que calcula un valor y que puede producir un efecto lateral. Los operandos en COOL pueden ser literales, identificadores, invocaciones de métodos u otras expresiones encerradas entre paréntesis. Toda expresión tiene una clase asociada, que es la del valor que evalúa.

Orden de evaluación de Expresiones

El resultado de la evaluación de una expresión en la que aparecen varios operadores depende del orden en que se evalúen estos. Este orden viene determinado por la prioridad de cada operador respecto a los demás, y por su asociatividad.

La expresión asociada al operador de mayor prioridad se evalúa antes que la del de menor prioridad. Si ambos operadores tienen igual prioridad, el orden de evaluación depende de la asociatividad del operador. Si la asociatividad del operador es de izquierda a derecha, se evalúa primero la expresión que esté más a la izquierda. Si la asociatividad del operador es de derecha a izquierda, se evalúa primero la expresión que esté más a la derecha.

La siguiente tabla muestra las relaciones de dependencia entre operadores así como su asociatividad. Los operadores que están en una misma fila, tienen igual precedencia entre sí, pero mayor que los de la fila anterior.

Operador	Asociatividad
=, +=, -=, *=, /=, %=	De derecha a izquierda
OR	De izquierda a derecha
AND	De izquierda a derecha
XOR	De izquierda a derecha
==, <>	De izquierda a derecha
>, >=, <, <=, MATCHES, LIKE, IN, IS NULL, IS NOT NULL	De izquierda a derecha
+, -	De izquierda a derecha
*, /, %	De izquierda a derecha
**	De izquierda a derecha
++, --, NOT, -	De izquierda a derecha

Dada una expresión en la que aparecen subexpresiones primarias (primary_expression), estas se evalúan antes de aplicar el operador.

Es posible alterar el orden de evaluación mediante el uso de paréntesis.
Ejemplo:
(expression1 op1 expression2) op2 expression3:
op1 se aplica antes que op2 independientemente de relaciones de precedencia y asociatividad de op1 y op2.

expression1 op1 (expression2 op2 expression3):

op2 se aplica antes que op1 independientemente de relaciones de precedencia y asociatividad de op1 y op2.

Condiciones

Una condición es una expresión cuya clase asociada es la clase predefinida Boolean. Las instrucciones de selección y de iteración especifican este tipo de expresiones en su sintaxis.

Cuando en la sintaxis de una instrucción se especifica una condición pero la expresión utilizada no es de clase Boolean, el compilador intenta convertir el valor de la expresión a la clase Boolean mediante la inserción de un método conversor a dicha clase.

Instrucciones

Una instrucción es la unidad de ejecución de un método. Controla la forma y el orden en que se utilizan los objetos.

Las instrucciones se clasifican en las siguientes categorías:

Instrucciones de invocación de métodos	Evalúan una expresión bien por sus efectos laterales o bien por el valor que devuelve. CALL.
Instrucciones vacías	Se utilizan dónde la sintaxis lenguaje requiere una instrucción, pero no se quiere realizar ninguna acción.
Instrucciones compuestas	Son una agrupación de instrucciones que se puede utilizar donde la sintaxis del lenguaje requiere una única instrucción. BEGIN ... END.
Instrucciones de selección	Evalúan una condición y dependiendo de su resultado ejecutan una o u otra instrucción. IF, SWITCH.
Instrucciones iterativas	Repiten la ejecución de una instrucción en función del resultado de la evaluación de una condición. DO, FOR, FOREVER, REPEAT, WHILE
Instrucciones de ruptura de secuencia:	Terminan la ejecución de un método o bien alteran la secuencia de ejecución de una instrucción iterativa. BREAK, CONTINUE, RETURN

Las instrucciones se ejecutan secuencialmente excepto cuando una instrucción de invocación a método, de ruptura de secuencia, de selección o de iteración, altera explícitamente dicha secuencia.

Instrucción CALL

Provoca la evaluación de una expresión. Su ejecución no provoca la transferencia de control o la iteración de otra instrucción.

Esta instrucción termina siempre con el caracter punto y coma(;). Todos los efectos laterales que pueda provocar la evaluación de la expresión se realizan antes de transferir el control a la siguiente instrucción a ésta.

Ejemplo:

```
//  2 instrucciones CALL idénticas
c = "hola mundo";
call c = "hola mundo";

//  4 instrucciones CALL idénticas
c.Trace;
c.Trace();
call c.Trace;
call c.Trace();

//  4 instrucciones CALL idénticas
(c + "!!").Trace;
(c + "!!").Trace();
call (c + "!!").Trace;
call (c + "!!").Trace();
```

Instrucción Vacia

Se utiliza cuando la sintaxis del lenguaje obliga a utilizar un instrucción, pero no se desea realizar ninguna acción.

Una utilización típica de esta instrucción es en el cuerpo de una instrucción iterativa.

Instrucción Compuesta

Agrupa una secuencia de instrucciones. Se utiliza cuando la sintaxis del lenguaje espera una instrucción pero se quiere ejecutar varias.

Una utilización típica es en el cuerpo de una instrucción iterativa o en la alternativa de una instrucción de selección, en las que se quiere ejecutar más de una instrucción.

Ejemplo:

```
  "Primera instrucción".Trace;
//  instrucción compuesta
begin
    "Entrando en una instrucción compuesta".Trace;
        ;
    "Saliendo de una instrucción compuesta".Trace;
end
"Última instrucción".Trace;
```

Control de flujo

Instrucción IF

Provoca la ejecución condicional de una instrucción.

Sintaxis:

if_statement ::= IF condition THEN statement1 [ELSE statement2]

Parámetro	Significado
condition	Condición de bifurcación
statement1	Instrucción a la que se cede el control si el resultado de evaluar condition es TRUE
statement2	Instrucción a la que se cede el control si el resultado de evaluar condition es distinto de TRUE

Si condition evalúa a NULL, por tanto es distinta de TRUE, se ejecuta statement2, si se ha especificado.

Ejemplo:

```
if today.Char[1,5] between "03-21" and "06-20"
    then
        "¡Es primavera!".Trace;
    else
        "¡No es primavera!".Trace;
```

Instrucción SWITCH

Provoca la ejecución condicional de una instrucción. Tiene dos posibles formatos:

- *Formato 1*

Recorre la lista de condiciones de cada rama CASE hasta que una de ellas evalúe a TRUE, en cuyo caso ejecuta la instrucción asociada.

Sintaxis:

switch_statement ::=

```
SWITCH
BEGIN
        [{CASE condition,... : statement1}...]
        [DEFAULT : statement2]
END
```

Parámetro	Significado
condition	Si evalúa a TRUE, provoca la ejecución de statement1
statement1	Se ejecuta si alguna condición de las lista de condiciones de su rama CASE evalúa a TRUE
Statement2	Se ejecuta si ninguna de las condiciones anteriores evalúa a TRUE

Su funcionamiento es equivalente a una serie de instrucciones IF anidadas. El orden en que se evalúan las condiciones de cada rama CASE es indeterminado y puede variar en futuras versiones. Una vez que una de las expresiones de una rama evalúa a TRUE, se dejan de evaluar el resto de las demás condiciones de ésta y de las demás ramas CASE .

Ejemplo:

```
public isposition(i as smallint)
begin
    switch
    begin
        case i>0: "i=2 es positivo".Trace;
        case i<0: "i=2 es negativo".Trace;
        default: "i=2 no es digito".Trace;
    end
end
```

- *Formato 2*

Ejecuta un conjunto de instrucciones dependiendo del valor resultante de evaluar una expresión.

Sintaxis:

```
switch_statement ::=
                    SWITCH expression1
                    BEGIN
                        [{CASE switch_condition,... : statement1}...]
                        [DEFAULT : statement2]
                    END

switch_condition ::= [switch_comparison_operator] expression
                    | IN (expression,...)
                    | IS [NOT] NULL
                    | BETWEEN expression AND expression

switch_comparison_operator ::= < | <= | > | >= | MATCHES | LIKE
```

Parámetro	Significado
Expression1	Es la expresión a comparar
Expression	Es la expresión con la que se compara expresión1, aplicando el operador indicado
Statement1	Se ejecuta si alguna condición de las lista de condiciones de su rama CASE evalúa a TRUE
Statement2	Se ejecuta si ninguna de las condiciones anteriores evalúa a TRUE

expresión1 se evalúa una única vez y antes de la evaluación de las expresiones de cada rama CASE

Si no se especifica switch_comparison_operator, se realiza una comparación por igualdad mediante el operador ==.

Ejemplo:

```
public isdigit(i as smallint)
begin
    switch i
    begin
        case 0: "i=2 es digito par".Trace;
        case 2: "i=2 es digito impar".Trace;
        default: "i=2 no es digito".Trace;
    end
end
```

Instrucción DO

Es una instrucción iterativa que comprueba su condición de finalización al final cada iteración.

Sintaxis:

do_statement ::= DO statement WHILE condition;

Parámetro	Significado
statement	Se ejecuta mientras condition evalúe a TRUE
condition	Si evalúa a TRUE, se produce una nueva iteración de statement. En caso contrario se cede el control a la siguiente instrucción a do_statement

Esta instrucción termina siempre con el caracter ';'.

Dado que condition se evalúa tras cada iteración, statement se ejecuta al menos una vez.

Es posible romper la secuencia de iteración mediante las instrucciones BREAK, CONTINUE o RETURN.

Intrucción FOR

Es una instrucción iterativa que comprueba su condición de finalización al inicio de cada iteración. Se recomienda su utilización cuando se conoce de antemano el número de iteraciones.

- *Formato 1*

Inicializa un objeto con un valor dado y ejecuta una instrucción repetidas veces hasta que el valor de dicho objeto deje de cumplir una condición dada.

Sintaxis:

for_statement ::= FOR identifier = expression1 [DOWN] TO expression2
[STEP expression3] DO statement

Parámetro	Significado
identifier	Referencia un objeto. Este puede ser de cualquier clase
expression1	Valor inicial que toma identifier
expression2	Valor que determina la continuación del bucle. Si se especifica DOWN, identifier ha de aceptar la operación >= con expresión2. En caso contrario ha de aceptar la operación <= con expresión2 Esta comparación se realiza al principio de cada iteración.Si el resultado la operación es TRUE, se ejecuta statement
expression3	Es opcional. Si no se especifica STEP, identifier ha de aceptar la operación –, si se ha especificado DOWN o la operación ++ si no se ha especificado DOWN. Si se especifica STEP, identifier ha de aceptar la operación -= con expression3, si se ha especificado DOWN o la operación += con expression3 si no se ha especificado DOWN. Esta operación se realiza al final de cada iteración
Statement	Instrucción que se itera

Las tres expresiones (expression1, expression2 y expression3) se evalúan una única vez al principio del bucle, y no en cada iteración de éste.

Es posible romper la secuencia de iteración mediante las instrucciones BREAK, CONTINUE o RETURN.

- *Formato 2*

Recorre una lista de expresiones, inicializando un objeto con cada una de ellas y ejecutando una instrucción.

Sintaxis:

for_statement ::= FOR identifier AS expression,... DO statement

Parámetro	Significado
identifier	Referencia un objeto. Este puede ser de cualquier clase
expression	Expresión que se asigna identifier al inicio de cada iteración. identifier ha de aceptar la operación = con expression
statement	Instrucción que se itera. Se ejecuta tantas veces como expresiónes aparezcan

Cada expresión de la lista se evalúa en el momento de realizar la operación de asignación con identifier

Es posible romper la secuencia de iteración mediante las instrucciones BREAK, CONTINUE o RETURN.

Ejemplo:

```
//  distintas maneras de calcular el sumatorio de 1 a 5
//  de 1 a 5 de 1 en 1
suma = 0;
for s = 1 to 5 do suma += s;
//  de 5 a 1 de 1 en 1
suma = 0;
for s = 5 down to 1 do suma += s;
//  de 0 a 10 de 2 en 2 sumando la mitad
suma = 0;
for s = 0 to 10 step 2 do suma += s / 2;
//  de 10 a 0 de 2 en 2 sumando la mitad
suma = 0;
for s = 10 down to 0 step 2 do suma += s / 2;
//  sumando expresamente 1 2 3 4 y 5
suma = 0;
for s as 1,2,3,4,5 do suma += s;
//  concatenando 'x' a c y sumando su longitud
suma = 0;
for c = "x" to "xxxxx" step "x" do suma += c.Length;
```

Instrucción FOREVER

Es una instrucción iterativa sin condición de finalización.

Sintaxis:

forever_statement ::= FOREVER statement

Parámetro	Significado
statement	Instrucción que se itera

Es posible romper la secuencia de iteración mediante las instrucciones BREAK, CONTINUE o RETURN.

Ejemplo:

```
//  Nos pregunta si queremos terminar hasta
//  que contestemos afirmativamente.
    forever if Ok("¿Terminar?") then break;
```

Instrucción REPEAT

Es una instrucción iterativa que comprueba su condición de finalización al final cada iteración.

Sintaxis:

repeat_statement ::= REPEAT statement UNTIL condition;

Parámetro	Significado
statement	Se ejecuta mientras condition no evalúe a FALSE
condition	Si evalúa a FALSE, se produce una nueva iteración de statement. En caso contrario se cede el control a la siguiente instrucción a repeat_statement

Esta instrucción termina siempre con el caracter ';'.

Si condition evalúa a NULL, es decir distinto de FALSE, se cede el control a la siguiente instrucción a repeat_statement.

Dado que condition se evalúa tras cada iteración, statement se ejecuta al menos una vez.

Es posible romper la secuencia de iteración mediante las instrucciones BREAK, CONTINUE o RETURN.

Ejemplo:

```
t = now + 10;
repeat
  ("Faltan"+now.SecondsTo(t)+"segundos").Trace;
until now > t;
```

Instrucción WHILE

Es una instrucción iterativa que comprueba su condición de finalización al principio de cada iteración.

Sintaxis:

while_statement ::= WHILE condition DO statement

Parámetro	Significado
condition	Si evalúa a TRUE., se produce una nueva iteración de statement. En caso contrario se cede el control a la siguiente instrucción a while_statement
statement	Se ejecuta mientras condition evalúe a TRUE

Si condition evalúa a NULL, es decir distinto de TRUE, se cede el control a la siguiente instrucción a while_statement.

Dado que condition se evalúa al inicio de cada iteración, statement puede no llegar a ejecutarse ninguna vez.

Es posible romper la secuencia de iteración mediante las instrucciones BREAK, CONTINUE o RETURN.

Ejemplo:

```
 i = 0;
// bucle que cuenta hasta 11
while i <= 10 do begin
    ++i;
end
i.Trace;
```

Instrucción BREAK

Rompe la secuencia de iteración de una instrucción iterativa y cede el control a la siguiente instrucción a ésta. Se descartan por tanto las instrucciones del bloque posteriores a ésta.

Sintaxis:

break_statement ::= BREAK;

Esta instrucción termina siempre con el caracter ';'. Sólo puede aparecer dentro de una instrucción iterativa.

Ejemplo:

```
forever begin // Bucle que cuenta hasta 11 con i
                // inicializado a 0
            ++i;
            i.Trace;
            if i>10 then
                      break;
        end
```

Instrucción CONTINUE

Rompe la secuencia de iteración de una instrucción iterativa y cede el control a la primera instrucción del bucle. Se descartan por tanto las instrucciones del bloque posteriores a ésta.

Sintaxis:

continue_statement ::= CONTINUE;

Esta instrucción termina siempre con el caracter ';'. Sólo puede aparecer dentro de una instrucción iterativa.

Provoca la evaluación de la condición de terminación asociada al bucle, y una nueva iteración de acuerdo con el resultado de la evaluación de ésta

Ejemplo:

```
while  --i > 0 do
 begin
      x=i*2;
      if  x==16 then
       begin
          ("x=" + x).Trace;
          continue;
       end
      y += x*x;
 end
```

Instrucción RETURN

La instrucción RETURN finaliza la ejecución de un método y permite devolver un valor al objeto que lo invocó. Se descartan por tanto las instrucciones posteriores a ésta.

Un método que no incluya esta instrucción finalizará su ejecución cuando se ejecute su última instrucción. Sin embargo, si un método devuelve un valor, la única forma de hacerlo es mediante esta instrucción.
Sintaxis :

return_statement ::= RETURN [expression];

Parámetro	Significado
expression	Ha de evaluar a una clase Simple

Esta instrucción termina siempre con el caracter ';'.

Si el prototipo del método especifica una clase de retorno, se ha de indicar una expresión obligatoriamente. Si el prototipo no especifica clase de retorno, es decir, la clase de retorno es la misma que la clase a la que pertenece el método y además se devuelve SELF, no se ha de indicar ninguna expresión.

Si el prototipo del método especifica una clase de retorno y la ejecución de éste no termina mediante una instrucción RETURN, se asume la instrucción de finalización RETURN NULL; .

Ejemplo:

```
public compara(i as smallint, c as smallint) return boolean
objects begin
    b as boolean
end
begin
    i = 3;
    b= i < c;
    if b==TRUE then
        ("i=" + i + " es menor que c =" +c).Trace;
    else
        ("i=" + i + " es mayor que c =" +c).Trace;
    return b;
end
```

5. Definición de Constantes

Una constante es un sinónimo de un literal, es decir, es un literal con un nombre asociado. Tiene por tanto una clase asociada y un valor que no puede ser modificado.

Sintaxis:

constant_section ::= [CONSTANTS BEGIN [constant_definition...] END]
constant_definition ::= identifier = literal

Parámetro	Significado
Identifier	Identificador único en su ámbito

Podemos ver una constante como un nombre alternativo a un literal. El compilador reemplaza cada identificador de constante por el valor de ésta.

Ejemplo:

Definimos una constante de nombre PI y valor 3.14159. La clase de la constante PI es la clase Decimal.

PI.Trace; es equivalente a 3.14159.Trace;

6. Clases y objetos

Definición de clases

Una clase es la modelización de una abstracción de entidad, caracterizada por un identificador único y una Interfaz.

Durante las fases de análisis, diseño y programación, los objetos con iguales atributos y comportamiento se agrupan en clases.

Durante la ejecución de un programa, las clases se materializan en el código que permite la ejecución del comportamiento dinámico de los objetos pertenecientes a la misma. Todos los objetos de clase comparten una interfaz, que caracteriza completamente a la clase desde el punto de vista de los restantes objetos del sistema.

Sintaxis:

class_section ::= [CLASSES BEGIN [class_definition...] END]
class_definition::=[access]identifier1IS [ABSTRACT] identifier2 [class_interface]
access ::= PUBLIC | PROTECTED | PRIVATE

Parámetro	Significado	
access	Indica la visibilidad de la clase fuera del módulo en el que se ha declarado. Puede ser de tres tipos	
	PUBLIC	Una clase declarada como pública siempre es visible fuera del módulo
	PROTECTED	Una clase declarada como protegida siempre es visible fuera del módulo, pero no será derivable fuera del módulo
	PRIVATE	Una clase privada solamente puede ser utilizada dentro del módulo en el que se ha declarado
identifier1	Identificador único en su ámbito. Es el nombre de la nueva clase	
identifier2	Nombre de la clase padre de la que estamos definiendo. Si se especifica ABSTRACT, la clase que estamos definiendo se considera abstracta o incompleta. En este caso, la clase identifier2 ha de ser también abstracta y no se debe especificar class_interface	
class_interface	Completa la interfaz de la nueva clase, si no se ha especificado ABSTRACT	

Una clase abstracta solo puede descender de otra clase abstracta.

Si la clase padre identifier2 es una de las clases Simples predefinidas, se puede especificar en la definición un valor por defecto.

Si la clase padre identifier2 es una de las clases Container predefinidas, se puede especificar en la definición su tamaño y la clase de los elementos contenidos.

Si la clase padre identifier2 es la clase Struct predefinida, se puede especificar una definición análoga a una tabla del repositorio donde cada campo de la tabla es un atributo de la definición, o bien, una lista de atributos.

Si la clase padre identifier2 es la clase Form predefinida, se puede especificar en la definición atributos adicionales como tablas, menús y una interfaz gráfica.

Si la clase padre identifier2 es la clase Page predefinida, se puede especificar en la definición atributos adicionales y una interfaz gráfica.

Si la clase padre identifier2 es la clase Menu predefinida, se puede especificar en la definición una interfaz gráfica adicional.

Definición de Objetos

En esta sección se definen los objetos que son atributos de un módulo.

Sintaxis:

```
object_section ::= [OBJECTS BEGIN [object_definition...] END]
object_definition ::= [access] identifier... AS class [DEFAULT literal]
access ::= PUBLIC | PROTECTED | PRIVATE
```

Parámetro	Significado	
access	Nos limita el acceso a la interfaz del objeto. Puede ser de tres tipos	
	PUBLIC	La interfaz del objeto no está restringida. Si el módulo es de tipo include, el objeto es visible en cualquier otro módulo que dependa de éste
	PROTECTED	La interfaz del objeto está restringida al propio módulo y a todos los métodos de las clases que se definan en el módulo. El objeto no es visible en ningún otro módulo
	PRIVATE	La interfaz del objeto está restringida al propio módulo. El objeto no es visible en ningún otro módulo
identifier	Ha de ser único en la sección de objetos	
Class	Puede ser cualquier clase no abstracta. Si es una de las clases simples Smallint, Integer, Decimal, Char, Time, Date o Boolean, se puede especificar literal como valor por defecto mediante la cláusula DEFAULT. Si la clase es Char, se puede especificar una longitud. Si la clase es Decimal o Money, se puede especificar una longitud y una precisión	

Si no se especifica access, el acceso por defecto es público.

Ámbito y visibilidad de un objeto

El ámbito de un objeto es la sección donde se ha definido éste. A la hora de referirnos desde nuestro código a un determinado objeto, hemos de tener en cuenta su ámbito, que es el que determinará si dicho identificador está o no accesible.

Distinguimos tres tipos de ámbitos local (o de método), de clase, global (o de módulo):

Ámbito Local o de método El objeto se define en el prototipo o en la seccion de objetos de un método.

Ámbito de Clase o de atributo El objeto se ha definido en la interfaz de la clase. Solo las clases Struct, Form, Page y Menu pueden definir este tipo de objetos.

Ámbito Global o de módulo El objeto se ha definido en la sección de objetos de un módulo.

La visibilidad de un objeto determina el conjunto de métodos donde puede ser utilizado. Un objeto es visible solo en una porción del programa determinado por su ámbito que puede estar limitado al método, clase o módulo en el que está definido.

Dependiendo del ámbito de un objeto, un objeto es visible :

Local El objeto solo es visible en el método donde está definido.

De Clase El objeto es visible en los métodos de su clase, en los métodos de las clases descendientes si su acceso no es privado y en cualquier otro método si su acceso es público.

Global El objeto es visible en los métodos del módulo, en los métodos de las clases que define el módulo si su acceso no es privado y en cualquier otro método de los módulos que incluyan a éste como include si su acceso es público.

El compilador al analizar el código de un método busca un identificador en este orden.

- Como parámetro u objeto local al método.
- Si no se ha encontrado el identificador se busca en la interfaz de la clase a la que pertenece el método, para ver si se ha definido como atributo o identificador de método.
- Si no se ha encontrado el identificador se busca en la interfaz de la clase padre a la que pertenece el método, para ver si se ha definido como atributo o identificador de método.
- Si aún no se ha encontrado el identificador se busca en la interfaz del módulo (incluyendo la sección de constantes) y en los módulos de los que depende éste (Includes y librerias).

Sin embargo, este orden se puede modificar mediante los cualificadores de ámbito Parent,Module y Self.

- Un nombre precedido de “Self.” Limita la búsqueda a los puntos segundo y tercero.
- Un nombre precedido de “Parent.” Limita la búsqueda al punto tercero.
- Un nombre precedido de “Module.” Limita la búsqueda al punto cuatro.

El Objeto Self

Todos los objetos de una clase comparten el mismo código. Si en el método de una clase tiene que acceder a objeto sobre el que se está aplicando ese código, debe referirse a él mediante la palabra reservada Self.

Ejemplo:

Si declaramos un objeto miEntero de la clase miClaseEntera (derivada de smallint) y seguidamente le mandamos el mensaje Doble que es un método definido en la clase miClaseEntera.

miEntero.Doble;

El método “Doble” puede estar definido como se indica a continuación:

```
public function Doble() return miClaseEntera
begin
        return Self * 2;
end
```

7. Clases predefinidas

Conversiones entre clases predefinidas

Al aplicar un método conversor sobre un objeto de una clase, da como resultado otro objeto de una clase diferente a la primera. A este proceso se le denomina conversión. El propósito de una conversión es obtener un objeto de una clase diferente con un valor considerado como equivalente.

Si un clase tiene definido un conversor a una clase C, se dice que los objetos de esta clase son convertibles a C. Así un objeto Char es convertible a Integer porque tiene definido un conversor a esta clase.

En la llamada a un método, el compilador provocará un error semántico si se intenta pasar como argumento de la llamada un objeto de distinta clase a la definida en el parámetro y además no se ha definido un conversor para ello.

Una clase siempre es convertible a su clase padre por definición.

A continuación se muestra una tabla de las conversiones para las clases predefinidas.

Desde \ Hasta	char	Numeric	Time	Date	Boolean	Decimal	Money	Integer	Smallint
Char	-	Si	Si	Si	Si	Si	Si	Si	Si
Numeric	Si	-	No	No	Si	Si	-	Si	Si
Time	Si	No	-	No	No	No	No	No	No
Date	Si	No	No	-	No	No	No	No	No
Boolean	No	No	No	No	-	No	No	No	No
Decimal	Si	Si	No	No	No	-	Si	Si *	Si *
Money	Si	Si	No	No	No	Si	-	Si *	Si *
Integer	Si	Si	No	No	No	Si	Si	-	Si *
Smallint	Si	Si	No	No	No	Si	Si	Si	-

La conversión será válida cuando los datos a convertir cumplan las reglas de rango especificadas para cada tipo en particular.

El * indica que la conversión es válida si el valor esta dentro del rango de valores que puede admitir el nuevo tipo de dato. Por ejemplo, un objeto i de tipo Integer se podrá convertir a Smallint si -32767 <= i >= 32767. Si el valor de i no este dentro del rango indicado (ver tabla) no dará error en compilación pero se dará un error de desbordamiento (overflow) en ejecución.

El siguiente cuadro recoge de forma resumida los valores de los diferentes tipos de datos:

Tipo de dato	Valor mínimo	Valor máximo	Ocupación(bytes)
CHAR(n)	0 carácteres	32.767 caracteres	n
SMALLINT	-32.767	+32.767	2
INTEGER	-2.147.483.647	+2.147.483.647	4
TIME	00:00:01	24:00:00	4
DECIMAL (m,n)	10^{-130}	10^{125}	1+m/2
DATE	01/01/0001	31/12/9999	4
MONEY(m,n)	10^{-130}	10^{125}	1+m/2

8. Definición de métodos

Cuando un objeto necesita utilizar una operación especificada en la interfaz de otro objeto, se lo hace saber mediante un mensaje que consta de las siguientes partes:

Emisor	Es el objeto que envía el mensaje.
Receptor	Es el objeto que recibe el mensaje.
Selector	Es el identificador de la operación que el objeto emisor solicita al objeto receptor.
Lista de Argumentos	Es la información adicional que necesita el objeto receptor para completar la operación solicitada.
Valor de Retorno	Es un valor que devuelve el objeto receptor al objeto emisor. Es el resultado de la operación solicitada.

Llamamos método a la implementación de una operación especificada en la interfaz de una clase. Si la interfaz de la clase padre ya especifica dicha operación, entonces decimos que la clase hija redefine el método que implementa la operación.

Sintaxis:

method_definition ::= prototype object_section method_body

Parámetro	Significado
Prototipo	Es la interfaz del método. Describe las características del mensaje asociado a la operación que implementa el método. Contiene toda la información necesaria para invocarlo correctamente

Prototype

Sintaxis:

prototype ::= [access] method_identifier [([parameter,...])] [RETURN class1]
access ::= PUBLIC | PROTECTED | PRIVATE
parameter ::= [VAR] identifier... AS class2 [DEFAULT literal]

Parámetro	Significado	
access	Nos limita "quíen" puede invocar el método, es decir, restringe la clase del objeto emisor del mensaje asociado a la operación que implementa el método. Hay tres tipos de acceso:	
	PUBLIC	la clase del objeto emisor no está restringida
	PROTECTED	la clase del objeto emisor ha de ser la misma que la clase del objeto receptor, o descender de ésta
	PRIVATE	la clase del objeto emisor ha de ser la misma que la clase del objeto receptor
method_identifier	Tipo de método junto con un identificar adicional si es necesario. Especifica la operación asociada al método. Ha de ser único en el interfaz de la clase en la que se define el método. Los métodos se clasifican en: Función, Dll, Operador, Conversor, Notificación, Comando, Main.	
Class1	Es la clase del valor de retorno. Si se especifica, ha de ser de clase Simple y además el valor de retorno será un objeto temporal. En caso contrario el valor de retorno es el mismo objeto sobre el cual se ha invocado al método (SELF)	

- *Parameter*

Un parámetro es un objeto que describe los requisitos que ha de cumplir un argumento en la invocación de un método. Su identificador identifier ha de ser único en la lista de parámetros.

Es posible especificar un literal como valor por defecto para el parámetro mediante la cláusula DEFAULT. De esta manera el compilador insertará literal en la invocación del método, si no se especifica un argumento. La cláusula DEFAULT solo se puede especificar si class2 es una de las siguientes clases simples: Smallint, Integer, Decimal, Char, Time, Date o Boolean. Una vez que se especifica que un parámetro tiene un valor por defecto, se ha de especificar en los parámetros siguientes.

Es posible especificar que se desea recibir una lista variable de argumentos especificando la clase ArgList para el último parámetro. Por tanto, podemos subdvidir en tres partes la lista de parámetros:

- lista de parámetros sin valor por defecto
- lista de parámetros con valor por defecto
- parámetro de clase Arglist.

Cada una de estas partes es opcional, pero de aparecer, lo han de hacer en en el orden anterior.

Si no se especifica VAR, el runtime realiza una copia del argumento antes de la invocación del método y la destruye después. En caso contrario, el estado resultante de las operaciones realizadas sobre el argumento durante la invocación del método es permanente. Si la clase class2 del parámetro no es Simple, se considera que se ha especificado VAR. class2 puede ser cualquier clase, incluso una clase abstracta.

Object Section (Sección de objetos locales)

Define objetos auxiliares. Estos objetos se crean al inicio de la invocacíon del método, y se destruyen en la finalización de éste.

Sintaxis:

object_section ::= [OBJECTS BEGIN [object_definition...] END]
object_definition ::= identifier... AS class [DEFAULT literal]

Parámetro	Significado
identifier	Ha de ser único tanto en la sección de objetos locales, como en la lista de parámetros
class	Puede ser cualquier clase no abstracta. Si es una de las clases simples Smallint, Integer, Decimal, Char, Time, Date o Boolean, se puede especificar literal como valor por defecto mediante la cláusula DEFAULT. Si la clase es Char, se puede especificar una longitud. Si la clase es Decimal o Money, se puede especificar una longitud y una precisión

Method Body (Cuerpo del método)

Consta de una secuencia de instrucciones orientadas a realizar una tarea específica. La invocación del método finaliza cuando se ejecuta la última instrucción de la secuencia o cuando se ejecuta la instrucción RETURN. En este caso, es obligatorio especificar un valor de retorno, si el prototipo del método lo exige.

Sintaxis:

method_body ::= [compound_statement]

Si una clase redefine un método, el prototipo de éste ha de ser idéntico al del método que se está redefiniendo, salvo la clase de retorno, que puede ser descendiente de la clase de retorno del método original.

Un paso de mensaje, que implica la invocación de un método, se implementa mediante una expresión. Esta expresión es de la forma:

message ::= [expression .] operation [([argument,...])]
argument ::= expression

Parámetro	Significado
expression	Es el receptor del mensaje. La evaluación de esta expresión determina el objeto receptor del mensaje. Su especificación es opcional, pues dicho objeto puede estar especificado implícitamente. Consulte el apartado ámbito y visibilidad de un objeto
operation	Es el selector del mensaje. Es la única parte de éste cuya especificación es obligatoria
argument,...	Es la lista de argumentos del mensaje. Cada argumento se valida con el parámetro que ocupa la misma posición en el prototipo. Para que esta validación sea correcta, se ha de cumplir una y solo una de estas condiciones: 1. El argumento ha de ser el literal NULL 2. La clase del argumento ha de ser la misma que la clase del parámetro, o descender de ésta 3. La interfaz de la clase del argumento ha de especificar un método conversor hacia la clase del parámetro 4. Si la clase del parámetro es igual o descendiente de la clase ArgList, el argumento y todos los que le siguen, han de cumplir una de las tres condiciones anteriores con respecto a la clase asociada a la clase ArgList

Dado que operation puede ser polimórfica, pues puede haber más de una clase cuya interfaz especifique esta operación, se determina sobre la clase asociada a expression.

Tanto expression como argument, se evalúan antes de la ejecución de la primera instrucción del método asociado a operation.

Todo mensaje aparece en un método, ya que éste es la implementación de una operación. El objeto que emite este mensaje se le llama “objeto emisor”. Nos podemos referir a él con la palabra reservada SELF.

Ejemplo:

```
MAIN BEGIN "hello".Trace; END
```

Estudiemos el mensaje “hello”.Trace:

Emisor: el objeto de clase Module al que pertenece el método main (SELF).
Receptor: el literal “hello” de clase Char.
Selector: la operación Trace de la clase Char.
Lista de argumentos: vacía.
Valor de retorno: objeto de la clase Char, que no se utiliza.

```
MAIN BEGIN Beep; END
```
que es equivalente a
```
MAIN BEGIN SELF.Beep; END
```
En este caso el objeto receptor y el objeto emisor son el mismo, es decir, SELF.

```
MAIN BEGIN "hello world".Count('l').Trace; END
```
Tenemos dos mensajes anidados, pues aparecen dos operaciones, Count y Trace.
Emisor del primer mensaje: el objeto de la clase Module al que pertenece el método main (SELF).
Receptor del primer mensaje: el literal “hello world” de la clase Char.
Selector del primer mensaje: la operación Count de la clase Char.
Lista de argumentos del primer mensaje: (‘l’). Hay un único argumento que es un literal de la clase Char.
Valor de retorno del primer mensaje: objeto de la clase Smallint que actúa como receptor del segundo mensaje.

Emisor del segundo mensaje: el objeto de la clase Module al que pertenece el método main (SELF).
Receptor del segundo mensaje: el objeto de la clase Smallint devuelto por el primer mensaje..
Selector del segundo mensaje: la operación Trace de la clase Smallint.
Lista de argumentos del segundo mensaje: vacía.
Valor de retorno del segundo mensaje: objeto de la clase Smallint , que no se utiliza.

Paso de parámetros por referencia y por valor

Un argumento se pasa por referencia si en la definición del parámetro del método se especifica la cláusula VAR. También se pasan por referencia todos los objetos que su clase no desciende de la clase Simple, aunque no se especifique la cláusula VAR.

Si un objeto se pasa por valor, el runtime realiza una copia del argumento antes de la invocación del método y la destruye después. En caso contrario (paso por referencia), el estado resultante de las operaciones realizadas sobre el argumento durante la invocación del método es permanente, es decir las operaciones realizadas sobre el parámetro dentro del método se realizarán sobre el objeto original pasado como argumento.

Los siguientes puntos deben tenerse en cuenta cuando se va a decidir como pasar los parámetros:

- Defina un parámetro por valor (sin cláusula VAR) cuando quiera que el estado del objeto pasado como argumento permanezca inalterado durante la invocación del método.
- Defina un parámetro por referencia (cláusula VAR) cuando quiera que el método pueda modificar el estado del objeto pasado como argumento.
- Defina los parámetros de tipo Char por referencia (aunque no se desee modificar el valor del mismo), si desea optimizar el tiempo y gasto de memoria consumidos en la llamada, siempre que en el código del método no se altere el valor del mismo.

Definición de parámetros por valor por defecto

Un parámetro de un método puede definirse con un valor por defecto. El compilador permite la invocación a un método sin pasar valores en los parámetros definidos con valor por defecto. En este caso el compilador inserta en la llamada todos los argumentos no especificados usando para ello el valor por defecto definido para cada uno.

Ejemplo:

Para el método MessageBox de la clase Module cuyo prototipo es :

```
MessageBox(msg as char,
           title as char default NULL,
           flags as smallint default 1) return smallint
```

Permite que :

MessageBox(“Hello World”) sea equivalente a MessageBox(“Hello World”, NULL, 1)

Si se declaran parámetros por defecto en la definición de un método, estos deben indicarse consecutivamente sin interrupciones hasta el final de la lista de argumentos.

Redefinición de métodos

Si una clase define un método que ya estaba definido en la interfaz de su *clase padre*, decimos que esta clase redefine este método.

El prototipo ha de ser igual al del método que se redefine, exceptuando la clase del valor de retorno que puede ser una clase descendiente de la clase de retorno del método redefinido.

Si en el código de esta clase se desea invocar al método de la clase padre, es necesario especificar el cualificador Parent.

En una clase se permite redefinir los operadores aritméticos y el operador de asignación de su clase base.

Ejemplo:

```
CLASSES  begin
      sm is Smallint
          end
CODE CLASS sm BEGIN
//{{CODEBEGIN
public Trace() // Redefinición del método Trace
begin
     ("Estoy llamando al metodo Trace de la clase sm"+ self).Trace;
     "A continuación se llama al metodo Trace de la clase
     Numeric".Trace;
    parent.Trace;
end
//{{CODEEND
END
```

Clasificación de los métodos

Sintaxis:

method ::= function | dll | operator | conversor | notification | command | main

Como se puede observar en la sintaxis un método puede ser de tipo:

Función	Un método función es un método que se referencia por un identificador.
Dll	Un método dll es un método que se implementa mediante una llamada un procedimiento de una librería de enlace dinámico (DLL).
Operador	Un método operador es un método que se referencia por un símbolo matemático o bien por el identificador de éste.
Conversor	Un método conversor es un método que se referencia por el identificador de una clase. Su finalidad consiste en devolver un objeto de dicha clase, equivalente a la clase sobre la que se invoca el método.
Notificación	Un método notificación es la implementación de una respuesta a un mensaje provocado por un suceso o evento debido a una interacción con la interfaz gráfica o con la interfaz SQL. Los sucesos o eventos que pueden enviar una notificación están predeterminados.
Comando	Un comando se define como una acción genérica de una aplicación.
Main	El método MAIN es un caso especial de método. Sólo se puede definir en la sección de código de un módulo. Sirve de punto de entrada para su ejecución.

9. Función

Un método función es un método que se referencia por un identificador.

Sintaxis de definición:

function ::= function_prototype object_section function _body
function_prototype ::= access function_identifier[([parameter,...])]
[RETURN class1]
function_identifier ::= [FUNCTION] identifier
function _body ::= compound_statement

Parámetros	Significado
Identifier	Es el nombre de la operación que implementa el método. Este nombre ha de ser único, es decir no debe coincidir con el nombre de otra operación definida en la misma clase ni con el identificador de algún atributo de ésta

Sintaxis de invocación:

function_message ::= [expression .] identifier [([argument,...])]
argument ::= expression

Parámetros	Significado
Identifier	Identificador de la función que queremos invocar

10. DLL

Un método dll es un método que se implementa mediante una llamada un procedimiento de una librería de enlace dinámico (DLL).

Las librerías de enlace dinámco (DLL) son una característica clave del sistema operativo Windows. Como su nombre indica, las DLLs son bibliotecas de funciones que las aplicaciones pueden vincular y usar en tiempo de ejecución en lugar de hacerlo estáticamente en tiempo de compilación. Esto significa que las bibliotecas pueden actualizarse independientemente de la aplicación y que muchas aplicaciones pueden compartir una misma DLL.

Para poder llamar en nuestro programa a una función de una DLL debemos indicar de alguna forma al compilador en que DLL se encuentra el método y que parámetros recibe. Es decir debemos declarar el método para que el compilador sea capaz de encontrarla.

Sintaxis de definición:

dll ::= function_prototype object_section dll_body
dll_prototype ::= access dll_identifier [([parameter,...])] [RETURN class1]
dll_identifier ::= [SELF] DLL dll_path identifier
dll_body ::=

Parámetros	Significado
identifier	Es el nombre de la operación que implementa el método. Este nombre ha de ser único, es decir no debe coincidir con el nombre de otra operación definida en la misma clase ni con el identificador de algún atributo de ésta. Este identificador ha de coincidir además con el nombre del procedimiento exportado por la librería dinámica
dll_path	Literal de clase Char. Es el path en el que se busca la dll, en la primera invocación al método. No es necesario indicar el path completo de la Dll, basta con indicar su nombre siempre que esta se vaya a ser accesible para Windows. Sólo se comprueba en tiempo de ejecución y no en tiempo de compilación

La clase de los parámetros ha de ser Simple. No obstante, no se da ningún error en caso contrario.

Sintaxis de invocación:

dll_message ::= function_message

La sintaxis de invocación de un método dll es idéntica a la de un método función. Podemos ver un método dll como un método función con implementación externa.

Si se especifica SELF en la definición del método, el objeto sobre el que se invoca se pasará como primer parámetro.

El paso de argumentos al procedimiento de la Dll se realiza mediante el convenio Pascal. El paso de argumentos se realiza de la siguiente manera:

Clase	Por Valor	Por referencia
Smallint	entero de 16 bits	puntero de 32 bits
Integer	entero de 32 bits	puntero de 32 bits
Decimal	no implementado	no implementado
Char	puntero de 32 bits	puntero de 32 bits
Date	entero de 32 bits	puntero de 32 bits
Time	entero de 32 bits	puntero de 32 bits
Enum	entero de 32 bits	puntero de 32 bits
EnumSet	entero de 32 bits	puntero de 32 bits
Complex	no implementado	no implementado
Container	no implementado	no implementado

Si un argumento se pasa por valor y su estado es desconocido (IS NULL), el valor que se pasa es indeterminado.

Si un argumento con valor NULL se pasa por referencia, tras la invocación del método, se asume que su valor ha sido modificado, por lo que a partir de este momento, sea cual sea su valor se tomará como valor real (NOT NULL).

Si el argumento es el literal NULL, se pasa 0.

El valor de retorno se realiza de manera análoga al paso de argumentos por valor. Si la clase del valor de retorno es la clase Char y el valor de retorno es 0, se considera el valor de retorno como desconocido (IS NULL). En caso contrario el estado del valor de retorno no es desconocido (IS NOT NULL).

La librería dinámica se carga la primera vez que se invoca un procedimiento de ella, y se descarga cuando finaliza la ejecución de la aplicación.

11. Operador

Un método operador es un método que se referencia por un símbolo matemático o bien por el identificador de éste.

Sintaxis de definición:

operator ::= operator_prototype object_section operator_body
operator_prototype::=access operator_identifier[([parameter])][RETURN class1]
operator_identifier ::= OPERATOR op
operator_body ::= compound_statement

Parámetros	Significado
op	Es el nombre de la operación que implementa el método. Este nombre ha de ser único, es decir no debe coincidir con el nombre de otra operación definida en la misma clase ni con el identificador de algún atributo de ésta

Sólo es posible redefinir los operadores aritméticos (*, *=, /, /=, %, %=, +, +=, ++, -, -=, --, =).

Operador unario

operator_message::=op expression | expression op|[expression .]
OPERATOR op [()]

Ejemplo:
++x es equivalente x.OPERATOR ++ que es equivalente a x. OPERATOR ++();
x IS NULL es equivalente a x.OPERATOR IS NULL que es equivalente a x.OPERATOR IS NULL ();

Operador binario

operator_message::=[expression] op argument | [expression .]
OPERATOR op (argument)
argument ::= expression

Ejemplo:
x+5 es equivalente a x.OPERATOR +(5);
x LIKE “%.smd” es equivalente a x.OPERATOR LIKE (“%.smd”)

Operador binario con parámetro Arglist

operator_message::=[expression] op (argument,...) | [expression .]
OPERATOR op (argument,...)
argument ::= expression

Ejemplo:
x IN (1,2,3,4) es equivalente a x.OPERATOR IN(1,2,3,4)

Operador ternario

operator_message::=[expression] op argument1 op1 argument2 | [expression .]
OPERATOR op (argument1, argument2)
argument ::= expression
Ejemplo: x BETWEEN 5 AND 10 es equivalente a x.OPERATOR BETWEEN(5,10)

12. Conversor

Un método conversor es un método que se referencia por el identificador de una clase. Su finalidad consiste en devolver un objeto de dicha clase, equivalente a la clase sobre la que se invoca el método.

Sintaxis de definición:

conversor ::= conversor_prototype object_section conversor_body
conversor_prototype ::= access conversor_identifier
conversor_identifier ::= CONVERSOR identifier
conversor_body ::= compound_statement

Parámetros	Significado
identifier	Es la clase del valor de retorno. Siempre debe ser un identificador de una clase Simple. Dicha clase no puede ser una clase padre de la misma que lo implementa, aunque si una clase hija

Los métodos de este tipo no aceptan parámetros pues la conversión ha de ser única.

Este método devuelve un valor de la clase indicada, que se supone equivalente al objeto sobre el que se ha invocado el método conversor. Nunca modifica el objeto sobre el que se invoca.

Podemos ver un método conversor como un método función sin parámetros y que devuelve un valor de la clase indicada. También podemos ver un método conversor como un método operador sin parámetros y cuyo operador es la clase.

En el proceso de validación de un argumento respecto a un parámetro, el compilador puede insertar una invocación a un conversor sobre el argumento con el fin de validar el parámetro.

Sintaxis de invocación:

conversor_message ::= [expression.] [CONVERSOR] identifier [()]

Parámetros	Significado
identifier	Es el identificador de una clase Simple. Dicha clase no puede ser una clase padre, aunque si una clase hija

Ejemplo:

5 < “6” debería provocar un error pues el método operador “<” de Numeric solo acepta un parámetro de clase Numeric mientras que “6” es de la clase Char. En este caso el compilador inserta una invocación al método conversor a clase Numeric sobre “6”. Es decir:

5 < “6” es reemplazado por 5 < “6”.Numeric que es equivalente a 5 < “6”.Numeric() que es equivalente a 5 < “6”.CONVERSOR Numeric que es equivalente a 5 < “6”.CONVERSOR Numeric() que es equivalente a 5. OPERATOR <(“6”. CONVERSOR Numeric()).

```
i=1;
s = "número " + i.conversor Char;
s = "número " + i.Char;
s.Trace ; // mostrará en pantalla el literal : número 1
```

13. Main

El método MAIN es un caso especial de método. Sólo se puede definir en la sección de código de un módulo. Sirve de punto de entrada para su ejecución. Al ejecutar la última instrucción de este método, se terminará la ejecución de este módulo.

Sintaxis de definición:

main ::= main_prototype object_section main_body
main_prototype ::= main_identifier [([parameter,...])]
main_identifier ::= MAIN
main_body ::= compound_statement

No es posible invocar el método main explícitamente. Ha de invocarse indirectamente, bien a través del comando cosrun o bien a través del método Run de la clase Module.

El método MAIN siempre recibe sus argumentos por referencia, aunque no se especifique la palabra reservada VAR en su definición de parametros.

La finalización de la ejecución de un método MAIN implica una descarga del módulo en el que se ha definido si este no es referenciado por ninguno de los módulos de la aplicación que están en memoria en ese momento.

Si un módulo va a ser invocado mediante el comando cosrun, ha de tener un prototipo tal que no necesite argumentos para su invocación, o bien que solo necesite un argumento de tipo Char.

Ejemplo:
MAIN BEGIN END solo acepta una línea de comando sin la opción -arg
MAIN (c **AS** char **DEFAULT** ”hola mundo”) **BEGIN END** acepta una línea de comando con o sin la opción -arg
MAIN (c **AS** char,s **AS** smallint **DEFAULT** 7) **BEGIN END** solo acepta una línea de comando con la opción -arg
MAIN (s **AS** smallint) **BEGIN END** provocará un error.

14. Notificaciones

Un método notificación es la implementación de una respuesta a un mensaje provocado por un suceso o evento debido a una interacción con la interfaz gráfica o con la interfaz SQL. Los sucesos o eventos que pueden enviar una notificación están predeterminados.

Sintaxis:

notification ::= form_notification | control_notification | table_notification

Estos métodos no pueden ser invocados explícitamente.

Se puede observar en la sintaxis que las notificaciones pueden ser de tres tipos:

Notificaciones de control	Eventos mandados por un control de la screen, son procesados por su Form.
Notificaciones de Tabla	Eventos mandados por una tabla, que son procesados por su Form.
Notificaciones de Form	Eventos mandados y procesados por el Form.

Una notificación solo se puede definir dentro del código de una clase Form. Las notificaciones están ya predefinidas, el usuario no puede crear notificaciones. Es decir, solo puede definir las instrucciones a ejecutar cuando se manda una notificación de las que hay definidas en Cosmos.

La lista de las diferentes Notificaciones la puede encontrar en el Anexo Notificaciones.

Notificaciones de Control

Un método notificación de control es la respuesta a un mensaje que envía un control gráfico a la clase Form en la que se ha definido para informarle de un suceso o evento.

Sintaxis de definición:

```
control_notification::=control_notification_prototype
                       object_section
                       notification_body
control_notification_prototype ::= ON event control [(idm AS Integer)]
                                   | ON event (control AS Char[, idm AS Integer])
notification_body ::= compound_statement
```

Parámetros	Significado
Event	Identificador de una de las notificaciones que puede mandar un objeto de clase Control a su Form, tales como Click, RClick, Dblclick, SelChange, etc..
Control	Identificador del objeto de clase Control que envía la notificación al Form
Idm	Indice del elemento seleccionado en un control de tipo: Lista, grupo de cajas, grupo de botones, grupo de botones radio

Estos métodos no pueden ser invocados explícitamente. Solo pueden ser invocados como consecuencia de una interacción con un control de la interfaz gráfica.

Cuando se recibe una notificación de un control, se busca primero si se ha definido un método de prototipo “ON event control [(idm AS Integer)] “ específico para este caso. En caso contrario se buscará un método generico “ON event (control AS Char[, idm AS Integer]) “ que captura las notificaciones de tipo “event” para cualquer control, pasando el nombre del control como parámetro.

Si el control gráfico tiene asociado un comando, esta notificación se envía como comando.

Notificaciones de Form

Un método notificación de form es la respuesta a un mensaje que envía a sí mismo un objeto de clase Form informar de un suceso o evento.

Sintaxis de definición:

```
form_notification::= form_notification_prototype object_section notification_body
form_notification_prototype ::= notification_identifier
notification_body ::= compound_statement
notification_identifier ::= ON event
```

Parámetros	Significado
event	Identificador de una de las notificaciones que puede mandar un objeto de clase Form, tales como Open y Close

Estos métodos no pueden ser invocados explícitamente. Solo pueden ser invocados como consecuencia de la ejecución de determinados métodos (Open, Close, Run y Exit).

Notificaciones de Tabla

Una notificación de tabla es un mensaje que envía un objeto de la clase FormTable al objeto de la clase Form en la que se ha definido, para informarle de un suceso o evento.

Sintaxis de definición:

table_notification::=table_notification_prototype object_section notification_body
table_notification_prototype ::= ON event TABLE table
| ON event TABLE (table AS Char)
notification_body ::= compound_statement

Parámetros	Significado
event	Identificador de una de las notificaciones que puede mandar un objeto de clase FormTable a su Form definidas en Cosmos, puede ser una de las siguientes: Add, New, Update, etc..
tabla	Objeto de la clase FormTable afectado por el evento
identifier	Identificador del objeto de la clase FormTable afectado por el evento

15. Comandos

Un comando se define como una acción genérica de una aplicación. Esto permite asociar la misma respuesta por ejemplo al pulsar un botón, opción de menú o una secuencia de teclas (acelerador).

Si el control gráfico que envía el comando esta definido en el área gráfica asociada a una tabla del Form, Este comando se enviará en primer lugar a la tabla para ver si se procesa, y en caso contrario se enviará al Form.

El hecho de que tanto la clase Form y la Clase FormTable tenga preimplementados métodos de respuesta a comandos predefinidos, permite un comportamiento automático de los forms y tablas sin más que definir controles gráficos que envíen estos comandos

Además de los comandos predefinidos Cosmos, el programador puede definir todos aquellos que necesite en cada aplicación.

Sintaxis:

command ::= form_command | table_command

Como se puede observar en la sintaxis los comandos pueden ser de dos tipos:

Comandos de Tabla	Código de respuesta al realizar una determinada acción sobre una tabla de un Form (añadir, modificar, borrar,etc.). Estos comandos son procesados por la tabla activa del Form.
Comandos de Form	Código de respuesta al realizar una determinada acción sobre el Form y/o su tabla maestra.

Comandos de Form

Sintaxis:

```
form_command ::= form_command_prototype object_section command_body
form_command_prototype ::= ON COMMAND cmd
                                    | ON COMMAND (cmd AS Char)
command_body ::= compound_statement
```

Parámetros	Significado
Cmd	Es un identificador de comando. Puede aparecer explícito, o como el valor de un objeto de clase Char

Comandos de Tabla

Sintaxis:

```
table_command ::= table_command_prototype object_section command_body
table_command_prototype ::= ON COMMAND cmd TABLE table
                          | ON COMMAND cmd TABLE (table AS Char)
                          | ON COMMAND table TABLE (cmd AS Char)
                          | ON COMMAND TABLE(tableASChar,cmd AS Char)
command_body ::= compound_statement
```

Parámetros	Significado
cmd	Es un identificador de comando. Puede aparecer explícito, o como el valor de un objeto de clase Char
table	Es un identificador de tabla receptora del comando. Puede aparecer explícito, o como el valor de un objeto de clase Char

Estos métodos no pueden ser invocados explícitamente.

16. Propiedades

Una propiedad es un atributo de un objeto tal que su estado está asociado al estado de dicho objeto. Así se puede decir que el estado de un objeto puede ser manipulado a través de sus propiedades.

El concepto de propiedad se suele asociar a objetos gráficos (controles, menús ...). Las clases gráficas definen en su interfaz además de una lista de métodos, una lista de propiedades.

El uso de algunas de estas propiedades está limitado al tiempo de diseño. El resto de propiedades pueden ser usadas en tiempo de ejecución. El Editor Visual de Cosmos permite definir las propiedades de cada control mostrando para ello el diálogo de propiedades que corresponda a ese control.

Cada propiedad como cualquier otro atributo tiene una clase asociada que indica las operaciones que pueden realizarse con ella. Una propiedad se maneja de igual forma que cualquier otro atributo del objeto.

Una propiedad no puede pasarse por referencia como argumento de una llamada a un método.

Capítulo 6
CTSQL

Contenidos

1. Introducción

Cosmos consta de un servidor de base de datos relacional, denominado CTSQL.

La idea de modelo relacional se basa en el concepto de *relación*. Una relación no es más que un conjunto de objetos de la misma clase. Un grupo de objetos puede ser clasificado en varios tipos de objetos o conjuntos. Cada objeto tendrá varias propiedades (atributos) características de su forma de uso, las cuales definen agrupamientos en conjuntos. Cada objeto de un conjunto puede describirse por los valores de sus atributos. Esta descripción se llama *tupla*. Si se consideran todas las tuplas que describen todos los objetos del conjunto, se habla de una relación.

En bases de datos relacionales, la representación de una relación es una tabla (fichero). Esencialmente, una tabla está compuesta de filas (registros) y columnas (campos), que definen los datos característicos de los valores almacenados.

Una base de datos relacional se describe fundamentalmente por dos características:

- Los datos se ven como tablas y no existen apuntadores entre ellos.
- Las operaciones sobre los datos reciben como entrada tablas y producen como salida tablas.

El SQL tiene su origen en el prototipo de base de datos relacional *SYSTEM R*, desarrollado por IBM en la década de los 70. Se trata de la evolución de los lenguajes denominados *SQUARE* y *SEQUEL*. Debido a su gran aceptación se decidió su normalización, primero por el Instituto Americano de Estandarización (ANSI) y posteriormente por la Organización Internacional de Estandarización (ISO). El estándar SQL tiene como finalidad principal la portabilidad de las definiciones de bases de datos y los programas de aplicación.

2. Expresiones, Variables y Funciones CTSQL

▪ *Expresiones*

Una expresión es cualquier dato, constante o variable, o la combinación de varios de ellos mediante operadores, que tras un proceso de evaluación produce un resultado.

Los operadores aritméticos y de formato que pueden emplearse en expresiones de tipo CTSQL son los siguientes:

Operador	Efecto
+	Suma
-	Resta
*	Multiplicación
/	División
||	Concatenación

Las expresiones válidas en CTSQL son las siguientes:

Expresion	Indica
Número	Número expresado como una constante en instrucciones CTSQL
Literal	Cadena de texto entre comillas. Este literal se expresa como constante en instrucciones CTSQL y puede representar una fecha, un tipo de dato hora o una cadena de caracteres alfanuméricos. Los tipos de datos correspondientes a fecha y hora se representan respectivamente como *dd/mm/yyyy* y *hh:mm:ss*
-expresión	En expresiones numéricas invierte el signo
(expresión)	Evalúa en primer lugar la expresión entre paréntesis (precedencia explícita)
Expresión operador expresión	Cualquier combinación de expresiones mediante los operadores aritméticos y de formato indicados anteriormente
Función(expresiones)	Cualquiera de los valores agregados o funciones admitidas por CTSQL
Subselect	Tabla derivada devuelta por una instrucción SELECT que servirá como condicionante de una expresión
ROWID	El rowid es el número de orden que ocupa físicamente una fila en una tabla. Mediante este valor se produce el acceso más rápido a los datos de la tabla, por lo que su uso es prioritario

En la mayoría de los casos, los operandos utilizados en la formación de estas expresiones serán columnas de las tablas de la base de datos.

▪ *Variables internas CTSQL*

Al igual que ocurre en el lenguaje de cuarta generación de Cosmos (COOL), el CTSQL dispone de determinadas variables internas cuyo mantenimiento es automático. Estas variables son las siguientes:

Variable	Efecto
today	Devuelve la fecha del sistema
now	Devuelve la hora del sistema
user	Evalúa el nombre del usuario con el que se accede a la base de datos. Este nombre será el definido en la variable de entorno DBUSER

▪ *Funciones del CTSQL*

CTSQL puede utilizar funciones de valores agregados, funciones de fecha, hora y funciones del sistema. Estas funciones son las siguientes:

Agregados	Fecha	Hora	Sistema
count()	**date()**	**Time()**	**typelength()**
sum()	**day()**	**Hour()**	
avg()	**mdy()**	**Minute()**	
max()	**month()**	**Second()**	
min()	**weekday()**	**Hms()**	
	year()	**Mt()**	
		Tomt()	

Valores Agregados

count(*) Devuelve el número de filas que cumplen la cláusula WHERE.

count(distinct x**)** Devuelve el número de valores únicos en la columna x en las filas que cumplen la cláusula WHERE.

sum([distinct] x**)** Devuelve la suma de todos los valores de la columna x para las filas que cumplen la cláusula WHERE. Esta función sólo podrá utilizarse con valores numéricos. En caso de utilizarse la palabra clave distinct, el resultado será la suma de los valores distintos de la columna x.

avg([distinct] x**)** Devuelve el promedio de todos los valores de la columna x para las filas que cumplen la cláusula WHERE. Esta función sólo podrá utilizarse con valores numéricos. En caso de utilizarse la palabra clave distinct, el resultado será el promedio de los valores distintos de la columna x.

max(x) Devuelve el valor máximo de la columna x de aquellas filas que cumplen la cláusula WHERE.

min(x) Devuelve el valor mínimo de la columna x de aquellas filas que cumplen la cláusula WHERE.

La cláusula WHERE sólo puede especificarse en las instrucciones de CTSQL: SELECT, UPDATE o DELETE.

En las funciones sum(x), avg(x), max(x) y min(x), x puede ser una expresión en lugar de una columna. En este caso, la función se evalúa sobre los valores de la expresión como cómputo de cada fila que satisface la cláusula WHERE.

La palabra clave distinct sólo puede utilizarse con columnas, nunca con expresiones.

La palabra clave distinct sólo puede utilizarse una sola vez en una select_list.

Los valores nulos afectan a los valores agregados de la siguiente forma:

- count(*) cuenta todas las filas aunque el valor de cada columna sea nulo.
- count(distinct x), avg(x), sum(x), max(x) y min(x) ignoran las filas con valores nulos para x. Sin embargo, si x contiene solamente valores nulos, devuelven nulo (NULL) para esa columna.

Funciones de Fecha

date(expresión**)**	Devuelve el valor de expresión convertido a tipo DATE.
day(expresión**)**	Devuelve el día de la fecha que resulte de la conversión de expresión.
month(expresión**)**	Devuelve el mes de la fecha que resulte de la conversión de expresión.
year(expresión**)**	Devuelve el año de la fecha que resulte de la conversión de expresión.
weekday(expresión**)**	Devuelve el día de la semana de la fecha que resulte de la conversión de expresión.
mdy(mes, día, año**)**	Devuelve un valor de tipo DATE (mes, día y año son expresiones que representan los componentes de la fecha).

Funciones de Hora

time(expresión**)**	Devuelve el valor de expresión convertido a tipo TIME.
hour(expresión**)**	Devuelve las horas de la hora que resulten de la conversión de expresión.
minute(expresión**)**	Devuelve los minutos de la hora que resulten de la conversión de expresión.
second(expresión**)**	Devuelve los segundos de la hora que resulten de la conversión de expresión.
hms(hora, minutos, segundos**)**	Devuelve un valor de tipo TIME. hora, minutos y segundos son expresiones numéricas que representan los componentes de la hora.
mt(expresión**)**	Devuelve AM o PM, dependiendo del valor de la hora que devuelva expresión (Meridian Time).
tomt(expresión**)**	Devuelve el valor de tipo TIME correspondiente a la hora del meridiano.

Función de Sistema

typelength(expr1, expr2**)**	Devuelve el número de bytes ocupados por un tipo de dato SQL. expr1 debe dar el número correspondiente al tipo de dato (0 al 3 y del 5 al 8) y expr2 debe dar la longitud, siendo este último significativo sólo en el tipo de dato CHAR.

3. Elementos básicos del CTSQL

▪ Construcción de Identificadores CTSQL

Un identificador CTSQL es el nombre de un elemento del CTSQL. Estos elementos son los siguientes:

- Bases de datos.
- Tablas.
- Columnas.
- Índices.
- Claves primarias y claves referenciales (primary keys y foreign keys).
- Views.
- Sinónimos.
- Alias.

Un identificador puede estar formado por letras, números y signos de subrayado (_). El primer carácter deberá ser siempre una letra, pudiendo ser la longitud del identificador de 1 a 18 caracteres. En el caso de introducir letras en mayúsculas en un identificador del CTSQL, éste las convertirá a minúsculas.

La estructura de un identificador CTSQL es la siguiente:

letra [subrayado | letra | número]

Donde:

Elemento	Características
letra	Puede ser mayúscula o minúscula. Rango: a-z.
subrayado	Símbolo de subrayado (_).
número	Carácter numérico. Rango: 0-9

En una misma base de datos, los identificadores de cada tipo deben ser unívocos, aplicándose adicionalmente esto a los siguientes grupos de elementos en su conjunto:

- Tablas y views
- Índices, claves primarias y claves referenciales.
- Columnas (en la misma tabla) y alias (en la misma instrucción SELECT).

▪ Tipos de Datos del CTSQL

Un tipo de datos es un conjunto de valores representables. Un valor es atómico y, por tanto, no permite subdivisiones. Un valor puede ser una cadena de caracteres alfanuméricos o un número. El tipo de datos determina los valores que acepta una columna de una tabla, así como su modo de almacenamiento.

Los números correspondientes a cada tipo de dato en el diccionario de la base de datos son los siguientes:

Número	Tipo
0	CHAR
1	SMALLINT
2	INTEGER
3	TIME
5	DECIMAL
6	SERIAL
7	DATE
8	MONEY

0.- CHAR(n)

Cadena de caracteres alfanuméricos de longitud n, que tiene que ser mayor o igual que uno y menor o igual que 32.767 (1<= n <= 32767). El número de bytes que reserva en disco este tipo de dato es el indicado en n. El parámetro n es necesario indicarlo para la definición del elemento en concreto.

1. - SMALLINT

Este tipo de dato permite almacenar números enteros. El rango de valores numéricos admitidos está entre -32.767 y +32.767, ambos inclusive. Este tipo de dato ocupa siempre 2 bytes, independientemente del número de caracteres numéricos a grabar. Es decir, si se introduce el valor 1, éste ocupará lo mismo que el valor 32.767. Por lo tanto, al estar ya predefinida la longitud de este tipo de datos no será necesario indicarla.

2. - INTEGER

Este tipo de dato permite almacenar números enteros. El rango de valores numéricos que admite está entre -2.147.483.647 y +2.147.483.647, ambos inclusive. Este tipo de dato ocupa siempre 4 bytes, independientemente del número de caracteres numéricos que se incluyan como dato. Al igual que en el tipo de dato anterior, su longitud está ya predefinida.

3. - TIME

Este tipo de dato admite horas con el formato por defecto HH:MM:SS y almacenada como el número de segundos desde las 00:00:01. Este tipo de dato es equivalente a un INTEGER, siendo en este caso el rango de valores admitidos entre 1 y 86.400, que se corresponden con los valores 00:00:01 y 24:00:00, respectivamente. Al igual que el tipo de dato INTEGER, ocupa 4 bytes.

Los tipos TIME se introducen como una secuencia de hora, minutos y segundos, sin separador entre ellos (tampoco espacios en blanco). Su representación depende de la variable de entorno DBTIME. Asimismo, estos tipos de datos permiten ordenaciones y comparaciones horarias entre dos columnas de tipo TIME.

5 - DECIMAL [(m [, n])]

Tipo de dato que almacena números decimales de coma flotante con un total de m dígitos significativos (precisión) y n dígitos a la derecha de la coma decimal (escala). m puede ser como máximo menor o igual a 32 (m <= 32) y n menor o igual que m (n <= m).

Cuando se asignan valores a m y n, la variable decimal tendrá punto aritmético fijo. El segundo parámetro n es opcional, y si se omite se tratará como un decimal de coma flotante. Un elemento decimal(m) tiene una precisión m y un rango de valor absoluto entre 10^{-130} y 10^{125}. En caso de no especificar parámetros a un elemento decimal, éste será tratado como decimal(16).

6 - SERIAL[(n)]

Este tipo de dato sólo puede utilizarse en CTSQL, no así en el lenguaje de cuarta generación COOL. Los valores que admite son números enteros secuenciales y únicos asignados automáticamente por CTSQL (contador automático). El valor inicial de este tipo de dato, por defecto, es uno (1). No obstante, este valor inicial puede cambiarse al asignar un valor a n en la definición del elemento (columna). El CTSQL no permitirá la introducción ni la modificación de ningún valor ya utilizado sobre una columna SERIAL. La ocupación en disco es de 4 bytes.

Comportamiento del tipo de dato SERIAL:

- Al grabar el primer valor sobre la columna definida como SERIAL, el valor asignado automáticamente por defecto es 1. Esto es así, excepto si se asigna un valor inicial en la definición de la columna. Asimismo, dicha columna SERIAL también puede inicializarse mediante la asignación de un valor concreto en una inserción o actualización. Si dicha columna SERIAL dispone de datos, el siguiente valor a insertar será el máximo más uno, siempre y cuando no haya existido alguna baja.
- En caso de que la columna SERIAL haya tenido ciertos valores y se hayan borrado, el siguiente valor a insertar será el máximo valor que haya existido o exista más uno.
- Por defecto, en las inserciones de valores nuevos en la columna SERIAL siempre se incrementará al valor máximo existente, o que haya existido, más uno.

7 - DATE

Tipo de dato que admite fechas con el formato por defecto dd/mm/yyyy y las almacena como el número de días transcurridos partiendo desde el 01/01/1900. Dicho número será positivo o negativo dependiendo de si la fecha es posterior o anterior, respectivamente, al valor inicial indicado. Este tipo de dato es equivalente a un INTEGER, ocupando por tanto 4 bytes en disco.

Los tipos DATE son introducidos como una secuencia de día, mes y año con caracteres numéricos:

- El día puede representarse como el día del mes (1 ó 01, 2 ó 02, etc.).
- El mes se representa como un número (1 equivale a enero, 2 a febrero, etc.).
- El año se representa como un número de cuatro dígitos (0001 a 9999). En caso de visualizar dos dígitos para el año, CTSQL asume que el año es 19yy.

Se puede ordenar por una columna de tipo DATE y hacer una comparación cronológica entre dos columnas de tipo DATE.

El orden en que se introduzcan el año, mes y día dependerá del valor de la variable de entorno DBDATE.

8 - MONEY [(m [, n])]

Tipo de dato que almacena números decimales de coma flotante con un total de m dígitos significativos (precisión) y n dígitos a la derecha de la coma decimal (escala). Ésta es la misma definición que se asignó al tipo de dato DECIMAL. Este tipo de dato se utiliza para representar cantidades referentes a unidades monetarias, empleando la variable de entorno DBMONEY para identificar la unidad monetaria a la que se refiere el importe definido como valor.

El tipo de dato MONEY(m) se define como el tipo de dato DECIMAL(m,2), y si no se especifican parámetros, MONEY se trata como una variable de tipo DECIMAL(16,2).

El siguiente cuadro recoge de forma resumida los valores de los diferentes tipos de datos:

Tipo de dato	Valor mínimo	Valor máximo	Ocupación(bytes)
CHAR(n)	1	32.767	n
SMALLINT	-32.767	+32.767	2
INTEGER	-2.147.483.647	+2.147.483.647	4
TIME	00:00:01	24:00:00	4
DECIMAL(m,n)	10^{-130}	10^{125}	1+m/2
SERIAL(n)	-2.147.483.647	+2.147.483.647	4
DATE	01/01/0001	31/12/9999	4
MONEY(m,n)	10^{-130}	10^{125}	1+m/2

En caso de existir valores introducidos en las tablas, la conversión de tipos de datos entre sí es válida en los siguientes casos:

Desde \ Hasta	Char	Smallint	Integer	Time	Decimal	Serial	Date	money
Char	-	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Smallint	No	-	Sí	Sí	Sí	Sí	Sí	Sí
Integer	No	Sí	-	Sí	Sí	Sí	Sí	Sí
Time	No	Sí	Sí	-	Sí	Sí	Sí	Sí
Decimal	No	Sí	Sí	Sí	-	Sí	Sí	Sí
Serial	No	Sí	Sí	Sí	Sí	-	Sí	Sí
Date	No	Sí	Sí	Sí	Sí	Sí	-	Sí
Money	No	Sí	Sí	Sí	Sí	Sí	Sí	-

La conversión SÍ será válida cuando los datos a convertir cumplan las reglas de rango especificadas para cada tipo en particular.

▪ *Atributos del CTSQL*

Un atributo es una característica asignada a las columnas que integran las tablas de la base de datos. Estas características son muy importantes en la definición de la base de datos, pues indican al sistema reglas de integridad sobre los datos, evitando su verificación por programa.

Los atributos del CTSQL sólo podrán utilizarse en las instrucciones CREATE TABLE y ALTER TABLE dentro del Lenguaje de Definición de Datos (DDL).

Los atributos son los siguientes (Véase el anexo de Atributos del CTSQL):

Atributos	Afectan al CTSQL	Efecto
CHECK	X	Este atributo se utiliza para definir una expresión de tipo booleana (condición)
DEFAULT	X	Este atributo se utiliza para asignar un valor por defecto
DOWNSHIFT	X	Este atributo convierte las letras mayúsculas de los valores de una variable de tipo CHAR en minúsculas
FORMAT	X	Este atributo permite establecer el formato de presentación de los valores de las variables a las que se asigne
LABEL	X	Este atributo define una etiqueta (título) para la variable que se está definiendo
LEFT	X	Alinea a la izquierda el valor asignado a una variable
NOENTRY	X	Este atributo no permite insertar un valor a la variable
NOT NULL		Este atributo obliga a que la columna tenga un valor distinto de nulo (NULL)
NOUPDATE	X	Este atributo no permite actualizar un valor a la variable
PICTURE	X	Este atributo especifica la máscara de edición de una variable de tipo CHAR
RIGHT	X	Alinea a la derecha el valor asignado a una variable
UPSHIFT	X	Este atributo convierte las letras minúsculas de los valores de una variable de tipo CHAR en mayúsculas
ZEROFILL	X	Este atributo alinea a la derecha el valor sin tener en cuenta el tipo de dato (numérico o alfanumérico), y rellena con ceros por la izquierda

▪ Columna

Una columna es un conjunto de valores del mismo tipo de datos, no necesariamente distintos y variables en el tiempo. Una columna se define con un nombre (identificador), el tipo de datos al que pertenecen sus valores y los atributos de éstos. Dichos atributos son los que se han comentado en el epígrafe anterior para el CTSQL.

Un valor de una columna es la unidad más pequeña que puede seleccionarse o actualizarse. Una columna representa un atributo en una entidad tabla de la base de datos.

Una columna se puede nombrar simplemente por su identificador o por medio de su identificador precedido por el nombre de la tabla y un punto decimal (tabla.columna). Esta forma de identificar las columnas es de uso obligado en caso de existir ambigüedades en una instrucción de CTSQL, por ejemplo, si se utilizan dos columnas del mismo nombre pero diferentes tablas en una misma instrucción de CTSQL.

Ejemplo:

```
create table provincias (
1    provincia smallint not null label "Cod. Provincia",
     descripcion char(20) label "Provincia",
     prefijo smallint label "Prefijo");
```

La instrucción CREATE TABLE será la que provoque la creación de una tabla con sus respectivas columnas. En este ejemplo, la tabla *provincias* contiene tres columnas, denominadas *provincia, descripcion* y *prefijo*, con distintos tipos de datos.

Una columna no puede denominarse igual que la tabla a la que pertenece. El CTSQL no tiene problema, ya que distingue perfectamente entre una columna y una tabla con el mismo nombre, pero con el lenguaje de cuarta generación COOL se producirían errores de compilación.

▪ Tabla Base

Una tabla base es un conjunto de columnas no necesariamente distintas. Una tabla se define con un nombre (identificador) y la definición de sus correspondientes columnas. Una tabla representa una entidad de la base de datos.

El concepto de tabla es similar al de fichero en otros lenguajes de programación. Las diferencias entre una tabla de la base de datos y un fichero son las siguientes:

- Las filas de una tabla no guardan ningún orden particular en el almacenamiento, sin embargo, pueden recuperarse en cualquier orden con sólo especificarlo en la operación que la define.
- Las columnas de una tabla pueden recuperarse en cualquier orden con sólo especificarlo al indicar sus nombres.
- No es necesario acceder a todas las filas ni a todas las columnas, existen operaciones que permiten manejar un subconjunto de las filas, o un subconjunto de las columnas, o ambas posibilidades a la vez (tablas derivadas o views).

Ficheros *.dat y *.idx

Cuando se crea una tabla, ésta se representa físicamente por dos ficheros, con extensiones .dat e .idx. Los nombres de estos ficheros están formados por una parte alfabética y otra numérica:

- La parte alfabética se toma del nombre asignado a la tabla en el momento de crearla.
- La parte numérica corresponde al número de identificación de la tabla dentro de la base de datos. Este número se denomina *tabid* y se trata de una de las columnas de la tabla de la base de datos *systables*. Este número es secuencial y comienza en el 150 para las tablas del usuario, y cambiará automáticamente cada vez que se altere la estructura de la tabla.

Con estas dos partes (alfabética y numérica) se construye un nombre de fichero con ocho caracteres más la extensión .dat e .idx. Por defecto, estos ficheros se sitúan en el subdirectorio .dbs que representa la base de datos.

Ejemplo:

La tabla de provincias estaría representada físicamente por los ficheros:

provi150.dat
provi150.idx.

El fichero .dat almacenará la información introducida por el operador a la tabla en concreto. Por su parte, el fichero .idx almacena la estructura de los índices creados para dicha tabla. Todos los índices se encuentran en el mismo fichero.

Al crear la tabla, los ficheros .dat e .idx ocupan 0 y 2.048 bytes en disco, respectivamente, incrementándose ambos en múltiplos de 1 Kbyte. En este espacio, el fichero de índices (.idx) incluye la siguiente información:

- Número de índices.
- Descripción de cada índice (si es único o duplicado y de qué columnas está compuesto).
- Número de registros, incluyendo huecos provocados por borrado de filas, en el fichero de datos.
- Longitud (en bytes) de una fila.
- Número de registros del fichero de índices.
- Lista de huecos de los ficheros de datos e índices.
- Una estructura en árbol (B+ tree) por índice.

El espacio que ocupará el fichero de datos .dat para una tabla se calcula de la siguiente forma:

(número_filas + número_huecos) * (bytes_fila + 1)

Concepto	Significado
número_filas	Número de filas existentes en la tabla. Este número se encuentra en la columna nrows de la tabla del catálogo de la base de datos systables. Para que la información de esta columna sea coherente con el número de filas reales de la tabla hay que ejecutar previamente la instrucción UPDATE STATISTICS del CTSQL
número_huecos	Número de huecos existentes en el fichero provocados por el borrado de filas
bytes_fila	Número de bytes que ocupa una fila en la tabla de la que se desea calcular su ocupación en disco. Esta información se encuentra en la columna rowsize de la tabla del catálogo de la base de datos systables

Almacenamiento de la Información

CTSQL aumenta de forma automática los ficheros físicos .dat e .idx correspondientes a la tabla en la que se inserta información. En el fichero de datos (.dat), este aumento se realiza cuando no exista ninguna fila marcada.

Una fila se marca cuando se produce su borrado mediante la instrucción DELETE del DML. Esto significa que cuando el CTSQL produce un borrado de filas, éstas no se borran, sino que se marcan. Estas filas marcadas se sustituirán por nuevas filas agregadas a la tabla mediante la instrucción INSERT del DML.

La única forma de compactar estas filas marcadas (borradas) para la recuperación de espacio, sin inserción de filas nuevas, es mediante la ejecución del comando *trepidx* sobre dicha tabla.

Ejemplo:

Supongamos que disponemos de una tabla con la siguiente información y cuyo orden físico de almacenamiento de las filas es el siguiente:

Provincia	Descripción
1	ALAVA
2	ALBACETE
3	ALICANTE
4	ALMERIA

Si en esta tabla se produce un borrado de la fila 3:

```
delete from provincias
      where provincia = 3
```

la tabla quedaría con la siguiente información:

Provincia	Descripción	M
1	ALAVA	
2	ALBACETE	
3	ALICANTE	*
4	ALMERIA	

M es una *marca* interna que ocupa 1 byte y que indica, en el caso de contener información (*), que la fila se encuentra borrada.

Cuando se inserta una nueva fila, por ejemplo: 5, ÁVILA:

```
insert into provincias (provincia, descripcion)
      values (5,"AVILA")
```

ésta se almacenará en el mismo lugar que la borrada previamente:

Provincia	Descripción	M
1	ALAVA	
2	ALBACETE	
5	AVILA	
6	ALMERIA	

En caso de realizar una lectura secuencial de las filas de esta tabla, la información que devolverá es la expuesta en el ejemplo anterior.

▪ *Tabla temporal*

Una tabla temporal es igual que una tabla base, pero con la diferencia de que su borrado será automático en el momento que termine el programa que la creó. Asimismo, las tablas temporales no pertenecen a ninguna base de datos. Debido a esta característica, estas tablas son un buen soporte para compartir información entre distintas bases de datos.

Una tabla temporal se crea mediante la instrucción CREATE TEMP TABLE del sublenguaje DDL del CTSQL o bien mediante la cláusula INTO TEMP de la instrucción SELECT.

▪ *Tabla derivada*

Una tabla derivada es una tabla resultado que contiene la información obtenida por la ejecución de la instrucción de lectura SELECT del CTSQL. Una tabla derivada no es un objeto persistente, no tiene nombre y no está definida por separado de la operación que la genera (como ya se ha indicado, esta operación la realiza la instrucción SELECT).

Ejemplo, ejecutar la siguiente instrucción:

```
select provincia, descripcion from provincias
      where provincia = 28
```

En este caso, la tabla derivada está compuesta de una sola fila, que es la siguiente:

Provincia	Descripción
28	MADRID

Ahora bien, una tabla derivada también puede estar compuesta de varias filas.

Ejemplo:

```
select provincia, descripcion from provincias
      where provincia < 10
```

La tabla derivada que se generaría con el ejemplo anterior sería:

Provincia	Descripción
1	ALAVA
2	ALBACETE
3	ALICANTE
4	ALMERIA
5	AVILA
6	BADAJOZ
7	BALEARES
8	BARCELONA
9	BURGOS

▪ *View*

Una view es una tabla derivada a la que se le asigna un nombre. Asimismo, podría tratarse como una tabla base de la base de datos, pero sin ficheros .dat e .idx físicos. Pese a no existir físicamente en el almacenamiento, se trata de un objeto persistente en la base de datos. El contenido de una view es evaluado cada vez que se la invoca en una instrucción CTSQL.

La creación de una view la provoca la instrucción CREATE VIEW del sublenguaje DDL del CTSQL, mientras que su borrado se efectúa con la instrucción DROP VIEW. La estructura física de una view está definida por la instrucción SELECT asociada en su creación. Dicha estructura física no puede alterarse.

Las views tienen fundamentalmente dos utilidades:

- Presentar los datos de las tablas en una estructura más próxima al concepto que posee el usuario de una aplicación sobre cómo se estructuran dichos datos. Esto es debido a que, por motivos de normalización de tablas, es posible que se llegue a una fragmentación que suponga constantes operaciones de join a la hora de recuperar dichas tablas. También puede ser un medio de nominar instrucciones SELECT cuya estructura sea de uso muy frecuente.
- Delimitar el acceso a los datos en conjuntos preestablecidos, de forma que se pueda establecer una privacidad lógica sobre ellos. Dado que la tabla derivada resultante de la view consta de columnas y filas, esta delimitación de acceso se puede aplicar en uno o en ambos sentidos. Sobre columnas significa no acceder a determinados datos de cada entidad (tabla). Por ejemplo: Ver el precio de venta de un artículo, pero no el de coste. Aplicado este criterio a las filas, significa acceder a un subconjunto de las entidades posibles en una tabla. Por ejemplo: Un departamento sólo puede ver sus filas.

Asimismo, una view puede utilizarse para modificar la(s) relación(es) base, ya sea insertando nuevas tuplas, ya actualizando las existentes o borrándolas. Esto supone que cualquiera de estas operaciones se descompondrá en las correspondientes sobre las tablas base que la componen (tablas relacionadas en la instrucción SELECT de creación de la view) y no sobre la view, que es el resultado de la instrucción SELECT asociada a ella.

Antes de la creación de una view destinada a la actualización de tablas, es imprescindible una definición clara de la integridad referencial de las tablas base implicadas (claves primarias, *primary keys* y claves referenciales, *foreign keys*).

Actualización de una View derivada de una sóla tabla

Para poder actualizar una view es necesario que ésta cumpla las siguientes restricciones:

- La lista_select de la instrucción SELECT que define la view no puede incluir expresiones. Un valor agregado se considera una expresión.
- Asimismo, la instrucción SELECT que define la view no puede incluir las siguientes cláusulas: DISTINCT, GROUP BY y HAVING.
- Ninguna columna de la tabla base y no incluida en la view debe estar definida como NOT NULL.
- Debe incluir la clave primaria (primary key) de la tabla base o aquellas columnas que identifiquen unívocamente la tupla fila en la tabla.

Actualización de una View derivada de otra View

Si la view de la que se deriva la que queremos actualizar se deriva a su vez de una única relación base (tabla), se aplicarán las mismas restricciones comentadas en el apartado anterior, aun existiendo una cadena de views.

Si, por el contrario, la view que se desea actualizar y/o la view base incluye(n) algún join (enlace con otra tabla), se aplicarán las restricciones de la actualización de views derivadas de más de una tabla que se comentan en el apartado siguiente.

Actualización de una View derivada de dos tablas

En este caso se aplican todas las restricciones indicadas en el supuesto de derivación de una sola tabla, salvo la referente a la inclusión de la clave primaria en la select_list de la instrucción SELECT que define la view. Esto se debe a que la nueva relación procede de una operación de join siempre interno, esto es, que la instrucción SELECT devolverá las tuplas de ambas tablas que satisfagan la condición de tener valores iguales en la(s) columna(s) especificada(s) en dicha operación.

Así pues, el tratamiento de la identificación unívoca de las tuplas de cada tabla base implicada en la view presenta dos casos paralelos a la relación que establezca el join:

- Si la relación es unaria, es decir, si a cada tupla recuperada de la primera tabla le corresponde una sola tupla de la segunda, se refiere a un join entre las claves primarias (primary key) de ambas tablas.
- Si la relación es n-aria, es decir, si a cada tupla de la primera tabla le corresponde una o más de la segunda, se refiere a un join entre la clave primaria (primary key) de la primera tabla y la clave referencial de la segunda (foreign key).

Esta distinción se aplicará al definir las restricciones de actualización, en dependencia de la operación de actualización de las tablas base (INSERT, DELETE y UPDATE).

Ejemplos:

En los siguientes ejemplos, se denominará A y B respectivamente a las partes izquierda y derecha del join.

1. En actualización de un join con relación entre claves primarias (primary key):
   - INSERT:
     Si existe clave en A y no en B o a la inversa:
     - Los datos del que existen deben coincidir. Si fuera así, no se insertaría.
     - INSERT sobre la tabla que no existe.
   - DELETE:
     - Si existen en ambas tablas, se borran.
   - UPDATE:
     - Si existen en ambas tablas, se actualizan.
     - En caso de no existir se produce un INSERT.

2. En actualización de un join con relación entre clave primaria (primary key) en A y clave referencial (foreign key) en B:
   - INSERT:
     Si no existe una clave para ninguno de A y B:
     - Se produce un INSERT en ambas tablas.

     Si existe clave de A (clave primaria):
     - Sus datos deben coincidir.

     Si existe clave de B (clave referencial):
     - La(s) columna(s) de B que establezca(n) el join debe(n) ser nula(s).
     - Los demás datos deben coincidir.
     - UPDATE de la(s) columna(s) de join al valor de la clave primaria (primary key) de A.
   - DELETE:
     No se puede borrar A.
     Si la(s) columna(s) del join en B está(n) definida(s) como NOT NULL:
     - DELETE de B.

     Si ésta(s) admite(n) nulos:
     - Se pone(n) a nulos (borrado lógico).
   - UPDATE:
     No se puede actualizar A.
     Si existe B:
     - UPDATE sobre B.

Actualización de una View derivada de mas de dos tablas

Se descompondrá la view en sus joins (al igual que sucede en el caso de views derivadas de dos tablas) y se aplicarán las reglas ya definidas, con la restricción adicional en este caso de que una tabla mediante su clave referencial no puede enlazar dos tablas mediante sus claves primarias. Por el contrario, una tabla con clave primaria sí puede enlazar dos tablas mediante las claves referenciales de éstas. Al resultado de cada join se le aplicará el siguiente join.

▪ *Fila o Tupla*

Una fila es un conjunto no vacío de valores tal que:

- Pertenece a una tabla.
- El valor i-ésimo de la fila pertenece a la columna i-ésima de la tabla.

Una fila se corresponde con un caso concreto de la tabla a la que pertenece. Una fila es la unidad más pequeña de datos que puede ser insertada o borrada en una tabla. El concepto de fila equivale al de registro en términos de ficheros clásicos.

▪ *ROWID*

El rowid es el número de fila en una tabla, y la forma más rápida de acceso a ésta, por lo que su utilización es prioritaria.

Ejemplo, la tabla de provincias incluye la siguiente información:

Provincia	Descripción	Prefijo	ROWID
1	ALAVA	945	1
2	ALBACETE	967	2
3	ALICANTE	965	3
4	ALMERIA	951	4
33	ASTURIAS	985	5
5	AVILA	918	6
6	BADAJOZ	924	7
7	BALEARES	971	8
8	BARCELONA	93	9

Como se puede comprobar en esta tabla, el rowid es un número secuencial.

▪ *Índices*

El esquema relacional no incluye definición alguna sobre el método de acceso a los datos, por lo que las instrucciones relativas a índices incluidas en CTSQL son una extensión de éste. Los motivos de definición de un índice son fundamentalmente los siguientes:

- Aumentar la velocidad de acceso a la tabla reduciendo el tiempo de respuesta.
- Asegurar la identificación unívoca de filas de una tabla.
- Facilitar los enlaces entre tablas (join), la ejecución de instrucciones SELECT del SQL con condiciones de subselect y la realización de entradas de datos multitabla mediante el objeto Form del lenguaje de cuarta generación COOL de Cosmos.

Un índice se puede crear o borrar en cualquier momento durante el desarrollo de una aplicación.

Una de las características de los sistemas relacionales es la transparencia de la estructura física de los datos y el acceso a éstos para usuarios y programas. El tratamiento de índices en CTSQL es igualmente transparente e independiente de dicha estructura mediante las instrucciones CREATE INDEX y DROP INDEX. No obstante, en el diseño de índices de una base de datos han de tenerse en cuenta las siguientes características de funcionamiento:

- La adición indiscriminada de índices puede ralentizar las operaciones de inserción y actualización de filas (INSERT y UPDATE, respectivamente).
- La estructura de una tabla específica determina el tipo de consultas que se van a realizar sobre ella, ya sea por su naturaleza (es distinta una tabla de clientes a una de líneas de albaranes), ya por el ámbito de aplicación de distintos usuarios (consultan de forma diferente una tabla de líneas de albarán, un departamento de expediciones y otro de marketing). Por ello, deben considerarse los criterios restrictivos (cláusula WHERE) así como la ordenación deseada (cláusula ORDER BY) de las consultas más frecuentes, para así crear los índices adecuados. En este sentido, el optimizador de CTSQL utilizará preferentemente la clave primaria (primary key), u otros índices si se especifican condiciones sobre sus componentes, o se desea una ordenación con una estructura similar a un índice existente.
- Los índices juegan un papel fundamental en el enlace de tablas, por lo que es preferible que exista al menos un índice en la tabla que pueda tener más filas.

Las siguientes recomendaciones se deben tener en cuenta cuando se usen índices:

- Identifique y defina siempre la clave primaria de la tabla, ya que con ello asegura la consistencia de sus datos (además de los restantes aspectos que se mencionan en el apartado sobre Integridad Referencial).
- No se deben construir índices que no se vayan a utilizar en las consultas normales. Si ciertas consultas son periódicas o poco frecuentes, y además no son interactivas, por ejemplo un listado mensual, cree el índice sólo antes de realizar consulta, borrándolo al final.
- Al igual que en el caso anterior, si periódicamente va a cargar datos mediante la instrucción LOAD y tiene la certeza de que los registros a cargar no van a generar duplicidades en la clave primaria de la tabla receptora, borre los índices antes de iniciar la carga, regenerándolos posteriormente.
- No construya índices sobre tablas pequeñas (menos de 200 filas), ya que su rendimiento se degrada.
- No construya índices duplicados sobre columnas con un conjunto pequeño de valores posibles, por ejemplo: sexo, estado civil, respuestas sí/no, etc., puesto que ya de por sí el acceso al índice consume bastante tiempo y en este caso se convertiría en una búsqueda secuencial de todas las filas que contuviesen el mismo valor.
- Utilice condiciones sobre columnas indexadas en la cláusula WHERE y, si se desea una ordenación específica, defina un índice con las columnas de la cláusula ORDER BY.
- A la hora de optimizar una sentencia de CTSQL (instrucciones SELECT, UPDATE y DELETE), utilice la variable de entorno OUTOPT.

▪ *Integridad*

De un modo genérico, y referido al esquema lógico de una base de datos, se puede definir la integridad como la consistencia de las definiciones que forman dicho esquema y de la utilización de las mismas.

Los niveles de integridad que pueden considerarse son:

♦ Integridad de Diccionario.

Este tipo de integridad se refiere a la definición de características de una columna de forma centralizada en el diccionario de la base de datos (tablas sys*). Se corresponde con los atributos de columnas que suponen restricciones de los valores posibles para esa columna (INCLUDE y CHECK), formato de la misma (FORMAT y PICTURE), operaciones posibles con ella (NOENTRY y NOUPDATE), etc.

Mediante la centralización de dichos atributos de columnas, que son mantenidos automáticamente por CTSQL, se evita incluir estas definiciones en los distintos programas COOL que manejen dichas columnas, asegurando así la integridad de éstas en todo el desarrollo de la aplicación.

♦ Integridad de Tablas.

Considerando una relación tabla como un conjunto de objetos del mismo tipo, y un tupla (fila) como una instancia de esa relación, podemos definir la integridad de tablas como la ausencia de redundancias en esa relación, esto es, cada tupla es única en esa relación.

Así, en el diseño de una base de datos se utiliza el procedimiento de normalización de tablas, cuyo objetivo es agrupar los distintos fragmentos de información redundantes en nuevas tablas. En este procedimiento surge el concepto fundamental de clave primaria (primary key), que no es otra cosa sino la columna o combinación de columnas de una tabla que identifican unívocamente a cada una de sus filas. Dado el carácter unívoco de esta clave, ésta no podrá contener valores nulos en ninguna de sus columnas.

♦ Integridad Referencial.

La integridad referencial es la consistencia de datos entre las tuplas de dos o más tablas. La relación entre dichas tablas se establece mediante una o varias columnas de enlace (máximo ocho), estableciendo así una relación de interdependencia entre ellas. Dichas columnas de enlace (join) serán los componentes de la definición de la clave primaria (primary key) en una tabla y de las claves referenciales (foreign keys) en el resto de tablas.

Por tanto, la clave primaria (primary key) de una tabla es un índice único, simple o compuesto, una o varias columnas, respectivamente, cuyos componentes identifican una fila. La clave primaria (primary key) de una tabla se crea mediante las instrucciones CREATE TABLE o ALTER TABLE del CTSQL.

Las claves referenciales son los identificadores que establecen la relación con las claves primarias (primary key) de otra tabla. Esta relación consiste en controlar la existencia de ciertos datos en la tabla que contiene la clave primaria (primary key) para poder utilizarlos en la tabla con claves referenciales (foreign keys). Dicha relación se controla a nivel de modificaciones (UPDATE) y borrado (DELETE) de filas en la tabla con la clave primaria (primary key). Las acciones que se pueden tomar son:

RESTRICT:	Si el operador intenta borrar o modificar una fila en la tabla que contiene la clave primaria, la cual establece la integridad referencial, y dichos valores existen en otras tablas que contienen claves referenciales (RESTRICT) significa que dicha operación no será permitida.
SET NULL:	Esta opción indica que la actualización o el borrado sobre la tabla que contiene la clave primaria, la cual establece la integridad referencial, implicará el poner a nulos todas aquellas columnas componentes de la clave referencial (foreign key) en las filas afectadas.

Ejemplo: las tablas de clientes y proveedores referencian a la tabla de provincias para controlar la integridad de datos. La estructura de estas tablas es la siguiente:

```
create table provincias(
     provincia smallint not null label "Cod. Provincia",
     descripcion char(20) label "Provincia",
     prefijo smallint label "Prefijo")
primary key (provincia) ;
create table clientes(
     cliente integer not null label "Codigo Cliente",
     empresa char(25) upshift label "Empresa",
     apellidos char(25) label "Apellidos",
     nombre char(15) label "Nombre",
     direccion1 char(25) label "Direccion1",
     direccion2 char(25) label "Direccion2",
     poblacion char(15) label "Poblacion",
provincia smallint label "Provincia",
     distrito integer label "Distrito",
     telefono char(13) label "Telefono",
     formpago char(2) label "Forma de Pago",
     total_factura money(11,2) label "Total Facturado")
primary key (cliente) ;
create index i2_cliente on clientes(empresa );
alter table clientes foreign key for1_cli(provincia) references
provincias
     on update restrict
     on delete restrict ;
create table proveedores(
     proveedor integer not null label "Codigo Proveedor",
empresa char(25) upshift label "Empresa",
     apellidos char(25) label "Apellidos",
     direccion1 char(25) label "Direccion1",
     direccion2 char(25) label "Direccion2",
     poblacion char(15) label "Poblacion",
     provincia smallint label "Provincia",
     distrito integer label "Distrito",
     telefono char(13) label "Telefono")
primary key (proveedor) ;
create index i2_proveedor on proveedores(empresa );
alter table proveedores foreign key for1_pro(provincia) references
provincias
on update restrict
     on delete restrict ;
```

Como se puede comprobar en estas tablas, existe una columna común a todas ellas. Esta columna es el código de la provincia, denominada provincia en las tres tablas. En este caso, el nombre de la columna que establece la relación coincide en las tres tablas, si bien esta denominación común no es obligatoria. Por contra, lo que sí es necesario es que dicha columna coincida en tipo y longitud de dato SQL en las tres tablas. En nuestro ejemplo, las tres columnas provincia son SMALLINT.

La columna provincia en la tabla de provincias es el componente de la clave primaria (primary key). Esto significa que no pueden existir dos provincias con el mismo código, es decir, establece la unicidad de datos.

```
primary key (provincia);
```

Sin embargo, la columna provincia en las tablas de clientes y proveedores forma dos claves referenciales: for1_cli y for1_pro, respectivamente, estableciendo relación con la tabla de provincias. Esta relación siempre será por medio de la clave primaria:

```
alter table clientes foreign key for1_cli (provincia) references
provincias
     on update restrict
     on delete restrict ;
alter table proveedores foreign key for1_pro (provincia) references
provincias
     on update restrict
     on delete restrict ;
```

Como se puede comprobar en estos ejemplos, ambas claves referenciales se crean con la cláusula RESTRICT tanto en actualización como en borrado (UPDATE y DELETE, respectivamente). Esto significa que si el operador intenta borrar o modificar el código de la provincia en la tabla de provincias a la que pertenece algún cliente o proveedor, dicha operación se cancelará de forma automática (RESTRICT).

Una vez identificadas las claves primarias y referenciales surge la necesidad de definir qué acciones tomar cuando se presente algún caso de los mencionados. Dichas acciones afectan a ambas tablas, puesto que son interdependientes. En este caso consideraremos las siguientes situaciones:

- Se actualiza cualquiera de las columnas que componen la clave primaria (primary key) en la tabla referenciada.
- Se borra cualquiera de las columnas de la clave primaria (primary key) de la tabla referenciada.
- Se insertan filas en la tabla referenciante, clientes o proveedores, sin enlace en la tabla referenciada (provincias).

Las acciones a tomar serían:

- Impedir la acción iniciada sobre la tabla referenciada (actualización/borrado).
- Poner a nulos las columnas componentes de la clave referencial en la tabla referenciante, es decir, deshacer el enlace de las filas afectadas, lo que podríamos denominar un borrado lógico.
- Impedir la inserción de tuplas en la tabla referenciante que no tengan enlace con la tabla referenciada, forzando la existencia previa de una tupla en esta tabla con el (los) valor(es) adecuado(s) en la(s) columna(s) de enlace.

▪ *Transacciones*

La seguridad física y lógica de los datos es un aspecto primordial en una aplicación de base de datos. Son muchas las causas que pueden dañar una base de datos, dejándola en un estado inconsistente e incluso inaccesible, por ejemplo fallos del sistema, tales como caídas del sistema, sobrecarga de procesos, memoria de paginación insuficiente, terminación anormal de CTSQL, etc.

Para sistematizar la seguridad de los datos de una base de datos ante este tipo de problemas, se pueden activar, opcionalmente, las transacciones.

Una transacción es el conjunto de operaciones relativas a la recuperación y actualización sobre la base de datos que se realizan de forma unitaria e indivisible, esto es, o se realizan totalmente o no se realizan. En otras palabras, es el modo de agrupar varias instrucciones DML (INSERT, DELETE y UPDATE) y asegurar que dicho grupo se ejecute completamente.

Para ello hay que marcar explícitamente el principio y fin de dicho grupo de instrucciones mediante las siguientes:

- BEGIN WORK: inicia la transacción.
- COMMIT WORK: finaliza la transacción haciendo definitivos los cambios.
- ROLLBACK WORK: finaliza la transacción deshaciendo los cambios realizados en la base de datos.

No obstante, para poder activar una transacción es preciso que la base de datos, ya sea en su creación, ya en otro momento, tenga definido un fichero de transacciones (log file). Las instrucciones que las activan son las siguientes:

CREATE DATABASE base_datos WITH LOG IN "fichero_log"

Crea una base de datos transaccional.
START DATABASE base_datos WITH LOG IN "fichero_log"

Esta instrucción convierte en transaccional una base de datos creada previamente sin transacciones. Asimismo, en una base de datos definida con transacciones sirve para cambiar el fichero transaccional (log file). Es decir, asigna un nuevo fichero transaccional a una base de datos que no lo tuviera o crea uno nuevo.

La consecuencia práctica de estas instrucciones es que en el fichero transaccional (log file) se registrarán tanto el comienzo de la transacción como su fin. Esta operación permite:

- Que el programador decida en base a condiciones de error de determinadas instrucciones la realización o no de las modificaciones incluidas en la transacción como un todo.

- Que si ocurre algún fallo del sistema durante el proceso de la transacción, al faltar la marca de cierre de la misma se pueda producir una vuelta a un estado consistente de la base de datos.

Así pues, mediante este mecanismo se puede establecer un nuevo nivel de integridad de datos de las tablas bajo control del programador en operaciones tales como borrado de datos obsoletos de tablas relacionadas por un enlace (join), asegurando además el cumplimiento de todas las instrucciones implicadas.

Una transacción supone el bloqueo de todas las filas implicadas en las operaciones que reciben este tratamiento, siendo liberadas al finalizarla (COMMIT WORK y ROLLBACK WORK) y cerrando todos los cursores abiertos durante la misma. Dicho bloqueo no es automático, sino que para cada una de dichas filas hay que provocar una actualización (UPDATE), aunque ésta sea absurda (por ejemplo, actualizar una fila dejando el mismo valor). En este momento dicha fila se encuentra bloqueada para el resto de usuarios que tengan acceso a la base de datos.

En instalaciones cliente-servidor hay que tener en cuenta que cada sistema operativo UNIX tiene un número máximo de bloqueos (tunable parameter), por lo que habrá que ajustar éste o utilizar, en su caso, el bloqueo de tabla (LOCK TABLE) que inhibe el bloqueo de fila.

En MS-DOS y Windows sólo existe la posibilidad de bloqueo en instalaciones de red local. En estos sistemas, el comando share es el que permite el bloqueo de filas y tablas. Dicho comando tiene un número máximo de bloqueos (parámetro /L), por lo que habrá que ajustarlo o utilizar, en su caso, el bloqueo de tabla (LOCK TABLE) que inhibe el bloqueo de fila.

▪ *Catálogo de la Base de Datos*

En una base de datos existen tres tipos de tablas: tablas base o de usuario, tablas del catálogo de la base de datos y tablas del Entorno de Desarrollo.

- Las tablas base contienen información de los datos manejados por la aplicación, y su mantenimiento es realizado por el operador .
- Las tablas del catálogo de la base de datos se generan de forma automática a la hora de crear la base de datos (mediante la instrucción CREATE DATABASE). Estas tablas son mantenidas de forma automática por el CTSQL cada vez que se utiliza una instrucción del sublenguaje DDL, y su ubicación es dentro del subdirectorio que representa la base de datos (base_datos.dbs). Su nombre comienza por sys y son las siguientes:

Nombre	Contenido
systables	Esta tabla contiene una fila por cada tabla existente en la base de datos, incluidas las del catálogo y programación
syscolumns	Contiene información de todas las columnas de cada una de las tablas que componen la base de datos
sysindexes	Índices de cada una de las tablas de la base de datos
systabauth	Permisos sobre las tablas de la base de datos
syscolauth	Permisos sobre las columnas que componen las tablas de la base de datos
sysusers	Permisos de acceso a la base de datos
sysviews	Contiene las instrucciones SELECT que construyen las views existentes en la base de datos
sysdepend	Lista de las tablas base y derivadas componentes de las views de la base de datos
syssynonyms	Contiene los sinónimos definidos en la base de datos
sysforeign	Lista de las claves referenciales (foreign keys) existentes en cada una de las tablas de la base de datos
syscolattr	Atributos CTSQL asignados a cada una de las columnas que componen las tablas de la base de datos
syscollating	Secuencia de ordenación empleada en la base de datos

Por su parte, en instalaciones con gateways se genera un catálogo específico para el intercambio de información entre los distintos gestores de bases de datos (Oracle, Informix, Ingres). Las tablas de este catálogo equivalen a las expuestas anteriormente, con la diferencia de que sus nombres comienzan por mb. Dichas tablas son las siguientes:

mbtables	mbviews
mbcolumns	mbdepend
mbindexes	mbsynonyms
mbtabauth	mbforeign
mbcolauth	mbcolattr
mbusers	

▪ Alias

Un alias es un nombre alternativo asignado en una instrucción de lectura SELECT a una tabla o columna. La construcción de un alias es como un identificador más de la base de datos. Estos identificadores pueden emplear caracteres en mayúsculas, siendo válidos únicamente para dicha instrucción SELECT. Equivale a un renombrado de tabla o columna en ejecución, y es temporal mientras se está ejecutando la instrucción de lectura.

Un alias de columna se define en la lista_select de la instrucción SELECT, y sólo puede emplearse en la cláusula ORDER BY de aquélla. Un alias de columna se emplea para mostrar una etiqueta distinta de la columna a la que se le asigna cuando se muestra en una ventana la tabla derivada devuelta por su lectura.

Por su parte, un alias de tabla es especialmente útil cuando se desea enlazar una tabla consigo misma. Un alias de tabla se define en la cláusula FROM de una instrucción SELECT.

Un alias se define mediante un espacio en blanco entre su nombre y el de la columna o tabla en cada caso. Por ejemplo:

- Alias de columna:

```
select provincia Codigo, descripcion from provincias
```

En este ejemplo, se ha definido un alias (Codigo) a la columna provincia. A la hora de presentar la tabla derivada devuelta por esta instrucción SELECT, la etiqueta de dicha columna será Codigo en lugar de Cod. provincia (label de dicha columna).

- Alias de tabla:

```
select provincias.descripcion, prov2.descripcion
     from provincias, provincias prov2
     where provincias.provincia = 28
      and prov2.provincia = 19
```

En este ejemplo, se ha definido un alias a la tabla provincias que se lee dos veces en esta instrucción. El alias es el que distingue entre ambas tablas que físicamente es una sola.

▪ Sinónimos

Un sinónimo es un nombre alternativo asignado a una tabla de la base de datos. Cuando se asigna un sinónimo a una tabla, ésta podrá ser llamada indistintamente por su propio nombre o por el sinónimo.

Este concepto es similar al explicado anteriormente para el alias de una tabla. La diferencia principal entre ambos conceptos es que el sinónimo persiste hasta la ejecución de la instrucción DROP SYNONYM, mientras que el alias sólo es válido para la instrucción de lectura SELECT que lo define.

Un sinónimo se crea mediante la instrucción CREATE SYNONYM y se borra de la base de datos mediante DROP SYNONYM.

4. Uso de Instrucciones de SQL

Las reglas y particularidades de la sintaxis de una instrucción de tipo CTSQL son las siguientes:

- Una instrucción CTSQL puede introducirse en tantas líneas como se precisen. Es decir, cada palabra que constituye una instrucción CTSQL podría introducirse en una línea, pulsando [Intro] después de cada una de ellas.
- Asimismo, puede utilizarse la barra espaciadora o el tabulador para la indentación de las instrucciones. Por ejemplo:

```
select provincia, descripcion, prefijo
            from provincias
            where provincia between 1 and 10;
```

- Las constantes de tipo alfanumérico (CHAR), hora (TIME) o fecha (DATE) tienen que encerrarse entre comillas, simples o dobles (' ' o " "). Por ejemplo:

```
"27/06/1965"              Fecha
"16:10:01"                Hora
"Manuel Ruiz López"       Alfanumérico
```

- Una instrucción CTSQL puede introducirse en mayúsculas o minúsculas. Sin embargo, en aquellos valores constantes entrecomillados sí existe diferencia entre minúsculas y mayúsculas:

```
sElEcT CLIENTE, empresa, DIRECCION1 from clientes
       WHERE empresa = "CIFERSA";

 select cliente, empresa, direccion1
       from clientes
       where empresa = "CIFERSA";
```

El resultado de estas dos instrucciones de CTSQL es idéntico, ya que el valor constante a buscar está escrito en mayúsculas. El resto de la instrucción es indiferente cómo se escriba.

- Si se incluye más de una instrucción CTSQL en el mismo fichero, éstas tienen que separarse por un punto y coma (;). Este signo es opcional en la última instrucción especificada.

Módulo COOL

Todas las instrucciones propias del CTSQL pueden incluirse en los módulos de programación del lenguaje de cuarta generación (COOL) como parámetros en las llamadas a los siguientes métodos predefinidos en Cosmos:

A la hora de desarrollar una aplicación con Cosmos, al manejar los datos de la base de datos hay que incluir las llamadas a los métodos correspondientes en el módulo de programación.

Carga y Descarga de datos Desde/Hacia Ficheros ASCII

CTSQL dispone de dos instrucciones específicas que permiten generar o leer un fichero ASCII con los datos a insertar en una tabla o como resultado de una proyección de algunas o todas las columnas de ésta respectivamente.

Estas instrucciones son LOAD y UNLOAD. Realmente, estas instrucciones permiten establecer el vínculo entre ficheros del sistema operativo, con un formato específico, y las instrucciones del CTSQL, INSERT y SELECT respectivamente. La sintaxis de cada una de estas instrucciones es la siguiente:

- Instrucción LOAD:

```
LOAD FROM "fichero_ascii" INSERT INTO tabla [(lista_columnas)]
```

- Instrucción UNLOAD:

```
UNLOAD TO "fichero_ascii"
    SELECT [ALL | DISTINCT | UNIQUE ] lista_select
    FROM lista_tablas
      [USING lista_variables [ON tabla]]
      [WHERE {condición_comparación | condición_subselect}]
      [GROUP BY lista_grupos]
      [HAVING condición_grupos]
      [ORDER BY columna [ASC | DESC] [, columna  [ASC | DESC] [ , ... ]]
```

El formato que debe tener un fichero para la instrucción LOAD es la representación ASCII (en texto) de los valores a cargar, seguido cada uno de ellos de un carácter delimitador, de tal forma que por cada línea haya tantos caracteres de delimitación como columnas a cargar. El delimitador empleado por defecto es el carácter ASCII 124 (|). No obstante, este carácter se puede definir mediante la variable de entorno DBDELIM. Si el separador se encuentra como carácter en uno de los valores de una columna, aquél debe ir precedido de un backslash (\) al objeto de que no sea considerado como separador sino como un carácter más. A su vez, cada registro viene separado por un carácter de nueva línea. Este mismo formato es el que genera la instrucción UNLOAD. Su aspecto es el siguiente:

5. El Lenguaje de Definición de Datos (DDL)

Este sublenguaje del CTSQL es el encargado de la definición y mantenimiento de aquellos elementos SQL que se manejarán en el desarrollo de la aplicación.

Dichos elementos son los siguientes:

- Bases de datos.
- Tablas.
- Filas o tuplas.
- Columnas.
- Índices.
- Sinónimos.
- Views

Bases de datos

Las operaciones que se pueden llevar a cabo sobre una base de datos son las siguientes:

- Creación de la base de datos.
- Borrado de la base de datos.
- Selección de la base de datos.
- Cierre de la base de datos.

Creación de la base de datos

La primera operación que hay que realizar siempre a la hora de comenzar el desarrollo de una aplicación es crear la base de datos. Esta operación se realiza mediante la instrucción CREATE DATABASE, cuya sintaxis es la siguiente:

CREATE DATABASE base_datos [WITH LOG IN "fichero_log"]
[COLLATING "fichero_ordenación"]

Esta instrucción crea un subdirectorio cuyo nombre será igual al de la base de datos indicada en base_datos más la extensión .dbs (base_datos.dbs). Este subdirectorio se genera por defecto dentro del directorio en curso. No obstante, en caso de tener activa la variable de entorno DBPATH, este subdirectorio se generará por defecto en el primer directorio indicado en dicha variable.

Dentro del directorio base_datos.dbs se crean asimismo unos ficheros, con extensiones .dat e .idx, que se corresponden con las tablas del catálogo de la base de datos (tablas sys*).

Ejemplo:

```
create database almacen
```

Esta instrucción crea un directorio, denominado almacen.dbs, dentro del cual se generarán las tablas del catálogo de la base de datos (sys*).

La base de datos que se genera por defecto no es transaccional. Para crear una base de datos transaccional habría que ejecutar necesariamente la instrucción CREATE DATABASE del CTSQL con la cláusula WITH LOG IN.

Ejemplo:

```
create database almacen with log in "\cosmos\demo\lunes"
```

Esta instrucción generaría el fichero \cosmos\demo\lunes para recoger las transacciones realizadas sobre la base de datos.

Asimismo, todas las bases de datos que se generen utilizarán la tabla de caracteres GCS#2 de IBM para las funciones de ordenación. Si se desea modificar esta tabla habrá que utilizar la cláusula COLLATING junto al nombre del fichero compilado con la secuencia de ordenación deseada.

La compilación de este fichero de ordenación se tiene que realizar desde el sistema operativo mediante el comando tcollcom.

Borrado de la base de datos

La operación de borrado de una base de datos consiste en eliminar el directorio propio de la base de datos junto con las tablas que se hubiesen generado para la misma, tanto si dichas tablas se encuentran dentro del mismo directorio como si no. Esta operación de borrado de la base de datos se realiza mediante la instrucción DROP DATABASE, cuya sintaxis es la siguiente:

DROP DATABASE base_datos

Selección y cierre de la base de datos

En CTSQL, cualquier operación que se vaya a realizar siempre tendrá que ser sobre una base de datos previamente seleccionada. CTSQL sólo permite tener seleccionada una base de datos. En caso de querer seleccionar una segunda, la primera se cerrará de forma automática. Las instrucciones encargadas de abrir y cerrar una base de datos tienen la siguiente estructura:

- Para abrir una base de datos:

  DATABASE base_datos [EXCLUSIVE]

Parámetro	Significado
base_datos	Identificador CTSQL que se corresponde con el nombre de la base de datos
EXCLUSIVE	Palabra reservada que provoca el bloqueo de la base de datos para el resto de usuarios

- Para cerrarla:

  CLOSE DATABASE

Ambas instrucciones no distinguen entre bases de datos transaccionales y no transaccionales.

En instalaciones cliente-servidor, la instrucción DATABASE busca por defecto la base de datos a seleccionar en UNIX en el directorio en curso, mientras que en el caso del Windows lo hace en el directorio de trabajo especificado en las propiedades del icono.

No obstante, si hubiésemos definido la variable de entorno DBPATH, la instrucción DATABASE buscará la base de datos en cualquiera de los directorios especificados en dicha variable. Esta búsqueda en los paths de la variable se realiza de izquierda a derecha.

Tablas

Las operaciones que se pueden realizar sobre las tablas son:
- Creación.
- Definición de la integridad referencial.
- Modificación de la estructura.
- Renombrado.
- Borrado.

Creación

Una vez creada la base de datos según lo explicado anteriormente, el siguiente paso a realizar para el desarrollo de una aplicación consiste en la creación de las tablas que compondrán dicha base de datos. La creación de estas tablas se realiza mediante la instrucción CREATE TABLE del CTSQL, cuya sintaxis es la siguiente:

```
CREATE [TEMP] TABLE nombre_tabla (
        nombre_columna tipo_sql atributos,
        nombre_columna tipo_sql atributos, …)
[IN "directorio"]
[PRIMARY KEY (lista_columnas)]
[FOREIGN KEY identificador (lista_columnas)
REFERENCES nombre_tabla
        ON UPDATE {RESTRICT | SET NULL}
        ON DELETE {RESTRICT | SET NULL}]
[FOREIGN KEY …]
```

Definición de la integridad referencial

Una vez generada la base de datos y las respectivas tablas que la componen, el siguiente paso a ejecutar será la definición de las claves primarias y referenciales que controlarán la integridad entre todas las tablas de la base de datos.

Con esta relación entre claves primarias y referenciales se intenta evitar la inconsistencia de datos entre las tablas de la base de datos. Dentro de este epígrafe consideraremos los siguientes apartados:

- Definición de la clave primaria.

La definición y borrado de la clave primaria se encuentra implícita respectivamente en la sintaxis de las instrucciones CREATE TABLE y ALTER TABLE.

La columna o columnas que compongan la clave primaria deberán tener asignado el atributo NOT NULL. La definición de la clave primaria implica la creación de un índice único que estará compuesto por dichas columnas. El número máximo de columnas que pueden formar una clave primaria es 8, o bien que entre todas ellas no superen los 120 bytes. Esta limitación es la misma que fija ISAM para la definición de los índices.

- Definición de las claves referenciales.

La definición de las claves referenciales se encuentra implícita en la sintaxis de las instrucciones CREATE TABLE y ALTER TABLE. La instrucción ALTER TABLE permite asimismo el borrado de claves referenciales.

La definición de una clave referencial no supone la creación de un índice. Como norma general, una clave referencial tiene que estar compuesta por el mismo número de columnas que la respectiva clave primaria en la tabla referenciada. Asimismo, dichas columnas tienen que especificarse en el mismo orden y, además, ser del mismo tipo de dato SQL.

Modificación de la estructura

Una tabla puede alterar su estructura en cualquier momento durante el desarrollo de una aplicación, existan o no datos en ella. Este cambio de estructura puede consistir en las siguientes operaciones:

- Agregar una columna nueva.
- Modificar el tipo o atributos de una columna ya existente en la tabla.
- Borrar un columna existente en la tabla.
- Agregar o borrar la clave primaria de la tabla.
- Agregar o borrar claves referenciales de la tabla.

La forma de modificar la estructura de una tabla es mediante la instrucción ALTER TABLE del CTSQL, cuya sintaxis es:

```
ALTER TABLE nombre_tabla
    [ADD (nombre_columna tipo_sql atributos
    [BEFORE columna_existente],…)][,]
    [DROP (columna_existente)][,]
    [MODIFY (columna_existente atributos
    [REMOVE lista_atributos],…)][,]
    [PRIMARY KEY (lista_columnas)][,]
    [FOREIGN KEY identificador (lista_columnas)
    REFERENCES nombre_tabla
      ON UPDATE {RESTRICT | SET NULL}
      ON DELETE {RESTRICT | SET NULL}][,]
    [DROP PRIMARY KEY][,]
    [DROP FOREIGN KEY identificador][,]
```

En caso de borrar una columna que componga un índice o parte de él, dicho índice se borrará igualmente. Esta regla se aplica también a las claves primarias y referenciales.

Cuando se efectúa una modificación en la estructura de una tabla, el nombre de sus ficheros físicos también se modifica. Este cambio de nombre se produce al cambiar el número de identificador interno de dicha tabla, ya que éste es parte del nombre de los ficheros. Este número se corresponde con la columna tabid de la tabla systables.

Asimismo, en el momento en que se produce una modificación de la estructura de una tabla se compactan sus ficheros, eliminando todas aquellas filas marcadas que indican que se encuentran borradas por la instrucción DELETE.

Renombrado

Una de las operaciones que se pueden realizar sobre las tablas es cambiarles el nombre. Este cambio puede implicar muchos problemas si la aplicación se encuentra acabada, ya que habría que modificar el nombre de dicha tabla en todos aquellos módulos COOL donde se hubiese hecho referencia a ella.

La operación de renombrado de una tabla se realiza mediante la instrucción RENAME TABLE del CTSQL, cuya sintaxis es la siguiente:

```
RENAME TABLE nombre_tabla TO nuevo_nombre_tabla
```

Borrado

El borrado de una tabla provoca la desaparición física de los ficheros .dat e .idx, y consecuentemente la de la información contenida en ellos. Esta operación se lleva a cabo con la instrucción DROP TABLE del CTSQL, cuya sintaxis es la siguiente:

```
DROP TABLE nombre_tabla
```

Filas o tuplas

A nivel de filas, la única operación que puede realizarse desde el sublenguaje DDL del CTSQL es la actualización de estadísticas para la optimización de los accesos a dicha tabla.

La actualización de estadísticas consiste en modificar la columna nrows, perteneciente a la tabla del catálogo de la base de datos systables. El dato que se graba en dicha columna corresponde al número real de filas que contiene la tabla o tablas en cuestión en ese momento. Esta operación la realiza la instrucción UPDATE STATISTICS del CTSQL, cuya sintaxis es la siguiente:

```
UPDATE STATISTICS [FOR TABLE nombre_tabla]
```

Si no se especifica el nombre de la tabla de la cual se desean actualizar sus estadísticas, esta operación se realizará sobre todas las tablas de la base de datos seleccionada.

Columnas

Además de lo indicado a nivel de columnas en la creación o modificación de la estructura de las tablas, la única operación que se puede realizar sobre una columna con el sublenguaje DDL es cambiarle el nombre.

El cambio de nombre de una columna puede implicar muchos problemas si la aplicación se encuentra acabada, ya que habría que modificar el nombre de dicha columna en todos aquellos módulos COOL donde se hubiese hecho referencia a ella.

La operación de renombrado de una columna se realiza con la instrucción RENAME COLUMN del CTSQL, cuya sintaxis es la siguiente:

```
RENAME COLUMN nombre_tabla.nombre_columna
TO nuevo_nombre_columna
```

Índices

El sublenguaje DDL del CTSQL contempla únicamente dos operaciones respecto a los índices: creación y borrado.

Los índices pueden crearse y borrarse en cualquier momento durante el desarrollo de una aplicación. La ejecución de cualquiera de estas dos operaciones lleva implícita la modificación del fichero de índices (.idx), construyendo o eliminando respectivamente su árbol correspondiente.

Creación de Índices

La creación de índices para las tablas de la base de datos se lleva a cabo con la instrucción CREATE INDEX del CTSQL, cuya sintaxis es la siguiente:

```
CREATE [UNIQUE | DISTINCT] INDEX nombre_índice
ON tabla (columna [ASC | DESC] [, columna [ASC | DESC] [, …])
```

Las características y restricciones respecto a la creación de índices son las siguientes:

- El nombre del índice que se asigna en esta instrucción (nombre_índice) sirve para identificar dicho índice a la hora de su posible borrado.
- Un índice puede construirse sobre una o más columnas (máximo 8) de una tabla, siendo simple o compuesto, respectivamente.
- Si las columnas componentes de un índice pueden tener varios valores iguales en distintas filas de la tabla, entonces a éste se le denomina índice duplicado. Por el contrario, si una determinada combinación de valores sólo puede aparecer una vez en la tabla, entonces se llamará índice único.
- La longitud máxima de un índice es 120 bytes, ya sea único o duplicado. Es decir, la suma de las longitudes en bytes de las columnas que lo componen no debe superar los 120 bytes.
- Al crear un índice se puede establecer el orden de recuperación de los datos asociados, ascendente (por defecto) o descendente.
- No se puede crear un índice con una estructura (lista de columnas componentes) idéntica a la de otro índice existente sobre la misma tabla.
- Un índice no se puede crear con columnas de distintas tablas de la base de datos.
- Un índice no se puede crear con parte de una columna.
- No se puede crear un índice duplicado, simple o compuesto, para una tabla en la que existan más de 32.767 filas con valores duplicados para el conjunto de columnas que componen dicho índice.
- Los tipos de datos de las columnas que componen un índice pueden ser distintos.

Borrado de Índices

El borrado de índices se efectúa mediante la instrucción DROP INDEX del CTSQL, cuya sintaxis es la siguiente:

```
DROP INDEX nombre_índice
```

El nombre del índice (nombre_índice) es el identificador que se le asigna en el momento de crearlo. Como ya se ha indicado anteriormente, este nombre sólo sirve como referencia para la operación de borrado.

Sinónimos

Las dos únicas operaciones que se pueden llevar a cabo con los sinónimos desde el sublenguaje DDL del CTSQL son las que hacen referencia a su creación y borrado.

Creación de Sinónimos

La operación de creación de sinónimos de tablas se lleva a cabo a través de la instrucción CREATE SYNONYM del CTSQL, cuya sintaxis es la siguiente:

```
CREATE SYNONYM sinónimo FOR tabla
```

Esta instrucción crea un sinónimo para la tabla especificada en tabla. Este sinónimo persistirá en la base de datos hasta que se borre.

Borrado de Sinónimos

El borrado de sinónimos se realiza mediante la instrucción DROP SYNONYM del CTSQL, cuya sintaxis es la siguiente:

```
DROP SYNONYM sinónimo
```

Views

Al igual que sucedía con los sinónimos y los índices, en el caso de las views el sublenguaje DDL del CTSQL contempla únicamente la realización de las operaciones de creación y borrado.

Creación de Views

La creación de views se realiza mediante la instrucción CREATE VIEW, cuya sintaxis es la siguiente:

```
CREATE VIEW nombre_view [(columnas)]
AS instrucción_select [WITH CHECK OPTION]
```

La única forma de modificar la estructura de una view es borrándola y volviéndola a crear. La cláusula WITH CHECK OPTION se emplea cuando la view es utilizada para actualizar las tablas originales que componen dicha view.

Una view persistirá en la base de datos hasta que se borre.

Borrado de Views

Para proceder al borrado de una view habrá que ejecutar la instrucción DROP VIEW del CTSQL, cuya sintaxis es la siguiente:

```
DROP VIEW nombre_view
```

Esta instrucción se encarga de eliminar del catálogo de la base de datos (tablas sys) toda la información generada por la instrucción CREATE VIEW. No obstante, hay que indicar que una view no tiene representación física en el disco. La instrucción DROP VIEW es a las views lo que la DROP TABLE es a las tablas.

6. El Lenguaje de Control de Acceso a Datos (DCL)

Dentro de la definición de una base de datos, el sublenguaje de control de acceso a datos del CTSQL contempla los siguientes procesos:

- Permisos.
- Bloqueos.

Permisos

Uno de los aspectos a tener en cuenta en el diseño de una base de datos es quiénes pueden acceder y a qué datos, así como el tipo de acceso que se autoriza.

En función de los datos y del tipo de usuario se podrán establecer distintos niveles de privacidad en el acceso. En este sentido se pueden considerar dos niveles en el control de accesos:

- Referido a los datos.
- Referido a los usuarios.

El control de los permisos de acceso sólo tiene sentido en instalaciones de Cosmos de tipo multiusuario.

Niveles de acceso respecto a los datos

En una base de datos existen tres niveles de acceso respecto a los datos:

- Acceso a la base de datos, que delimita:
  - Los derechos de uso de instrucciones de definición de datos (DDL), tales como CREATE, DROP, etc.
  - La manipulación de los datos de la base de datos (QL y DML), tales como SELECT o UPDATE.
  - La posibilidad de establecer permisos sobre tablas, columnas, etc. (DCL).
  - Activar el tratamiento de transacciones.

- Acceso a tablas, referido al acceso a sus datos en instrucciones de recuperación o modificación de cualquier tupla de la misma, modificación de su esquema añadiendo columnas o atributos de éstas o creando índices.

- Acceso a las columnas de una tabla, que afecta a la consulta (SELECT) y modificación (UPDATE) de columnas específicas de cada tupla de una tabla.

Niveles de acceso respecto a los usuarios

Al igual que en el caso anterior se contemplan tres niveles:

- DBA: Este tipo de usuario es el administrador de la base de datos, y su acceso es absoluto a todos los niveles de datos. Además, incluye control sobre el tratamiento de transacciones. Un usuario DBA puede conceder o retirar cualquier permiso a cualquier otro usuario. El creador de la base de datos se convierte automáticamente en usuario DBA y no puede dejar de serlo, esto es, ningún usuario puede retirarle este privilegio, ni siquiera él mismo. El administrador de la base de datos puede conceder este permiso a otros usuarios, convirtiéndose éstos a su vez en usuarios DBA, aunque con menor prioridad.

- RESOURCE: Este nivel es el inmediato inferior a DBA y permite la creación de tablas, índices, etc., así como la concesión de permisos a otros usuarios sobre lo creado o sobre aquellas tablas no creadas por él, pero sobre las que se le hayan concedido permisos mediante la cláusula WITH GRANT OPTION de la instrucción GRANT.

- CONNECT: Este nivel es el más bajo y sólo permite acceso a instrucciones de consulta o actualización de tablas.

Las instrucciones del CTSQL que permiten establecer estas jerarquías de acceso son GRANT y REVOKE.

Permisos sobre Bases de Datos

Cuando se crea una base de datos, el único usuario que por defecto tiene acceso a ella es el que la crea (usuario DBA propietario). Este usuario administrador será quien conceda o revoque permisos sobre esa base de datos y sus respectivas tablas. La operación de conceder o revocar permisos sobre la base de datos se realiza respectivamente mediante las instrucciones GRANT y REVOKE, cuyas sintaxis son:

```
GRANT {DBA | RESOURCE | CONNECT} TO {PUBLIC | lista_usuarios}
REVOKE {DBA | RESOURCE | CONNECT} FROM {PUBLIC | lista_usuarios}
```

En instalaciones de Cosmos en redes de área local, así como en arquitecturas cliente-servidor, el nombre del usuario corresponde al valor que contenga la variable de entorno DBUSER en cada uno de las máquinas cliente.

Permisos sobre Tablas y Columnas

Cuando se crea una tabla, la instrucción CREATE TABLE se encarga también de conceder todos los permisos sobre dicha tabla a todos los usuarios PUBLIC. Los permisos que pueden asignarse a las tablas son los siguientes:

Permisos	Efecto
SELECT [(lista_columnas)]	Lectura de la tabla. En caso de indicar columnas el permiso de lectura se aplicará sólo a éstas
UPDATE [(lista_columnas)]	Actualización de la tabla. En caso de indicar columnas el permiso de actualización se aplicará sólo a éstas
INSERT	Inserción de filas en la tabla
DELETE	Borrado de filas en la tabla
INDEX	Creación de índices para dicha tabla
ALTER	Alteración de la estructura de la tabla. Éste es el único permiso que no se asigna por defecto en la creación de la tabla mediante la instrucción CREATE TABLE
ALL [privileges]	Engloba todos los permisos anteriores

Como ya ha quedado indicado en el epígrafe anterior, para que un usuario pueda acceder a los datos será necesario haberle concedido previamente el permiso de acceso a la base de datos.

Para cambiar los permisos generados por la instrucción CREATE TABLE se podrán emplear las instrucciones GRANT o REVOKE con el siguiente formato:

```
GRANT permisos ON tabla TO {PUBLIC | lista_usuarios} [WITH GRANT OPTION]
REVOKE permisos ON tabla FROM {PUBLIC | lista_usuarios}
```

Bloqueos

En aplicaciones de bases de datos y entornos multiusuario surge como problema efectivo el de la concurrencia. Ésta se puede definir como la simultaneidad de accesos de distintos usuarios a los mismos datos para operaciones completas de consulta y/o actualización. Aunque CTSQL procese una petición sobre una base de datos cada vez, virtualmente, para cada peticionario, su operación es única (primero consulta, después actualiza). Esto podría llevar a inconsistencias tales como lectura de valores irreales, actualización de valores aleatoria e, incluso, pérdida de información.

Para solventar estas posibles inconsistencias, el CTSQL dispone de un mecanismo, denominado bloqueo, que consiste básicamente en la limitación del acceso a determinados datos por parte del primer programa que los solicita, de forma que los demás no puedan leerlos ni modificarlos.

En función de los diferentes objetos manejados por el CTSQL, existen distintos niveles de bloqueos, según afecten a la base de datos, a las tablas o a las filas.

Bloqueo de la Base de Datos

El bloqueo de una base de datos se refiere a la selección o apertura de ésta por un solo usuario. La instrucción que permite bloquear la base de datos es la siguiente:

```
DATABASE base_datos [EXCLUSIVE]
```

En este caso, la cláusula EXCLUSIVE debe especificarse para producir el bloqueo de la base de datos indicada en base_datos. Esta operación sólo podrá ejecutarla el administrador de la base de datos, es decir, aquellos usuarios que tengan permiso DBA sobre la misma.

En el caso de ejecutar esta instrucción sobre una base de datos que ya estuviese seleccionada por otro usuario se produciría un error indicando la imposibilidad del bloqueo. Este error también se produciría si un usuario con permiso sobre la base de datos intentase seleccionarla y ésta ya hubiese sido seleccionada previamente por un usuario DBA en modo exclusivo (EXCLUSIVE).

Desbloqueo de la Base de Datos

Para desbloquear una base de datos habrá que provocar la ejecución de la instrucción CLOSE DATABASE, ya que al cerrar la base de datos se pierde el bloqueo existente. La sintaxis de esta instrucción es la siguiente:

```
CLOSE DATABASE
```

Hay que tener en cuenta que esta instrucción se encuentra implícita en la instrucción DATABASE del sublenguaje de manipulación de datos (DML) del CTSQL. Al ejecutar DATABASE se cierra la base de datos en curso, seleccionándose la que se hubiese indicado en su sintaxis.

Bloqueo de Tablas

La forma de bloquear una tabla de una base de datos es mediante la ejecución de la instrucción LOCK TABLE, cuya sintaxis es la siguiente:

```
LOCK TABLE tabla IN {SHARE | EXCLUSIVE} MODE
```

Como puede verse en esta sintaxis, existen dos niveles de bloqueo sobre las tablas:

Niveles de Bloqueo	Significado
IN SHARE MODE	Esta cláusula indica que se permite la lectura de la tabla sobre la que se aplica el bloqueo a otros usuarios, pero no su actualización
IN EXCLUSIVE MODE	Esta cláusula indica que no se permite a otros usuarios ni la lectura ni la actualización de la tabla sobre la que se aplica el bloqueo. Este tipo de bloqueo es el más adecuado cuando la actualización de la tabla implique un número elevado de filas o cualquier proceso masivo sobre la misma

Desbloqueo de Tablas

La forma de desbloquear una tabla previamente bloqueada por la instrucción LOCK TABLE es ejecutando UNLOCK TABLE, con independencia del modo de bloqueo aplicado. Su sintaxis es la siguiente:

```
UNLOCK TABLE tabla
```

Bloqueo de Filas

El CTSQL no dispone de una instrucción específica para bloquear una fila de la tabla. Este tipo de bloqueo es automático en el momento en que se intenta modificar dicha fila mediante la instrucción UPDATE del sublenguaje de manipulación de datos (DML) del CTSQL. Si esta instrucción detecta que la fila a actualizar se encuentra bloqueada por otro usuario, se queda en espera hasta que se desbloquee.

Hay ocasiones en las que interesa bloquear un número indeterminado de filas de una tabla. Esta operación sólo podrá realizarse si la base de datos sobre la que actúa es transaccional. Este tipo de bloqueo se tiene que realizar comenzando una transacción (BEGIN WORK) y actualizando todas las filas que se desean bloquear, aunque ésta sea una actualización absurda. Dichas filas quedarán bloqueadas hasta que finalice la transacción (COMMIT WORK o ROLLBACK WORK).

7. El Lenguaje de Consultas (QL)

La única forma de leer las filas contenidas en las tablas de una base de datos es mediante la instrucción SELECT del sublenguaje de consultas (Query Language) del CTSQL.

La particularidad de cualquier SQL existente en el mercado a la hora de manipular los datos grabados de las tablas es que éstos son siempre tratados en bloque. Es decir, la instrucción SELECT se encarga de leer de una o varias tablas y el resultado de esta lectura es una tabla derivada compuesta por un número determinado de columnas.

Una instrucción SELECT es una operación entre tablas cuyo resultado puede ser interpretado como otra tabla (tabla derivada). Esta tabla sería el producto cartesiano de las tablas contenidas en la cláusula FROM. En el caso de existir una cláusula WHERE se eliminarían de la tabla resultado todas aquellas filas que no cumpliesen las condiciones especificadas en dicha cláusula. De todas las filas resultantes se eliminarán, además, todas aquellas columnas que no hayan sido indicadas en la cláusula lista_select.

La sintaxis de esta instrucción SELECT puede tener los siguientes formatos:

Formato 1

```
SELECT [ALL | DISTINCT | UNIQUE ] lista_select
FROM lista_tablas
[WHERE {condición_comparación | condición_subselect}]
[GROUP BY lista_grupos]
[HAVING condición_grupos]
[ORDER BY columna [ASC | DESC] [,columna [ASC | DESC] [, …]]
[INTO TEMP tabla_temporal]
```

Formato 2

```
SELECT [ALL | DISTINCT | UNIQUE ] lista_select
FROM lista_tablas
[WHERE {condición_comparación | condición_subselect}]
[GROUP BY lista_grupos]
[HAVING condición_grupos]
[UNION [ALL]]
SELECT [ALL | DISTINCT | UNIQUE ] lista_select
FROM lista_tablas
[WHERE {condición_comparación | condición_subselect}]
[GROUP BY lista_grupos]
[HAVING condición_grupos]
[UNION [ALL]]
SELECT …
[ORDER BY columna [ASC | DESC] [,columna [ASC | DESC] [, …]]
[INTO TEMP tabla_temporal]
```

Según esta sintaxis, la instrucción SELECT mínima estará compuesta por las cláusulas lista_select y FROM. Todas las cláusulas de esta instrucción SELECT tienen que especificarse según el orden indicado en esta sintaxis.

Clausula SELECT

Esta cláusula indica las columnas que contendrá la tabla derivada obtenida como resultado de la ejecución de la instrucción SELECT. Esta cláusula es una de las obligatorias dentro de la estructura de una instrucción SELECT, y su sintaxis es la siguiente:

SELECT [ALL | DISTINCT | UNIQUE] lista_select FROM lista_tablas

Parámetro	Significado	
ALL	Palabra reservada que indica que se seleccionarán todas las filas que satisfagan la cláusula WHERE (si existe), sin eliminar las filas duplicadas que pudieran surgir como resultado en la tabla derivada. Esta cláusula es la que se utilizará por defecto en cualquier selección	
DISTINCT	Palabra reservada que indica que se eliminarán aquellas filas duplicadas del resultado de la selección. De cada grupo de estas filas duplicadas, en la tabla derivada sólo aparecerá una de ellas. Esta palabra puede aparecer una sola vez en la instrucción	
UNIQUE	Palabra reservada sinónimo de DISTINCT	
lista_select	Lista compuesta por elementos separados por comas. La lista de elementos posibles es la siguiente	
	* (asterisco)	Indica que se deben seleccionar todas las columnas de las tablas especificadas en la cláusula FROM
	tabla.*	Este elemento indica que se seleccionarán todas las columnas de la tabla especificada
	Expresión	Cualquiera de las expresiones válidas de CTSQL
	Expresión [alias]	Un alias es el nombre que se asignará a expresión en una instrucción SELECT.

Clausula FROM

Esta cláusula, obligatoria dentro de una instrucción SELECT, determina la tabla o lista de tablas de las que se desea leer datos. Si se especifica más de una tabla en esta cláusula, la tabla derivada resultante será el producto cartesiano de éstas si no se enlazan en la cláusula WHERE. Su sintaxis es la siguiente:

FROM lista_tablas

Parámetro	Significado	
lista_tablas	Lista de tablas separadas por comas. Cada una de estas tablas puede especificarse con la siguiente sintaxis [OUTER] tabla [alias]	
	OUTER	Palabra reservada que se emplea para componer outer joins
	Tabla	Nombre de la tabla perteneciente a la base de datos en curso de la cual se van a extraer datos
	Alias	Nombre alternativo asignado a la tabla tabla. Esta opción es especialmente útil cuando se desea enlazar una tabla consigo misma

Cláusula WHERE

Mediante esta cláusula se podrán especificar diferentes criterios de selección. Aquellas filas que cumplan las condiciones indicadas serán las que formarán la tabla derivada. Su sintaxis es la siguiente:

WHERE condición

Parámetro	Significado
Condición	Lista de una o más subcondiciones separadas por los operadores lógicos AND, OR o NOT

Condición de Comparación

Una condición de comparación puede definirse mediante cualquiera de las siguientes sintaxis:

- **Formato 1**

expresión operador_relación expresión

Esta condición devuelve verdadero o falso si ambas expresiones cumplen la relación establecida por el operador.

Parámetro	Significado	
Expresión	Cualquier expresión válida	
Operador_relación	Operador de relación. Puede ser cualquiera de los siguientes	
	=	Igual
	!= o <>	Distinto
	>	Mayor que
	>=	Mayor o igual que
	<	Menor que
	<=	Menor o igual que

Al establecer una comparación entre, al menos, una columna de una tabla y una columna de otra, se crea un enlace (join) entre ambas. El efecto de este enlace es crear una tabla virtual compuesta. Cada fila de esta tabla virtual es una composición de las filas de las tablas enlazadas que satisfacen la condición de enlace. Este mismo concepto puede extenderse al enlace de más de dos tablas. Este enlace siempre se realizará mediante la comparación entre las columnas que compondrán la clave primaria (primary key) y la clave referencial (foreign key) en caso de haber definido integridad referencial.

- **Formato 2**

expresión [NOT] BETWEEN expresión_desde AND expresión_hasta

Esta condición devuelve verdadero o falso en caso de que el resultado de una expresión esté o no comprendido en el rango dado por otras dos, ambas inclusive. La partícula NOT NULL invierte el resultado.

Parámetro	Significado
Expresión	Cualquier expresión válida. Este concepto ya se ha comentado en el capítulo 2 de este mismo volumen
NOT	Palabra reservada que invierte el resultado
BETWEEN	Palabra reservada que indica el rango
expresión_desde	Cualquier expresión válida cuyo resultado es el valor ini-cial del rango. Este valor inicial es incluido en el rango
expresión_hasta	Cualquier expresión válida cuyo resultado es el valor final del rango. Este valor final es incluido en el rango. Esta expresión deberá ser mayor que el de expre-sión_desde

- **Formato 3**

expresión [NOT] IN (lista_valores)

Esta condición devuelve verdadero o falso en caso de que el resultado de una expresión sea igual o no a uno de los valores incluidos en lista_valores. La partícula NOT invierte el resultado.

Parámetro	Significado
Expresión	Cualquier expresión válida. Este concepto ya se ha comentado en el capítulo 2 de este mismo volumen
NOT	Partícula que invierte el resultado
IN	Palabra reservada
Lista_valores	Lista de valores, separados por comas y encerrados entre paréntesis. Estos valores serán numéricos o alfanuméricos dependiendo del tipo de dato de expresión. Los valores de los tipos de dato CHAR, TIME y DATE tienen que ir entre comillas

- **Formato 4**

expresión [NOT] LIKE "literal"

Condición de comparación de expresiones de tipo alfanumérico. Es decir, esta condición sólo podrá aplicarse a las columnas de tipo CHAR. Devuelve verdadero o falso en caso de que el resultado de una expresión se ajuste o no al patrón definido en literal. La partícula NOT invierte el resultado.

Parámetro	Significado	
expresión	Cualquier expresión válida cuyo tipo sea alfanumérico (CHAR)	
NOT	Partícula que invierte el resultado	
LIKE	Palabra reservada	
"literal"	Literal que forma el patrón de comparación con expresión. Este literal puede contener dos caracteres que tienen un significado especial, pudiendo especificarse cada uno de ellos más de una vez. Dichos caracteres son los siguientes:	
	%	El lugar donde aparezca este carácter se podrá sustituir por ningún carácter, por uno o por más de un carácter
	_ (subrayado)	Este carácter indica que se tiene que sustituir obligatoriamente por un carácter en el lugar que ocupe

- **Formato 5**

expresión [NOT] MATCHES "literal"

Condición de comparación de expresiones de tipo alfanumérico. Es decir, sólo podrá aplicarse a las columnas de tipo CHAR. Esta condición es similar a LIKE, pero con la posibilidad de emplear más metacaracteres en el literal de la condición. Devuelve verdadero o falso en caso de que el resultado de una expresión se ajuste o no al patrón definido en literal. La partícula NOT invierte el resultado.

Parámetro	Significado	
Expresión	Cualquier expresión válida cuyo tipo sea alfanumérico (CHAR)	
NOT	Partícula que invierte el resultado	
MATCHES	Palabra reservada	
"literal"	Literal que forma el patrón de comparación con expresión. Este literal puede contener varios caracteres que tienen un significado especial, pudiendo especificarse cada uno de ellos más de una vez. Dichos caracteres son los siguientes:	
	*	El lugar donde aparezca este carácter se puede sustituir por ningún carácter, por uno o por más de un carácter
	?	Este carácter indica que se tiene que sustituir obligatoriamente por un carácter en el lugar que ocupe
	[caracteres]	Los corchetes indican que cualquiera de los caracteres incluidos entre ellos pueden aparecer en la posición que ocupan dentro del literal. Se puede indicar un rango de caracteres separando el inicial y el final por medio de un guión
	[^caracteres]	Similar al anterior, si bien el carácter ^ indica que los caracteres o rangos no pueden aparecer en la posición que ocupan dentro del literal
	\	Este símbolo indica que el carácter al que precede no debe ser considerado como metacarácter, sino como un carácter más dentro del literal

- **Formato 6**

expresión IS [NOT] NULL

Devuelve verdadero o falso en caso de que el resultado de una expresión sea o no un valor nulo. La partícula NOT invierte el resultado. En caso de no utilizar este operador para la detección de valores nulos, no se recuperarán las filas cuyos valores en la columna a la que se aplica la condición sean nulos (NULL).

Condición de Subselección

Una condición de subselect consiste en comparar una expresión, que por lo general será una columna, con una tabla derivada devuelta por otra instrucción SELECT. En CTSQL existen tres formas de realizar condiciones de subselect:

- **Formato 1**

expresión operador_relación {ALL | [ANY | SOME]} (subselect)

Esta condición devuelve verdadero o falso dependiendo de si se cumple o no cierta relación entre el resultado de una expresión y el resultado de un subquery en función del operador de relación que intervenga.

Parámetro	Significado
expresión	Cualquier expresión válida. Por lo general, será una columna de la tabla cuyos datos generarán la tabla derivada
operador_relación	Operador de relación
Subselect	Instrucción SELECT cuya lista_select sólo podrá contener una columna en la tabla derivada que genere. Esta columna tendrá que ser del mismo tipo de dato SQL que la columna que condiciona expresión
ALL	Palabra reservada que indica que el resultado de la comparación debe ser verdadero para cada uno de los valores devueltos por la subselect
ANY	Palabra reservada que indica que el resultado de la comparación debe ser verdadero al menos para uno de los valores devueltos por la instrucción subselect
SOME	Sinónimo de ANY

- **Formato 2**

expresión [NOT] IN (subselect)

Esta condición devuelve verdadero o falso dependiendo de si se cumple o no cierta relación entre expresión y el resultado de la instrucción subselect.

Parámetro	Significado
Subselect	Instrucción SELECT cuya selección sólo podrá contener una columna en la tabla derivada que genere. Esta columna tendrá que ser del mismo tipo de dato SQL que la columna que condiciona expresión
NOT	Partícula que invierte el resultado
IN	Palabra reservada que pregunta si expresión es alguno de los valores devueltos por subselect

- **Formato 3**

[NOT] EXISTS (subselect)

Esta condición devuelve verdadero o falso dependiendo de si existe o no alguna fila resultado de la instrucción subselect. La partícula NOT invertirá el resultado de la condición.

Parámetro	Significado
NOT	Partícula que invierte el resultado
EXISTS	Palabra reservada que evalúa la existencia o no de alguna fila en la tabla derivada de la instrucción subselect
subselect	Instrucción SELECT válida. La lista_select de esta instrucción puede contener más de una columna o expresión, ya que ésta no se utilizará para la comparación de expresiones de la condición

Cláusula GROUP BY

Esta cláusula devuelve una fila a partir de un conjunto de filas que tienen un denominador común para las columnas indicadas en ella.

GROUP BY lista_grupos

Parámetro	Significado
lista_grupos	Nombre de una columna, o lista de nombres de columnas separados por comas, que determinan el grupo. En lugar de los nombres de columnas se pueden introducir uno o más números enteros. Estos números enteros se refieren a la posición de las columnas en la lista_select. Esto último es obligatorio si los elementos de la lista_select por los que se desea agrupar no son nombres de columnas sino expresiones

El máximo de columnas que se pueden incluir en una cláusula GROUP BY es 8. Asimismo, la suma de las longitudes de las columnas indicadas en esta cláusula no puede exceder de 120 bytes. Estas dos condiciones vienen impuestas por el XOPEN-ISAM.

Cláusula HAVING

Esta cláusula establece condiciones de selección sobre los grupos determinados por la cláusula GROUP BY. Una de las expresiones de la condición obligatoriamente tiene que ser una función de valor agregado. Por lo general, esta cláusula siempre acompañará a la GROUP BY. No obstante, en caso de que no se incluya esta última en la sintaxis se considerará que todas las filas componen un único grupo. Su sintaxis es la siguiente:

HAVING condición_grupos

Parámetro	Significado
Condición_grupos	Cualquiera de las condiciones explicadas en la cláusula WHERE de una instrucción SELECT. La condición puede descomponerse en una lista de comparaciones separadas por AND u OR. Estas comparaciones relacionan un valor agregado sobre el grupo con una constante o con otro valor agregado sobre el mismo grupo

Cláusula ORDEN BY

Esta cláusula ordena el resultado de la selección por el valor de una o más columnas de forma ascendente o descendente. Esta tabla derivada sólo puede ordenarse por las columnas que intervienen en la selección de la instrucción SELECT. Su sintaxis es:

ORDER BY columna [ASC | DESC][,
columna [ASC | DESC][, …]

Parámetro	Significado
columna	Nombre de una columna (o alias) por el que se desea ordenar el resultado de la selección. En lugar de nombres de columnas también se pueden introducir uno o más números enteros que se refieren a la posición de las columnas dentro de la cláusula selección de la instrucción SELECT.
ASC	Palabra reservada que especifica que el resultado debe ordenarse de forma ascendente
DESC	Palabra reservada que especifica que el resultado debe ordenarse de forma descendente

Las columnas de la cláusula ORDER BY tienen que aparecer de forma implícita, es decir, por medio del símbolo * (asterisco, que indica que se listan todas las columnas de la tabla) o explícita en la cláusula lista_select de la instrucción SELECT.

El máximo de columnas que se pueden incluir en una cláusula ORDER BY es 8. Asimismo, la suma de las longitudes de las columnas indicadas en esta cláusula no puede exceder de 120 bytes. Estas dos condiciones vienen impuestas por el XOPEN-ISAM.

Cláusula INTO TEMP

Esta cláusula crea una tabla temporal con la tabla derivada devuelta por la selección. Las tablas temporales creadas por medio de esta cláusula no pertenecen a ninguna base de datos. Esto significa que podrían suponerse compartidas por todas las bases de datos. Una tabla temporal no desaparece mientras no lo haga el programa que la creó, lo que significa que se pueden volcar datos sobre una tabla temporal, cambiar de base de datos y trabajar con la tabla temporal creada con la base de datos anterior. Esta cláusula es INCOMPATIBLE con la ORDER BY, y su sintaxis es la siguiente:

```
INTO TEMP tabla_temporal
```

Parámetro	Significado
tabla_temporal	Nombre de identificador que se desea asignar a la tabla temporal. Los nombres de las columnas de la tabla temporal son los que aparecen en la lista_select de la instrucción SELECT que la crea. No obstante, en caso de utilizar una expresión (operaciones aritméticas, funciones de valores agregados, substrings, etc.) hay que asignarles obligatoriamente un nombre de alias. En este caso, los nombres de los alias serán los de las columnas de la tabla temporal

Cláusula UNION

Esta cláusula devuelve una tabla derivada compuesta por el resultado de varias instrucciones SELECT concatenadas (unidas). La sintaxis de una instrucción SELECT con la cláusula UNION es la siguiente:

```
SELECT [UNION [ALL]] SELECT [UNION [ALL]] …
[ORDER BY columna [ASC | DESC][, columna [ASC | DESC][, …]]
[INTO TEMP tabla_temporal]
```

Parámetro	Significado
UNION	Palabra reservada que indica la unión de los resultados de cada una de las instrucciones SELECT que se incluyan
ALL	La tabla derivada final (UNION) no contendrá filas duplicadas a menos que se especifique esta palabra reservada

Las características de la cláusula UNION son las siguientes:

- El número de columnas y los tipos de las columnas correspondientes de todas las cláusulas lista_select de las instrucciones SELECT deben ser idénticos.
- En la cláusula ORDER BY las columnas tienen que referenciarse por número y no por nombre.
- El operador UNION no puede aparecer en un subquery ni en la definición de una view.
- Los nombres de las columnas de la tabla resultante coinciden con los de la primera instrucción SELECT.

8. El Lenguaje de Manipulación de Datos (DML)

Este sublenguaje del CTSQL contempla las siguientes operaciones:

- Apertura y cierre de la base de datos.
- Inserción de filas.
- Borrado de filas.
- Modificación de filas.

Apertura y cierre de la base de datos

En CTSQL, cualquier operación que se vaya a realizar siempre tendrá que ser sobre una base de datos previamente seleccionada. CTSQL sólo permite tener seleccionada una base de datos. En caso de querer seleccionar una segunda, la primera se cerrará de forma automática. Las instrucciones encargadas de abrir y cerrar una base de datos tienen la siguiente estructura:

- Para abrir una base de datos:

DATABASE base_datos [EXCLUSIVE]

Parámetro	Significado
base_datos	Identificador CTSQL que se corresponde con el nombre de la base de datos
EXCLUSIVE	Palabra reservada que provoca el bloqueo de la base de datos para el resto de usuarios

- Para cerrarla:

CLOSE DATABASE

Ambas instrucciones no distinguen entre bases de datos transaccionales y no transaccionales.

En instalaciones cliente-servidor, la instrucción DATABASE busca por defecto la base de datos a seleccionar en UNIX (base_datos) en el directorio en curso, mientras que en el caso del Windows lo hace en el directorio de trabajo especificado en las propiedades del icono.

No obstante, si hubiésemos definido la variable de entorno DBPATH, la instrucción DATABASE buscará la base de datos en cualquiera de los directorios especificados en dicha variable. Esta búsqueda en los paths de la variable se realiza de izquierda a derecha.

Inserción de filas

CTSQL permite la inserción de valores sobre todas las columnas de una tabla o bien de parte de ellas. Esta facilidad de manejo de datos es estándar de cualquier SQL del mercado. Cuando una tabla está vacía, todas las filas que se graben a continuación se insertarán físicamente de forma secuencial en el fichero de datos (.dat), mientras que el fichero de índices (.idx) se irá actualizando automáticamente por cada una de ellas.

Como característica fundamental del CTSQL, el manejo de datos puede realizarse de fila en fila o bien en un bloque indeterminado de filas (tabla derivada). La instrucción que permite la inserción de filas en las tablas es INSERT. Esta instrucción tiene dos posibles sintaxis:

- Inserción de una sola fila:

  INSERT INTO tabla [(columnas)] VALUES (valores)

- Inserción de un bloque indeterminado de filas (tabla derivada):

  INSERT INTO tabla [(columnas)] instrucción_select

Borrado de filas

El borrado de filas en CTSQL consiste en marcar físicamente aquellas filas que cumplan las condiciones impuestas por la instrucción DELETE, cuya sintaxis es la siguiente:

```
DELETE FROM tabla
        [WHERE {condicion_comparación | condición_subselect} ]
```

Modificación de filas

CTSQL permite la modificación de todos los valores de todas las columnas de una tabla o bien de parte de ellas. La modificación de los valores de las columnas que componen una tabla se realiza mediante la ejecución de la instrucción UPDATE. En caso de no incluir condiciones, una instrucción UPDATE actualizará todas las filas que contenga la tabla. La sintaxis de esta instrucción es la siguiente:

```
UPDATE tabla SET columna = expresión
        [, columna = expresión] [,…]
        [WHERE {condicion_comparación | condición_subselect} ]
```

9. Transacciones

Una transacción es el conjunto de operaciones relativas a recuperación y actualización sobre la base de datos que se realizan de forma unitaria e indivisible, esto es, o se realizan totalmente o no se realizan. En otras palabras, es el modo de agrupar varias instrucciones DML (INSERT, DELETE y UPDATE) y asegurar que dicho grupo se ejecute completamente.

Para poder contemplar transacciones en una base de datos, ésta tiene que ser una base de datos transaccional. En los siguientes epígrafes se indican las operaciones necesarias para crear una base de datos transaccional y, además, la forma de convertir una base de datos que no es transaccional en transaccional y viceversa.

Creación de una Base de Datos Transaccional

Para crear una base de datos transaccional hay que emplear la instrucción CREATE DATABASE del CTSQL con la opción WITH LOG IN, que indica el fichero transaccional. La sintaxis de la instrucción es la siguiente:

```
CREATE DATABASE base_datos [WITH LOG IN "fichero_log"]
        [COLLATING "fichero_ordenación"]
```

Además del directorio que representa la base de datos (.dbs), en el directorio en curso o en cualquiera de los indicados en la variable de entorno DBPATH se generará el fichero especificado en fichero_log. El nombre de fichero_log tiene que incluir el path completo. Este fichero_log no tiene formato ASCII, por lo que NUNCA deberá ser editado. La única forma de manejar su información es mediante las instrucciones del CTSQL que se comentan en este capítulo.

Convertir una Base de Datos en Transaccional y viceversa

Para convertir en transaccional una base de datos que no lo es hay que emplear la instrucción START DATABASE del CTSQL, cuya sintaxis es la siguiente:

```
START DATABASE base_datos WITH LOG IN "fichero_log"
```

Al igual que sucedía en la instrucción CREATE DATABASE, el nombre del fichero transaccional (fichero_log) tendrá que incluir el path completo donde se ubicará.

Esta instrucción sólo puede realizarla el propietario de la base de datos, o bien el usuario que tenga permiso DBA sobre ella. Asimismo, esta operación se tiene que realizar con la base de datos cerrada y sin que nadie la tenga seleccionada. Por lo tanto, antes de ejecutar esta operación habrá que especificar la instrucción CLOSE DATABASE:

```
CLOSE DATABASE
```

En caso de que deseásemos convertir una base de datos transaccional en no transaccional habría que ejecutar igualmente la instrucción START DATABASE, pero con el nombre del fichero transaccional vacío:

```
START DATABASE base_datos WITH LOG IN ""
```

Cambiar de Fichero Transaccional

Como ya hemos comentado anteriormente, la instrucción START DATABASE convierte en transaccional una base de datos que no lo es y viceversa. Del mismo modo, esta instrucción se emplea también para cambiar de fichero transaccional. Estos ficheros transaccionales crecen con bastante facilidad, por lo que se recomienda cambiar de fichero transaccional cuando éste ocupe un espacio considerable en el disco (por ejemplo, cada vez que se realice una copia de seguridad).

Control de Transacciones

Una vez que la base de datos seleccionada dispone de un fichero transaccional es posible controlar las transacciones. Una transacción consiste en controlar un determinado proceso, de tal forma que, en el hipotético caso de que éste finalizase de manera anómala, tuviésemos la posibilidad de deshacer dicha operación. Para ello hay que marcar explícitamente el principio y el fin de dicho proceso (grupo de instrucciones) con las siguientes instrucciones del CTSQL:

BEGIN WORK	Inicia la transacción.
COMMIT WORK	Finaliza la transacción haciendo definitivos los cambios realizados por el proceso en cuestión.
ROLLBACK WORK	Finaliza la transacción deshaciendo los cambios realizados en la base de datos por dicho proceso.

La consecuencia práctica de estas instrucciones es que en el fichero transaccional (log file) se registrarán tanto el comienzo de la transacción como su finalización. Esta operación permite que el programador decida en base a condiciones de error de determinadas instrucciones la realización o no de las modificaciones incluidas en la transacción como un todo.

Así pues, mediante este mecanismo se puede establecer un nuevo nivel de integridad de datos de las tablas bajo control del programador en operaciones tales como borrado de datos obsoletos de tablas relacionadas por un enlace (join), asegurando además el cumplimiento de todas las instrucciones implicadas.

Recuperacion de Datos

Como ya se ha indicado anteriormente, la instrucción START DATABASE asigna un nuevo fichero transaccional a la base de datos indicada en su sintaxis. Esta operación debe realizarse cada vez que se obtenga una copia de seguridad de la base de datos. Dado que las operaciones de actualización, inserción y borrado sobre las tablas de la base de datos se reflejan sobre dicho fichero transaccional, en caso de pérdida de información en la base de datos aquélla podría recuperarse. La instrucción que se encarga de esta operación es la siguiente:

ROLLFORWARD DATABASE base_datos [TO fichero_resultado]

Mediante la ejecución de esta instrucción sería posible la reconstrucción de la base de datos partiendo de la información registrada en un fichero transaccional sobre una copia de seguridad. La instrucción ROLLFORWARD DATABASE sólo recuperará aquellas transacciones que hayan finalizado con COMMIT WORK y no con ROLLBACK WORK.

La base de datos indicada en esta instrucción (base_datos) tiene que ser transaccional. El fichero indicado en la cláusula TO (fichero_resultado) sirve para almacenar el resultado de la ejecución de dicha instrucción.

Capítulo 7
Editor de Configuración

Contenidos

1. Introducción

Como se vio en el capítulo 4, el editor de configuración permite modificar el fichero de configuración “COSMOS.INI”, sin necesidad de recurrir a un editor de textos.

En este capítulo se verán las diferentes partes del editor de configuración, así como las posibles opciones a modificar en el fichero de configuración.

Para acceder al Editor de Configuración existen dos formas:

- Haciendo click sobre el icono

Figura 7. 1. Icono del Editor de Configuración.

- Mediante la instrucción:

```
Consconf[-v][-h][Fichero]
```

Parámetro	Significado
-h	Muestra esta lista de opciones
-v	Muestra la versión del comando, así como su upgrade.
fichero	Nombre del fichero de configuración que se desea editar. Este nombre se indicará con su path completo y su extensión.

2. Fichero de Configuración

El fichero de configuración contiene algunas de las características con las que deberán arrancar las aplicaciones Cosmos. Este fichero se encuentra en el subdirectorio etc del directorio donde se encuentre instalado Cosmos.

Este fichero está especialmente diseñado para:

1. Evitar un número excesivo de parámetros en la línea de comando de las aplicaciones Cosmos.

2. Facilitar el paso de variables de entorno entre aplicaciones.

3. Facilitar el manejo de las aplicaciones Cosmos al evitar al usuario el tener que introducir un número excesivo de datos para poder acceder a ellas.

Cualquier línea de comentario en el fichero de configuración deberá comenzar siempre por un punto y coma (;).

Este fichero está dividido en las secciones que se explican a continuación.

Las variables de entorno que se citan en los ejemplos de las secciones que componen el fichero de configuración se encuentran explicadas en el Anexo D: Variables de Entorno.

Sección [Passwords]

En esta sección se definen los usuarios que pueden acceder a las aplicaciones Cosmos que utilizan el fichero de configuración.
Contendrá una línea y sólo una por cada usuario definido. La sintaxis de un elemento de esta sección es la siguiente:

[Passwords]
usuario=contraseñaEncriptada

Cadena	Significado
Usuario	Nombre con el que se desea identificar al usuario de las aplicaciones Cosmos.
ContraseñaEncriptada	Contraseña del usuario encriptada por el editor de configuracion

En esta sección debe estar siempre definido el usuario system, que será el administrador del sistema.

Ejemplo:

```
[[Passwords]
;dos usuarios definidos: system y Cris
;system tiene contraseña NULL pero tambien ;aparece
encriptada
system=010XQ\~ryh~ptb~
Cris=010hZFxTfrt@rn`
```

Sección [Cosmos]

Esta sección contendrá entradas genéricas para todas las aplicaciones Cosmos. En esta sección se define el usuario que van a utilizar por defecto las aplicaciones Cosmos al arrancar. Dicho usuario tiene que estar definido en la sección [Passwords].

La sintaxis de esta sección es la siguiente:

[Cosmos]
USER=nombre
Licensed Company=nombreEmpresa
Licensed User=nombreUsuario

Cadena	Significado
Nombre	Identifica el nombre del usuario con el que van a arrancar las aplicaciones Cosmos por defecto
nombreEmpresa	Nombre de la empresa que tiene la licencia de explotación de Cosmos
nombreUsuario	Nombre del usuario que utiliza Cosmos

Si esta entrada aparece no se preguntará por el usuario, aunque sí por la contraseña si ésta no es nula.

Ejemplo:

```
[Cosmos]
;usuario por defecto es Cris
USER=Cris
Licensed Company=Personal
Licensed User=B.Cristina Pelayo Garcia-Bustelo
```

Secciones [Environment]

En estas secciones se definen entornos de usuario o de conexión y un entorno global común a todos los usuarios y conexiones.

El valor inicial de las variables de entorno se guardará en el fichero COSMOS.INI en varias secciones. Estas secciones son las siguientes:

[Environment]	Contiene las variables de entorno cuyos valores son comunes a todos los usuarios.
[Environment nombre]	Contiene las variables de entorno definidas para “nombre”.

Cadena	Significado
nombre	- un usuario definido en la sección Passwords - una conexión definida en la sección Connections - nombre arbitrario para definir un entorno independiente.

La sintaxis de estas secciones es la siguiente:

variable=valor

Cadena	Significado
Variable	Nombre de la variable de entorno que se desea definir para el nombre indicado anteriormente.
Valor	Valor que se desea asignar a la variable de entorno (puede ser vacio).

Cada grupo variable=valor puede ocupar una línea como máximo.

Ejemplo:

Entorno GLOBAL:

```
[Environment]
DBTEMP=c:\tmp
DBEDIT=c:\cosmos\bin\cedit
DBPATH=c:\datos\alumnos;c:\datos\asignaturas
TRWPATH=c:\datos\ecd
DBDATE=DMY4/
DBMONEY=,ptas
DBTIME=MMT
CRWPATH=c:\datos\informes
```

Entorno del usuario “Cris”:

```
[Environment Cris]
DBDATE=DMY2
DBTIME=S
```

Entorno de conexión de Conexion1 (los valores en blanco significan que todavia no se han definido):

```
[Environment Conexion1]
DBHOST=
DBUSER=
DBSERVICE=ctsql
DBPATH=
DBNAME=
```

Sección [Connections]

En esta sección se definen las conexiones locales o remotas para facilitar la utilización de otros gestores de base de datos.

Su sintaxis es la siguiente:

[Connections]
nombre=tipo

Cadena	Significado
Nombre	Indica el nombre con el que se desea identificar la conexión.
Tipo = local	Para conexiones locales
Tipo = remote	Para conexiones remotas (cliente-servidor)

Ejemplo:

```
[Connections]
Conexion1=remote
Conexion2=local
```

Sección [Icons]

En esta sección se definen los ficheros de iconos que utilizarán las aplicaciones Cosmos y que han sido creados con el editor de iconos de Cosmos (cosicons.exe).

La sintaxis de un elemento de esta sección es la siguiente:

[Icons]
NombreLógico=nombreFichero.bmp,anchoBoton,altoBotón, colorFondo

Cadena	Significado
NombreLogico	Identificador del fichero de iconos
NombreFichero	Path completo o relativo al subdirectorio etc del directorio donde se encuentre instalado Cosmos
anchoBotón	Ancho en pixels de los botones
AltoBotón	Alto en pixels de los botones
ColorFondo	Puede ser RGB (rojo, verde, azul) o cualquier nombre de color reconocido por MultiBase.

Ejemplo:

```
[Icons]
Small Buttons=tbu16x15.bmp,16,15,RGB(192,192,192)
Mini Buttons=TBUT8X8.BMP,8,8,RGB(192,192,192)
List Icons=listicon.bmp,16,15,RGB(255,255,255)
More List Icons=jcllist.bmp,16,15,RGB(255,255,255)
Icons Office=ICOOFFIC.BMP,32,32,RGB(192,192,192)
Country Flags=ICOFLAGS.BMP,32,20,RGB(192,192,192)
IconMessageSymbols=ICOSIMBO.BMP,32,32, RGB(192,192,192)
Icons Arrows=ICOARROW.BMP,32,32,RGB(192,192,192)
Icon Hardware=ICOHARD.BMP,32,32, RGB(192,192,192)
Windows 95 Icons=IcoWin95.BMP,32,32,RGB(0,128,128)
```

3. Aspecto general del Editor de Configuración

Al ejecutar el editor de configuración se muestra un cuadro de diálogo conteniendo tres fichas:

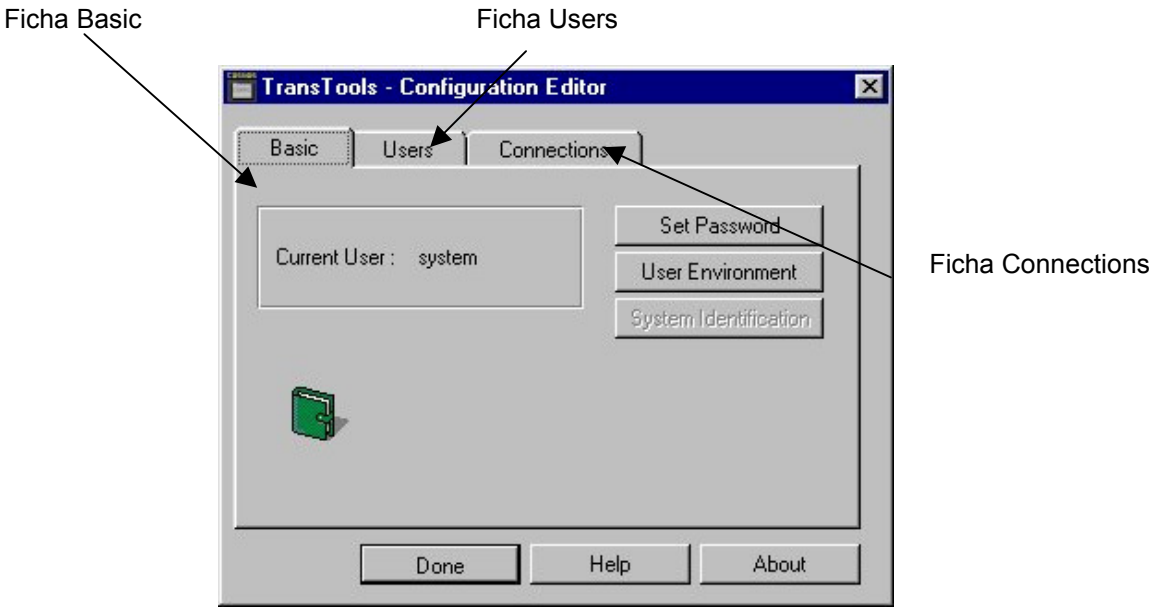


Figura 7. 2. Aspecto del Editor de Configuración

En los siguientes apartados veremos cada una de estas fichas en detalle.

4. Ficha Basic

Estará siempre activa al ser accesible por cualquier usuario. Las acciones que contempla son las siguientes:

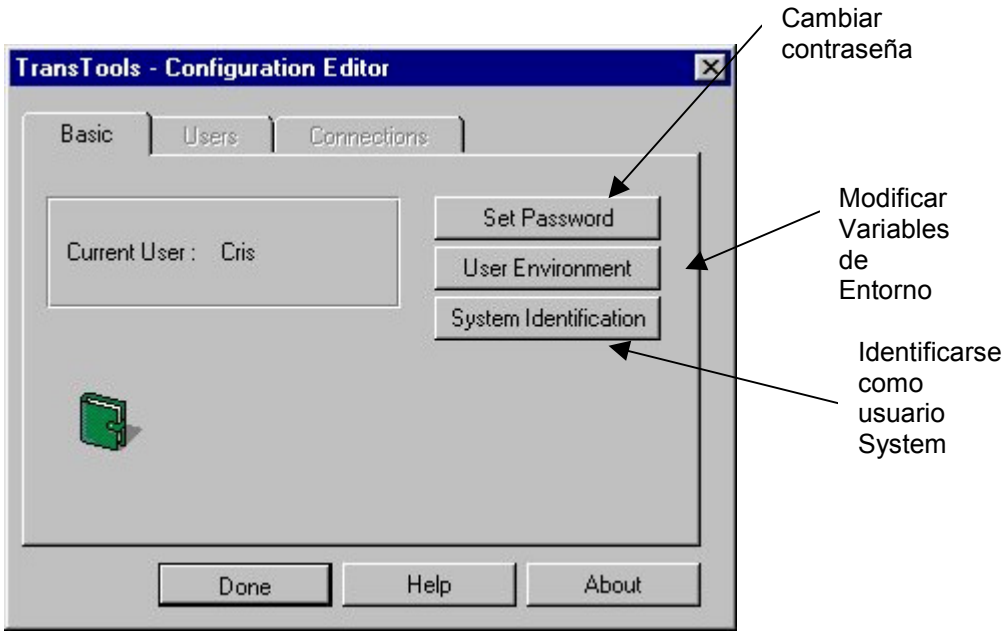


Figura 7. 3. Aspecto de la Ficha Basic

User Environment

Definir, modificar y borrar el valor de las variables definidas en su entorno de usuario.

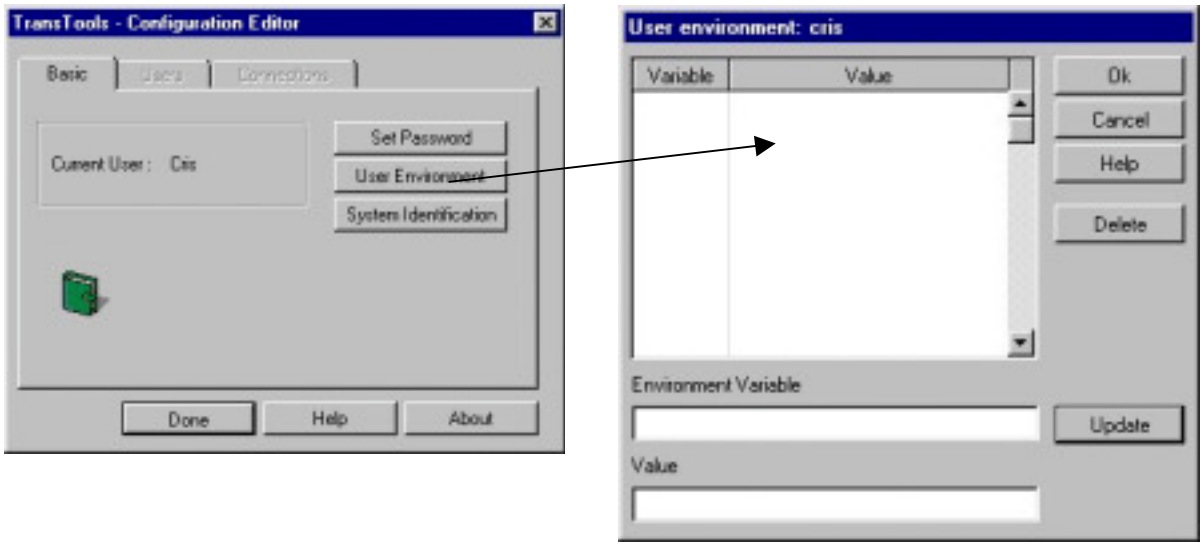


Figura 7. 4. Cambiar las variables de Entorno de un Usuario

Set Password

Modificar su contraseña. Para mayor seguridad pide el nuevo valor dos veces.

Para mayor seguridad el sistema solicita la contraseña que tenía establecida previamente antes de pedir la nueva. El valor de la nueva contraseña se pide dos veces.

Esta opción modifica la sección Passwords del fichero de configuración.

Si un usuario desea modificar su contraseña deberá realizar los siguientes pasos:

1. Pulse el botón Set Password de la ficha Básic. Aparecerá el cuadro de diálogo Change Password.
2. Escriba la contraseña antigua.
3. Escriba el valor de la nueva contraseña y pulse el botón Ok. Si no especifica nueva contraseña, ésta será nula.
4. Si la contraseña antigua indicada es incorrecta se mostrará un mensaje advirtiéndolo y se volverá a la ventana principal de la aplicación sin realizar ninguna modificación.
5. Escriba por segunda vez el valor de la nueva contraseña y pulse el botón Ok.
6. Si los dos valores introducidos para la nueva contraseña son idénticos, ésta quedará modificada, mientras que si son distintos se mostrará un mensaje avisando de esta circunstancia y no se modificará la contraseña.

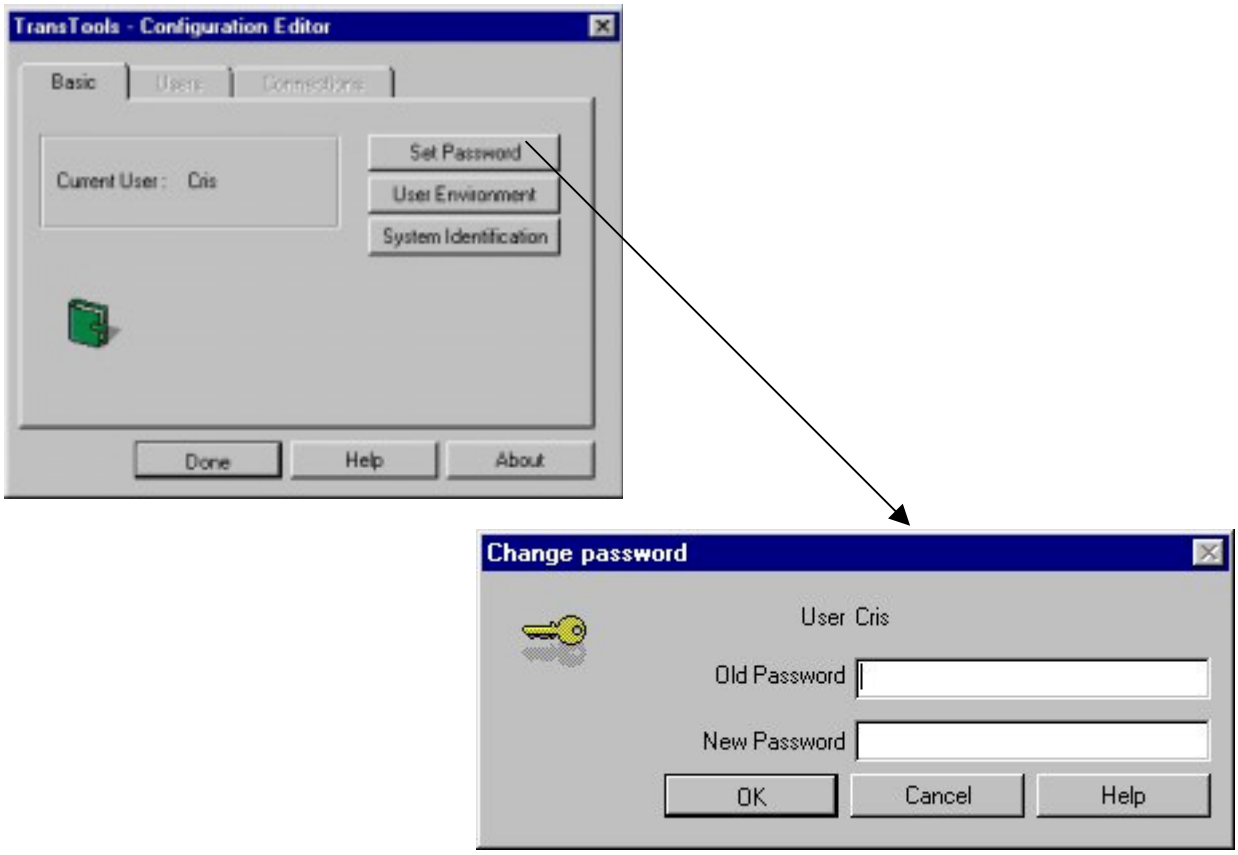


Figura 7. 5. Modificar la contraseña

System Identification

Identificación como usuario system para poder realizar las acciones exclusivas para este usuario conociendo su contraseña.

Para mayor protección y seguridad, algunas de las acciones que se pueden ejecutar en el Editor de Configuración son exclusivas del usuario system.

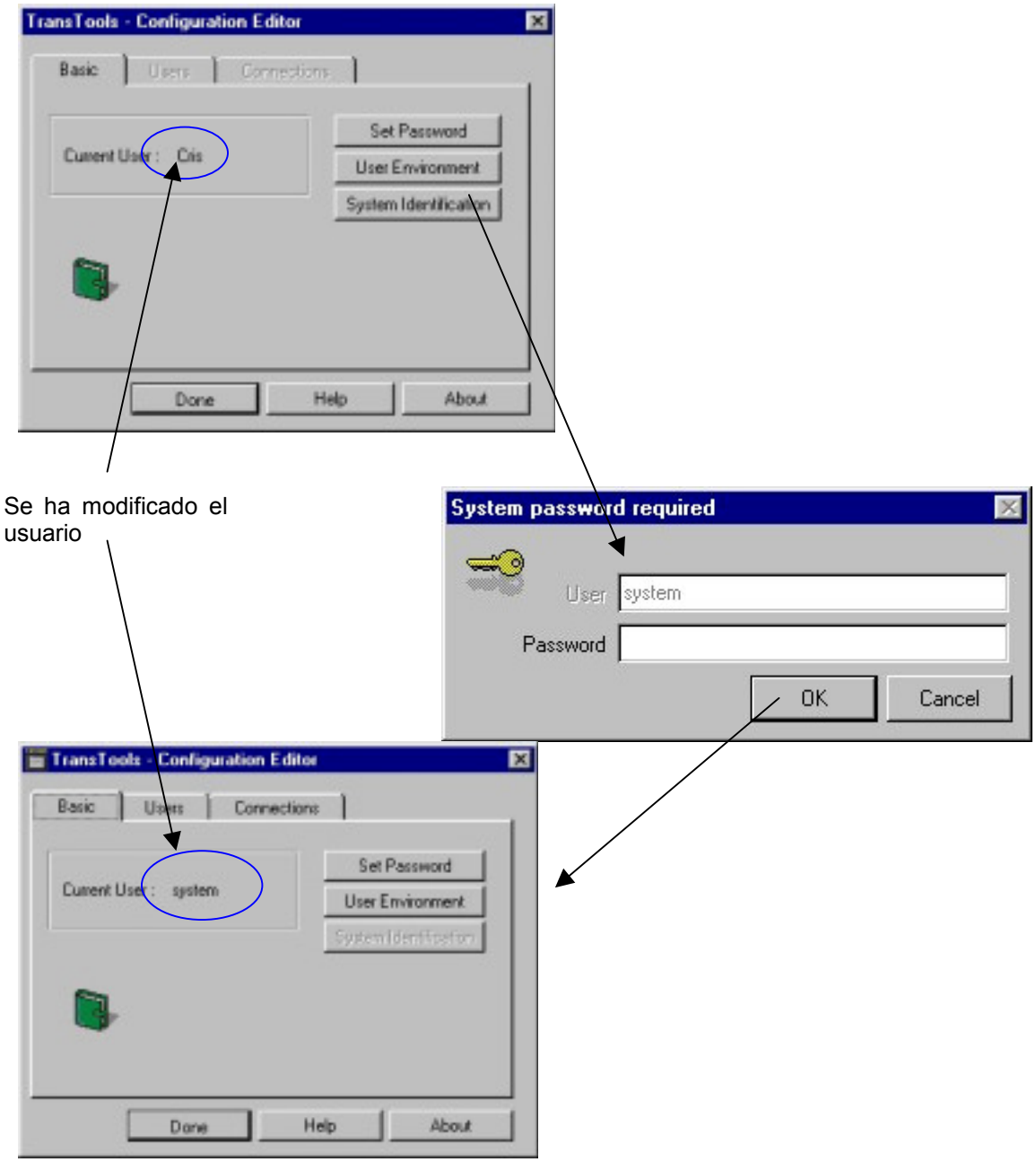


Figura 7. 6. Modificación del Usuario

5. Ficha Users

Solo estará activa para el usuario system. Desde esta ficha se podrán llevar a cabo las siguientes acciones:

- Añadir usuarios al fichero de configuración.
- Borrar usuarios definidos en el fichero de configuración.
- Modificar la contraseña de cualquier usuario, incluida la suya propia.
- Definir el usuario por defecto con el que se ejecutarán las aplicaciones Cosmos.
- Eliminar el usuario definido por defecto.
- Definir o modificar el valor de variables del entorno global, común para todos los usuarios.
- Definir o modificar el entorno definido para un usuario.

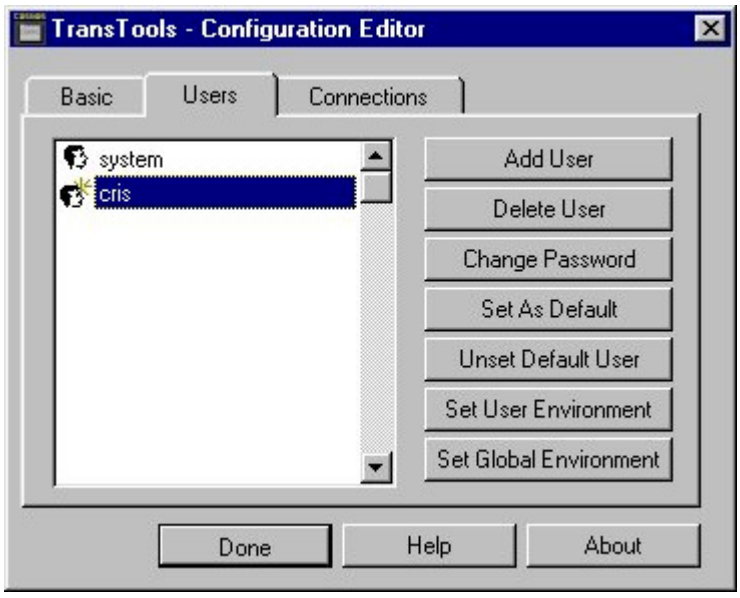


Figura 7. 7. Aspecto de la Ficha Users

Add Users

Esta opción permite añadir nuevos usuarios en la sección passwords del fichero de configuración. Su ejecución sólo puede ser realizada por el usuario system.

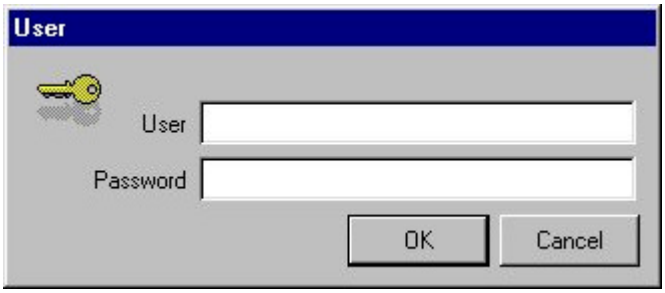


Figura 7. 8. Añadir un usuario

Para añadir un usuario:

1. Introduzca el nombre del usuario y la contraseña, si lo desea.
2. Una vez conforme con los valores introducidos pulse el botón Ok, en caso contrario pulse Cancel.

Delete Users

Esta opción sólo puede ser ejecutada por el usuario system. Su función es borrar de la sección passwords del fichero de configuración el usuario que se tenga seleccionado en ese momento. Antes de proceder al borrado se pide confirmación al usuario.

Al borrar un usuario se borra también su sección de entorno correspondiente.El usuario system es el único que no puede borrarse.

Change Password

El usuario system puede modificar la contraseña de cualquier usuario sin necesidad de conocer para ello la contraseña antigua.

Esta opción modifica la sección Passwords del fichero de configuración. Para mayor seguridad la nueva contraseña se solicita dos veces.

Figura 7. 9. Cambiar la contraseña como usuario system

Para modificar la contraseña de un usuario desde system se deberán realizar los siguientes pasos:

1. Seleccione en la lista de la ficha Users aquel que desee modificar.
2. Pulse el botón Change Password. Se mostrará el cuadro de diálogo Change Password.
3. Escriba la nueva contraseña y pulse el botón Ok. Si no se especifica la nueva contraseña, ésta será nula.
4. Escriba por segunda vez la nueva contraseña y pulse el botón Ok.
5. Si los dos valores introducidos para la nueva contraseña son idénticos, la contraseña quedará modificada.
6. Si los dos valores indicados son distintos el sistema presentará un mensaje avisando de esta circunstancia y no modificará la contraseña.

Los usuarios distintos de system sólo podrán modificar su propia contraseña. Para mayor seguridad el sistema solicita la contraseña que tenía establecida previamente antes de pedir la nueva. Esta opción modifica la sección Passwords del fichero de configuración.

Si un usuario desea modificar su contraseña deberá hacerlo desde la ficha basic.

Set As Default

Esta opción permite establecer cuál el usuario por defecto con el que se ejecutarán las aplicaciones Cosmos. Su ejecución modifica la entrada user de la sección COSMOS del fichero de configuración.

Esta opción sólo puede ser ejecutada por el usuario system.

Para seleccionar un usuario por defecto bastará con seleccionarlo en la lista de la ficha Usuarios y ejecutar esta opción.

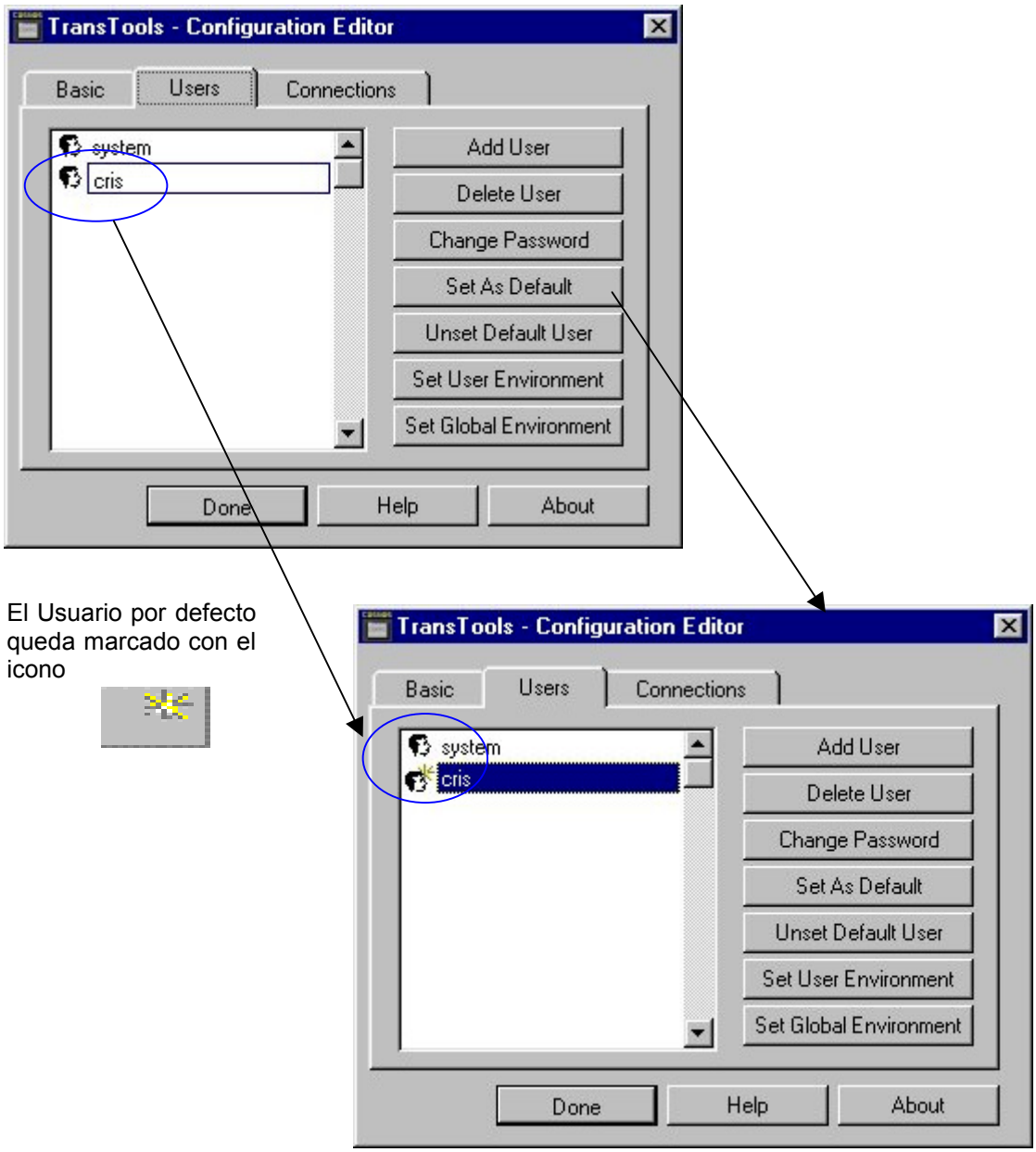


Figura 7. 10. Seleccionar un usuario como usuario por defecto

Si el usuario por defecto tiene definida contraseña nula, al ejecutar una aplicación Cosmos no se le pedirá identificación.

Si no hay usuario definido por defecto, o si tiene definida contraseña no nula, al ejecutar la aplicación se pedirá al usuario que se identifique, pudiendo si lo desea identificarse como otro usuario.

Unset Default User

Al ejecutar esta opción se desactiva el usuario por defecto que estuviese definido en ese momento. Si no hubiese ningún usuario definido por defecto, al arrancar cualquier aplicación Cosmos que necesitase del fichero de configuración se solicitaría al usuario que se identificase con un nombre y una contraseña.

La ejecución de esta opción borra la entrada user de la sección Cosmos del fichero de configuración COSMOS.INI.

Set User Environment

Esta opción permite definir o modificar el entorno de los usuarios definidos en el fichero de configuración.

El usuario system puede modificar el entorno definido para cualquier usuario. Para ello bastará con seleccionarlo en la lista de la ficha Users y hacer doble clic sobre él o pulsar el botón Set User Environment. A continuación aparecerá el cuadro de diálogo de Environment que permite realizar las modificaciones necesarias.

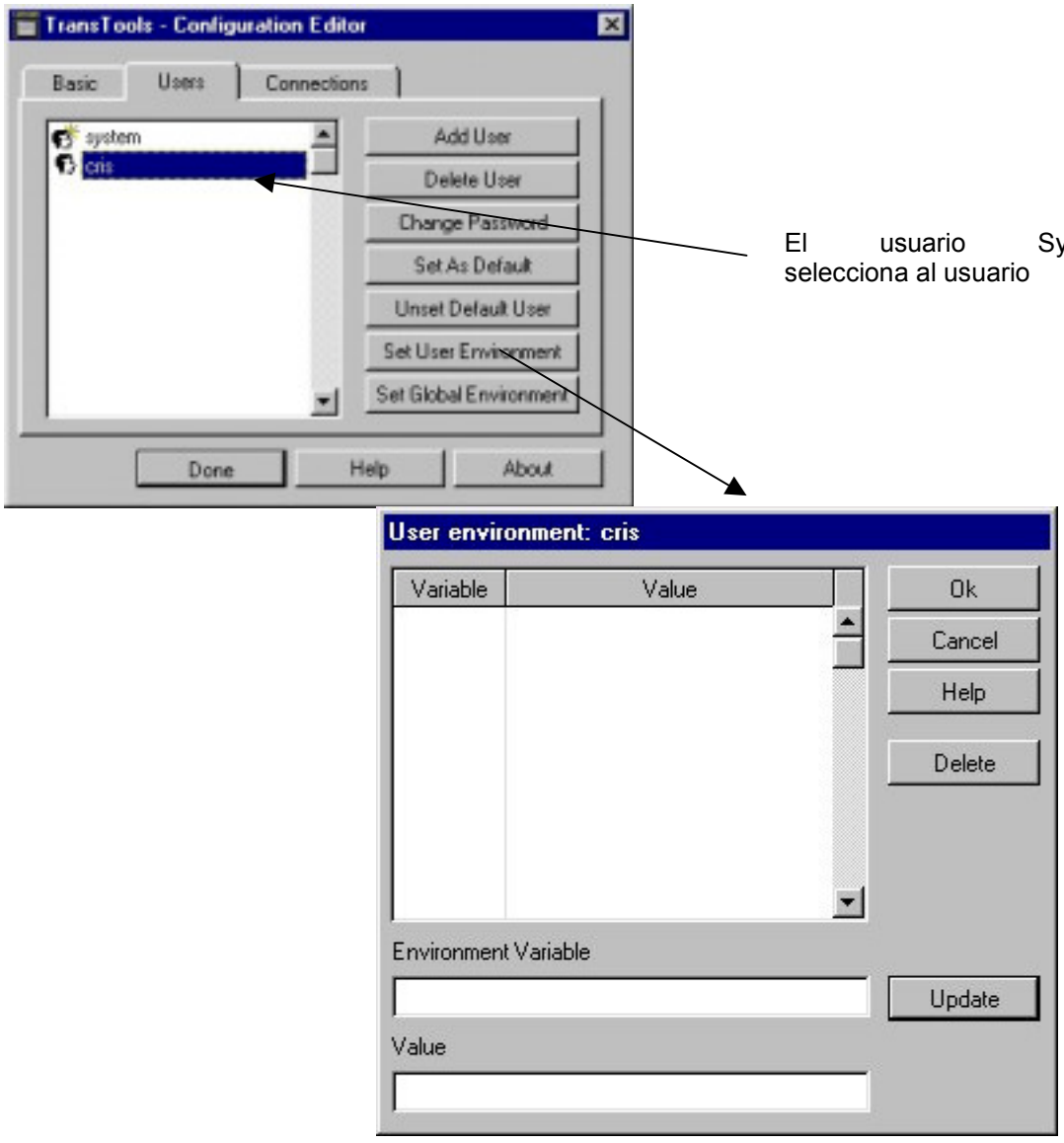


Figura 7. 11. Definición del entorno de usuario

Cuando un usuario distinto de system desee modificar su entorno deberá pulsar el botón User Environment de la ficha Basic.

Las variables definidas en el entorno de usuario quedarán definidas en el entorno de ejecución cuando el usuario se identifique como tal al ejecutar una aplicación que utilice el fichero de configuración o bien cuando este usuario sea el usuario por defecto y tenga contraseña nula.

Las variables aquí definidas se antepondrán a las definidas en el entorno global.

Los entornos aquí definidos podrán ser usados por las aplicaciones que utilizan el fichero de configuración con independencia del usuario en curso. Cuando esté ejecutando una aplicación que use el fichero de configuración, al ejecutar otra aplicación no se pedirá identificación de usuario, definiéndose el entorno del mismo usuario.

Set Global Environment

Esta opción permite definir o modificar el entorno global común a todos los usuarios y conexiones, y sólo podrá ser ejecutada por el usuario system.

Al ejecutarla se mostrará el cuadro de diálogo de Global Environment para realizar las modificaciones necesarias.

Las variables del entorno global estarán siempre definidas al ejecutar cualquier aplicación que utilice el fichero de configuración, pudiéndose definir con otro valor o definir como vacías posteriormente en un entorno de usuario o de conexión

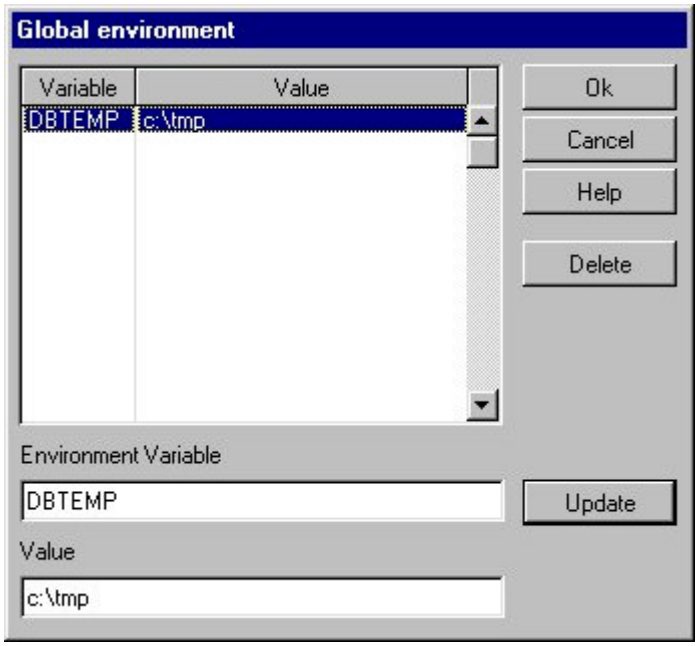


Figura 7. 12. Cuadro de diálogo del Entorno Global

6. Ficha Connections

El Editor de Configuración permite definir conexiones locales, remotas (cliente-servidor) y ODBC para tener acceso a distintos servidores de base de datos. Se da por lo tanto la posibilidad de establecer comunicaciones con otras bases de datos heterogéneas. Esto es posible gracias a la tecnología MultiWay soportada por el producto, que hace que el gestor SQL sea transparente a cualquier aplicación, ya sea ésta del propio Cosmos o bien del de otros fabricantes (Oracle, Informix, Ingres, Sybase, etc.).

Esta ficha unicamente esta activa para el usuario system, puede realizar las siguientes acciones:

1. Añadir nuevas conexiones locales o remotas (cliente-servidor), posibilitando la utilización de otros gestores de bases de datos.
2. Definir entornos de conexión (locales o remotas) para simplificar el manejo de las variables de entorno.
3. Borrar conexiones ya existentes.

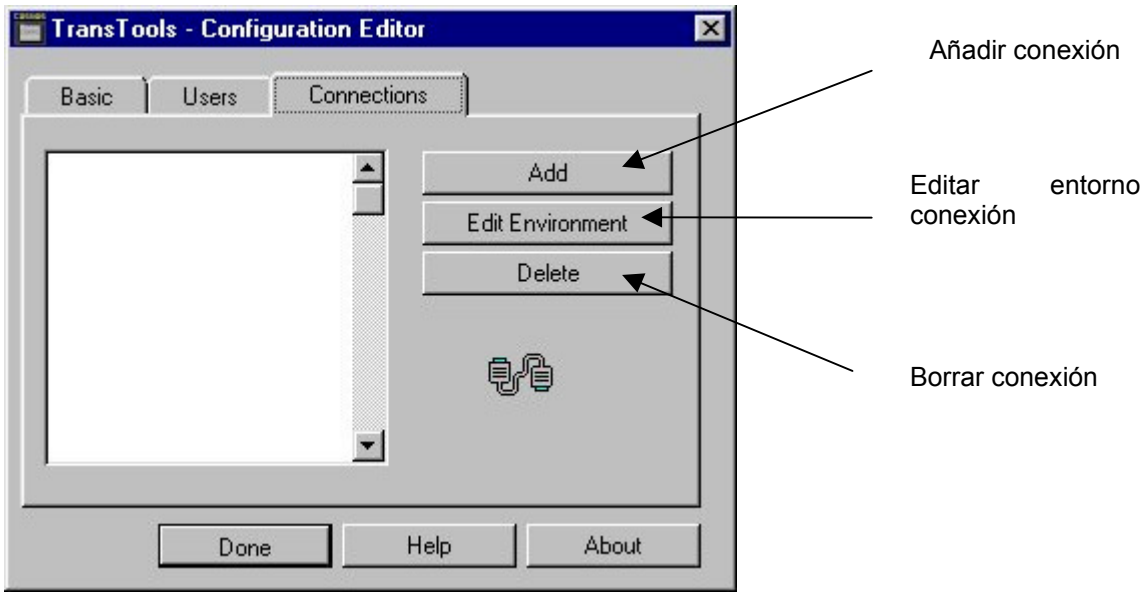


Figura 7. 13. Aspecto de la Ficha Connections

Add

Esta opción permite al usuario system añadir una conexión (local, remota u ODBC) al fichero de configuración. Al ejecutarla se muestra un cuadro de diálogo con los siguientes campos:

Name	Nombre con el que se desea identificar a la conexión. Este campo es obligatorio.
Local	La conexión será local.
MultiWay	Define la conexión Cliente-Servidor (remota). Es el valor por defecto.
ODBC	Define una conexión tipo ODBC.
Server	Permite seleccionar el tipo de servidor que desea utilizar en la conexión. En modo cliente-servidor podrá seleccionarse el gestor propio de TransTOOLs (CTSQL), es el valor por defecto, como el de otros fabricantes (Oracle, Informix, Ingres, etc.).

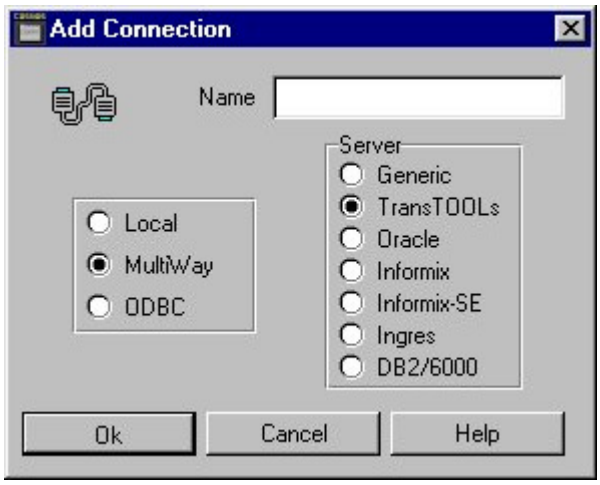


Figura 7. 14. Añadir una nueva conexión

Para añadir una conexión:

1. Escriba el nombre con el que desea identificar la conexión.
2. Elija el tipo de conexión y el servidor (si la conexión es remota).
3. Pulse Ok o Cancel, dependiendo de si está conforme o no con los datos indicados.
4. Si pulsa Ok se mostrará el cuadro de diálogo de Edición de variables de entorno que permite definir el entorno de la nueva conexión. El nombre de la nueva conexión es añadido a la lista de conexiones. Las conexiones locales están representadas con el icono  y las remotas con  .

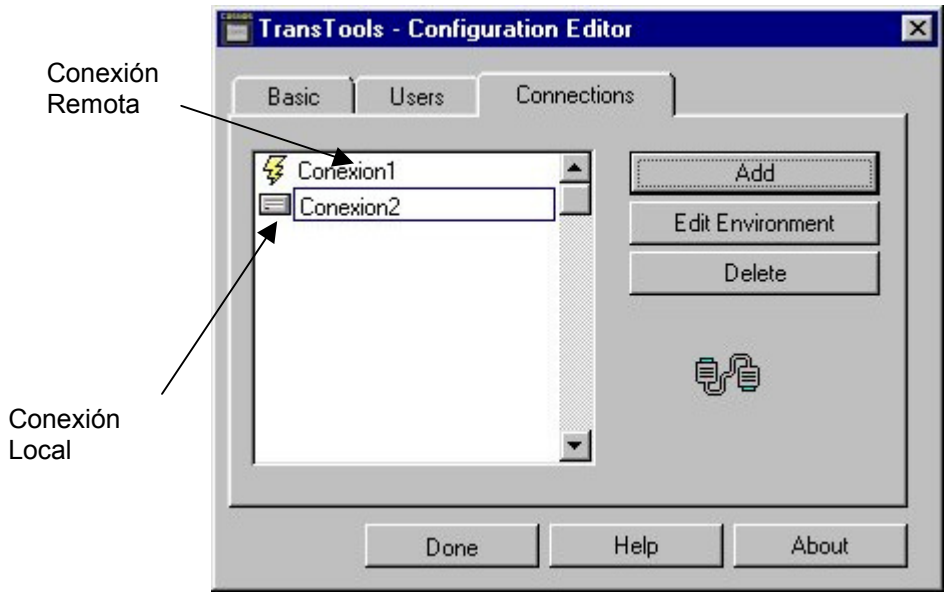


Figura 7. 15. Aspecto de la ficha Connections con 2 conexiones definidas

En el caso de existir una conexión y un usuario con el mismo nombre, el entorno será el mismo para los dos. Es decir, sólo existirá una sección Environment con dicho nombre en el fichero de configuración.

Si se crea una conexión local con el mismo nombre de un usuario, y el entorno de dicho usuario tiene definido la variable DBHOST, automáticamente se cambiará el tipo de la conexión a remota.

Si en el entorno de una conexión local se define la variable DBHOST, automáticamente se cambiará el tipo de la conexión a remota.

Si en el entorno de una conexión remota se elimina la variable DBHOST, automáticamente se cambiará el tipo de la conexión a local.

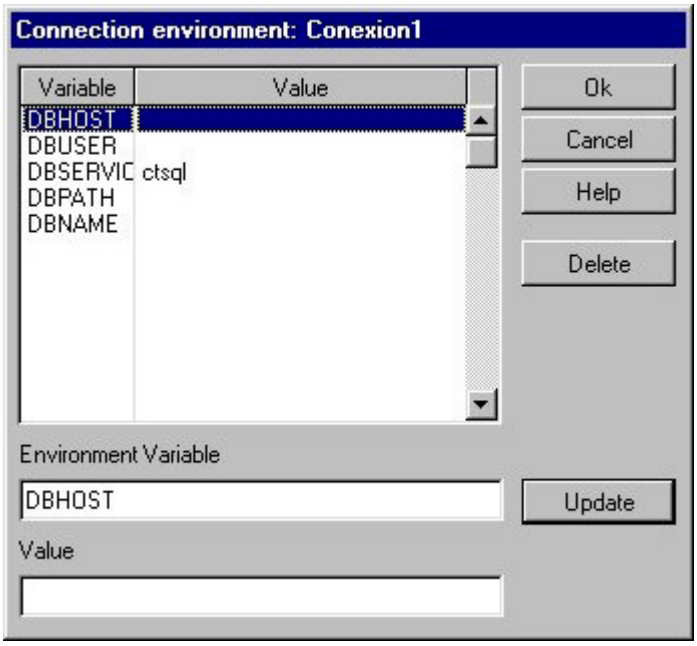
Edit Environment

Esta opción permite editar el entorno de las conexiones definidas en el fichero de configuración, y sólo podrá ser ejecutada por el usuario system.

Una aplicación desarrollada en modo local funcionará en modo cliente-servidor con sólo modificar el valor de una variable de entorno en el cliente.

Cuando se pulsa esta opción teniendo seleccionada una conexión, se muestra el cuadro de diálogo de edición de entorno que permite realizar las modificaciones necesarias.

Los diferentes valores que pueden tomar las variables de entorno dependiendo de las conexiones se encuentran explicados en el Anexo Conexiones.



Figrua 16. Entorno de la conexión remota Conexion1

Delete

Esta opción permite borrar conexiones definidas en el fichero de configuración y sólo puede ser ejecutada por el usuario system.

Para borrar una conexión bastará con seleccionarla en la lista de conexiones y pulsar el botón Delete. Antes de proceder al borrado se pedirá conformidad al usuario.

Cuando se borra una conexión se borra también su entorno asociado, salvo que éste tenga asociado también un usuario.

Capítulo 8
El Editor de Repositorio

Contenidos
1. Introducción
2. Conceptos generales
3. Aspecto general del Administrador de Repositorios
4. Definición y mantenimiento de los Repositorios
5. Comunicación con el SQL y con Bases de Datos
6. Ficheros generados y utilizados por el Administrador de Repositorios

1. Introducción

El primer paso para desarrollar con Cosmos una aplicación que necesite acceder a una base de datos será crear un repositorio para dicha base de datos.

Un repositorio esta formado por el conjunto de datos necesarios para definir el esquema de una base de datos. Asimismo, un repositorio contiene información valida para el desarrollo de programas y para su generación automática.

Estos repositorios una vez definidos, pueden ser utilizados como paletas dentro del Entorno de Desarrollo de Cosmos.

Cosmos dispone de un Editor de Repositorios (*Cosmos Repository Manager)* que automatiza el diseño de estructuras de datos.

En este capítulo se explicará todos los temas relativos a los repositorios y al editor de repositorios.

2. Conceptos generales

Una tabla base es un conjunto de columnas no necesariamente distintas que representa una entidad de la base de datos. Una tabla se define con un nombre único y con la definición de sus correspondientes columnas.

Por su parte un grupo está compuesto por un conjunto de tablas y plantillas de características similares para facilitar la legibilidad y el diseño del repositorio. Los grupos no se guardan el la base de datos, solo se guardan en el repositorio.

Por último las plantillas pueden definirse para facilitar el diseño del repositorio o para definir estructuras de datos que no aparezcan en la base de datos. En una plantilla se puede agrupar un conjunto de columnas y/o estructuras. Las plantillas son tablas que no se guardan en la base de datos, solo se guardan en el repositorio.

En una plantilla se pueden definir claves primarias y claves referenciales al igual que en una tabla.

Las plantillas son especialmente útiles para:

1. Definir un contenedor de columnas tipo que se referencien desde otras partes del repositorio. Por ejemplo si se define una plantilla con una columna “codpost” de tipo integer que guarde el código postal, todas las tablas del repositorio que necesiten el código postal tendrán una columna que enlace con la columna “codpost” de la plantilla.
2. Definir un contenedor de columnas que no se desea que pertenezcan a la base de datos.
3. Definir tablas temporales que se crearán solo en tiempo de ejecución.
4. Definir tablas de otras bases de datos a las que se va a tener acceso solo en tiempo de ejecución.

3. Aspecto general del Administrador de Repositorios

El Administrador de Repositorios permite la definición del tipo de interfaz gráfico para cada una de las columnas de la base de datos: En el momento de arrastrar una columna desde una paleta del repositorio hasta un diálogo, esta columna aparecerá en el diálogo con el interfaz gráfico que se haya definido.

El Administrador de Repositorios de Cosmos automatiza el diseño de estructuras de bases de datos e incluye todas las opciones necesarias para la definición y el mantenimiento de los repositorios :

1. Definir la estructura de un repositorio.
2. Crear una base de datos a partir de la información contenida en el repositorio.
3. Crear un repositorio a partir del diccionario de una base de datos.
4. Importar datos de una base de datos o de otro repositorio.
5. Actualizar un repositorio con los datos de una BD o viceversa.
6. Generar un procedimiento SQL de creación de las tablas del repositorio para permitir la regeneración de éstas en otra instalación o bien como medida de seguridad del estado de la estructura del repositorio en un momento determinado.

Para invocar al editor de Repositorios existen dos formas:

- Haciendo doble click en el icono correspondiente:

cosrep

- Mediante el comando:

```
Cosrep[-h][-v][nombreRepositorio.crf]
```

Opciones	Significado
-h	Muestra en una ventana los parámetros de la línea de comandos.
-v	Muestra la versión del editor de Repositorio
nombreRepositorio.crf	Nombre del Repositorio con el que se desea trabajar

Ventana Principal

La ventana principal del Administrador de Repositorios presenta los siguientes elementos:

1. Una estructura en árbol del repositorio en edición que contendrá todo el conjunto de grupos, tablas y plantillas incluidos en el repositorio. Al iniciar la aplicación el repositorio estará vacío.
2. Un conjunto de botones que permiten realizar acciones sobre la tabla o plantilla que tenga seleccionada en la lista.
3. Una barra de estado donde se muestra la etiqueta asociada a la tabla, plantilla o grupo seleccionado.

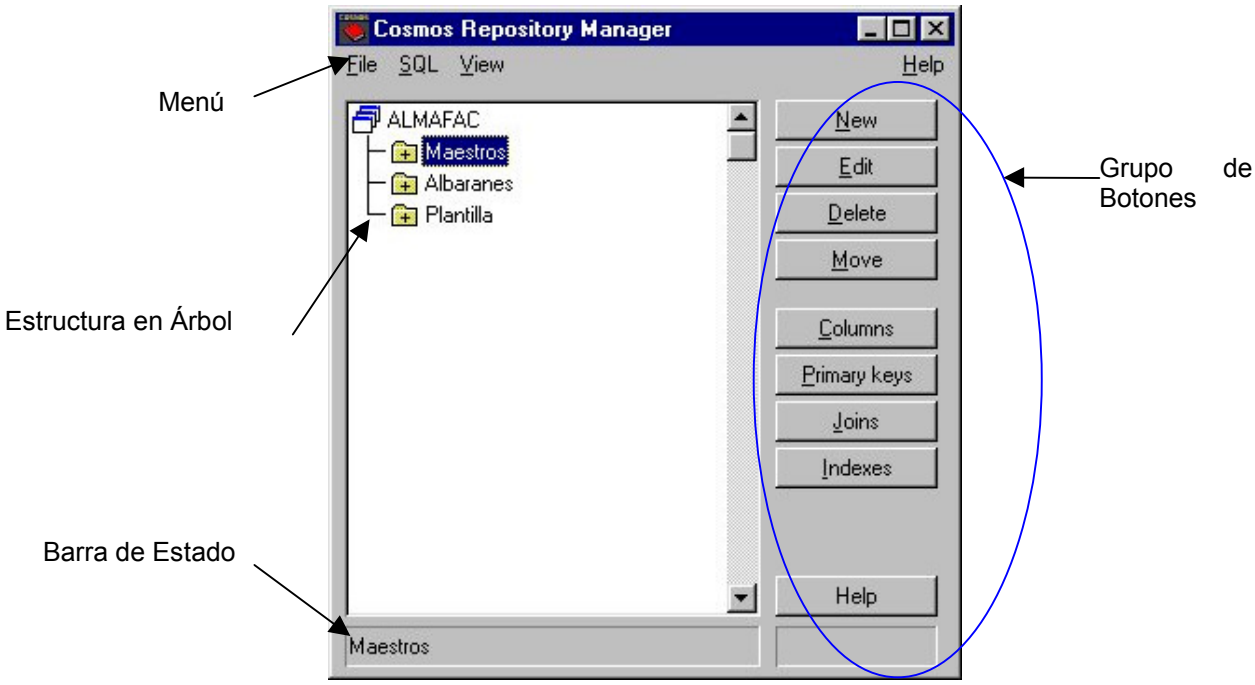


Figura 8.1. Aspecto del Administrador de Repositorios

Menú

Permite realizar todas las funciones básicas del Administrador de Repositorios.

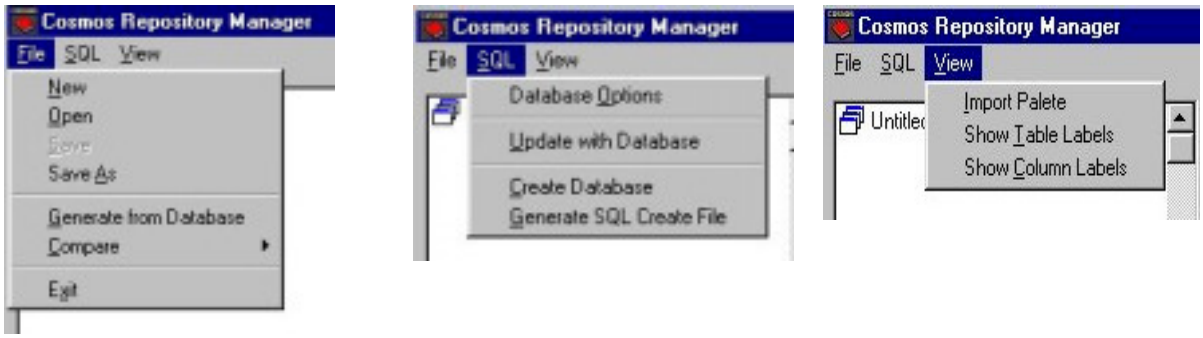


Figura 8.2. Aspecto de los elementos del Menú

Estructura en Árbol

Es posible controlar la cantidad de información presentada en dicha lista. Se puede expandir toda una rama del árbol , o solo un nivel de la rama, o todas las ramas.

Para expandir un grupo :

1. Seleccione en la lista uno de los grupos marcados con el icono .
2. Haga doble clic sobre el grupo, se mostrará marcado con el icono y podrá ver su contenido.

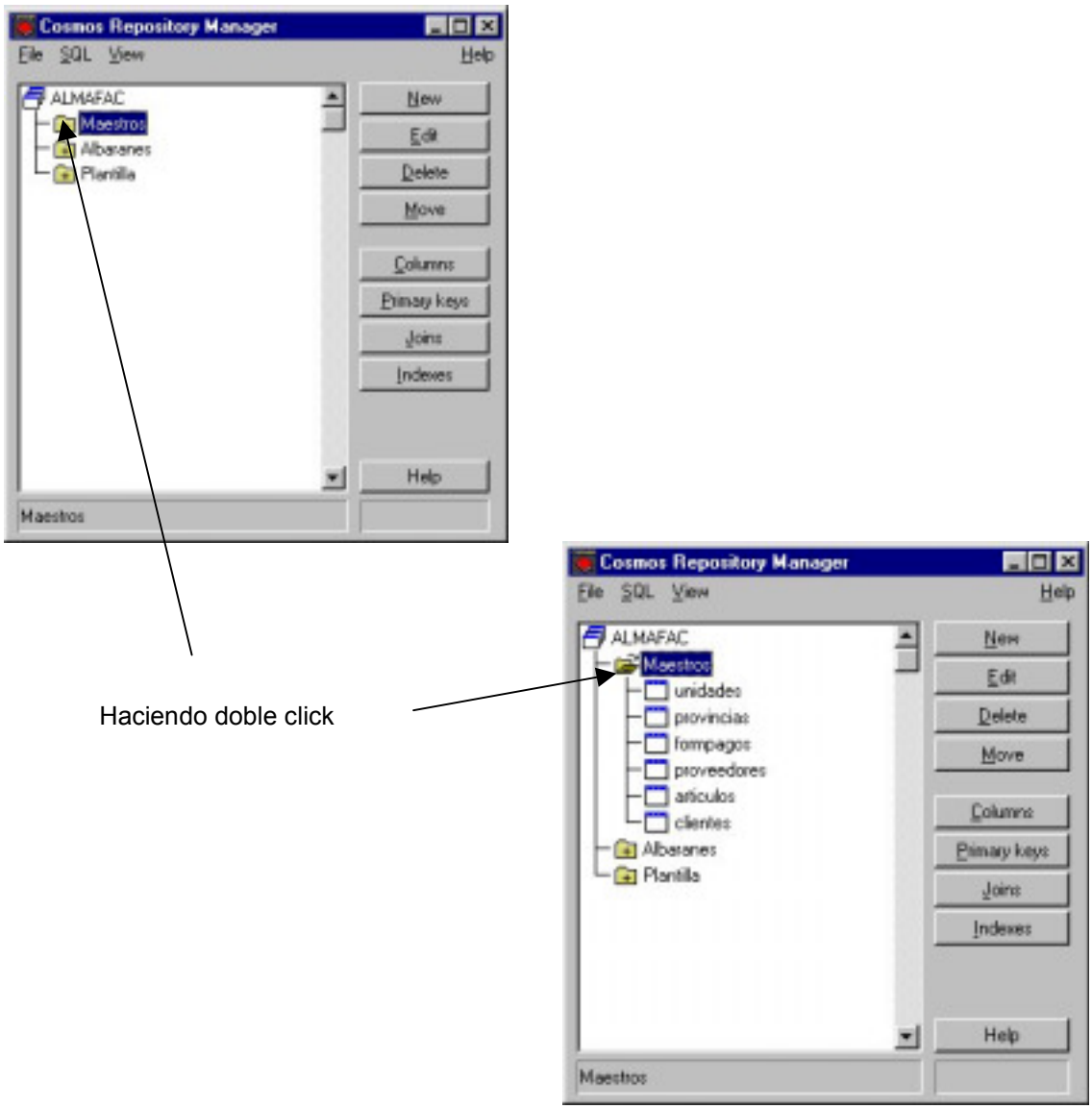


Figura 8.3. Forma de expandir un grupo

Para contraer un grupo de modo que no aparezca su contenido:

1. Seleccione en la lista uno de los grupos marcados con el icono .
2. Haga doble clic sobre el grupo, se mostrará marcado con el icono y no se mostrará su contenido.

Las tablas se muestran marcadas con el icono ,las plantillas con el icono y los grupos con o con dependiendo si están desplegados o no.

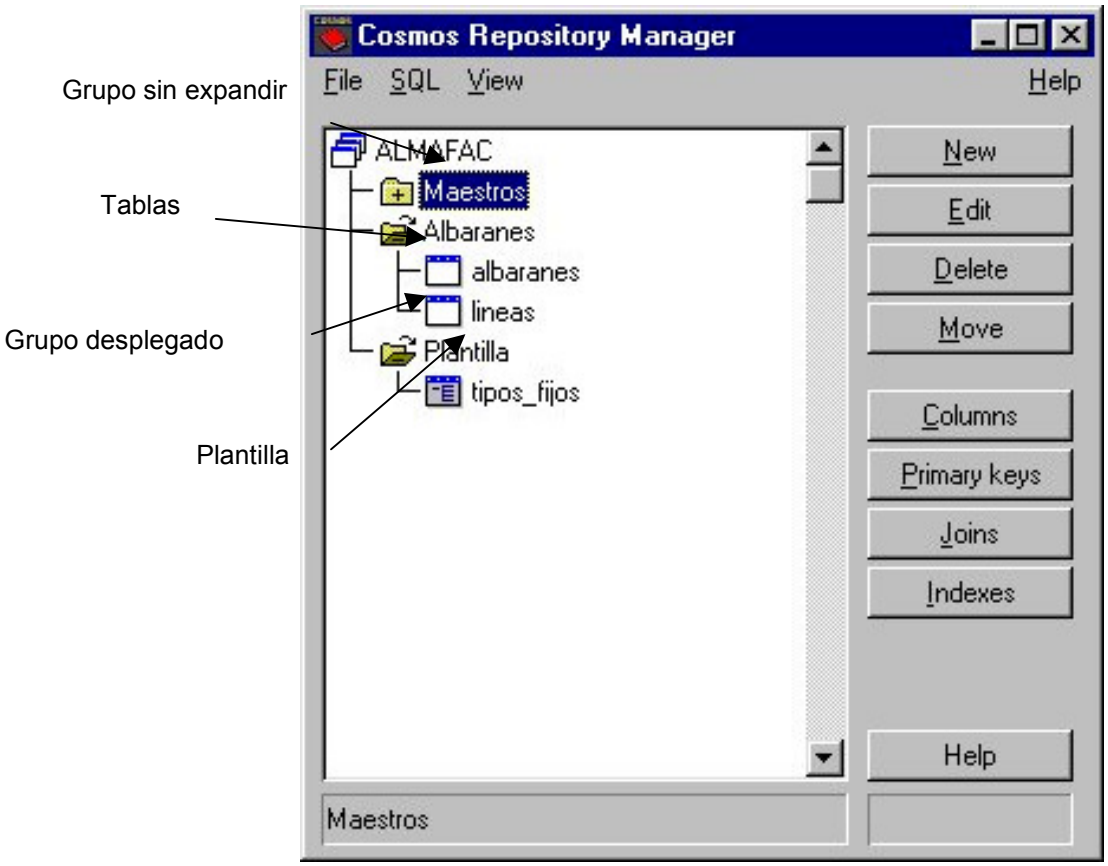


Figura 8.4. Tipo de componentes del Administrador de Repositorio

Si hace doble clic sobre una tabla, automáticamente se muestra el cuadro de diálogo *Columns for Table* que permite añadir nuevas columnas o estructuras a una tabla, estructura o plantilla del repositorio en edición.

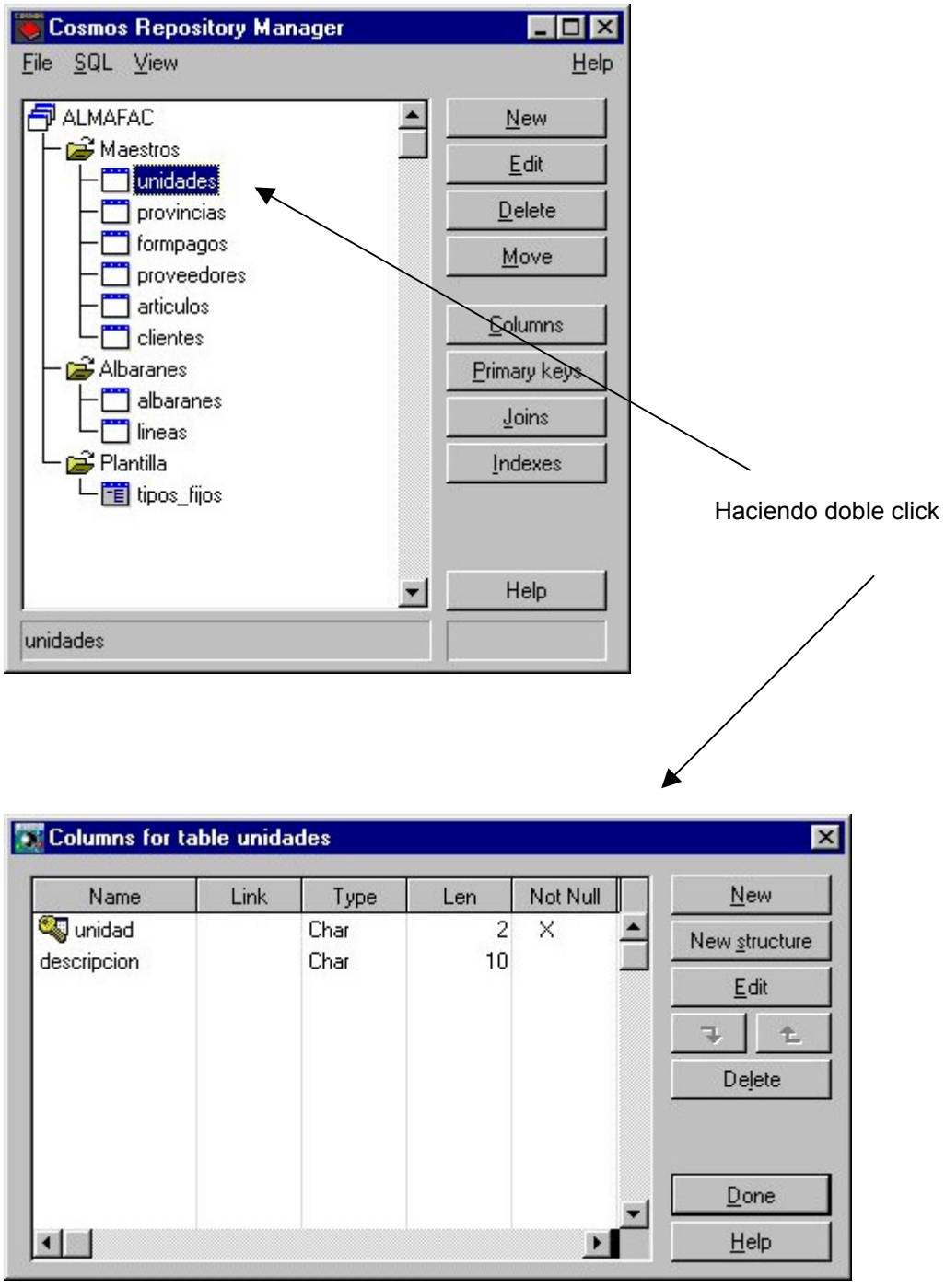
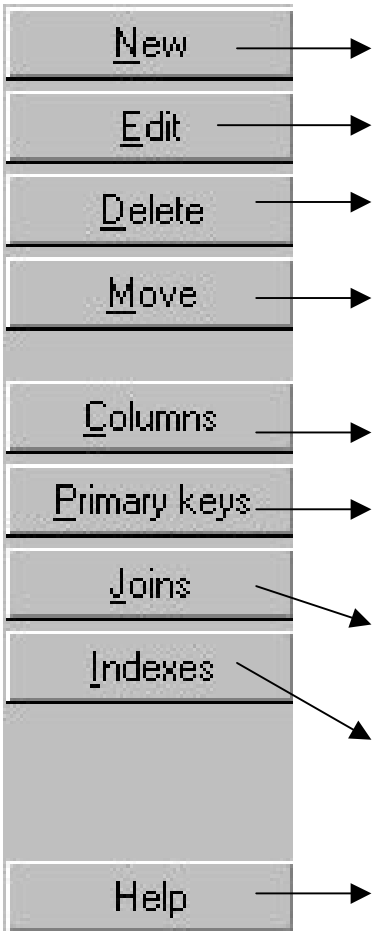


Figura 8.5. Como mostrar el diálogo *Columns for table*

Conjunto de Botones

El conjunto de botones, dentro del Administrador de Repositorios, permiten realizar las siguientes acciones sobre la tabla o plantilla que tenga seleccionada en la lista:



Permite añadir una nueva tabla, grupo o plantilla al repositorio en edición.

Permite modificar los parámetros de definición de un grupo, tabla o plantilla.

Permite borrar tablas, plantillas y grupos del repositorio.

Esta opción permite cambiar a otro grupo, dentro de la estructura en árbol de repositorio, una tabla, una plantilla o un grupo.

Esta opción permite añadir, borrar y modificar columnas o estructuras se una tabla, o plantilla del repositorio en edición.

Una vez definidas todas las tablas de la base de datos, el siguiente paso a realizar será la definición de la clave primaria para cada una de ellas.

Esta opción permite editar, crear, modificar y borrar claves referenciales y joins de la tabla seleccionada.

Esta opción permite editar, crear, modificar y borrar los índices de la tabla seleccionada.

Muestra la Ayuda

Barra de Estado

Se muestra la etiqueta asociada a la tabla, plantilla o grupo seleccionado.

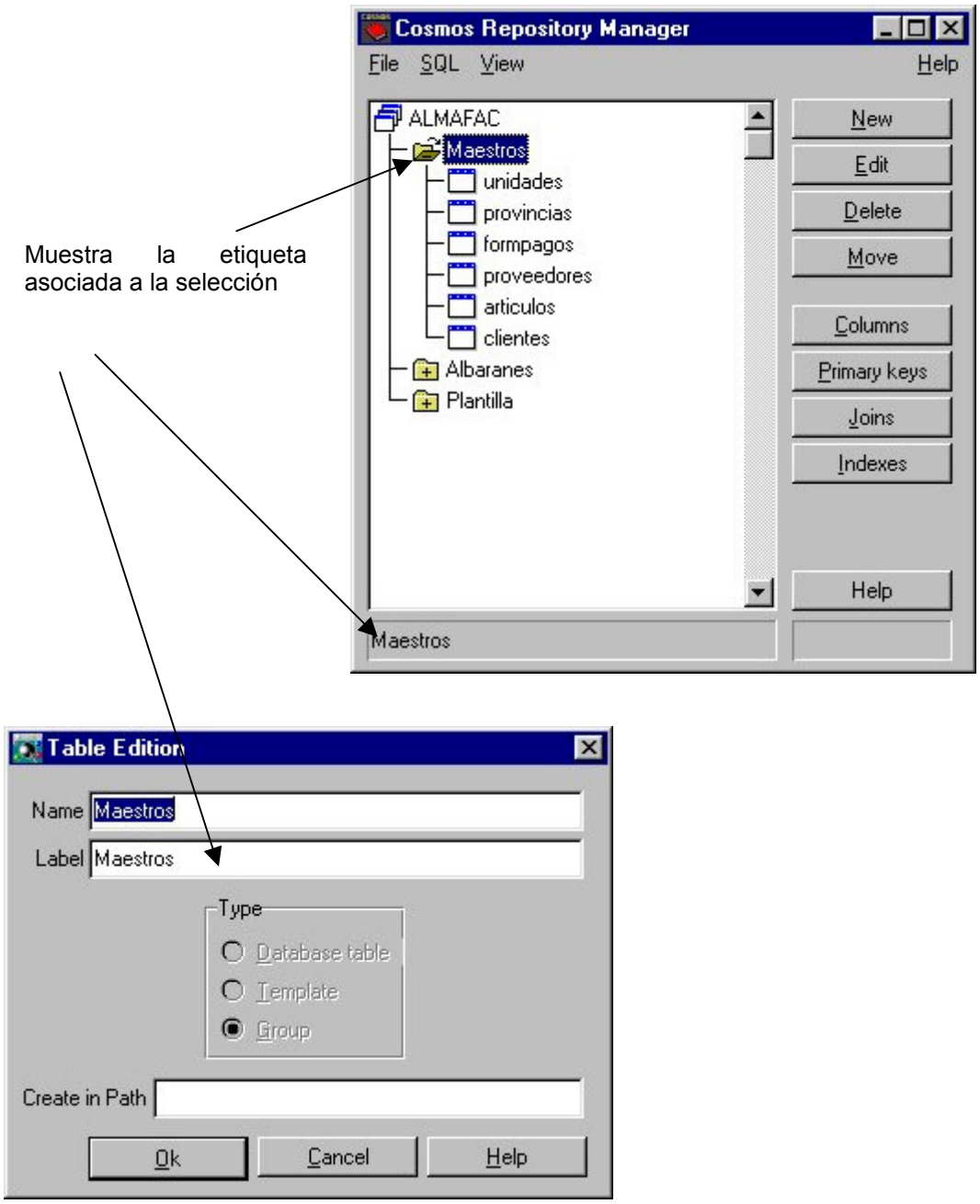


Figura 8.6. Barra de Estado

Configuración del Administrador de Repositorios

El Administrador de Repositorios tiene dos opciones en el menú *View* que permiten configurarlo. Estas opciones afectan a la opción *Import palette* que se verá posteriormente, a la ventana principal del administrador que muestra la estructura en árbol del repositorio.

Show Table Labels	Si se activa esta opción se mostrarán las etiquetas de las tablas, cuando estén definidas, en la estructura en árbol del repositorio, en caso contrario se mostrarán sus nombres.
Show Column Labels	Si se activa esta opción se mostrarán las etiquetas de las columnas, cuando estén definidas, en la estructura en árbol del repositorio, en caso contrario se mostrarán sus nombres.

4. Definición y mantenimiento de los Repositorios

En esta sección se explica detalladamente el procedimiento a seguir para crear un repositorio y para a partir de éste generar la base de datos.

Para crear un repositorio hay que seguir estos pasos:

1. Se Crea una tabla, una plantilla o un grupo.
2. Se edita la tabla, la plantilla o el grupo para hacer alguna modificación si se desea.
3. Se editan las columnas y/o estructuras para las tablas y las plantillas.
4. Se editan las claves primarias.
5. Se editan los índices de las tablas.
6. Se editan las claves referenciales y joins.
7. Se guarda el repositorio.
8. Se crea la base de datos a partir del repositorio.

Crear una tabla, grupo o plantilla (Botón New)

Los pasos a seguir para añadir una tabla, un grupo o una plantilla al repositorio en edición son:

1. Seleccione en la estructura en árbol del repositorio en edición el grupo al cual desea añadir la tabla, el grupo o plantilla.

    - Si no hace ninguna selección, el nuevo elemento se añadirá al final del grupo raíz del repositorio.
    - Si selecciona una tabla o plantilla el nuevo elemento se añadirá al final del grupo al que pertenece.
    - Si selecciona un grupo, el nuevo elemento se añadirá al final del mismo.

2. Pulse el botón *New* de la ventana principal de la aplicación.

3. A continuación se mostrará el cuadro de diálogo *Table Edition*, que permite introducir los datos necesarios para crearla.

4. Para aceptar los datos introducidos pulse el botón *Ok* en el cuadro de diálogo. Automáticamente se mostrará la tabla creada en la estructura en árbol del repositorio.

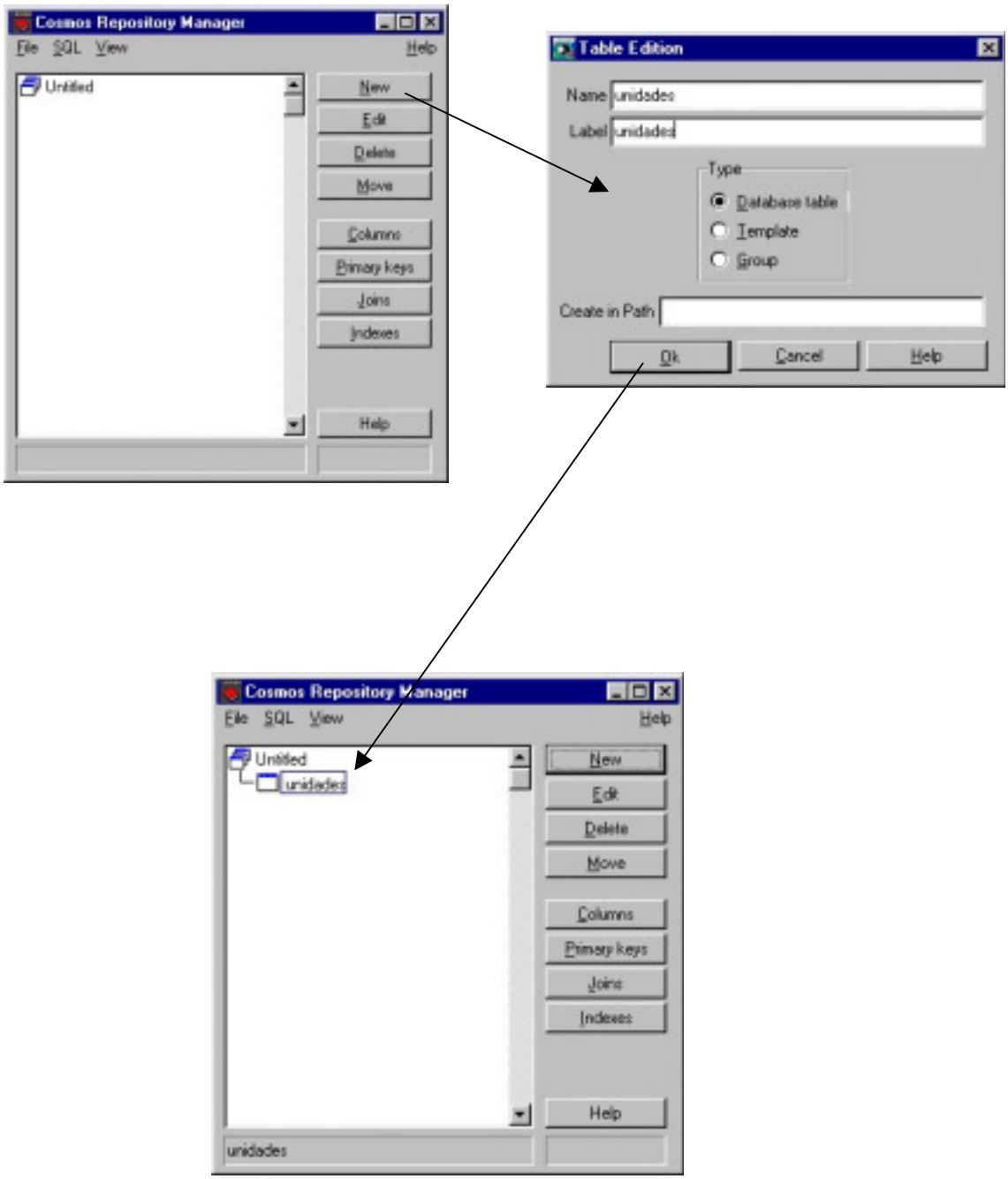
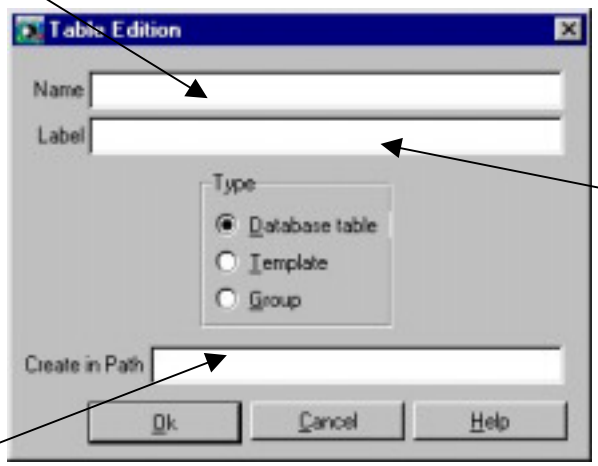


Figura 8.7. Creación de una tabla

Cuadro de diálogo Table Edition

Este cuadro de diálogo permite definir tablas, plantilla y grupos del repositorio. Los campos que se muestran en dicho cuadro de diálogo son los siguientes:

Name En este campo se deberá indicar el nombre de la tabla, plantilla o grupo del repositorio.
Deberemos indicar un nombre que no coincida con el de ninguna tabla, plantilla o grupo existente en el repositorio.



Label: Nombre de 256 caracteres para identificar a la tabla. Este atributo no se guarda en el diccionario de la base de datos, solo se guarda en el repositorio.

Figura 8.8. Cuadro de diálogo *Table Edition*

Create in Path: En este campo se deberá indicar el directorio donde se ubicará la tabla cuando se cree la base de datos. Si no se especifica ningún directorio se creará en el mismo directorio de la base de datos a la que pertenece. Cualquier otro valor ha de corresponder necesariamente con un directorio existente. Use este campo solo cuando sea estrictamente necesario.

El Tipo del elemento que se va a definir en el repositorio que puede ser uno de los siguientes:

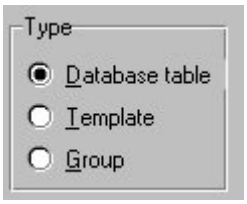


Figura 8.9. Componentes de *Type*

Tipo	Significado
Database table	Tabla de la base de datos. Una tabla de la base de datos es un conjunto de columnas no necesariamente distintas. Una tabla se define con un nombre y la definición de sus correspondientes columnas. Una tabla representa una entidad de la base de datos.
Template	Se permiten definir plantillas para facilitar el diseño del repositorio. Las plantillas no se crearán en la base de datos.
Group	Se permite agrupar varias tablas y plantillas de características similares para facilitar la legibilidad y el diseño del repositorio. Los grupos no se guardan en la base de datos, solo se guardan en el repositorio.

Editar una tabla, grupo o plantilla (Botón Edit)

El Administrador de Repositorios permite modificar los parámetros de definición de un grupo, tabla o plantilla.

Para modificar dichos parámetros proceda de la siguiente manera:

1. Seleccione en la estructura en árbol del repositorio en edición la tabla, grupo o plantilla que desea modificar.
2. Pulse el botón *Edit* de la ventana principal de la aplicación.
3. Se mostrará el cuadro de diálogo de *Table Edition* para poder modificar los parámetros de definición.
4. Realice las modificaciones correspondientes y pulse el botón *Ok* del cuadro de diálogo.

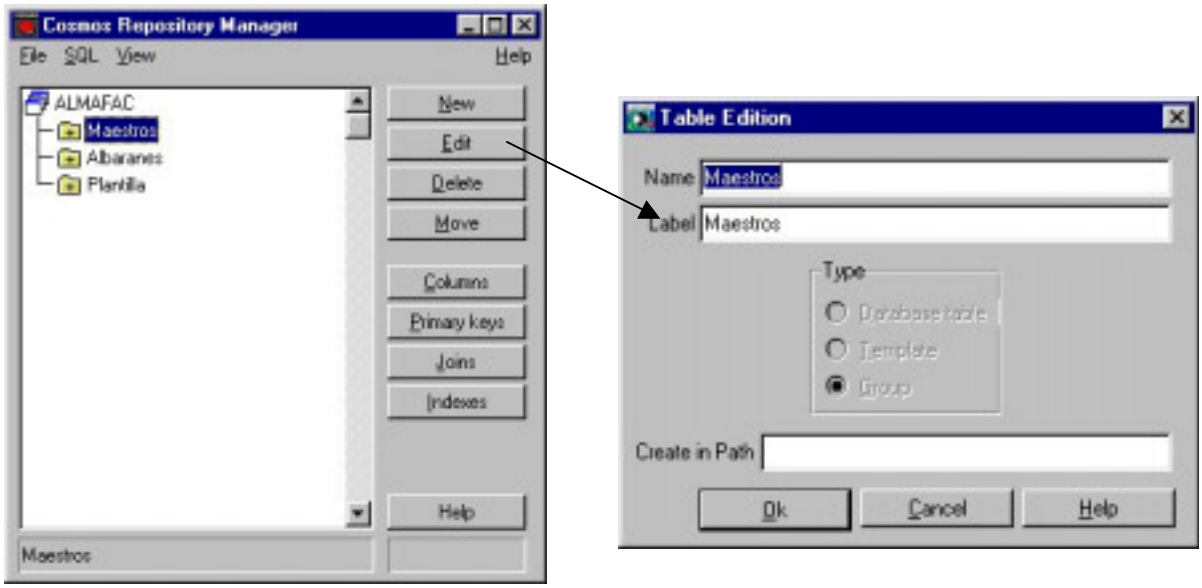


Figura 8.10. Edicción del grupo Maestros

Borrar una tabla, grupo o plantilla (Botón Delete)

El administrador de repositorios permite borrar tablas, plantillas y grupos del repositorio en edición. Antes de proceder al borrado se pide conformidad.

Para borrar una tabla, plantilla o grupo proceda de la siguiente manera:

1. Seleccione en la estructura en árbol del repositorio en edición de la ventana principal la tabla, plantilla o grupo que desee borrar.
2. Pulse el botón *Delete* de la ventana principal del Administrador de repositorios.

Al borrar un grupo se dan dos posibilidades al usuario:

1. Borrar todos los elementos del grupo, sin pedir confirmación.
2. Borrar pidiendo confirmación para cada uno de los elementos del grupo.

Si se intenta borrar una tabla o plantilla referenciada en una foreign key o en un enlace se mostrará el mensaje correspondiente y no se procederá al borrado.

Si se procede al borrado de un elemento que no tiene ninguna referencia de ningún tipo, y se da conformidad para realizarlo, automáticamente dicho elemento desaparecerá de la estructura en árbol del repositorio.

Mover una tabla, grupo o plantilla (Botón Move)

El administrador de repositorios le permite cambiar a otro grupo, dentro de la estructura en árbol del repositorio, una tabla, una plantilla o un grupo.

Para mover una tabla, plantilla o grupo proceda de la siguiente manera:

1. Seleccione en la estructura en árbol del repositorio en edición de la ventana principal la tabla, plantilla o grupo que desee mover.
2. Pulse el botón *Move*, automáticamente se muestra en una ventana la estructura en árbol de los grupos existentes en el repositorio. Seleccione en dicha ventana el grupo al que desea mover el elemento, y haga doble clic sobre el o pulse el botón *Ok*.
3. La tabla, plantilla o grupo seleccionado en el punto uno se habrá movido al final del grupo seleccionado en el punto dos.

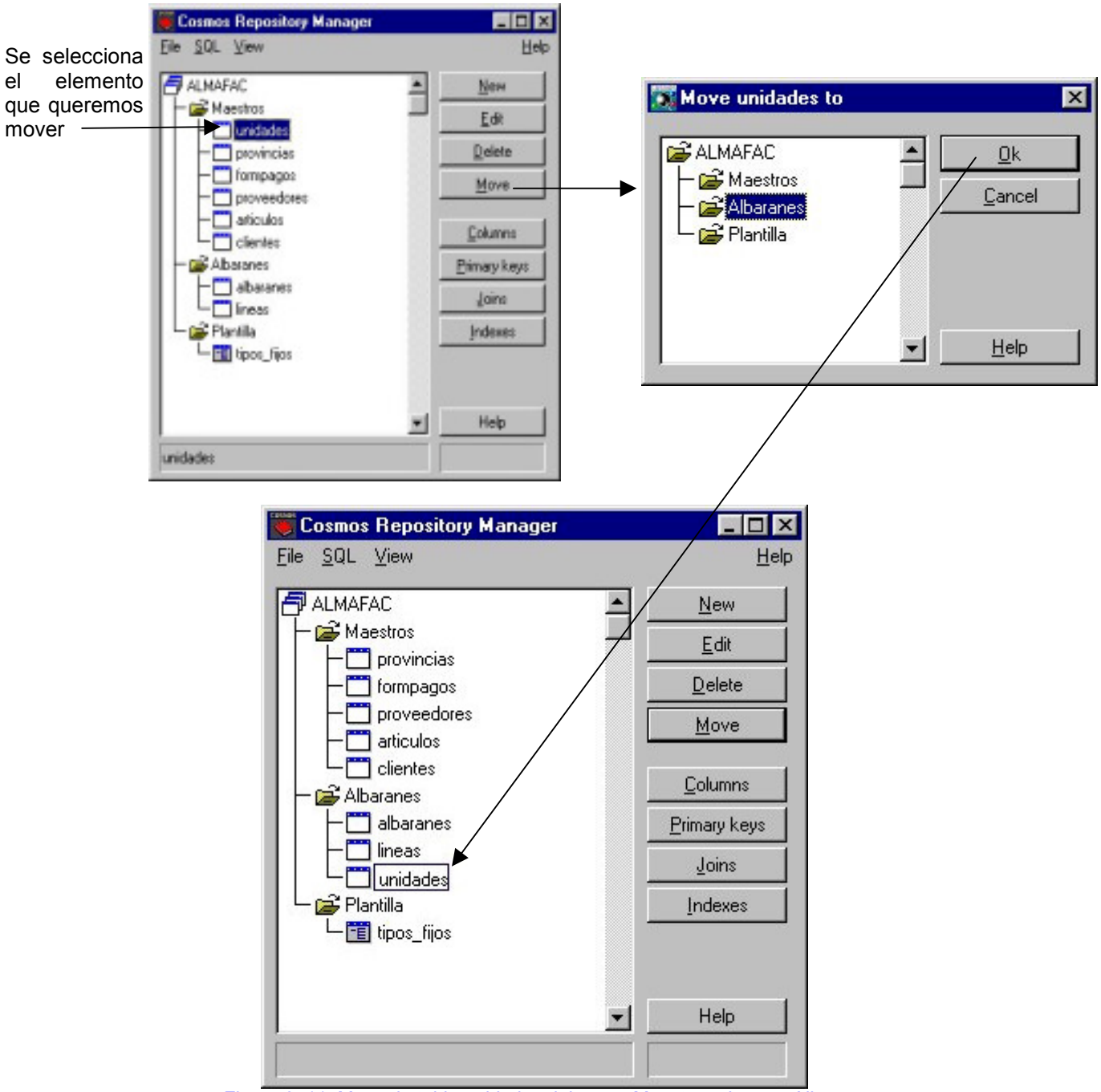


Figura 8. 11. Mover la tabla unidades del grupo Maestros al grupo Albaranes

Edición de Columnas y/o estructuras (Botón Columns)

El paso siguiente a la creación de una tabla y/o plantilla es crear sus columnas y/o estructuras.

En una tabla, estructura o plantilla no pueden existir dos columnas con el mismo nombre.

Para editar columnas y/o estructuras proceda de acuerdo a los siguientes pasos:

1. Seleccione en la estructura en árbol de repositorio la tabla, plantilla o estructura a la que desea añadir una columna o estructura. A continuación pulse el botón *Columns* de la ventana principal de la aplicación.

2. Se muestra un cuadro de diálogo *Columns for Table*. Este cuadro de diálogo permite añadir nuevas columnas o estructuras a una tabla, estructura o plantilla del repositorio en edición.

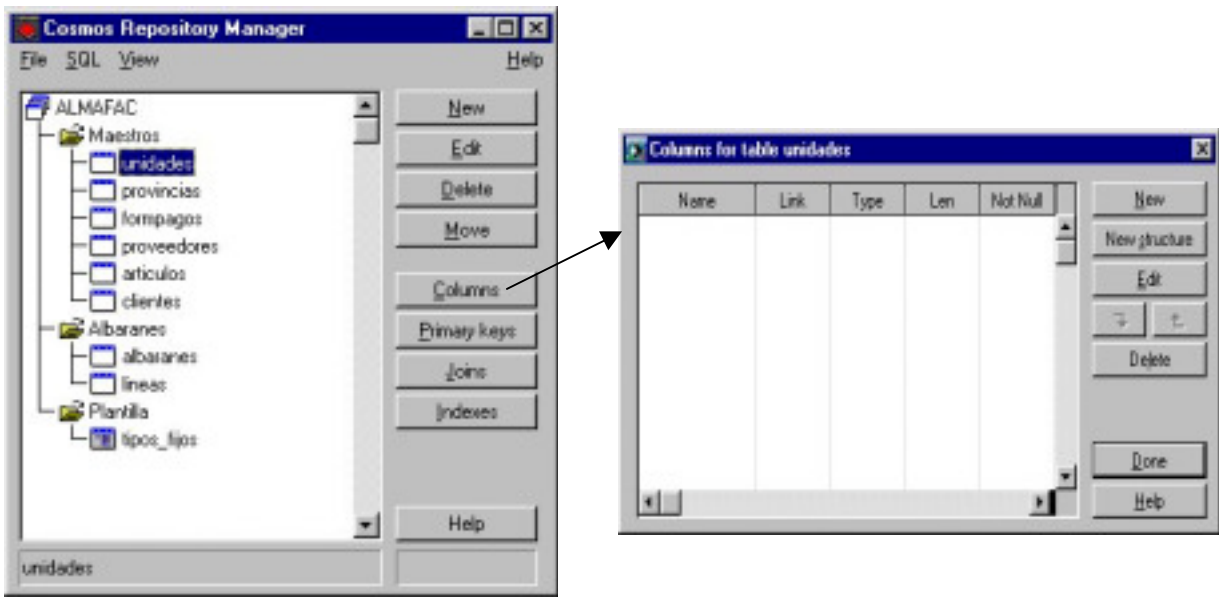


Figura 8.12. Añadir columnas o estructuras la tabla unidades

Cuadro de diálogo Columns for table

Este cuadro de diálogo permite añadir nuevas columnas o estructuras a una tabla, estructura o plantilla del repositorio en edición.

La descripción de cada uno de los campos que aparecen en el cuadro de diálogo es la siguiente:

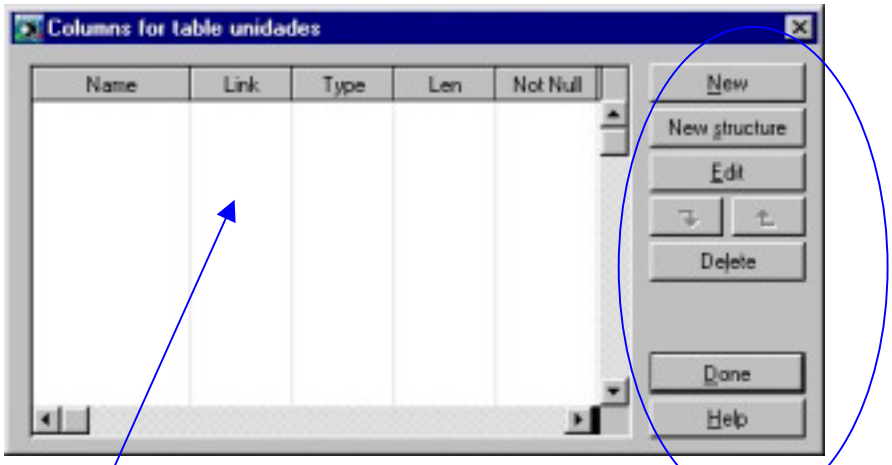


Figura 8.13. Cuadro de diálogo Columns for table

Lista Multicolumna Muestra las columnas y estructuras, con algunos de sus atributos, de la tabla, plantilla o estructura que se tiene seleccionada. En esta lista puede modificar el orden de sus elementos

Botones Permiten realizar las acciones de edición sobre las columnas o estructuras de la tabla, plantilla o estructura que tenga en edición

Para añadir una nueva columna hay que seguir los siguientes pasos:

1. En el cuadro de diálogo *Columns for Table* pulse el botón *New*.
2. Se muestra el cuadro de diálogo de *Column Edition* (se explica mas adelante), que permite definir los atributos de la columna.
3. Las columnas añadidas se irán mostrando automáticamente en el cuadro de diálogo *Columns for Table*.

Para modificar una columna hay que seguir los siguientes pasos:

1. Seleccione en la lista la columna que desea modificar.
2. Haga doble clic sobre la columna seleccionada o pulse el botón *Edit*, se muestra el cuadro de diálogo de *Column Edition* que permite modificar sus atributos.

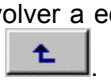
Para añadir una nueva estructura hay que seguir los siguientes pasos:

1. Pulse el botón *New structure*, se muestra el cuadro de diálogo de *Structure Edition for Table* que permite definir sus características (se explica mas adelante).
2. La estructura añadida se mostrará automáticamente en este cuadro de diálogo.

Para editar una estructura hay que seguir los siguientes pasos:

1. Seleccione en la lista la estructura que desea modificar.
2. Pulse el botón *Edit*, se muestra un cuadro de diálogo que permite modificar el nombre y la etiqueta de dicha estructuras.

Para modificar una estructura hay que seguir los siguientes pasos:

1. Seleccione en la lista la estructura que desea modificar.
2. Si la estructura no esta enlazada:
   - Haga doble clic sobre ella o pulse el botón  , automáticamente se muestran en este cuadro de diálogo los elementos de dicha estructura.
   - Para volver a editar el elemento padre de la estructura, ciérrela pulsando el botón  .
3. Si la estructura esta enlazada, haga doble clic sobre ella, se muestra un cuadro de diálogo que permite modificar el nombre y la etiqueta de dicha estructura. Para modificar sus elementos, tendrá que hacerlo sobre la estructura con la que este enlazada.

Para borrar una columna o una estructura hay que seguir los siguientes pasos:

1. Seleccione en la lista el elemento que desea borrar.
2. El sistema pedirá conformidad para realizar el borrado.
3. Si se esta borrando una estructura se da opción a:
   - Realizar el borrado automáticamente.
   - Pedir confirmación para cada uno de los elementos de la estructura.
4. Si la columna o estructura que se desea borrar forma parte de una clave referencial o de la clave primaria o si otra columna u otra estructura esta enlazada con ella, se muestra el mensaje correspondiente y no se procede al borrado.

Para modificar el orden de las columnas y/o estructuras, hay que seguir los siguientes pasos:

1. Seleccione en la lista la columna o estructura que desea cambiar de posición.
2. Arrástrela hasta la posición deseada.
3. Suelte el botón del ratón.

La lista multicolumna que muestra las columnas de la tabla dispone de panel doble que se activa mediante ese rectángulo negro que aparece en la parte inferior derecha de la lista, haciendo clic sobre el y arrastrándolo a la izquierda sin soltar el botón del ratón, de esta forma puede visualizar a la vez atributos de las columnas que aparecen separados.

Figura 8.14. Lista multicolumna con el panel doble

Cuadro de diálogo Column Edition

En este cuadro de diálogo se añaden columnas a la tabla en edición definiendo los atributos (de programación y SQL) de las columnas.

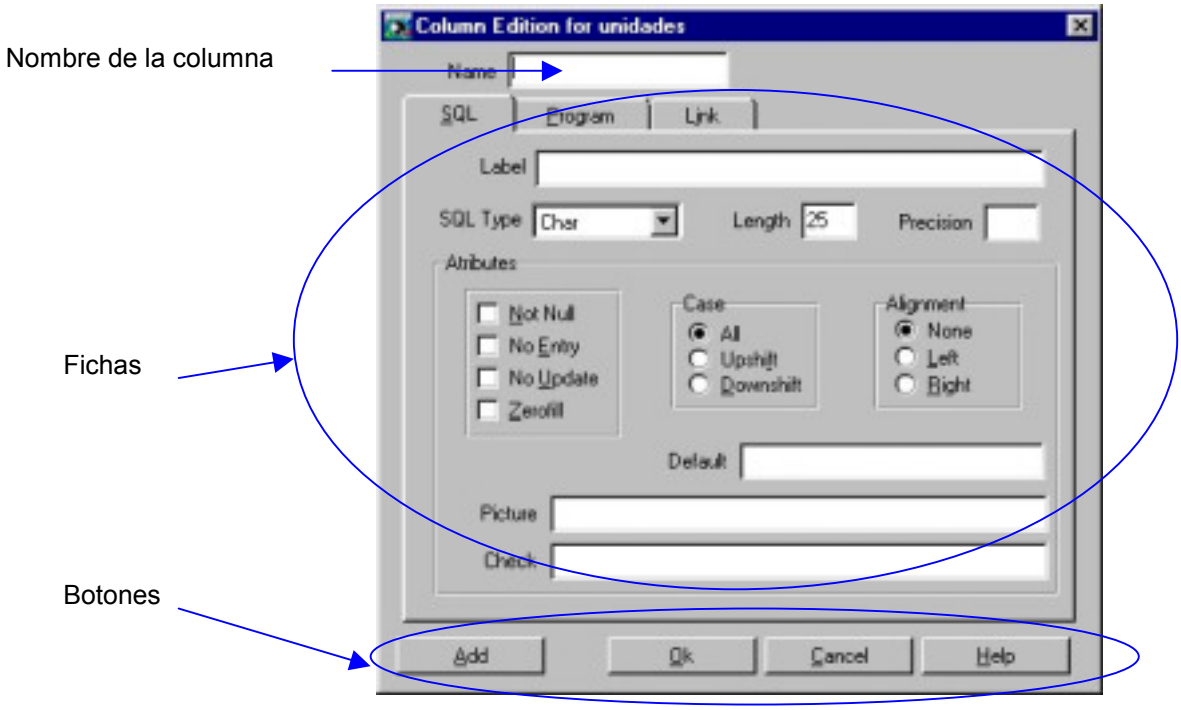


Figura 8. 15. Cuadro de diálogo de Column Edition

Los campos que se muestran en el cuadro de diálogo son los siguientes:

Campos	Significado
Name	Nombre de la columna Este nombre deberá ser un identificador SQL único en la tabla donde se define. Este campo es obligatorio
Fichas	Hay tres fichas diferentes que definen las características de la columna, cada una de ellas se explica posteriormente en detalle.
Botones	Tienen toda la operatividad necesaria para el manejo de este cuadro. El botón Add solo aparecerá cuando se editan nuevas columnas, utilice este botón cuando desee añadir más de una columna a la misma tabla. El botón Ok añade la columna en edición cerrando el cuadro de diálogo.

Ficha SQL

En esta ficha se encuentran todos los atributos SQL que se pueden aplicar a una columna de la base de datos.

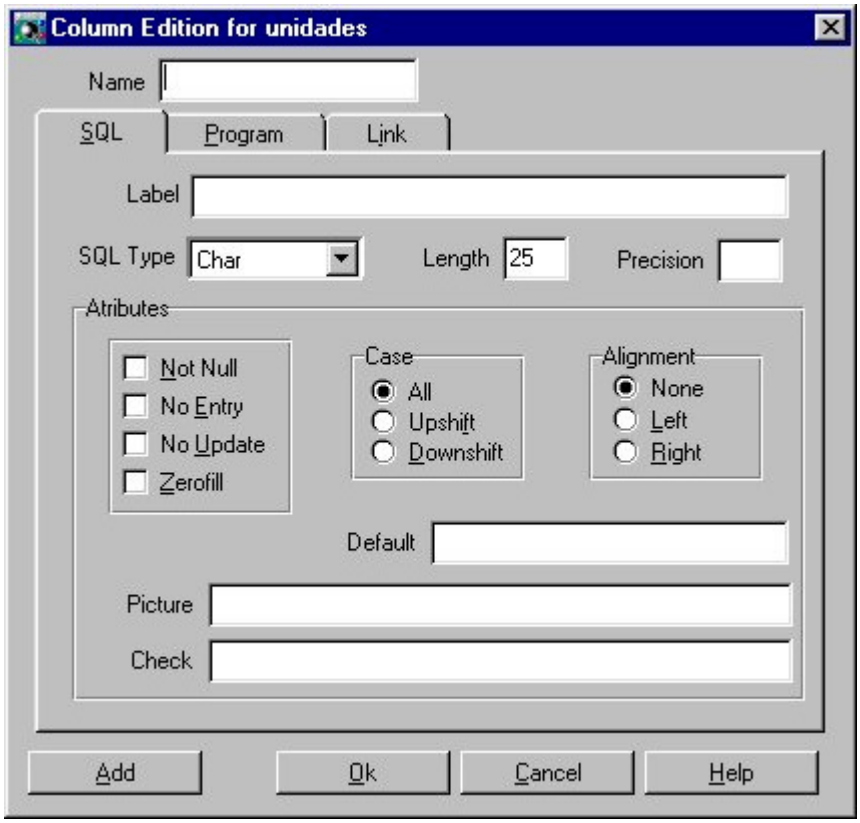


Figura 8.16. La ficha SQL

Label Siempre que se ejecute una instrucción SQL de consulta en la que aparezca esta columna se utilizará la etiqueta definida como encabezamiento de los datos.

SQL Type Identifica el tipo de dato de la columna que se está definiendo. Sus posibles valores son:

Tipo de Dato	Se aplica a:
CHAR	Caracteres alfanumérico
SMALLINT	Números enteros
INTEGER	Números enteros
TIME	Hora
DECIMAL	Números decimales
SERIAL	Contador automático
DATE	Fecha
MONEY	Números decimales

Length

Longitud de ocupación o definición del tipo de dato elegido en el campo anterior. Los tipos de datos cuya longitud no puede ser modificada, y que aparecerá automáticamente después de su selección son los siguientes:

Tipo de Dato	Longitud
SMALLINT	2
INTEGER	4
TIME	4
SERIAL	4
DATE	4

Por su parte, los tipos de datos en los que habrá que indicar su longitud a la hora de definirlos son:

Tipo de Dato	Longitud
CHAR	Número de caracteres
DECIMAL	Número de cifras significativas, enteras y decimales
MONEY	Número de cifras significativas, enteras y decimales

Precision

Número de decimales a utilizar para los tipos de datos DECIMAL y MONEY.

Attributes

Conjunto de atributos SQL referidos a la columna, son los siguientes:

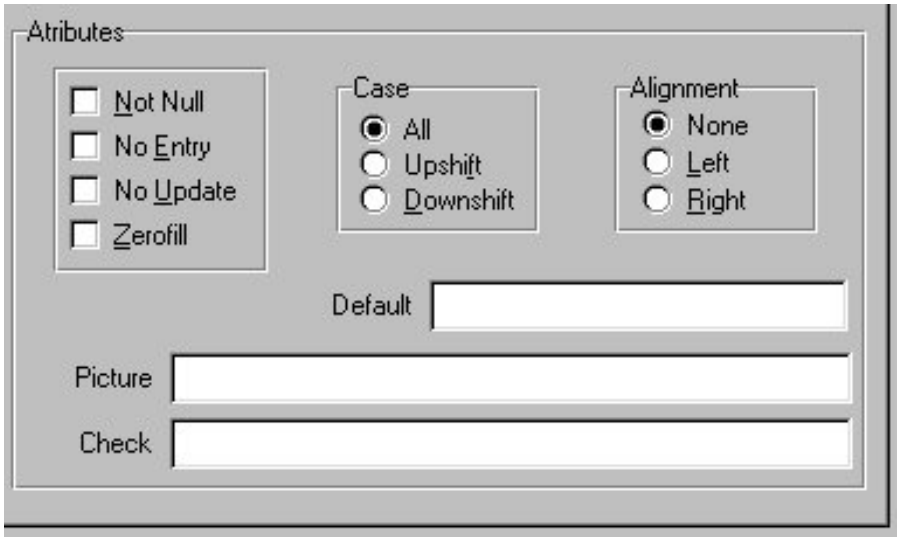


Figura 8.17. Conjunto de Atributos de la ficha SQL

Atributo	Significado	
Not Null	Mediante este atributo podremos determinar si una columna acepta o no valores nulos	
No Entry	En este campo indicaremos si la columna podrá incluirse o no en la sintaxis de una instrucción INSERT	
No Update	En este campo indicaremos si la columna podrá incluirse o no en la sintaxis de una instrucción UPDATE	
Zerofill	Permite indicar si en la definición de las columnas (CHAR o numéricas) la alineación será a la derecha y rellenando con ceros a la izquierda	
Case	En este campo indicaremos si una columna de tipo CHAR admitirá únicamente mayúsculas, minúsculas o ambas indistintamente. Los valores son:	
	All	Admitirá mayúsculas y minúsculas
	Upshift	Sólo admitirá mayúsculas
	Downshift	Sólo admitirá minúsculas
Alignment	En este campo se indica el tipo de alineación para la columna. Sus posibles valores son	
	None	Los tipos de datos numéricos se alinearán a la derecha
	Right	Alineación a la derecha
	Left	Alineación a la izquierda
Default	En este campo podremos indicar el valor por defecto que se utilizará en caso de que la columna no participe en la instrucción INSERT. Este valor por defecto será utilizado igualmente en los módulos de Entrada de Datos que se generen. Los valores que podrán especificarse son constantes y variables de SQL (today, para tipos DATE, y now, para tipos TIME). Si se desea que el valor por defecto sea un cadena de blancos no nula introduzca un espacio	
Picture	En este campo podremos indicar si se va aplicar una máscara a las columnas de tipo CHAR a la hora de su edición y presentación en pantalla	
Format	En este campo podremos indicar si se va aplicar una máscara a las columnas que no son de tipo CHAR a la hora de su edición y presentación en pantalla.	
Check	Mediante este campo podremos indicar una condición válida de SQL que se deberá cumplir en cualquiera de las operaciones de INSERT o UPDATE, cuyo resultado deberá ser TRUE. El nombre de la columna en la condición puede ser sustituido por el carácter $	

Ficha Programa

En esta ficha se encuentran todos los atributos de programación que se pueden aplicar a una columna de la base de datos.

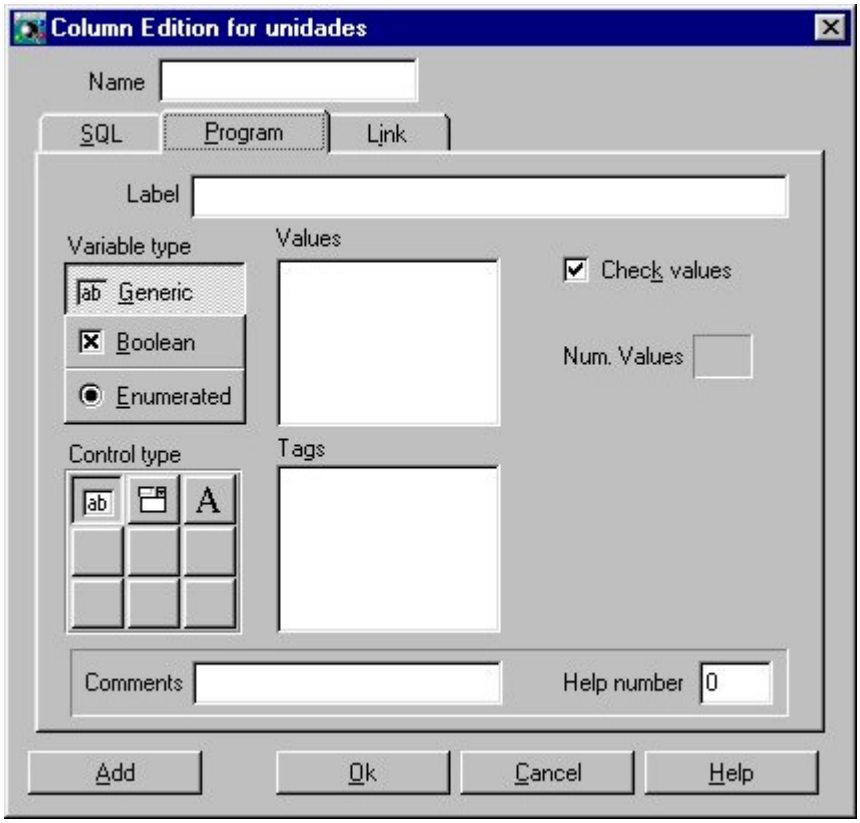


Figura 8.18. Ficha Program

Label

Siempre que se ejecute una instrucción SQL de consulta en la que aparezca esta columna se utilizará la etiqueta definida como encabezamiento de los datos. Es el mismo campo que aparece en la Ficha SQL. Este campo tiene el mismo valor en todas las fichas.

Variable type

Puede ser de tres tipos:

Tipos	Significado
Generic	Admitirá cualquier valor. Es decir, admitirá un conjunto de valores indefinido
Boolean	Admitirá dos valores diferentes, aparte del valor nulo
Enumerated	Admitirá un conjunto enumerado o limitado de valores

Values

Campo de edición en el que se indicará el conjunto de valores que puede tomar la columna. Los valores irán separados por un retorno de carro.

Check values

Si esta casilla de verificación esta marcada, la columna admitirá valores distintos de los indicados en el campo *Values*, en caso contrario solo admitirá los indicados en dicho campo.

Num. values Número total de valores asignados a la columna.

Tags Etiquetas asociadas a cada uno de los valores que puede tomar la columna. Las Etiquetas irán separadas por un retorno de carro. El número de etiquetas debe coincidir con el número de valores que puede tener la columna.

Mask Tipo de máscara. que se utilizará en la representación gráfica de la columna en la generación de programas. Este campo solo está visible para el *Variable type* *Generic*.

Control type Muestra un grupo de botones con diferentes tipos controles, compatibles con el tipo de variable seleccionado, el tipo de control seleccionado será usado como representación gráfica de la columna en la generación de programas. La interrogación representa el control por defecto para el tipo de variable seleccionado.

Help number En este campo podremos indicar un número de mensaje de ayuda. De este modo, una vez construido el fichero de ayudas y mensajes, éstos se utilizarán desde los módulos de entrada de datos a través del atributo HELP generado en aquellos. Sus posibles valores son:

Valores	Significado
0	No utiliza número de mensaje
N	Número entre -32.767 y 32.767

Comments En este campo se podrá indicar una línea de texto descriptiva de la columna, de forma que durante la edición de ésta dicho texto aparezca en el panel de comentarios, facilitando así su edición. Su valor por defecto es en blanco.

Ficha Link

En esta ficha se definen los enlace entre columnas. Un enlace se puede definir entre una columna o estructura de una tabla, plantilla o estructura con una columna/estructura de otra tabla, plantilla o estructura.

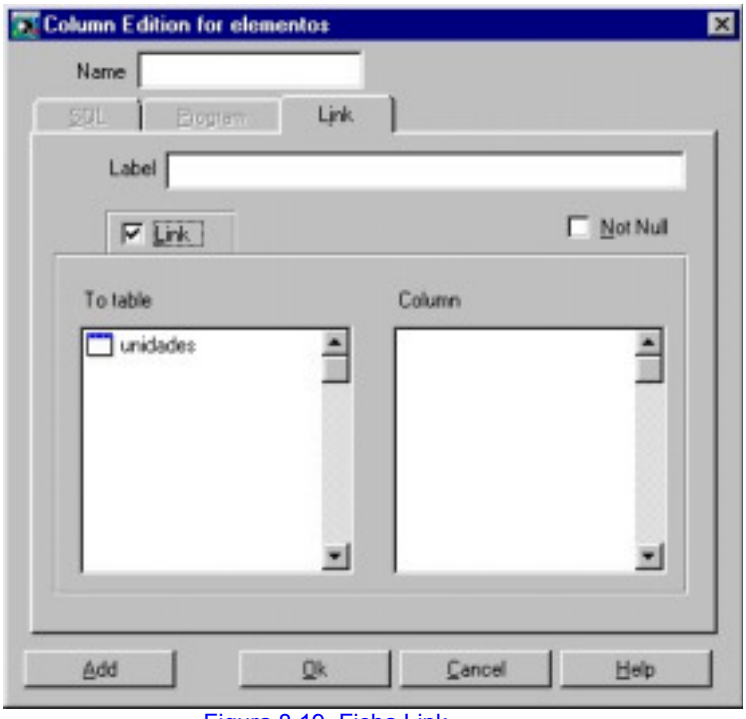


Figura 8.19. Ficha Link

Label Siempre que se ejecute una instrucción SQL de consulta en la que aparezca esta columna se utilizará la etiqueta definida como encabezamiento de los datos. Es el mismo campo que aparece en la Ficha SQL. Es opcional. Este valor se respeta siempre independientemente de que la columna está o no enlazada.

Not Null Mediante este atributo podremos determinar si una columna acepta o no valores nulos. Es el mismo campo que aparece en la Ficha SQL. Este valor se respeta siempre independientemente de que la columna está o no enlazada.

Link Casilla de verificación que permite indicar si la columna se va a definir o no como un enlace.

To table Lista de tablas y plantillas del repositorio ordenada alfabéticamente, aparecerá seleccionada la tabla o plantilla a la que pertenezca la columna con la que se establece el enlace.

Column Lista de columnas de la tabla o plantilla seleccionada en el campo *To table*. Para definir un enlace. Aparecerá seleccionada la columna con la que se establece el enlace. Este campo es obligatorio cuando se desea definir un enlace.

Para definir un enlace proceda de la siguiente manera:

1. Escriba el nombre de la columna.
2. Si lo desea de un nombre a la etiqueta, también puede determinar el valor del atributo Not Null .
3. Marque la casilla de verificación Link. Automáticamente se habilitarán los campos To Table y Column que le permiten definir el enlace. Igualmente se deshabilitarán las otras fichas del cuadro de diálogo por que la columna enlazada tomará los atributos de la columna enlazante. Dichos atributos solo se podrán modificar en la columna enlazante.
4. Seleccione en el campo To Table la tabla o plantilla con la que desea enlazar. Automáticamente se mostrará en el campo Column la lista de columnas de la tabla o plantilla seleccionada.
5. Seleccione en el campo Column la columna con la que desea enlazar y pulse el botón Ok.

Cuadro de diálogo Structure Edition for Table

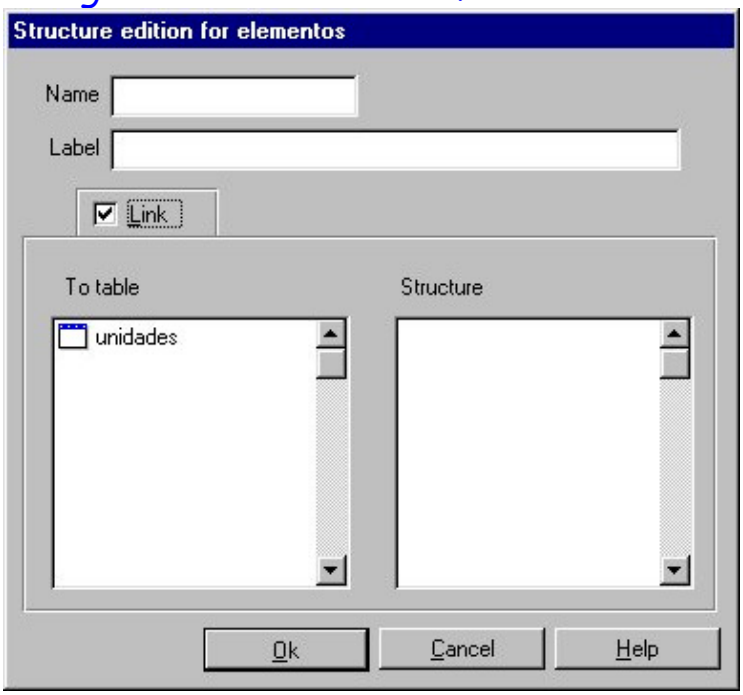


Figura 8.20. Aspecto del cuadro de diálogo Structure Edition for Table

En este cuadro de diálogo se permite crear o modificar una estructura. Los campos que se muestran en el cuadro de diálogo son los siguientes:

Name	Campo de edición en el que se indicará el nombre de la estructura. Este campo es obligatorio y a de ser único en la tabla o plantilla que lo contenga.
Label	Identificador de la estructura, es opcional.
Link	Casilla de verificación que permite indicar si la estructura se va a definir o no como un enlace.
To Table	Lista de tablas y plantillas del repositorio por orden alfabético.
Structure	Lista de estructuras de la tabla o plantilla seleccionada en el campo *To Table*. Este campo es obligatorio cuando se desea definir un enlace.

Para definir un enlace proceda de la siguiente manera:

1. Escriba el nombre de la estructura.
2. Si lo desea de un nombre a la etiqueta.
3. Marque la casilla de verificación *Link*. Automáticamente se habilitarán los campos *To Table* y *Structure* que le permiten definir el enlace.
4. Seleccione el campo *To Table* la tabla o plantilla con la que desea enlazar. Automáticamente se mostrará en el campo *Structure* la lista de estructuras de la tabla o plantilla seleccionada.
5. Seleccione en el campo *Structure* la estructura con la que desea enlazar y pulse el botón *Ok*.

Editar Claves Primarias (Botón Primary Key)

Una vez definidas todas las tablas de la base de datos, el siguiente paso a realizar será la definición de la clave primaria para cada una de ellas. La clave primaria, junto a las claves referenciales, es uno de los componentes de la integridad referencial.

Una clave primaria es un conjunto de columnas, ninguna de las cuales ha de admitir valores nulos (NOT NULL), que va a identificar de forma unívoca cada una de las filas de la tabla. Una tabla admite únicamente una clave primaria.

La definición de una clave primaria lleva implícita la creación de un índice único para cada columna componente de dicha clave, siendo 8 el número máximo de columnas.

La definición de la clave primaria es imprescindible si se desean generar procesos automáticos con actualización y borrado.

Los pasos necesarios para editar la clave primaria de una tabla son:

1. Seleccione en la estructura en árbol del repositorio la tabla para la que desea crear la clave primaria y pulse el botón *Primary keys*.

2. Se mostrará el cuadro de diálogo *Primary Key for Table* que permite editar la clave primaria de una tabla del repositorio.

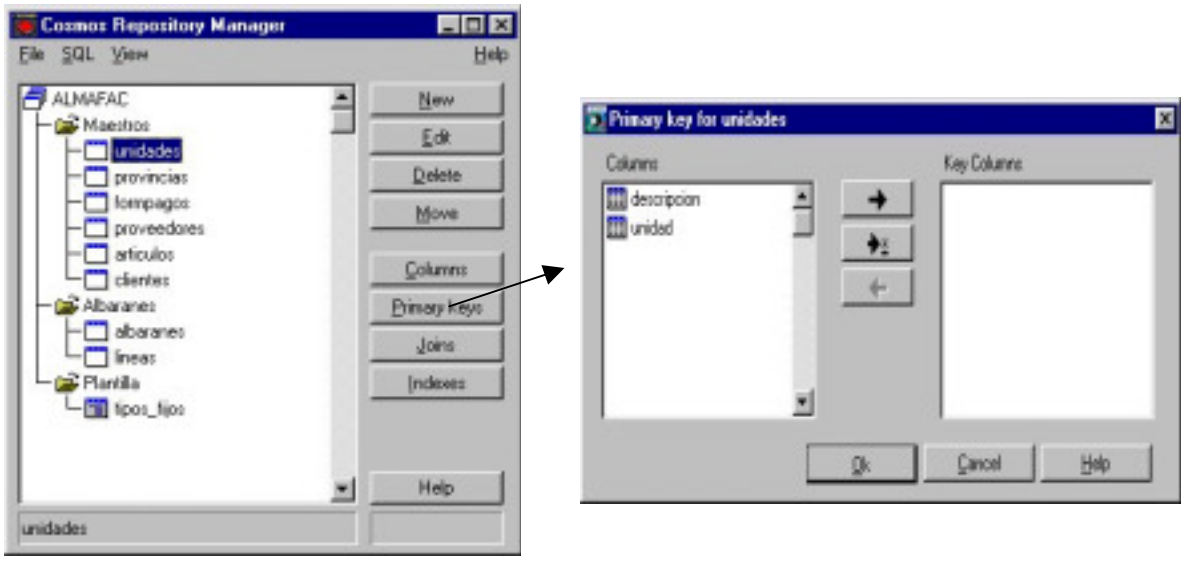


Figura 8.21. Crear una clave primaria

Cuadro de diálogo Primary Key for Table

Este cuadro de diálogo permite editar la clave primaria de una tabla del repositorio.

La descripción de cada uno de los campos que aparecen en el cuadro de diálogo es la siguiente:

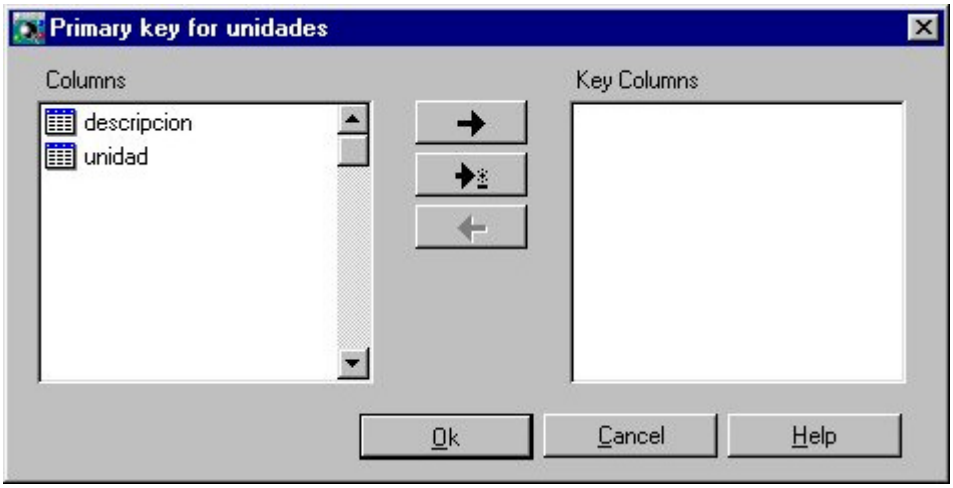


Figura 8.22. Cuadro de diálogo Primary Key for unidades

Columns Muestra una lista completa de las columnas de la tabla en edición.

Key Columns Lista ordenada de columnas seleccionadas para la clave primaria.

Para añadir una columna a la clave primaria :

1. Selecciónela en la lista de la izquierda y haga doble clic sobre ella o pulse el botón *Add*  .
2. Automáticamente se muestra en la lista *Key Columns* la columna seleccionada.
3. Repita los pasos uno y dos para el resto de las columnas que van a formar parte de la clave primaria.
4. Pulse el botón *Ok*.

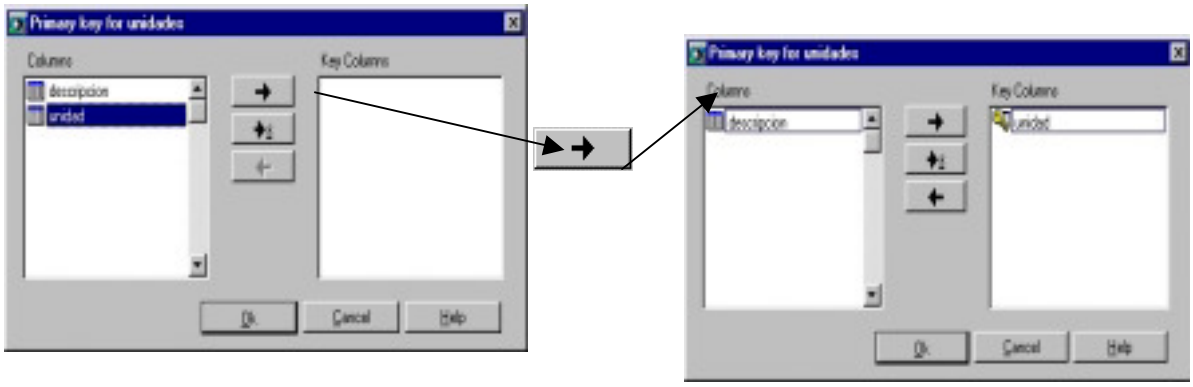


Figura 8.23. Insertar una columna a la clave primaria

Para insertar una columna, a la clave primaria, en una posición determinada seleccione la columna en la lista de la izquierda; a continuación, en la lista de la derecha seleccione la columna delante de la cual quiere que aparezca, seguidamente pulse el botón *Insert*  .

Para eliminar una columna de la clave primaria selecciónela en la lista de la derecha y pulse el botón *Remove*  .

Para borrar una clave primaria elimine todas la columnas de la lista de la derecha (*Key Columns*).

Editar indices (Botón Indexes)

Un índice es una utilidad manejada por el SQL para el acceso a los datos de las tablas. Opcionalmente, y si se conoce la forma de operar del SQL respecto a los accesos a las tablas, se pueden crear los índices de aquellas tablas previamente creadas. No obstante, en caso de duda, es preferible no crearlos hasta concluir el desarrollo de la aplicación, o bien hasta que verdaderamente estemos seguros de su función y utilidad.

Los motivos de definición de un índice son fundamentalmente los siguientes:

- Aumentar la velocidad de acceso a la tabla reduciendo el tiempo de respuesta.
- Asegurar la identificación unívoca de filas de una tabla.
- Facilitar los enlaces entre tablas (*join*), la ejecución de instrucciones SELECT del SQL con condiciones de *subselect* y la realización de entradas de datos multitabla mediante el objeto Form del lenguaje de cuarta generación COOL de Cosmos.

Las características y restricciones respecto a la creación de índices son las siguientes:

- El nombre del índice sirve para identificar dicho índice a la hora de su posible borrado, por tanto este debe ser único en toda la base de datos.
- Un índice puede construirse sobre una o más columnas (máximo 8) de una tabla, siendo simple o compuesto, respectivamente.
- Si las columnas componentes de un índice pueden tener varios valores iguales en distintas filas de la tabla, entonces a este se le denomina *índice duplicado*. Por el contrario, si una determinada combinación de valores sólo puede aparecer una vez en la tabla, entonces se llamará *índice único*.
- La longitud máxima de un índice es de 120 bytes, ya sea único o duplicado. Es decir, la suma de las longitudes en bytes de las columnas que lo componen no debe superar los 120 bytes.
- Al crear un índice se puede establecer el orden de recuperación de los datos asociados, ascendente (por defecto) o descendente.
- No se pude crear un índice con una estructura (lista de columnas componentes) idéntica a la de otro índice existente sobre la misma tabla.
- Un índice no se puede crear con columnas de distintas tablas de la base de datos.
- No se puede crear un índice duplicado, simple o compuesto, para una tabla en la que existan más de 32.767 filas con valores duplicados para el conjunto de columnas que componen dicho índice.
- Los tipos de datos de las columnas que componen un índice pueden ser distintos.
- Si en el servidor de SQL se borra una columna que compone un índice o una tabla de la base de datos se borran también sus índices correspondientes, el administrador de repositorios no permitirá borrar la columna hasta que no se elimine el índice.
- Todos los índices generados para una tabla se graban en la tabla *sysindexes* del catálogo.
- La existencia de una clave primaria (*primary key*) supone la existencia de un índice único. Sin embargo la existencia de claves referenciales (*foreign keys*) no supone la creación de un índice.

Para editar los índices de una tabla proceda de la siguiente manera:

1. Seleccione en la estructura en árbol del repositorio la tabla para la que desea crear el índice y pulse el botón *Indexes*.
2. Se muestra el cuadro de diálogo *Indexes for Table* que muestra los índices definidos en la tabla en edición.

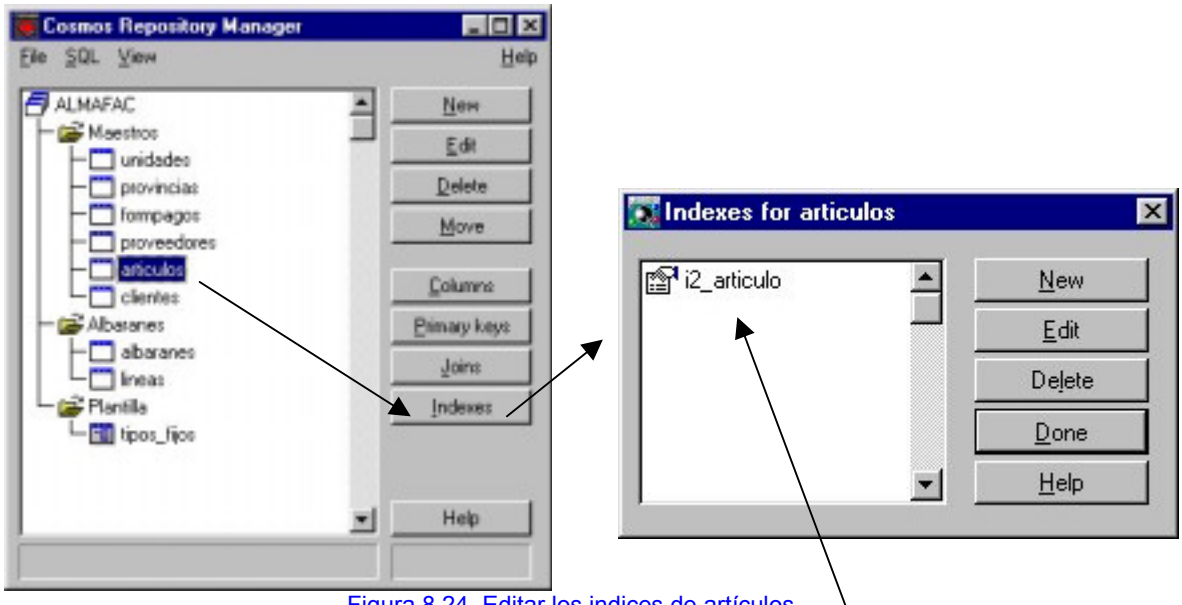


Figura 8.24. Editar los indices de artículos

La tabla artículos tiene definido un índice: i2_articulo

Cuadro de diálogo Indexes for Table

En este cuadro de diálogo se permite editar, crear, modificar y borrar índices de la tabla en edición.

Lista de índices definidos en la tabla en edición.

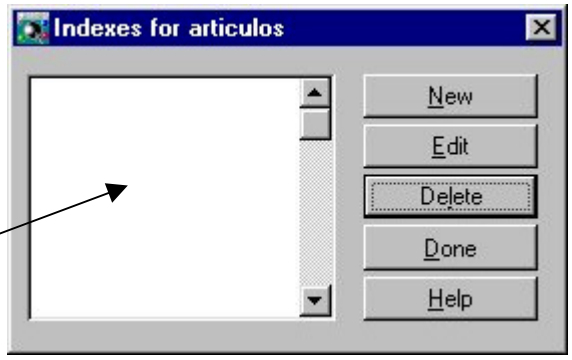


Figura 8.25. Cuadro de diálogo Indexes for la tabla artículos

Para añadir un índice proceda de la siguiente manera:

1. Pulse el botón *New* del. cuadro de diálogo.
2. Se muestra el cuadro de diálogo *Index for Table* que permite seleccionar las columnas de la tabla que formarán parte del índice.

Para modificar un índice proceda de la siguiente manera:

1. Seleccione en la lista de índices el que desea modificar.
2. Haga doble clic sobre él o pulse el botón *Edit*.
3. Se muestra el cuadro de diálogo *Index for Table* que le permite modificar sus propiedades. Pulse el botón *Ok*, para que las modificaciones tomen efecto.

Para borrar un índice proceda de la siguiente manera:

1. Seleccione en la lista de índices el que desea borrar.
2. Pulse el botón *Delete*. El sistema pedirá conformidad para realizar el borrado.
3. Si se procede al borrado el índice será borrado automáticamente de la lista.

Una vez realizadas todas las operaciones necesarias pulse el botón *Done*.

Cuadro de diálogo Index for Table

En este cuadro de diálogo se definen las propiedades de los índices de las tablas.

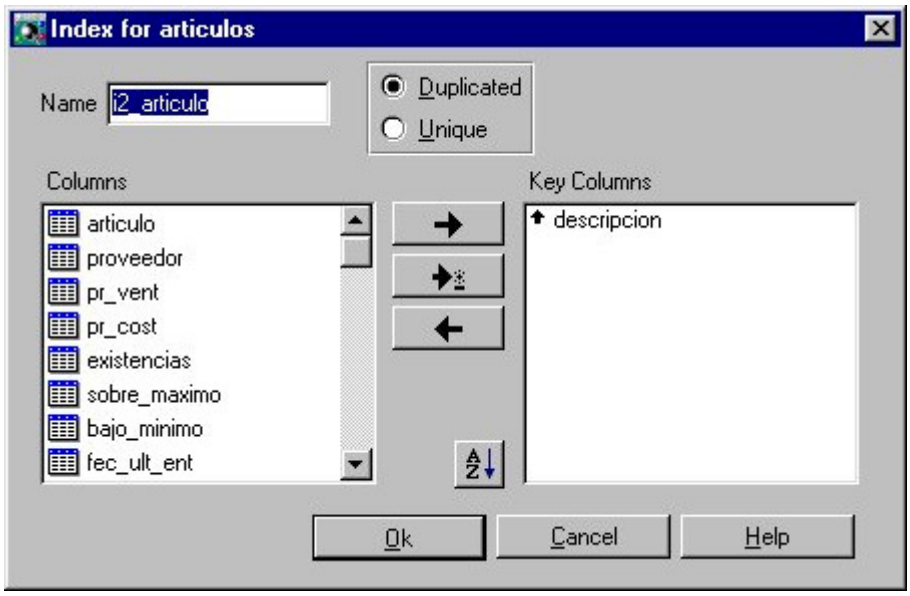


Figura 8.26. Cuadro de diálogo Index for artículos

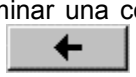
La descripción de cada uno de los campos que aparecen en el cuadro de diálogo es la siguiente:

Name	Nombre de un índice perteneciente a la tabla activa. Dicho nombre deberá ser un identificador SQL único que no haya sido utilizado anteriormente para la definición de otro índice en la misma base de datos. Este campo es obligatorio.
Duplicated	Permite definir el índice como duplicado.
Unique	El índice no admitirá valores duplicados.
Columns	Muestra una lista completa de las columnas de la tabla en edición.
Key Columns	Lista de columnas seleccionadas para el índice.

Para definir el índice :

1. Escriba el nombre del índice.
2. Indique si desea definir el índice como único o duplicado.
3. Para añadir una columna al índice selecciónela en la lista de la izquierda y haga doble clic sobre ella o pulse el botón *Add*  .
4. Automáticamente se muestra en la lista *Key Columns* la columna seleccionada.
5. Seleccione en la lista *Key Columns* la columna y pulse el botón  para indicar el orden de recuperación de los datos asociados, ascendente (por defecto ↑) o descendente ↓ .
6. Repita los pasos tres, cuatro y cinco para el resto de las columnas que van a formar parte del índice.

Para insertar una columna, al índice, en una posición determinada seleccione la columna en la lista de la izquierda; a continuación, en la lista de la derecha seleccione la columna delante de la cual quiere que aparezca, seguidamente pulse el botón de *Insert*  .

Para eliminar una columna del índice selecciónela en la lista de la derecha y pulse el botón *Remove*  .

Una vez realizadas todas las operaciones necesarias pulse el botón *Ok*.

Editar Joins y Claves Referenciales

Una clave referencial establece la relación entre la tabla para la que se crea y la tabla que referencia. Mediante esta relación el usuario nunca podrá actualizar la columna o columnas que componen la clave primaria en la tabla referenciada o borrar la fila si existe información de ella en la tabla referencial.

Una clave referencial tiene que estar compuesta por el mismo número de columnas que la respectiva clave primaria en la tabla referenciada. Estas columnas tienen que especificarse en el mismo orden y, además, ser del mismo tipo de dato SQL.

Una tabla puede tener más de una clave referenciada apuntando a varias tablas de la misma base de datos.

El concepto de join es equivalente al de clave referencial, la única diferencia es que un join solo existirá en el repositorio, indicando una relación entre tablas, pero no se reflejarán en la base de datos. Las claves referenciales son joins que existen en el repositorio y en la base de datos. Todas las claves referenciales son joins pero no todos los joins son claves referenciales.

Una clave referencial no es un índice. Una clave referencial o un join no se puede crear si no se ha creado previamente la clave primaria en la tabla referenciada, ya que ésta es la clave por la cual se establece la relación entre las dos tablas.

El nombre de la columna o columnas que componen la clave referencial y la clave primaria no tienen por qué ser iguales.

Para editar un join o una clave referencial de una tabla del repositorio:

1. Seleccione en la estructura en árbol del repositorio la tabla para la que desea crear el join o la clave referencial y pulse el botón *Joins*.

2. Se mostrará el cuadro de diálogo *Joins for Table* que permite establecer la relación entre dos tablas del repositorio.

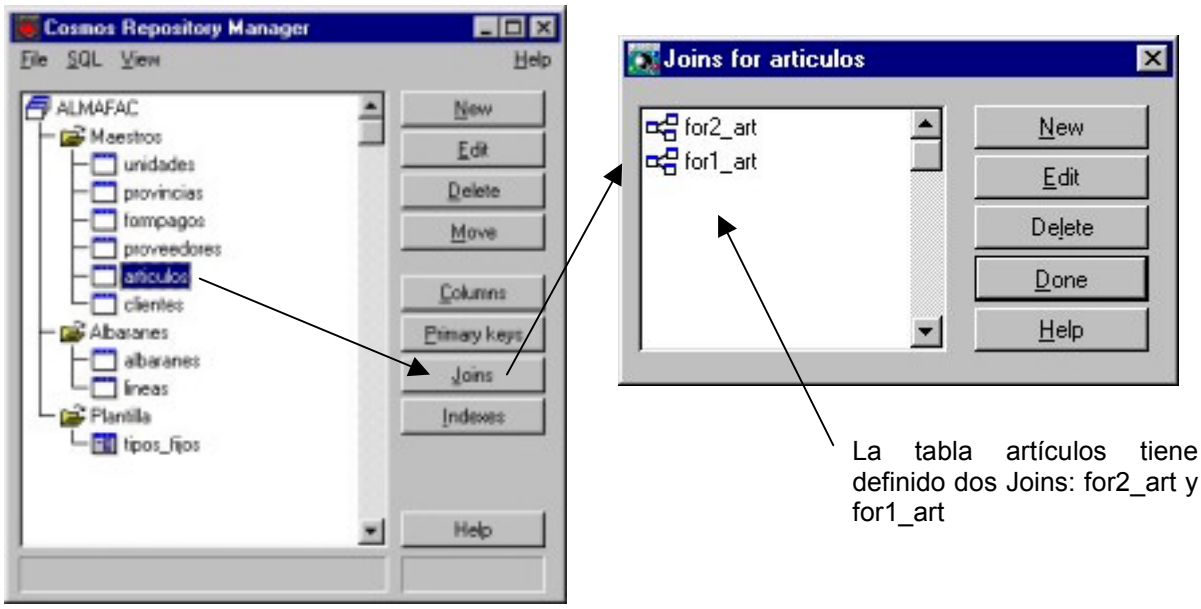


Figura 8.27. Editar Joins

Cuadro de diálogo Joins for Table

En este cuadro de diálogo se permite editar, crear, modificar y borrar claves referenciales o joins de la tabla en edición.

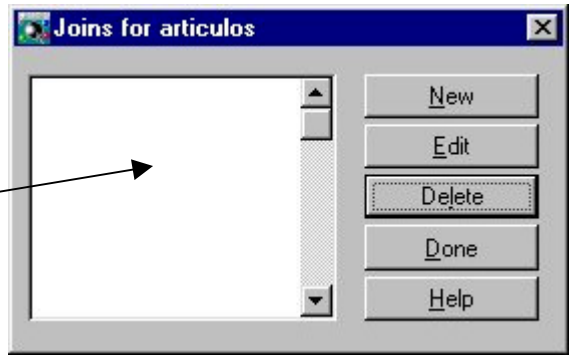


Figura 8.28. Cuadro de diálogo Joins for artículo

Para añadir un join o una claves referenciales :

1. Pulse el botón *New*.
2. Se mostrara el cuadro de diálogo *Join Edition for Table* que permite introducir los datos necesarios para definirla. Pulse el botón *Ok*, automáticamente aparecerá el nuevo join o la nueva clave referencial en la lista de joins y claves referenciales de este cuadro de diálogo.

Para modificar un join o una clave referencial :

1. Seleccione en la lista de joins el que desea modificar.
2. Haga doble clic sobre el o pulse el botón *Edit*.
3. Se muestra el cuadro de diálogo *Join Edition for Table* que le permite modificar los datos de la clave referencial o join. Pulse el botón *Ok*, para que las modificaciones tomen efecto.

Para borrar un join o una clave referencial :

1. Seleccione en la lista de joins y claves referenciales la que desea borrar.
2. Pulse el botón *Delete*. El sistema pedirá conformidad para realizar el borrado.
3. Si se procede al borrado dicha clave referencial o join será borrado automáticamente de la lista.

Una vez realizadas todas las operaciones necesarias pulse el botón *Done*.

Cuadro de diálogo Join Edition for Table

En este cuadro de diálogo se definen los joins y las claves referenciales para la tabla en edición. Mediante la definición de claves referenciales se establecen relaciones de integridad entre las tablas.

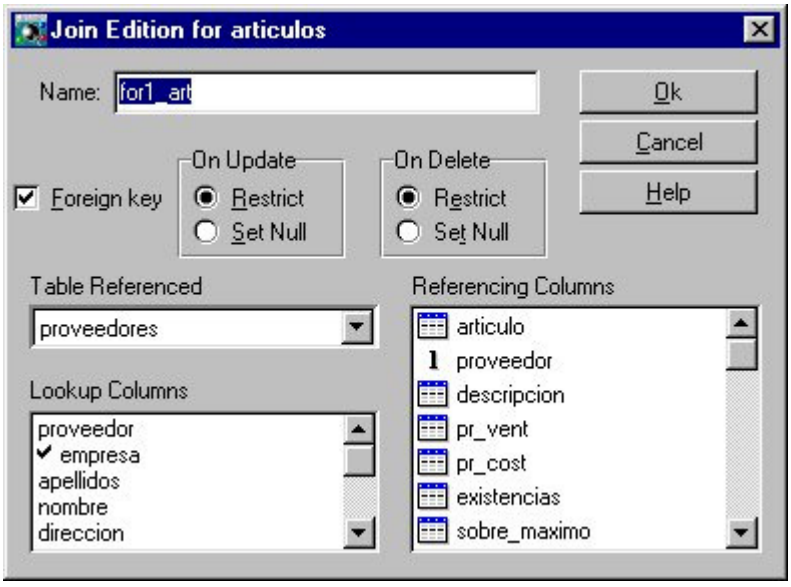


Figura 8.29. Cuadro de diálogo Join Edition for artículos

La descripción de cada uno de los campos que aparecen en el cuadro de diálogo es la siguiente:

Name — Identificador único de SQL que indica el nombre de la relación. Su definición es obligatoria.

Foreing key — Si no queda marca esta casilla de verificación, el join solo existirá en el repositorio, no existirá en la base de datos. Es decir, cuando se marca esta casilla de verificación se esta creando una clave referencial, en caso contrario se esta creando un join, y no se comprobará la integridad referencial al borrar o modificar columnas de la tabla referenciada.

On Update

Indica la acción que se deberá ejecutar en caso de modificar el contenido de las columnas que forman la clave primaria de la tabla referenciada. Sus posibles valores son los siguientes:

Restrict — Impide la modificación de las filas de la tabla referenciada en el caso de que exista alguna fila con los mismos valores sobre la tabla en edición.

Set Null — Permite la modificación de las filas de la tabla referenciada, modificando a su vez las columnas de relación con el valor *NULL* en todas aquellas filas que tengan los mismos valores dentro de la tabla en edición.

On Delete

Indica la acción que se deberá ejecutar en caso de borrar filas de la tabla referenciada. Sus posibles valores son los siguientes:

Restrict — Impide el borrado de filas de la tabla referenciada en el caso de que exista alguna fila con los mismos valores dentro de la tabla en edición.

Set Null — Permite el borrado de las filas de la tabla referenciada y modifica las columnas de relación con el valor *NULL* en todas aquellas filas que tengan los mismos valores dentro de la tabla en edición.

Table Referenced

Nombre de la tabla a la que hará referencia la tabla en edición. Este nombre deberá ser el de una tabla existente en el repositorio, la cual deberá tener definida además la clave primaria, ya que ésta es la clave por la cual se establece la relación.

Referencing Columns

Columnas que formarán la relación entre la tabla en edición y la clave primaria de la tabla referenciada (máximo 8).

Lookup Columns

En este campo se podrán seleccionar los nombres de las columnas de la tabla referenciada que se enlazarán automáticamente y que se mostrarán en la paleta del repositorio en el entorno de desarrollo. Son las columnas de lookup.

Para añadir una clave referencial o join:

1. Introduzca el nombre del join o clave referencial.
2. Indique el valor deseado para el atributo *Foreing key*.
3. Indique el modo de integridad en las operaciones de Update y Delete.
4. Seleccione la tabla referenciada en el campo *Table Referenced*. Automáticamente se muestran en el campo *Lookup Columns* las columnas de la tabla seleccionada.
5. Seleccione las columnas de lookup haciendo doble clic sobre ellas, aparecerán marcadas con ✔. Puede eliminar una columna seleccionada haciendo otra vez doble clic sobre ella.
6. Seleccione las columnas referenciadas haciendo doble clic sobre ellas, en el orden en el que intervienen en la clave, aparecerán marcadas con 1 u otro número dependiendo del orden que ocupen en la clave. Puede eliminar una columna de la clave referencial haciendo otra vez doble clic sobre ella, al eliminar una columna serán eliminadas también las que le sucedan en orden, seleccione entonces la siguiente columna de la clave en orden.
7. Una vez conforme con los datos introducidos pulse el botón *Ok*.

Guardar el Repositorio

Una vez creadas todas las tablas del repositorio con sus claves primarias, sus claves referenciales y sus joins el siguiente paso será guardar el repositorio. Los pasos para hacerlo son:

1. Ejecute la opción *Save* o *Save As* del menú *File*.
2. Automáticamente se muestra el cuadro de diálogo *Save As*. Escriba el nombre del repositorio, seleccione el directorio y la unidad donde desea guardarlo y pulse el botón *Ok*.

Figura 8.30. Opciones Save y Save as .

La opción *Save As* permite guardar tanto un repositorio de nueva creación como uno ya existente asignándole otro nombre. Al ejecutar esta opción se presentará en pantalla el mismo cuadro de diálogo que el de la opción *Save* cuando se ejecuta por primera vez.

En el caso de que el nombre asignado al repositorio coincidiese con el de algún otro ya existente, el sistema avisaría al usuario presentando en pantalla un cuadro de mensaje.

A través de la opción *Save As* el usuario podrá elegir entre:

- Guardar el repositorio con el mismo nombre (*Sobre-escribir*), con lo cual se grabarían las modificaciones que hubieran podido realizarse sobre él (equivalente a la opción *Save*). En este caso el sistema pediría confirmación avisando de que se trata de un repositorio ya existente que se va a sobreescribir.
- Asignar un nombre distinto al actual, con lo cual estaríamos creando un nuevo repositorio.
- Cancelar la operación.

Crear un Nuevo Repositorio

Esta opción el menú *File* permite generar un repositorio nuevo. La forma de ejecución dependerá de si el repositorio en edición ha sido guardado previamente o no y de las posibles modificaciones realizadas:

1. En el caso de estar trabajando con un repositorio existente, la ejecución de esta opción dependerá de las modificaciones realizadas, pidiendo conformidad o no para guardar los cambios antes de abrir el nuevo repositorio (de nombre *Untitled.crf*).
2. Si el repositorio en edición no tuviese aún asignado un nombre, la ejecución de esta opción dependerá igualmente de las posibles modificaciones. En caso afirmativo, la ejecución sería idéntica a la de la opción *Save As* (para asignar un nombre al repositorio), mientras que en caso contrario se generaría un nuevo *Untitled.crf*.

Abrir un Repositorio existente

Esta opción el menú *File* permite abrir un repositorio existente. Su ejecución dependerá de las posibles modificaciones que se hubiesen realizado en el repositorio en edición y de si éste es un repositorio nuevo o existente.

Cuando se ejecuta esta opción se muestra un cuadro de diálogo que permite seleccionar el nuevo repositorio con el que se desea trabajar.

Salir del Administrador de Repositorios

La ejecución de esta opción cierra el administrador de repositorios de Cosmos y retorna al sistema operativo o al programa desde el que fue invocado dicho administrador.

5. Comunicación con el SQL y con Bases de Datos

Crear la Base de Datos

Después de diseñar un repositorio, el siguiente paso es generar la base de datos para dicho repositorio.

Para generar la base de datos proceda de la siguiente manera:

1. Ejecute la opción *Create Database* del menú *SQL*.

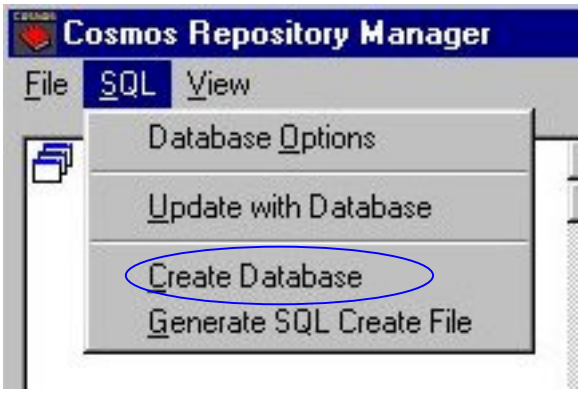


Figura 8.31. Menú SQL

2. Se muestra el cuadro de diálogo *Database Options*.

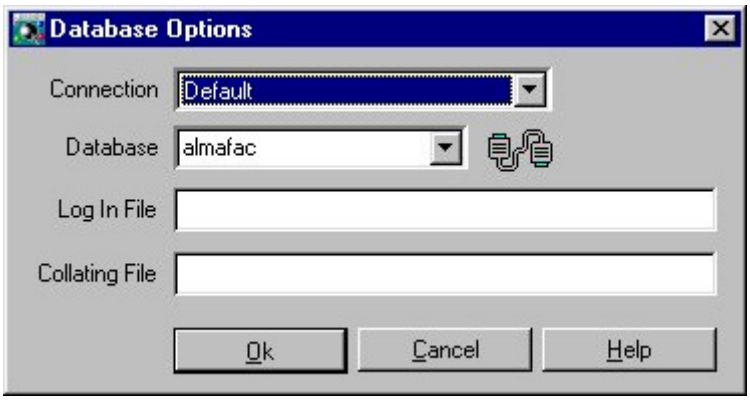


Figura 8.32. Cuadro de diálogo Database Options

3. Este cuadro de diálogo muestra los datos necesarios para definir las características principales de una base de datos.
4. Introduzca los valores necesarios en el cuadro de diálogo y pulse el botón *Ok*.
5. Se creará *on line* la base de datos en el servidor SQL seleccionado. Si la conexión el local se creará un directorio, de extensión *.dbs* en el que se incluirán una serie de ficheros, correspondientes tanto al catálogo de la base de datos (diccionario SQL formado por las tablas *sys*.**) como a las tablas de usuario (ficheros *.dat* y *.idx*).

Cuadro de diálogo Database Options

Muestra los datos necesarios para conectar con el servidor SQL y definir las características principales de una base de datos.

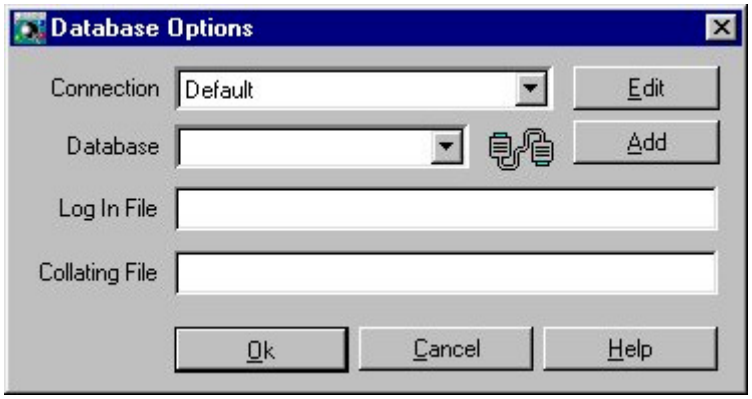


Figura 8.33. Cuadro de diálogo Database Options

Connection Lista de conexiones definidas en el fichero de configuración COSMOS.INI.

Database Muestra las bases de datos encontradas por el sistema para la conexión seleccionada en el campo anterior.

Botón Edit Esta opción permite editar el entorno de las conexiones definidas en el fichero de configuración COSMOS.INI. Para modificar el entorno de una conexión seleccione en la lista *Connection* aquella que desee modificar. A continuación pulse el botón *Edit*. Acto seguido se mostrará el cuadro de diálogo de *Connection* que permite realizar las modificaciones necesarias. Esta opción solo aparecerá cuando se arranque Cosmos con el usuario *system*.

Botón Add Al pulsar este botón se muestra el cuadro de diálogo *Add Connection* que permite añadir una conexión (local, remota u ODBC) al fichero de configuración. Esta opción solo aparecerá cuando se arranque Cosmos con el usuario *system*.

Los campos siguientes solo se muestran al ejecutar la opciones *Create database* y *Database Options* del menú *SQL*:

Log In File Nombre completo del fichero que contendrá el LOG FILE (histórico) que se utilizará para el tratamiento de transacciones.

Collating File La ordenación de los datos sigue el criterio de orden establecido en la tabla ASCII. Sin embargo, en algunos casos, este orden no se corresponde con el orden alfabético en nuestro idioma, como ocurre por ejemplo con las vocales acentuadas, la letra eñe (ñ), etc. En la creación de la base de datos podemos establecer los cambios de orden necesarios a través de un fichero de ordenación (*collating sequence*), en el que se especifican estas diferencias. Mediante este campo podremos indicar si queremos utilizar o no esta característica.

Actualizar un Repositorio con una Base de Datos

El Administrador de Repositorios permite realizar las siguientes acciones:

- Actualizar los datos del repositorio en edición con una base de datos seleccionada por el usuario.
- Actualizar una base de datos con los datos del repositorio en edición.
- Ver las diferencias entre el repositorio en edición y una base de datos sin realizar ninguna actualización.
- Actualizar una base de datos y el repositorio en edición de forma bidireccional, preguntando al usuario la acción a tomar para cada diferencia.
- Generar un fichero SQL con las diferencias existentes entre la base de datos y el repositorio. Puede ejecutar dicho fichero cuando desee actualizar la base de datos.

Para actualizar el repositorio proceda de la siguiente manera:

1. Ejecute la opción *Update with Database* del menú *SQL*.
2. Se muestra el cuadro de diálogo *Database Options* que permite seleccionar la base de datos con la que se desea actualizar el repositorio.
3. A continuación aparecerá el cuadro de diálogo *Update with database* que permite definir los parámetros necesarios para realizar las modificaciones que se crean convenientes.

Cuadro de diálogo Update with Database

Los campos que se muestran en el cuadro de diálogo son los siguientes:

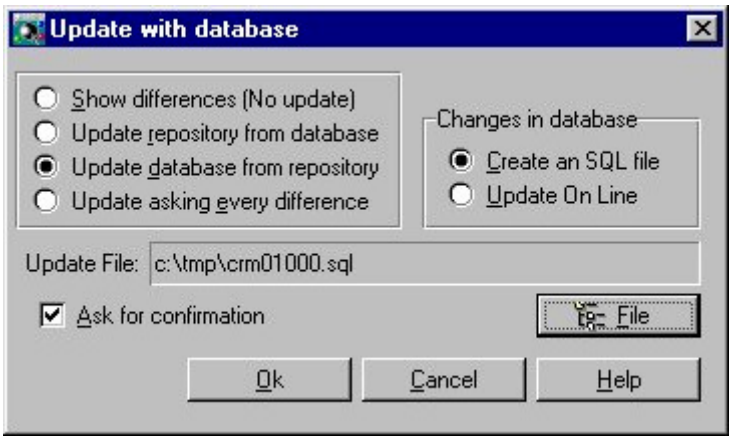


Figura 8.34. Cuadro de diálogo Update with Database

Show differences (No update)	No realiza ninguna modificación. Solamente muestra una a una las diferencias encontradas.
Update repository from database	Permite añadir al repositorio las diferencias encontradas con una base de datos.
Update database from repository	Permite añadir a la base de datos las diferencias encontradas con el repositorio.
Update asking every difference	Para cada una de las diferencias encontradas se preguntara si se desea hacer la actualización.
Changes in database	Solo estará activa cuando esta seleccionada la opción *Update database from repository* o la opción *Update asking every difference*:
Create an SQL file	No realiza ninguna actualización. Solo genera un fichero SQL con las diferencias existentes entre la base de datos y el repositorio.

Update On Line	Actualiza la base de datos con las diferencias encontradas. Si esta activa la opción *Update asking every difference* o la opción *Ask for confirmation* solo se harán las modificaciones para las que se ha dado confirmación. Si no están activadas ninguna de estas dos opciones se harán las modificaciones automáticamente.
Update file	Nombre del fichero SQL, que se utilizará por defecto para guardar las actualizaciones. Por defecto crea un archivo temporal el directorio DBTEMP.
Ask for confirmation	Para cada una de las diferencias encontradas pedirá confirmación para hacer la actualización en el repositorio o en la base de dados, dependiendo de donde haya encontrado la diferencia.
File	Permite cambiar el nombre y el directorio del fichero SQL cuando se selecciona *Create an SQL file*. Al pulsar este botón se muestra un cuadro de diálogo idéntico al de la opción *Save As*.

Una vez seleccionadas las opciones deseadas pulse el botón *Ok*.

A continuación se importará en catalogo de la base de datos seleccionada para proceder a su comparación.

Si se encuentran diferencias se muestra el cuadro de diálogo *Update* en el cual se podrán ver una a una las diferencias encontradas.

A continuación, si seleccionó *Update database from repository* y *Create an SQL file* se muestra en una ventana similar a la de la figura el fichero SQL generado.

Cuadro de diálogo Update

En este cuadro de diálogo se muestra una a una las diferencias encontradas entre el repositorio y la base de datos.

Las acciones que se pueden realizar desde este cuadro de diálogo dependen de los parámetros que se hayan definido en el cuadro de diálogo *Update with database*.

Con los datos encontrados en el repositorio que no se encuentran en la base de datos se permite:

- Borrarlo o modificarlo en el repositorio.
- Añadirlo o modificarlo en la base de datos.
- No hacer ninguna modificación.
- Salir sin realizar más modificaciones.

Con los datos encontrados en la base de datos que no se encuentran en el repositorio se permite:

- Borrarlo o modificarlo en la base de datos.
- Añadirlo o modificarlo en el repositorio.
- No hacer ninguna modificación.
- Salir sin realizar más modificaciones.

Figura 8.35. Aspecto del cuadro de diálogo Update

Si se marca la casillla de verificación *Update ALL* de este cuadro de diálogo se desactivan las opciones *Update asking every difference* y *Ask for confirmation* del cuadro de diálogo *Update with database*, en la opción que se pulse a continuación (*Delete in database, Create in database* o *Update in database,* o bien *Create in repository, Delete in repository,* o *Update in repository*), y esta se realizará para el resto de las diferencias.

Si se produce algún error en la actualización de la base de datos, debido a alguna incompatibilidad no contemplada en el repositorio, o a algún problema con los permisos de acceso, se le mostrará al usuario la frase SQL correspondiente a la acción que no se ha podido realizar, corrija el repositorio o la base de datos (según corresponda) y vuelva a intentarlo. Los cambios ya realizados quedarán reflejados en la base de datos y en el repositorio en edición, pero no se realizarán más comparaciones.

Si renombra algún elemento del repositorio que contuviese datos en la base de datos, renombrelo también en la base de datos desde el administrador de su base de datos o mediante el SQL interactivo, antes de proceder a la actualización, o de lo contrario se perderán los datos, ya que el repositorio no guarda registro de los nombres antiguos.

Generar un Fichero SQL con la estructura del Repositorio

El Administrador de Repositorios le permite generar un fichero SQL con la estructura del repositorio en edición. Dicho fichero recogerá todas las instrucciones SQL referidas a la definición de tablas que se encuentran en el repositorio.

El fichero que se genera tendrá extensión *.sql*.

Para generar el fichero SQL ejecute la opción *Generate SQL Create File* del menú *SQL*.

Se muestra un cuadro de diálogo idéntico al de la opción *Save As* que le permite indicar el nombre y el directorio del fichero. Una vez introducidos los datos pulse el botón *Ok*, automáticamente se presentará una ventana con el fichero SQL generado.

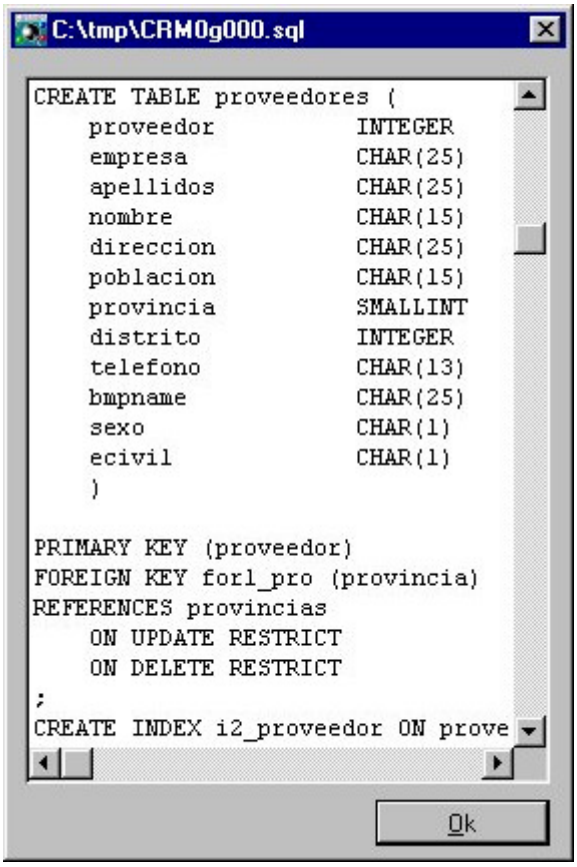


Figura 8.36. Fichero SQL generado

En el fichero SQL no quedará reflejada la información contenida en las plantillas, los joins sin clave referencial, la información relativa a grupos de tablas, la clasificación en estructuras, los enlaces entre elementos, ni los atributos de programación.

Importar datos de un Repositorio o una Base de Datos al Repositorio en edición

El Administrador de repositorios permite importar datos de un repositorio o una base de datos al repositorio en edición. Se podrán importar grupos, tablas, plantillas, estructuras y columnas.

Para importar datos ejecute la opción *Import palette* del menú *View*.

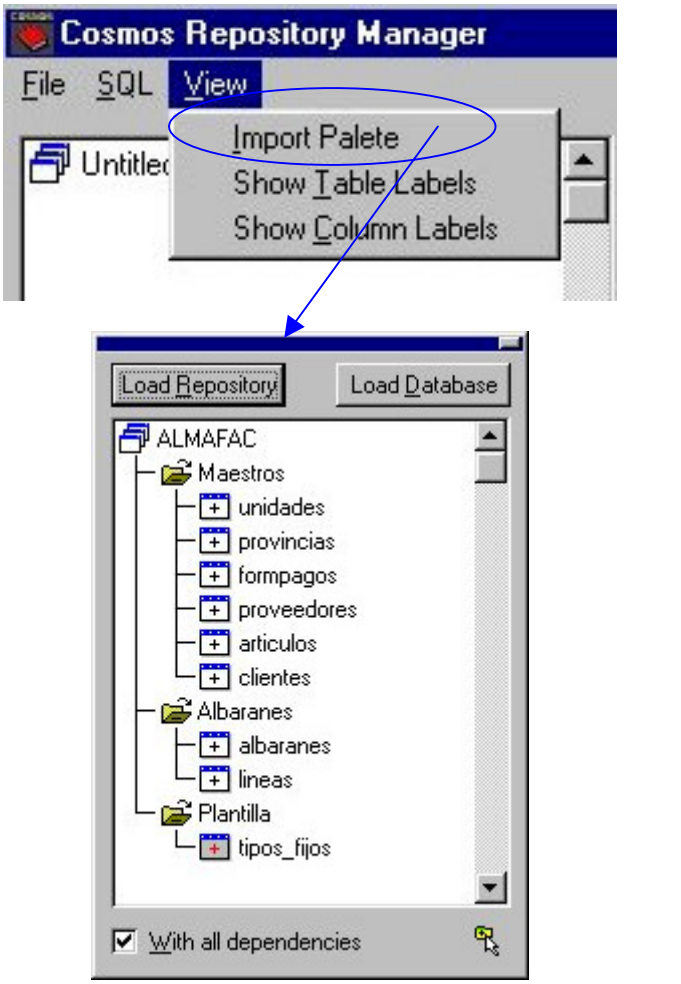


Figura 8.37. Aspecto del cuadro de diálogo después de cargar un Repositorio

Este cuadro de diálogo tiene los siguientes campos:

Load Repository — Al pulsar este botón se presenta un cuadro de diálogo idéntico al de la opción *Open* para seleccionar un repositorio ya existente, si lo desea puede abrir el mismo repositorio que esta editando, si este ha sido salvado previamente.

Load Database — Al pulsar este botón se presenta el cuadro de diálogo *Database Options* para seleccionar una base de datos existente.

Lista — Muestra la estructura en árbol del repositorio o de la base de datos que se han cargado utilizando una de las dos opciones anteriores.

With all dependencies — Si se marca esta casilla de verificación, se intentará resolver internamente los enlaces definidos en los datos que desea importar, en caso contrario los enlaces no quedarán definidos. Esta opción esta seleccionada por defecto.

La importación de tablas, grupos, plantillas, estructuras y columnas se realiza únicamente con la técnica de *arrastrar y soltar (drag & drop).*

Para importar una tabla, plantilla, o grupo al repositorio en edición:

1. Cargue la base de datos o el repositorio desde el cual desee importar los datos pulsando el botón correspondiente.
2. Seleccione en la lista: la tabla, plantilla, o grupo que desee importar.
3. Si desea que queden definidos los enlaces marque la casilla de verificación *With all dependencies*.
4. Con el botón izquierdo del ratón pulsado arrastre el elemento o elementos seleccionados hasta el grupo destino de la estructura en árbol del repositorio en edición que se encuentra en la ventana principal de la aplicación. Si arrastra un grupo, moverá con el todos los grupos, tablas y plantillas que contenga. Si arrastra una tabla o una plantilla moverá con ella todas las columnas y estructuras que contenga.
5. Suelte el botón del ratón.

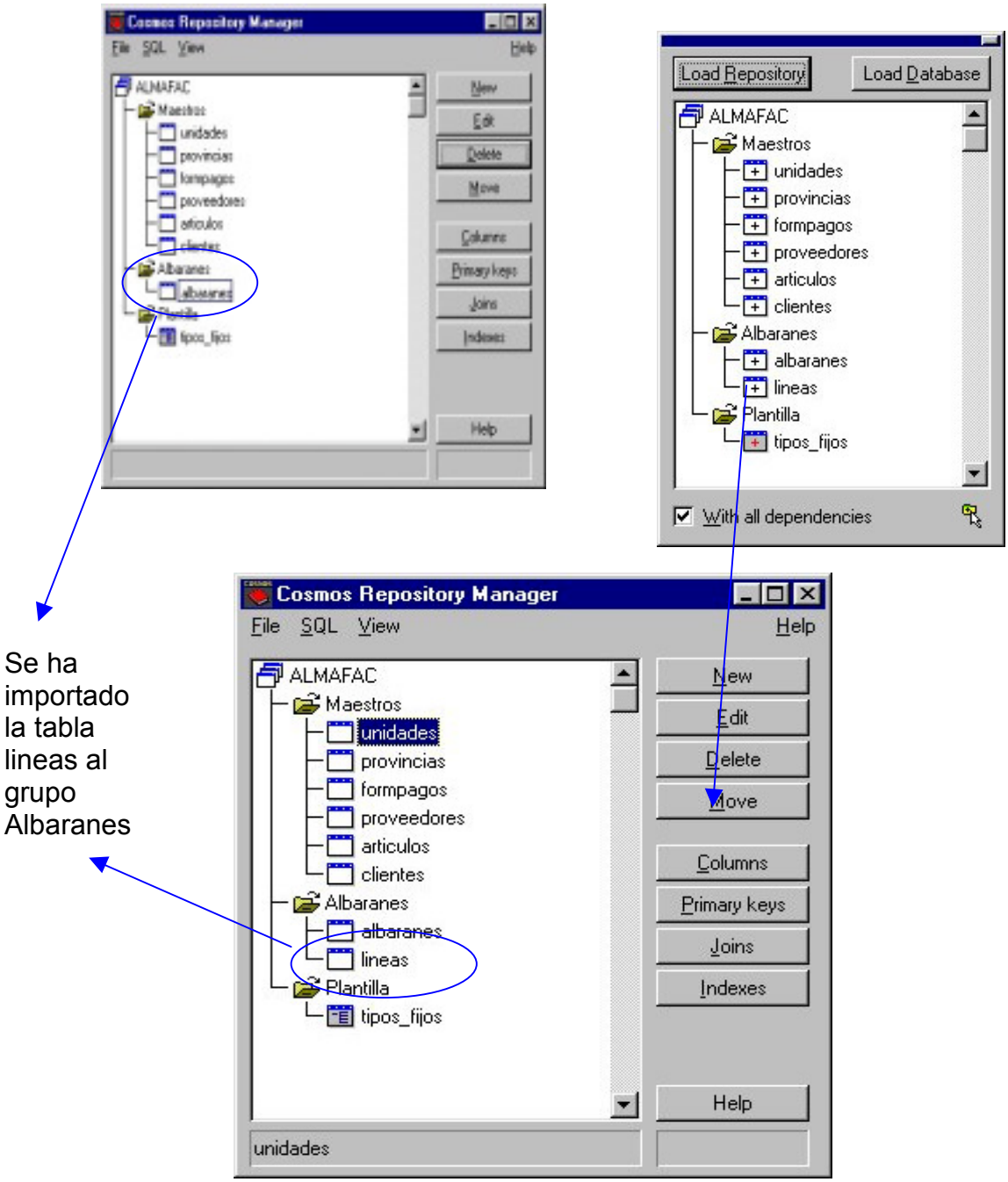


Figura 8.38. Importar de otro repositorio la tabla lineas al repositorio en edición.

Para importar una o más columnas o estructura a una tabla, plantilla o estructura del repositorio en edición:

1. Cargue la base de datos o el repositorio desde el cual desee importar una columna pulsando el botón correspondiente.
2. Seleccione en la lista del repositorio en edición: la tabla, plantilla o estructura a la cual desea importar la columna y haga doble clic sobre ella o pulse el botón *Columns*. Aparece un cuadro de diálogo *Columns for Table* que muestra las columnas y estructuras del elemento seleccionado.
3. Seleccione en la lista del repositorio cargado la columna o estructura que desee importar.
4. Si desea que queden definidos los enlaces marque la casilla de verificación Con todas las dependencias.
5. Con el botón izquierdo del ratón pulsado arrastre el elemento seleccionado hasta el cuadro de diálogo *Columns for Table.*
6. Suelte el botón del ratón.

En un repositorio no pueden existir dos grupos, tablas o foreign keys con el mismo nombre. Si se importa un grupo, una tabla o foreign key cuyo nombre ya existiera en el repositorio en edición se mostrará el mensaje correspondiente y se pedirá el nuevo nombre. Esto mismo ocurre al importar una columna a una estructura, tabla o plantilla en la cual ya existe una columna con ese nombre.

Si desea importar con dependencias tablas que tienen dependencias cruzadas y que no existen aún en el repositorio en edición, hágalo seleccionando dichas tablas simultáneamente, bien utilizando la selección múltiple haciendo clic con las teclas [MAYUSCULAS] o [CONTROL] pulsadas, bien seleccionando un grupo que las contenga.

Generar un Repositorio desde una Base de Datos

El Administrador de Repositorios permite generar un repositorio a partir del diccionario de una base de datos, pudiendo está ser local o remota. En consecuencia, el Administrador de Repositorios permite utilizar distintos gestores de bases de datos.

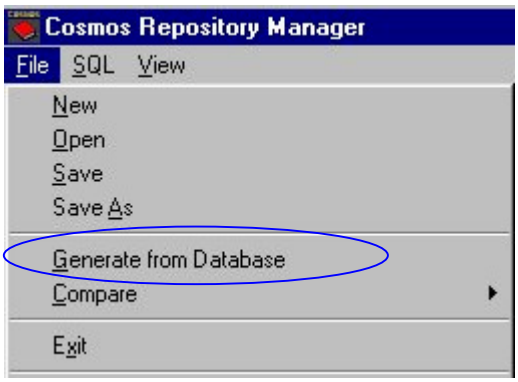


Figura 8.39. Menú File

Para generar un repositorio proceda de la siguiente manera:

1. Ejecute la opción *Generate from Database* del menú *File*.
2. Al ejecutar esta opción se muestra el cuadro de diálogo *Database Options* que permite seleccionar la base de datos desde la que se desea hacer la generación. Escriba en dicho cuadro de diálogo las opciones necesarias para seleccionar la base de datos.

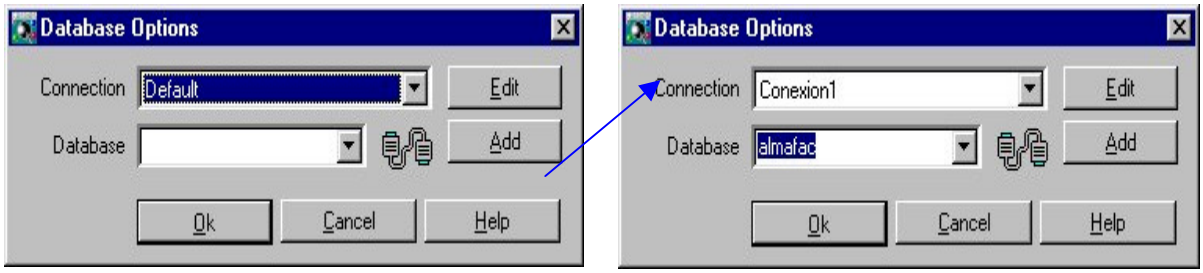


Figura 8.40. Cuadro de diálogo Database Options

3. Pulse el botón *Ok*.
4. Si los datos introducidos son correctos, automáticamente comienza la generación, en caso contrario se muestra el mensaje correspondiente. Mientras se genera el repositorio se muestra un cuadro de diálogo en el que se indica el porcentaje de tarea realizado y el nombre de la tabla de la base de datos para la que se está haciendo la generación, para las tablas con muchas columnas se indicará el desarrollo de la generación de las columnas de la tabla.

Figura 8.41. Progreso del proceso de generación

5. En la ventana principal de la aplicación se muestra el repositorio generado.

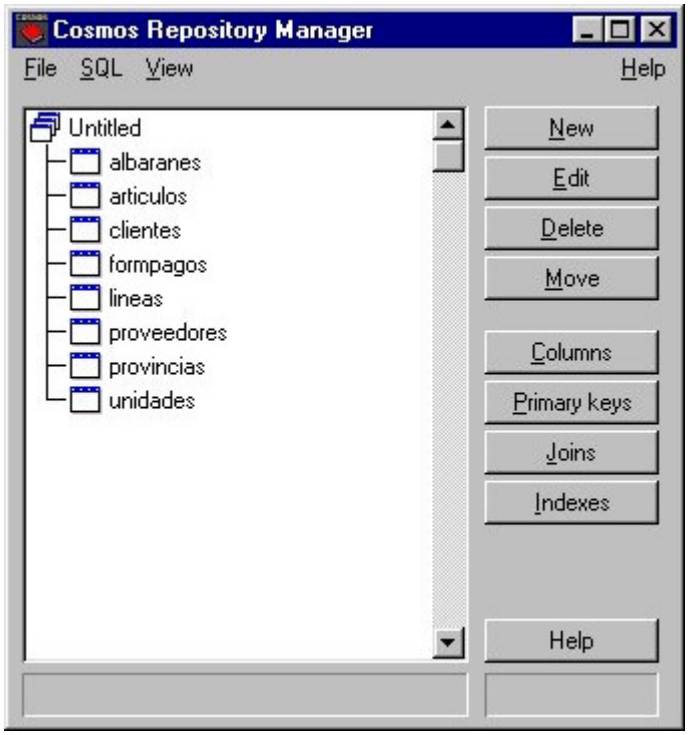


Figura 8.42. Repositorio generado

Comparar el Repositorio en edición con otro Repositorio o con una Base de Datos

El Administrador de repositorios permite comparar el repositorio en edición con otro repositorio seleccionado por el usuario.

El Administrador de repositorios permite comparar el repositorio en edición con una base de datos local o remota.

Para realizar la comparación ejecute las siguientes instrucciones:

1. Ejecute la opción *Compare* del menú *File*. En el submenú que se despliega seleccione la opción to *database o to Othen Repository*.

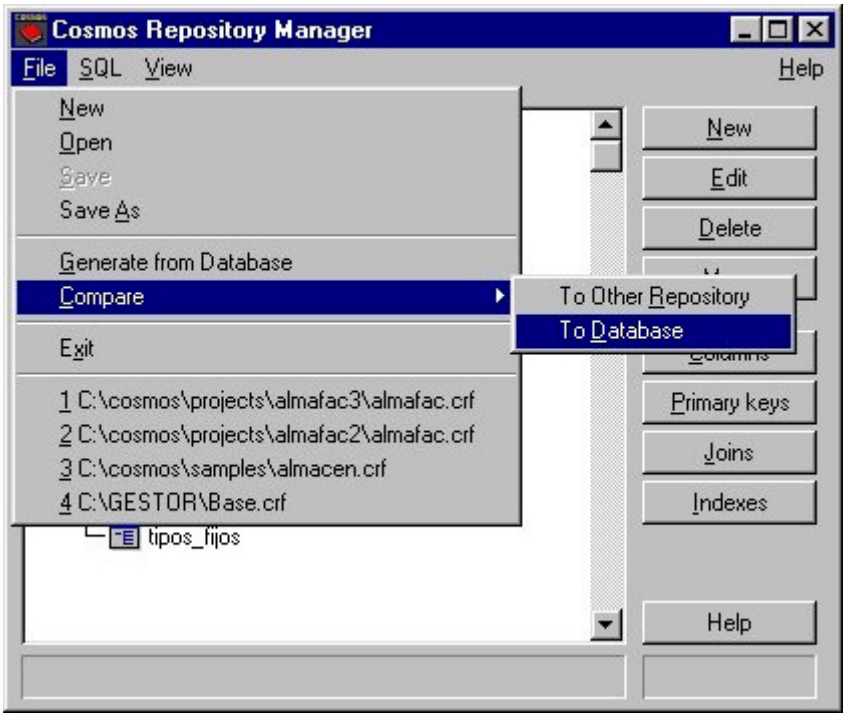


Figura 8. 43. Elección de Compare to Database

2. Se muestra el cuadro de diálogo *Database Options* que permite seleccionar la base de datos con la que se desea comparar el repositorio o un cuadro de diálogo idéntico al de la opción *Open* para seleccionar el repositorio con el que quiere comparar el que tiene en edición, dependiendo de la opción seleccionada.

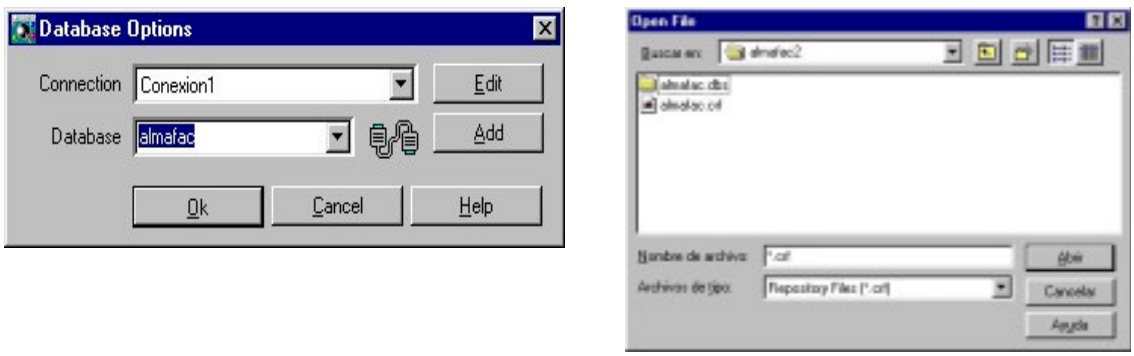


Figura 8.44. Elegir la Base de Datos o el Repositorio

3. Automáticamente comienza la lectura de la base de datos.

Figura 8.45. Lectura de la Base de Datos

Si la base de datos o el repositorio son equivalentes al repositorio en edición se muestra el mensaje correspondiente.

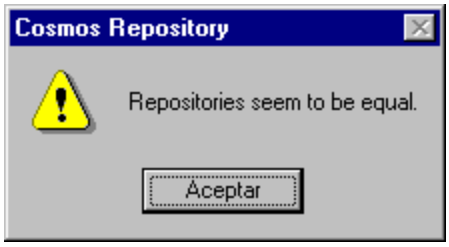


Figura 8.46. El repositorio y la Base de Datos son iguales

Si son distintos se muestra el cuadro de diálogo *Differences between repositorys* que muestra las diferencias entre ambos.

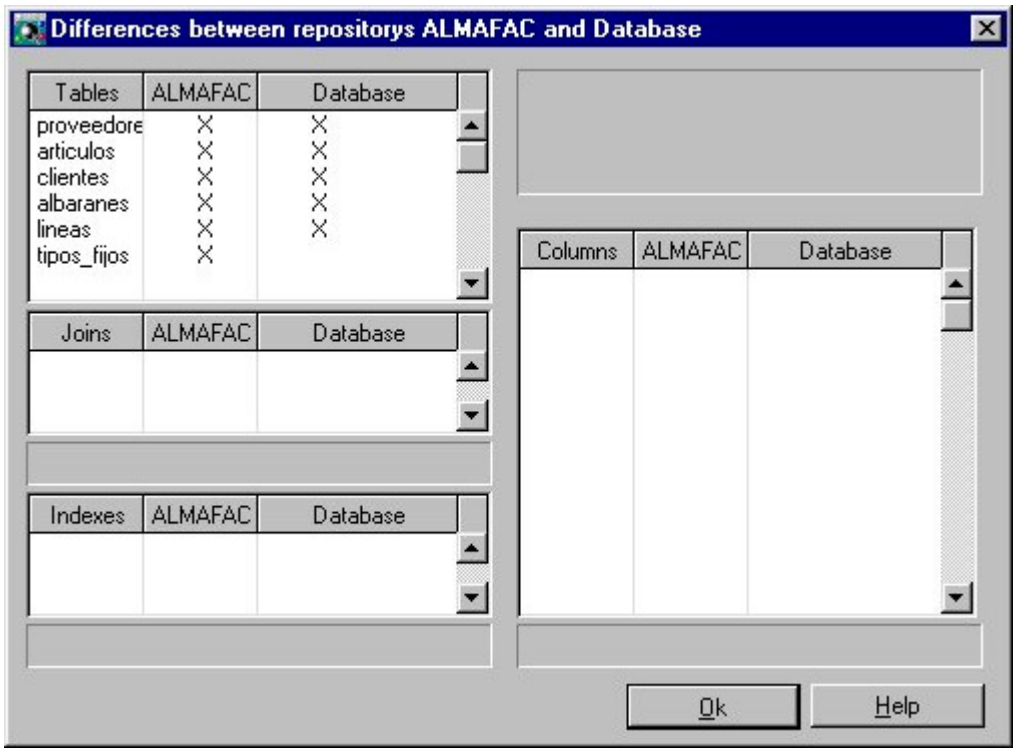


Figura 8.47. Diferencias entre el repositoro y la Base de Datos

Cuadro de diálogo Differences between repositorys

Este cuadro de diálogo muestra las diferencias entre dos repositorios. Se muestra el cuadro de diálogo que se compone de 4 cuadros comparativos entre los dos repositorios.

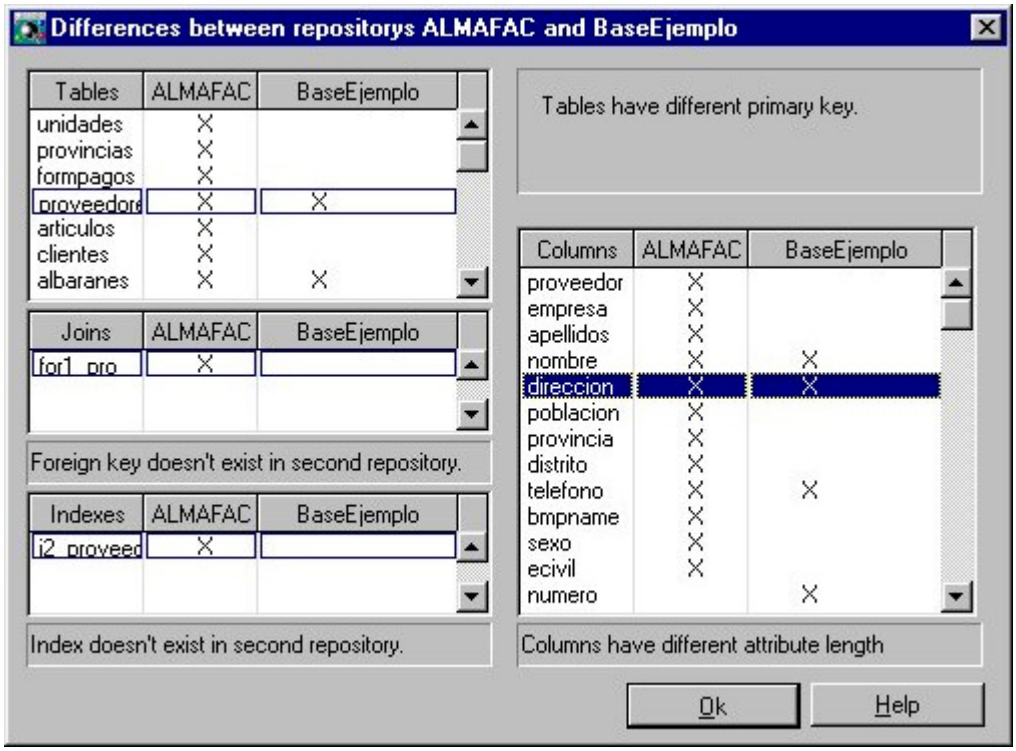


Figura 8.48. Cuadro de diálogo Differences between repositorys

Cuadro comparativo de tablas
Muestra la lista de tablas de los dos repositorios, en las que se han detectado diferencias indicando (con una X) para cada una de ellas el repositorio al que pertenecen.

Panel (esquina superior derecha)
Muestra la diferencia existente entre los dos repositorios para la tabla seleccionada en el cuadro comparativo de tablas.

Cuadro comparativo de Joins
Muestra la lista de foreign keys y joins de la tabla seleccionada en el cuadro comparativo de tablas, indicando (con una X) para cada una de ellas el repositorio al que pertenecen.

Panel (lado izquierdo)
Muestra la diferencia existente entre los dos repositorios para la foreign key o join que se tenga seleccionada en el cuadro comparativo de joins.

Cuadro comparativo de índices
Muestra la lista de índices la tabla seleccionada en el cuadro comparativo de tablas, indicando (con una X) para cada una de ellos el repositorio al que pertenecen.

Panel (esquina inferior izquierda)
Muestra la diferencia existente entre los dos repositorios para el índice que se tenga seleccionado en el cuadro comparativo de índices.

Cuadro comparativo de columnas
Muestra la lista de columnas de la tabla seleccionada en el cuadro comparativo de tablas, indicando (con una X) para cada una de ellas el repositorio al que pertenecen.

Panel (esquina inferior derecha)	Muestra la diferencia existente entre los dos repositorios para la columna que se tenga seleccionada en el cuadro comparativo de columnas.

Cuando existe más de una diferencia en posible que solo se muestre la primera que se detecta.

6. Ficheros generados y utilizados por el Administrador de Repositorios

El Administrador de repositorios de Cosmos utiliza y genera una serie de ficheros identificados por sus extensiones. Estas extensiones y su significado son las siguientes.

Extensión	Significado
*.crf	Fichero binario donde se guarda la estructura del repositorio
*.dbs	Directorio que representa la base de datos creada por el CTSQL. Las tablas de la base de datos se almacenarán en este directorio
*.dat	Fichero binario que contendrá los datos de una base de datos. La única forma de acceso a este fichero es mediante instrucciones del CTSQL. Este fichero tiene que estar siempre en el mismo directorio que su respectivo **.idx*..
*.idx	Fichero binario que contendrá la estructura de todos los índices relativos a una tabla de la base de datos. Este fichero tiene que estar siempre en el mismo directorio que su respectivo **.dat*.
*.scs	Estos ficheros se encargan de componer el orden de caracteres en la tabla ASCII para utilizarla en las operaciones de comparación y ordenación, y son asignados a la base de datos cuando esta se crea con la instrucción *CREATE DATABASE* y mediante la cláusula *COLLATING*
*.ocs	Ficheros objetos generados por el compilador de ficheros de ordenación (tcollcom)
*.sql	Ficheros ASCII que incluyen instrucciones del SQL.

No editar nunca los ficheros de extensión **.dat y *.idx*

Capítulo 9
El Editor Visual

Contenidos

1. Introducción

El entorno de desarrollo de Cosmos esta compuesto por un conjunto de herramientas o utilidades diseñadas cada una de ellas para asistir al proceso de elaboración de los componentes de una “aplicación Cosmos”. Todas ellas están embebidas o enlazadas a través del “Editor Visual”.

La unidad de trabajo en cualquier aplicación Cosmos es el proyecto. Un Proyecto se define por el conjunto de componentes necesarios para el desarrollo de una aplicación determinada.

Los componentes de un proyecto Cosmos se almacenan siempre en ficheros del sistema, cada uno de estos ficheros tiene una extensión determinada en función del tipo de componente que contiene. Cuando varios componentes están interrelacionados, éstos se almacenan juntos formando parte de otro componente de mas alto nivel. Así una clase es un componente que siempre forma parte del modulo donde esta definida. Llamamos documento al componente de mas alto nivel que se almacena en un fichero del sistema.

Para cada tipo de componente existe un editor especializado que nos permite su manejo (creación / modificación / borrado).

Los distintos tipos de documentos que forman un proyecto son :

- Fichero de Proyecto.
- Módulos de Programación.
- Repositorios de Datos.
- Ficheros de Mensajes.
- Fichero de Configuración.
- Fichero de Ayuda.
- Ficheros de Iconos.

El entorno de desarrollo proporciona una serie de editores gráficos que facilitan el diseño de todos aquellos componentes de la aplicación.

- Editor de proyecto.
- Editor de aceleradores de teclado.
- Editor de ficheros de mensajes.
- Editor de módulos.
- Editor de constantes.
- Editor de clases.
- Editor de objetos.
- Editor de variables.
- Editor de columnas de una tabla de un Form.
- Editor de código.
- Editor de screen.
- Editor de menús.
- Editor de páginas de impresión.

Hay dos formas de ejecutar el editor Visual de COSMOS:

- mediante la instrucción

cosmos [-v] [nombre_proyecto]

Opción	Significado
-v	Muestra la versión del Editor Visual
-n	Este parámetro ignora el último proyecto editado, presentando vacía la pantalla del Editor Visual
nombre_proyecto	Nombre del proyecto que se desea editar

- mediante el icono 

2. Editor de Proyecto

La unidad de trabajo en cualquier aplicación, hecha desde este entorno de desarrollo es el proyecto. Dentro de un proyecto se almacenan sus módulos, repositorios, ficheros de mensajes, ficheros de configuración, etc. El Editor Visual permite la creación de componentes y objetos de un proyecto empleando la técnica de “arrastrar y soltar” (drag & drop). Los elementos y los objetos se incorporan desde sus paletas o repositorios al programa en construcción por medio del ratón.

El Editor Visual dispone de todas las herramientas necesarias para la edición de cualquier componente software a incluir en un proyecto. Edición del propio proyecto, de módulos, de repositorios o edición de iconos, de menús, formatos de pantalla o impresora, son algunas de sus capacidades de edición gráfica.

Los pasos típicos que se deben seguir para desarrollar una aplicación en Cosmos son:

1. Crear un proyecto.
2. Organizar el proyecto
3. Importar un repositorio de datos.
4. Creación y gestión de módulos.
5. Creación de las clases del módulo.
6. Creación de los objetos del módulo.
7. Escritura del código fuente.

Crear un proyecto

El primer paso para el desarrollo de una aplicación con Cosmos será crear el proyecto. Para ello, proceda de acuerdo a los siguientes pasos:

1. Ejecute la opción New del menú File. Alternativamente podrá emplear el botón New del menú de iconos o la combinación de teclas [Ctrl]+[N]. En caso de estar trabajando con un proyecto, la forma de ejecución de esta opción dependerá de las modificaciones realizadas, pidiendo conformidad o no al usuario para guardar los cambios antes de generar el nuevo proyecto.

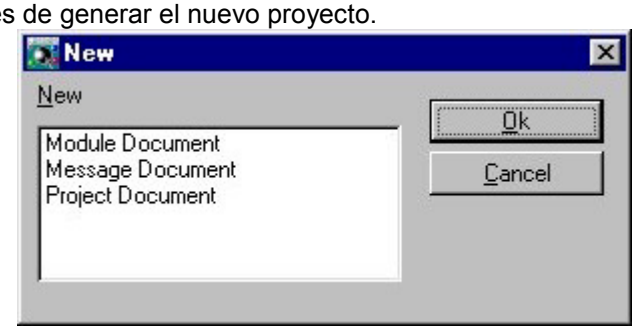


Figura 22.1. Crear un proyecto

2. Aparecerá el cuadro de diálogo New Proyect en el que se solicitan el nombre del proyecto y su directorio de ubicación.
   En este cuadro de diálogo se piden los datos necesarios para crear un proyecto:

Name	Nombre del proyecto.
Location	Directorio de ubicación del proyecto.
Browse...	Permite cambiar el nombre y el directorio del proyecto. Al pulsar este botón se muestra un cuadro de diálogo idéntico al de la opción Save Copy As ....

   Introduzca el nombre del proyecto, su directorio de ubicación y pulse el botón Create .

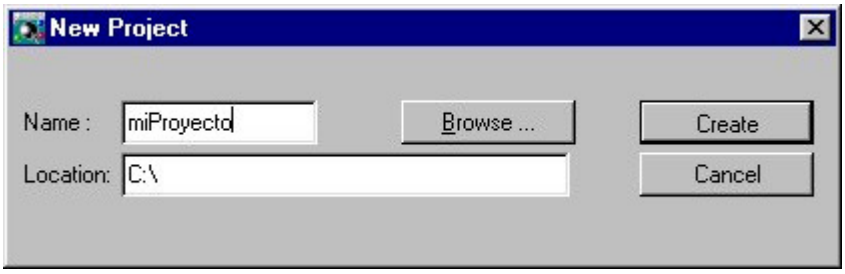


Figura 22.2. Cuadro de diálogo New Project

3. Al crear el proyecto aparecerá una ventana con el nombre asignado, similar a la que se muestra en la figura.

Figura 22.3. Aspecto del Editor de Proyecto

En esta ventana se puede ver la estructura en árbol del proyecto con todos los elementos que lo componen:

- Fichero de ayuda.
- Fichero de configuración.
- Aceleradores
- Repositorios.
- Módulos.
- Objetos externos.
- Ficheros de mensajes.

En la estructura del proyecto es posible elegir entre mostrar las etiquetas o los nombres de los nodos ejecutando la opción Show node labels del menú View. Esta selección se guardará para sesiones posteriores. Si se elige mostrar las etiquetas (opción por defecto), al seleccionar un nodo se mostrará su nombre en la barra de estado. Si se elige mostrar los nombres, al seleccionar un nodo se mostrará su etiqueta en la barra de estado.

Añadir un documento al proyecto

En un proyecto pueden añadir:

- Grupos.
- Módulos.
- Objetos externos.
- Ficheros de mensajes.

Para añadir un elemento de los indicados anteriormente:

1. Seleccione el nodo del árbol (marcado con el icono o) asociado al tipo de documento que desea crear.
2. En este segundo paso tiene las siguientes posibilidades:
   - Pulse [Ctrl] + [A].
   - Ejecute la opción Add del menú Edit.
   - Ejecute la opción Add del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el nodo que tiene seleccionado.
3. Se muestra el cuadro de diálogo Node Properties que permite definir las características del documento.

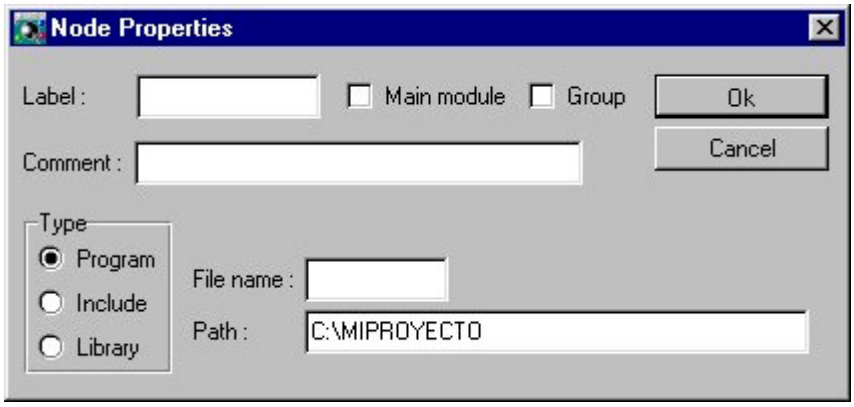


Figura 22.4. Cuadro de diálogo de Propiedades de Nodo

4. El documento o grupo añadido se mostrará automáticamente en la estructura en árbol del proyecto.

Cuadro de diálogo Node Properties

Los elementos que se muestran en el cuadro de diálogo son los que se indican a continuación:

Label Nombre identificador del documento o grupo.

Group Indica si el nodo es un grupo o un documento. Si se marca esta casilla de verificación solo quedan visibles los campos necesarios para definir un grupo: Label y Comment.

Comment Texto que se mostrará en el panel de comentarios de la barra de estados del editor visual.

Type Permite indicar si el módulo a desarrollar es un programa, un include o una librería. Se permite modificar el tipo de un módulo si no esta cargado en memoria.

- Program El módulo es un programa. Es la opción seleccionada por defecto.
- Include El módulo es un include.
- Library El módulo es una librería.

File name Nombre físico del documento sin su extensión.

Browse... Se muestra el mismo cuadro de diálogo de la opción Open para seleccionar el directorio de ubicación del documento. Este campo solo estará visible cuando se importa un documento de otro proyecto.

Path Directorio de ubicación del documento. Por defecto será el mismo directorio donde se encuentra el proyecto.

Main module Si se marca esta casilla de verificación, se define el módulo (programa, librería o include) en curso como módulo principal de inicio de la aplicación desarrollada en el proyecto. Solo puede existir un módulo principal en el proyecto.

Al importar un documento de otro proyecto también se muestran los siguientes campos:

Link Incorpora el módulo al proyecto conservando el directorio en el que estuviera ubicado.

Copy Incorpora el módulo al proyecto copiándolo en el directorio del proyecto.

Insertar un documento en el Proyecto

En un proyecto pueden insertar:

- Grupos.
- Módulos
- Objetos externos
- Ficheros de mensajes.

Para insertar un elemento de los indicados anteriormente:

1. Seleccione el documento de la estructura árbol delante del cual desea insertar uno nuevo.
2. En este segundo paso tiene las siguientes posibilidades:
   - Pulse [Insert].
   - Ejecute la opción Insert del menú Edit .
   - Ejecute la opción Insert del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el nodo que tiene seleccionado.
3. Se muestra el cuadro de diálogo Node Properties que permite definir las características del documento.
4. El documento añadido se mostrará automáticamente en la estructura en árbol del proyecto.

Borrar un documento del Proyecto

El Editor Visual permite borrar un grupo o cualquier tipo de documento que este incluido en el proyecto:

- Grupos.
- Repositorios.
- Módulos
- Objetos externos
- Ficheros de mensajes.

Para borrar un documento o un grupo proceda de la siguiente manera:

1. Seleccione en la estructura en árbol del proyecto el elemento que desea borrar.
2. En este segundo paso tiene las siguientes posibilidades:
   - Pulse [Del].
   - Ejecute la opción Delete del menú Edit .
   - Ejecute la opción Delete del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el elemento seleccionado.
3. Si se puede proceder al borrado, se elimina el elemento del proyecto . Si se esta borrando un documento se pide conformidad para borrar además el fichero físicamente.

Importar un documento al Proyecto

Cosmos permite importar documentos. Es decir, se pueden incluir documentos ya existentes en el proyecto. Estos documentos podrán ser comunes a varios proyectos. si se marca la opción Link del cuadro de diálogo Node Properties o se hará una copia del documento en el directorio del proyecto si se marca la opción Copy.

En un proyecto se pueden importar:

- Repositorios.
- Módulos
- Objetos externos
- Ficheros de mensajes.

Para importar un documento al proyecto proceda de la siguiente manera:

1. Seleccione en la estructura en árbol del proyecto el nodo asociado al tipo de documento que desea importar o seleccione el documento delante del cual desea insertar el nuevo documento.
2. En este segundo paso tiene las siguientes posibilidades:
   - Ejecute la opción Import del menú Edit .
   - Ejecute la opción Import del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el nodo que tiene seleccionado.
3. Se muestra el cuadro de diálogo Node Properties que permite seleccionar el documento que se desea importar.
4. Si el documento seleccionado ya está en el proyecto, se muestra el mensaje correspondiente sin realizar ninguna acción. En caso contrario, el documento importado se mostrará automáticamente en la estructura en árbol del proyecto.

Editar un documento de un Proyecto

Esta opción permite editar cualquier tipo de documento que este incluido en el proyecto:
- Fichero de ayuda.
- Fichero de configuración.
- Repositorios.
- Módulos
- Ficheros de mensajes.

Para editar un documento del proyecto proceda de la siguiente manera:

1. Seleccione en la estructura en árbol del proyecto el documento que desea editar.
2. En este segundo paso tiene las siguientes posibilidades:
   - Pulse [Ctrl] + [E].
   - Ejecute la opción Edit del menú Edit .
   - Ejecute la opción Edit del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el elemento seleccionado.
   - Haga doble clic sobre el elemento seleccionado.
3. Automáticamente se muestra en una ventana el contenido del documento seleccionado.

Editar las propiedades de un documento del proyecto

Se puede editar las propiedades de un grupo o de cualquier tipo de documento que este incluido en un proyecto:

- Fichero de ayuda.
- Fichero de configuración.
- Repositorios.
- Módulos
- Objetos externos
- Ficheros de mensajes.

Para editar las propiedades de un documento o un grupo proceda de la siguiente manera:

1. Seleccione en la estructura en árbol del proyecto el elemento que desee.
2. En este segundo paso tiene las siguientes posibilidades:
   - Pulse [Alt] + [Enter].
   - Ejecute la opción Properties del menú Edit .
   - Ejecute la opción Properties del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el elemento seleccionado.

3. Se muestra el cuadro de diálogo Node Properties que permite ver y modificar las propiedades del elemento.

Buscar un nodo en el Proyecto

El Editor Visual permite buscar un nodo en el proyecto.
Para localizar un nodo en el proyecto proceda de la siguiente manera:

1. Seleccione en la estructura en árbol del proyecto el nodo a partir del cual desea realizar la búsqueda.
2. En este segundo paso tiene las siguientes posibilidades:
   - Pulse [Ctrl] + [F].
   - Ejecute la opción Find del menú Edit.
   - Ejecute la opción Find del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el elemento seleccionado.
3. Se muestra el cuadro de diálogo Find Node que permite introducir el nombre o la etiqueta del nodo que desea buscar, teniendo la posibilidad de repetir la búsqueda sin salir del cuadro de diálogo.

La búsqueda no distingue entre mayúsculas y minúsculas. No es necesario especificar el nombre o la etiqueta completa, seleccionará el primer nodo cuyo nombre o etiqueta comience por la cadena especificada.

Abrir un Proyecto

La opción Open del menú File permite abrir un proyecto. Al igual que ocurría en la opción New, su ejecución dependerá de las posibles modificaciones que se hubiesen realizado en el documento activo y de si se trata de un documento nuevo o ya existente a efectos de solicitar la salvaguarda de los posibles cambios.

Si al ejecutar esta opción tiene algún proyecto activo podrá abrir cualquier tipo de documento incluido en el proyecto en caso contrario solo se le permitirá abrir algún proyecto creado con anterioridad.

Esta opción también se puede ejecutar pulsando el botón  en el menú de iconos o la tecla [Ctrl] + [O].

Guardar un Proyecto o un documento

La opción Save del menú File guarda el contenido del proyecto o del documento activo. Por ejemplo, si se tiene activa la ventana de una screen, se guarda el módulo a la cual pertenece.

Esta opción también se puede ejecutar pulsando el botón  del menú de iconos o la tecla [Ctrl] + [S].

Guardar una copia del proyecto o de un documento

La opción Save Copy As del menú File permite guardar una copia del proyecto o del documento en curso asignándole otro nombre.

Si el nombre asignado al documento o al proyecto coincidiese con el de algún otro ya existente, el sistema avisaría al usuario presentando un mensaje en pantalla, en cuyo caso el usuario podrá elegir entre:

- Guardar el documento o el proyecto con el mismo nombre (Sobre-escribir), con lo cual se grabarían las modificaciones que hubieran podido realizarse sobre él (equivalente a la opción Save). En este caso el sistema pediría confirmación avisando de que se trata de un documento o proyecto ya existente que se va a sobreescribir.
- Asignar un nombre distinto al actual, con lo cual estaríamos generando un nuevo documento o proyecto
- Cancelar la operación.

La forma de ejecutar esta opción es equivalente a la opción Save as de cualquier aplicación. La diferencia estriba en que después de ejecutar esta opción, se sigue trabajando con el documento o proyecto original, no se trabaja con la copia generada. En la opción Save As , cuando se asigna al documento un nombre distinto al actual, se comienza a trabajar con el documento que se acaba de crear.

Especificar impresora

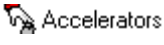
La opción Print Setup del menú File permite seleccionar como impresora de salida una distinta de la definida por defecto en el Administrador de Impresión de Windows.

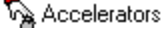
En aquellos casos en los que se desee emplear una impresora distinta de la definida por defecto, es aconsejable ejecutar esta opción antes de imprimir al objeto de que los ajustes del documento (márgenes, tipos de letra disponibles, etc.) se adecuen a las características de la nueva impresora, evitando de esta forma sorpresas desagradables a la hora de imprimir (saltos de página no deseados, problemas con las fuentes de letra, etc.).

3. Editor de Aceleradores de Teclado

Los aceleradores son combinaciones de teclas para la llamada rápida al proceso asociado (comando) a un ítem de un menú o a un botón de comando. Están contempladas las teclas [Ctrl] + [tecla], las teclas de función hasta la [F12], en combinación o no con [Ctrl] y/o [Mayúsculas] y demás teclas auxiliares como [Del], etc. El literal de la abreviatura del teclado aparece a continuación del ítem del menú al que se le ha asociado.

Para especificar un método abreviado del teclado:

1. En la estructura en árbol del proyecto seleccione el nodo 

.
2. En este paso tiene las siguientes opciones:
   - Pulse [Ctrl] + [E].
   - Ejecute la opción Edit del menú Edit .
   - Ejecute la opción Edit del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el nodo 

..
   - Haga doble clic con el botón izquierdo de ratón.
3. Automáticamente aparece el editor de Aceleradores de teclado que muestra una lista multicolumna con métodos abreviados del teclado que se tienen definidos en el proyecto.

Figura 22.5. Cuadro de Aceleradores de Teclado

Editor de Aceleradores de Teclado

Este editor permite añadir, borrar y modificar los aceleradores de teclado para los comandos definidos.

Este editor solo consta de una lista multicolumna que muestra el nombre del comando y la combinación de teclas asociada.

Para añadir un acelerador de teclado proceda de la siguiente manera:

1. En este paso tiene las siguientes posibilidades:
   - Pulse [Ctrl] + [A].
   - Ejecute la opción Add del menú Edit .
   - Ejecute la opción Add del menú tipo popup que aparece al hacer clic con el botón derecho del ratón en este editor.
2. Automáticamente se muestra el cuadro de diálogo Accelerator properties que permite definir los aceleradores de teclado.
3. Introduzca los valores deseados en el cuadro de diálogo y pulse el botón Ok , automáticamente queda añadido el acelerador en este editor:

Para modificar un acelerador de teclado proceda de la siguiente manera:

1. Seleccione en el editor el acelerador de teclado que desee modificar.
2. En este segundo paso tiene las siguientes posibilidades:
   - Pulse [Alt]+[Enter].
   - Ejecute la opción Properties del menú Edit .
   - Ejecute la opción Properties del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el acelerador.
   - Haga doble clic sobre el acelerador.
3. Se muestra el cuadro de diálogo Accelerator properties que permite modificar la definición del acelerador.

Para borrar un acelerador de teclado proceda de la siguiente manera:

1. Seleccione en el editor el acelerador que desea borrar.
2. Pulse [Del] o ejecute la opción Delete del menú Edit o la opción Delete del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el acelerador.

Cuadro de diálogo Acelerators Properties

Este cuadro de diálogo permite definir los aceleradores de teclado para los comandos definidos.

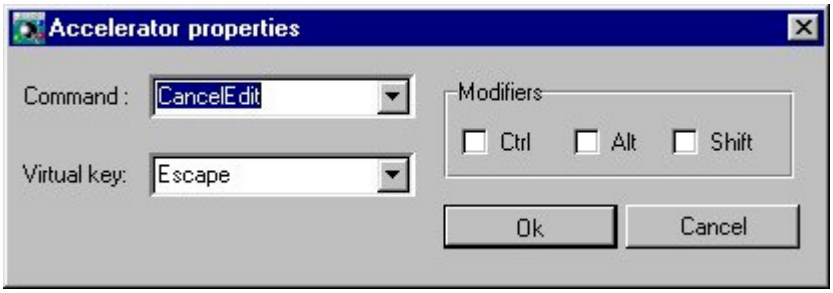


Figura 22.6. Cuadro de diálogo Acelerator Properties

Los campos del cuadro de diálogo son los siguientes:

Command	Nombre del comando para el cual se define el acelerador.
Virtual Key	Identificador de la tecla que formará parte del acelerador. Muestra una lista de los identificadores standard de las teclas virtuales.
Modifiers	Permite indicar si el acelerador es una combinación formada con la tecla [Ctrl], [Alt], o [Shift] y la indicada en el campo Virtual Key .

4. Ficheros de mensajes

Los ficheros de mensajes son ficheros ASCII que permiten definir textos fuera de los módulos de programación. Estos textos podrán ser utilizados posteriormente en la aplicación.

Los ficheros de mensajes se pueden editar con cualquier editor de textos ASCII, debiendo indicarles la extensión .shl . La estructura de este tipo de ficheros es la siguiente:

```
.[número | COMMENTS]
líneas de texto con el mensaje asociado al número o comentario.
.[número | COMMENTS]
texto ...
...
```

Una vez escritos los números y sus textos asociados, para que el fichero sea utilizable desde un programa debe compilarse mediante la opción Compile del menú Tools al tener el foco de entrada en el editor de ficheros de mensajes en el Editor Visual. Esta opción emplea el comando cosmsg para la compilación. Dicho comando producirá un fichero homónimo pero con extensión .ohl .

En COOL se considera mensaje cualquier texto constante que se produzca en el curso de la ejecución de un programa, ya sea por pantalla, impresora o un fichero de resultados. Estos textos se corresponden con cualquier cadena entrecomillada en el fuente del programa. Por lo tanto, las instrucciones del lenguaje COOL que empleen dichos literales podrán utilizar los mensajes de un fichero. La clase Module dispone de un método MsgText que indicándole el número del mensaje y el fichero, extraerá el texto asignado a dicho número en el fichero de mensajes indicado.

Añadir un Fichero de Mensajes al Proyecto

Los ficheros de mensajes son ficheros ASCII que permiten definir textos fuera de los módulos de programación. Estos textos podrán ser utilizados posteriormente en la aplicación.

Los ficheros de mensajes se pueden editar con cualquier editor de textos ASCII, debiendo indicarles la extensión .shl .

Para añadir un fichero de mensajes al proyecto proceda de la siguiente manera.

1. En la estructura en árbol del proyecto, seleccione el nodo del árbol “Message files”.
2. En este segundo paso tiene las siguientes posibilidades:
   - Pulse [Ctrl] + [A].
   - Ejecute la opción Add del menú Edit .
   - Ejecute la opción Add del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el nodo que tiene seleccionado.
3. Se muestra el cuadro de diálogo Node properties que permite definir las características del documento. Introduzca los datos deseados y pulse el botón Ok .
4. El fichero de mensajes añadido se mostrará en la estructura en árbol del proyecto. Para editarlo, pulse dos veces (haciendo “dobleclic”) con el ratón sobre él. Automáticamente aparece el editor de ficheros de mensajes.

5. Editor de Módulos

Edición de Módulos

Una aplicación Cosmos esta compuesta de módulos de programación. Un módulo es una unidad de ejecución que contiene componentes (clases, objetos, etc.) que definen el funcionamiento de la aplicación.

Al editar un módulo. se muestra una ventana con la estructura en árbol del mismo similar a la que se muestra a continuación:

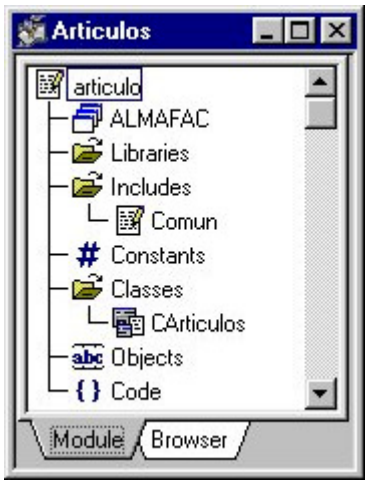


Figura 22.7. Estructura de un módulo

Un módulo puede contener los siguientes elementos:

Fichero de mensajes	Fichero de mensajes definido en el proyecto que utilizará el módulo por defecto.
Repositorio	Repositorio utilizado por el módulo. Debe estar definido en el proyecto.
Librerías	Librerías del proyecto con las que enlaza el módulo. Solo se pueden editar desde la ventana del proyecto.
Includes	Includes del módulo previamente definidos en el proyecto.
Constantes	Constantes definidas en el módulo. Las constantes son valores fijos que el programa no puede alterar.
Clases	Clases definidas en el módulo. El elemento base del diseño orientado a objetos es la clase. Una clase equivale a la generalización o abstracción de un tipo específico de objetos. Una clase es una colección de objetos que poseen los mismos atributos, presentan el mismo comportamiento, siguen el mismo conjunto de reglas y tienen idénticas relaciones con otros objetos En un módulo podrá:
	Crear e insertar nuevas clases: El programador solo puede definir clases derivadas de las clases predefinidas o de las clases definidas en el módulo o en sus includes.
	Borrar una clase: Se permite borrar una clase definida por el usuario. No se puede borrar una clase predefinida de Cosmos.
	Editar las propiedades: Propiedades de las clases definidas por el usuario para modificar su definición.
Objetos	Los objetos son instancias de las clases predefinidas o de las clases definidas por el usuario en el módulo y en sus includes. En un módulo podrá crear borrar e insertar objetos y editar sus propiedades.
Código	El módulo en sí es un objeto que pertenece a una clase derivada de la clase predefinida Module y por tanto puede escribir el código fuente de sus métodos.
Browser	El browser es un examinador que se puede utilizar para visualizar las clases y los métodos disponibles en el módulo en curso. También proporciona una forma fácil de desplazarse rápidamente de una parte a otra del código. Es especialmente útil para proyectos grandes. Haga clic con el botón izquierdo del ratón sobre esta ficha de la ventana para ver la jerarquía de clases y los métodos definidos en cada una de las clases.

En Cosmos cada estructura de datos está representada por una clase. Un módulo (programa, librería, include) se puede considerar como una clase derivada de la clase abstracta Module a la que añade nuevos métodos y atributos.

Utilizando la técnica de arrastrar y soltar se pueden copiar clases creadas por el programador de un módulo a otro del proyecto. Esto le permite utilizar las clases como plantillas para la creación de otras nuevas. Después de copiarla, puede modificarla y añadirle nuevos métodos. Al arrastrar una clase a un módulo, tenga en cuenta que se copia su interfaz y su implementación (sus métodos).

Definir un Repositorio en un Módulo

No es obligatorio definir un repositorio en un módulo. Su función es determinar que repositorio se utilizará en compilación. Su definición es obligatoria en el momento que se define cualquier objeto del módulo mediante la cláusula Like table column . Asimismo, será necesaria su definición cuando el módulo tenga una clase Form que tenga definidas variables o columnas de una tabla mediante las cláusulas Like table column ó Is column En este caso el Form estará relacionado directamente con el repositorio.

Un módulo solo puede tener asociado un repositorio de los que se han Importado en el proyecto.

Es necesario asociar un repositorio a un módulo, cuando en dicho módulo se quiere acceder a tablas o columnas definidos en un repositorio o por ejemplo para crear un Form de mantenimiento de una o más tablas del repositorio.

Para definir un repositorio en un módulo ejecute los siguientes pasos:

1. Asocie el repositorio al proyecto, si no lo ha hecho aún, ejecutando la opción Import . Un proyecto puede tener asociado más de un repositorio
2. Seleccione en la estructura en árbol del módulo el nodo raíz .
3. Ejecute la opción Properties del menú Edit .
4. El cuadro de diálogo Module Properties muestra una lista de los ficheros de mensajes y repositorios del proyecto. Seleccione el repositorio y pulse el botón Ok .

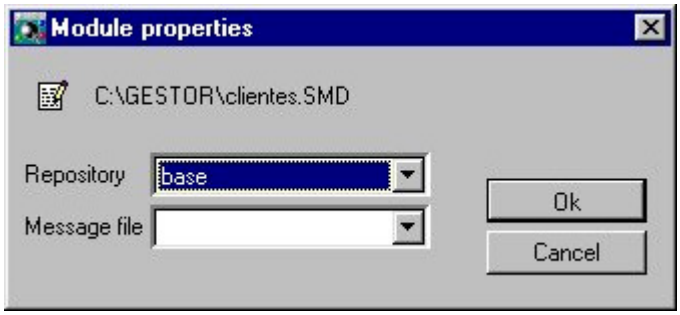


Figura 22.8. Cuadro de diálogo Module Properties

5. Automáticamente se añade el repositorio a la estructura en árbol del módulo.

Definir un Fichero de Mensaje al Módulo

No es obligatorio definir un fichero de mensajes en un módulo. Su función es determinar que fichero de mensajes utilizará por defecto.

Un módulo solo puede tener asociado un fichero de mensajes de los que se han definido en el proyecto.

Para poder asociar un fichero de mensajes a un módulo siga las instrucciones que se indican a continuación:

1. Añada el fichero de mensajes al proyecto , si no lo ha hecho aún.
2. Seleccione en la estructura en árbol del módulo el nodo raíz .
3. Ejecute la opción Properties del menú Edit .
4. El cuadro de diálogo Module Properties muestra una lista de los ficheros de mensajes y repositorios asociados al proyecto. Seleccione el fichero de mensajes y pulse el botón Ok .
5. Automáticamente se añade el fichero de mensajes a la estructura en árbol del módulo.

Añadir un Include o una librería a un módulo

Previamente deberá estar definida la librería o el include en el proyecto al que pertenece el módulo. Si desea añadir una librería o un include de otro proyecto utilice la opción Import .

Para añadir una librería o un include al modulo en curso proceda de la siguiente manera:

1. Seleccione el nodo del árbol (marcado con el icono  o ) asociado a las librerías o includes del módulo.
2. En este segundo paso tiene las siguientes posibilidades:
   - Pulse [Ctrl] + [A].
   - Ejecute la opción Add del menú Edit .
   - Ejecute la opción Add del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el nodo que tiene seleccionado.
3. Se muestra el cuadro de diálogo Library / Include Properties que permite seleccionar la librería o include que se desea añadir al módulo en curso. Seleccione un elemento de la lista y pulse el botón Ok .
4. El módulo añadido se mostrará automáticamente en la estructura en árbol del módulo.

Editar un documento del módulo

Desde el editor de módulos se permiten editar los documentos incluidos en el módulo del tipo:

- Fichero de mensajes.
- Repositorio.
- Includes.

Desde un módulo no se pueden editar las librerías que tiene incluidas. Se tienen que editar desde la ventana del proyecto, esto es así porque las librerías no se cargan en memoria al editar un módulo que las incluye, mientras que los includes si.

Para editar un documento incluido en el módulo proceda de la siguiente manera:

1. Seleccione en la estructura en árbol del módulo el documento que desea editar.
2. En este segundo paso tiene las siguientes posibilidades:
   - Pulse [Ctrl] + [E].
   - Ejecute la opción Edit del menú Edit .
   - Ejecute la opción Edit del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el elemento seleccionado.
   - Haga doble clic sobre el elemento seleccionado.
3. Automáticamente se muestra una ventana que muestra el contenido del documento seleccionado.

Borrar un documento del Módulo

El Editor Visual permite borrar cualquier tipo de documento que este incluido en un módulo:

- Ficheros de mensajes.
- Repositorios.
- Librerías.
- Includes.

Para borrar un documento proceda de la siguiente manera:

1. Seleccione en la estructura en árbol del módulo el elemento que desea borrar.
2. En este segundo paso tiene las siguientes posibilidades:
   - Pulse [Del].
   - Ejecute la opción Delete del menú Edit .
   - Ejecute la opción Delete del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el elemento seleccionado.
3. Si se puede proceder al borrado, se elimina el documento del módulo.

No se permite borrar un include del módulo si se hace referencia a alguna de sus clases.

Browser

El browser es un examinador que se puede utilizar para visualizar las clases y los métodos disponibles en el módulo en curso. También proporciona una forma fácil de localizar rápidamente el código asociado a las clases.

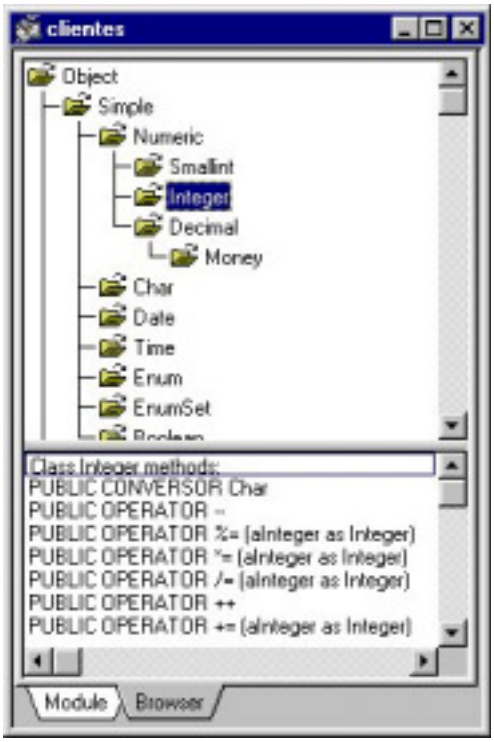


Figura 22.9. Aspecto del Browser

6. Editor de Clases

Dado que Cosmos utiliza un lenguaje de programación orientado a objetos, al añadir una clase se esta creando una clase derivada cuya clase base será una de las que hay predefinidas en Cosmos u otra de las que han sido definidas por el usuario.

En Cosmos la herencia no permite añadir nuevas características o atributos a los objetos. En las clases heredadas solo se permite definir nuevos métodos.

Para añadir una clase al módulo proceda de la siguiente manera:

1. En este paso tiene las siguientes posibilidades:
   - Seleccione el nodo Classes de la estructura en árbol (marcado con el icono o) del módulo.
   - Seleccione la clase base en la jerarquía de clases de la ventana del browser.
2. En este segundo paso tiene las siguientes posibilidades:
   - Pulse [Ctrl] + [A].
   - Ejecute la opción Add del menú Edit .
   - Ejecute la opción Add del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el nodo que tiene seleccionado.
3. Se muestra el cuadro de diálogo Class Properties que permite definir las propiedades de la clase.
4. La clase añadida se mostrará automáticamente en la estructura en árbol del módulo.

Para insertar una clase proceda de la siguiente manera:

1. Seleccione en el nodo Classes de la estructura en árbol (marcado con el icono o) del módulo la clase delante de la cual desea insertar una nueva.
2. En este segundo paso tiene las siguientes posibilidades:
   - Pulse [Insert].
   - Ejecute la opción Insert del menú Edit .
   - Ejecute la opción Insert del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el elemento que tiene seleccionado.
3. Se muestra el cuadro de diálogo Class Properties que permite definir las propiedades de la clase.
4. La clase insertada se mostrará automáticamente en la estructura en árbol del modulo.

Solo se permite insertar una clase derivada de una clase predefinida o de una clase definida por el usuario situada por encima de la nueva clase en la estructura en árbol del módulo.

Cuadro de diálogo Class Properties

En este cuadro de diálogo se definen las propiedades de las clases.

La descripción de cada uno de los campos que aparecen en el cuadro de diálogo es la siguiente:

Campo	Significado	
Name	Nombre de la clase. Este campo es obligatorio. En un módulo no pueden existir dos clases con el mismo nombre.	
Parent class	Muestra una lista de las clases predefinidas y de las clases creadas por el usuario. Permite seleccionar la clase base de la clase en edición	
Access	Indica la visibilidad de la clase fuera del módulo en el que se ha declarado. Puede ser de tres tipos	
	Public	Una clase declarada como pública siempre es derivable y será instanciable si no se ha definido como abstracta)
	Private	Una clase privada solamente puede ser utilizada dentro del módulo en el que se ha declarado. Es decir, esta clase es derivable e instanciable (si no se ha definido como abstracta) en el módulo en el que se ha declarado
	Protected	Una clase declarada como protegida será instanciable (si no se ha definido como abstracta) y no derivable fuera del módulo

Campo	Significado
Table	Lista de las tablas definidas en el repositorio asociado al módulo. Este campo sólo esta visible cuando se marca la casilla de verificación Like table column o Like table (para el caso de la clase struct) y será obligatorio seleccionar una tabla en ese caso
Column	Permite seleccionar el nombre de una columna de la tabla indicada anteriormente cuyo tipo de dato y algunos atributos del SQL (longitud, precisión, etc.) se asignarán a la clase que se esta definiendo. Este campo solo esta visible cuando se marca la casilla de verificación Like table column en cuyo caso será obligatorio seleccionar una columna
Default	Valor con el que se inicializarán los objetos de la clase. Este campo solo esta visible cuando la clase base es simple

Dependiendo de la clase base seleccionada, en este cuadro de diálogo se piden también otros parámetros necesarios para definir dicha clase.

Borrar una clase de un módulo

No se pueden borrar las clases predefinidas, solo se pueden borrar las clases creadas por el usuario.

Para borrar una clase de un módulo proceda de la siguiente manera:

1. En este paso tiene las siguientes posibilidades:
   - Seleccione en el nodo Classes de la estructura en árbol del módulo la clase que desee borrar.
   - Seleccione la clase que desee borrar en la jerarquía de clases de la ventana del browser.
2. En este segundo paso tiene las siguientes posibilidades:
   - Pulse [Del].
   - Ejecute la opción Delete del menú Edit .
   - Ejecute la opción Delete del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el nodo que tiene seleccionado.
3. Si la clase seleccionada está refererenciada, se muestra el mensaje correspondiente y no se procede al borrado. En caso contrario la clase es eliminada automáticamente del módulo.

Editar las propiedades de una clase

El Editor Visual permite editar las características principales de las clases definidas por el usuario.

Para editar las propiedades de una clase proceda de la siguiente manera:

1. En este paso tiene las siguientes posibilidades:
   - Seleccione en el nodo Classes de la estructura en árbol del módulo la que desee editar.
   - Seleccione la clase que desee editar en la jerarquía de clases de la ventana del browser.
2. En este segundo paso tiene las siguientes posibilidades:
   - Pulse [Alt] + [Enter].
   - Ejecute la opción Properties del menú Edit .
   - Ejecute la opción Properties del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el elemento seleccionado.
3. Se muestra el cuadro de diálogo Class Properties que permite definir o modificar sus características.

Creación de plantillas

Cualquier clase que haya creado en un módulo puede utilizarla como plantilla en otro módulo del proyecto para la creación de otra nueva.

Para crear y utilizar una plantilla proceda de la siguiente manera:

1. Cree una clase.
2. Defina en la clase el interfaz y los métodos que desee reutilizar en otros módulos.
3. Arrastre y suelte (drag & drop) la clase sobre el módulo en el que desee reutilizarla.
4. Se muestra el cuadro de diálogo Clone class que le pide un nombre para la clase. Introduzca el nombre deseado y pulse el botón Ok .
5. Automáticamente queda añadida la clase al módulo.
6. A continuación puede modificarla y adaptarla a sus necesidades.

Al arrastrar una clase de un módulo a otro arrastrará con ella todos sus métodos.

Clase Page

La Clase <Page> permite diseñar en su definición, un formato de impresión mediante un editor gráfico. Un informe impreso se construirá enviando uno o varios de estos formatos a una impresora por medio de un objeto <PrnDocument>.

- **Estructura de la clase Page**

Al agregar una clase de tipo Page se añade a la estructura en árbol del módulo un nodo como el que se muestra a continuación:

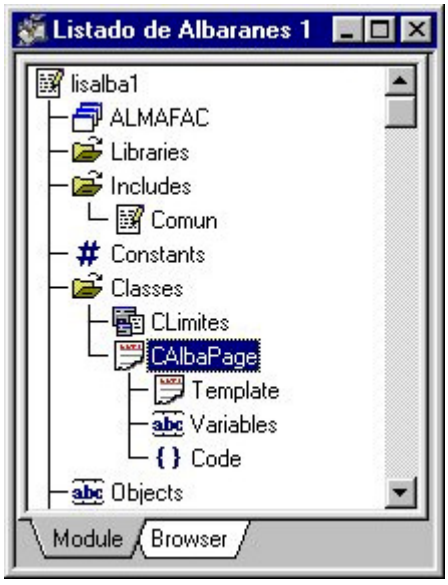


Figura 22.10. Componentes de una clase Page

Como puede verse en la figura una página está definida por las siguientes secciones:

Template	Sección en la que se define el diseño de la plantilla de la página para hacer formularios, impresos, listados, etc.
Code	Sección en la que se definen los métodos asociados a la clase Page.
Variables	Sección opcional en la que se definen las variables que se utilizan en la página de impresión.

Clase Form

Esta clase deriva de la clase <Window>, es abstracta. Esta clase define la funcionalidad de todos los objetos de tipo <Form> de una aplicación. Un Form contiene fundamentalmente un componente gráfico denominado <Screen> que permite dialogar con el usuario final mostrando información de la aplicación y recogiendo la respuesta de éste. Así mismo un <Form> de Cosmos tiene predefinida toda la funcionalidad necesaria para la edición directa de tablas de una Base de Datos.

La clase Form será la clase padre de las diferentes clases de formato de pantalla que defina el programador. Una vez definida una clase de tipo Form, el programador podrá definir objetos de esta clase en su programa si esta no ha sido definida como abstracta.

Esta clase tiene predefinidos una serie de métodos que permiten realizar tareas sencillas, como añadir, borrar o modificar filas de una tabla sin necesidad de programarlas. También permite realizar tareas más complejas, por ejemplo, el mantenimiento de tablas de la base de datos enlazadas mediante una relación “1 a N”.

Por lo general se hará uso de un Form siempre que se desee establecer un dialogo con el usuario.

- **Estructura de un Form**

Al agregar una clase derivada de la clase Form se añade a la estructura en árbol del módulo un nodo como el que se muestra a continuación:

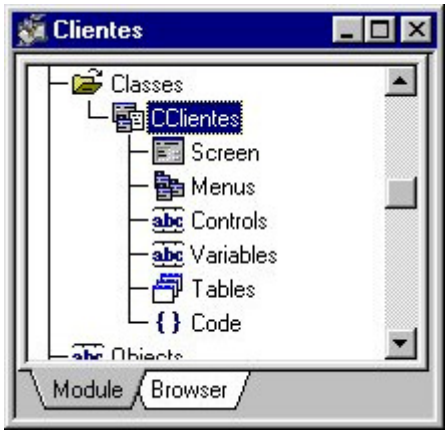


Figura22.10. Estructura de un Form.

Como puede observar un Form esta definido por 6 secciones:

Screen — Sección opcional en la que se define el diseño de la pantalla de presentación del Form

Menús — Los menús proporcionan a los usuarios un modo sencillo de ejecutar comandos agrupados lógicamente. Esta sección es opcional y en ella se definen los menús del Form. Un Form puede tener asociado más de un menú, por programa podrá seleccionar el menú que se desea ejecutar en cada momento.
Para definir un menú en el Form:

1. Cree o inserte un menú en el Form.
2. Edite el menú.

En esta sección también puede:

1. Eliminar un menú asociado a un Form.
2. Editar las propiedades del menú para cambiarlo de nombre.

Variables — Sección opcional en la que se definen las variables que utiliza el Form.

Tablas — Sección opcional en la que se definen las tablas que utiliza el Form. Si la tabla no es de tipo vartable tendrá que estar definida en el repositorio asociado al modulo del Form. Si se añade más de una tabla al Form, estas deben estar enlazadas obligatoriamente mediante un join.
En esta sección puede:

1. Añadir tablas al Form.
2. Insertar tablas en el Form.
3. Editar las propiedades de una tabla para modificarla.
4. Eliminar tablas del Form.
5. Editar columnas de las tablas.
6. Definir, borrar y modificar joins entre tablas para los Form cabeceras-líneas.

Controles — Este nodo de la estructura del Form permite consultar el identificador y el tipo de los controles que tiene definidos su screen. Una screen interactúa con el usuario a través de uno o más controles. Un control es un elemento gráfico de entrada o salida.

Código — Sección opcional en la que se definen los métodos asociados a la clase Form para poder manejar sus menús, variables y tablas.

Clase Menú

Un menú Cosmos es un objeto gráfico que permite al usuario de una aplicación elegir entre una serie de opciones disponibles.

En función de cómo se presenta un menú al usuario, podemos definir los siguientes tipos de menús :

Pulldown Menus — Aparecen en la parte superior de una ventana de WINDOWS bajo su título y permanecen siempre visibles.

Popup Menus — Aparecen como ventanas en cualquier parte de la pantalla, permitiendo seleccionar entre sus opciones para desaparecer a continuación.

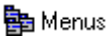
Button Menus — Es una variedad de los “Pulldown Menus” que muestra cada opción del menú como un control de tipo <Button>. Estos menús son realmente controles gráficos definibles en la <Screen> de un Form a los que se asocia un <Menu>.

La Clase abstracta <Menu> define toda la funcionalidad necesaria para manejar todos los aspectos de un menú.

Una clase derivada de la clase abstracta <Menu>, permite definir las opciones que la componen en la creación de los objetos de dicha clase, no obstante, durante la ejecución de un programa se puede modificar añadiendo, modificando o eliminando dichas opciones.

▪ Edicción de Menús

Se pueden editar menús en:

1. La sección menú 

de los Forms que se tiene definidos en el módulo.
2. Las clases Menú  definidas en el módulo.

Por lo tanto para poder editar un menú deberá tener definida en el módulo por lo menos una clase Form o una clase Menú.

Para editar un menú proceda de la siguiente manera:

1. En la sección menú de la estructura de una clase Form seleccione uno de los menús que tenga definidos o cree uno nuevo. Los menús están identificados con el icono .
   En una clase Menú definida en el módulo seleccione el nodo Interface.
2. En este paso tiene las siguientes posibilidades:
   - Pulse [Ctrl] + [E].
   - Ejecute la opción Edit del menú Edit .
   - Ejecute la opción Edit del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el nodo seleccionado.
   - Haga doble clic sobre el nodo.
3. Automáticamente se ejecuta el editor de menús cuyo aspecto se muestra en la siguiente figura.

7. Editor de Objetos

Para crear un objeto, lo primero que se necesita es tener alguna clase accesible en el módulo por que los objetos son instancias de la clase.

Para editar los objetos de una clase proceda de la siguiente manera:

1. Seleccione en la estructura en árbol del módulo el nodo Objects .
2. En este segundo paso tiene las siguientes posibilidades:
   - Pulse [Ctrl] + [E].
   - Ejecute la opción Edit del menú Edit .
   - Ejecute la opción Edit del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el elemento seleccionado.
   - Haga doble clic sobre el nodo Objects.
3. Automáticamente se muestra el editor de objetos que permite añadir, insertar, borrar y modificar los objetos definidos en el módulo en curso.

Este editor permite añadir, insertar, borrar y modificar los objeto del módulo en curso.

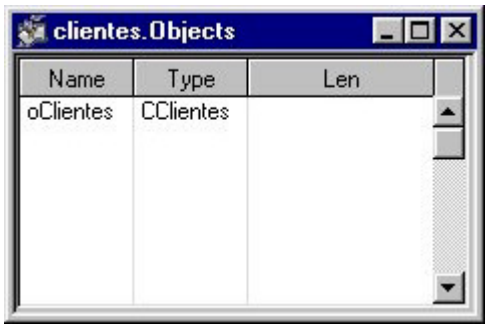


Figura 22.11. Editor de objetos

Este cuadro de diálogo solo consta de una lista multicolumna y multiselección que muestra el nombre; tipo y longitud de los objetos definidos en el módulo en curso. En dicha lista se permite modificar el orden de sus elementos.

Para añadir un objeto proceda de la siguiente manera:

1. Pulse [Ctrl] + [A] o ejecute la opción Add del menú Edit o la opción Add del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el cuadro de diálogo.
2. Se muestra el cuadro de diálogo Object Properties que permite definir el objeto.
3. El objeto añadido se mostrará automáticamente en este editor.

Para insertar un objeto proceda de la siguiente manera:

1. Seleccione en la lista el objeto delante del cual desea insertar uno nuevo:
2. En este segundo paso tiene las siguientes posibilidades:
   - Pulse [Insert].
   - Ejecute la opción Insert del menú Edit .
   - Ejecute la opción Insert del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el objeto.
3. Se muestra el cuadro de diálogo Object Properties que permite definir el objeto.
4. El objeto insertado se mostrará automáticamente en este editor.

Para modificar un objeto proceda de la siguiente manera:

1. Seleccione en la lista el objeto que desee modificar.
2. En este segundo paso tiene las siguientes opciones:
   - Pulse [Alt] + [Enter].
   - Ejecute la opción Properties del menú Edit .
   - Ejecute la opción Properties del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el objeto.
   - Haga doble clic sobre el objeto.
3. Se muestra el cuadro de diálogo Object Properties que permite modificar la definición del objeto.

Para borrar un objeto proceda de la siguiente manera:

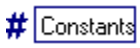
1. Seleccione en la lista el objeto que desea borrar.
2. Pulse [Del] o ejecute la opción Delete del menú Edit o la opción Delete del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el objeto.

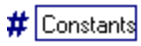
8. Editor de constantes

Frecuentemente, encontramos que el código fuente de nuestros programas contienen valores que se repiten una y otra vez. O es posible que el código dependa de ciertos números que son difíciles de recordar, números que, por si mismos, no tienen un significado obvio.

En estos casos se puede mejorar mucho la legibilidad del código, y facilitar su mantenimiento, mediante el uso de constantes. Una constante es un nombre significativo que toma el lugar de un número o cadena que no cambia, es un valor fijo que el programa no puede alterar. A las constantes se les asigna un valor en el momento en que se declaran. Al igual que los objetos, las constantes también tienen tipo, pero a diferencia de éstos, las constantes asumen el tipo de valor declarado en la parte de declaración de las constantes. Por ejemplo, la constante PI = 3.14159 es Decimal, porque el valor 3.14159 es Decimal. En una operación de asignación el identificador que representa una constante no puede aparecer a la izquierda.

El Editor Visual permite editar las constantes del módulo en curso ejecutando las siguientes instrucciones:

1. Seleccione en la estructura en árbol del módulo el nodo 

.
2. En este segundo paso tiene las siguientes posibilidades:
   - Pulse [Ctrl] + [E].
   - Ejecute la opción Edit del menú Edit .
   - Ejecute la opción Edit del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el elemento seleccionado.
   - Haga doble clic sobre el nodo 

.
3. Automáticamente se muestra el Editor de constantes que permite añadir, insertar, borrar y modificar constantes del módulo en curso.

En un módulo no pueden existir dos constantes con el mismo nombre.

Este editor permite añadir, insertar borrar y modificar constantes del módulo en curso. Solo consta de una lista multicolumna que muestra el nombre y el valor de las constantes definidas en el módulo en curso. En dicha lista se permite modificar el orden de sus elementos.

Figura 22.13. Editor de Constantes

Para añadir una constante proceda de la siguiente manera:

1. Pulse [Ctrl] + [A] o ejecute la opción Add del menú Edit o la opción Add del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el cuadro de diálogo.
2. Se muestra el cuadro de diálogo Constant Properties que permite definir la constante.
3. La constante añadida se mostrará automáticamente en el editor.

Para insertar una constante proceda de la siguiente manera:

1. Seleccione en editor la constante delante de la cual desea insertar una nueva:
2. En este segundo paso tiene las siguientes posibilidades:
   - Pulse [Insert].
   - Ejecute la opción Insert del menú Edit .
   - Ejecute la opción Insert del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre la constante.
3. Se muestra el cuadro de diálogo Constant Properties que permite definir la constante.
4. La constante insertada se mostrará automáticamente en este editor.

Para modificar una constante proceda de la siguiente manera:

1. Seleccione en la lista la constante que desee modificar.
2. En este segundo paso tiene las siguientes posibilidades:
   - Pulse [Alt] + [Enter].
   - Ejecute la opción Properties del menú Edit .
   - Ejecute la opción Properties del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre la constante.
   - Haga doble clic sobre la constante.
3. Se muestra el cuadro de diálogo Constant Properties que permite modificar la definición de la constante.

Para borrar una constante proceda de la siguiente manera:

1. Seleccione en la lista la constante que desea borrar.
2. Pulse [Del] o ejecute la opción Delete del menú Delete o la opción Delete del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre la constante.

9. Editor de Código

Para completar el programa solo resta escribir el código que realmente constituye la aplicación, como por ejemplo los métodos que calculan datos, etc.., es decir el programa en sí.

Además se deberá escribir el contenido de los métodos que se ejecutan cuando se generan los eventos.

Para editar el código fuente de un módulo o de una clase proceda de la siguiente manera:

1. Dos pasos dependiendo del código que se quiera editar:
   - Si desea editar código de un módulo:
     Seleccione el nodo { } Code en la estructura en árbol del módulo.
   - Si desea editar métodos de una clase:
     Si es una clase derivada de clase Form, Page o Menu seleccione el nodo { } Code en la estructura en árbol de la clase, en caso contrario seleccione la clase que desea modificar en la estructura en árbol del módulo.

   A continuación siga las siguientes instrucciones:
2. En este segundo paso tiene las siguientes opciones:
   - Pulse [Ctrl] + [E].
   - Ejecute la opción Edit del menú Edit .
   - Ejecute la opción Edit del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el nodo seleccionado.
   - Haga doble clic sobre el nodo seleccionado.
3. Automáticamente se muestra el Editor de código que permite añadir, borrar y modificar funciones del módulo o clase en curso.

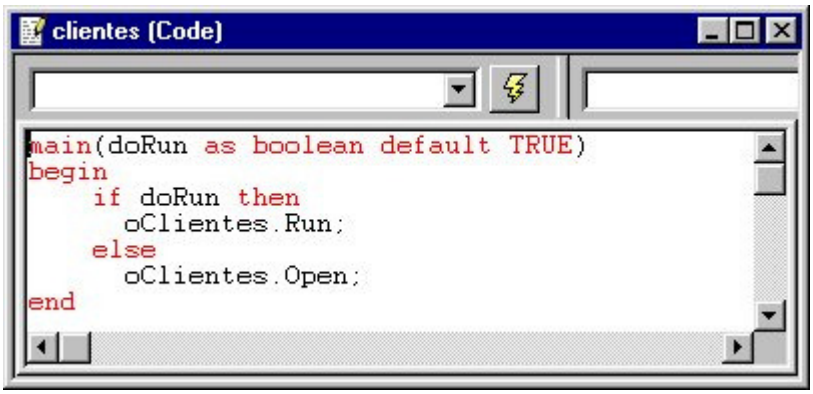


Figura22.14. Editar código.

10. Wizard de Clases

El Wizard de clases es un asistente que ayuda al programador a escribir código y permite realizar consultas sobre las clases definidas en el módulo en curso y sus métodos:

1. Consultar un método definidos por el usuario en una clase o en el módulo (el propio módulo es una clase derivada de la clase Module).
2. Añadir nuevos comandos a un Form, una tabla del Form.
3. Consultar comandos definidos y predefinidos en un Form y una tabla del Form.
4. Añadir código de respuesta a un evento para un Form, un control de una screen de un Form o una tabla.
5. Consultar eventos definidas en Cosmos.
6. Consultar los controles y las propiedades que se pueden aplicar a dichos controles en una screen de un Form o en una página de impresión.
7. Consultar las propiedades de los menús.

Al ejecutar esta opción se muestra el cuadro de diálogo Class Wizard que permite realizar la consulta.

Esta opción también se puede ejecutar pulsando  en el menú de botones o [Ctrl] + [W].

11. Paletas de Repositorio

Los repositorios pueden ser utilizados como paletas dentro del Editor Visual de Cosmos. El editor de repositorios permite la definición del tipo de interfaz gráfico para cada una de las columnas de la base de datos: En el momento de arrastrar una columna desde una paleta del repositorio hasta una screen de un Form, esta columna aparecerá en la screen con el interfaz gráfico que se haya definido.

Un proyecto puede tener asociado más de un repositorio. Para editar un repositorio desde la ventana del proyecto:

1. En la estructura en árbol del proyecto seleccione el nodo Repositories.
2. Ejecute la opción Import . Automáticamente queda añadido el repositorio a la estructura en árbol del proyecto.
3. Seleccione el repositorio. Los repositorios se identifican con el icono .
4. En este paso tiene las siguientes posibilidades:
   - Pulse [Ctrl] + [E].
   - Ejecute la opción Edit del menú Edit .
   - Ejecute la opción Edit del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el repositorio seleccionado.
   - Haga doble clic con el botón izquierdo de ratón.
5. Automáticamente aparece una paleta con la estructura en árbol del repositorio.

Cuando un módulo tiene asociado un repositorio, también puede editar el repositorio que tiene asociado desde la ventana del módulo, ejecutando los siguientes pasos:

1. Seleccione en la estructura en árbol del módulo el repositorio, esta identificado por su nombre y el icono .
2. Ejecute los pasos 4) y 5) indicados anteriormente.

Figura 22.15. Estructura en árbol del Repositorio

Desde la paleta del repositorio se pueden arrastrar columnas, tablas y plantillas a:

1. La sección tablas de un Form.
2. La sección screen de un Form.
3. Una página de impresión.

También se puede editar un repositorio desde un módulo (librería, programa o include) que tiene definido un repositorio, para ello repita los pasos 3) 4) 5) indicados anteriormente en la ventana que muestra la estructura en árbol de dicho módulo.

Las tablas se identifican con el icono  . Las columnas se identifican con el icono  . Las columnas que forman parte de la clave primaria de la tabla se identifican con el icono  . Las columnas que forman parte de una clave referencial se identifican con el icono  . Las columnas que forman parte de una clave referencial y de la clave primaria se identifican con el icono  . Los enlaces de la tabla se identifican con el icono  . Las estructuras se identifican con el icono  . Las plantillas se identifican con el icono  .

Desde la estructura en árbol del repositorio se puede también:

- Consultar los atributos de las columnas.
- Consultar los enlaces definidos entre tablas.

sin necesidad de ejecutar el editor de repositorios.

En la paleta del repositorio es posible elegir entre mostrar las etiquetas o los nombres de las las tablas y columnas mediante las opciones Show table labels y Show column labels del menú View . Esta selección se guardará para sesiones posteriores. Si se elige mostrar las etiquetas (opción por defecto), al seleccionar una tabla o una columna se mostrará su nombre en la barra de estado. Si se elige mostrar los nombres, se mostrará la etiqueta en la barra de estado.

12. Paleta de Proyecto

El Editor Visual de Cosmos permite utilizar paletas de proyecto para facilitar la tarea de incluir en el proyecto en curso componentes de otros proyectos. Esta opción estará visible cuando tenga el foco de entrada la ventana del proyecto.

Al ejecutar la opción Project Palette del menú View se muestra un cuadro de diálogo similar al de la opción Open que permite seleccionar uno de los proyectos existentes. Una vez seleccionado se muestra una ventana con su contenido. Desde esta paleta, empleando la técnica de “arrastrar y soltar” (drag & drop) podrá incorporar cualquiera de sus componentes al proyecto en curso.

Para incorporar un elemento de la paleta al proyecto proceda de la siguiente manera:

1. Seleccione un componente de la paleta.
2. Arrástrelo sobre el elemento o nodo correspondiente de la ventana del proyecto en curso.
3. Automáticamente queda añadido dicho elemento al proyecto.

Al arrastrar un elemento de la paleta se añade y copia dicho elemento al proyecto en curso tal y como se muestra en la siguiente figura. Puede observar en las propiedades que el Path del elemento arrastrado es el del proyecto en curso.

Si arrastra un elemento manteniendo la tecla [Mayúsculas] pulsada, incorpora dicho elemento al proyecto conservando el directorio en el que estuviera ubicado. Como se muestra en la siguiente figura, esto puede comprobarlo editando las propiedades del elemento arrastrado. El campo Path de las propiedades del módulo MIFORM sigue siendo el mismo que tenia en el proyecto Personal.

13. Compilación

Para compilar un modulo o un fichero de mensajes del proyecto, selecciónelo y ejecute la opción Compile del menu Tools en el menú principal del Editor Visual, o pulse el botón  del menú de iconos.

Si se detectan errores durante la compilación se mostrará la ventana output con los errores detectados. Al hacer doble clic sobre cualquiera de ellos, se da el foco de entrada a la ventana de código con el cursor situado en la línea de código que lo produjo.

El Editor Visual tiene otras 4 opciones en el menú Tools que permiten la compilación masiva de un conjunto de módulos del proyecto.

Rebuild All	Compila de nuevo todos los modulo del proyecto. Esta opción también la puede ejecutar pulsando el botón  del menú de botones del Editor Visual.
Build All	Compila sólo los módulos que estén desactualizados del proyecto. Esta opción también la puede ejecutar pulsando el botón  del menú de botones del Editor Visual.

Las 2 opciones que se explican a continuación solo están visibles cuando tiene el foco de entrada la ventana del proyecto. y afectan al nodo del proyecto que se tenga seleccionado.

Rebuild “branch”	Compila de nuevo los módulos definidos dentro del nodo del proyecto con nombre “branch”
Build “branch”	Compila los módulos desactualizados definidos dentro del nodo del proyecto con nombre “branch”

Un modulo se entiende que está desactualizado si :

1. No existe el módulo compilado (objeto) de éste modulo
2. La fecha del fuente es posterior a la del objeto.
3. Alguno de sus includes o librerías tiene fecha posterior al módulo.

Para realizar la comprobación de dependencias asegúrese de que la fecha y la hora se han establecido correctamente en su ordenador. Si la fecha y la hora no son correctas, la comprobación de dependencias no funcionará adecuadamente.

Un módulo externo (extensión .omd) no se puede compilar por que no se tiene el código fuente. Pero se chequean sus dependencias y se compilan los includes y librerías no externos incluidos en dicho módulo y que se encuentran en el proyecto.

Comando Cosmake

Este comando es el encargado de compilar los ficheros de código fuente Cosmos (identificados por la extensión .smd). Los ficheros generados por este comando tendrán el mismo nombre que sus respectivos de código fuente, pero con la extensión .omd .

Su sintaxis es la siguiente.

cosmake [-prj <projects.prj> [-pf <fichero> [-smd <f1.smd> - smd <f2.smd> ... [-all] [-reb] [-com] [-grp <label>]

Parámetro	Significado
-prj <projects.prj>	Nombre del proyecto con extensión .prj al que pertenece el módulo que desea compilar
-pf <fichero>	Fichero del que se leerán los parámetros
-smd <f1.smd>..	Nombre del programa o programas del proyecto <projects.prj> al que se desea aplicar el comando cosmake. Este nombre se indicará sin path y sin extensión
-grp <label>	Ejecuta el comando cosmake para todos los módulos del grupo <label> del proyecto
-all	Ejecuta el comando cosmake para todos los módulos del proyecto. Esta opción no tiene en cuenta las opciones -grp y -smd
-reb	Compila todos los módulos indicados con las tres opciones anteriores y todas sus dependencias (los includes y librerías no externos que forman parte de los módulos indicados). Un módulo externo incluido en el proyecto es un módulo compilado de otra aplicación Cosmos. Un módulo externo (extensión .omd) no se puede compilar por que no se tiene el código fuente. Pero se chequean sus dependencias y se compilan los includes y librerías no externos de dicho módulo y que se encuentran en el proyecto
-com	Compila los módulos indicados con las tres opciones anteriores y las dependencias que hagan falta

Comando cosmsg

Este comando permite compilar ficheros de mensajes con extensión .shl . El resultado de la ejecución de este comando es un fichero cuya extensión es .ohl .

Su sintaxis es la siguiente:

cosmsg [-sms <source_path>] [-oms <object_path>] [-cod <code>] [-hex]

Parámetro	Significado
-sms <source_path>	Path del fichero de mensajes que contiene los textos de los mensajes utilizados por la aplicación
-oms <object _path>	Path del fichero que contendrá el resultado de la compilación
-cod <code>	Permite consultar el texto definido para el código de mensaje code
-hex	Indica que los códigos se expresan en formato hexadecimal

14. Ejecución

Después de compilar todos los módulos de la aplicación el siguiente paso es ejecutar el programa.

Para ejecutar un programa concreto debe tener el foco de entrada en alguna de las ventanas de edicion del módulo., o en la ventana del proyecto teniendo seleccionado el programa que se quiere ejecutar. A continuación ejecute la opción Execute del menú Tools en el menú principal del Editor Visual, o pulse el botón  del menú de iconos.

Si no existe ningún módulo en curso, y se ha definido un modulo principal (“Main Module”) en el proyecto, será este módulo el que se ejecute,

La opción Debug del menú Tools o el botón  de la barra de herramientas permiten ejecutar el programa en modo depuración.

El botón  de la barra de herramientas permite ejecutar el módulo principal (definido como Main Module en sus propiedades) del proyecto.

La opción Arguments del menú Tools muestra un cuadro de dialogo que le permite introducir los argumentos que se desean pasar al programa en el momento de la ejecución.

Comando Cosrun

Este comando se utiliza para la ejecución de programas Cosmos compilados.

cosrun [-prj <projects.prj>] [-pf <paramfile>] [-omd <omdfile>] [-dbg] [-arg <argument>]

Parámetro	Significado
projects.prj	Nombre del proyecto con extensión .prj al que pertenece el módulo que desea compilar
paramfile	Fichero del que se leerán el resto de los parámetros
omdfile	Nombre del fichero binario resultado de la compilación del módulo del proyecto. Este nombre se indicará sin path y sin extensión. Si no se indica este parámetro, se asume por defecto el módulo principal (“Main Module”) del proyecto por que es el módulo que inicia la aplicación
dbg	Permite ejecutar el programa en modo depuración
argument	Argumento pasado a la función Main del módulo que arranca, va entre comillas

Capítulo 10
Editor de Screen

Contenidos

1. Introducción

Un Form permite manipular los datos de las tablas de la base de datos a través de una serie de variables con una cierta correspondencia con columnas de la base de datos y de operaciones que dependen de esas variables. Además, pueden utilizarse variables auxiliares que no tengan correspondencia con columnas de las tablas de la base de datos.

Una screen describe el formato de presentación de la pantalla de un Form.

El Editor de Screen es una herramienta que permite:

- Diseñar las pantallas de entrada de datos y mantenimiento de un form.
- Diseñar menús de botones y paletas de herramientas de un form.
- Simular el comportamiento en ejecución del screen sin necesidad de compilar el módulo al que pertenece.

Utilizando este editor podrá:

- Definir las propiedades principales de la screen.
- Añadir, borrar y modificar los controles de la screen.
- Editar y Modificar las propiedades de un control.
- Mover, alinear, centrar, ajustar, espaciar y redimensionar un grupo de controles.
- Distribuir los controles en la screen.
- Bloquear el movimiento de los controles de la screen activa. Esta opción es útil aplicarla cuando se quiere seleccionar un control o un grupo de controles y no se desea que al seleccionarlo el control cambie de posición o tamaño.
- Asociar un menú a la screen.
- Definir el menú asociado como menú de botones o de tipo pulldown.
- Eliminar el menú que tiene asignado la screen.
- Añadir o eliminar una barra de estado a la screen.
- Asociar iconos a cierto tipo de controles.
- Utilizar diferentes colores.
- Utilizar diferentes tipos de fuentes.
- Definir el orden en el cual el tabulador cambia el foco de entrada de un control al siguiente dentro de la screen o de su control padre.
- Definir un control como padre de un grupo de controles.
- Realizar una presentación preliminar para ver el aspecto que tendrá una screen cuando se imprima.
- Asociar una variable o una columna de una tabla del Form a un control de la screen.
- Asociar una tabla del Form a cierto tipo de controles.
- Añadir tablas y o columnas del repositorio al Form.

Para realizar las acciones mencionadas anteriormente el Editor de Screen de Cosmos dispone de los siguientes elementos:

- Un conjunto de paletas.
- Una barra de herramientas.
- Una barra de estado para mostrar información al usuario.

Las modificaciones realizadas con este editor se guardan en el código fuente del módulo (programa, librería o include) al que pertenece el form.

2. Paletas

El Editor de Screen de Cosmos incluye un conjunto de paletas que permiten el diseño de un alto porcentaje de la screen arrastrando objetos hacia un lugar determinado:

Paleta de controles	Permite añadir un control a la screen.
Paleta de colores	Permite cambiar el color de fondo y el color de primer plano de los controles de la screen.
Paleta de iconos	Permite asociar iconos a cierto tipo de controles.
Paleta de ajuste	Permite alinear, centrar, ajustar, espaciar y redimensionar un grupo de controles.
Paleta de fuentes	Esta paleta permite modificar el font definido por defecto para la screen o el font asociado a uno de sus controles.

Paleta de Controles

Una screen interactúa con el usuario a través de uno o más controles. Un control es un elemento gráfico de entrada o salida, que puede ser o no contededor de otros controles.

Cosmos ha sido diseñado con el fin de proporcionar al programador acceso a todos los controles de Windows de forma sencilla, sin requerir el conocimiento de los cientos de funciones de interfaz gráfico de las librerías de Windows ni de su complejo funcionamiento.

En esta paleta se encuentran todos los tipos de controles necesarios para diseñar una screen.

Después de añadir el control, edite sus propiedades para modificar su estilo y sus características.

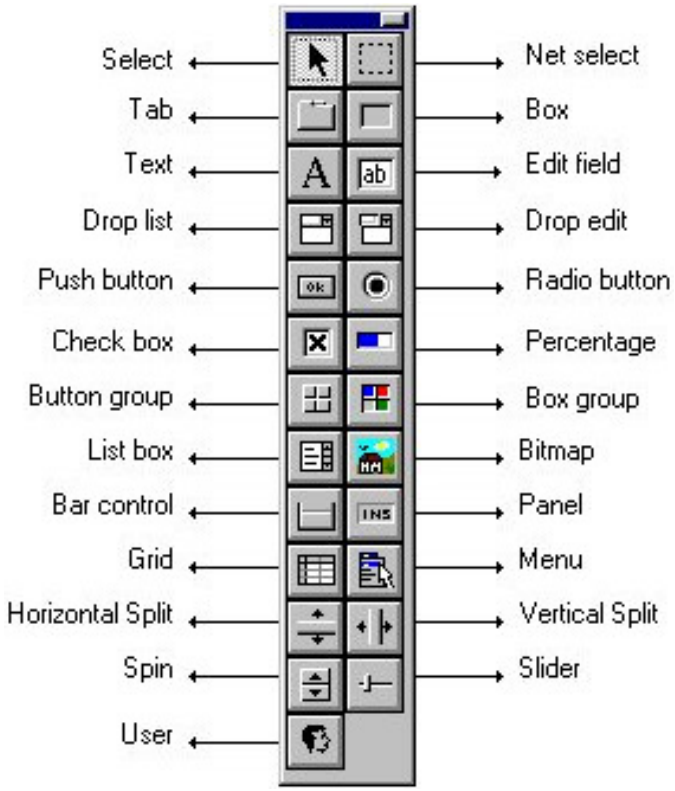


Figura 10.1. Paleta de controles.

Elemento	Efecto
Select	Herramienta que permite seleccionar, mover, copiar, borrar, dimensionar y editar las propiedades de un control
Net select	Herramienta que permite seleccionar un grupo de controles para copiar, borrar, mover o editar en conjunto
Tab	Herramienta para crear controles de tipo ficha
Box	Herramienta para crear controles de tipo caja
Text	Herramienta para crear controles de texto
Edit field	Herramienta para crear controles de edición
Drop list	Herramienta para crear listas desplegables
Drop edit	Herramienta para crear listas desplegables de edición
Push button	Herramienta para crear botones de comando
Radio button	Herramienta para crear botones tipo radio
Check box	Herramienta para crear casillas de verificación
Percentage box	Herramienta para crear barras de porcentaje o progreso
Button group	Herramienta para crear grupos de botones
Box group	Herramienta para crear grupos de cajas
List box	Herramienta para crear controles de tipo lista
Bitmap	Herramienta para crear controles de tipo bitmap
Bar	Herramienta para crear controles de tipo barra
Panel	Herramienta para crear controles de tipo panel
Grid	Herramienta para crear controles de tipo Grid
Menu	Herramienta para crear controles de tipo Menu
Horizontal Split	Herramienta para crear barras divisorias horizontales
Vertical Split	Herramienta para crear barras divisorias verticales
Slider	Herramienta para crear controles deslizantes o desplazadores
Spin	Herramienta para crear controles de tipo incremento
User	Herramienta para crear controles de usuario

Todas las opciones de la paleta de controles se encuentran también en el menú Controls de la aplicación.

Los valores por defecto de los controles de esta paleta pueden ser personalizados a gusto del programador a través del menú que aparece al pulsar en la paleta con el botón derecho del ratón, estos valores se pueden mantener durante la sesión o bien guardar para posteriores sesiones.

Para mostrar u ocultar la paleta de controles tiene dos opciones:

1. Ejecute la opción Objects Palette del menú View .
2. Pulse el botón  del menú de botones.

Paleta de Colores

Esta paleta permite cambiar el color de fondo (background) y el color de primer plano (foreground) de los controles de una screen o una página de impresión.

Figura 10.2. Paleta de colores.

El botón  de la paleta de colores permite seleccionar el color definido por defecto para el control sobre el que se arrastre.

El botón  de la paleta de colores permite seleccionar el color transparente.

Para cambiar el color de fondo (background):

1. Pulse el botón  .
2. Seleccione con el botón izquierdo de ratón el color que desee en la paleta y arrástrelo hasta el control de la screen o página de impresión que desee cambiar de color.

Para cambiar el color de primer plano (foreground):

1. Pulse el botón  .
2. Seleccione con el botón izquierdo de ratón el color que desee en la paleta y arrástrelo hasta el control de la screen o página de impresión que desee cambiar de color.

También se puede arrastrar el mismo color de la paleta sobre varios controles a la vez siguiendo las siguientes instrucciones :

1. Seleccione con el botón izquierdo del ratón el color que desee en la paleta y arrástrelo hasta el control (de la screen o página de impresión) cuyo color desee cambiar.
2. Sin soltar el botón izquierdo del ratón pulse la barra espaciadora sobre todos los controles a los que desee cambiar el color.

Para mostrar u ocultar la paleta de colores tiene 2 opciones:

1. Ejecute la opción Colors Palette del menú View .
2. Pulse el botón  del menú de botones.

Paleta de Fuentes

La paleta de fuentes tiene el aspecto que se muestra en la figura.

Figura 10.3. Paleta de fonts.

Esta paleta permite modificar:

- El font asociado a un control de una página de impresión o una screen.
- El font definido por defecto para la página de impresión o la screen.

El primer font que aparece en la paleta es el font seleccionado por defecto para la página de impresión o la screen. Por defecto este font es de tipo MS Sans Serif y tamaño 8.

Para modificar el tipo de fuente proceda de la siguiente manera:

1. Seleccione la fuente deseada en la paleta.
2. Si desea modificar la fuente por defecto, arrastre y suelte la fuente seleccionada sobre la página de impresión o sobre la screen.

Si desea modificar la fuente asociada a un control arrastre y suelte la fuente seleccionada sobre dicho control.

Si pulsa el botón Añadir de la paleta se muestra un cuadro de diálogo idéntico al de la opción Font del menú Layout que permite seleccionar una fuente para agregarla a la paleta.

Al modificar el tipo de fuente que utiliza por defecto la screen o la página de impresión, solo se modificará el de los controles que utilizan la fuente definida por defecto.

También puede modificar el tipo de fuente ejecutando la opción Font del menú Layout o editando las propiedades del control.

Para mostrar u ocultar la paleta de fuentes tiene 2 opciones:

1. Ejecute la opción Fonts Palette del menú View .
2. Pulse el botón  del menú de botones.

Paleta de Ajuste

La paleta de ajuste tiene las opciones necesarias para distribuir los controles sobre la screen o página de impresión activa y redimensionarlos.

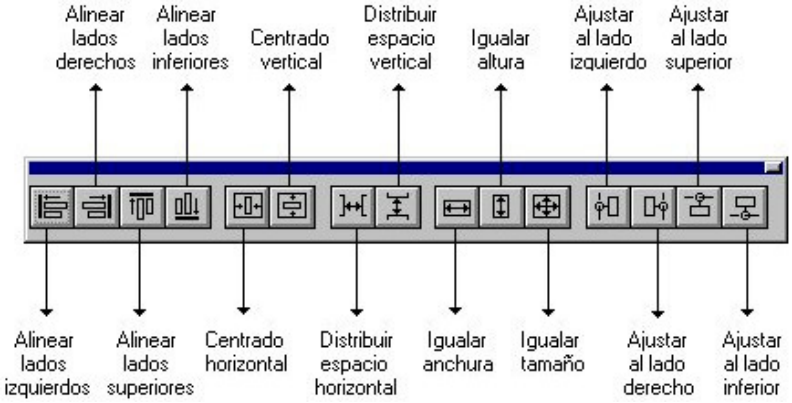


Figura 10.4. Paleta de ajuste.

Las herramientas de la paleta de ajuste estarán activas o inactivas dependiendo del numero de controles seleccionados, ya que ciertas operaciones solo se pueden realizar sobre un único control.

En algunas de las operaciones en las que intervienen más de un control hay uno de ellos que se considera dominante (el primero que se seleccionó, se marca de distinta forma) y que se toma como referencia.

Elemento	Efecto
Alinear lados izquierdos	Alinea los lados izquierdos de los controles seleccionados
Alinear lados derechos	Alinea los lados derechos de los controles seleccionados
Alinear lados superiores	Alinea los lados superiores de los controles seleccionados
Alinear lados inferiores	Alinea los lados inferiores de los controles seleccionados
Centrado horizontal	Centra horizontalmente los controles seleccionados
Centrado Vertical	Centra verticalmente los controles seleccionados
Distribuir espacio horizontal	Distribuye horizontalmente los controles seleccionados
Distribuir espacio vertical	Distribuye verticalmente los controles seleccionados
Igualar anchura	Iguala la anchura de los controles seleccionados
Igualar altura	Iguala la altura de los controles seleccionados
Igualar tamaño	Iguala el tamaño de los controles seleccionados
Ajustar al lado izquierdo	Fija la posición al lado izquierdo del padre
Ajustar al lado derecho	Fija la posición al lado derecho del padre
Ajustar al lado superior	Fija la posición al lado superior del padre
Ajustar al lado inferior	Fija la posición al lado inferior del padre

Todas las opciones de la paleta de ajuste también se encuentran en el menú Layout del menú principal de la aplicación.

Para mostrar u ocultar la paleta de ajuste tiene 2 opciones:

- Ejecute la opción Adjust Palette del menú View .
- Pulse el botón  del menú de botones.

Paleta de Iconos

La paleta de iconos tiene el aspecto que se muestra a continuación:

Figura 10.5. Paleta de iconos.

Utilizando la técnica de arrastrar y soltar (drag & drop), esta paleta permite:

1. Asociar iconos a los controles de una screen de tipo: botón de comando (Push Button), caja (Box), Grupo de botones (Button Group), Grupo de cajas (Box Group), listas (List box).
2. Asociar iconos a los controles de impresión de tipo caja (Box) y Grupo de cajas (Box Group).
3. Asociar iconos a los ítems de un menú.

En la lista desplegable de la paleta se muestran los ficheros de iconos existentes en la sección [Icons] del fichero de configuración COSMOS.INI . Una vez seleccionado el fichero se muestra su contenido, si la totalidad de los iconos no cabe en la paleta podrán verse los restantes pulsando los botones  y  .

Para mostrar u ocultar la paleta de iconos tiene 2 opciones:

- Ejecute la opción Icons Palette del menú View .
- Pulse el botón  del menú de botones.

3. Tipos de Controles

Tab 

Un control tipo Tab (Ficha) imita las etiquetas de un conjunto de carpetas de archivo. Cada vez que se elige una etiqueta (página del control), se ve su carpeta asociada (página de la Ficha). Usando este control puede definir múltiples páginas para un mismo área de la screen. Cada página agrupará un conjunto de controles que se mostrarán cuando el usuario seleccione la correspondiente página.

Este tipo de control es útil para agrupar un cierto número de controles (por ejemplo todos los controles asociados a un tabla de una base de datos). También es útil cuando el número de controles de la screen es tan grande que no caben en la caja que define la screen.

Después de crear el control, edite sus propiedadespara modificar su estilo y sus características.

Box 

La utilidad principal de un control de tipo caja (Box) es agrupar un conjunto de controles que guarden alguna relación entre si, para facilitar al usuario la programación. Por ejemplo, todos los controles asociados a un tabla de una base de datos. Usando este control puede definir múltiples páginas para un mismo área de la screen. Cada página agrupará un conjunto de controles que se mostrarán cuando el usuario habilite dinámicamente por software la correspondiente página.

Después de crear el control, edite sus propiedades para modificar su estilo y sus características.

Text 

Este tipo de control sirve para mostrar un texto estático no accesible para el usuario, como los títulos. Suele tratarse de texto que áctua como etiqueta de otro control.

Este tipo de control puede ser de dos tipos:

- Texto simple Texto estático en una sola línea. El texto puede ser centrado o alineado a la derecha (por defecto lo esta a la izquierda), etc.
- Texto multilínea Como su nombre indica puede tener varias líneas de texto. El texto puede estar centrado o alineado a la derecha (por defecto lo esta a la izquierda),etc.

Edit field 

Un control de edición (Edit) es un rectángulo en el cual se escribe información. Al seleccionar un control de edición vacío, aparecerá un punto de inserción (barra vertical intermitente) en el extremo izquierdo del mismo. El texto que se escriba será introducido a partir de ese lugar.

Este tipo de control puede ser de dos tipos:

- Edición simple: Permite editar texto en una sola línea. El texto puede ser centrado o alineado a la derecha (por defecto lo esta a la izquierda), etc.
- Edición multilínea: Como su nombre indica, se puede editar texto en varias líneas. Puede usar la tecla [Enter] para que actúe como tecla de retorno de línea, y el texto puede ser centrado o alineado a la derecha (por defecto lo esta a la izquierda),etc.

Drop List 

Las listas desplegables se presentan inicialmente como un rectángulo con el elemento predeterminado resaltado. Si se selecciona la flecha que aparece en la casilla situada la derecha, se desplegará una lista con todos los elementos disponibles para la elección.

En una Lista desplegable el usuario solo puede elegir un ítem de la lista. Si hay varios ítems que empiezan por la misma letra, al pulsar repetidamente esa letra, la selección ira rotando entre dichos ítems.

Push Button 

Los botones de comando sirven para iniciar una acción de inmediato, son los encargados de lanzar los procesos, cerrar las ventanas secundarias, dar conformidad a los datos, cancelar un comando, etc.; por este motivo están presentes en cualquier aplicación por sencilla que ésta sea. Los botones que no estén disponibles aparecerán atenuados. El botón seleccionado se mostrará con un borde interno discontinuo y un botón predeterminado con un borde externo negro.

Radio button 

Los botones de opción también llamados botones tipo radio (Radio Button) se usan para presentar opciones mutuamente exclusivas. Cada vez que se selecciona un botón se desactiva el botón seleccionado previamente. Es imposible seleccionar más de un botón al mismo tiempo. El botón correspondiente a la opción seleccionada contendrá un punto negro. Las opciones no disponibles se mostrarán atenuadas.

Check box 

Una casilla de verificación (Check Box) se incorpora normalmente en una aplicación para permitir que el usuario active o desactive una opción. Cuando una opción esté activada, su casilla de verificación aparece marcada generalmente con una equis. Cuando el control no esté disponible se presenta su texto atenuado.

Percentage box 

Este control es una barra de porcentaje o progreso que se puede utilizar para mostrar la cantidad de trabajo realizado por una tarea. También puede servir para hacer un diagrama de barras.

Button group 

Este tipo de control se puede utilizar por ejemplo, para crear un menú de botones o una paleta de herramientas. Puede tener un comportamiento similar a :

- Un grupo de Botones Radio (Radio Button).
- Un grupo de casillas de verificación (Check Box).

Box group 

Este tipo de control se puede utilizar por ejemplo, para crear un menú de botones o una paleta de herramientas. Puede tener un comportamiento similar a :

- Un grupo de Botones Radio (Radio Button).
- Un grupo de casillas de verificación (Check Box).

List box 

Como su propio nombre indica, se trata de una caja que contiene una lista de elementos que pueden ser seleccionados. Si hay más elementos de los que caben en el cuadro, este podrá tener barras de desplazamiento que permiten moverse por la lista.

Las teclas [Flecha arriba] y [Flecha abajo] nos permitirán movernos a lo largo de la lista. Por su parte, las teclas [Inicio] o [Home] y [Fin] o [End] nos permitirán desplazarnos respectivamente al primero y al último elemento de la lista.

Las listas pueden ser de diferentes tipos:

Lista de selección simple

Son las más comunes. En este tipo de control solo se puede tener seleccionado un ítem a la vez, cuando pulsamos sobre otro, se resaltara este último y se desactivará el anterior. Las teclas de dirección mueven tanto el cursor como el elemento seleccionado, y pueden desplazar el contenido del cuadro de la lista. Las teclas [RE PAG] y [AV PAG] también desplazan el contenido del cuadro de la lista moviendo el cursor pero no la selección. La pulsación de cualquier tecla de carácter mueve el cursor y la selección al primer elemento de la lista (o siguiente) que empiece con esa letra. También se puede seleccionar un elemento con una pulsación única o doble del botón del ratón sobre ese elemento.

Lista de selección múltiple

Se puede seleccionar más de un ítem a la vez pulsando el botón izquierdo del ratón y la teclas [Ctrl] o [Mayúsculas]. Las teclas de dirección descartan todos los elementos previamente seleccionados y mueven el cursor y la selección, de la misma forma que en una lista de selección simple. Sin embargo la tecla de [Ctrl] con las teclas de dirección pueden mover el cursor sin mover la selección. La tecla [Mayúsculas] junto con las teclas de dirección amplían la selección. Una pulsación única o doble del botón izquierdo del ratón sobre la lista, descarta todos los elementos seleccionados y selecciona el elemento sobre el que se ha pulsado. Sin embargo una pulsación del ratón, con la tecla [Ctrl] pulsada, conmuta el estado del elemento sobre el que se ha pulsado, sin cambiar el estado de los otros elementos.

Lista sin selección

Solo se mostrarán sus elementos, no se podrá hacer ningún tipo de selección sobre ellos.

Listas en árbol

Lista con estructura en forma de árbol.

Para expandir un nodo:

1. Seleccione en la lista uno de los nodos marcados con el icono .
2. Haga doble clic sobre el nodo, se mostrará marcado con el icono y podrá ver su contenido.

Para contraer un nodo de modo que no aparezca su contenido:

1. Seleccione en la lista uno de los nodos marcados con el icono .
2. Haga doble clic sobre el nodo, se mostrará marcado con el icono y no se verá su contenido.

Estas lista pueden ser de selección simple o múltiple.

Listas Sql

Se utilizan para mostrar el resultado de una instrucción SELECT del Sql.

Bitmap 

Esta control permite mostrar una imagen (fichero con extensión .bmp) en la screen. Un bitmap es una representación digital de una imagen. Cada pixel en la imagen corresponde a uno o más bits en el bitmap. Los bitmap monocromos soló requieren un bit por pixel ; los bitmap de color requieren bits adicionales para indicar el color de cada pixel.

Este control puede ser contenedor (padre) de otros controles. Esta característica le permite utilizar un bitmap de fondo en su screen. Usando este control puede definir múltiples páginas para un mismo área de la screen. Cada página agrupará un conjunto de controles que se mostrarán cuando el usuario habilite dinámicamente por software la correspondiente página.

Bar control 

La utilidad más frecuente de este control será para crear una barra de estado. Una barra de estado es un elemento gráfico de sólo salida que suelen estar en la parte inferior de la ventana principal de un programa. El uso más común de una barra de estado es proporcionar texto de ayuda a medida que el usuario selecciona un elemento de un menú. cuando no se está examinando un menú, los programas suelen mostrar información suplementaria en la barra de estado. Una barra de estado puede ser contenedora de otros controles, generalmente contendrá paneles en los que se muestra la información al usuario.

Panel 

En este tipo de controles se mostrará información de utilidad al usuario:

-Comentarios.
-Mensajes.
-Opción de menú, comando u operación que se esta ejecutando.
-Indicar si se esta en modo inserción o en modo reemplazo en un editor de texto.

El programador podrá darle cualquier otra utilidad que desee.

Grid 

El aspecto de este control es similar al de una tabla, en la cual se indicará las etiquetas con las que se desea identificar a sus columnas y el número de columnas del control. Para poder mostrar todas las filas y/o columnas del control, este podrá disponer de barra de desplazamiento horizontal y/o vertical. Este control se podrá definir como padre de otros controles, en este caso el control tendrá tantas columnas como controles hijos. Si el control no se define como padre de otros se puede definir en sus características el número de columnas que va a tener.

La utilidad principal de este tipo de control es:

- Mostrar hojas de cálculo.
- Mostrar las columnas de la tabla de líneas en una pantalla de mantenimiento de una tabla que esta enlazada con otra en un programa de cabeceras - líneas.

Vertical / Horizontal Split  / 

Una barra divisoria (horizontal o vertical) divide su control padre en dos secciones. Esta barra puede desplazarse para dejar más espacio de presentación de una de las secciones.

Para mover la barra divisoria:

1. Desplace el cursor del ratón hasta la barra divisoria. El cursor se convertirá en una barra doble con dos flechas.
2. Pulse y mantenga presionado el botón izquierdo del ratón y mueva el cursor para colocar la barra divisoria. La barra divisoria se desplaza con el cursor.
3. Cuando la barra se encuentre en la posición deseada suelte el botón del ratón.

Al desplazar una barra divisoria se redimensiona el control anterior y el posterior a la barra según el tab order , achicando uno en favor del otro.

Slider 

Un desplazador (Slider) es similar a los botones deslizantes que hay en varios aparatos electrónicos, como en los equipos de música. Consiste en un puntero que se mueve sobre una pista.

Spin 

Este control se puede usar como una barra de desplazamiento independiente o junto a un control de edición para aumentar o disminuir su valor. Cuando va asociado a un control de edición recibe el nombre de control de incremento.

En un Form de mantenimiento de una o más tablas de una base de datos se puede utilizar este control para recorrer la lista en curso del Form.

Menu 

En Windows el elemento de control más común es el menú. Casi todas las ventanas principales llevan asociadas algún tipo de menú. Debido a que los menús son tan comunes e importantes en las aplicaciones de Windows, Cosmos incorpora este tipo de controles.

Dado que un menú va asociado a una ventana principal o secundaria, la lógica creación consiste en la implementación de uno de estos menús, y una vez terminado, su asociación posterior a la ventana. Una vez que el menú se crea está disponible para cualquier ventana.

Utilizando este control podrá situar un menú el cualquier punto de la Screen. Este control no tiene sentido si no se asocia a un menú del Form puesto que el numero de opciones y las etiquetas de estas no se almacenan en el control si no que se toman directamente del menú asociado.

User 

Un control de usuario le permite diseñar un control a su medida.

El método SetUserProc de la clase SimpleControl permite indicar la función que manejará el control.

4. Personalización del Editor de Screen

La paleta de controles tiene un menú de contexto que se activa al pulsar el botón derecho del ratón.

Este menú tiene las siguientes opciones:

Edit	Edita las propiedades del tipo de control seleccionado para personalizar sus valores por defecto.
Help	Muestra la ayuda en línea para el control seleccionado.
Restore	Restaura los valores por defecto de las propiedades del control seleccionado.
Restore All	Restaura los valores por defecto de todos los controles.
Save	Guarda los valores de las propiedades personalizadas por el usuario para sesiones posteriores del Editor Visual.

Las opciones de este menú también se pueden ejecutar desde el submenú Preferences del menú Controls del menú principal de la aplicación.

Si se desea personalizar un control gráfico de modo que sea igual a uno definido en la screen, seleccione este control y pulsando el botón derecho ejecute la opción “Set As Default” del menú de contexto que aparece.

5. Barra de Estado y Barra de Herramientas

Barra de Estado

La barra de estado muestra información y mensajes en la parte inferior de la ventana para ayudarle a utilizar el Editor Visual.

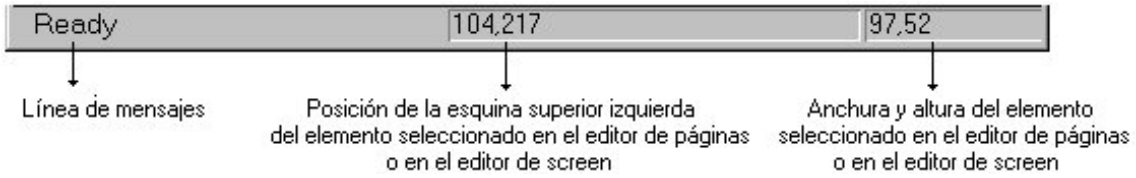


Figura23.6. Barra de estado.

Para mostrar u ocultar la barra de estado ejecute la opción Status Bar del menú View.

Barra de Heramientas

Las barras de herramientas contienen botones que le permiten acceder rápidamente con el ratón a muchos comandos y funciones del Editor. A continuación se explican las opciones relacionadas con el Editor de Screen.

Cortar	Elimina el control o controles seleccionados y los almacena en el portapapeles de Windows para permitir su inserción en otro punto de la página de impresión o de la screen activa o en otra distinta.
Copiar	Copia el control o controles seleccionados al portapapeles de Windows para permitir su inserción en otro lugar de la misma página de impresión o de la screen o en otra distinta.
Pegar	Inserta en la screen o página de impresión activa el control o controles previamente copiados al portapapeles de Windows.
Deshacer	Deshace los últimos cambios.
Rehacer	Recupera los cambios que han sido anulados mediante la opción Deshacer .
Test	Simula el comportamiento en ejecución de la screen desde el Editor Visual sin necesidad de compilar el programa. En el caso del editor de páginas muestra el aspecto que tendrá la página de impresión cuando se imprima.

 Cuadrícula	Activa o desactiva la cuadricula que ayuda a distribuir con mayor precisión los controles.
 Bloquear	Permite bloquear el movimiento de los controles de la screen o de la página de impresión activa. Esta opción es útil aplicarla cuando se quiere seleccionar un control o un grupo de controles y no se desea que al seleccionarlo el control cambie de posición o tamaño.
 Paleta de Controles	Muestra u oculta la paleta de controles.
 Paleta de ajuste	Muestra u oculta la paleta de ajuste.
 Paleta de iconos	Muestra u oculta la paleta de iconos.
 Paleta de colores	Muestra u oculta la paleta de colores.
 Paleta de fuentes	Muestra u oculta la paleta de fuentes.

Para mostrar u ocultar la barra de herramientas (menú de botones de la aplicación) ejecute la opción ToolBar del menú View .

6. Edición de Controles

El Editor de Screen ofrece las siguientes posibilidades para la edición y manejo de controles de controles:

- Crear un control
- Deshacer los últimos cambios
- Rehacer modificaciones
- Cortar controles
- Copiar controles
- Pegar controles
- Borrar controles
- Seleccionar todos los controles.
- Editar las propiedades de un control
- Test.
- Cuadrícula.
- Bloquear el movimiento de controles.
- Duplicar controles.
- Mover controles.
- Cambiar de tamaño un control.
- Cambiar de padre un control.
- Asociar un icono a un control.
- Eliminar el icono asociado a un control
- Asociar una tabla a un control.
- Asociar una variable o una columna de una tabla a un control.
- Identificación automática de controles

Crear un control

Para Crear un control en la screen tiene dos métodos:

- *Primer método:*

1. Ejecute la opción del menú Controls asociado al control que desea Crear, o seleccione el control deseado en la paleta de controles
2. Haga clic sobre el punto de la screen donde desee añadir el control.
3. Arrastre el ratón hasta conseguir el tamaño deseado.
4. Suelte el boton (si suelta el botón en la misma posición en la que hizo click sin arrastrar se creará automáticamente el control con un tamaño por defecto).

- *Segundo método:*

1) Haga clic en la paleta de controles sobre el control que desee Crear.
2) Arrástrelo hasta la posición de la screen donde desee Crear el control.
3) Suelte el botón del ratón. Automáticamente se muestra el control en la screen.

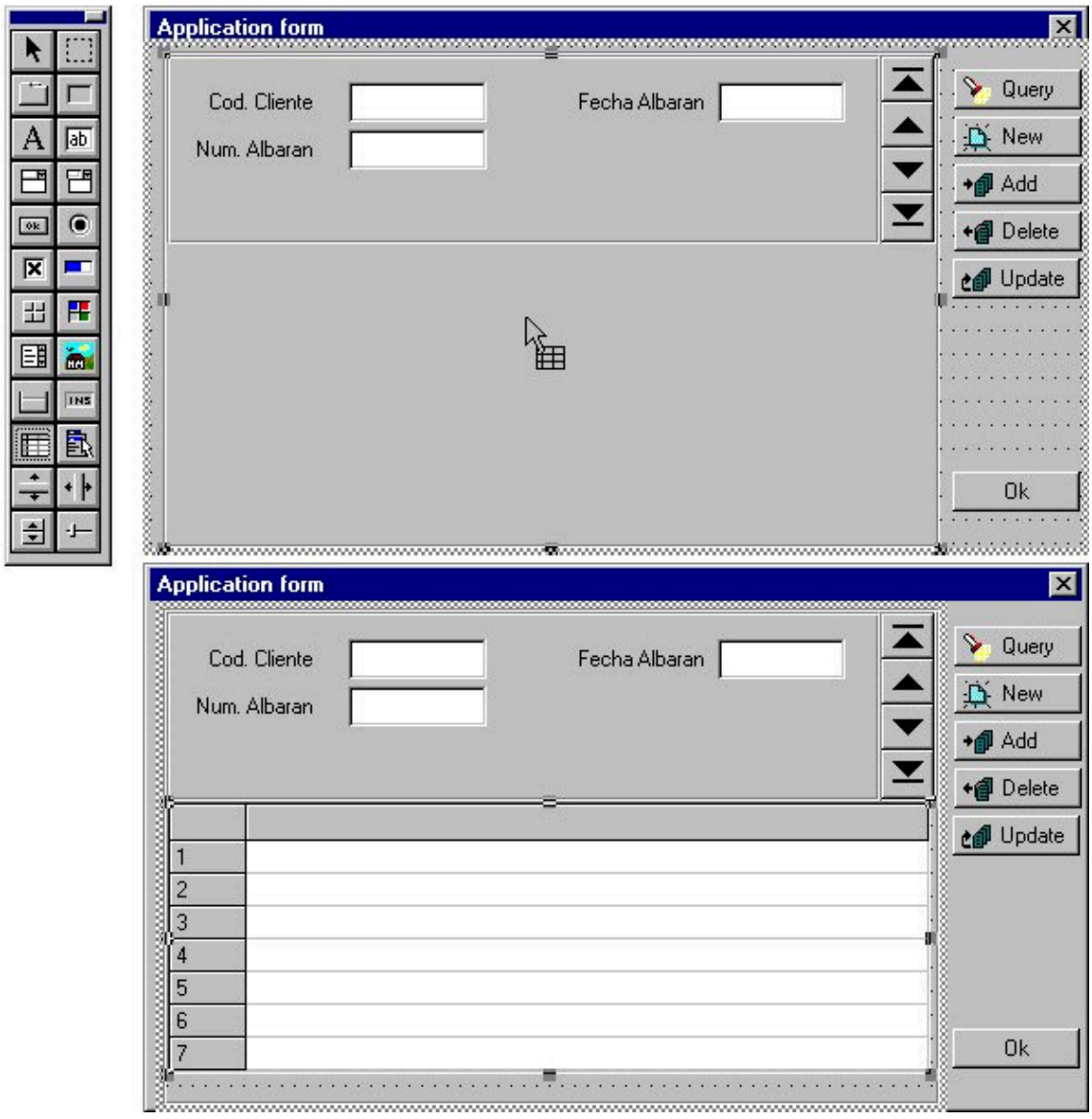


Figura 10. 7. Crear un control en la screen.

Editar las propiedades de un control

El Editor de Screen y el Editor de Páginas de impresión le permiten editar las propiedades de un control para poder personalizar sus características principales.

Para editar las propiedades de un control tiene las siguientes opciones:

1. Haga doble clic sobre el control.
2. Seleccione un control y pulse [Alt] + [Enter].
3. Ejecute la opción Properties del menú tipo popup que aparece al pulsar el botón derecho del ratón sobre el control del que se desean editar sus propiedades.
4. Seleccione un control y ejecute la opción Properties del menú Edit del menú principal de la aplicación.

Se muestra el cuadro de diálogo de edición de propiedades del elemento de la screen o de la página de impresión en curso que se tiene seleccionado en ese momento.

En el Editor de Páginas de impresión permite:

1. Editar las características generales y especiales del control de impresión seleccionado.
2. Editar las características de diseño de la página para definir los márgenes.

En el Editor de Screen permite:

1. Editar las características generales, especiales y de Storage del control seleccionado.
2. Editar las características generales de la screen
3. Editar las características del menú asociado a la screen, en caso de tenerlo.

Seleccionar todos los controles

El Editor permite seleccionar todos los controles hijos del control padre activo de la screen o página de impresión en curso para permitir la realización de determinadas operaciones (copiarlo, cortarlo, borrarlo, moverlo etc.).

Para seleccionar todos los controles hijos de un control padre proceda de la siguiente manera:

1. Seleccione como padre en curso el control que se desee y pulse las teclas [Ctrl] + [E].
2. Ejecute la opción Select All del menú tipo popup que aparece al pulsar el botón derecho del ratón sobre un control cualquiera de los que se desea seleccionar.
3. Seleccione como padre en curso el control que se desee y ejecute la opción Select All del menú Edit del menú de la aplicación.

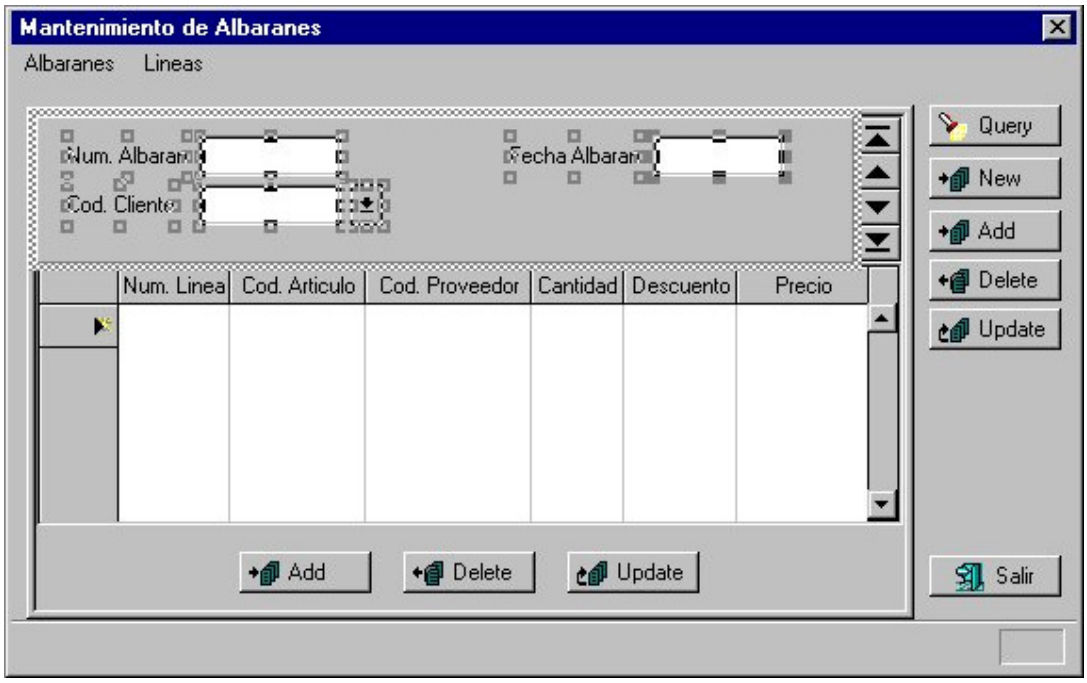


Figura23.8. Selección de todos los controles del formulario de la tabla albaranes

En la figura se puede observar que el padre en curso se encuentra marcado con una línea discontinua.

Puede excluir un control de la selección haciendo doble clic sobre él manteniendo la tecla [Mayúsculas] pulsada.

Duplicar un control

Para duplicar un control o un grupo de controles proceda de la siguiente manera:

1. Seleccione el control o controles que desee duplicar.
2. Arrastre los controles pulsando la tecla [Ctrl] y el botón izquierdo del ratón sobre uno de loas controles seleccionados.
3. Una copia del control/es original/es sigue al cursor, cuando este en la posición deseada suelte la tecla y el botón del ratón.

Al duplicar controles de esta forma no se copian en el portapapeles.

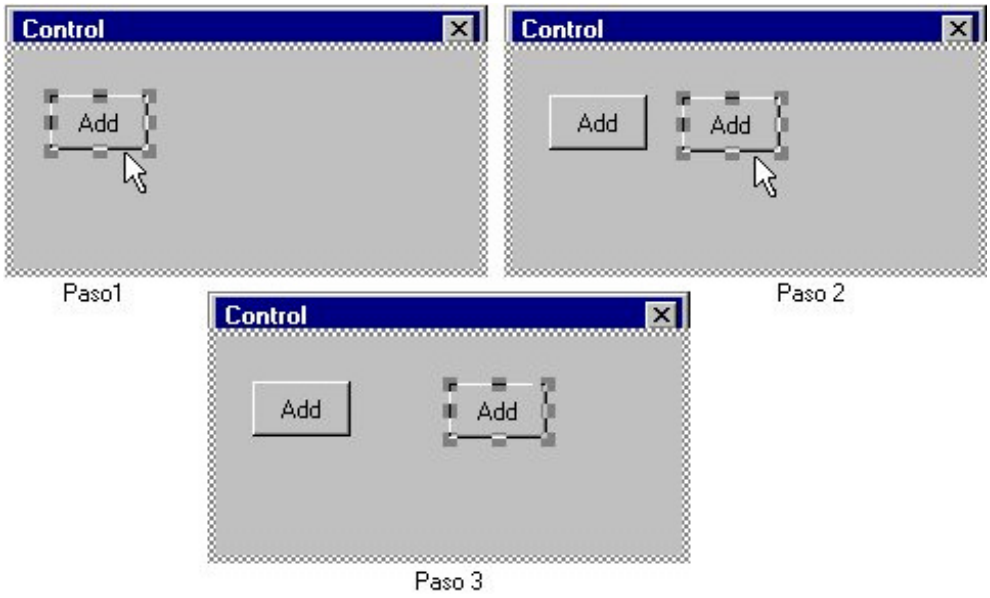


Figura 10. 9. Duplicar un control en una screen.

También puede duplicar un control ejecutando las opciones copiar y pegar (pulsando [Ctrl] + [C] y [Ctrl] + [V])

Mover controles

Puede distribuir los controles dentro de una screen o una página de impresión cambiándolos de posición.

Para mover un control o un conjunto de controles proceda de la siguiente manera:

1. Seleccione el control o controles que desee mover.
2. Con el botón izquierdo del ratón pulsado sobre uno de ellos arrástrelos hasta la posición deseada.
3. Suelte el botón del ratón.

Si desea precisar con mayor exactitud la posición del control, puede moverlo utilizando las teclas de dirección. Si esta trabajando con la rejilla el control se moverá a la siguiente posición de la cuadrícula, en caso contrario se moverá pixel a pixel.

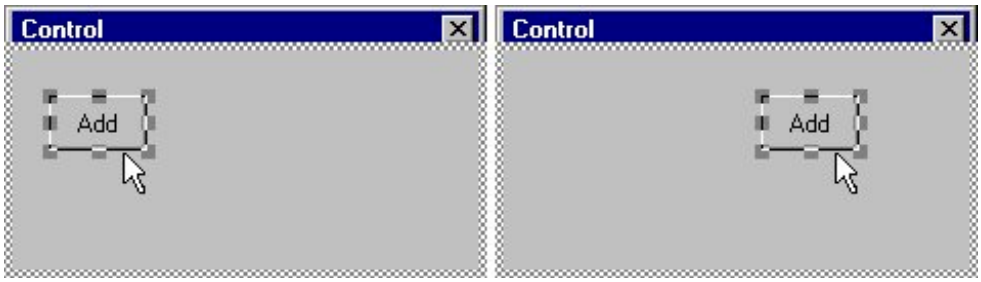


Figura 10. 10. Mover un control.

También puede mover un control o un grupo de controles utilizando las teclas:

[F. izquierda] y [F. derecha]	Mueve el control seleccionado hacia la izquierda o la derecha.
[F. arriba] y [F. abajo]	Mueve el control seleccionado hacia arriba o hacia abajo.

Un control solo se puede mover dentro de los contornos de su padre. Para sacar un control del padre que lo contiene utilice la tecla [Mayúsculas].

Cambiando de posición los controles cuyo padre sea un Grid puede modificar el orden de sus columnas.

Cambiar de tamaño un control

Al añadir un control a una screen o a una página de impresión este se muestra con un tamaño que hay definido por defecto para cada tipo de control. Pero el editor permite modificar el tamaño de los controles y que el usuario diseñe la screen o la página de impresión a su gusto.

Para cambiar el tamaño de un control proceda de la siguiente manera:

1. Seleccione el control haciendo clic sobre el o con el tabulador. Aparecen en su perímetro unos pequeños cuadros denominados cuadros de redimensionamiento, que sirven para cambiar de tamaño un control.
2. Al situar el cursor del ratón sobre uno de los cuadros de redimensionamiento, este cambia de forma indicando las direcciones en las que se puede modificar el tamaño del control. Los cuadros de las esquinas cambian el tamaño horizontal y verticalmente, el resto de cuadros de redimensionamiento permiten cambiar el tamaño horizontal o verticalmente.
3. Arrastre uno de los cuadros de redimensionamiento hasta obtener el tamaño deseado.
4. Suelte el botón del ratón.

Si desea precisar con mayor exactitud el tamaño del control, puede cambiarlo utilizando la tecla [mayúsculas] junto con las teclas de dirección:

[Mayúsculas]+[F. derecha/izquierda]	Modifica la anchura del control seleccionado.
[Mayúsculas]+[F. arriba/abajo]	Modifica la altura del control seleccionado..

Si esta trabajando con la rejilla el tamaño cambia según el tamaño de la cuadrícula, en caso contrario se cambiará pixel a pixel.

Borrar controles

En cualquier momento, puede eliminar un control de la screen o página de impresión activa. Para borrar un control o un grupo de controles proceda de la siguiente manera:

1. Seleccione en la screen o en la página de impresión el control o controles que desee borrar.
2. Pulse la opción Delete del menú Edit o la tecla [Del] ([Supr]).

Asociar un icono a un control

Como se ha mencionado anteriormente, los controles de una screen (Botón de comando, Caja; Grupo de Botones, Grupo de Cajas, Listas (List Box)) y los controles de ímpresión (Caja (Box) y Box Group) pueden tener asociado un icono que los represente gráficamente.

Para asociar un icono a un control proceda de la siguiente manera:

1. Muestre la paleta de iconos.
2. Seleccione en la lista desplegable el fichero de iconos que desee utilizar. Automáticamente se muestran en el panel los iconos de dicho fichero.
3. Haga clic sobre el icono que desee y arrástrelo sobre el control al que desee asignárselo.

A un Grupo de Botones, un Grupo de Cajas y una Lista de Cajas le puede asociar un icono a cada uno de sus elementos utilizando la misma técnica que se ha explicado anteriormente.

A los controles de tipo Lista (List Box) se le asocia el fichero de iconos, pero posteriormente por software deberá asociar a cada uno de los elementos de la lista un icono del fichero utilizando la propiedad Icon.

También se puede arrastrar el mismo icono de la paleta sobre varios controles a la vez siguiendo las siguientes instrucciones :

1. Seleccione con el botón izquierdo del ratón el icono que desee en la paleta y arrástrelo hasta el control (de la screen o página de impresión) cuyo icono desee cambiar.
2. Sin soltar el botón izquierdo del ratón pulse la barra espaciadora sobre todos los controles a los que desee cambiar el icono.

Eliminar el icono asociado a un control

El Editor de Screen y el Editor de Páginas de impresión permiten eliminar el icono que tiene asociado un control.

Si desea eliminar el icono que tiene asociado un control:

1. Seleccione el control
2. Edite las propiedades del control
3. Quite la marca de la casilla de verificación Icon de la ficha Special haciendo clic sobre ella. El icono es eliminado automáticamente.

En un Grupo de Botones, un Grupo de Cajas y en una Lista de Cajas se eliminarán los iconos de todos sus elementos.

Cambiar de padre un control

Como se mencionó anteriormente, el padre de un control es el control que lo contiene. El definir un control como hijo de otro facilita el trabajo de diseño y programación de la screen:

- Al mover el control padre se moverán con el todos los hijos.
- Al hacer invisible o inhabilitar un control padre se harán invisibles o se inhabilitarán los controles hijos.

No es necesario realizar las acciones mencionadas anteriormente para cada control por separado, es suficiente con realizarlas sobre el control padre y por tanto se facilita el trabajo al usuario y al programador.

Para cambiar de padre un control o un conjunto de controles proceda de la siguiente manera:

1. Seleccione los controles que desee cambiar de padre.
2. Maneniendo la tecla [Mayúsculas] pulsada, arrastre los controles hasta situarlos sobre el control que desea que actúe como padre, sacándolos por completo del contorno del padre en curso.
3. Suelte el botón izquierdo del ratón y a continuación suelte la tecla [Mayúsculas].

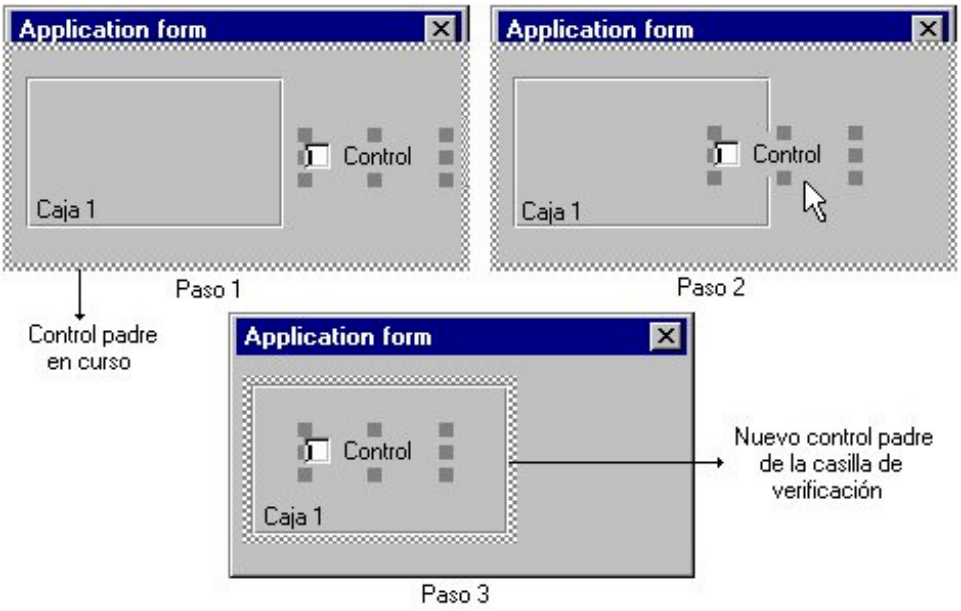


Figura 10. 11. Definir un control como hijo de otro en una screen.

En la figura se puede observar que inicialmente el control padre de la casilla de verificación era la caja principal de la screen, al final del proceso su control padre es la caja 1.

Los únicos controles que pueden actuar como padres de otros son los controles de tipo Ficha (Tab), barra (Bar), Grid, Caja (Box), Bitmap y los controles de impresión de tipo Grupo (Group), por que estos controles son los únicos que pueden contener un conjunto de controles.

Al añadir un control, automáticamente se le asigna como padre el control sobre el cual se a creado. Si se añade un control sobre una página o una screen, su padre será la página o la caja de la screen respectivamente.

Identificación automática de controles

La opción Automatic Id del submenú Preferences del menú Controls le permite identificar automáticamente los controles que añade a la screen. Es decir, si activa esta opción cada vez que se cree un control se le asignará automáticamente un identificador único. Editando las propiedades del control puede observar que se le ha dado un valor al campo Id .

Asociar una tabla a un control de la screen

Si desea hacer un programa de mantenimiento de una o más tablas de una base de datos asocie una tabla a un control de tipo:

- Ficha.
- Caja.
- Grid.
- Bitmap.

Para asociar una tabla a un control ejecute las instrucciones indicadas en uno de los tres métodos que se explican a continuación.

Primer método: Editando las propiedades del control

Para asociar un tabla a un control editando sus propiedades lo primero que debe hacer es anadir una tabla en el Form al que pertenece la screen y después siga las instrucciones de uno de los métodos que se explican a continuación.

- **En un control de tipo Ficha, Bitmap o Caja**:
  1. Edite las propiedades del control.
  2. Haga clic sobre la ficha Storage del cuadro de diálogo y marque la casilla de varificación Form Table .
  3. Seleccione en el campo Table la tabla del Form que desee asociar al control. En dicho campo se muestra la lista de tablas del Form que todavía no se han utilizado en la screen.
- **En un control de tipo Grid:**
  1. Edite las propiedades del control.
  2. Haga clic sobre la ficha Special del cuadro de diálogo y active la casilla de verificación Parent Grid .
  3. Haga clic sobre la ficha Storage del cuadro de diálogo y marque la casilla de varificación Form Table .
  4. Seleccione en el campo Table la tabla del Form que desee asociar al control. En dicho campo se muestra la lista de tablas del Form que todavía no se han utilizado en la screen.

Segundo método: Arrastrar desde un repositorio

1. Si el módulo en curso no tiene asociado un repositorio, abra uno de los repositorios incluidos en el proyecto, en caso contrario abra el que ya tiene asociado.
2. En la estructura en árbol de repositorio seleccione una tabla.
3. Arrastre la tabla seleccionada sobre un control contenedor (Tab, Bitmap, Grid o Box) sin tabla asociada.
4. Se muestra el cuadro de diálogo New table que permite definir sus características.
5. Automáticamente se añade la nueva tabla a la estructura en árbol del Form.
6. El control contenedor quedará asociado a la tabla y se crearán controles para cada una de las columnas de la tabla si se seleccionó la opción Add columns en el diálogo New table

Asociar una variable o una columna de una tabla a un control de una screen

Se puede asociar una variable o una columna de una tabla definida en el Form en curso a un control de tipo:

- Lista desplegable (Drop Edit y Drop List).
- Texto.
- Edición.
- Casilla de Verificación (Check Box).
- Barra de porcentaje (Percentage Box).
- Grupo de Cajas (Box Group).
- Grupo de Botones (Button Group).
- Panel.

Para asociar una variable o una columna de una tabla a un control siga las instrucciones de uno de los métodos que se explican a continuación :

Primer método: Editando las propiedades del control

1. Edite las propiedades del control
2. Haga clic sobre la ficha Storage del cuadro de diálogo y active la casilla de verificación Variable .
3. Seleccione en el campo Variable el elemento que desea asociar al control. Si el control padre del control en curso tiene una tabla asociada, en dicho campo se muestra se muestra una lista de las columnas de dicha tabla. En caso contrario se muestra una lista de las variables del Form compatibles con el control y que todavía no han sido asociadas a ningún control. También puede pulsar el botón New para crear una variable o columna nueva.

Segundo método: Arrastrar desde un repositorio

Los repositorio pueden ser utilizados como paletas dentro del Editor Visual de Cosmos. El Editor de Repositorios permite la definición del tipo de interfaz gráfico para cada una de las columnas de la base de datos. En el momento de arrastrar una columna desde una paleta del repositorio hasta una screen de un Form, esta columna aparecerá en la screen con el interfaz gráfico que se haya definido.

1. Si el módulo en curso no tiene asociado un repositorio, abra uno de los repositorios incluidos en el proyecto en caso contrario abra el que ya tiene asociado.
2. En la estructura en árbol de repositorio seleccione una columna de una tabla .
3. Arrastre la columna seleccionada sobre un control contenedor (Ficha, Bitmap o Caja) que esté asociado a la tabla a la que pertenece la columna ó que todavía no esté asociado a ninguna tabla.
4. Automáticamente se crean tantos controles en el contenedor como columnas arrastradas. Cuando se crea un repositorio se define el tipo de control asociado a cada una de las columnas, para su representación gráfica en la screen.
5. Si la tabla aún no existe en el Form,se muestra el cuadro de diálogo New table que permite definir sus características para añadirla a la estructura en árbol del Form. El contenedor quedará asociado a la tabla.

Tercer método: Arrastrar desde el Editor de columnas:

1. Edite las columnas de la tabla del Form que desea asociar al control.
2. Seleccione una o varias columnas de la tabla y arrástrelas sobre el control contenedor (Tab, Bitmap, Grid o Box) que esté asociado esa tabla ó sobre uno que todavía no esté asociado a ninguna tabla.
3. Automáticamente se crean tantos controles en el contenedor como columnas arrastradas.
4. El contenedor quedará asociado a la tabla, si no lo estaba ya.

Cuarto método: Arrastrar desde el Editor de variables del Form:

Este método también lo puede utilizar para arrastrar variables sobre un control de tipo Bitmap, Grid, Ficha o Caja.

1. Edite las variables del Form
2. Seleccione una de las variables del Form y arrástrela sobre la screen.
3. Automáticamente se obtiene la representación gráfica en la screen de la variable arrastrada.

Capítulo 11
Editor de Páginas de Impresión

Contenidos

1. Introducción

El Editor de Páginas de impresión es una herramienta que permite diseñar el formato de una Página (clase Page) para crear los impresos, formularios y listados de una aplicación.

Utilizando este editor podrá:

1. Añadir, borrar y modificar los controles de impresión.
2. Editar y modificar las propiedades de un control de impresión.
3. Mover, alinear, centrar, ajustar, espaciar y redimensionar un grupo de controles.
4. Distribuir los controles en la página de impresión.
5. Bloquear el movimiento de los controles de la página de impresión activa. Esta opción es útil aplicarla cuando se quiere seleccionar un control o un grupo de controles y no se desea que al seleccionarlo el control cambie de posición o tamaño.
6. Asociar iconos a cierto tipo de controles.
7. Utilizar diferentes colores.
8. Utilizar diferentes tipos de fuentes.
9. Definir el orden en el que se dibujarán los controles dentro de la página de impresión o de su control padre.
10. Definir un control como padre de un grupo de controles.
11. Realizar una presentación preliminar para ver el aspecto que tendrá una página de impresión cuando se imprima.
12. Arrastrar tablas o columnas de un repositorio a la página de impresión.

Para realizar las acciones mencionadas anteriormente el Editor de Páginas de impresión de Cosmos dispone de los siguientes elementos:

- Un conjunto de paletas.
- Una barra de herramientas.
- Una barra de estado para mostrar información al usuario.

Este Editor hace uso de los botones derecho e izquierdo del ratón, indicándose oportunamente aquellos casos en que es necesario utilizar el botón derecho. Si no se indica ninguno se asumirá por defecto el botón izquierdo, el más utilizado habitualmente.

Las modificaciones realizadas en este editor se guardan en el código fuente del módulo (programa, librería o include) al que pertenece la página de impresión.

2. Paletas

El Editor de Páginas de impresión de Cosmos incluye un conjunto de paletas que permiten la programación de un alto porcentaje de la aplicación arrastrando objetos hacia un lugar determinado del programa:

Paleta de controles	Permite añadir un control de impresión a la página.
Paleta de colores	Permite cambiar el color de fondo y el color de primer plano de los elementos de la página de impresión.
Paleta de iconos	Permite asociar un icono a cierto tipo de controles.
Paleta de ajuste	Permite alinear, centrar, ajustar, espaciar y redimensionar un grupo de controles.
Paleta de fuentes	Esta paleta permite modificar el font definido por defecto para la página de impresión o el font asociado a uno de sus controles.

Paleta de controles de impresión

En esta paleta se encuentran todas las opciones necesarias para dibujar en la página de impresión cualquier tipo de control. Después de añadir el control, edite sus propiedades para modificar su estilo y sus características.

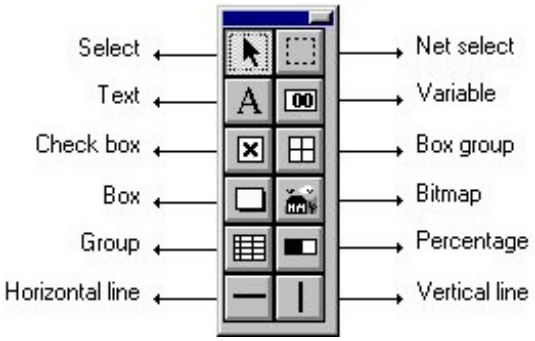


Figura 9.1. Paleta de controles de impresión.

Las opciones que se encuentran en esta paleta son las que se indican a continuación:

Select	Herramienta que permite seleccionar, mover, copiar, borrar, dimensionar y editar las propiedades de un control.
Net select	Herramienta que permite seleccionar un grupo de controles para copiar, borrar, mover o editar en conjunto
Text	Herramienta para crear controles de texto .
Variable	Herramienta para crear controles de tipo variable.
Check box	Herramienta para crear casillas de verificación.
Box group	Herramienta para crear grupos de cajas.
Box	Herramienta para crear controles de tipo caja .
Bitmap	Herramienta para crear controles de tipo bitmap.
Grupo	Herramienta para crear controles de tipo Grupo.
Percentage box	Herramienta para crear barras de porcentaje o progreso.
Horizontal line	Herramienta para crear líneas horizontal.
Vertical line	Herramienta para crear líneas verticales.

Todas las opciones de la paleta de controles de impresión también se encuentran en el menú Controls del menú principal de la aplicación.

Los valores por defecto de los controles de esta paleta pueden ser personalizados a gusto del programador a través del menú que aparece al pulsar en la paleta con el botón derecho del ratón, estos valores se pueden mantener durante la sesión o bien guardar para posteriores sesiones.

Para mostrar u ocultar la paleta de controles de impresión tiene 2 opciones:

- Ejecute la opción Objects Palette del menú View .

Pulse el botón  del menú de botones.

Paleta de Colores

Esta paleta permite cambiar el color de fondo (background) y el color de primer plano (foreground) de los controles de una screen o una página de impresión.

Figura 9.2. Paleta de colores.

El botón  de la paleta de colores permite seleccionar el color definido por defecto para el control sobre el que se arrastre.

El botón  de la paleta de colores permite seleccionar el color transparente.

Para cambiar el color de fondo (background):

1. Pulse el botón  .
2. Seleccione con el botón izquierdo de ratón el color que desee en la paleta y arrástrelo hasta el control de la screen o página de impresión que desee cambiar de color.

Para cambiar el color de primer plano (foreground):

1. Pulse el botón  .
2. Seleccione con el botón izquierdo de ratón el color que desee en la paleta y arrástrelo hasta el control de la screen o página de impresión que desee cambiar de color.

También se puede arrastrar el mismo color de la paleta sobre varios controles a la vez siguiendo las siguientes instrucciones :

1. Seleccione con el botón izquierdo del ratón el color que desee en la paleta y arrástrelo hasta el control (de la screen o página de impresión) cuyo color desee cambiar.
2. Sin soltar el botón izquierdo del ratón pulse la barra espaciadora sobre todos los controles a los que desee cambiar el color.

Para mostrar u ocultar la paleta de colores tiene 2 opciones:

1. Ejecute la opción Colors Palette del menú View .
2. Pulse el botón  del menú de botones.

Paleta de Fuentes

La paleta de fuentes tiene el aspecto que se muestra en la figura.

Figura 9.3. Paleta de fonts.

Esta paleta permite modificar:

- El font asociado a un control de una página de impresión o una screen.
- El font definido por defecto para la página de impresión o la screen.

El primer font que aparece en la paleta es el font seleccionado por defecto para la página de impresión o la screen. Por defecto este font es de tipo MS Sans Serif y tamaño 8.

Para modificar el tipo de fuente proceda de la siguiente manera:

1. Seleccione la fuente deseada en la paleta.
2. Si desea modificar la fuente por defecto, arrastre y suelte la fuente seleccionada sobre la página de impresión o sobre la screen.

Si desea modificar la fuente asociada a un control arrastre y suelte la fuente seleccionada sobre dicho control.

Si pulsa el botón Añadir de la paleta se muestra un cuadro de diálogo idéntico al de la opción Font del menú Layout que permite seleccionar una fuente para agregarla a la paleta.

Al modificar el tipo de fuente que utiliza por defecto la screen o la página de impresión, solo se modificará el de los controles que utilizan la fuente definida por defecto.

También puede modificar el tipo de fuente ejecutando la opción Font del menú Layout o editando las propiedades del control.

Para mostrar u ocultar la paleta de fuentes tiene 2 opciones:

1. Ejecute la opción Fonts Palette del menú View .
2. Pulse el botón  del menú de botones.

Paleta de Ajuste

La paleta de ajuste tiene las opciones necesarias para distribuir los controles sobre la screen o página de impresión activa y redimensionarlos.

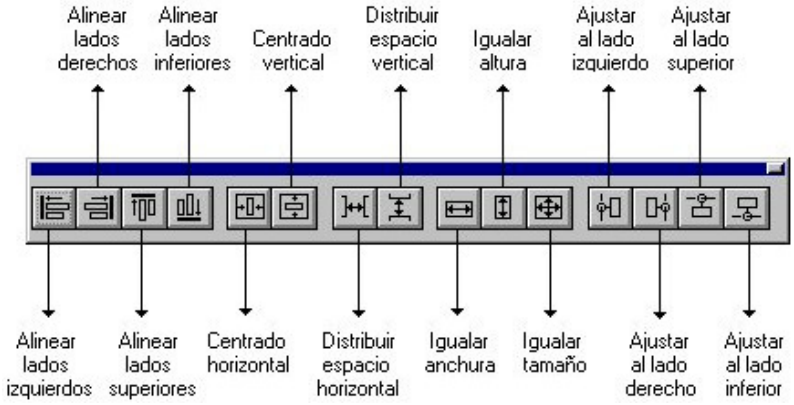


Figura 9.4. Paleta de ajuste.

Las herramientas de la paleta de ajuste estarán activas o inactivas dependiendo del numero de controles seleccionados, ya que ciertas operaciones solo se pueden realizar sobre un único control.

En algunas de las operaciones en las que intervienen más de un control hay uno de ellos que se considera dominante (el primero que se seleccionó, se marca de distinta forma) y que se toma como referencia.

Elemento	Efecto
Alinear lados izquierdos	Alinea los lados izquierdos de los controles seleccionados
Alinear lados derechos	Alinea los lados derechos de los controles seleccionados
Alinear lados superiores	Alinea los lados superiores de los controles seleccionados
Alinear lados inferiores	Alinea los lados inferiores de los controles seleccionados
Centrado horizontal	Centra horizontalmente los controles seleccionados
Centrado Vertical	Centra verticalmente los controles seleccionados
Distribuir espacio horizontal	Distribuye horizontalmente los controles seleccionados
Distribuir espacio vertical	Distribuye verticalmente los controles seleccionados
Igualar anchura	Iguala la anchura de los controles seleccionados
Igualar altura	Iguala la altura de los controles seleccionados
Igualar tamaño	Iguala el tamaño de los controles seleccionados
Ajustar al lado izquierdo	Fija la posición al lado izquierdo del padre
Ajustar al lado derecho	Fija la posición al lado derecho del padre
Ajustar al lado superior	Fija la posición al lado superior del padre
Ajustar al lado inferior	Fija la posición al lado inferior del padre

Todas las opciones de la paleta de ajuste también se encuentran en el menú Layout del menú principal de la aplicación.

Para mostrar u ocultar la paleta de ajuste tiene 2 opciones:

- Ejecute la opción Adjust Palette del menú View .
- Pulse el botón  del menú de botones.

Paleta de Iconos

La paleta de iconos tiene el aspecto que se muestra a continuación:

Figura 9.5. Paleta de iconos.

Utilizando la técnica de arrastrar y soltar (drag & drop), esta paleta permite:

1. Asociar iconos a los controles de una screen de tipo: botón de comando (Push Button), caja (Box), Grupo de botones (Button Group), Grupo de cajas (Box Group), listas (List box).
2. Asociar iconos a los controles de impresión de tipo caja (Box) y Grupo de cajas (Box Group).
3. Asociar iconos a los ítems de un menú.

En la lista desplegable de la paleta se muestran los ficheros de iconos existentes en la sección [Icons] del fichero de configuración COSMOS.INI . Una vez seleccionado el fichero se muestra su contenido, si la totalidad de los iconos no cabe en la paleta podrán verse los restantes pulsando los botones  y  .

Para mostrar u ocultar la paleta de iconos tiene 2 opciones:

- Ejecute la opción Icons Palette del menú View .
- Pulse el botón  del menú de botones.

3. Controles de Impresión

Text 

Esta opción permite crear un control de tipo texto en la página de impresión. Este tipo de control se utiliza para cuando se quiere añadir a la página un texto que no va a variar en tiempo de ejecución, es decir un texto estático. Por ejemplo el título del formulario o impreso que se esta diseñando.

Este tipo de control puede ser de dos tipos:

Texto simple	Texto estático en una sola línea. El texto puede ser centrado o alineado a la derecha (por defecto lo esta a la izquierda), etc.
Texto multilínea	Como su nombre indica puede tener varias líneas de texto. El texto puede estar centrado o alineado a la derecha (por defecto lo esta a la izquierda),etc.

Después de crear el control, edite sus propiedades para modificar su estilo y sus características.

Variable 

Este tipo de control se utilizará para mostrar en la página de impresión un dato que varíe en tiempo de ejecución, como puede ser por ejemplo el total facturado por proveedor.

Este control puede ser de dos tipos:

- Variable simple Permite por ejemplo mostrar variables o campos de una base de datos en una línea de texto. El texto puede ser centrado o alineado a la derecha (por defecto lo esta a la izquierda), etc.
- Variable multilínea: Similares a los anteriores excepto que como su nombre indica, se pueden mostrar varias líneas de texto. El texto puede estar centrado o alineado a la derecha (por defecto lo esta a la izquierda),etc.

Después de crear el control, edite sus propiedades para modificar su estilo y sus características generales y especiales.

Check Box 

Un control de tipo casilla de verificación (Check Box) puede estar activo o inactivo. Cuando esté activado, su casilla de verificación aparece marcada generalmente con una equis.

Después de crear el control, edite sus propiedades para modificar su estilo y características.

Box 

La utilidad principal de un control de tipo caja (Box) es agrupar un conjunto de controles que guarden alguna relación entre si, para facilitar al usuario la programación. Usando este control puede definir múltiples páginas para un mismo área de la página de impresión. Cada página agrupará un conjunto de controles que se mostrarán cuando el usuario habilite dinámicamente por software la correspondiente página.

Después de crear el control, edite sus propiedades para modificar su estilo y sus características generales y especiales.

Box Group 

A cada una de las cajas se le puede asociar una etiqueta y un icono.

Después de crear el control, edite sus propiedades para modificar su estilo y sus características generales y especiales.

Bitmap 

Esta control permite agregar una imagen (fichero con extensión .bmp) a la página de impresión. Un bitmap es una representación digital de una imagen. Cada pixel en la imagen corresponde a uno o más bits en el bitmap . Los bitmap monocromos soló requieren un bit por pixel ; los bitmap de color requieren bits adicionales para indicar el color de cada pixel.

Este control puede ser contenedor (padre) de otros controles. Esta característica le permite utilizar un bitmap de fondo en su página de impresión. Usando este control puede definir múltiples páginas para un mismo área de la página de impresión. Cada página agrupará un conjunto de controles que se mostrarán cuando el programador habilite dinámicamente por software la correspondiente página.

Group 

Un Grupo es un control de impresión que sirve para agrupar un conjunto de controles que se van a repetir por filas y/o columnas. La repetición de los controles del grupo se define en las propiedades del control. Estas repeticiones no se reflejan durante la edición de la página de impresión, sino al ejecutar la opción Test del menú Edit o la opción Print Preview del menú File .

Después de crear el control, edite sus propiedades para modificar su estilo y sus características generales y especiales.

Percentage 

Es una barra que se utiliza para mostrar porcentajes, también puede servir para hacer un diagrama de barras.

Después de crear el control, edite sus propiedades para modificar su estilo y características.

Horizontal/Vertical Line 

Una línea vertical u horizontal se puede utilizar por ejemplo para separar un grupo de controles que guardan alguna relación del resto.

Después de crear el control, edite sus propiedades para modificar su estilo y características.

4. Barra de Estado y Barra de Herramientas

Barra de Estado

La barra de estado muestra información y mensajes en la parte inferior de la ventana para ayudarle a utilizar el Editor Visual.

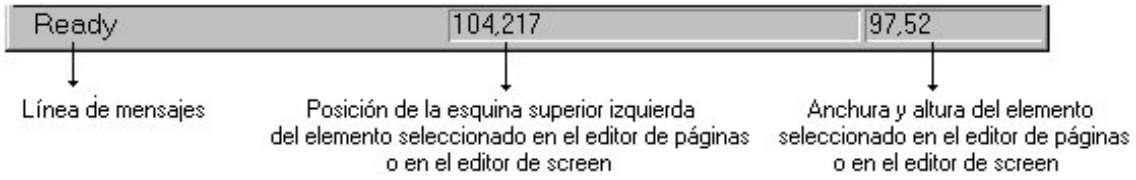


Figura11.6. Barra de estado.

Para mostrar u ocultar la barra de estado ejecute la opción Status Bar del menú View .

Barra de Heramientas

Las barras de herramientas contienen botones que le permiten acceder rápidamente con el ratón a muchos comandos y funciones del Editor. A continuación se explican las opciones relacionadas con el Editor de Páginas.

 Cortar	Elimina el control o controles seleccionados y los almacena en el portapapeles de Windows para permitir su inserción en otro punto de la página de impresión o de la screen activa o en otra distinta.
 Copiar	Copia el control o controles seleccionados al portapapeles de Windows para permitir su inserción en otro lugar de la misma página de impresión o de la screen o en otra distinta.
 Pegar	Inserta en la screen o página de impresión activa el control o controles previamente copiados al portapapeles de Windows.
 Deshacer	Deshace los últimos cambios.
 Rehacer	Recupera los cambios que han sido anulados mediante la opción Deshacer .
 Test	Simula el comportamiento en ejecución de la screen desde el Editor Visual sin necesidad de compilar el programa. En el caso del editor de páginas muestra el aspecto que tendrá la página de impresión cuando se imprima.
 Cuadrícula	Activa o desactiva la cuadricula que ayuda a distribuir con mayor precisión los controles.
 Bloquear	Permite bloquear el movimiento de los controles de la screen o de la página de impresión activa. Esta opción es útil aplicarla cuando se quiere seleccionar un control o un grupo de controles y no se desea que al seleccionarlo el control cambie de posición o tamaño.
 Paleta de Controles	Muestra u oculta la paleta de controles.
 Paleta de ajuste	Muestra u oculta la paleta de ajuste.
 Paleta de iconos	Muestra u oculta la paleta de iconos.
 Paleta de colores	Muestra u oculta la paleta de colores.
 Paleta de fuentes	Muestra u oculta la paleta de fuentes.

Para mostrar u ocultar la barra de herramientas (menú de botones de la aplicación) ejecute la opción ToolBar del menú View .

5. Edición de Controles

Crear un control de impresión

Para crear un control de impresión en la página tiene dos métodos:

- **Primer método:**

    1. Ejecute la opción del menú Controls asociado al control que desea crear, o seleccione el control deseado en la paleta de controles de impresión.
    2. Haga clic sobre el punto de la screen donde desee añadir el control.
    3. Arrastre el ratón hasta conseguir el tamaño deseado.
    4. Suelte el botón (si suelta el botón en la misma posición en la que hizo click sin arrastrar se creará automáticamente el control con un tamaño por defecto).

- **Segundo método:**

    1. Haga clic en la paleta de controles de impresión sobre el control que desee dibujar.
    2. Arrástrelo hasta la posición de la página donde desee dibujar el control.
    3. Suelte el botón del ratón.

    Una vez dibujado el control edite sus Propiedades para modificar sus características.

Editar las propiedades de un control

El Editor de Screen y el Editor de Páginas de impresión le permiten editar las propiedades de un control para poder personalizar sus características principales.

Para editar las propiedades de un control tiene las siguientes opciones:

1. Haga doble clic sobre el control.
2. Seleccione un control y pulse [Alt] + [Enter].
3. Ejecute la opción Properties del menú tipo popup que aparece al pulsar el botón derecho del ratón sobre el control del que se desean editar sus propiedades.
4. Seleccione un control y ejecute la opción Properties del menú Edit del menú principal de la aplicación.

Se muestra el cuadro de diálogo de edición de propiedades del elemento de la screen o de la página de impresión en curso que se tiene seleccionado en ese momento.

En el Editor de Páginas de impresión permite:

1. Editar las características generales y especiales del control de impresión seleccionado.
2. Editar las características de diseño de la página para definir los márgenes.

En el Editor de Screen permite:

1. Editar las características generales, especiales y de Storage del control seleccionado.
2. Editar las características generales de la screen
3. Editar las características del menú asociado a la screen, en caso de tenerlo.

Seleccionar todos los controles

El Editor permite seleccionar todos los controles hijos del control padre activo de la screen o página de impresión en curso para permitir la realización de determinadas operaciones (copiarlo, cortarlo, borrarlo, moverlo etc.).

Para seleccionar todos los controles hijos de un control padre proceda de la siguiente manera:

1. Seleccione como padre en curso el control que se desee y pulse las teclas [Ctrl] + [E].
2. Ejecute la opción Select All del menú tipo popup que aparece al pulsar el botón derecho del ratón sobre un control cualquiera de los que se desea seleccionar.
3. Seleccione como padre en curso el control que se desee y ejecute la opción Select All del menú Edit del menú de la aplicación.

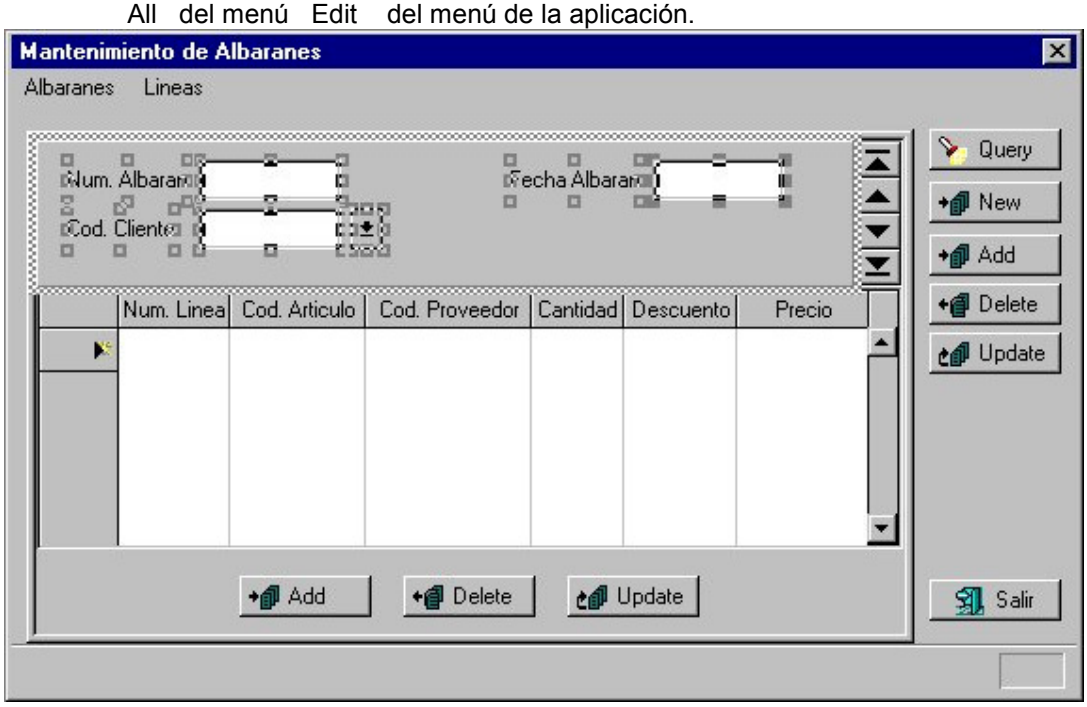


Figura11.7 . Selección de todos los controles del formulario de la tabla albaranes

En la figura se puede observar que el padre en curso se encuentra marcado con una línea discontinua.

Puede excluir un control de la selección haciendo doble clic sobre él manteniendo la tecla [Mayúsculas] pulsada.

Duplicar un control

Para duplicar un control o un grupo de controles proceda de la siguiente manera:

1. Seleccione el control o controles que desee duplicar.
2. Arrastre los controles pulsando la tecla [Ctrl] y el botón izquierdo del ratón sobre uno de loas controles seleccionados.
3. Una copia del control/es original/es sigue al cursor, cuando este en la posición deseada suelte la tecla y el botón del ratón.

Al duplicar controles de esta forma no se copian en el portapapeles.

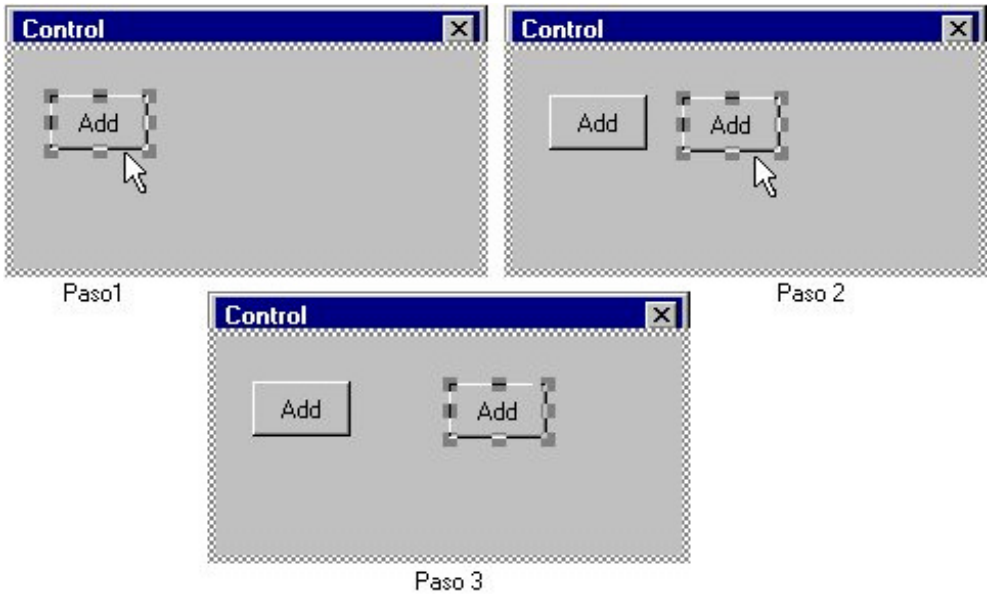


Figura 11.8 . Duplicar un control en una screen.

También puede duplicar un control ejecutando las opciones copiar y pegar (pulsando [Ctrl] + [C] y [Ctrl] + [V])

Mover controles

Puede distribuir los controles dentro de una screen o una página de impresión cambiándolos de posición.

Para mover un control o un conjunto de controles proceda de la siguiente manera:

1. Seleccione el control o controles que desee mover.
2. Con el botón izquierdo del ratón pulsado sobre uno de ellos arrástrelos hasta la posición deseada.
3. Suelte el botón del ratón.

Si desea precisar con mayor exactitud la posición del control, puede moverlo utilizando las teclas de dirección. Si esta trabajando con la rejilla el control se moverá a la siguiente posición de la cuadrícula, en caso contrario se moverá pixel a pixel.

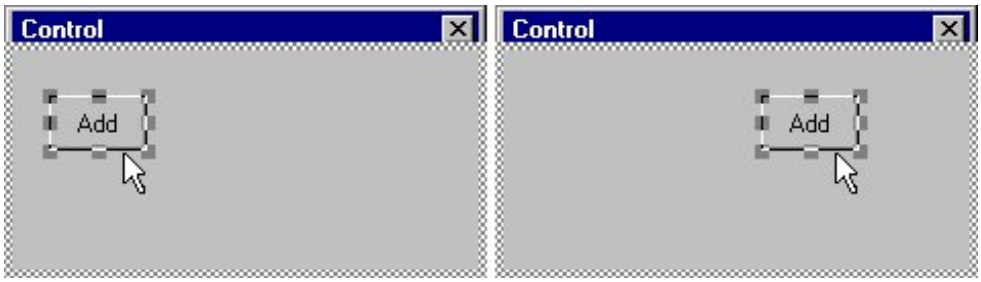


Figura 11. 9. Mover un control.

También puede mover un control o un grupo de controles utilizando las teclas:

[F. izquierda] y [F. derecha]	Mueve el control seleccionado hacia la izquierda o la derecha.
[F. arriba] y [F. abajo]	Mueve el control seleccionado hacia arriba o hacia abajo.

Un control solo se puede mover dentro de los contornos de su padre. Para sacar un control del padre que lo contiene utilice la tecla [Mayúsculas].

Cambiando de posición los controles cuyo padre sea un Grid puede modificar el orden de sus columnas.

Cambiar de tamaño un control

Al añadir un control a una screen o a una página de impresión este se muestra con un tamaño que hay definido por defecto para cada tipo de control. Pero el editor permite modificar el tamaño de los controles y que el usuario diseñe la screen o la página de impresión a su gusto.

Para cambiar el tamaño de un control proceda de la siguiente manera:
1. Seleccione el control haciendo clic sobre el o con el tabulador. Aparecen en su perímetro unos pequeños cuadros denominados cuadros de redimensionamiento, que sirven para cambiar de tamaño un control.
2. Al situar el cursor del ratón sobre uno de los cuadros de redimensionamiento, este cambia de forma indicando las direcciones en las que se puede modificar el tamaño del control. Los cuadros de las esquinas cambian el tamaño horizontal y verticalmente, el resto de cuadros de redimensionamiento permiten cambiar el tamaño horizontal o verticalmente.
3. Arrastre uno de los cuadros de redimensionamiento hasta obtener el tamaño deseado.
4. Suelte el botón del ratón.

Si desea precisar con mayor exactitud el tamaño del control, puede cambiarlo utilizando la tecla [mayúsculas] junto con las teclas de dirección:

[Mayúsculas]+[F. derecha/izquierda]	Modifica la anchura del control seleccionado.
[Mayúsculas]+[F. arriba/abajo]	Modifica la altura del control seleccionado..

Si esta trabajando con la rejilla el tamaño cambia según el tamaño de la cuadrícula, en caso contrario se cambiará pixel a pixel.

Borrar controles

En cualquier momento, puede eliminar un control de la screen o página de impresión activa. Para borrar un control o un grupo de controles proceda de la siguiente manera:
1. Seleccione en la screen o en la página de impresión el control o controles que desee borrar.
2. Pulse la opción Delete del menú Edit o la tecla [Del] ([Supr]).

Asociar un icono a un control

Como se ha mencionado anteriormente, los controles de una screen (Botón de comando, Caja; Grupo de Botones, Grupo de Cajas, Listas (List Box)) y los controles de ímpresión (Caja (Box) y Box Group) pueden tener asociado un icono que los represente gráficamente.

Para asociar un icono a un control proceda de la siguiente manera:
1. Muestre la paleta de iconos.
2. Seleccione en la lista desplegable el fichero de iconos que desee utilizar. Automáticamente se muestran en el panel los iconos de dicho fichero.
3. Haga clic sobre el icono que desee y arrástrelo sobre el control al que desee asignárselo.

A un Grupo de Botones, un Grupo de Cajas y una Lista de Cajas le puede asociar un icono a cada uno de sus elementos utilizando la misma técnica que se ha explicado anteriormente.

A los controles de tipo Lista (List Box) se le asocia el fichero de iconos, pero posteriormente por software deberá asociar a cada uno de los elementos de la lista un icono del fichero utilizando la propiedad Icon.

También se puede arrastrar el mismo icono de la paleta sobre varios controles a la vez siguiendo las siguientes instrucciones :

1. Seleccione con el botón izquierdo del ratón el icono que desee en la paleta y arrástrelo hasta el control (de la screen o página de impresión) cuyo icono desee cambiar.
2. Sin soltar el botón izquierdo del ratón pulse la barra espaciadora sobre todos los controles a los que desee cambiar el icono.

Eliminar el icono asociado a un control

El Editor de Screen y el Editor de Páginas de impresión permiten eliminar el icono que tiene asociado un control.

Si desea eliminar el icono que tiene asociado un control:

1. Seleccione el control
2. Edite las propiedades del control
3. Quite la marca de la casilla de verificación Icon de la ficha Special haciendo clic sobre ella. El icono es eliminado automáticamente.

En un Grupo de Botones, un Grupo de Cajas y en una Lista de Cajas se eliminarán los iconos de todos sus elementos.

Cambiar de padre un control

Como se mencionó anteriormente, el padre de un control es el control que lo contiene. El definir un control como hijo de otro facilita el trabajo de diseño y programación de la screen:

- Al mover el control padre se moverán con el todos los hijos.
- Al hacer invisible o inhabilitar un control padre se harán invisibles o se inhabilitarán los controles hijos.

No es necesario realizar las acciones mencionadas anteriormente para cada control por separado, es suficiente con realizarlas sobre el control padre y por tanto se facilita el trabajo al usuario y al programador.

Para cambiar de padre un control o un conjunto de controles proceda de la siguiente manera:

1. Seleccione los controles que desee cambiar de padre.
2. Maneniendo la tecla [Mayúsculas] pulsada, arrastre los controles hasta situarlos sobre el control que desea que actúe como padre, sacándolos por completo del contorno del padre en curso.
3. Suelte el botón izquierdo del ratón y a continuación suelte la tecla [Mayúsculas].

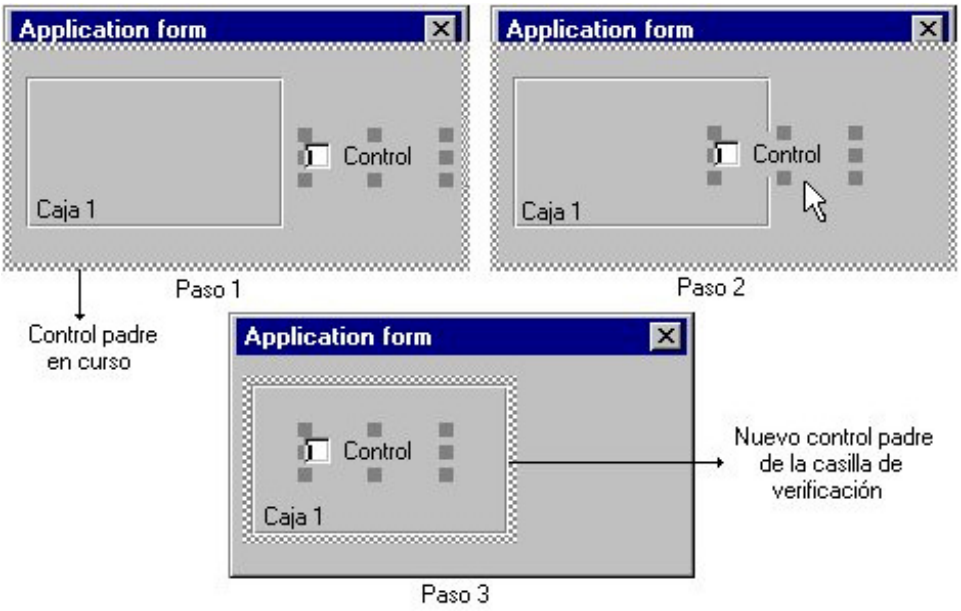


Figura 11. 10. Definir un control como hijo de otro en una screen.

En la figura se puede observar que inicialmente el control padre de la casilla de verificación era la caja principal de la screen, al final del proceso su control padre es la caja 1.

Los únicos controles que pueden actuar como padres de otros son los controles de tipo Ficha (Tab), barra (Bar), Grid, Caja (Box), Bitmap y los controles de impresión de tipo Grupo (Group), por que estos controles son los únicos que pueden contener un conjunto de controles.

Al añadir un control, automáticamente se le asigna como padre el control sobre el cual se a creado. Si se añade un control sobre una página o una screen, su padre será la página o la caja de la screen respectivamente.

Identificación automática de controles

La opción Automatic Id del submenú Preferences del menú Controls le permite identificar automáticamente los controles que añade a la screen. Es decir, si activa esta opción cada vez que se cree un control se le asignará automáticamente un identificador único. Editando las propiedades del control puede observar que se le ha dado un valor al campo Id .

Asociar una variable a un control de impresión

Se puede asociar una variable de la página a un control de impresión si su tipo es:

- Variable.
- Casilla de Verificación (Check Box).

Para asociar una variable a un control de impresión tiene tres métodos:

- **Primer método: Editando las propiedades del control de impresión**

Para asociar una variable de la página de impresión proceda de la siguiente manera:

1. Edite las propiedades del control.
2. Haga clic sobre la ficha Storage del cuadro de diálogo y active la casilla de verificación Variable.
3. Seleccione en el campo Variable el elemento que desea asociar al control de impresión. En este campo se muestra una lista de las variables de la página de impresión compatibles con el control y que no estén ya asociadas a otro control de la página de impresión. También puede pulsar el botón New para crear una variable nueva.

- **Segundo método: Arrastrar desde un repositorio**

Véase el Apartado siguiente Arrastrar tablas o columnas a un control de impresión.

- **Tercer método: Arrastrar desde el Editor de variables de la página:**

Este método también lo puede utilizar para arrastrar variables sobre un control de tipo Bitmap, Grupo o Caja.

1. Edite las variables de la página de impresión.
2. Seleccione la/s variable/s y arrástrela/s sobre la página de impresión.
3. Automáticamente se obtiene la representación gráfica en la página de impresión de la variable arrastrada.

Solamente se puede asociar una variable cuyo tipo sea compatible con el del control y que y no esté ya asociada a otro control.

Arrastrar tablas o columnas de un repositorio a un control de impresión

Se permite arrastrar tablas o columnas de un repositorio sobre la página de impresión en curso o sobre uno de los controles de tipo contenedor que pueden ser padre de otros controles (un Bitmap, un Grupo, una Caja).

Para arrastrar una tabla o una columna del repositorio proceda de la siguiente manera:

1. Si el módulo en curso no tiene asociado un repositorio, edite uno de los repositorios incluidos en el proyecto, en caso contrario edite el que ya tiene asociado.
2. En la estructura en árbol de repositorio haga clic sobre una tabla o una columna.

Figura11.11. Selección de una columna en el repositorio.

3. Arrastre la tabla o columna seleccionada sobre la página de impresión o un control de los mencionados anteriormente. Al arrastrar una tabla, arrastrará todas sus columnas.

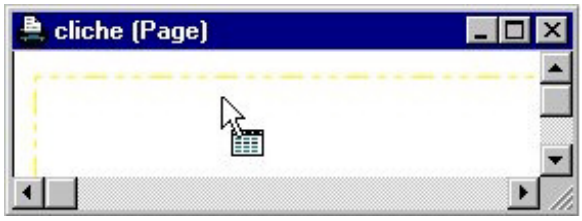


Figura11.12. Arrastrar la columna sobre la página de impresión.

4. Si se ha arrastrado sobre un control distinto de grupo de líneas, se preguntará si se desea que se creen automáticamente controles de tipo texto con la etiqueta de cada columna, si la tiene o con el nombre definido en el repositorio .

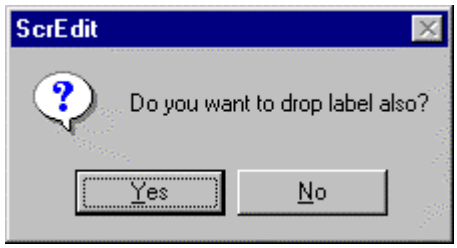


Figura11.13. Mensaje mostrado al usuario.

5. En cualquier caso se creará un control de tipo variable con el tipo de dato SQL. de la columna, máscara, longitud y alineación que tiene definida dicha columna en el repositorio.

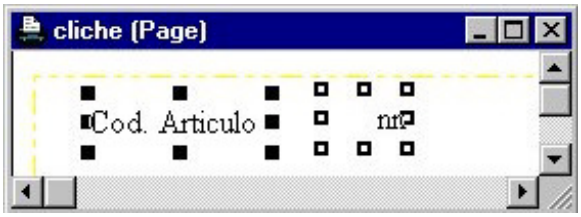


Figura11.14. Controles creados para la columna articulo de la tabla artículo.

Automáticamente se añadirán tantas variables como columnas se han arrastrado en la sección variables de la página de impresión asociadas a los controles correspondientes.

Si arrastra una tabla o una columna sobre un Grupo definido como As lines en sus propiedades, las etiquetas de las columnas serán visibles en la parte superior del Grupo como título.

Al arrastrar una columna sobre la página de impresión no se establece ningún tipo de enlace entre el control y la columna de la Base de Datos. Esto significa que:

1. La página de impresión hay que rellenarla al escribir su código.
2. El control de tipo variable que se crea para la columna, queda asociado a una variable de la página cuyo valor no se actualiza automáticamente con el contenido que tenga dicha columna en la base de datos.

Imprimir una página

La opción Print del menú Edit permite obtener por impresora el contenido de la página de impresión en curso. Al ejecutarla aparecerá un elemento de diálogo para establecer los distintos parámetros que intervienen en la impresión (orientación del papel, tamaño, impresora de salida, alimentación del papel, etc.).

Antes de proceder a la impresión de cualquier documento es recomendable guardar los cambios realizados para evitar posibles pérdidas de información ante cualquier error o problema.

El botón Options nos permitirá especificar características adicionales que dependerán del modelo de impresora elegido.

La opción Print puede ejecutarse directamente desde el menú de iconos pulsando el botón  o la tecla [Ctrl] + [P].

Tamaño de la página de impresión en edición

Es posible ampliar o reducir el tamaño de la página de impresión en el editor ejecutando la opción Zoom out/Zoom in del menú View del menú principal de la aplicación.

Su utilidad es poder ver en la ventana de edición la totalidad de la pagina, si se ejecuta Zoom out , o editar la pagina en un formato más cómodo con Zoom in . Si la página no cabe en la ventana siempre es posible moverse con las barras de desplazamiento.

Capítulo 12
Editor de Menús

Contenidos

1. Introducción

En Windows el elemento de control más común es el menú. Los menús proporcionan a los usuarios un modo sencillo de ejecutar comandos agrupados lógicamente. Casi todas las ventanas principales llevan asociadas algún tipo de menú. Debido a que los menús son tan comunes e importantes en las aplicaciones de Windows, Cosmos incorpora un editor de menús que es muy fácil de manejar.

Un menú pulldown está siempre asociado a una ventana, es fijo y aparece como una barra exactamente debajo del título de la ventana. Esta barra de menú es el menú principal o el menú de primer nivel. Un menú de tipo popup es una ventana independiente que aparece al ejecutar el método correspondiente y desaparece al seleccionar una opción Los ítems del menú de primer nivel pueden a su vez llamar a menús desplegables, que también se llaman submenús. Este editor de menús permite definir submenús anidados; lo que quiere decir que un submenú puede desplegar otro submenú. Los elementos o ítems de un menú pueden ser marcados, esto significa que se pinta una marca de verificación a la izquierda del texto del menú. La utilización de las marcas de verificación permite al usuario elegir entre las diferentes opciones del menú. Estas opciones pueden ser exclusivas, pero no tienen por qué serlo. Los elementos del primer nivel no muestran la marca a no ser que se trate de un menú popup.

La forma de desplazarnos a través de las distintas opciones (“persianas”) de este primer nivel será mediante las teclas de flechas (a derecha e izquierda), con la barra espaciadora, o bien mediante el uso del ratón como es habitual en cualquier aplicación Windows. Una vez situados en cualquier opción de este primer nivel, pulsando [flecha abajo], [flecha arriba] o [Intro], “desplegaremos” la correspondiente persiana.

Otra forma de seleccionar una opción determinada, ya sea en el primer nivel del menú o dentro de una persiana, es pulsando directamente el carácter que figure subrayado junto con [Alt]. En cualquier caso, sólo podrán seleccionarse aquellas opciones que figuren en una intensidad de vídeo normal. Las opciones que en un determinado momento no puedan ser ejecutadas aparecerán en pantalla con una intensidad de vídeo más atenuada.

Los ítems tanto del primer nivel como de cualquier menú desplegable pueden ser activados, desactivados o inhabilitados. Los ítems inhabilitados aparecen con el texto en gris.

Una vez dadas estas nociones generales, pasaremos a describir el funcionamiento y los elementos del editor de menús.

Figura 12.1. Elementos del editor de menús

Como se puede ver en la figura 12.1. los elementos del editor de menús son los que se indican a continuación:

1. Línea donde se especifican las distintas opciones principales que componen el cuerpo central del menú. Es donde se define la barra de menú.
2. Líneas donde se especifican las distintas opciones secundarias de las opciones principales del menú.
3. Líneas donde se especifican las distintas subopciones anidadas.
4. Las teclas de acceso (hot keys) permiten al usuario abrir un menú presionando la tecla [Alt] y la letra designada. Una vez abierto un menú, el usuario puede elegir una de sus opciones escribiendo la letra (tecla de acceso) asignada a esa opción. Por ejemplo [Alt] + [F] puede abrir la persiana Fichero y [G] puede seleccionar la opción Guardar de la persiana.

Con el editor de menús puede:

1. Crear los siguientes tipos de menús:

   - Menús de iconos Se caracterizan por que cada una de sus opciones lleva asociado un icono.
   - Menús pulldown Su presentación se realiza en la parte superior de la ventana donde se muestran horizontalmente las distintas opciones (también denominadas persianas).
   - Menús popup Muestran sus opciones verticalmente una debajo de otra.
2. Crear menús en cascada.
3. Crear y borrar ítems de un menú:
   - Crear y borrar opciones principales del menú.
   - Crear y borrar opciones secundarias en una opción principal del menú.
   - Crear y borrar subopciones en una opción secundaria del menú.
4. Borrar o insertar un separador entre dos subopciones.
5. Mover ítems del menú de una posición a otra. Para facilitar al usuario la ejecución de esta acción, el editor permite utilizar la técnica de arrastrar y soltar (drag & drop).
6. Asociar un icono a un ítem del menú.
7. Eliminar los iconos de un submenu.
8. Eliminar el icono de un ítem del menú.
9. Convertir un menú popup en pulldown o viceversa.

A diferencia de otras herramientas, este editor hace uso de los botones derecho e izquierdo del ratón, indicándose oportunamente aquellos casos en que es necesario utilizar el botón derecho. Si no se indica ninguno se asumirá por defecto el botón izquierdo, el más utilizado habitualmente.

Las modificaciones realizadas en este editor se guardan en el código fuente del módulo (programa, librería o include) al que pertenece el menú.

2. Utilidades del Editor de Menús

Seleccionar un ítem

Puede seleccionar un ítem de un menú para:

- Añadir un nuevo ítem (seleccionando una opción vacía para un nuevo ítem).
- Borrarlo.
- Modificar sus propiedades.
- Eliminar el icono que tiene asociado.
- Moverlo de una posición a otra del menú.

Para seleccionar un ítem haga clic sobre el ítem con el botón izquierdo del ratón. También puede utilizar las teclas de dirección para desplazarse por los ítems del menú. El ítem seleccionado se marca con un rectángulo alrededor.

La forma de desplazarnos a través de las distintas opciones (“persianas”) de primer nivel será mediante las teclas de flechas (a derecha e izquierda) o bien mediante el uso del ratón como es habitual en cualquier aplicación Windows. Una vez situados en cualquier opción de este primer nivel, pulsando [flecha abajo], [flecha arriba], “desplegaremos” la correspondiente persiana.

Crear un ítem en un menú

En este apartado se explica como crear una opción principal de un menú, una opción secundaria o una opción de un submenú.

Al editar un menú se añade automáticamente una opción vacía por cada submenú para hacer posible la inserción de opciones nuevas, estas opciones aparecen como una caja vacía con borde discontinuo y aunque aparecen al final de cada submenú pueden desplazarse a la posición deseada

Para crear un ítem:

1. Seleccione una opción vacía.
2. Escriba el nombre de ítem. Cuando comience a escribir, automáticamente cambia el foco de entrada al cuadro de diálogo Menu Item properties y el texto que escribe aparece simultáneamente en la caja del ítem y el campo Caption del cuadro de diálogo.
3. Defina el resto de las características del ítem en el cuadro de diálogo y pulse [Intro].

Borrar un ítem de un menú

El editor de menús de Cosmos permite eliminar ítems del menú en curso. Si borra un ítem que tiene asociado un menú en cascada automáticamente se borra también dicho menú.

Para borrar un ítem del menú tiene las siguientes opciones:

1. Seleccione el ítem que desea borrar y pulse [Del] ([Supr]).
2. Seleccione el ítem que desea borrar y ejecute la opción Delete del menú Edit.
3. Ejecute la opción Delete del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el ítem que desea borrar.

Asociar un icono a un ítem de un menú

Los ítems de un menú en Cosmos pueden tener asociado un icono que represente gráficamente su opción o conjunto de opciones.

Para asociar un icono a un ítem utilice la técnica de arrastrar y soltar (drag & drop):

1. Muestre la paleta de iconos.
2. Seleccione en la lista desplegable el fichero de iconos que desee utilizar. Automáticamente se muestran en el panel los iconos de dicho fichero.
3. Seleccione un icono, arrástrelo y suéltelo sobre el ítem del menú al que desee asociarlo.

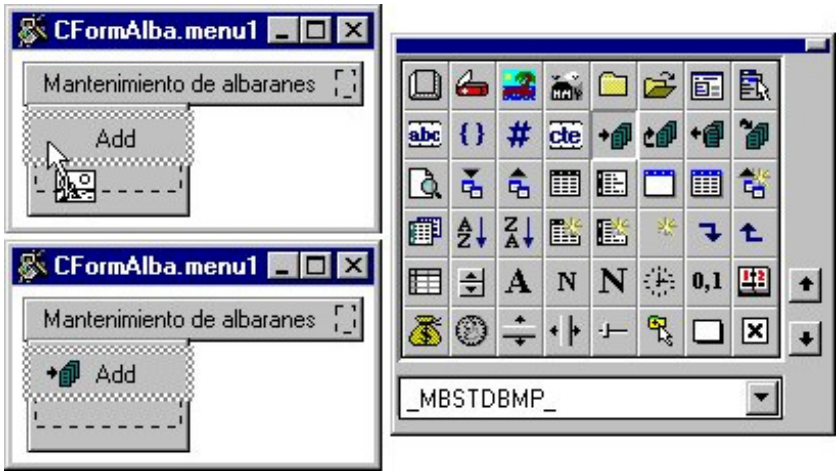


Figura 12. 2. Asociar un icono a un ítem

También se puede arrastrar el mismo icono de la paleta sobre varios ítems a la vez siguiendo las siguientes instrucciones :

1. Seleccione con el botón izquierdo del ratón el icono que desee en la paleta y arrástrelo hasta el ítem cuyo icono desee cambiar.
2. Sin soltar el botón izquierdo del ratón pulse la barra espaciadora sobre todos los ítems a los que desee cambiar el icono.

Eliminar el icono de un ítem

Para eliminar el icono que tiene asociado un ítem del menú:

1. Seleccione el ítem.
2. Edite las propiedades del ítem.
3. En el cuadro de diálogo Menu Item properties haga clic con el botón izquierdo del ratón sobre la casilla de verificación Delete icon.
4. Automáticamente queda eliminado el icono de dicho ítem.

En un ítem de tipo Popup se puede eliminar todos los iconos asociados a su submenú. No será necesario eliminar el icono ítem a ítem.

Para eliminar todos los iconos del Popup asociado a un ítem:

1. Seleccione el ítem.
2. Edite las propiedades del ítem.
3. En el cuadro de diálogo Menu Item properties haga clic con el botón izquierdo del ratón sobre la casilla de verificación Delete icons.
4. Automáticamente quedan eliminados todos los iconos del Popup.

Eliminar los iconos de un submenu

El editor de menús permite eliminar los iconos asociados a las opciones de un submenu. No será necesario eliminar el icono opción por opción:

Para eliminar todos los iconos del Popup asociado a un ítem:

1. Edite las propiedades del menú.
2. En el cuadro de diálogo Menu Item properties haga clic con el botón izquierdo del ratón sobre la casilla de verificación Delete icons.
3. Automáticamente quedan eliminados todos los iconos de las opciones principales.

Mover ítems de un menú

El editor de menús permite mover ítems de un menú de una posición a otra utilizando la técnica de arrastrar y soltar (drag & drop).

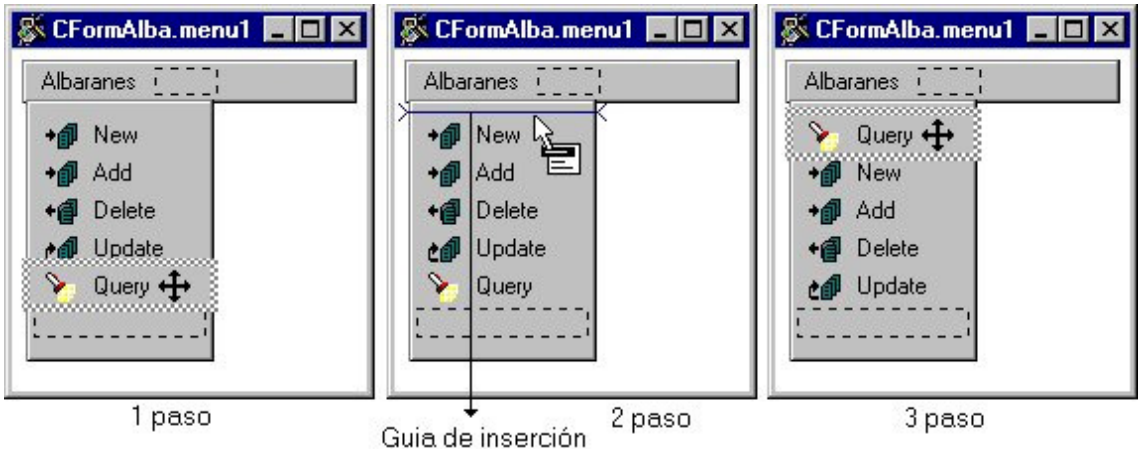


Figura 12.3. Mover un ítem de un menú

En la figura 12.3 se muestran los pasos que se deben seguir para mover un ítem:

1. Haga clic sobre el ítem que desee mover y arrástrelo a una nueva posición
2. Suelte el ítem cuando la guía de inserción este en la posición deseada.

Crear un menú en cascada

El editor de menús permite crear menús en cascada.

Para crear un menú en cascada:

1. Seleccione el ítem del menú al que desea añadir un menú en cascada.
2. Edite las propiedades del ítem.
3. En el cuadro de diálogo Menu Item properties marque la casilla de verificación Popup.
4. Automáticamente aparece en el menú una nueva caja vacía a la derecha del ítem en curso para crear el submenú.
5. Seleccione la caja vacía para el nuevo ítem y añada las opciones del submenu creando los ítems correspondientes.

Si desea borrar el menú en cascada asociado a un ítem, edite sus propiedades y quite la marca a la casilla de verificación Popup haciendo clic sobre ella.

Convertir un menú popup en pulldown o viceversa

El editor de menús permite crear menús de tipo popup y de tipo pulldown.

Se puede convertir un menú de tipo popup en pulldown y viceversa ejecutando las siguientes instrucciones:

- **Primer método:**
  1. Edite las propiedades del menú.
  2. Haga clic sobre la casilla de verificación Show as pulldown para modificar el tipo del menú.

- **Segundo método:**
  1. Seleccione la barra de menú (donde están las opciones principales del mismo) haciendo clic sobre ella o sobre cualquier parte de la ventana de edición).
  2. Ejecute el comando As pulldown del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el menú.

3. Definición de Propiedades

Editar las propiedades del menú

Puede editar las propiedades de un menú para realizar las siguientes acciones:

1. Eliminar el icono de las opciones principales del menú.
2. Convertir un menú popup en pulldown o viceversa.
3. Modificar la etiqueta del menú, etc.

Para editar las propiedades de un menú tiene las siguientes opciones:

1. Seleccione la barra de menú (donde están las opciones principales del mismo) haciendo clic sobre ella o en cualquier parte de la ventana de edición) y pulse [Intro].
2. Seleccione la barra de menú y ejecute la opciópn Properties del menú Edit.
3. Ejecute la opción Properties del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre la barra del menú o en cualquier parte de la ventana de edición.
4. Haga doble clic sobre la barra del menú o en cualquier parte de la ventana de edición.

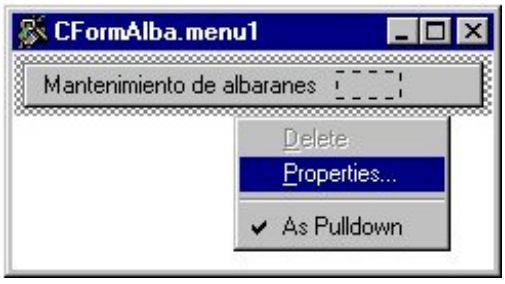


Figura 12.4. Editar las propiedades de un menu

5. Se muestra el cuadro de diálogo Menu Item properties que permite editar sus propiedades para modificarlas.

Todos los menús y todos los ítems que van a ser referenciados posteriormente en el código fuente del programa deben tener definido el identificador (Id) en sus propiedades.

Propiedades del menú

En este cuadro de diálogo se definen las características principales de los menús.

Este cuadro de diálogo tiene los siguientes elementos:

Elementos	Significado
Id	Nombre identificador único del menú.
Show as pulldown	Permite definir el menú de tipo popup o pulldown.
Disabled	Habilita o inhabilita todas las opciones del menú.
Allow icons	Muestra u oculta los iconos asociados a las opciones principales de un menú popup.
Allow check	Muestra u oculta las marcas de las opciones principales de un menú popup
Delete icons	Borra los iconos asociados a las opciones principales de un menú popup.

Editar las propiedades de un ítem

Puede editar las propiedades de un ítem para:

1. Modificar su identificador.
2. Modificar su nombre.
3. Eliminar el icono que tiene asociado.
4. Crear o borrar un menú en cascada.
5. Definir el comando que se ejecutará al pulsar el ítem, etc

Para editar las propiedades de un ítem tiene las siguientes opciones:

1. Seleccione el ítem que desee editar y pulse [intro].
2. Seleccione el ítem que desee editar y ejecute la opción Properties del menú Edit.
3. Ejecute la opción Properties del menú tipo popup que aparece al hacer clic con el botón derecho del ratón sobre el ítem que desea editar.
4. Haga doble clic sobre el ítem que desea editar.

Figura 12.5. Editar las propiedades de un ítem

Se muestra el cuadro de diálogo Menu Item properties que permite editar sus propiedades para modificarlas

Propiedades de los ítems

En este cuadro de diálogo se definen las características principales de los ítems del menú..

Este cuadro de diálogo tiene los siguientes elementos:

Elementos	Significado
Id	Nombre identificador único del ítem.
Caption	Título o literal descriptivo del ítem. En este punto es donde se puede añadir el carácter & para subrayar la letra que se desea utilizar como tecla de acceso (hot key) . Las teclas de acceso (hot keys) permiten al usuario abrir un menú presionando la tecla [Alt] y la letra designada. Una vez abierto un menú, el usuario puede elegir una de sus opciones escribiendo la letra (tecla de acceso) asignada a esa opción. Por ejemplo [Alt] + [F] puede abrir la persiana Fichero y [G] puede seleccionar la opción Guardar de la persiana.
Separator	Permite añadir o eliminar una línea horizontal separadora entre ítems del menú. Se suele utilizar para separar grupos de opciones relacionadas.
Disabled	Habilita o inhabilita el ítem y sus subopciones.
Checked	Añade o quita la marca al ítem del menú. Esta marca indica que la opción del menú esta chequeada. Esta tipo de símbolos se utiliza cuando deseamos que una cierta operación este siempre sujeta, o no, a un comportamiento estándar. Por ejemplo habilitar que las ventanas aparezcan siempre en mosaico.
Popup	Permite definir un submenú o conjunto de subopciones anidadas para el ítem.
Delete Icon	Este campo solo esta habilitado cuando el ítem tiene un icono asociado. Haga clic sobre esta casilla de verificación para eliminar el icono asociado al ítem.
Allow icons	Muestra u oculta los iconos asociados al Popup (submenú o subopciones anidadas) del ítem en curso.
Allow check	Muestra u oculta las marcas de verificación. del Popup (submenú o subopciones anidadas) del ítem en curso.
Delete icons	Borra los iconos asociados al Popup del ítem en curso. Este campo solo esta habilitado cuando alguna opción del Popup del ítem tiene un icono asociado.
Command	Nombre del comando que se desea ejecutar al pulsar la opción de menú. Por software se indicarán las acciones a realizar con dicho comando. Los comandos simplifican el código de los programas, ya que permiten definir un conjunto de acciones que se pueden ejecutar con cualquier botón que se tenga en la screen o con cualquier opción de un menú. Este campo solo esta visible cuando no se define un Popup para el ítem.
Comments	Texto que se mostrará en el panel de comentarios de la barra de estado de la screen, en caso de existir, al tener el cursor sobre el ítem.

Capítulo 13
El Editor de Iconos

Contenidos

1. Introducción

El Editor de Iconos permite añadir, borrar, modificar y editar iconos de un fichero de iconos. Estos ficheros podrán ser usados como paletas dentro del Editor Visual. Se pueden arrastrar iconos sobre las opciones de un menú y sobre ciertos tipos de controles de un formato de pantalla (screen de un Form) y de una página de impresión, por ejemplo para construir menús de botones.

El Editor de Iconos permite crear dibujos sencillos o elaborados a color. A diferencia de otras herramientas, este Editor hace uso de los botones derecho e izquierdo del ratón, indicándose oportunamente aquellos casos en que es necesario utilizar el botón derecho. Si no se indica ninguno se asumirá por defecto el botón izquierdo, el más utilizado habitualmente.

El editor de ficheros de iconos se invoca de dos formar:

- por medio del comando cosicons, cuya sintaxis es la siguiente:

cosicons [nombre_fichero]

Opcion	Significado
nombre_fichero	Nombre lógico del fichero de iconos. Es el identificador del fichero de iconos en las herramientas de desarrollo. Este fichero deberá ser alguno de los existentes en la sección *[Icons]* del fichero de configuración COSMOS.INI

- Haciendo doble click en el icono:

2. Aspecto general del Administrador de Iconos

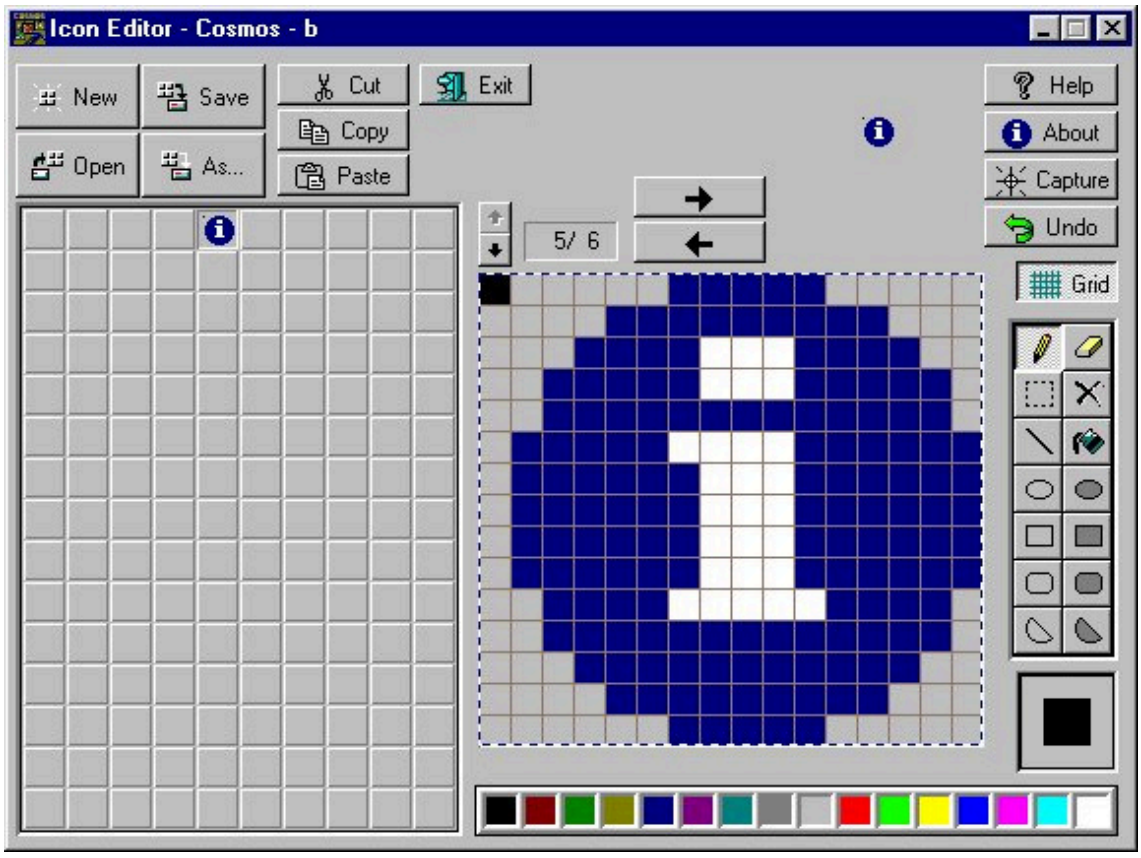


Figura 13.1. Aspecto del Editor de Iconos

La ventana del Editor de Iconos contiene los siguientes elementos:

Área de edición	Es el espacio donde se crean los iconos.
Panel de iconos	Situado a la izquierda de la ventana principal de la aplicación, es donde se muestran el contenido del el fichero de iconos con el que se está trabajando.
Cursor	Indica el lugar donde aparecerán los objetos al comenzar a dibujar o el elemento elegido fuera del área de edición. Dentro de esta área la forma del cursor cambiará según la herramienta que se utilice, mientras que fuera de dicha área aparecerá siempre en forma de flecha.
Caja de herramientas	Contiene las herramientas que se pueden utilizar para crear los iconos. Estas herramientas permiten pintar, llenar, borrar y hacer correcciones en el área de edición.
Paleta de colores	Contiene los colores que se pueden utilizar con las herramientas de dibujo.

3. Manejo de ficheros de Iconos

Un fichero de iconos es un fichero de mapas de bits (con extensión *.bmp*) que contiene un conjunto de iconos que pueden ser su utilizados en la generación de aplicaciones de Cosmos. Estos ficheros podrán ser usados como paletas dentro del Editor visual.

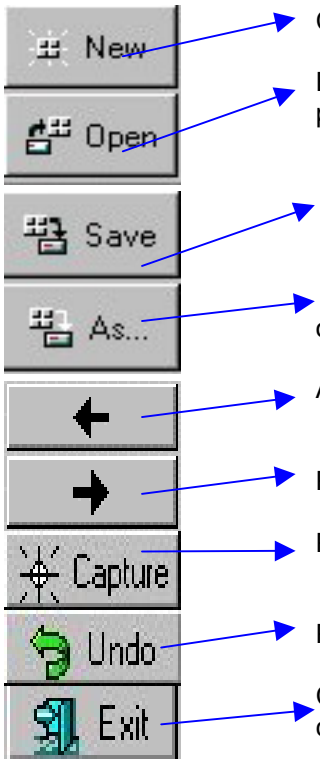
Estos ficheros deben estar definidos en la sección *Icons* del fichero de configuración COSMOS.INI.

La sintaxis de un elemento de esta sección es la siguiente:

NombreLógico=nombreFichero.bmp, anchoBotón, altoBotón, colorFondo

Campos	Significado
NombreLógico	Identificador del fichero de iconos en las herramientas de desarrollo
nombreFichero.bmp	Puede contener el path completo o relativo al subdirectorio etc del directorio donde se encuentra instalado Cosmos
anchoBotón	Ancho en pixeles de cada icono
altoBotón	Alto en pixeles de cada icono
colorFondo	Puede ser RGB(rojo, verde, azul) o uno de los siguientes: LTGRAY, WHITE, GRAY, DKGRAY, BLACK, RED, YELLOW, GREEN y BLUE

Para acceder a los ficheros de Iconos utilice el grupo de opciones de la esquina superior izquierda de la ventana principal.



Crea un fichero de iconos vacío.

Esta opción permite cargar un fichero de iconos existente, previamente definido en el fichero de configuración COSMOS.INI.

Guarda el contenido del fichero de iconos en curso.

Permite salvar el fichero de iconos en curso con otro nombre o en otro directorio.

Añadir iconos a un fichero de iconos

Para llevar un icono al área de edición

Para capturar una imagen

Para deshacer cambios

Cierra la ventana principal y retorna al sistema operativo o al comando desde el que fue invocado el entorno.

Crear un Fichero

Esta opción permite crear un fichero de iconos vacío. Esta opción se ejecuta pulsando el botón 

Al ejecutarla se mostrará el cuadro de diálogo *New Button File* que tiene los siguientes campos:

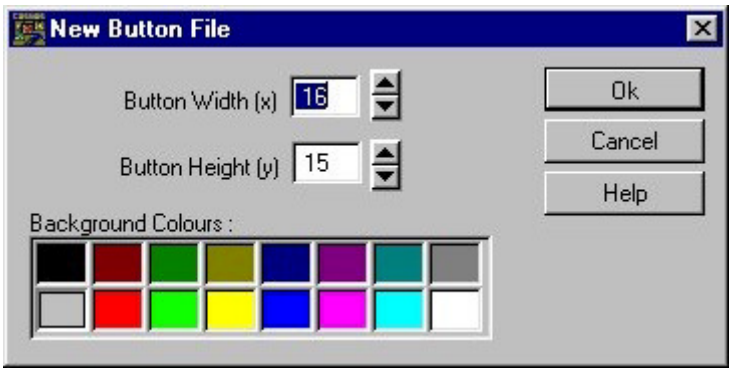


Figura 13.2. Cuadro de diálogo New Button File

Button Width Campos de edición numérico en el que se indicará el ancho en pixeles de cada icono.

Button Height Campos de edición numérico en el que se indicará el alto en pixeles de cada icono.

Background Colours Permite seleccionar el color de fondo de los iconos. Para ello bastará con hacer click sobre el color deseado.

El fichero se crea con espacio para un botón. Si se necesita más espacio éste se irá asignando automáticamente.

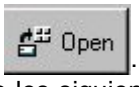
Si desea optimizar el espacio que se necesita para salvar el fichero, cree los iconos secuencialmente.

El nombre y la ubicación de fichero de iconos se definirá al guardarlo por primera vez.

El color de fondo de un fichero de iconos será sustituido por el color de fondo del botón durante la ejecución de los programas. Por lo tanto, es muy importante poner especial cuidado en su elección. Habitualmente, se usará el gris claro para botones de diálogo y el blanco en los controles tipo lista y en impresora.

Una vez creado el fichero de iconos no se podrá cambiar el tamaño de los elementos ni el color de fondo. No obstante, sí podrá hacerse desde el Editor editando la línea correspondiente del fichero de configuración COSMOS.INI, aunque esta acción puede suponer una pérdida de información en el fichero de iconos.

Abrir un fichero

Esta opción permite abrir un fichero de iconos existente previamente definido en el fichero de configuración COSMOS.INI. Esta opción se ejecuta pulsando el botón 

.

Al ejecutarla se mostrará un cuadro de diálogo *Load Button File* que tiene los siguientes campos:

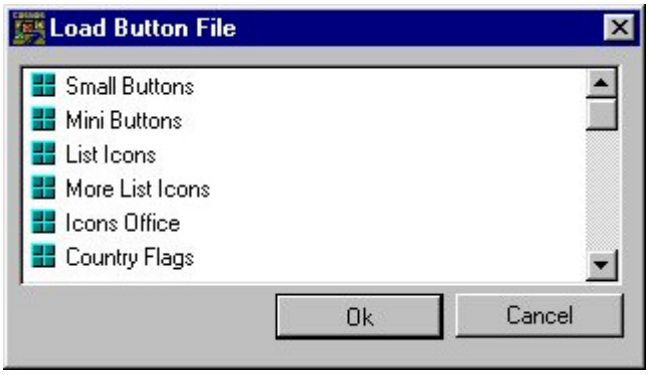


Figura 13. 3. Cuadro de diálogo Load Button File

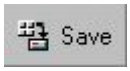
Files Muestra los ficheros de iconos existentes para elegir aquel que se desea abrir. Las teclas *[FLECHA ARRIBA]* y *[FLECHA ABAJO]* nos permitirán movernos a lo largo de la lista. Asimismo las teclas *[INICIO]* y *[FIN]* nos permitirán desplazarnos respectivamente al primero y al último elemento de la lista. Para abrir un fichero bastará con hacer doble click sobre su nombre.

Ok Abre el fichero de iconos seleccionado en la lista anterior.

Cancel Retorna a la ventana principal sin realizar ninguna acción.

Una vez abierto el fichero de iconos su contenido se muestra en el panel de la esquina inferior izquierda de la ventana principal. Si la totalidad de los iconos no cabe en la ventana podrán verse los restantes pulsando las teclas *[AV.PÁG.]* y *[RE.PÁG.]* o los botones  y  .

Salvar un fichero

Guarda el contenido del fichero de iconos en curso. Esta opción se ejecuta pulsando el botón 

Salvar un fichero con otro nombre

La opción *Guardar como...* permite salvar tanto un fichero de iconos de nueva creación como otro ya existente asignándole otro nombre. Esta opción se ejecuta pulsando el botón 

.

Al ejecutarla se mostrará el cuadro de diálogo *Save Button File As ...* que tiene los siguientes campos:

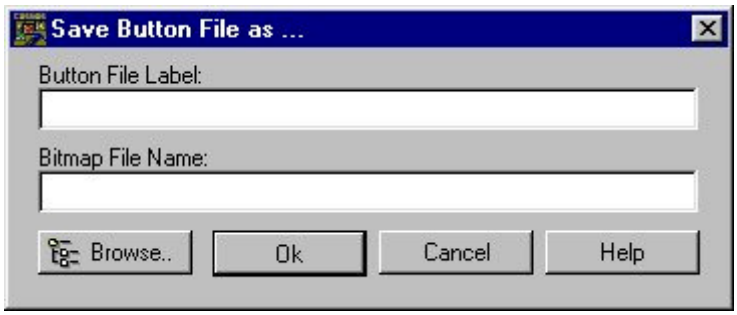


Figura 13.4. Cuadro de diálogo Save Button File as...

Button File Label	Nombre con el que se desea identificar al fichero de iconos en programas de desarrollo.
Bitmap File Name	Campo de edición en el que se indicará la ubicación física del fichero (con extensión *.bmp*).
Browse..	Muestra un cuadro de diálogo para seleccionar el nombre del fichero, el directorio y la unidad de disco.

Si el nombre asignado al fichero coincidiese con el de algún otro ya existente, el sistema avisaría al usuario presentando un mensaje en pantalla.

Con esta opción el usuario podrá elegir entre:

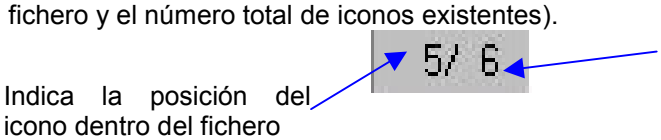
- Guardar el fichero de iconos con el mismo nombre (*Sobre-escribir*), con lo cual se grabarían las modificaciones que hubieran podido realizarse sobre él (equivalente a la opción *Guardar*). En este caso el sistema pediría confirmación avisando de que se trata de un fichero existente que se va a sobreescribir.
- Asignar un nombre distinto al actual, con lo cual estaríamos creando un nuevo fichero de iconos.
- Cancelar la operación.

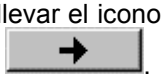
El tamaño de los iconos y el color de fondo no se pueden cambiar desde esta opción. El fichero de iconos se salva con 16 colores.

Modificar un fichero de iconos

Para modificar un icono del fichero de iconos proceda de la siguiente manera:

1. Abra el fichero de iconos que contenga el icono a modificar.
2. Seleccione con el ratón en el panel de iconos de la izquierda el que desee modificar (encima del área de edición se mostrará la posición del icono elegido dentro del fichero y el número total de iconos existentes).



3. Para llevar el icono seleccionado al área de edición arrástrelo con el ratón o pulse el botón  .
4. Realice las modificaciones necesarias valiéndose de la caja de herramientas y de la paleta de colores.
5. Una vez efectuadas las modificaciones, pulse el botón  para reintegrarlo a su posición en el panel.
6. Salve el fichero de iconos.

Añadir o reemplazar un icono del fichero de iconos

Para añadir un icono al fichero de iconos proceda de la siguiente manera:

1. Abra el fichero de iconos en el que desee añadir el icono.
2. Dibuje el icono utilizando la caja de herramientas y la paleta de colores del Editor.
3. A continuación, seleccione una posición vacía en el panel de iconos de la izquierda y pulse el botón  .
4. El nuevo icono aparecerá en el panel de iconos en la posición elegida.
5. Guarde el fichero de iconos para salvar los cambios.

Si el icono recién generado se añade al panel dejando casillas vacías respecto al último icono existente, el tamaño del fichero de iconos aumentará innecesariamente. Por esta razón es recomendable añadir el icono en la casilla vacía inmediata al último icono del panel. No obstante, en el caso de que por error hubiese dejado iconos vacíos, podrá restaurar el fichero a su tamaño original mediante la opción de truncado (pulsando *[Ctrl+T]*).

Capturar imágenes

El Editor de Iconos de Cosmos permite capturar directamente una zona de la pantalla para incluirla en el área de edición. La imagen a capturar podrá ser tanto del propio Editor de Iconos como de cualquier otro programa.

Para capturar una imagen proceda de acuerdo a los siguientes pasos:

1. Configure su pantalla de forma que pueda ver simultáneamente el botón 

del Editor de Iconos y la zona donde se encuentra la imagen que desea incluir en el área de edición. Si la imagen pertenece a otro programa deberá tener abiertos a la vez el Editor de Iconos y el programa en cuestión.
2. Pulse el botón 

.
3. Mueva el ratón hasta situar el cursor sobre la imagen que desea capturar. En todo momento, en el área de edición se irá mostrando aumentada la zona de la pantalla por la que se esté desplazando. Si desea obtener mayor precisión en el control del cursor utilice las teclas de flechas para desplazarlo.
4. Cuando en el área de edición aparezca el elemento que desea incluir, podrá utilizar el botón izquierdo del ratón, la barra espaciadora o la tecla [*RETURN]* para que la imagen quede capturada.

Para cancelar la captura de una imagen pulse la tecla *[ESC].*

Mientras esté capturando una imagen podrá modificar su tamaño mediante las siguientes teclas:

Teclas	Efecto
Página arriba	Aumenta el ancho y el alto del área capturada
Página abajo	Disminuye el ancho y el alto del área capturada
Mayúsculas + flecha izquierda	Disminuye el ancho del área capturada
Mayúsculas + flecha derecha	Aumenta el ancho del área capturada
Mayúsculas + flecha arriba	Aumenta la altura del área capturada
Mayúsculas + flecha abajo	Disminuye la altura del área capturada

Deshacer cambios

Si comete algún error mientras está dibujando puede utilizar el botón 

o la combinación de teclas *[CTRL]+[U]* para deshacer la última acción realizada.

El botón 

eliminará todas las acciones realizadas con el pincel, el borrador o la herramienta de relleno desde el momento en que se seleccionó cualquiera de ellas, mientras que con las restantes herramientas sólo afectará a la última acción.

4. Caja de herramientas

La caja de herramientas del Editor de Iconos se encuentra situada verticalmente a lo largo del borde derecho de la ventana, y su finalidad es facilitar la edición de iconos, para lo cual incluye los siguientes elementos:

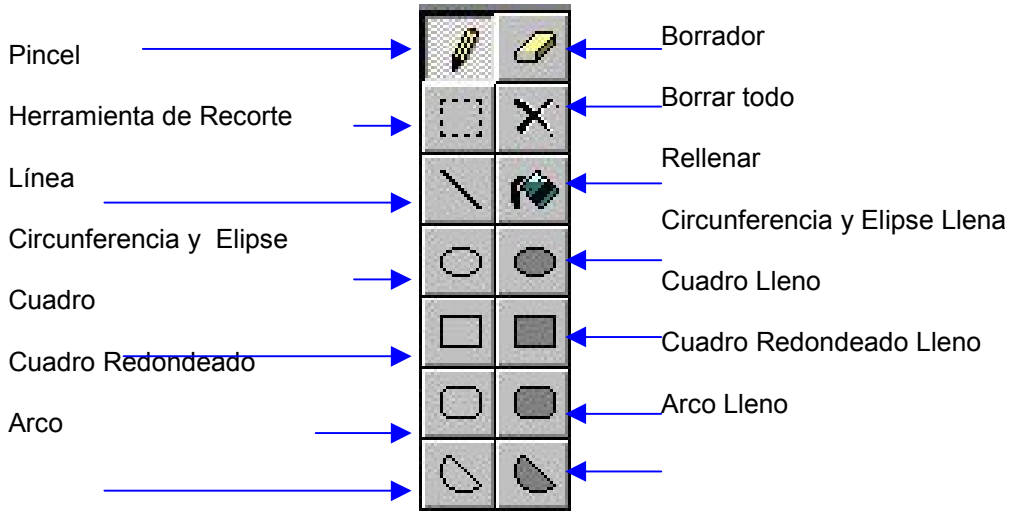


Figura 13.5. Aspecto de la Caja de Herramientas

Elemento	Acción
Pincel	Permite dibujar punto a punto líneas y formas de diseño libre
Borrador	Simula una goma de borrar para hacer las correcciones necesarias
Herramienta de recorte	Permite seleccionar un área del dibujo para realizar con ella ciertas operaciones (borrar, cortar, copiar)
Borrar todo	Limpia la totalidad del área de dibujo para empezar de nuevo
Línea	Permite dibujar líneas rectas
Rellenar	Permite aplicar un color de relleno a cualquier área
Circunferencia y Elipse	Permite dibujar circunferencias y elipses
Circunferencia y Elipse llena	Permite dibujar circunferencias y elipses rellenas del color que se tenga seleccionado
Cuadro y cuadro redondeado	Permite dibujar cuadrados con los vértices en ángulo recto o redondeados
Cuadro lleno y cuadro lleno redondeado	Igual a la anterior pero aplicando como relleno el color que se tenga seleccionado
Arco	Permite dibujar un arco
Arco lleno	Igual a la anterior pero aplicando color de relleno

.

Para seleccionar cualquiera de las herramientas anteriores bastará con hacer un click sobre ella. Alternativamente podrá emplearse también el tabulador hasta activar la caja de herramientas y emplear las teclas de movimiento del cursor para seleccionar una de ellas, pulsando a continuación la barra espaciadora para activarla. La herramienta elegida aparecerá resaltada.

Excepto en aquellos casos en que se encuentre capturando una imagen, el botón derecho del ratón podrá emplearse para borrar un pixel aplicando el color de fondo seleccionado.

Herramientas para pintar

- *Pincel*

Permite dibujar formas de diseño libre y líneas punto a punto con el color seleccionado.

Para pintar con el pincel:

7. Seleccione el color de dibujo que desee utilizar en la paleta de colores.
8. Seleccione en la caja de herramientas el icono  .
9. Sitúe el cursor en el área de edición.
10. Utilice el botón izquierdo del ratón para pintar el pixel sobre el que se encuentre situado el cursor y manténgalo presionado si desea dibujar puntos adyacentes.
11. Suelte el botón del ratón para dejar de dibujar.

- *Rellenar*

Esta herramienta permite rellenar con el color seleccionado cualquier área cerrada o bien la totalidad del área de edición, pudiéndose aplicar también sobre cualquier objeto dibujado (línea, pixel, etc.) para cambiar su color.

Para aplicar un color de relleno:

1. Seleccione con el botón izquierdo del ratón el color de primer plano que desee utilizar en la paleta de colores.
2. Seleccione en la caja de herramientas el icono  .
3. Lleve el cursor al área de edición.
4. Coloque la cruz del cursor sobre del área que desee rellenar y haga click con el botón izquierdo del ratón. El área se pintará con el color de dibujo seleccionado.
5. Si el área sobre la que se aplica no estuviese completamente cerrada, el color se filtrará al área adyacente, rellenándose una zona mayor de la deseada. Si sucediese esto, pulse el botón 

para limpiar el área y empezar de nuevo (defina entonces exactamente el área que desea rellenar).

Si aplica la herramienta de relleno a una línea que estuviese en contacto con otra, o a un pixel, ambos elementos modificarán su color.

Herramientas de borrado

- *Goma de Borrar*

Esta herramienta permite borrar pixeles para corregir un dibujo. Al aplicar el borrador sobre un pixel éste se mostrará en el color de fondo que se tenga seleccionado para el icono.

Para utilizar el borrador:

1. Cerciórese de que el color de fondo seleccionado coincide con el color de fondo del área que desea borrar. Si los colores no coinciden se aplicará el color de fondo elegido en lugar de borrar
2. Seleccione en la caja de herramientas el icono  .
3. Lleve el cursor al área de edición.
4. Pulse el botón izquierdo del ratón si desea borrar toda el área ocupada por el borrador o bien el botón derecho si desea eliminar solamente el pixel situado en la punta del borrador.
5. Con el botón pulsado arrastre el cursor hasta borrar el área deseada.

- *Borrar todo*

Esta herramienta permite limpiar la totalidad del área de edición. Antes de proceder al borrado se pedirá conformidad.

Para borrar el área de edición seleccione en la caja de herramientas el icono  .

Herramientas de dibujo

- *Línea*

Permite dibujar líneas rectas (horizontales, verticales o diagonales) con el color seleccionado. El ancho de línea es siempre de un pixel.

Para pintar una línea:

1. Seleccione el color de dibujo que desee utilizar en la paleta de colores.
2. Seleccione en la caja de herramientas el icono  .
3. Lleve el cursor al área de edición.
4. Presione el botón izquierdo del ratón para fijar uno de los extremos de la línea y arrastre el cursor hasta el punto en que desee finalizar la línea.
5. Cuando la línea tenga la longitud deseada suelte el botón del ratón.
6. Si no está satisfecho con el resultado, pulse el botón  para borrar la línea y empezar de nuevo.

- *Circunferencias y Elipses*

Esta herramienta permite dibujar circunferencias y elipses.

Para dibujar cualquiera de estos elementos:

1. Seleccione con el botón izquierdo del ratón el color que desee utilizar en la paleta de colores.
2. Seleccione en la caja de herramientas el icono  .
3. Lleve el cursor al área de edición.
4. Presione el botón izquierdo del ratón en el punto donde desee que comience la circunferencia o elipse y arrástrelo hasta el punto en el que desee finalizar.
5. Cuando la elipse/circunferencia tenga el tamaño deseado, suelte el botón del ratón.
6. Si no está satisfecho con el resultado, pulse el botón  para borrar la elipse y empezar de nuevo.

- *Circunferencias y Elipses con color de relleno*

Esta herramienta permite dibujar directamente círculos o elipses con un color de relleno.

Para ello, proceda de acuerdo a los siguientes pasos:

1. Seleccione en la paleta de colores el color que desee emplear como relleno.
2. Seleccione en la caja de herramientas el icono  .
3. Lleve el cursor al área de edición.
4. Presione el botón izquierdo del ratón en el punto donde desee que comience el círculo o elipse y arrástrelo hasta el punto en el que desee finalizar.
5. Cuando la elipse o el círculo tenga el tamaño deseado, suelte el botón del ratón.
6. Si no está satisfecho con el resultado, pulse el botón  para borrar el dibujo y empezar de nuevo.

- *Cuadro y Cuadro con esquinas redondeadas*

Estas herramientas permiten dibujar cuadrados con esquinas en ángulo recto o redondeadas, respectivamente.

Para ello:

1. Seleccione en primer lugar el color que desee utilizar en la paleta de colores.
2. Seleccione en la caja de herramientas el icono  para pintar un cuadrado o rectángulo o el icono  si desea un cuadrado con ángulos redondeados.
3. Lleve el cursor al área de edición.
4. Presione el botón izquierdo del ratón en el punto donde desee que comience el cuadrado y arrástrelo hasta el punto en el que desee finalizar.
5. Cuando el cuadrado tenga el tamaño deseado, suelte el botón del ratón.
6. Si no está satisfecho con el resultado, pulse el botón 

para borrar el cuadrado y empezar de nuevo.

- *Cuadro y Cuadro con esquinas redondeadas con color de relleno*

Estas herramientas permiten dibujar cuadrados con esquinas en ángulo recto o redondeadas, respectivamente, aplicando además un color de relleno.

Para ello:

1. Seleccione en la paleta de colores el color de primer plano que desee utilizar.
2. Seleccione en la caja de herramientas el icono  para pintar un cuadrado lleno o el icono  para pintar un cuadrado lleno con los ángulos redondeados.
3. Lleve el cursor al área de edición.
4. Presione el botón izquierdo del ratón en el punto donde desee que comience el cuadrado y arrástrelo hasta el punto en el que desee finalizar.
5. Cuando el cuadrado tenga el tamaño deseado, suelte el botón del ratón.
6. Si no está satisfecho con el resultado, pulse el botón 

para borrar el cuadrado y empezar de nuevo.

- *Arco*

Esta herramienta permite dibujar un arco cerrado.

Para trazar un arco:

1. Seleccione el color que desee utilizar en la paleta de colores.
2. Seleccione en la caja de herramientas el icono  .
3. Lleve el cursor al área de edición.
4. Presione el botón izquierdo del ratón en el punto donde desee que comience el arco y arrástrelo hasta el punto en el que desee finalizar.
5. Cuando el arco tenga el tamaño deseado, suelte el botón del ratón.
6. Si no está satisfecho con el resultado, pulse el botón 

para borrar el arco y empezar de nuevo.

Tenga en cuenta que la orientación del arco variará según lo defina de arriba a abajo o de izquierda a derecha y viceversa.

- *Arco con color de relleno*

Esta herramienta permite dibujar un arco aplicándole un color de relleno.

Para trazar un arco con color de relleno:

1. Seleccione el color de relleno que desee utilizar en la paleta de colores.
2. Seleccione en la caja de herramientas el icono  .
3. Lleve el cursor al área de edición.
4. Presione el botón izquierdo del ratón en el punto donde desee que comience el arco y arrástrelo hasta el punto en el que desee finalizar.
5. Cuando el arco tenga el tamaño deseado, suelte el botón del ratón.
6. Si no está satisfecho con el resultado, pulse el botón 

para borrar el arco y empezar de nuevo.

Herramienta para definir un recorte

Esta herramienta permite definir un área de recorte rectangular en la zona de edición para permitir su almacenamiento en el portapapeles a través de las opciones *Cortar* o *Copiar*. Asimismo, se podrá emplear para borrar un área determinada mediante la tecla *[Supr]*.

El área de recorte podrá ser desde un solo pixel hasta la totalidad del área de edición.

Para seleccionar un área de recorte:

1. Seleccione en la caja de herramientas el icono  .
2. Sitúe el cursor en el punto donde desee que comience el área a recortar.
3. Pulse el botón izquierdo del ratón y arrástrelo hasta seleccionar el área deseada.
4. Una vez definida el área de recorte suelte el botón del ratón.
5. Si no estuviera satisfecho con el resultado obtenido, haga click con el botón izquierdo del ratón para empezar de nuevo.
6. Una vez definida el área de recorte podrá cortarla, copiarla o borrarla.

5. Paleta de Colores

Al utilizar el Editor de Iconos conviene distinguir entre el color de primer plano (color de dibujo) y el color de fondo.

El color de fondo de un fichero de iconos se establece al crear un nuevo fichero o bien modificando la sección *Icons* del fichero de configuración COSMOS.INI. No obstante, cuando se esté trabajando con un fichero existente se podrá modificar el color de fondo seleccionándolo en la paleta de colores con el botón derecho del ratón. El nuevo color elegido se aplicará cada vez que se pulse dicho botón sobre el área de edición o cuando se emplee el borrador.

Por su parte, el color de primer plano o de dibujo se selecciona con el botón izquierdo del ratón y se aplica a las herramientas de dibujo, relleno, línea, etc.

La asignación de los colores de fondo y primer plano a los botones derecho e izquierdo del ratón permite dibujar los iconos de una forma rápida y cómoda.

En la parte derecha de la ventana principal, debajo de la caja de herramientas, se muestran en todo momento los colores de fondo y primer plano seleccionados.

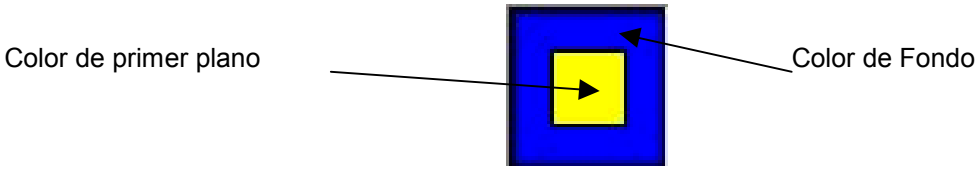
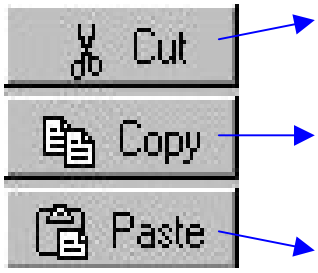


Figura 13.6. Color de fondo y de primer plano seleccionados

6. Manejo de Iconos a través del Portapapeles

El Editor de Iconos dispone de un conjunto de utilidades de recorte que permiten traspasar la totalidad o una parte de un icono a otro fichero o, incluso, a otra aplicación.

Estas utilidades son:



Transfiere al portapapeles de Windows el recorte definido eliminándolo del área de edición

Transfiere al Portapapeles de Windows el área de recorte definida.

Transfiere un mapa de bits del Portapapeles de Windows al área deedición

Copiar un Recorte

Esta herramienta permite transferir al Portapapeles de Windows el área de recorte definida.

Para copiar un recorte al Portapapeles:

1. Defina un recorte en el área de edición mediante el icono  de la caja de herramientas.
2. Pulse el botón 

o la combinación de teclas [CTRL]+[C].

Pegar un Recorte

Esta herramienta permite transferir un mapa de bits desde el Portapapeles de Windows al área de edición.

Hay que tener en cuenta que el Editor de Iconos no permite pegar un recorte cuyo tamaño sea mayor que el del área de edición. Si se intenta sólo se mostrará la parte del recorte que quepa en dicha área. Esta limitación no afectará cuando el contenido del Portapapeles se incluya en otra aplicación.

Para pegar un recorte desde el Portapapeles:

1. Pulse el botón 

o la combinación de teclas *[CTRL]+[V].*
2. Desplace el recorte en el área de edición valiéndose del ratón o de las teclas de control de movimiento del cursor.
3. Cuando el recorte esté en el lugar deseado pulse *[RETURN*].

Cortar y Mover un Recorte

Esta herramienta permite transferir al Portapapeles de Windows el recorte definido borrándolo del área de edición.

Para cortar una parte o la totalidad del área de edición:

1. Defina un recorte en el área de edición mediante el icono  de la caja de herramientas.
2. Pulse el botón 

o la combinación de teclas *[CTRL]+[X].*

El recorte definido se encontrará disponible para su inclusión en otro fichero o en otra aplicación mediante la opción pegar.

Para mover un recorte dentro del área de edición utilice la opción cortar seguida de pegar y desplácelo al lugar deseado mediante el ratón o las teclas de flechas. A continuación, pulse el botón izquierdo del ratón o *[RETURN]* para fijarlo en su nueva ubicación.

7. Otras Utilidades

Además de las opciones propias referidas a la edición, el Editor de Iconos de Cosmos dispone de ciertas utilidades que permiten repintar la pantalla, reducir el tamaño del fichero de iconos y manejar la cuadrícula del área de dibujo.

Estas utilidades son las siguientes:

Utilidades	Significado
Truncar	Para reducir el tamaño de un fichero de iconos, seleccione el último icono que desee conservar en el panel de iconos y pulse *[CTRL]+[T*]. Aparecerá un mensaje pidiendo confirmación para eliminar los iconos siguientes al seleccionado
Repintado	Si por alguna causa se deteriorase la pantalla podrá repintarla pulsando [*F9]*
Rejilla	Permite activar o desactivar la cuadrícula del área de edición

Rejilla

Esta opción permite activar o desactivar la cuadrícula del área de edición. Esta cuadrícula separa los pixeles del icono permitiendo distinguir al usuario la zona correspondiente a cada punto cuando éstos son del mismo color. La utilización de la cuadrícula permite un mayor control en el dibujo de iconos.

Esta opción se ejecuta pulsando el botón 

.

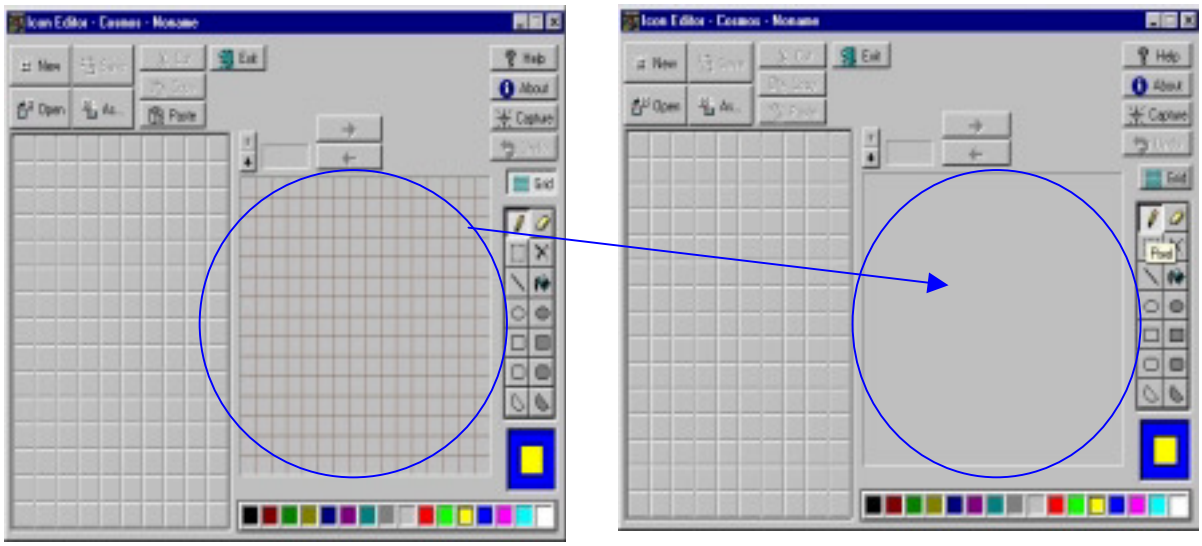


Figura 13.7. Desactivar la rejilla

Capítulo 14
El Editor de Código

Contenidos

1. Introducción

Cosmos incluye un editor de código fuente, de nombre *CEdit*, que permite la generación e impresión de documentos. Si bien su manejo es similar al de otros editores existentes en dicho entorno, *CEdit* incorpora, además, algunas técnicas de edición propias.

El editor de código se invoca de dos formas diferentes:

- Utilice el icono del grupo de programas Cosmos.

- Por medio del comando cedit, cuya sintaxis es la siguiente:

cedit [opciones] [nombreFichero]

Parámetros	Significado	
NombreFichero	Nombre del fichero que se desea editar. Si el fichero indicado no existe lo crea	
Opciones	Son las siguientes:	
	l “frase”	Edita el fichero y posiciona el cursor en el primer carácter de la “frase” encontrada. Si en “frase” sólo se indica una palabra no es necesario ponerla entre comillas
	- m	Abre el editor con la ventana maximizada
	- x extensión	Añade la extensión indicada al tipo de ficheros que puede utilizar el editor. Dicha extensión será la que se utilice por defecto al abrir un fichero
	- v	Muestra la versión del editor, así como su *upgrade*
	- ro	Abre el fichero para lectura solamente. Permite modificar el fichero para salvarlo con otro nombre
	-view	Abre el fichero para lectura solamente y no permite modificarlo

2. Aspecto general del Editor de Código

Esta pantalla incluye los siguientes elementos:

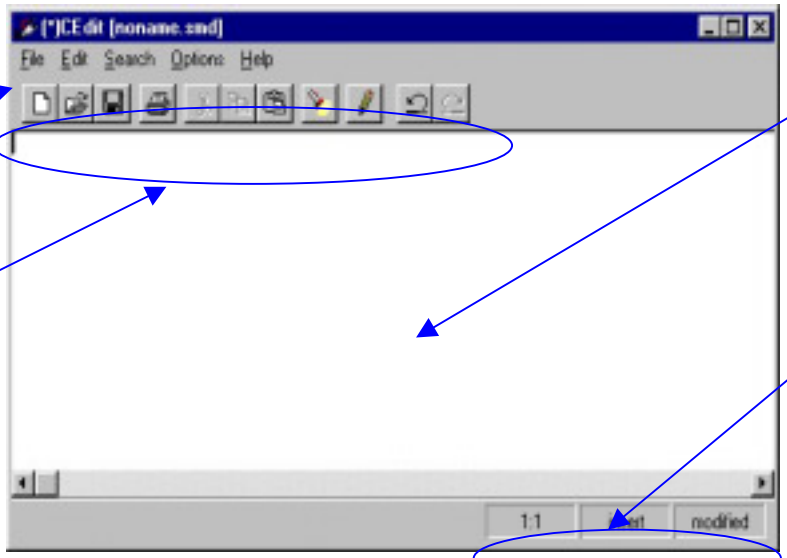


Figura 14.1. Aspecto del Editor de Código

El Editor de Código permite trabajar en dos modos de edición diferentes:

Inserción Inserta texto a partir del punto en que se encuentre situado el cursor y desplaza el texto existente.

Reemplazo Los caracteres que se escriban sustituirán al texto existente a partir del punto en que se encuentre situado el cursor.

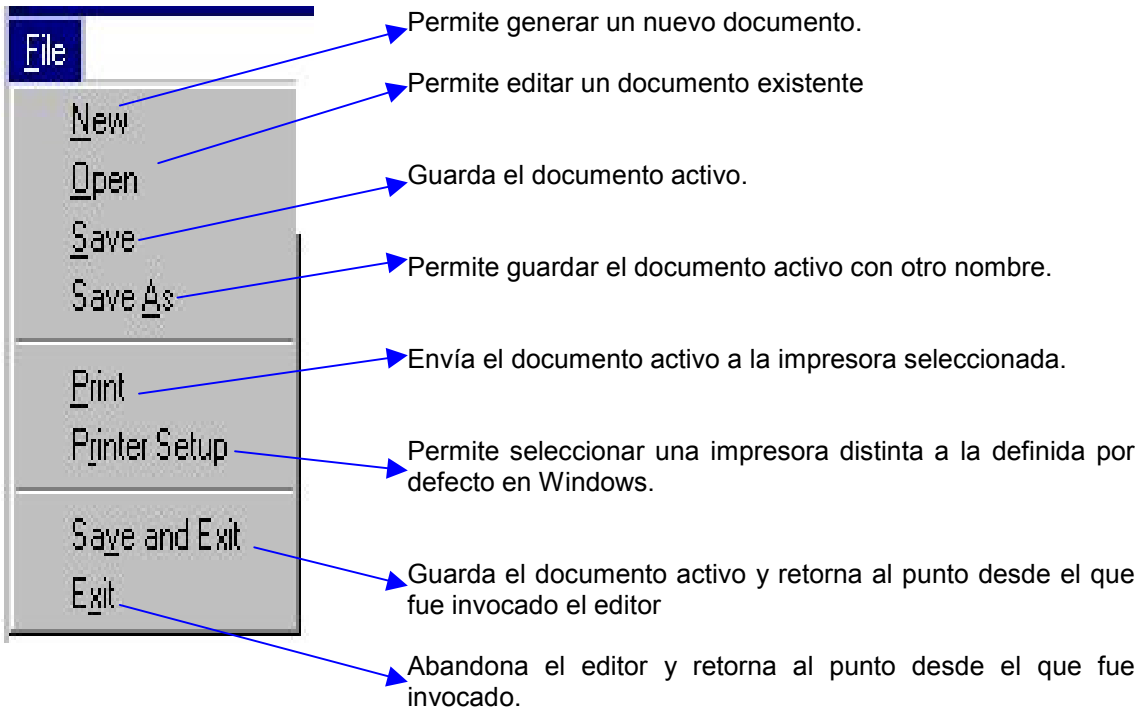
Para cambiar el modo de operación de inserción a reemplazo o viceversa pulse la tecla *[Insert]*.

El modo de operación por defecto al arrancar el Editor de Código es el de inserción. El modo de edición activo es el que figura en la parte inferior derecha de la pantalla en la línea de mensajes.

3. Elementos del menú del Editor de Código

Menú File

Las opciones incluidas este menú permiten la generación de documentos nuevos, acceder a documentos existentes, guardar, insertar e imprimir documentos.



Crear un documento (Opción New)

La opción *New* del menú *File* permite editar un documento nuevo. La forma de ejecución dependerá de si el documento activo ha sido guardado previamente o no y de las posibles modificaciones realizadas:

1. En caso de estar trabajando con un documento existente, la forma de ejecución de esta opción dependerá de las modificaciones realizadas, pidiendo conformidad o no al usuario para guardar los cambios antes de generar el nuevo documento.
2. Si el documento activo no tuviese aún asignado un nombre, la ejecución de esta opción dependerá igualmente de las posibles modificaciones. En caso afirmativo su ejecución será idéntica a la de la opción *Save As*, mientras que en caso contrario se abrirá automáticamente un documento nuevo denominado *nomame.smd*.

Esta opción también se puede ejecutar pulsando el botón  del menú de iconos.

Abrir un documento (Opción Open)

La opción *Open* del menú *File* permite abrir un documento existente. Al igual que ocurría en la opción *Nuevo*, su ejecución dependerá de las posibles modificaciones que se hubiesen realizado en el documento activo y de si se trata de un documento nuevo o ya existente a efectos de solicitar la salvaguarda de los posibles cambios.

La ejecución de esta opción muestra un cuadro de diálogo con la lista de ficheros y directorios para permitir la selección del documento con el que se desea trabajar.

Esta opción también se puede ejecutar pulsando el botón  en el menú de iconos.

Guardar un documento (Opción Save)

La opción *Save* del menú *File* guarda el contenido del documento activo. Si se trata de un documento nuevo, su ejecución será idéntica a la de la opción *Save As*, presentando un cuadro de diálogo para indicar el nombre que se desea asignar al documento y el directorio donde se desea guardar.

Esta opción también se puede ejecutar pulsando el botón  del menú de iconos o la tecla *[F3]* .

Guardar un documento con otro nombre (Opción Save As)

La opción *Save As* del menú *File* permite guardar tanto un documento de nueva creación como otro ya existente asignándole otro nombre. En el primer caso, el cuadro de diálogo que le aparecerá al usuario será idéntico al de la opción *Save* cuando ésta se ejecuta por primera vez.

Si el nombre asignado al documento coincidiese con el de algún otro ya existente, el sistema avisaría al usuario presentando un mensaje en pantalla, en cuyo caso el usuario podrá elegir entre:

- Guardar el documento con el mismo nombre (*Sobreescribir*), con lo cual se grabarían las modificaciones que hubieran podido realizarse sobre él (equivalente a la opción *Save*). En este caso el sistema pediría confirmación avisando que se trata de un documento ya existente que se va a sobreescribir.
- Asignar un nombre distinto al actual, con lo cual estaríamos generando un nuevo documento
- Cancelar la operación.

Imprimir un documento (Opción Print)

La opción *Print* del menú *File* permite obtener por impresora el contenido del documento activo. Al ejecutarla aparecerá un elemento de diálogo para establecer los distintos parámetros que intervienen en la impresión (orientación del papel, tamaño, impresora de salida, alimentación del papel, etc.).

Antes de proceder a la impresión de cualquier documento es recomendable guardar los cambios realizados para evitar posibles pérdidas de información ante cualquier error o problema.

El botón *Options* nos permitirá especificar características adicionales que dependerán del modelo de impresora elegido.

La opción *Print* puede ejecutarse directamente desde el menú de iconos pulsando el botón  .

Seleccionar impresora (Opción Printer Setup)

La opción *Printer Setup* del menú *File* permite seleccionar como impresora de salida una distinta de la definida por defecto en el Administrador de Impresión de Windows.

En aquellos casos en los que se desee emplear una impresora distinta de la definida por defecto, es aconsejable ejecutar esta opción antes de imprimir al objeto de que los ajustes del documento (márgenes, tipos de letra disponibles, etc.) se adecuen a las características de la nueva impresora, evitando de esta forma sorpresas desagradables a la hora de imprimir (saltos de página no deseados, problemas con las fuentes de letra, etc.).

Guardar un documento y Salir (Opción Save and Exit)

Permite guardar el documento y salir del editor. Realiza todas las funciones propias de salvar el documento y después ejecuta la opcion Exit.

Salir del editor (Opción Exit)

La opción *Exit* del menú *File* cierra el editor y retorna al punto desde el que fue invocado. En caso de haber realizado modificaciones en el documento activo el sistema pedirá conformidad al usuario para salvar o no los cambios.

Esta opción se puede ejecutar directamente pulsando [ESC].

Menú Edit

Se pueden introducir cambios globales que afecten a bloques de texto con sólo seleccionarlos y ejecutar a continuación algunas de las opciones del meú *Edit*. Para seleccionar texto en un documento podremos emplear indistintamente el ratón o el teclado:

Para seleccionar texto con el ratón proceda de la siguiente manera:

1. Sitúe el cursor delante del primer carácter del bloque de texto a seleccionar y pulse el botón izquierdo del ratón.
2. Arrastre el cursor hasta el último carácter que desee incluir en la selección.
3. Suelte el botón del ratón.

Para seleccionar una sola palabra haga doble click sobre ella.

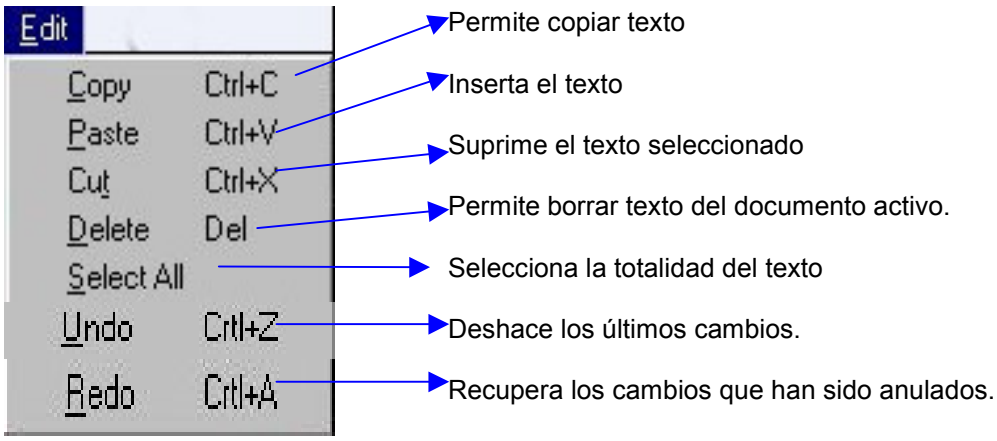
Para cancelar la selección haga click en cualquier lugar del documento.

Para seleccionar texto con el teclado proceda de la siguiente manera:

1. Utilice las teclas de dirección para desplazar el cursor hasta el primer carácter del bloque a seleccionar.
2. Pulse la tecla *[Mayúsculas]*. y manteniéndola presionada, utilice las teclas de dirección para mover el cursor hasta el último carácter que desee seleccionar.

Si desea cancelar la selección pulse cualquiera de las teclas de dirección.

Las opciones incluidas en el menú *Edit* permiten realizar ciertas operaciones de edición sobre el texto del documento activo.



Copiar y mover texto (Opción Copy)

La opción *Copy* del menú *Edit* copia el texto seleccionado del documento activo al portapapeles de Windows para permitir su inserción en otro lugar del mismo documento, en otro distinto o, incluso, en otra aplicación.

Para copiar texto:

1. Seleccione el texto que desee copiar.
2. Ejecute la opción *Copy* del menú *Edit* (podrá emplear también la tecla *[CTRL] + [C]* o el botón  del menú de iconos).

También puede copiar texto seleccionándolo y arrastrándolo con el botón izquierdo del ratón a la posición deseada.

Puede mover texto de sitio seleccionándolo y arrastrándolo con el botón izquierdo del ratón a la posición deseada manteniendo la tecla *[Mayúsculas]* pulsada.

Pegar Texto (Opción Paste)

La opción *Paste* del menú *Edit* inserta el texto previamente copiado al portapapeles de Windows en el lugar donde se sitúe el cursor.

Para pegar texto:

1. Copie o corte el texto que desee insertar en otro lugar.
2. Sitúe el cursor en el lugar donde desee insertar el texto copiado.
3. Ejecute la opción *Paste* del menú *Edit* o pulse el botón  del menú de iconos o la tecla *[CTRL] + [V]*.

Cortar texto (Opción Cut)

La opción *Cut* del menú *Edit* elimina el texto seleccionado y lo almacena en el portapapeles de Windows para permitir su inserción en otro punto del documento activo, en otro distinto o, incluso, en otra aplicación.

Para cortar texto:

1. Seleccione el texto que desee cortar.
2. Ejecute la opción *Cut* del menú *Edit* o pulse el botón  del menú de iconos o la tecla *[Ctrl] + [X]*.

Borrar texto (Opción Delete)

La opción *Delete* del menú *Edit* permite eliminar texto del documento activo.

Para borrar texto disponemos de varias posibilidades:

1. Para borrar caracteres a la derecha del punto de inserción pulse la tecla *[Del] ([Supr]).*
2. Para borrar caracteres a la izquierda del punto de inserción pulse la tecla *[Retroceso] ([Backspace*]).
3. Para borrar palabras o bloques de texto selecciónelos previamente y ejecute la opción *Delete* o pulse cualquiera de las teclas citadas en los dos puntos anteriores.

La opción *Delete* aparece inactiva en el menú si no hay texto seleccionado, aunque puede ejecutarse mediante *[Del]* o *[Retroceso]* para eliminar caracteres uno a uno.

Seleccionar todo el texto del documento (Opción Select All)

La opción *Select All* del menú *Edit* selecciona todo el contenido del documento activo para permitir la realización de determinadas operaciones (copiarlo, cortarlo, borrarlo, etc.).

Deshacer los últimos cambios (Opción Undo)

El editor de código fuente guarda un registro con algunas de las últimas modificaciones realizadas.

Si comete algún error o cambia de idea mientras esta escribiendo puede utilizar la opción *Undo* del menú *Edit* para hacer correcciones y deshacer los últimos cambios.

Para deshacer la última acción, pulse la tecla *[Ctrl]* + *[Z]* o el botón  del menú de botones. Para recuperar los cambios, una vez anulados utilice la opción *Redo*.

Si desea deshacer más cambios, ejecute otra vez el comando *Undo*.

Rehacer modificaciones (Opción Redo)

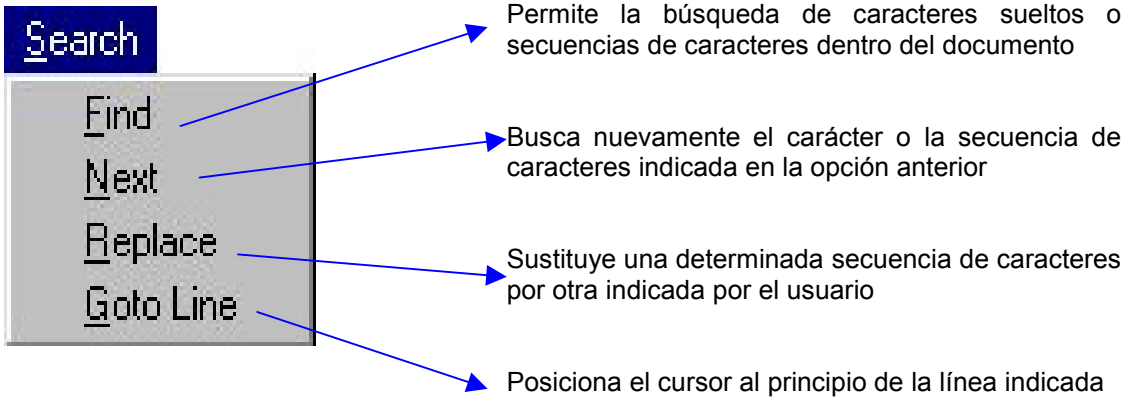
La opción *Redo* del menú *Edit* permite recuperar los cambios que han sido anulados mediante la opción *Undo*.

Para rehacer la última acción, pulse la tecla *[Ctrl]*+ *[A]* o el botón  del menú de botones. Para deshacer los cambios realizados con esta opción *Redo* del menú *Edit*, utilice la opción *Undo*.

Si desea rehacer más cambios, ejecute otra vez el comando *Redo*.

Menú Search

El menú *Search* incluye diversas opciones que permiten la búsqueda y sustitución de caracteres dentro del texto del documento activo. Dichas opciones son las siguientes



Buscar caracteres o palabras (Opción Find)

La opción *Find* del menú *Search* permite localizar dentro del documento activo un determinado carácter o secuencia de caracteres indicada por el usuario. El proceso de búsqueda comenzará siempre a partir del punto en que se encuentre situado el cursor.

Al ejecutarla aparecerá un cuadro de diálogo con los siguientes campos:

Figura 14.2. Aspecto del Cuadro de diálogo Find

Find	Campo de edición donde deberá indicarse la secuencia de caracteres a buscar.
Whole Word	Casilla de verificación para indicar que el texto a buscar deberá ser localizado como una palabra completa y cuando forme parte de otra.
Case	Casilla de verificación que permite indicar al editor que la secuencia a buscar deberá estar escrita exactamente igual a la indicada, distinguiendo entre mayúsculas y minúsculas.
Up/Down	Permite indicar que la búsqueda se realice desde el punto en que se encuentre situado el cursor hacia el principio del documento o hacia el final.

Una vez indicada la secuencia de caracteres a buscar bastará con pulsar el botón *Next* o la tecla *[RETURN]* para iniciar el proceso de búsqueda.

El cursor se posicionará sobre la secuencia indicada, resaltándola. Este proceso se repetirá tantas veces como se pulse la tecla o el botón indicados. Para finalizar la búsqueda pulse el botón *Cancel*.

Si no existe ninguna secuencia de caracteres igual a la especificada el programa mostrará en pantalla un mensaje indicando esta circunstancia, y lo mismo sucederá al llegar a la última de las secuencias encontradas.

Figura 14.3. Cuadro de diálogo de secuencia no encontrada

La opción *Find* puede ejecutarse también pulsando el botón  del menú de iconos.

Siguiente búsqueda (Opción Next)

La opción *Next* del menú *Search* permite repetir directamente desde la ventana de edición del documento activo el último proceso de búsqueda indicado a través de la opción *Find* o *Replace*.

Reemplazar texto (Opción Replace)

La opción *Replace* del menú *Search* permite buscar dentro del documento activo una determinada secuencia de caracteres para sustituirla por otra. Este proceso podrá realizarse automáticamente para todas las secuencias iguales a la indicada existentes en el documento (pulsando el botón *All* en el cuadro de diálogo que aparece al ejecutar la opción), o bien examinando una a una cada secuencia para confirmar o no la sustitución (utilice para ello el botón *Next*).

El cuadro de diálogo que aparece al ejecutar esta opción es similar al de *Find*, incluyendo en este caso un nuevo campo para indicar el texto que deberá reemplazar a la secuencia buscada.

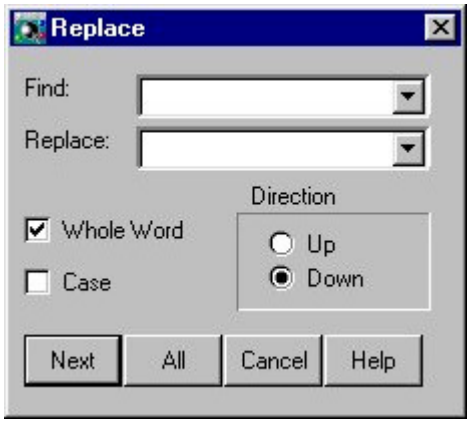


Figura 14.4. Cuadro de diálogo Raplace

Ir a una línea (Opción Goto line)

La opción *Go to Line* del menú *Search* permite ir directamente a una determinada línea de texto dentro del documento activo. Se trata de una opción especialmente útil cuando se trabaja con documentos de cierta extensión y se conoce con exactitud dónde se encuentra una determinada información, evitando el tener que ir página a página para llegar al punto deseado.

Al ejecutar la opción se presentará un elemento de diálogo para indicar el número de línea a la que deseamos acceder. Al pulsar el botón *Ok* el punto de inserción se posicionará al principio de la línea indicada.

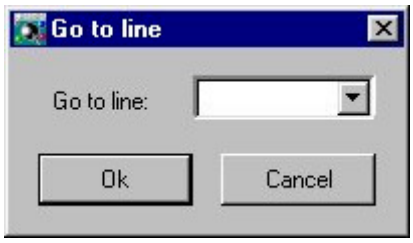
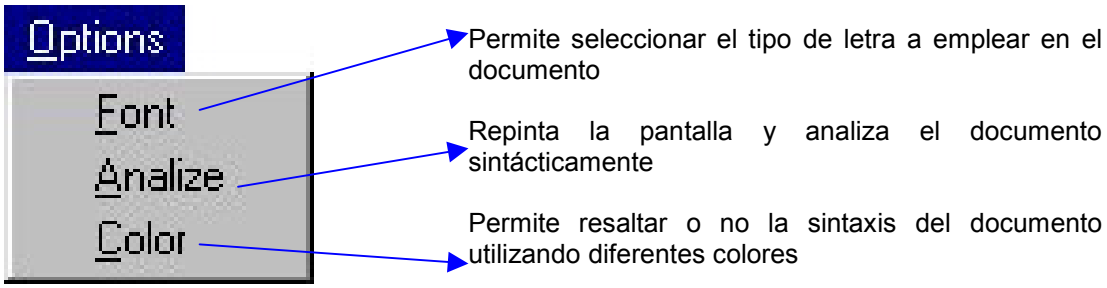


Figura 14.5. Cuadro de diálogo Go to line

Menú Options

Las opciones incluidas en el menú *Options* permiten configurar determinados aspectos generales de la edición.



Seleccionar la fuente (Opción Font)

Mediante la opción *Font* del menú *Options* el usuario podrá seleccionar la fuente de trabajo con la que se mostrará e imprimirá el documento. Al ejecutarla se mostrará un cuadro de diálogo para elegir el tipo de letra, su tamaño y estilo (normal, negrita, cursiva, etc.).

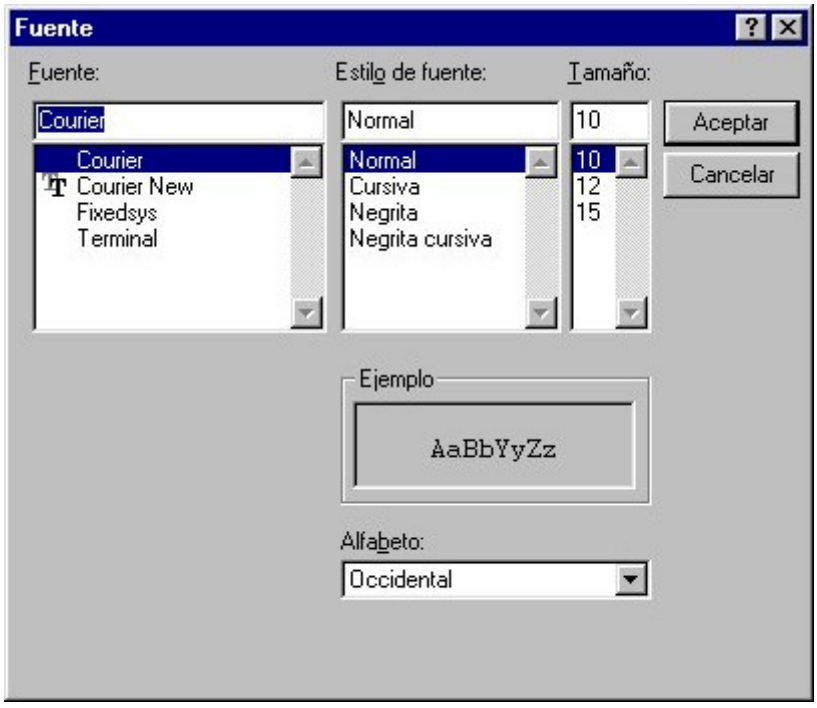


Figura 14. 6. Cuadro de elección de la Fuente

Las fuentes que se presentan en este diálogo son de tipo fijo, es decir, todos los caracteres se imprimirán con el mismo ancho.

Esta opción se puede ejecutar desde el menú de iconos pulsando el botón  .

Analizar sintacticamente un documento (Opción Analize)

La opción *Analize* del menú *Options* repinta la pantalla y analiza sintácticamente la estructura del documento.

Seleccionar los colores del documento (Opción Color)

La opción *Color* del menú *Options* permite resaltar o no la sintaxis del documento mostrando en distintos colores los diferentes elementos del lenguaje:

- Palabras reservadas.
- Identificadores.
- Números.
- Cadenas de caracteres (*string*).
- Símbolos de puntuación.
- Líneas de comentarios (entre llaves: *{* y *}* o *//*).

Capítulo 15
Editor de Esquemas Conceptuales de datos

Contenidos

1. Introducción

La realización de informes, a menudo llamados *listados*, a partir de la información contenida en una base de datos, es una tarea que consume gran cantidad de recursos de los departamentos de sistemas de información debido a las constantes peticiones de modificaciones que, si bien suelen ser sencillas, precisan de un proceso completo de análisis, implementación y pruebas.

EasyReport (como veremos en el siguiente capítulo) viene a resolver este problema, permitiendo al usuario final definir y obtener sus propios informes sin necesidad de conocer la estructura interna del sistema de información.

Esto es posible gracias a que EasyReport parte de un *Esquema Conceptual de Datos* (ECD), que será definido por el programador. Para esta tarea, Cosmos dispone de un generador automático de ECDs que permite al programador seleccionar las tablas que desee de la base de datos, estableciendo múltiples vistas y restricciones sobre dicha base de datos.

Con EasyReport el usuario puede definir y obtener sus informes a partir de un ECD previamente creado por el programador.

Para poder realizar informes con EasyReport es necesario, previamente, la creación de uno o varios Esquemas Conceptuales de Datos (ECDs) por un programador que conozca la estructura de la base de datos.

2. Esquema Conceptual de Datos (ECD)

Un *Esquema Conceptual de Datos* (ECD) está constituido por la información proporcionada por el analista o administrador de la base de datos. Esta información nos permitirá conocer la estructura semántica de la base de datos.

En cada ECD se suministrará la información necesaria de cada una de las tablas y columnas de la base de datos a las que se quiera permitir el acceso a un usuario o grupo de usuarios. La idea es que este acceso a la base de datos esté limitado a la información que aparezca definida en cada ECD.

Se puede definir más de un Esquema Conceptual de Datos para una misma base de datos, de forma que cada usuario o grupo de usuarios tenga acceso a uno solo de ellos, por tanto, cada ECD podrá proporcionar una visión diferente de la base de datos.

Un Esquema Conceptual de Datos es un fichero ASCII en el que se define la visión que se quiere dar de la base de datos sobre la que se va a trabajar. Estos ficheros deberán tener la extensión *.trw*, es decir el nombre de un ECD deberá tener la siguiente estructura:

nombre_ecd.trw

Las secciones de un Esquema Conceptual de Datos son:

Sección	Función
.CONFIGURATION(*)	Permite la definición de algunos parámetros de configuración del ECD, así como los distintos tipos de formato que podrá utilizar el usuario en ejecución para crear sus informes.
.VTABLE	Permite la definición de las tablas virtuales que proporcionarán al usuario final una visión determinada de la base de datos. En un ECD deberá existir al menos una sección .VTABLE.
.WINDOWS (*)	Permite la definición de ventanas de selección que ayuden al usuario en el momento de establecer condiciones sobre las columnas, o bien durante la edición de los parámetros a la hora de ejecutar.
.PARAMETERS (*)	Permite la definición de parámetros que se pedirán al usuario en el momento de ejecutar un informe. Pertenece siempre a las tablas definidas en la .VTABLE que la antecede
.JOINS (*)	Permite la definición de enlaces entre las tablas de la sección .VTABLE correspondiente y otras tablas de la base de datos.
.VCOLS	Permite la definición de columnas virtuales para las tablas definidas en una .VTABLE determinada. Cada sección .VTABLE debe tener forzosamente una sección .VCOLS

(*) Secciones opcionales

Estructura de un Esquema Conceptual de Datos

Un ECD estará compuesto por una o más secciones .CONFIGURATION y una o más secciones .VTABLE seguidas cada una de ellas por sus correspondientes secciones .WINDOWS, .PARAMETERS, .JOINS y .VCOLS.

Las secciones .WINDOWS, .PARAMETERS y .JOINS son opcionales y sólo se definirán si son necesarias.

Un Esquema Conceptual de Datos mínimo deberá tener al menos una sección .VTABLE y una .VCOLS.

Cada sección .VTABLE contendrá al menos la definición de una tabla virtual. Si se definen varias entradas, todas ellas compartirán las secciones .WINDOWS, .PARAMETERS, .JOINS y .VCOLS que van a continuación. La razón para definir varias entradas en una sección .VTABLE es precisamente el hecho de compartir la misma estructura. Así pues, en una misma sección .VTABLE será lógico definir:

- Varias tablas de la base de datos que contengan las mismas columnas.
- La misma tabla o *view* de la base de datos con diversas condiciones de selección u ordenación.

Así pues, un Esquema Conceptual de Datos aparecerá con la siguiente estructura:

[.CONFIGURATION]
...
[.VTABLE]
...
[.WINDOWS]
...
[.PARAMETERS]
...
[.JOINS]
...
[.VCOLS]
...
[.VTABLE]
...
[.PARAMETERS]
...
[.JOINS]
...
[.VCOLS]
...

Es decir, existe una sección en la que se definen ciertos parámetros de configuración (.CONFIGURATION) y grupos de secciones (.VTABLE, .WINDOWS, .PARAMETERS, .JOINS y .VCOLS) en los que se definen las diferentes tablas que verá el usuario final. Las secciones .WINDOWS, .PARAMETERS, .JOINS y .VCOLS van siempre asociadas a una sección .VTABLE.

Cualquier línea de un ECD que comience por el carácter # será considerada como comentario.

Ejemplo:

```
# Definición de la tabla de unidades de la base de datos
#
.VTABLE
Tabla de Unidades = unidades
.VCOLS
Cod. Unidad = unidad , A2
Unidades = descripcion , A10
```

La Directiva INCLUDE

Un Esquema Conceptual de Datos puede estar dividido en diferentes ficheros. Por esto mismo es posible incluir la siguiente directiva en cualquier lugar de un ECD:

$INCLUDE {"$VAR" | "fichero"}

Parámetros	Significado
VAR	Nombre de una variable de entorno exportada que contendrá el path completo del fichero a incluir
fichero	Path completo del fichero a incluir

Dentro de un ECD podrán existir tantas directivas INCLUDE como sea necesario. Un fichero incluido en un Esquema Conceptual de Datos puede contener a su vez otras directivas INCLUDE, siendo cinco el nivel máximo de anidamiento.

Ejemplo:

```
$INCLUDE "$VARINC"
$INCLUDE "/usr/demo/mifichero.inc"
```

La primera de estas líneas indica que existe una variable de entorno, denominada *VARINC*, que contendrá el path completo del fichero que se desea incluir; dicha variable deberá estar exportada. Por su parte, la segunda línea indica explícitamente el path completo del fichero que se desea incluir.

Sección .CONFIGURATION

Las cláusulas que se pueden incluir en esta sección son las siguientes:

Cláusulas	Significado
BUFFER	Para optimizar la repetición de la ejecución de un listado
DATABASE	Permite definir la base de datos que será utilizada por defecto por el ECD
DEFINE FORMAT	Permite la definición de formatos de salida de informes y la redefinición de los formatos por defecto
DISABLE	Permite desactivar ciertos permisos para el usuario
OUTPUT PATH	Permite definir el directorio por defecto para la salida de informes a ficheros
PRIV	Para determinar el privilegio de utilización del usuario

Cláusula BUFFER

Para optimizar la repetición de la ejecución de un listado. Su sintaxis es la siguiente:

BUFFER {ON | OFF}

Parámetro	Significado
ON	La primera vez que se ejecuta un informe se almacena una copia en un fichero temporal con el resultado de aquél. De este modo, si el informe se vuelve a ejecutar sin modificar sus condiciones ni cambiar las columnas seleccionadas se utilizará el resultado anterior, con lo que se ejecución será más rápida
OFF	No utiliza la característica expuesta en el párrafo anterior. Éste es el valor por defecto

Cláusula DATABASE

Permite definir la base de datos que será utilizada por defecto por el ECD. Su sintaxis es la siguiente

DATABASE base_datos

Parámetro	Significado
base_datos	Nombre de la base de datos que utilizará por defecto EasyReport

Cláusula DEFINE FORMAT

Permite la definición de formatos de salida de informes y la redefinición de los formatos por defecto. Su sintaxis es la siguiente:

DEFINE [DEFAULT]FORMAT "formato" TYPE [DEFAULT] tipoFormato opciones

Parámetro	Significado	
“formato"	Literal entre comillas que indica el nombre del formato. Este nombre será el que se presente al usuario para elegir un formato preestablecido.	
TipoFormato	Tipo de formato, que puede ser:	
	LISTING	Listado encolumnado con cabecera y pie de página, grupos de ruptura, totalización, etc
	EXPORT	Salida típica para envío de datos a otros programas o generación de ficheros de carga de otros programas
	FORM	Listado tipo ficha

Lista de Opciones para el Formato LISTING

Opciones	Significado
HEADER "cab"	Cabecera del listado
TRAILER "pie"	Pie de página del listado
LABELS {ON | OFF}	Salida con o sin etiquetas de columnas
LINELEN núm	Longitud de línea del listado
PAGELEN núm	Longitud de página del listado. Si num=0 indica que la página no tiene longitud, efectuándose entonces el listado de forma continua
GAP núm	Separación entre columnas del listado
TOP núm	Margen superior
BOTTOM núm	Margen inferior
LEFT núm	Margen izquierdo
RIGHT núm	Margen derecho
RUN “texto” "cmd"	Programa donde se enviará el listado
EXTFILE ext	Extensión de los ficheros de salida de informes
TOTALS{ON|OFF}	Indica que en el listado sólo se incluirán los totales de los grupos.

Lista de Opciones para el Formato EXPORT

Opciones	Significado
DELIMITER “ delim "	Delimitador de campos
HEADER {ON | OFF}	Salida con o sin etiquetas de columnas
UNDERLINE {ON | OFF}	Salida con cabeceras subrayadas o no
QUOTED {ON | OFF}	Literales entrecomillados o no en la salida
LASTDELIMITER {ON|OFF}	Limitador tras el último campo o no
RUN "texto" "cmd"	Programa al que se enviará el listado
EXTFILE ext	Extensión de los ficheros de salida de informes

Lista de Opciones para el Formato FORM

Opciones	Significado
LINELEN núm	Ancho de la ficha
PAGELEN núm	Altura de la ficha
TOP núm	Margen superior
BOTTOM núm	Margen inferior
LEFT núm	Margen izquierdo
RIGHT núm	Margen derecho
RUN "texto" "cmd"	Programa al que se enviará el listado
EXTFILE ext	Extensión de los ficheros de salida de informes
LABELS {ON | OFF}	Salida con o sin etiquetas de columnas

Donde:

cab Literal entre comillas que indica el texto y formato de la cabecera.
pie Literal entre comillas que indica el texto y formato del pie de página.
texto Literal entre comillas que se mostrará al usuario en la opción A Comando del menú Ejecución del EasyReport. Puede usar & delante de la letra que desee utilizar como acelerador.
Cmd Literal entre comillas que indica la línea de comando a la que se enviará la salida del informe. Se podrá especificar una variable de entorno, precedida del signo $, que contenga dicha línea de comando.
delim Literal entre comillas indicando el carácter que se utilizará como separador de campos.
núm Número.

La cláusula DEFAULT a continuación de DEFINE indica que ese formato será el que se utilizará al arrancar EasyReport.

La cláusula DEFAULT a continuación de TYPE indica la modificación del formato por defecto para ese tipo.

Cláusula DISABLE

Permite desactivar ciertos permisos para el usuario. Su sintaxis es la siguiente:

DISABLE lista_permisos

Parámetro	Significado
lista_permisos	Lista de posibles permisos del ECD a eliminar de los del usuario.

Los permisos son:

Permiso	Significado
SAVE	Permiso para guardar informes
OPEN	Permiso para leer informes creados
MODIFY	Permiso para modificar un informe ya creado
PRINT	Permiso para enviar la salida de un informe a un programa
PRINTFILE	Permiso para enviar la salida de un informe a un fichero
DELETE	Permiso para borrar informes
DESIGN	Permiso para modificar el diseño del formato de salida del informe y su tipo de letra
DATABASE	Permiso para seleccionar una base de datos distinta a la empleada por defecto
OUTPATH	Permiso para seleccionar el directorio de salida de informes a ficheros

Ejemplos

```
DISABLE SAVE MODIFY PRINTFILE DELETE OUTPATH
DISABLE DATABASE OUTPATH
```

Cláusula OUTPUT PATH

Permite definir el directorio por defecto para la salida de informes a ficheros. Si no se incluye esta cláusula se empleará por defecto el directorio en curso.

Su sintaxis es la siguiente:

OUTPUT PATH {"$VAR" | "fichero"}

Parámetro	Significado
“VAR"	Nombre de una variable de entorno (precedida de $) que contendrá el path completo del directorio al que se enviarán las salidas de informes a fichero
"fichero"	Path completo del fichero al que se enviará la salida del informe

Ejemplos:

```
OUTPUT PATH "/usr/salida/listados"
OUTPUT PATH "$SALIDA"
```

Cláusula PRIV

Para determinar el privilegio de utilización del usuario. Su sintaxis es la siguiente:

PRIV número

Parámetro	Significado
Número	Número de privilegio con el que actuará el usuario en caso de no estar definida la variable de entorno TRWPRIV

Sección .VTABLE (Tabla Virtual)

En estas secciones se definen una o varias entradas referidas a una tabla, view o sentencia SELECT de la base de datos. Se pueden definir varias entradas si todas ellas tienen la misma estructura (columnas con el mismo nombre y tipo).

La sintaxis de esta sección es la siguiente:

.VTABLE
usr_tab = tabla [,[DEFAULT] WHERE "cond"]
[,[DEFAULT] ORDER <lista_cols>]
[, PRIV núm]
usr_tab = tabla [,[DEFAULT] WHERE "cond"]
[,[DEFAULT] ORDER <lista_cols>]
[, PRIV núm]

Parámetro	Significado	
usr_tab	Nombre con el que el usuario va a identificar la tabla	
tabla	Nombre real de la tabla o view en la base de datos	
cond	Condición WHERE aplicable al SQL. En la cláusula WHERE sólo se podrán utilizar columnas reales de la tabla SQL	
lista_cols	Lista de columnas reales de la tabla para la cláusula ORDER BY, separadas por comas, aplicable al SQL. El número máximo de columnas que se puede incluir en una cláusula ORDER BY es 8. Asimismo, la suma de las longitudes de las columnas indicadas en esta cláusula no puede exceder de 120 bytes. El nombre de las columnas puede ir seguido de las palabras reservadas:	
	asc	Especifica que el resultado debe ordenarse de forma ascendente. Esta opción es la que el CTSQL utiliza por defecto, por lo que no será necesario especificarla
	desc	Especifica que el resultado debe ordenarse de forma descendente
PRIV núm	Privilegio de la tabla. El privilegio de una tabla por defecto es cero (0). Una tabla con privilegio mayor al del usuario no será visible para éste	

Las cláusulas WHERE y ORDER se añadirán a las seleccionadas por el usuario. Si se usa la cláusula DEFAULT WHERE o DEFAULT ORDER, éstas sólo se aplicarán si el usuario no especifica condiciones de selección u ordenación, respectivamente.

Sección .WINDOWS

En esta sección se podrán definir ventanas de selección que se podrán utilizar, pulsando el botón que tenga asociado, en las siguientes situaciones:

- Situados en la edición del campo Valor, en la opción Edición/Condiciones.
- Al utilizar la sección .PARAMETERS, en el momento de editar los valores que contendrán los parámetros de ejecución.

La sintaxis de esta sección es la siguiente:

.WINDOWS
winname "instr_select" [líneas LINES] [cols_núm COLUMNS] [LABEL "título"]
GET num_col

Parámetro	Significado
winname	Nombre que identifica a la ventana
instr_select	Instrucción SELECT que se mostrará en la ventana
líneas	Número de líneas que se mostrarán en la ventana
cols_núm	Número que identifica a las primeras columnas de la SELECT (que son las que aparecerán en pantalla)
título	Título de la ventana
núm_col	Número de la columna de la SELECT que se desea seleccionar. Este número puede ser mayor que el de las columnas que aparecen en pantalla

Sección .PARAMETERS

En esta sección se definen los parámetros usados en la sección .VTABLE para establecer una condición definida a priori.

Su sintaxis es la siguiente:

.PARAMETERS
parmlabel = param, {A|N|D|T} long [, DEFAULT {núm|"string"}]
[, WINDOW winname]

Parámetro	Significado	
parmlabel	Etiqueta con la que se va a pedir el parámetro al usuario	
param	Nombre del parámetro usado en la sección .VTABLE.	
A|N|D|T	Tipo	Alfanumérico (A) Numérico (N) Fecha (D) Hora (T)
long	Longitud de display	
núm	Valor numérico por defecto para tipo N	
string	Valor alfanumérico por defecto para tipos A, D y T	
winname	Identificador de una de las ventanas definidas en la sección .WINDOWS. La columna de selección indicada en la cláusula GET deberá tener el mismo tipo de dato que el parámetro donde se recoge (param)	

Los parámetros serán solicitados al usuario al ejecutar un informe que incluya alguna tabla que utilice parámetros en la sección .VTABLE. En el momento de edición de los parámetros, pulsando el botón,  se mostrará en pantalla la ventana de ayuda para poder recoger el valor. También se pueden enviar los parámetros a EasyReport a través de la línea de comando del Easyreport.

Sección .JOINS

En esta sección se definen los posibles enlaces que pueden realizarse desde la tabla definida en la sección .VTABLE.

Su sintaxis es la siguiente:

.JOINS
join = [MASTER|joinused] <desde_col[,desde_col] …]>
tabla <hasta_col[, hasta_col]>
…

Parámetro	Significado
join	Nombre que identifica el enlace
MASTER	Indica que el enlace se hace desde la tabla definida en la sección .VTABLE
joinused	Indica que el enlace se hace desde la tabla previamente enlazada con el join: joinused
desde_col	Nombre de una columna con la que se hará el join. Esta columna debe pertenecer a la tabla definida en la sección .VTABLE o a la enlazada previamente, según se defina MASTER o joinused, respectivamente.
tabla	Nombre de la tabla enlazada
hasta_col	Nombre de la columna en la tabla enlazada cuyo valor debe ser igual al de desde_col. Se pueden definir enlaces multicolumna separando por comas las columnas de enlace

Sección .VCOLS (Columnas Virtuales)

En esta sección se definen las columnas para cada .VTABLE definida anteriormente. Estas columnas podrán corresponder con columnas reales de la tabla, expresiones SQL, columnas obtenidas a partir de un join o bien resultados de una expresión SELECT.

La sintaxis de esta sección es la siguiente:

```
.VCOLS
virtual_col ={columna|EXP "expresión1"|
        SELECT "instr_select" [USING <lista_cols>] |
        EVAL "expresión2"} , {A|N|D|T}[+|-]long
        [, MASK núm_máscara] [, NDEC decimales] [, JOIN join]
        [, PRIV núm] [, ALIAS alias]    [, WINDOW ventana]
```

Parámetro	Significado
virtual_col	Nombre con el que el usuario va a identificar esta columna
columna	Nombre de una columna real de la tabla definida en .VTABLE o bien de la tabla enlazada con un join si esta columna se especifica con la cláusula JOIN
expresión1	Expresión SQL que obtiene el valor deseado
instr_select	Instrucción SELECT que obtiene el valor deseado
USING	Para parametrizar la instr_select
lista_cols	Lista de columnas usadas como variables host de ejecución de la SELECT
A, N, D, T	Tipo de columna (alfanumérica, numérica, fecha u hora)
+, -	Ajuste a derecha o izquierda, respectivamente. Si no se especifica se asumirá derecha para numéricos e izquierda para el resto
long	Longitud de display de la columna
núm_máscara	Número de máscara
decimales	Número de decimales para los campos numéricos.
join	Nombre del join a emplear
núm	Número que indicará el privilegio de la columna
expresión2	Expresión válida que podrá incluir: paréntesis, los operadores +,-, * o /, números, literales entre comillas, alias
alias	Identificador con el que se podrá referenciar a la columna virtual en la definición de otras columnas virtuales dentro de una EVAL.
ventana	Identificador de una de las ventanas definidas en la sección .WINDOWS. La columna de selección indicada en la cláusula GET deberá tener el mismo tipo de dato que la columna donde se recoge (columna). En la edición del campo Valor, en la opción del menú Edición/Condiciones del EasyReport, pulsando el botón  , se mostrará en pantalla la ventana de ayuda para poder recoger el valor

La diferencia entre las cláusulas EXP y EVAL radica en que la primera sólo podrá incluir expresiones válidas de SQL con columnas de tablas de la base de datos, mientras que la cláusula EVAL servirá para evaluar expresiones en las que se podrán incluir columnas virtuales a las que se haya asignado un alias.

En la siguiente tabla se muestran la máscaras que se pueden utilizar para los diferentes tipos de datos.

Máscaras para...	Número	Aspecto
Integer y Smallint	0	General
	1	0
	2	#,##0
Decimales	0	General
	1	0
	2	0.00
	3	#,##0
	4	#,##0.00
	5	0%
	6	0.00%
	7	0.00E+00
Fechas	0	General
	1	dd/mm/yy
	2	mm/dd/yy
	3	yy/mm/dd
	4	dd/mm/yyyy
	5	dd mmm yyyy
Horas	0	General
	1	hh:mm:ss
	2	hh:mm:ss M
	3	hh:mm
	4	hh:mm PM
	5	hh
	6	hh PM

3. Entorno de desarrollo de Esquemas Conceptuales de Datos (ECD's)

El editor EasyReport es una herramienta que permite construir Esquemas Conceptuales de Datos (ECD's) para una base de datos. Este editor incluye un completo entorno de desarrollo que permite la edición, generación automática y pruebas de ECD's.

Las formas de invocar dicho entorno es mediante:

- el icono 

- la línea de comando:

coscds [-v [-d base_datos] [nombre_fichero.trw]

Parámetros	Significado
-v	Muestra la versión del comando
Base_datos	Nombre de la base de datos sobre la que se desea construir el Esquema Conceptual de Datos
Nombre_fichero.trw	Indica el nombre del fichero, de extensión trw, con el que se quiere trabajar. Se puede incluir el PATH completo o relativo

Como puede observarse, el editor de esquemas conceptuales de datos trabaja por defecto con nombre de fichero Noname.trw. Esto indica que se está trabajando con un ECD nuevo.

Aspecto del Editor de ECD's

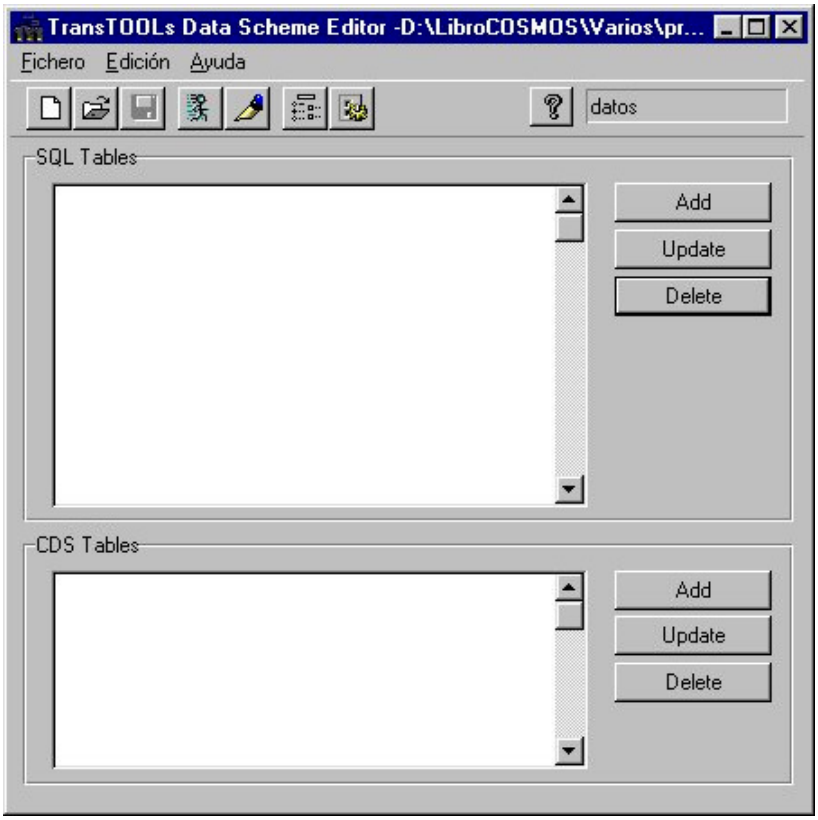


Figura 15.1. Aspecto del editor de ECD's

Menú Fichero del editor de ECD's

El menú Fichero permite crear un ECD nuevo, abrir ECD's existentes, su edición y/o borrado y la realización de pruebas de los ECD's generados.

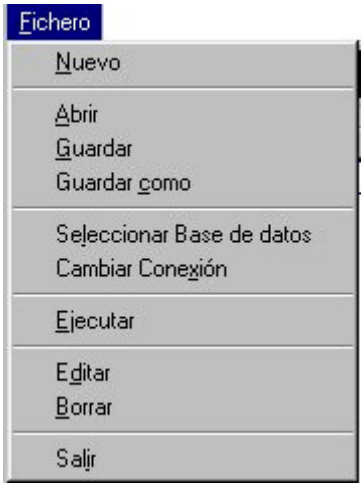


Figura 15.2. Menú Fichero

Opción	Significado
Nuevo	Permite crear un ECD
Abrir	Permite abrir un ECD existente
Guardar	Guarda el ECD en curso
Guardar Como...	Permite guardar el ECD en curso con otro nombre
Seleccionar base de datos	Permite elegir la base de datos con la que se desea trabajar
Cambiar Conexión	Esta opción permite establecer comunicación con otras bases de datos distintas de Cosmos, así como seleccionar el tipo de conexión (local o cliente-servidor)
Ejecutar	Ejecuta EasyReport para probar el ECD en curso
Editar	Edita el ECD en curso
Borrar	Borra el ECD en curso
Salir	Abandona editor de ECD y retorna al sistema operativo o a la aplicación que haya lanzado el editor

Opción Nuevo

Esta opción permite generar un ECD nuevo. La forma de ejecución dependerá de si el ECD en curso ha sido guardado previamente o no y de las posibles modificaciones realizadas:

- En el caso de estar trabajando con un ECD existente, la ejecución de esta opción dependerá de las modificaciones realizadas, pidiendo conformidad o no para guardar los cambios antes de generar el nuevo ECD (de nombre Noname.trw).
- Si el ECD en curso no tuviese aún asignado un nombre, la ejecución de esta opción dependerá igualmente de las posibles modificaciones. En caso afirmativo, la ejecución sería idéntica a la de la opción Guardar como... (para asignar un nombre al ECD), mientras que en caso contrario se generaría un nuevo Noname.trw.

Esta opción también se puede ejecutar pulsando el botón  .

Opción Abrir

Esta opción permite abrir un ECD existente. Al igual que ocurría en la opción Nuevo, su ejecución dependerá de las posibles modificaciones que se hubiesen realizado en el ECD en curso y de si éste es un ECD nuevo o existente.

La ejecución de esta opción presenta un cuadro de diálogo con la lista de ficheros de extensión *trw* existentes en el directorio indicado en la variable de entorno TRWPATH (o, en su defecto, en el directorio en curso) para seleccionar el nuevo ECD activo.

Esta opción se puede ejecutar igualmente pulsando el botón  .

Opción Guardar

Esta opción guarda el contenido del ECD en curso. Si se trata de un ECD nuevo, se presentará un cuadro de diálogo idéntico al de la opción Guardar como....

Esta opción también se puede ejecutar pulsando el botón  .

Opción Guardar Como

Esta opción permite guardar tanto un ECD de nueva creación como otro ya existente asignándole otro nombre. Al ejecutarla se presentará en pantalla el mismo cuadro de diálogo que el de la opción Guardar cuando se ejecuta por primera vez.

Si el nombre asignado al ECD coincidiese con el de algún otro ya existente, el sistema avisaría al usuario presentando un mensaje en pantalla, en cuyo caso el usuario podrá elegir entre:

- Guardar el ECD con el mismo nombre (Sobre-escribir), con lo cual se grabarían las modificaciones que hubieran podido realizarse sobre él. En este caso el sistema pediría confirmación avisando de que se trata de un ECD ya existente que se va a sobreescribir.
- Asignar un nombre distinto al actual, con lo cual estaríamos creando un nuevo ECD.
- Cancelar la operación.

Opción Seleccionar Base de Datos

Esta opción permite elegir la base de datos en curso. Al ejecutarla se mostrará un cuadro de diálogo con el nombre de la base de datos activa, pudiendo entonces indicar el nombre de la nueva base de datos con la que se desea trabajar. Este nombre deberá ser un literal alfanumérico de 10 caracteres de longitud como máximo.

Figura 15.3. Seleccionar una base de datos

Si la conexión definida en el entorno es local aparecerán los nombres de las bases de datos disponibles, mientras que si la conexión es remota sólo aparecerá la base de datos previamente seleccionada.

Una vez indicado el nombre de la base de datos pulse el botón Ok. El nombre de la nueva base de datos se mostrará en la esquina superior derecha de la ventana principal.

Opción Cambiar Conexión

Mediante esta opción se podrá establecer una conexión local o remota (cliente-servidor), pudiendo en este último caso acceder a otros gestores de bases de datos.

Al ejecutarla aparecerá un cuadro de diálogo con una lista desplegable, que incluye las conexiones definidas en el fichero de configuración, para seleccionar la conexión deseada. Si el entorno especificado para la conexión seleccionada no tiene definida la variable de entorno DBPASSWD automáticamente se pedirá la contraseña para poder establecer la conexión en la siguiente ejecución.

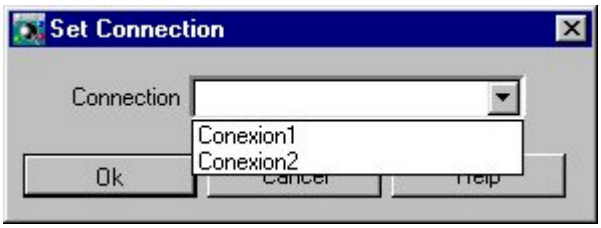


Figura 15.4. Seleccionar una conexión

Opción Ejecutar

Ejecuta EasyReport para probar el ECD en curso. Esta opción también se puede ejecutar pulsando el botón  .

Opción Editar

Esta opción permite editar el ECD en curso. El editor empleado será el definido en la variable de entorno DBEDIT. En el caso de que dicha variable no tuviese definido ningún editor, o se quisiese cambiar el existente, ejecute la opción Entorno del menú Edición y asigne a esta variable el nombre del editor deseado.

Esta opción también se puede ejecutar pulsando el botón  .

Opción Borrar

Esta opción permite borrar el ECD en curso. Antes de proceder al borrado el sistema pedirá conformidad al usuario.

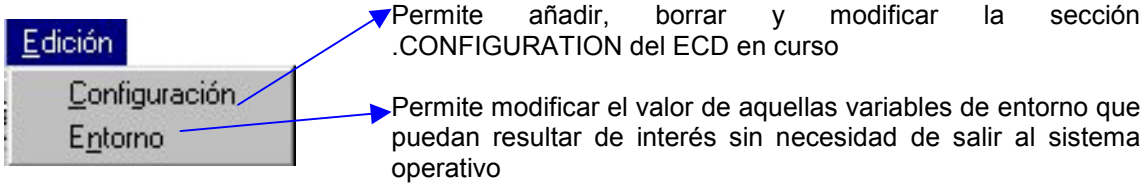
Opción Salir

La ejecución de esta opción cierra el Entorno de Desarrollo de ECDs y retorna al sistema operativo o al comando desde el que fue invocado dicho entorno.

Menú Edición del editor de ECD's

Las opciones incluidas en el menú Edición permiten modificar las variables de entorno utilizadas por el comando ceasyrep en ejecución, así como modificar o generar la sección .CONFIGURATION de un ECD. Esta opción también se puede activar mediante el icono 

Dichas opciones son las siguientes:



Opción Configuración

Esta opción permite añadir, borrar y modificar la sección .CONFIGURATION del ECD en curso.

Al ejecutar esta opción se muestra un cuadro de diálogo que contiene tres fichas que se explican a continuación:

- *Ficha Configuración*

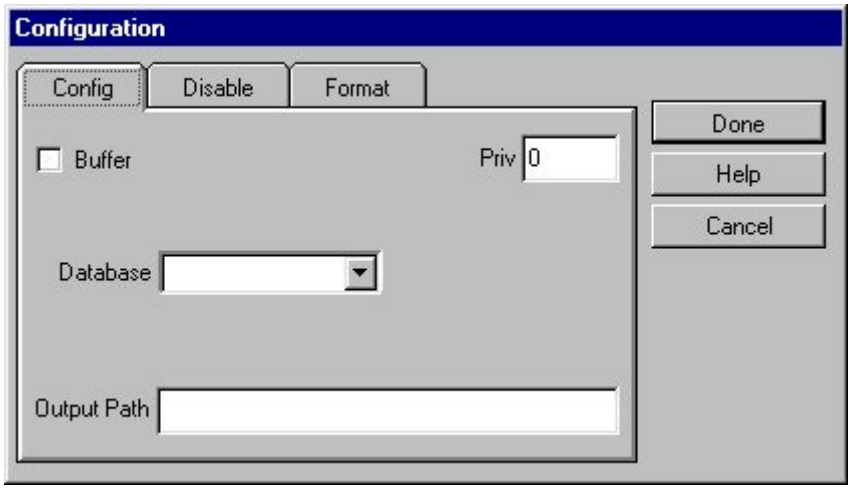


Figura 15.5. Ficha Configuración

Opciones	Significado
Buffer	Si activa esta casilla de verificación, la primera vez que se ejecuta un informe se almacena una copia en un fichero temporal con el resultado de aquél
Privilegio	Número de privilegio con el que actuará el usuario en caso de no estar definida la variable de entorno TRWPRIV
Base de datos	Nombre de la base de datos que utiliza por defecto EasyReport
Directorio de salida	Path completo del fichero al que se enviará la salida del informe.

- *Ficha Eliminar permisos*

Contiene una lista de casillas de verificación con los posibles permisos del Esquema Conceptual de Datos a eliminar de los del usuario.

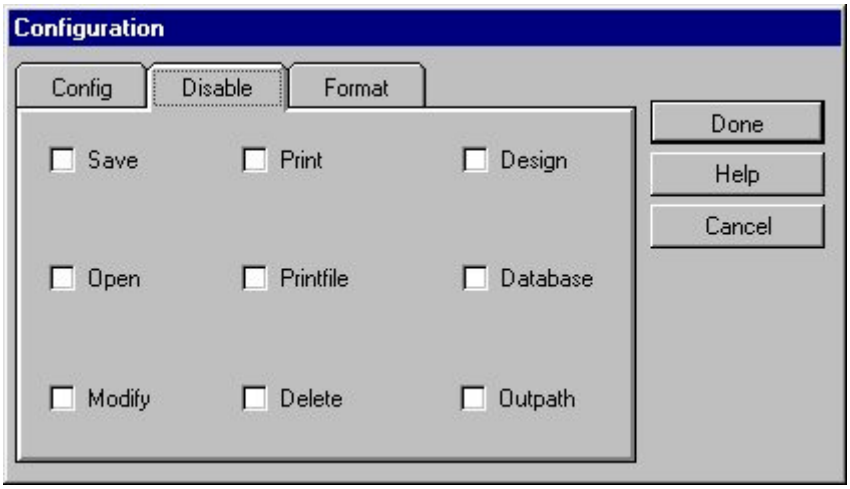


Figura 15. 6. Ficha de Eliminar permisos

Permisos	Significado
Save	Permiso para guardar informes
Open	Permiso para leer informes existentes
Modify	Permiso para modificar un informe existente
Print	Permiso para enviar la salida de un informe a la impresora
PrintFile	Permiso para enviar la salida de un informe a un fichero
Delete	Permiso para borrar informes
Design	Permiso para modificar el diseño del formato de salida del informe y su tipo de letra
Database	Permiso para seleccionar una base de datos distinta a la empleada por defecto
Outpath	Permiso para seleccionar el directorio de salida de informes a ficheros

Para inhabilitar un permiso active la casilla de verificación correspondiente.

- *Ficha Formatos*

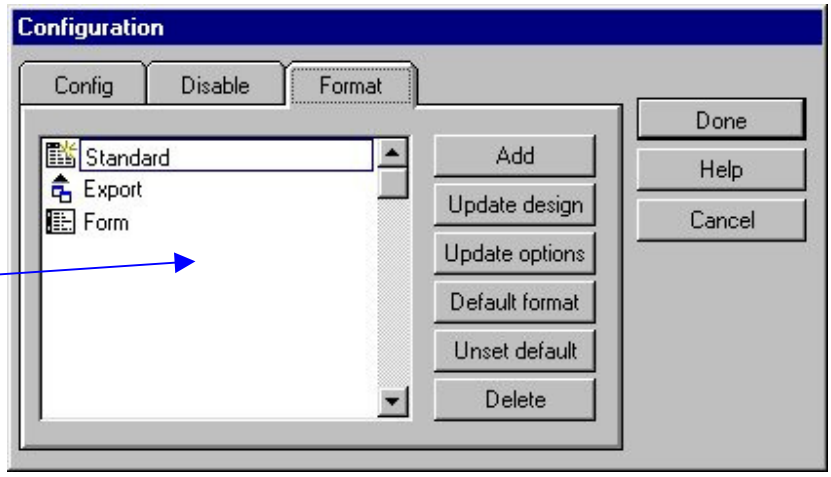


Figura 15.7. Ficha de Formato

Los formatos se presentan con los siguientes iconos:

Para formato Standard. Presenta las filas de datos encolumnadas.

Para formato Export. Éste es el formato de descarga de una base de datos, y se utiliza para exportar los datos que se seleccionen para el informes a otra base de datos o aplicación.

Para formato Form. Cada columna se presenta en una ficha.

Los botones que aparecen en esta ficha tienen la siguiente funcionalidad:

Add Permite añadir una nueva cláusula FORMAT al ECD en curso, es decir, añade un formato.

Al seleccionar esta opción se muestra un cuadro de diálogo con los siguientes campos:

Figura 15.8. Cuadro de diálogo para añadir un formato

Para añadir un nuevo formato:

1. Seleccione el tipo de formato.
2. Escriba en el campo Name el nombre que desea asignar al nuevo formato.
3. Si desea añadir el nuevo formato al ECD en curso, pulse el botón Ok; en caso contrario pulse el botón Cancel.
4. Al pulsar el botón Ok se crea automáticamente el nuevo formato con los parámetros por defecto para el tipo seleccionado. A continuación se muestra un cuadro de diálogo que permite establecer el formato de las páginas del documento para su salida por impresora, por pantalla o a un fichero. El nuevo formato se mostrará en la lista de la ficha Formatos del ECD en curso.

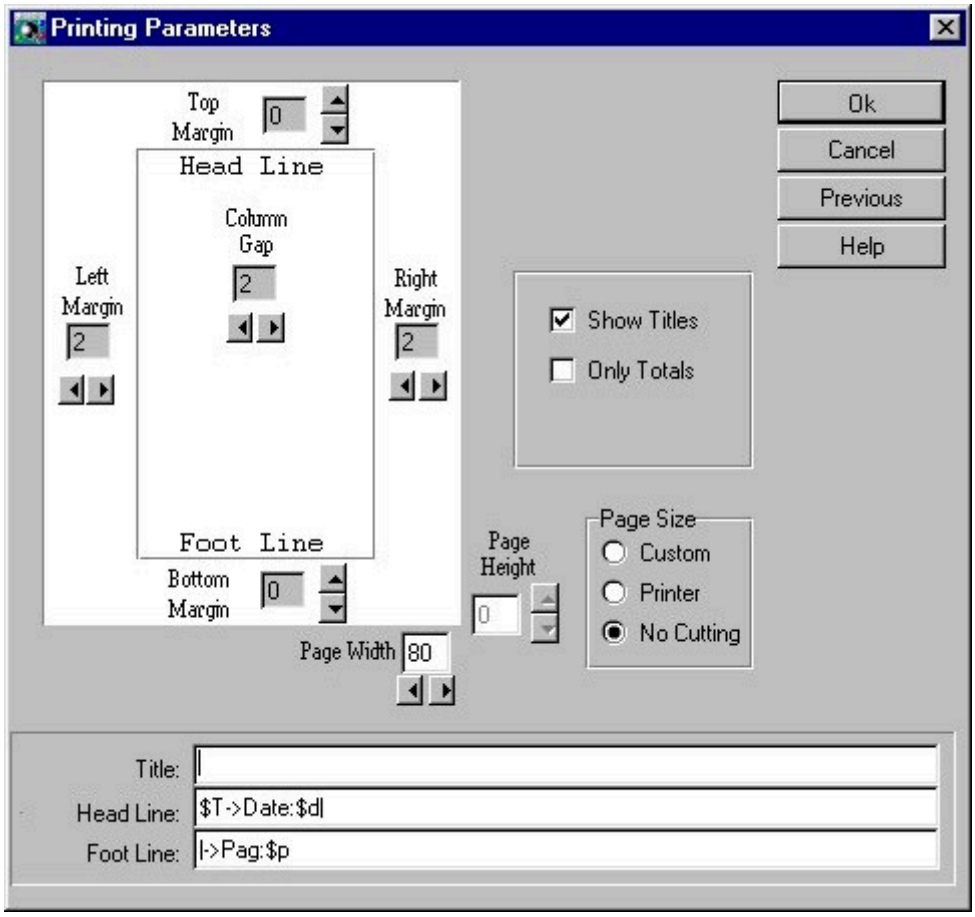


Figura 15.9. Cuadro de diálogo para seleccionar los parámetros de impresión del nuevo formato

Update design Permite modificar los parámetros de impresión del formato seleccionado:

1. Seleccione en la lista de formatos el que desee modificar.
2. Pulse el botón Update design.

Las opciones que aparecerán en el cuadro de diálogo que se muestra al ejecutar esta opción dependerán del tipo de formato elegido. Estas opciones podrán ser las siguientes:

Para formato LISTING:

Opciones	Significado
Mostrar títulos	Salida con o sin etiquetas de columnas
Ancho de página	Número de caracteres por línea del listado. Esta opción sólo estará activa cuando se seleccione Definida o No cortar en el campo Tam. Página, pudiendo entonces indicar el ancho de línea deseado o pulsar el botón para incrementar su valor o para disminuirlo
Longitud de página	Número de líneas por página. Si su valor es 0 indica que la página no tiene longitud, efectuándose entonces el listado de forma continua . Esta opción sólo estará activa cuando se seleccione Definida en el campo Tam. Página, pudiendo entonces indicar la longitud deseada o pulsar el botón para incrementar su valor o para disminuirlo
Separación	Espacio entre columnas del informe. Este valor se puede modificar escribiendo directamente sobre el campo o pulsando para incrementarlo o para disminuirlo
Márgenes	Márgenes superior, inferior, izquierdo y derecho del informe
Sólo totales	Indica que en el informe sólo se incluirán los totales de los grupos. Es decir, no se listarán las líneas que conforman dichos totales

Opciones	Significado
Título	Campo de edición en el que el usuario podrá indicar el título que desee asignar al informe
Cabecera	Campo de edición en el que el usuario podrá indicar el texto que desee utilizar como cabecera del informe
Pie de página	Pie de página del informe
Definida	Se tomarán los valores de ancho y alto definidos a la izquierda, pero sólo se imprimirá la parte que quede dentro del tamaño de la página de impresión. Esta opción se utilizará habitualmente para salida a fichero o en el caso de que se desee forzar un corte de la página a un tamaño más pequeño
Impresora	Calculará el número se filas y columnas que caben según el tamaño de hoja definido en la impresora por defecto o en la opción Seleccionar impresora. Esta opción no estará disponible cuando no se haya definido una impresora por defecto o si se produce algún error en la selección de la misma
No cortar	Se tomará el ancho definido a la izquierda y se ignorarán los saltos de página, considerando todo el informe como una sola página. Cuando se imprima, el ancho quedará limitado al de la impresora seleccionada, produciéndose un salto de página físico cada vez que se complete la capacidad de la hoja de la impresora seleccionada
Previos	Recupera los valores definidos inicialmente

Para formato EXPORT

Opciones	Significado
Delimitador	Carácter que se empleará como delimitador de campos. Si desea utilizar como delimitador un tabulador escriba \t
Imprimir cabecera	Salida con o sin etiquetas de columnas
Imprimir subrayado	Salida con cabeceras subrayadas o no
Literales	Utilizar entrecomillados o no en la salida
Delimitador al final	Limitador tras el último campo o no

Para formato FORM

Opciones	Significado
Imprimir títulos	Salida con o sin etiquetas de columnas
Ancho de página	Número de caracteres por línea del listado. Esta opción sólo estará activa cuando se seleccione Definida o No cortar en el campo Tam. página, pudiendo entonces indicar el ancho de línea deseado o pulsar el botón para incrementar su valor o para disminuirlo
Longitud de página	Número de líneas por página. Si su valor es 0 indica que la página no tiene longitud, efectuándose entonces el listado de forma continua (todo el listado es considerado como una sola página). Esta opción sólo estará activa cuando se seleccione Definida en el campo Tam. página, pudiendo entonces indicar la longitud deseada o pulsar el botón para incrementar su valor o para disminuirlo
Márgenes	Márgenes superior, inferior, izquierdo y derecho del informe
Definida	Se tomarán los valores de ancho y alto definidos a la izquierda, pero sólo se imprimirá la parte que quede dentro del tamaño de la página de impresión. Esta opción se utilizará habitualmente para salida a fichero o si se desea forzar un corte de la página a un tamaño más pequeño

Opciones	Significado
Impresora	Se calculará el número se filas y columnas que caben según el tamaño de hoja definido en la impresora por defecto o en la opción Seleccionar impresora. Esta opción no estará disponible cuando no se haya definido una impresora por defecto o si se produce algún error en la selección de la misma
No cortar	Se tomará el ancho definido a la izquierda y se ignorarán los saltos de página, considerando todo el informe como una sola página. Cuando se imprima, el ancho quedará limitado al de la impresora seleccionada, produciéndose un salto de página físico cada vez que se complete la capacidad de la hoja de la impresora seleccionada
Previos	Restaura los valores definidos inicialmente

Para modificar el diseño del formato seleccionado:

1. Modifique en el cuadro de diálogo los valores deseados.
2. Una vez introducidos los valores, pulse el botón Ok para validarlos, en caso contrario pulse Cancel.

Permite modificar los parámetros de salida de los informes del formato seleccionado:

1. Seleccione en la lista de formatos el que desee modificar.
2. Pulse el botón Update options.

Esta opción muestra un cuadro de diálogo que permite modificar los parámetros de salida de los informes para el formato seleccionado en ese momento. Las posibles opciones son:

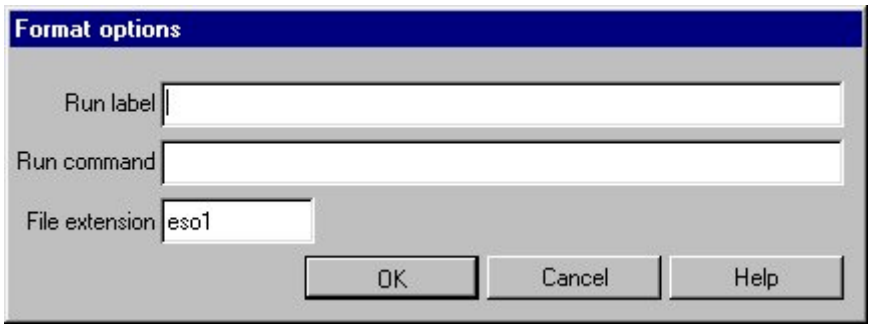


Figura 15.10. Cuadro de diálogo para modificar las opciones

Opciones	Significado
Etiqueta	Nombre que aparecerá al usuario en la opción A Comando de del menú Ejecución del EasyReport. Puede usar & delante de la letra que desee utilizar como acelerador
Comando de ejecución	Campo de edición en el que se pide la línea de comando a la que se enviará la salida del informe. Se podrá especificar una variable de entorno, precedida del signo $, que contenga dicha línea de comando
Extensión de fichero	Extensión de los ficheros de salida de informes

Para modificar los parámetros de salida del formato seleccionado:

1. Modifique en el cuadro de diálogo los valores deseados.
2. Una vez introducidos los valores, pulse el botón Ok para validarlos, en caso contrario pulse Cancel.

Default format Permite seleccionar uno de los formatos existentes en el ECD en curso como formato por defecto al arrancar EasyReport. Por defecto, el formato utilizado será el estándar.

Para definir un formato por defecto:

1. Seleccione en la lista de formatos aquel que desee utilizar como formato por defecto.
2. Pulse el botón Default Format. El formato elegido se mostrará automáticamente marcado con el icono .

Unset default Esta opción permite desactivar el formato que se tuviese definido por defecto al arrancar EasyReport, excepto el estándar.

Delete Permite borrar uno de los formatos existentes en el ECD en curso, excepto el estándar. Antes de proceder al borrado se pedirá conformidad al usuario.

En el caso de tener seleccionado uno de los tres formatos estándares se podrán resetear sus parámetros y recuperar sus valores iniciales, no permitiendo en este caso su borrado.

Para borrar un formato:

1. Seleccione en la lista de formatos aquel que desee borrar.
2. Pulse el botón Borrar.

Opción Entorno (Botón)

Esta opción permite modificar el valor de aquellas variables de entorno que puedan resultar de interés sin necesidad de salir al sistema operativo. Al ejecutar esta opción el programa presenta un cuadro de diálogo con los siguientes campos:

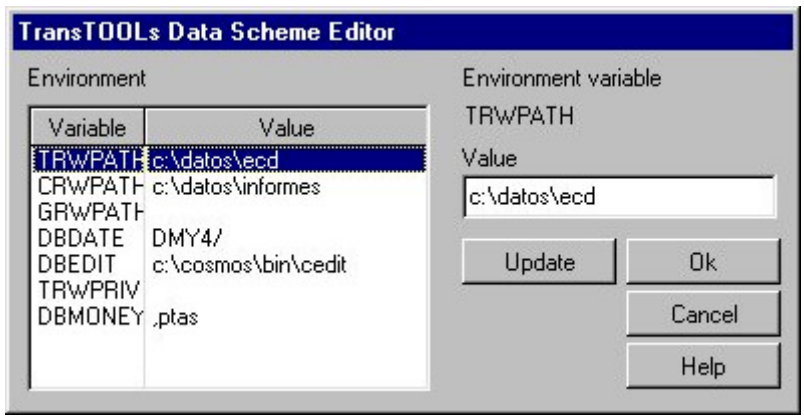


Figura 15. 11. Cuadro de diálogo de la opción entorno

Environment	Lista de las variables de entorno a las que se puede modificar su valor actual.
Environment variable	Campo de edición estático que muestra el nombre de la variable de entorno seleccionada en ese momento.
Value	Campo de edición que muestra el valor de la variable de entorno seleccionada en ese momento.
Update	Permite indicar un nuevo valor a la variable de entorno seleccionada, mostrándolo en la lista.

Para modificar las variables de entorno:

1. Seleccione la variable de entorno que desee modificar.
2. Escriba en el cuadro Value el nuevo valor de la variable de entorno.
3. Para asignar el nuevo valor a la variable de entorno, pulse el botón Update. La variable de entorno y su nuevo valor se mostrarán automáticamente en la lista de la izquierda.
4. Si desea que las modificaciones realizadas surtan efecto, pulse el botón Ok, en caso contrario pulse Cancel.

4. Crear Esquemas Conceptuales de Datos (ECD's)

En la ventana principal de la aplicación se permite modificar o generar una sección .VTABLE para el ECD en curso, así como sus secciones .WINDOWS, .PARAMETERS, .JOINS y .VCOLS.

Al ejecutar el comando coscds, es decir, el editor de esquemas conceptuales de datos, aparecerá un cuadro de con los siguientes campos:

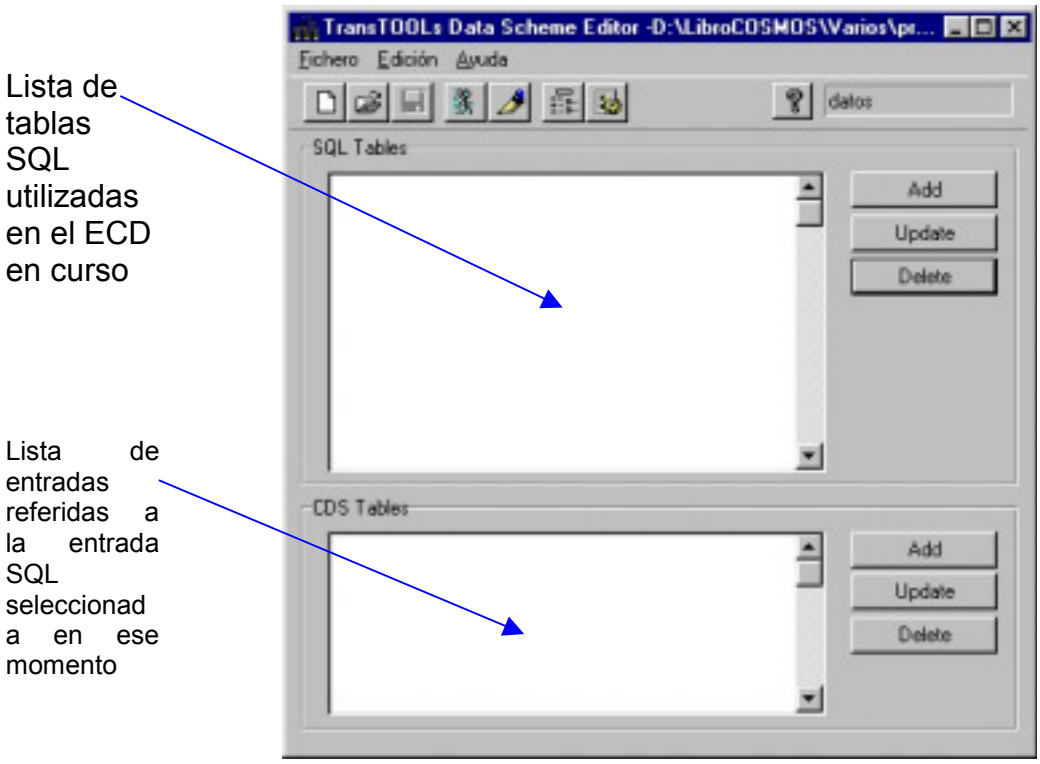


Figura 15.12. Aspecto del editor de esquemas conceptuales de datos

Las acciones a realizar por cada uno de los botones que aparecen en el cuadro de diálogo son las siguientes:

Añadir Tabla SQL	Permite crear una nueva sección .VTABLE para el ECD en curso. Para poder realizar la generación es preciso tener activa una base de datos. Se puede definir una base de datos por defecto en la sección .CONFIGURATION o seleccionar otra distinta.
Modificar Tabla SQL	Permite modificar la sección .VTABLE en curso.
Borrar Tabla SQL	Permite borrar una sección .VTABLE del ECD en curso.

Añadir Tabla ECD Permite añadir una nueva entrada referida a la tabla SQL en curso:

1. Pulse el botón Add.
2. Se mostrará el cuadro de diálogo de edición de tablas ECD. Introduzca en él las características de la nueva entrada.
3. Si desea que la acción realizada tome efecto, pulse el botón Ok, en caso contrario pulse Cancel.

Modificar Tabla ECD Permite modificar una entrada referida a la tabla SQL en curso:

1. Seleccione en la lista la entrada que desee.
2. Pulse el botón Update.
3. Aparecerá el cuadro de diálogo de edición en el que se muestran las características de la tabla ECD seleccionada.
4. Realice las modificaciones que estime oportunas.
5. Si desea que las modificaciones realizadas tomen efecto, pulse el botón Ok, en caso contrario pulse Cancel.

Borrar Tabla ECD Permite borrar una tabla del ECD. Para ello, seleccione en la lista la entrada que desee y pulse el botón Delete.

Añadir Tabla SQL

Esta opción permite generar un ECD para una o más tablas de las existentes en la base de datos, utilizando las claves referenciales (foreign keys) de cada tabla para definir los enlaces (joins). Asimismo, emplea las etiquetas (label) de cada columna de la base de datos (si existe) como nombre de columna para el usuario.

Para cada tabla se generarán las secciones .VTABLE y .VCOLS. En el caso de que la tabla tenga definida algún enlace también se generará la sección .JOINS.

Al seleccionar esta opción se muestra un cuadro de diálogo con la lista de tablas SQL de la base de datos que no han sido utilizadas en el ECD en curso. Para facilitar la generación automática esta lista es multiselección Las teclas [FLECHA ARRIBA] y [FLECHA ABAJO] nos permitirán movernos a lo largo de la lista. Asimismo, las teclas [INICIO] y [FIN] nos permitirán desplazarnos respectivamente al primero y al último elemento de la lista.

Para añadir tablas SQL al ECD en curso:

1. Seleccione en la lista todas las tablas que desee.
2. Una vez conforme con la selección pulse el botón Ok. Automáticamente comenzará la generación del ECD para las tablas seleccionadas.
3. Si pulsa el botón Cancel se cerrará el cuadro de diálogo sin añadir las tablas SQL al ECD en curso.

Edición Tabla SQL

Esta opción permite añadir, modificar y borrar los datos de las secciones .WINDOWS, .PARAMETERS, .JOINS y .VCOLS asociadas a la tabla en curso del ECD con el que se está trabajando.

Al ejecutarla se mostrará un cuadro de diálogo con los siguientes campos:

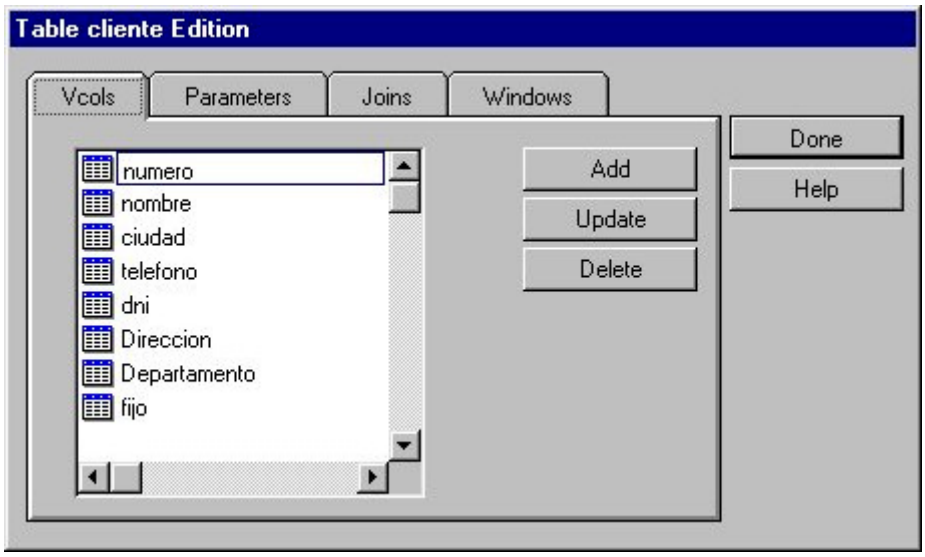


Figura 15.13. Cuadro de diálogo de edición de la tabla Clientes

- *Ficha Vcols*

Permite crear, añadir y modificar las diferentes secciones .VCOLS del ECD en curso. Incluye las siguientes opciones:

Añadir

Permite añadir una nueva columna a la sección .VTABLE en curso:

1. Pulse el botón Add.
2. Aparecerá el cuadro de diálogo de edición de columnas para definir las características de la nueva columna.
3. Si desea que la acción realizada tome efecto, pulse el botón Ok, en caso contrario pulse Cancel.

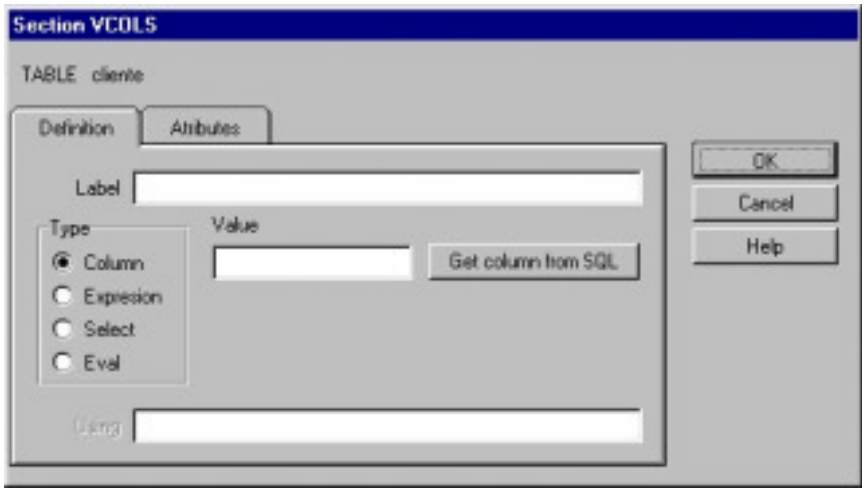


Figura 15.14. Cuadro de diálogo de la sección .Vcols

Modificar

Permite modificar las características de una columna:

1. Seleccione en la lista la columna que desee.
2. Pulse el botón Update.
3. Aparecerá el cuadro de diálogo de edición de columnas en el que se muestran las características de la columna seleccionada.
4. Realice las modificaciones que crea necesarias.
5. Si desea que las modificaciones realizadas tomen efecto, pulse el botón Ok, en caso contrario pulse Cancel.

Borrar

Para borrar una columna, seleccione en la lista la columna que desee y pulse el botón Delete.

- *Ficha Parameters*

Permite crear, añadir y modificar las diferentes secciones .PARAMETERS del ECD en curso. Incluye las siguientes opciones:

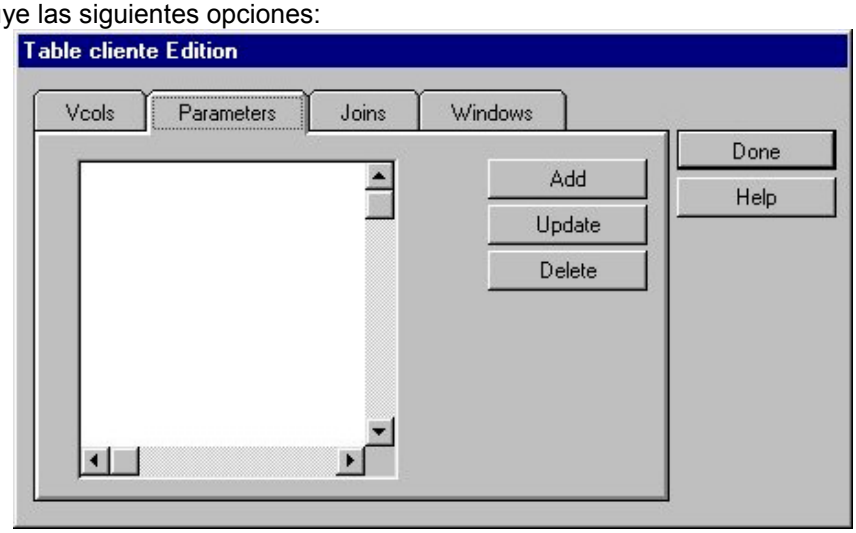


Figura 15.15. Ficha Parameters

Añadir

Permite añadir un nuevo parámetro a la sección .VTABLE en curso:

1. Pulse el botón Add.
2. Se mostrará el cuadro de diálogo de edición de parámetros. Introduzca en él las características del nuevo parámetro.
3. Si desea que la acción realizada tome efecto, pulse el botón Ok, en caso contrario pulse Cancel.

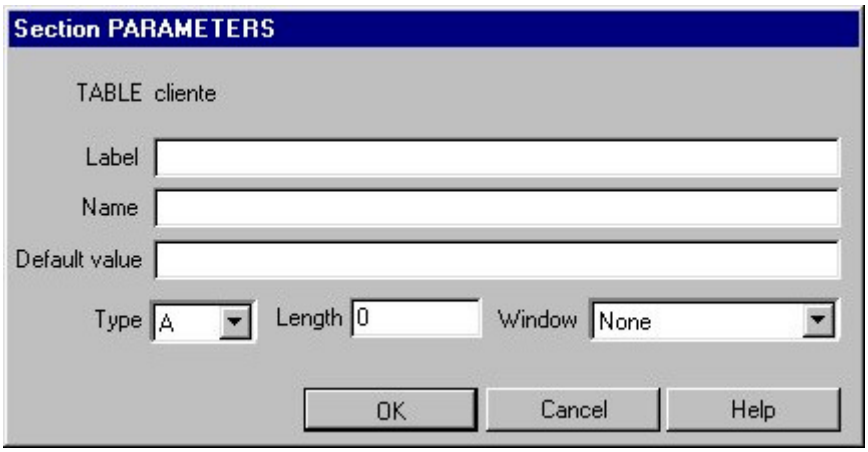


Figura 15.16. Cuadro de diálogo de la sección Parameters

Modificar

Permite modificar las características de un parámetro:

1. Seleccione en la lista el parámetro que desee modificar.
2. Pulse el botón Update.
3. Aparecerá el cuadro de diálogo de edición en el que se muestran las características del parámetro seleccionado.
4. Realice las modificaciones que crea necesarias.
5. Si desea que las modificaciones realizadas tomen efecto, pulse el botón Ok, en caso contrario pulse Cancel.

Borrar

Para borrar un parámetro, selecciónelo en la lista y pulse el botón Delete.

- ### *Ficha Joins*

Permite crear, añadir y modificar las diferentes secciones .JOINS del ECD en curso. Las opciones que incluye son las siguientes:

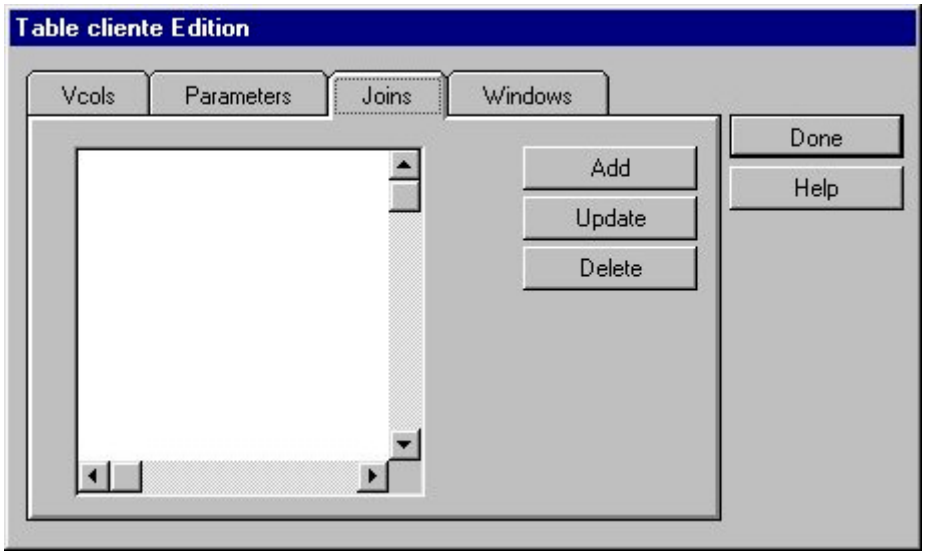


Figura 15.17. Ficha Joins

Añadir

Permite añadir un nuevo join a la sección .VTABLE en curso:

1. Pulse el botón Add.
2. En el cuadro de diálogo de edición de joins indique las características del nuevo join.
3. Si desea que la acción realizada tome efecto, pulse el botón Ok, en caso contrario pulse Cancel.

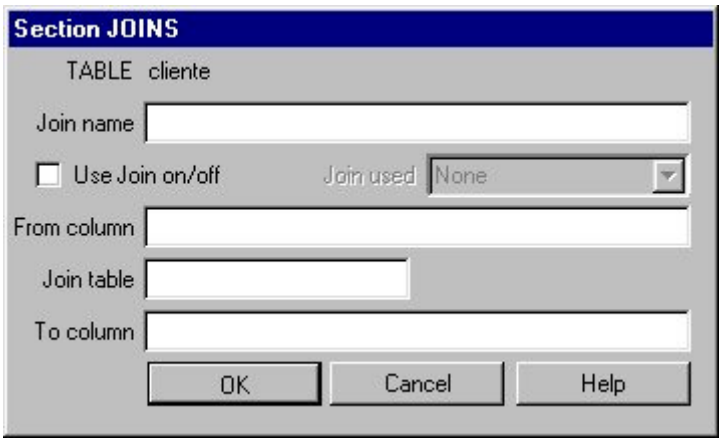


Figura 15.18. Cuadro de diálogo de la sección Joins

Modificar

Permite modificar las características del join:

1. Seleccione en la lista el join que desee modificar.
2. Pulse el botón Modificar.
3. Aparecerá el cuadro de diálogo de edición en el que se muestran las características del join seleccionado.
4. Realice las modificaciones que crea necesarias.
5. Si desea que las modificaciones realizadas tomen efecto, pulse el botón Aceptar, en caso contrario pulse Cancelar.

Borrar

Para borrar un join, selecciónelo en la lista y pulse el botón Borrar.

- *Ficha Windows*

Permite crear, añadir y modificar las diferentes secciones .WINDOWS del ECD en curso. Incluye las siguientes opciones:

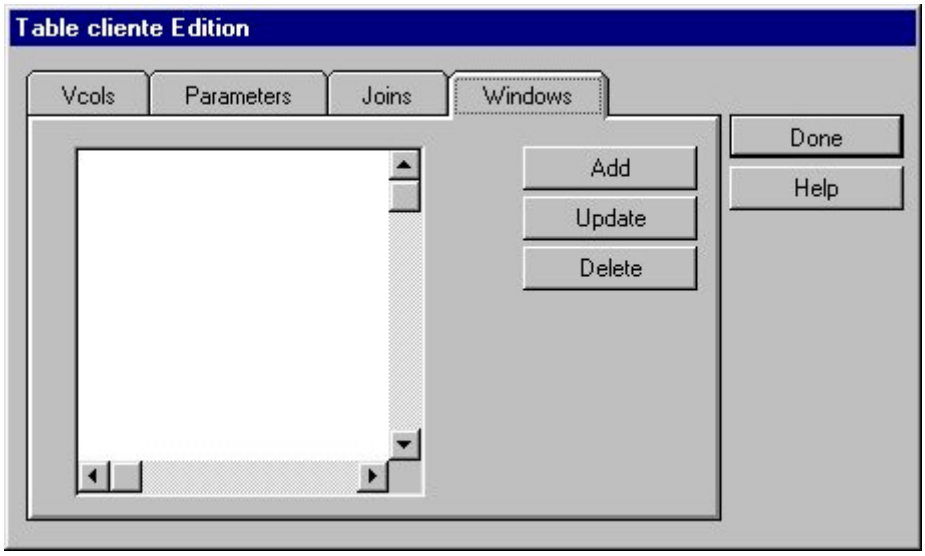


Figura 15.19. Ficha Windows

Añadir

Permite añadir una nueva window a la sección .VTABLE en curso:

1. Pulse el botón Add.
2. En el cuadro de diálogo de edición de Windows especifique las características de la nueva window.
3. Si desea que la acción realizada tome efecto, pulse el botón Ok, en caso contrario pulse Cancel.

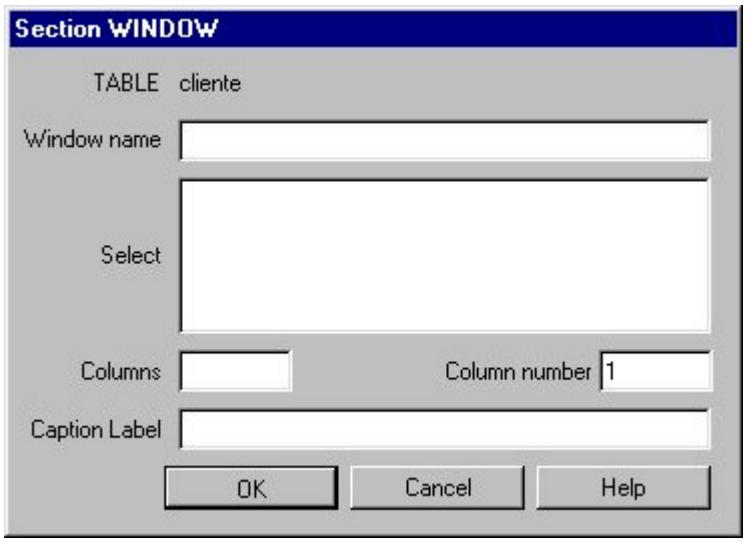


Figura 15.20. Cuadro de diálogo de la sección Window

Modificar

Permite modificar las características de una window:

1. Seleccione en la lista la window que desee modificar.
2. Pulse el botón Update.
3. Aparecerá el cuadro de diálogo de edición en el que se muestran las características de la window seleccionada.
4. Realice las modificaciones que crea necesarias.
5. Si desea que las modificaciones realizadas tomen efecto, pulse el botón Ok, en caso contrario pulse Cancel.

Borrar

Para borrar una window, selecciónela en la lista y pulse el botón Delete.

Borrar Tabla SQL

Permite borrar del ECD en curso la sección .VTABLE correspondiente a la tabla SQL seleccionada. Antes de proceder al borrado se pide confirmación al usuario.

Al borrar una sección .VTABLE se borran también sus correspondientes secciones .WINDOWS, .PARAMETERS, .JOINS y .VCOLS asociadas.

Para borrar una tabla SQL del ECD en curso:

1. Seleccione en la lista de tablas SQL la que desee borrar.
2. Pulse el botón Delete

Edición de Columna

Esta opción permite añadir o modificar una columna asociada a la sección .VTABLE en curso. Al ejecutarla se mostrará un cuadro de diálogo con los campos que se explican a continuación:

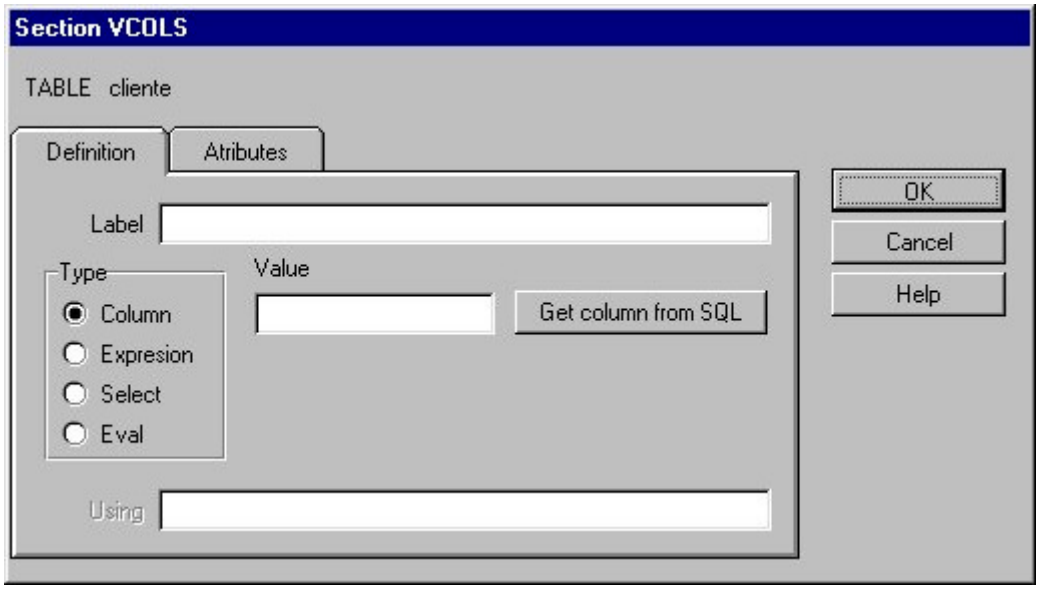


Figura 15.21. Cuadro de diálogo de la sección VCOLS (Ficha definición)

- *Ficha Definición*

Etiqueta — Nombre con el que el usuario va a identificar la columna. Este campo es obligatorio.

Tipo — Puede ser uno de los siguientes:

- Columna — Nombre de una columna real de la tabla definida en la sección .VTABLE, o bien de la tabla enlazada con un join si esta columna se especifica con la cláusula JOIN.
- Expresión — Expresión válida SQL con columnas de la tabla de la base de datos, que obtiene el valor deseado.
- Select — Instrucción SELECT que obtiene el valor deseado. Esta instrucción puede estar parametrizada usando la cláusula USING a continuación.
- Eval — Expresión válida que podrá incluir:
  - — Paréntesis.
  - — Los operadores +,-,* o /.
  - — Números.
  - — Literales entre comillas.
  - — Alias.

Valor — Campo de edición en el que obligatoriamente deberá indicarse el nombre de una columna de la tabla, una instrucción SELECT, una expresión o una EVAL.

Using — Lista de columnas (o números de columnas para .VTABLE del tipo SELECT) usadas como variables host de ejecución de la SELECT.

Obtener Columna del SQL — Permite importar una columna de la tabla SQL en curso.

- *Ficha Atributos*

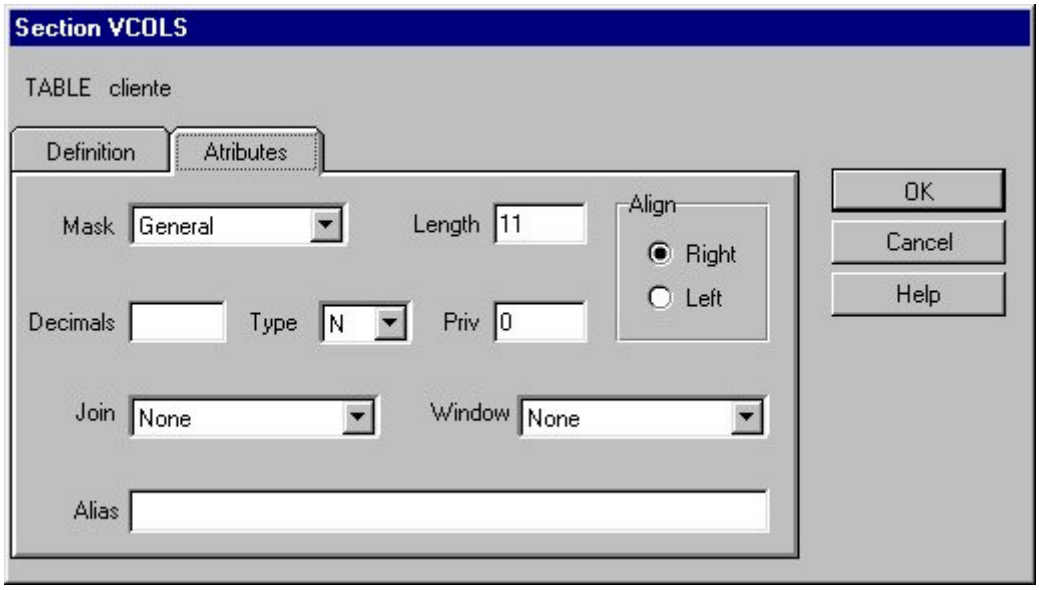


Figura 15.22. Ficha Atributos

Máscara	Tipo de máscara. Si no se especifica nada se asume el valor por defecto dependiendo de las variables de entorno utilizadas.
Longitud	Longitud de display de la columna. Debe ser mayor que 0.
Alineado	Ajuste a derecha o izquierda. Si no se especifica se asumirá derecha para numéricos e izquierda para el resto.
Decimales	Número de decimales para los campos numéricos. Si se especifica máscara, este número sólo tendrá validez en los formatos con decimales.
Tipo	Tipo de columna (alfanumérica, numérica, fecha u hora).
Privilegio	Número que indicará el privilegio de la columna. Un usuario con privilegio inferior al indicado no podrá visualizar la columna.
Join	Nombre del join a emplear (si la columna está definida en una tabla de enlace).
Window	Identificador de una de las ventanas definidas en la sección .WINDOWS. La columna de selección indicada en la cláusula GET deberá tener el mismo tipo de dato que la columna donde se recoge (columna). En la edición del campo Valor, en la opción del menú Edición/Condiciones del EasyReport, pulsando el botón Window se mostrará en pantalla la ventana de ayuda para poder recoger el valor.
Alias	Identificador con el que se podrá referenciar a la columna virtual en la definición de otras columnas virtuales dentro de una EVAL. El número máximo de caracteres que podrá tener el identificador es 50.

Edición de Parámetros

Esta opción permite añadir o modificar los parámetros asociados a la sección .VTABLE en curso. Al ejecutarla se mostrará un cuadro de diálogo que incluye los siguientes campos (Ver Figura 15.16):

Etiqueta	Etiqueta con la que se va a pedir el parámetro al usuario. Este campo es obligatorio.
Nombre	Nombre del parámetro utilizado en la sección .VTABLE. Este campo es obligatorio.
Valor por defecto	Valor numérico por defecto para tipo N o alfanumérico para tipos A, D y T.
Tipo	Tipo del parámetro: Alfanumérico (A), numérico (N), fecha (D) u hora (T).

Longitud	Longitud de display. Debe ser mayor que 0.
Ventana	Identificador de una de las ventanas definidas en la sección .WINDOWS. La columna de selección indicada en la cláusula GET deberá tener el mismo tipo de dato que el parámetro donde se recoge (param). Al ejecutar el informe, en el momento de edición de los parámetros, pulsando el botón se mostrará en pantalla la ventana de ayuda para poder recoger el valor.

Edición de Joins

Permite añadir o modificar un join asociado a la sección .VTABLE en curso. Al ejecutarla se mostrará un cuadro de diálogo con los siguientes campos (Ver figura28.18):

Nombre del Join	Nombre que identifica el enlace. Este campo es obligatorio
Usa Join Sí/No	Si esta casilla de verificación no esta activa indica que el enlace se hace desde la tabla definida en la sección .VTABLE.
Join usado	Este campo sólo estará disponible cuando la casilla de verificación Usa Join sí/no se encuentre activada. Indica que el enlace se hará desde la tabla previamente enlazada con el join que se seleccione en este campo.
Desde columna	Nombre de una columna con la que se hará el join. Esta columna debe pertenecer a la tabla definida en la sección .VTABLE o a la enlazada previamente, según se defina join usado o no. Este campo es obligatorio.
Tabla del Join	Nombre de la tabla enlazada. Este campo es obligatorio.
Hasta columna	Nombre de la columna en la tabla enlazada cuyo valor debe ser igual al de Desde columna. Se pueden definir enlaces multicolumna separando por comas las columnas de enlace. Este campo es obligatorio.

Edición de Windows

Permite añadir o modificar windows asociadas a la sección .VTABLE en curso. El cuadro de diálogo que aparece incluye los siguientes campos (Ver Figura 15. 20):

Nombre	Nombre que identifica a la ventana. Este campo es obligatorio.
Select	Instrucción SELECT que se mostrará en la ventana. Este campo es obligatorio.
Columnas	Número que identifica a las primeras columnas de la SELECT (las que aparecerán en pantalla).
Número de Columna	Número de la columna de la SELECT que se desea seleccionar. Este número puede ser mayor que el de las columnas que aparecen en pantalla.
Etiqueta	Título de la ventana.

Capítulo 16
TransTOOLs EasyReport

Contenidos

1. Introducción

La realización de informes, generalmente llamados *listados*, a partir de la información contenida en una base de datos, es una tarea que consume gran cantidad de recursos de los departamentos de sistemas de información, debido a las constantes peticiones de modificaciones que, si bien suelen ser sencillas, precisan de un proceso completo de análisis, diseño, implementación y pruebas.

EasyReport viene a resolver este problema, permitiendo al usuario final definir y obtener sus propios informes sin necesidad de conocer la estructura interna del sistema de información.

Esto es posible gracias a que EasyReport parte de un *Esquema Conceptual de Datos* (ECD), que será definido por el programador. Para esta tarea, Cosmos dispone de un generador automático de ECDs (*Data Scheme Editor*) que permite al programador seleccionar las tablas que desee de la base de datos, estableciendo múltiples vistas y restricciones sobre dicha base de datos.

EasyReport está especialmente diseñado para realizar, de forma sencilla, las siguientes tareas:

- Obtención dinámica de informes de la base de datos sin necesidad de crear un programa para ello.
- Facilitar el acceso a esta información para un usuario no experto.
- Ocultar la estructura interna de la base de datos (nombres de tablas y columnas, tipos de las columnas, relaciones entre tablas, columnas de enlace, etc.), de modo que el usuario de la herramienta pueda acceder fácilmente a toda la información contenida en la base de datos.
- Permitir a un usuario o grupo de usuarios el acceso restringido a la información contenida en la base de datos.
- Generar y catalogar fácilmente informes parametrizables para su posterior ejecución.

Una vez que se dispone del ECD, éste se pasa como parámetro al comando ceasyrep.

De forma esquemática, EasyReport permitirá:

- Crear informes a partir de la información contenida en el ECD.
- Ejecutar dichos informes hacia pantalla, impresora o fichero.
- Almacenar y catalogar los informes en disco.
- Cargar, modificar y ejecutar informes previamente generados.

Para ejecutar el EasyReport hay dos mecanismos:

- Mediante el icono

- Mediante el comando ceasyrep, cuya sintaxis es la siguiente:

ceasyrep [opciones] [nombre_ecd]

opciones :

Opción	Significado
-h	Muestra esta lista de opciones
-v	Muestra la versión del comando
-o informe	Ejecuta directamente el informe indicado
-f fichero	Envía la salida del informe al fichero indicado
-print	Envía la salida del informe a la impresora seleccionada por defecto en el Administrador de Impresión de Windows
-r programa	Envía la salida al programa indicado
-d base_datos	Ejecuta EasyReport sobre la base de datos indicada
-p parámetro	Envía el parámetro indicado al informe
-fp desde_pág	Ejecuta el informe desde la página indicada
-tp hasta_pág	Ejecuta el informe hasta la página indicada
-pl long_pág	Ajusta la longitud de página al valor indicado
-doc "texto"	Incluye el texto indicado como título de cabecera del informe. Dicho texto deberá ir entre comillas
-cat catálogo	Ejecuta sin ECD utilizando únicamente el catálogo.
-i privilegio	Ejecuta EasyReport concediendo al usuario un privilegio igual al indicado.
-err errfile	Escribe en el fichero errfile los errores detectados al leer el ECD
-var variable valor	Asigna valor a la variable de entorno indicada en variable
-env entorno	Carga el entorno definido en el fichero de configuración.
-con conexión	Carga el entorno de conexión definido en el fichero de configuración COSMOS.INI
-pas password	Asigna a la variable DBPASSWD el valor pasado en password
-tit “title”	Incluye el texto definido en title como título de la ventana principal del EasyReport.
-pf fichero	Fichero del que se leerán el resto de los parámetros.
nombre_ecd	Indica el nombre del ECD

2. Aspecto general de TransTOOLs EasyReport

Se compone de un menú despegable y una serie de iconos de las funciones mas comunes, asi como una barra de estado.

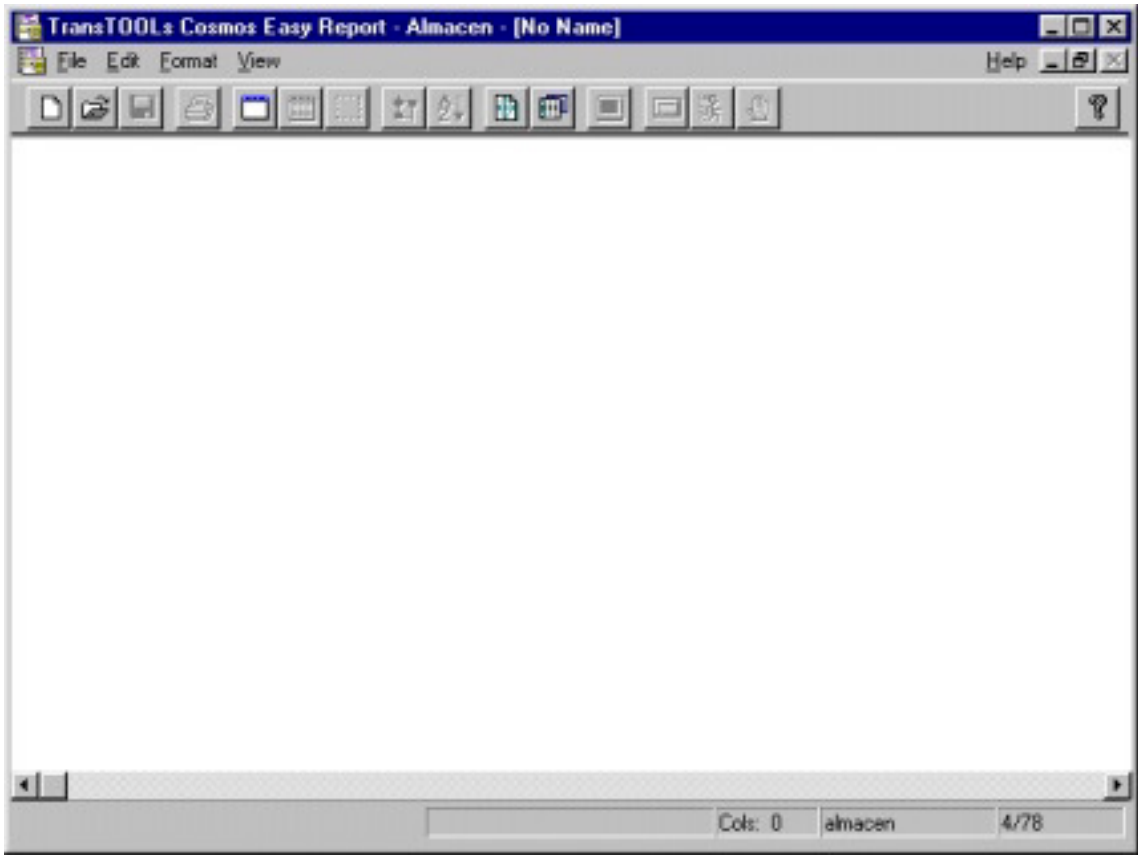


Figura 16.1. Aspecto general del EasyReport

3. Menú Fichero (File)

Las opciones incluidas en el menú Fichero permiten tanto el manejo de informes existentes como la generación de otros nuevos. Dichas opciones son las siguientes:

Nuevo	Crea un informe vacío.
Abrir	Carga un informe del catálogo previamente generado.
Salvar	Guarda el contenido del informe activo.
Salvar Como...	Permite guardar el informe activo con otro nombre.
Borrar	Permite eliminar un informe del catálogo.
Imprimir	Permite enviar el informe activo a la impresora seleccionada.
Especificar impresora	Esta opción permite cambiar la impresora empleada por defecto para salida del informe.
Seleccionar Base de Datos	Mediante esta opción podrá elegir la base de datos sobre la que desea construir el informe.
Cambiar Conexión	Esta opción permite establecer comunicación con otras bases de datos distintas de Cosmos (gracias a la tecnología MultiWay incluida en el producto), así como seleccionar el tipo de conexión (local o cliente-servidor).
Salir	Abandona EasyReport y retorna al punto desde el que fue invocado.

Crear un informe

La opción Nuevo del menú Fichero permite generar un informe nuevo. Al ejecutarla aparecerá un elemento de diálogo para seleccionar el tipo de formato que se desea aplicar al informe.

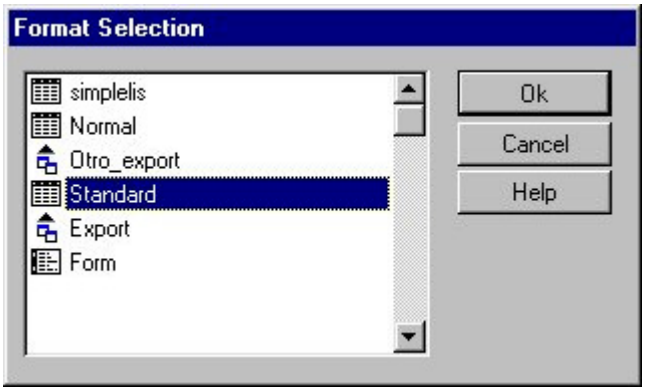


Figura 16.2. Cuadro de diálogo de selección de Formato

En la lista de formatos que aparece podremos desplazarnos utilizando las teclas [FLECHA ARRIBA] y [FLECHA ABAJO]. Asimismo, las teclas [INICIO] y [FIN] nos permitirán posicionarnos respectivamente en el primero y último elemento de la lista. Los 3 últimos formatos que aparecen son los empleados por defecto. Como puede observarse, en esta lista se muestra el nombre del formato y su tipo, el cual está representado por los siguientes iconos:

Para formato Standard. Presenta las filas de datos encolumnadas.

Para formato Export. Éste es el formato de descarga de una base de datos, y se utiliza para exportar los datos seleccionados en el informe a otra base de datos o aplicación.

Para formato Form. Cada columna se presenta en una ficha.

Los posibles formatos que aparecerán en este cuadro de diálogo dependerán de los que hubiese definido el administrador del sistema.

Para seleccionar un formato:

1. Seleccione en la lista el formato que desee.
2. Pulse el botón Ok.
3. Si pulsa Cancel el informe se obtendrá con el formato que tuviera definido por defecto el ECD. Si no hubiera definido ninguno se utilizará el formato estándar.

El nombre del informe y su descripción se definirán al guardarlo por primera vez.

Esta opción también se puede ejecutar pulsando el botón del menú de iconos.

Abrir un informe

La opción Abrir del menú Fichero permite abrir un informe previamente guardado (catalogado). Al ejecutarla aparecerá una ventana de selección con la lista del catálogo de informes existente. Dicha lista incluirá tanto los informes generados por el propio usuario como los disponibles en el catálogo general del sistema.

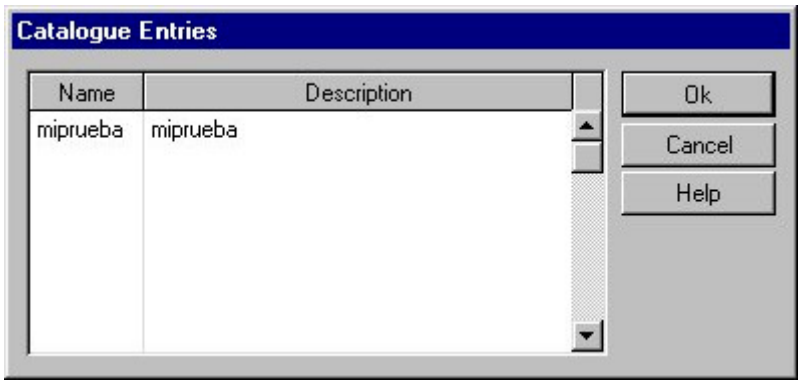


Figura 16.3. Cuadro de diálogo de selección de informe

Cada informe aparece identificado por su nombre y una pequeña descripción. Para seleccionar uno de ellos bastará con hacer doble click sobre él (o bien seleccionarlo y pulsar el botón Ok).

Si la descripción de un informe no cabe en la ventana de selección podremos verla completa utilizando las teclas de movimiento del cursor o bien ampliando las dimensiones de la ventana con el ratón (sitúese en cualquier vértice y arrastre el ratón hasta obtener el tamaño deseado).

Esta opción se puede ejecutar directamente pulsando el botón  del menú de iconos.

Guardar un informe

La opción Salvar del menú Fichero permite guardar el contenido de un informe. Si se trata de un informe ya existente guardará las modificaciones que hubieran podido realizarse sobre él. Por el contrario, si se trata de un informe nuevo el sistema pedirá el nombre que se le desea asignar y una breve descripción a través de un cuadro de diálogo idéntico al de la opción Salvar como.

Esta opción puede ejecutarse pulsando el botón  del menú de iconos.

Guardar un informe con otro nombre

La opción Salvar Como del menú Fichero permite guardar tanto un informe de nueva creación como otro ya existente asignándole otro nombre. El cuadro de diálogo que aparece es idéntico al de la opción Salvar cuando ésta se ejecuta por primera vez, solicitando el nombre que se desea asignar al informe y una breve descripción del mismo.

El botón 

presentará en pantalla la ventana de selección con la lista de informes y sus descripciones. Las teclas [FLECHA ARRIBA] y [FLECHA ABAJO] nos permitirán movernos a lo largo de la lista. Asimismo las teclas [INICIO] y [FIN] nos permitirán desplazarnos respectivamente al primero y al último elemento de la lista.

Si selecciona un informes del catálogo se tomará su nombre y descripción para el informe que desea grabar, sustituyendo el contenido del informe antiguo. Si, por el contrario, sólo desea comprobar los nombres ya asignados, salga de ese diálogo mediante el botón Ok.

Si el nombre asignado al informe coincidiese con el de algún otro ya existente, el sistema avisaría al usuario presentando un mensaje en pantalla.

A través de esta opción el usuario podrá elegir entre:

- Guardar el informe con el mismo nombre, con lo cual se grabarían las modificaciones que hubieran podido realizarse sobre él (equivalente a la opción Salvar). En este caso el sistema pediría confirmación avisando de que se trata de un informe ya existente que se va a sobreescribir.
- Asignar un nombre distinto al actual, con lo cual estaríamos generando un informe nuevo.
- Cancelar la operación.

Borrar un informe

La opción Borrar del menú Fichero permite borrar un informe existente en el catálogo. Al ejecutarla aparecerá en pantalla la lista de informes guardados para seleccionar aquel que se desea borrar. Antes de proceder al borrado se pedirá conformidad al usuario.

Imprimir un informe

La opción Imprimir del menú Fichero envía el informe activo a la impresora seleccionada a través del Administrador de Impresión de Windows.

Esta opción también se puede ejecutar directamente pulsando el botón  del menú de iconos.

Especificar impresora

La opción Seleccionar Impresora del menú Fichero permite seleccionar una impresora distinta de la indicada por defecto en el Administrador de Impresión de Windows.

Las posibles impresoras a emplear dependerán de las que se hubiesen dado de alta en el sistema a través del Panel de Control de Windows.

La utilización de una impresora u otra dependerá de las características del informe en cuanto al empleo de márgenes, tipos de letra, etc. Por esta razón, es aconsejable seleccionar la impresora de salida antes de generar un informe con el fin de evitar desagradables sorpresas a la hora de imprimirlo.

Seleccionar la Base de Datos

La opción Seleccionar Base de Datos del menú Fichero permite elegir la base de datos sobre la que desea trabajar. Si al ejecutar el informe el administrador no la hubiese dejado seleccionada por defecto, el sistema pedirá automáticamente el nombre de la base de datos.

Al ejecutar esta opción se muestra un cuadro de diálogo el que se indica el nombre de la base de datos en curso. Escriba el nombre de la nueva base de datos que desea seleccionar (este nombre deberá ser un literal alfanumérico de 10 caracteres de longitud como máximo).

Si la conexión definida en el entorno es local aparecerán los nombres de las bases de datos disponibles, mientras que si la conexión es remota sólo aparecerá la base de datos previamente seleccionada.

Cambiar la Conexión

La opción Cambiar Conexión del menú Fichero permite seleccionar una conexión distinta de la indicada en la línea de comando del EasyReport o de la definida en el entorno.

Mediante esta opción se podrá establecer una conexión local o remota (cliente-servidor), pudiendo en este último caso acceder a otros gestores de bases de datos.

Al ejecutarla aparecerá un cuadro de diálogo con una lista desplegable, que incluye las conexiones definidas en el fichero de configuración COSMOS.INI, para seleccionar la conexión deseada. Si el entorno especificado para la conexión remota seleccionada no tiene definida la variable de entorno DBPASSWD y/o DBUSER automáticamente se pedirá la contraseña y/o el nombre de usuario para poder establecer la conexión en la siguiente ejecución.

Salir de EasyReport

La opción Salir del menú Fichero cierra el EasyReport y retorna al sistema operativo o al comando desde el que fue invocada la aplicación.

4. Menú Edicción (Edit)

Las opciones del menú Edición permiten seleccionar la tabla sobre la cual se va a elaborar el informe, incluyendo el tratamiento de columnas y grupos, así como la definición de las condiciones que deberán cumplir dichas columnas y los criterios de ordenación deseados. Dichas opciones son las siguientes:

Tabla Permite seleccionar la tabla sobre la que se desea realizar el informe.

Columnas Permite seleccionar columnas de la tabla elegida en la opción anterior y definir sus atributos para el informe.

Grupos Permite agregar grupos de ruptura y definir sus características.

Condiciones Permite introducir condiciones sobre las columnas de la tabla seleccionada utilizando diferentes tipos de operaciones.

Orden Permite introducir condiciones de ordenación sobre las columnas de la tabla seleccionada para el informe.

Seleccionar tabla para el informe

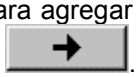
La opción Tabla del menú Edición permite seleccionar la tabla sobre la cual se desea realizar el informe y sólo se encontrará activa cuando el informe es de nueva creación o en caso de no haberle definido aún sus columnas. Al ejecutarla se mostrará un cuadro de diálogo con la lista de tablas a las que se tiene acceso desde el ECD en curso para seleccionar una de ellas.

Esta opción también se puede ejecutar pulsando el botón  del menú de iconos.

Seleccionar columnas para el informe

La opción Columnas del menú Edición permite seleccionar las columnas de la tabla en curso que se desea que aparezcan en el informe. Al ejecutarla aparecerá un cuadro de diálogo con dos ventanas: una en la que se muestran todas las columnas a las que se tiene acceso (las definidas en el ECD para la tabla seleccionada), y otra que recoge la lista de columnas seleccionadas para el informe.

Esta opción se puede ejecutar también pulsando el botón  del menú de iconos.

Para agregar una columna al informe selecciónela en la ventana de la izquierda y pulse el botón  . Esta operación también podrá realizarla arrastrándola directamente con el ratón a la ventana de la derecha o haciendo doble click sobre la columna en cuestión.

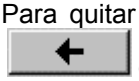
Si desea añadir todas las columnas de una vez pulse el botón  .

Para insertar una columna en una posición determinada proceda de la siguiente manera:

1. Seleccione en la ventana de la izquierda la columna que desee agregar al informe.
2. A continuación, en la ventana de la derecha, seleccione la columna delante de la cual quiere insertar la columna seleccionada en la ventana de la izquierda.
3. Pulse el botón  .

A medida que se van eligiendo columnas, éstas se muestran automáticamente en el formulario de la pantalla detrás del cuadro de diálogo.

Por su parte, en la barra de estado situada al pie de la pantalla se irá mostrando tanto el número de columnas incluidas en el informe como el ancho de caracteres ocupados sobre el máximo posible. En caso de sobrepasar este límite el sistema presentará un mensaje avisando de esta circunstancia, aunque no impedirá que se pueda continuar agregando columnas. Para solucionar este problema, ejecute la opción Diseño del menú Formato e indique el ancho correcto de caracteres de su informe en el campo Longitud de Línea, o seleccione otro tamaño de página a través de la opción Seleccionar Impresora del menú Fichero.

Para quitar una columna del informe selecciónela en la lista de la derecha y pulse el botón  o bien arrástrela directamente a la lista de la izquierda.

Para quitar todas las columnas de la lista de la derecha pulse el botón  .

El botón amplía o reduce el cuadro de diálogo mostrando o no la lista de la derecha. En este caso podrá agregar columnas al informe con sólo seleccionarlas y arrastrarlas con el ratón a la pantalla principal del documento.

Seleccionando una columna de la lista de la derecha y pulsando el botón se abrirá el cuadro de diálogo Edición de Columna en el que se mostrarán los atributos de dicha columna en el informe.

Definir formato de salida para las columnas

En este cuadro de diálogo se podrán editar y modificar los atributos de las columnas elegidas para un informe (su título, longitud y número de decimales).

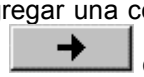
Asimismo, se podrán indicar los valores agregados que se desean calcular de esa columna (cuenta, suma, media, máximo y mínimo), así como modificar la máscara de impresión y las etiquetas con las que desea que aparezcan los valores agregados de aquellas columnas para los que se hayan elegido.

Los campos que aparecen en el cuadro de diálogo son los siguientes:

Título	Encabezamiento de la columna que aparecerá en el informe.
Longitud	Número máximo de caracteres de la columna.
Decimales	Número de decimales para columnas de tipo numérico.
Máscara	Tipo de máscara que se utilizará en el informe para mostrar el valor de la columna.
Alineamiento	Tipo de alineación de la columna (a izquierda o derecha).
Totales	Valores agregados que desea calcular y mostrar en las rupturas de grupo y al final del informe para la columna.
Etiquetas	Títulos que aparecerán en el informe para los totales.

Agregar grupos de ruptura

Mediante la opción Grupos del menú Edición se podrán agregar grupos de ruptura al informe y definir las características y comportamiento de las columnas que lo componen. Al ejecutarla se mostrará la lista de columnas de la tabla seleccionada para ir agregando los grupos por los que se desea una ruptura en el listado. El elemento de diálogo que aparece es prácticamente igual al de la opción Columnas comentada anteriormente, mostrando en la ventana de la izquierda la lista completa de las columnas a las que se tiene acceso (las reales de la tabla seleccionada y las obtenidas a partir de un join) y en la de la derecha las columnas seleccionadas como grupo de ruptura para el informe.

Para agregar una columna al grupo de ruptura selecciónela en la lista de la izquierda y pulse le botón  o bien arrástrela con el ratón a la lista de la derecha o directamente sobre el informe.

Si desea agregar todas las columnas de una sola vez pulse el botón  .

Para insertar una columna en una posición determinada proceda de la siguiente manera:

1. Seleccione en la ventana de la izquierda la columna que desee agregar.
2. A continuación, en la ventana de la derecha, seleccione la columna delante de la cual quiere insertar la columna seleccionada en la ventana de la izquierda.
3. Pulse el botón  .

A medida que vaya agregando grupos de ruptura, éstos irán apareciendo en el formulario de la pantalla.

Para quitar una columna del grupo de ruptura selecciónela en la lista de la derecha y pulse el botón  o bien arrástrela directamente con el ratón a la lista de la izquierda.

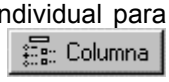
Para quitar todas las columnas de la lista de la derecha pulse el botón  .

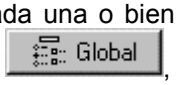
El botón  amplía o reduce el cuadro de diálogo mostrando o no la lista de la derecha. En este caso podrá agregar grupos de ruptura con sólo seleccionar las columnas y arrastrarlas con el ratón a la pantalla principal del documento.

Los botones Columna y Global permitirán respectivamente definir el comportamiento de las columnas de manera individual o globalmente a todo el grupo de ruptura.

Esta opción también se puede ejecutar pulsando el botón  del menú de iconos.

Definir características de un grupo de ruptura

El comportamiento de las columnas que forman parte de un grupo se podrá modificar bien de manera individual para cada una o bien de forma global para todas ellas. Para ello, utilice los botones 

y 

, respectivamente.

En ambos casos podrá decidir, además, si se desea totalizar, saltar página, subtitular y que la ruptura sea visible o no. Asimismo, se podrá modificar la etiqueta del título del grupo en aquellos grupos seleccionados.

Al pulsar cualquiera de los botones citados aparecerá un elemento de diálogo que incluye los siguientes campos:

Título del grupo	Este campo sólo aparece si se pulsa el botón Columna, y en el se muestra el título que presentará el grupo.
Totalizar	Casilla de verificación que permite indicar si los valores agregados se mostrarán en el informe después de cada ruptura del grupo.
Salto de página	Inserta un salto de página después de cada grupo de ruptura.
Subtitular	Activando esta casilla se mostrarán las etiquetas de las columnas para todos los grupos de ruptura de la página; en caso contrario sólo se mostrarán al principio de cada página.
Invisible	Si se activa esta casilla no se mostrarán los valores de los grupos de ruptura ni se producirán saltos de página en la ruptura del grupo.

Agregar condiciones sobre columnas

La opción Conditions del menú Edit permite introducir condiciones sobre columnas de la tabla seleccionada para realizar el informe. Al ejecutarla se mostrará el cuadro de diálogo Conditions con los siguientes campos:

Columna	Lista desplegable que contiene las columnas de la tabla seleccionada.
Operación	Lista de operaciones que se podrán realizar con la columna elegida.
Valor	Valor que se desea asignar a la columna en la operación indicada. Si la operación elegida es in, not in, en rangos y no en rangos se podrán indicar diferentes valores separados por un retorno de carro.
Botón	Este botón solamente estará visible si la operación elegida es in, not in, en rangos y no en rangos. Al pulsar este botón se muestra un cuadro de diálogo que permite seleccionar un fichero que tiene la lista de valores que se desea cargar en la lista Valor. En este fichero los diferentes valores deberán ir separados por un retorno de carro.

Una vez cumplimentados los campos anteriores, la condición se mostrará en la ventana del cuadro de diálogo con los valores definidos pulsando el botón Añadir.

Para eliminar una condición selecciónela en la ventana y pulse el botón Borrar. Asegúrese de que desea eliminar la condición antes de borrarla, ya que en este caso no se pide conformidad al usuario.

Si desea editar una condición de las que se muestran en la ventana haga doble click sobre ella.

El botón Modificar le permitirá sustituir la condición seleccionada en la ventana por la que estuviese definida en ese momento en los campos Columna, Operación y Valor.

Todas las condiciones de la lista se comportarán como restricciones AND en la ejecución.

Cuando la lista de condiciones esté completa pulse en el botón Aceptar.

Si la columna tuviese definida previamente una lista de valores aparecerá un botón denominado Window para elegir uno de esos valores.

Esta opción también se puede ejecutar pulsando el botón  del menú de iconos.

Operaciones

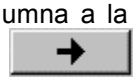
Las posibles operaciones que pueden asignarse a una condición son las siguientes:

Operación	Significado	
=	Igual	
<>	Distinto	
>	Mayor que	
<	Menor que	
>=	Mayor o igual que	
<=	Menor o igual que	
(alguno) in	Se hace cierta si la columna es igual a alguno de los valores incluidos en la lista de valores	
(ninguno) not in	Se hace cierta si la columna es distinta de todos los valores incluidos en la lista de valores	
no es nulo (is not null)	Se hace cierta si la columna tiene un valor no nulo	
es nulo (is null)	Se hace cierta si la columna tiene un valor nulo	
Contiene(matches)	(Sólo en columnas alfanuméricas). Se hace cierta si la columna se ajusta al patrón definido en Valor, que puede contener varios metacaracteres con un significado especial	
	*	Indica cero o más caracteres en la posición indicada. (Por ejemplo, empieza por T se indicaría: T*)
	?	Indica cualquier carácter (uno solo) en la posición donde aparezca
	[lista]	Indica que cualquiera de los caracteres incluidos en la lista puede aparecer en esa posición. Se puede indicar un rango de caracteres separando el inicial y el final por medio de un guión. Si el primer carácter de lista es ^ indica que los caracteres o rangos no pueden aparecer en esa posición
	\	Indica que el carácter al que precede no debe ser considerado como especial, sino como un carácter más dentro del valor
como(like)	(Sólo columnas alfanuméricas). Se hace cierta si la columna se ajusta al patrón definido en Valor, que puede contener varios metacaracteres con un significado especial	
	%	Indica cero o más caracteres en la posición donde aparezca
	_	(Subrayado) Indica cualquier carácter (uno solo) en la posición indicada

Operación	Significado
en rango	Solo puede aplicarse a las columnas de la tabla maestra. Se hace cierta cuando el valor de la columna está contenido en alguno de los rangos o valores definido en la lista
no en rango	Solo puede aplicarse a las columnas de la tabla maestra. Se hace cierta cuando el valor de la columna no está contenido en ninguno de los rangos o valores definido en la lista

Definir condiciones de ordenación sobre columnas

La opción Ordenar del menú Edición permite introducir condiciones de ordenación sobre las columnas principales de la tabla seleccionada. Al ejecutarla se mostrará un cuadro de diálogo con dos ventanas; la de la izquierda contendrá la lista de columnas reales de la tabla seleccionada, mientras que la de la derecha incluirá las columnas por las que se ordenará el informe.

Para agregar una columna a la ordenación selecciónela en la lista de la izquierda. A continuación, pulse el botón  o haga doble click sobre ella o arrástrela directamente con el ratón a la lista de la derecha.

Si desea agregar todas las columnas de una vez pulse el botón  .

Para insertar una columna en una posición determinada proceda de la siguiente manera:

1. Seleccione en la ventana de la izquierda la columna que desee agregar.
2. A continuación, en la ventana de la derecha, seleccione la columna delante de la cual quiere insertar la columna seleccionada en la ventana de la izquierda.
3. Pulse el botón  .

Para quitar una columna de la condición de ordenación, selecciónela en la lista de la derecha y pulse el botón  o bien arrástrela con el ratón a la lista de la izquierda.

Para quitar todas las columnas de la lista de la derecha pulse el botón  .

El tipo de ordenación de cada columna puede ser ascendente (por defecto) o descendente. Esto se refleja mediante ↓ para orden descendente y ↑ para ascendente.

Para cambiar el tipo de ordenación de ascendente a descendente y viceversa seleccione la columna en la lista de la derecha y pulse en el botón 

o haga doble click sobre ella.

Esta opción también se puede ejecutar pulsando el botón  del menú de iconos.

5. Menú Format

Las opciones incluidas en el menú Formato permiten establecer las características de presentación de los informes. Dichas opciones son las siguientes:

Diseño	Permite cambiar los parámetros de impresión del informe.
Cambiar Formato	Permite seleccionar el tipo de formato (Listing, Export o Form) que se desea para el informe.
Fuente	Permite seleccionar el tipo de letra (fuente, estilo y tamaño) que se desea emplear en el informe.

Diseñar un formato

La opción Diseño del menú Formato permite cambiar los parámetros de impresión de un informe (establecer cabeceras y pies de páginas, márgenes, etc.).

Los campos que aparecerán en el elemento de diálogo que se muestra al ejecutarla dependerán del formato elegido para el informe.

Para formato LISTING

Campo	Significado	
Tam. página	Permite definir el tamaño de la página que se obtendrá a la hora de imprimir, pudiendo elegir entre las siguientes opciones	
	Definida	Permite especificar la longitud y el ancho de página deseados
	Impresora	Ésta es la opción por defecto. Utiliza los valores idóneos según la impresora seleccionada
	No cortar	Imprime el informe como una sola página
Ancho de página	Permite establecer el número de caracteres por línea del listado. Esta opción sólo estará activa cuando se seleccione Definida o No cortar en el campo Tam. Página, pudiendo entonces indicar el ancho de línea deseado o pulsar el botón para incrementar su valor o para disminuirlo	
Longitud de página	Número de líneas por página. Esta opción sólo estará activa cuando se seleccione Definida en el campo Tam. página, pudiendo entonces indicar la longitud deseada o pulsar el botón para incrementar su valor o para disminuirlo	
Separación columnas	Espacio de separación entre columnas del listado expresado en número de caracteres	
Márgenes	Márgenes superior, inferior (expresados en número de líneas), izquierdo y derecho (expresados en número de caracteres) que se emplearán en el listado	
Mostrar Títulos	Permite indicar si se desea o no imprimir las cabeceras de las columnas	
Sólo Totales	Permite obtener únicamente los totales de un informe, sin mostrar las líneas que lo conforman	
Cortar textos	Mediante esta casilla podremos indicar si un texto determinado deberá aparecer en una o varias líneas, dependiendo de que quepa o no en la longitud indicada en el ancho de página	
Título	Campo de edición donde deberá indicarse el título que desea asignar al listado	
Cabecera	Campo de edición donde deberá indicarse el texto que se desea emplear como cabecera del listado	
Pie de página	Pie de página del listado	
Desde/Hasta página	Rango de páginas que se desean listar	
Previos	Restaura los valores establecidos inicialmente	

Para formato EXPORT

Campo	Significado
Delimitador	Carácter separador de campos. Si desea utilizar como delimitador un tabulador escriba \t
Imprimir Cabecera	Salida con o sin etiquetas de columnas
Imprimir Subrayado	Salida con cabeceras subrayadas o no
Literales entre comillas	Si se desea que los literales de los campos aparezcan o no entre comillas
Delimitador al final	Si se desea o no el carácter separador al final de cada línea

Para formato FORM:

Campo	Significado	
Tam. página	Permite definir el tamaño de la página que se obtendrá a la hora de imprimir, pudiendo elegir entre las siguientes opciones:	
	Definida	Permite especificar la longitud y el ancho de página deseados
	Impresora	Ésta es la opción por defecto. Utiliza los valores idóneos según la impresora seleccionada
	No cortar	Imprime el informe como una sola página
Ancho de página	Permite establecer el número de caracteres por línea del listado. Esta opción sólo estará activa cuando se seleccione Definida o No cortar en el campo Tam. Página, pudiendo entonces indicar el ancho de línea deseado o pulsar el botón para incrementar su valor o para disminuirlo	
Longitud de página	Número de líneas por página. Esta opción sólo estará activa cuando se seleccione Definida en el campo Tam. Página, pudiendo entonces indicar la longitud deseada o pulsar el botón para incrementar su valor o para disminuirlo	
Márgenes	Márgenes superior, inferior (expresados en número de líneas), izquierdo y derecho (expresados en número de caracteres) que se emplearán en el listado	
Imprimir Títulos	Permite indicar si se desea imprimir o no las cabeceras de las columnas	
Desde/Hasta página	Rango de páginas que se desean listar	
Previos	Restaura los valores establecidos inicialmente	

Esta opción también se puede ejecutar pulsando el botón  del menú de iconos.

Caracteres Especiales

En la descripción de la cabecera y del pie del listado, además de un texto fijo se podrán añadir ciertas variables del sistema cuyos códigos y significados son los siguientes:

Códigos	Significado
$T	Título del listado
$d	Fecha actual
$t	Hora actual
$p	Página en curso
$Pn	Valor del parámetro n de la tabla en curso, siendo n el índice dentro de la lista de parámetros en ejecución

En una cabecera o pie se podrá especificar también que el texto deberá aparecer en varias líneas. Para ello se empleará como indicador de salto de línea el carácter | (ASCII 124).

Ejemplo: Informe|de 1990 aparecería como:

Informe
de 1990

El carácter | se podrá utilizar igualmente como indicador de salto de línea en el título de una columna.

Asimismo, en una cabecera o pie se podrá especificar también para cada línea si una parte de ésta deberá alinearse a la izquierda, a la derecha o centrarse. Para ello, cada parte en cuestión deberá ir precedida de las siguientes claves:

Claves	Significado
<-	Ajustar a la izquierda
->	Ajustar a la derecha
<>	Centrar

Los parámetros de longitud de página, separación y márgenes se introducen en unidades de carácter, pudiendo definir en un informe tipo estándar la longitud de la página como 0 (No cortar) si no se desea la separación en páginas.

Cambiar el formato de un informe

Mediante la opción Cambiar formato del menú Formato podremos cambiar el formato del informe que hubiésemos definido al crearlo mediante la opción Nuevo del menú Fichero, siendo idéntico al de ésta el elemento de diálogo que se muestra al ejecutarla.

Los 3 últimos formatos que aparecen en la lista del cuadro de diálogo son los empleados por defecto. Como puede observarse, en esta lista se muestra el nombre del formato y su tipo, el cual está representado por los siguientes iconos:

 Para formato Standard. Presenta las filas de datos encolumnadas.

 Para formato Export. Éste es el formato de descarga de una base de datos, y se utiliza para exportar los datos seleccionados en el informe a otra base de datos o aplicación.

 Para formato Form. Cada columna se presenta en una ficha.

Los posibles formatos que aparecerán en este cuadro de diálogo dependerán de los que hubiese definido el administrador del sistema.

Para seleccionar un formato:

1. Seleccione en la lista el formato que desee.
2. Pulse el botón Ok.
3. Si pulsa Cancel el informe se obtendrá con el formato que tuviera definido por defecto el ECD. Si no hubiera definido ninguno se utilizará el formato estándar.

Esta opción también se puede ejecutar pulsando el botón  del menú de iconos.

Cambiar el tipo de fuente de un informe

La opción Fuente del menú Formato permite seleccionar la fuente con la que se mostrará e imprimirá el informe, incluyendo su tipo (Courier, Times, etc.), estilo (normal, cursiva, negrita, etc.) y tamaño.

Dependiendo de la fuente empleada y del tamaño de papel elegido en la opción Seleccionar Impresora del menú Fichero, el informe tendrá una longitud u otra según el ajuste de página establecido en la opción Diseño.

Las fuentes que se presentan en este diálogo son de tipo fijo, es decir, todos los caracteres se imprimirán con el mismo ancho.

6. Menú View

El menú View incluye varias opciones relativas a la ejecución de informes. Dichas opciones son las siguientes:

A Pantalla	Permite obtener el resultado del informe en la pantalla.
A Comando	Permite ejecutar un programa Windows enviándole como parámetro el resultado de la ejecución del informe.
A Fichero	Permite obtener el resultado del un informe en un fichero.
Cerrar Ejecución	Detiene la ejecución del informe.
Simulación	Permite simular la ejecución de un informe.

Cuando se hayan definido parámetros en un Esquema Conceptual de datos al seleccionar cualquiera de las opciones que conllevan su ejecución se presentará un elemento de diálogo al usuario para establecer dichos parámetros.

Ejecución de informes hacia la pantalla

La opción to screen del menú View ejecuta el informe hacia pantalla. Una vez en ejecución podrá utilizar las teclas de movimiento del cursor y las barras de desplazamiento para revisar el resultado de la ejecución.

La ejecución del informe se realiza a medida que desplace el informe hacia abajo.

Si desea parar la ejecución podrá hacerlo mediante la opción Close Running de este mismo menú o pulsando el botón  del menú de iconos.

Para ver el resultado del informe o ir al final del mismo, pinche con el ratón en el cuadro de desplazamiento vertical y arrástrelo hacia abajo. El cuadro de desplazamiento irá indicando la evaluación mientras en la barra de estado de la parte inferior de la pantalla aparecerá el mensaje *Ejecutando...* o el número de página o de línea si está a la espera de la orden del usuario.

Para interrumpir una orden de ir al final del informe bastará con pulsar el ratón sobre el informe o la barra espaciadora.

Durante la ejecución del informe se desactivan las opciones del menú, volviendo a activarse una vez alcanzado el final.

Mientras dura la ejecución podrá ver la ventana de definición del informe pulsando [CTRL]+[F6].

Esta opción también se puede ejecutar pulsando el botón  del menú de iconos.

Ejecución de informes hacia comando

La opción To Command del menú View permite ejecutar un programa Windows enviándole como parámetro el resultado de la ejecución del informe.

El comando que se ejecuta y el nombre de la opción de menú dependerá del formato seleccionado y deberá haber sido definido por el programador en el ECD. Esta opción estará inhabilitada cuando no exista un comando asociado.

Dependiendo de la definición de esta opción le podrá ser requerido al usuario un nombre de fichero, o bien se creará automáticamente.

Ejecución de informes hacia fichero

La opción To File del menú View guarda el resultado de la ejecución del informe en un fichero. Al ejecutarla aparecerá un elemento de diálogo para indicar el nombre del fichero y su directorio de ubicación.

Utilice esta opción cuando desee una salida en modo carácter.

Esta opción también se puede ejecutar pulsando el botón  del menú de iconos.

Esta opción es equivalente a ejecutar el informe hacia un fichero, en este caso deberá ser borrado el fichero creado por el usuario.

Cerrar Ejecución

La opción Close Running del menú View permite detener la ejecución de un informe.

Esta opción también se puede ejecutar pulsando el botón  del menú de iconos.

Simulación de Informes

La opción Simulation del menú View simula la ejecución de un listado sin generar ningún informe. Sólo indicará el número de páginas y líneas que se obtendrían, mostrando su resultado en una ventana.

Cuando se seleccione la opción de impresión o la de ejecución a fichero también se mostrará este diálogo a la vez que se genera la salida.

Esta opción también se puede ejecutar pulsando el botón  del menú de iconos.

Capítulo 17
El Depurador de programas (Debugger)

Contenidos

1. Introducción

Cosmos proporciona una herramienta para ayudarle a analizar la forma en que opera su aplicación. Esta herramienta es útil para localizar el origen de errores lógicos, pero también puede utilizarse para probar cambios en la aplicación o para aprender cómo funcionan otras aplicaciones.

La depuración de programas, así como la optimización de código, es una de las fases más costosas en cualquier desarrollo y donde se precisa un enfoque más sistemático, y por ello en la que más se advierte la carencia de herramientas de uso sencillo que permitan un control sobre la ejecución de programas.

Un depurador es una herramienta que permite ejecutar un programa bajo control directo del programador. He aquí algunas de las características más relevantes que nos debe permitir una herramienta de depuración.

- Ver tanto el programa fuente editado con indicación de la siguiente instrucción a ejecutar como los resultados que produce por pantalla dicha ejecución. Esto supone que podemos leer las instrucciones del programa en la secuencia real de ejecución de las mismas y simultánea y alternativamente los resultados que esta secuencia produce, es decir verificamos el flujo del programa.
- Ver/modificar los valores contenidos en los objetos de cada módulo.
- Conocer las secuencias de llamadas entre distintos bloques (programas, funciones, menús, etc.) en cada punto concreto del programa (anidamientos).
- Definir paradas en la ejecución en base al alcanzar una determinada línea en el fuente.

2. Tipos de Errores

Para comprender lo útil que resulta depurar considere los tres tipos de errores que puede encontrar:

Errores de compilación

Estos errores son resultado de una construcción incorrecta del código. Si escribe una palabra clave de forma incorrecta, omite alguna puntuación necesaria, como utilizar una instrucción while sin su correspondiente do, Cosmos detectará esos errores cuando compile la aplicación.

Errores lógicos

Los errores lógicos son errores de diseño o implementación y ocurren cuando una aplicación no funciona en la forma que se pretende. Una aplicación puede tener la sintaxis del código correcta, ejecutarse sin realizar operaciones que no sean válidas, y aún así producir resultados incorrectos. Se produce un error lógico por ejemplo cuando un objeto contiene un valor incorrecto.

Los errores lógicos son los errores más difíciles de localizar, por que se producen en lugares inesperados. Para garantizar que un programa funcione tal y como fue diseñado, se deben comprobar a fondo todos sus aspectos. El depurador ayuda a localizar errores lógicos permitiendo que se examinen los valores de los objetos del programa.

Errores en tiempo de ejecución

Los errores en tiempo de ejecución ocurren cuando una instrucción intenta realizar una operación que es imposible ejecutar. Es posible, que el programa trate de abrir un archivo que no existe o intente dividir un número por cero. El sistema operativo detecta estos errores e interrumpe la ejecución del programa cuando localiza un error de este tipo.

El depurador puede ayudarle a realizar un seguimiento rápido de este tipo de errores. Con el depurador puede ejecutar un programa hasta un punto determinado y, a partir de ahí, empezar a ejecutar el programa línea por línea, examinando su comportamiento en cada paso. Cuando ejecute la sentencia que hace fallar el programa, habrá encontrado el error. Entonces, puede corregir el código fuente, volver a compilar y reanudar la comprobación.

3. Opciones del Depurador

Iniciar la depuración

Si encuentra errores en un programa, puede iniciar una sesión de depuración ejecutando el programa bajo el control del depurador.

Para iniciar la depuración de un programa:

1. El programa debe estar compilado sin errores.
2. En la ventana del proyecto, seleccione y edite el módulo que desea compilar.
3. Ejecute la opción Debug del menú Tools en el menú principal del editor visual o pulse el botón del menú de iconos.

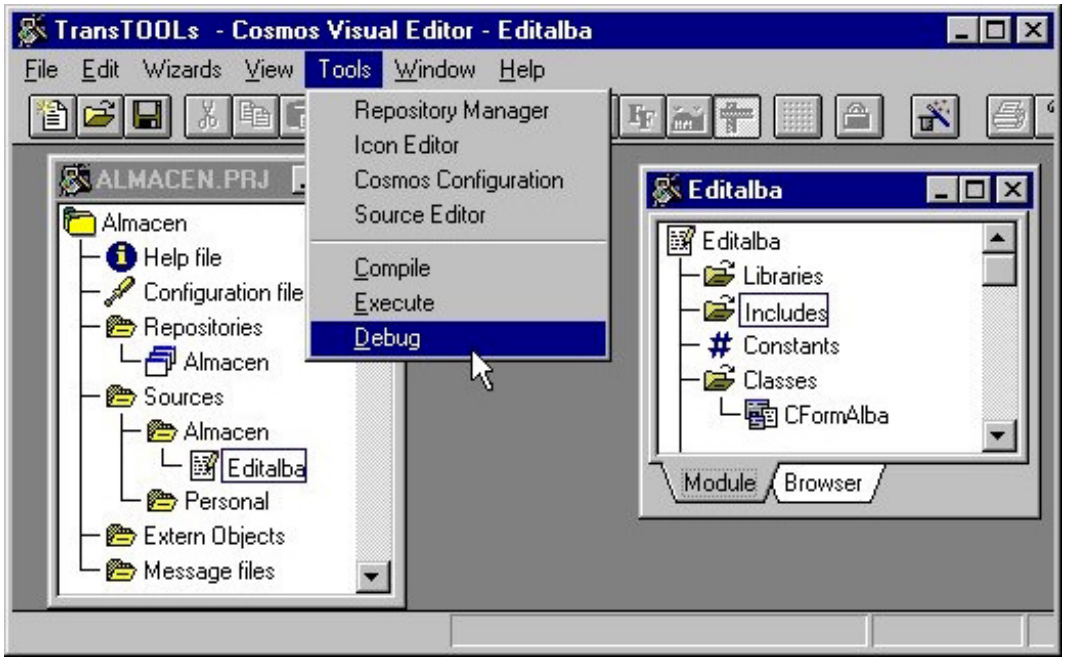


Figura 17.1 Pasos 2 y 3 de la depuración.

4. Se muestra la ventana del depurador cuyo aspecto es el siguiente:

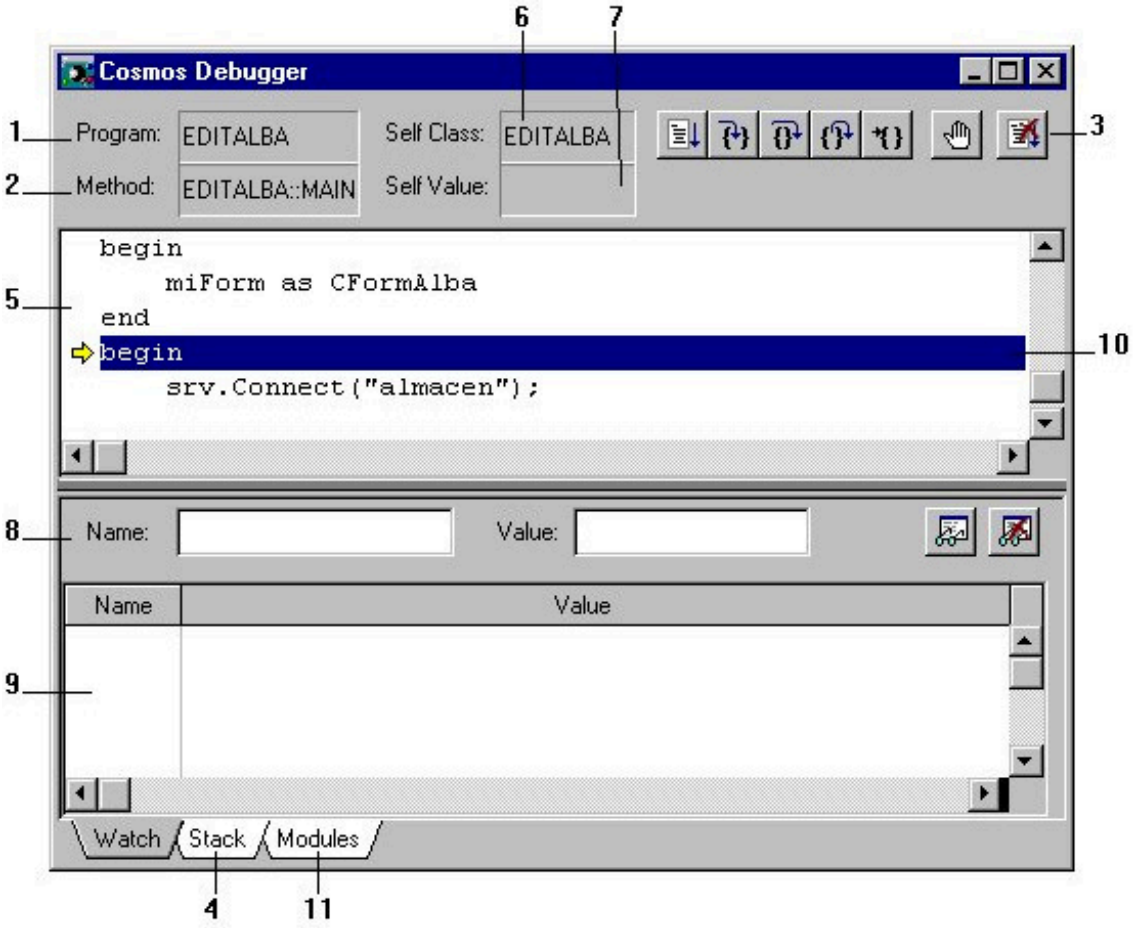


Figura 17.2. Ejecución del depurador

La ventana del depurador consta de las siguientes partes:

1. Campo donde se indica la librería, el include o el programa en curso.
2. Método que se está ejecutando.
3. Menú de botones que agrupa las opciones del debugger.
4. Stack:: Muestra la secuencia actual de llamadas a métodos y los argumentos con los que se ha realizado la llamada.
5. Un área donde se muestra el código fuente del módulo.
6. Muestra la clase a la que pertenece el método que se esta ejecutando.
7. Valor del objeto en curso.
8. Un área donde se permite modificar los valores de los objetos. Se indicará el objeto y su nuevo valor.
9. Watch List: Permite controlar como cambian los valores de objetos y expresiones durante la ejecución del programa.
10. El Punto de ejecución: indica la línea siguiente del código fuente que va a ser ejecutada por el depurador. Siempre que se detenga la ejecución de un programa dentro del depurador (por ejemplo cuando se realiza la ejecución paso a paso hasta un punto del programa), el depurador resalta una línea de código que indica la posición del punto de ejecución.
11. Muestra los módulos cargados en memoria.

Opciones del depurador

El depurador permite controlar la ejecución de un programa. Puede controlar si el programa ejecuta una sola línea de código, toda una función o un bloque del programa completo. Al indicar cuándo ejecutar el programa y cuando detenerse se puede desplazar con mayor rapidez por las secciones que funcionen correctamente y concentrarse en las secciones que contienen los errores.

La ejecución del programa se controla con el menú de botones de la figura 3 que agrupa las opciones del debugger:

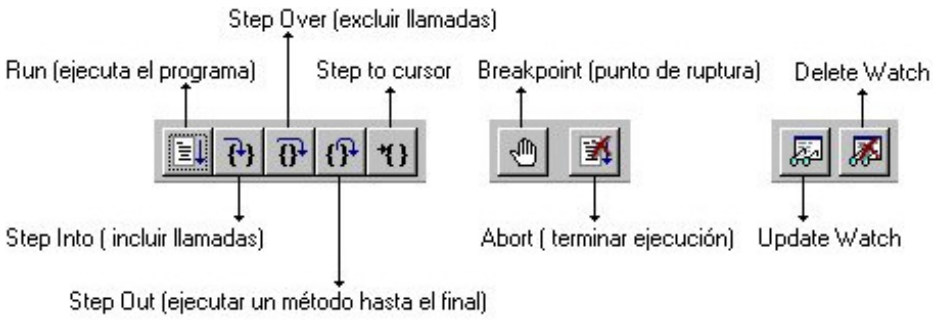


Figura 17.3. Opciones de depuración del menú de botones.

- Añadir y eliminar puntos de ruptura.
- Ejecutar el programa hasta un punto de ruptura.
- Incluir llamadas a métodos (Step into).
- Excluir llamadas a métodos (Step.over).
- Ejecutar hasta la última instrucción de un método (Step out).
- Ejecutar el programa hasta la posición del cursor (Step to cursor).
- Abortar la ejecución del programa.
- Examinar valores de datos del programa.

Al ejecutar el código por medio del depurador, toda la ejecución del programa se basa en líneas de código fuente. Se puede controlar la velocidad de depuración hasta el nivel de una sola línea de código; no se puede depurar una o varias sentencias que se encuentren en una sola línea. Asimismo, el depurador considera una sentencia simple que ocupe varias líneas de texto como una sola línea de código.

Step into (incluir llamadas)

Esta opción ejecuta una sola sentencia del programa cada vez. Si el punto de ejecución se encuentra en una llamada a un método no predefinido en Cosmos, al elegir esta opción se ejecuta paso a paso este método colocando el punto de ejecución en su primera instrucción.

Si el punto de ejecución se encuentra en la última instrucción de un método y elige Step Into, el depurador abandona el método y coloca el punto de ejecución en la línea de código siguiente en la llamada a esa función.

Para ejecutar el comando Step Into pulse el botón  del menú de iconos.

Step Over (excluir llamadas)

Este comando, al igual que Step Into, permite ejecutar sentencias del programa una a una. Sin embargo, si se utiliza el comando Step Over cuando el punto de ejecución se encuentra en una llamada a un método, el depurador ejecuta automáticamente todas las instrucciones del método (en lugar de paso a paso) y, a continuación, coloca el punto de ejecución en la sentencia siguiente a la llamada del método.

Para ejecutar el comando Step Over pulse el botón  del menú de iconos.

Conforme se realiza la depuración de un programa, se puede utilizar el comando Step Into en algunos métodos y el comando Step Over y Step Out en otros. Si sabe que un método se comporta tal y como se ha diseñado, puede excluir las llamadas a ese método con la seguridad de que la llamada al método no producirá un error. si no esta seguro de si el método se comporta correctamente, puede incluir llamadas al método para comprobar si funciona.

Step Out

Este comando, al igual que Step Into y Step Over, permite ejecutar sentencias del programa una a una. Sin embargo, si se utiliza el comando Step Out cuando el punto de ejecución se encuentra en mitad de un método, el depurador ejecuta las instrucciones que quedan del método (en lugar de paso a paso) y, a continuación, coloca el punto de ejecución en la sentencia siguiente a la llamada del método.

Para ejecutar el comando Step Out pulse el botón  del menú de iconos.

Step to cursor

Al iniciar una sesión de depuración, normalmente se ejecuta el programa hasta un punto situado antes de donde se sospecha que se encuentra el error. En ese punto, habrá que asegurar que todos los valores de los datos son correctos. De esta forma puede ejecutar el programa hasta otro punto y volver a comprobar que todo funciona correctamente.

Para ejecutar un programa hasta un determinado punto:

1. En el área de código de la ventana del debugger, coloque el cursor en la línea de código donde desee iniciar (o reanudar) la depuración.
2. Pulse el botón  del menú de iconos. El programa se ejecuta hasta que se llega a la posición indicada con el cursor de texto en el área de código. En este punto el depurador devuelve el control y coloca el punto de ejecución en esa línea de código.

Utilización de puntos de ruptura

Los puntos de ruptura se utilizan para detener la ejecución del programa en determinadas posiciones del código fuente durante la depuración. Al definir puntos de ruptura en posibles partes conflictivas del código fuente, puede ejecutar el programa con la certeza de que se detendrá en el lugar que se desee depurar.

Cuando el depurador encuentra un punto de ruptura detiene la ejecución del programa y muestra la línea del punto de ruptura en el área de edición de código. Puede utilizar entonces el depurador para examinar el estado del programa.

Definir y suprimir puntos de ruptura

Para que un punto de ruptura sea válido, debe definirse en una línea de código ejecutable. Los puntos de ruptura definidos en líneas de comentario, líneas en blanco o en líneas de código no ejecutable no son válidas y se desactivan al ejecutar el programa.

Para definir un punto de ruptura :

1. Seleccione la línea en el área de código de la ventana del debugger.
2. Pulse el botón  del menú de iconos.

Al definir un punto de ruptura, se resalta la línea en que este se ha definido con un punto rojo como puede verse en la figura 17.4.

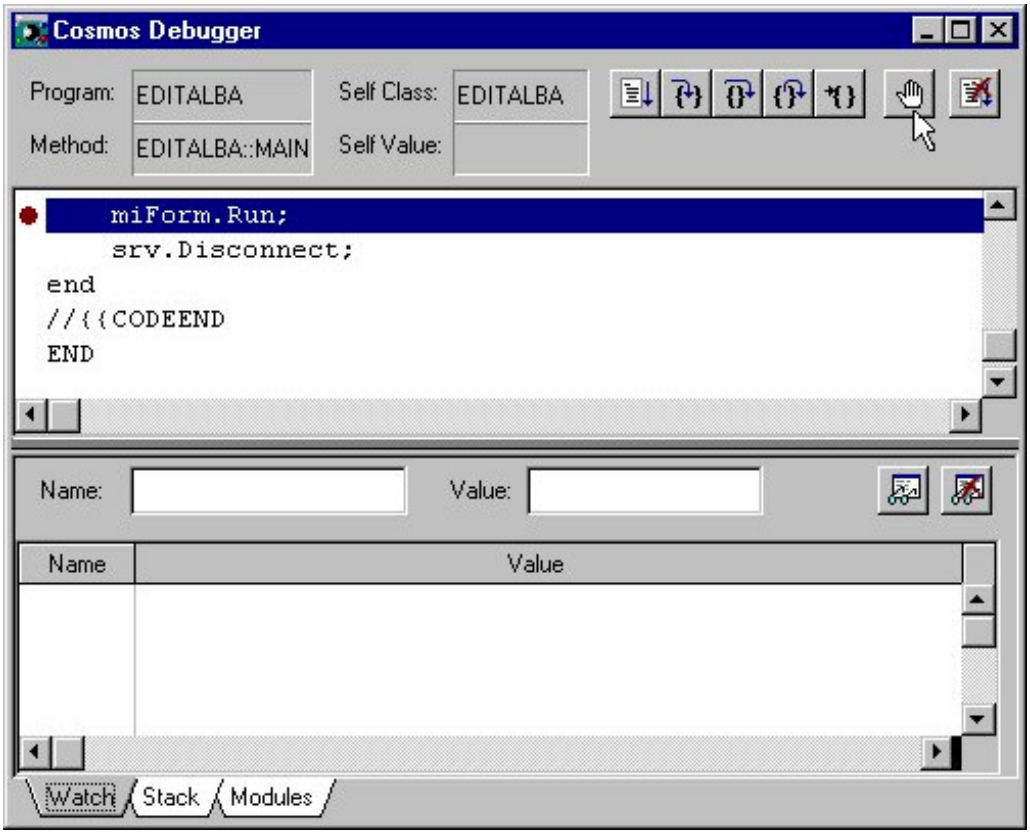


Figura 17.4. Definir un punto de ruptura

Cuando ya no necesite examinar el código en la posición de un punto de ruptura, puede suprimirlo de la sesión de depuración.

Para suprimir un punto de ruptura:

1. En el área de código de la ventana del debugger sitúe el cursor en la línea de código que contiene el punto de ruptura.
2. Pulse el botón  del menú de iconos.

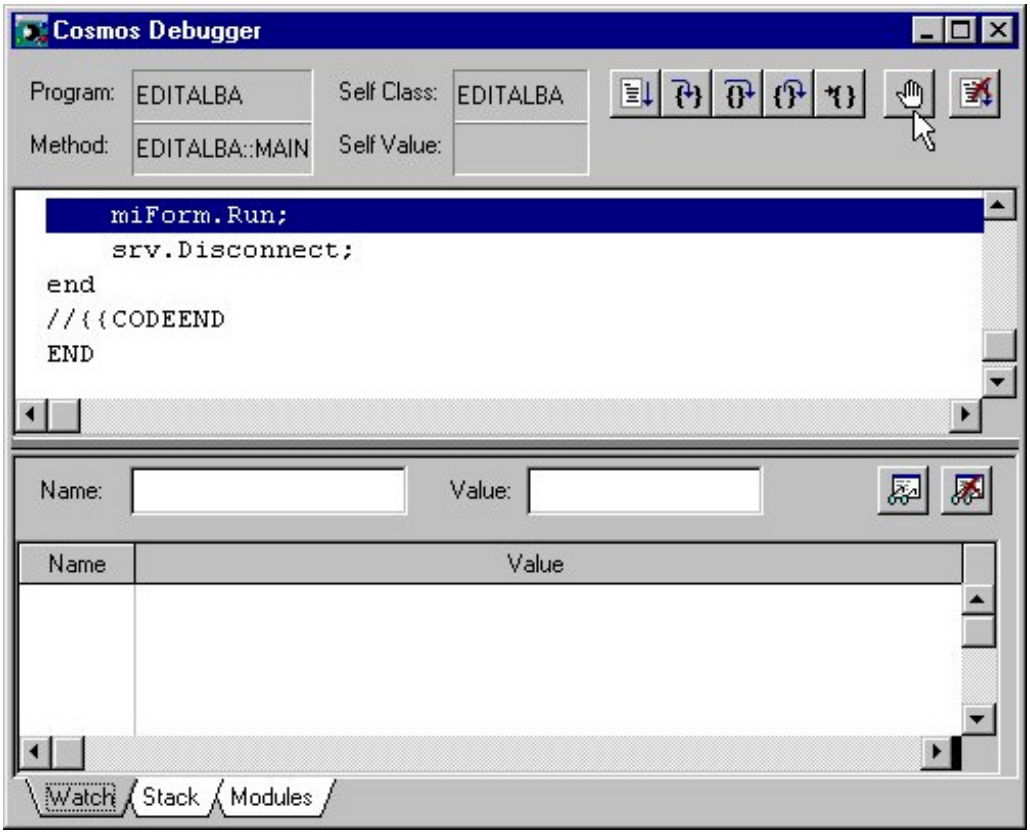


Figura 17.5. Suprimir un punto de ruptura

Ejecutar el programa hasta un punto de ruptura

Los puntos de ruptura se definen en líneas de código donde de desea detener la ejecución de un programa. Al ejecutar el programa hasta el punto de ruptura, el programa se ejecuta sin pararse hasta que llega a una determinada posición del código fuente.

Para ejecutar el programa hasta un punto de ruptura pulse el botón  del menú de iconos.

Si no inserto ningún punto de ruptura antes de ejecutar esta opción el programa se ejecutará hasta el final sin detenerse.

Terminar la ejecución del programa en depuración

En cualquier momento puede terminar la ejecución del programa en modo depuración pulsando el botón  del menú de iconos. Al ejecutar esta opción se cierra la ventana del debugger.

Examinar los valores de datos del programa

Normalmente será necesario examinar los valores de las variables del programa para localizar los errores. Por ejemplo puede ser útil conocer el valor de una variable de índice cuando se esta ejecutando un bucle for, o los valores de los parámetros pasados a una llamada de función.

El depurador contiene las siguientes herramientas para ayudarle a examinar los datos de un programa :

- La ventana de Watch (listado de puntos de observación).
- La ventana Stack (pila de llamadas).

Observar expresiones

Los puntos de observación se utilizan para controlar cómo cambian los valores de variables durante la ejecución del programa.

Para introducir un punto de observación en la ventana de watch:

1. Introduzca el nombre de la variable en el campo Name.
2. Pulse el botón  del menú de iconos.
3. En la ventana de watch (listado de puntos de observación) aparece el valor actual de la variable.

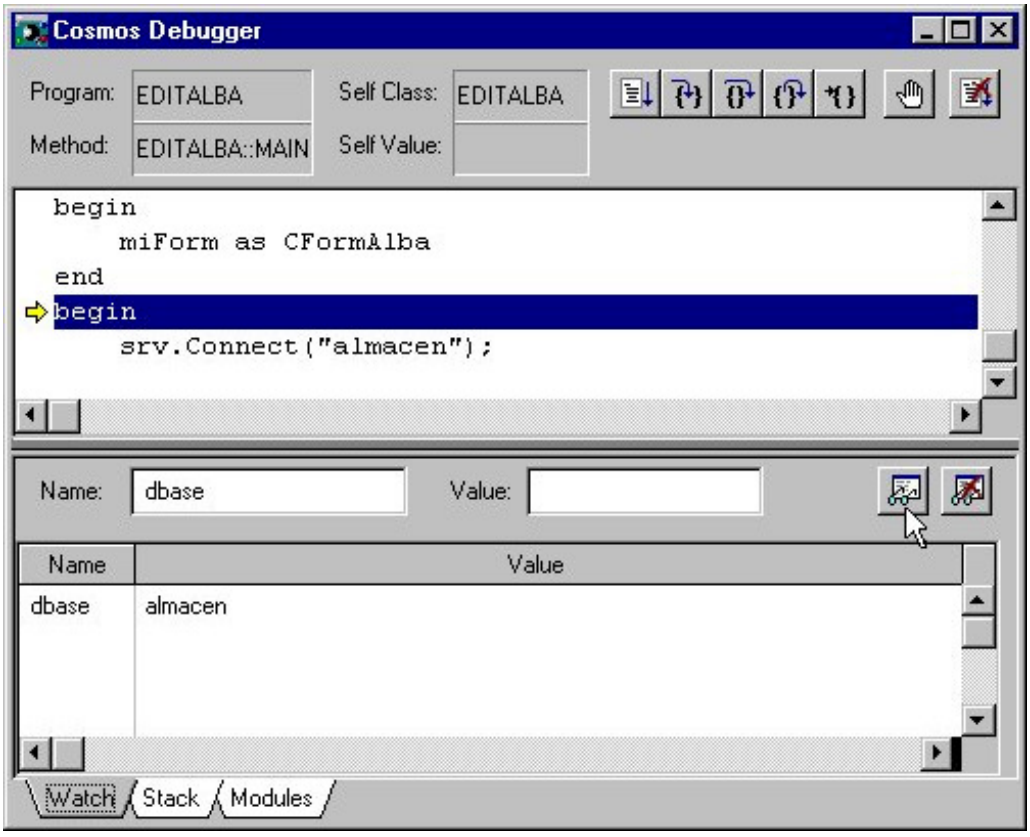


Figura 17.6. Añadir un punto de observación

Según se va ejecutando el programa, cambia el valor del punto de observación cuando el programa actualiza los valores de las variables contenidas en la ventana de watch.

Si el punto de ejecución se desplaza a una posición en la que no se ha definido ninguna de las variables de los puntos de observación, el valor del punto de observación queda sin definir. Si el punto de ejecución vuelve a una posición donde puede evaluarse la expresión del punto de observación, en la ventana de watch se vuelve a mostrar su valor.

Borrar un punto de observación

Para borrar un punto de observación en la ventana de watch:

1. Seleccione en la ventana de watch la variable que desee eliminar.
2. Pulse el botón  del menú de iconos.

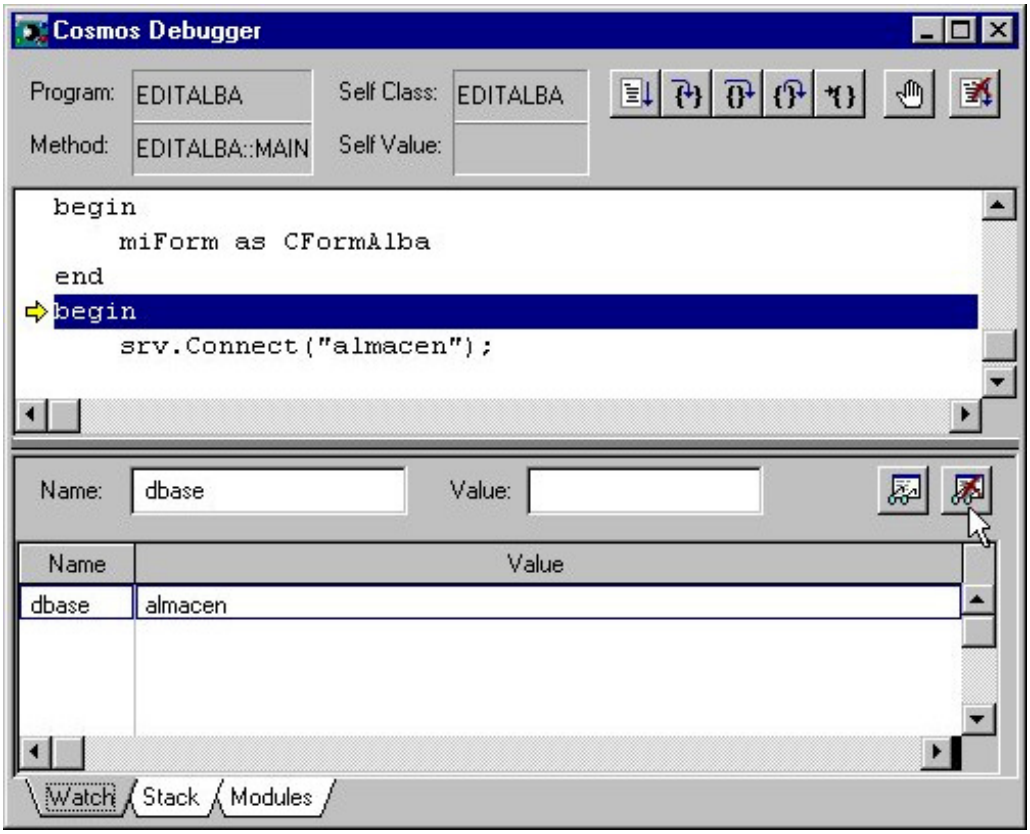


Figura 17.7. Eliminar un punto de observación

Modificación de los valores de variables

Una vez evaluada una variable, puede modificar su valor, La modificación del valor de un dato del programa durante una sesión de depuración permite probar varias hipótesis de error diferentes y comprobar como una sección del código se comporta en determinadas circunstancias.

Para modificar el valor de una variable:

1. Introduzca el nombre de la variable en el campo Name.
2. Escriba el nuevo valor en el campo Value.
3. Pulse el botón  del menú de iconos.
4. En la ventana de watch (listado de puntos de observación) se muestra el nuevo valor de la variable.

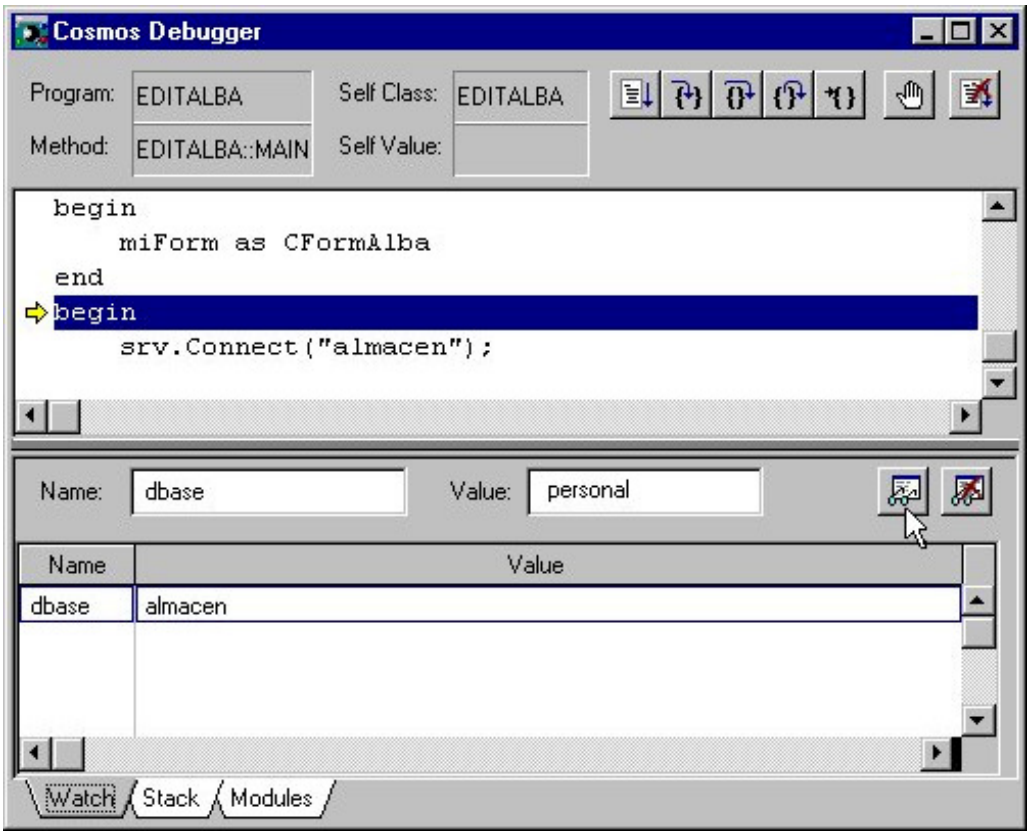


Figura 17.8. Modificar el valor de una variable

Cuando modifica el valor de una variable con el depurador, la modificación sólo es efectiva para esa ejecución del programa. Los cambios realizados con el debugger no tiene efecto en el código fuente del programa ni en el programa compilado. Para realizar cambios definitivos debe modificar el código fuente y volver a compilar el programa.

Stack

Al depurar un programa resulta útil conocer el orden de las llamadas a métodos realizadas en la posición del programa actual. Con el área stack de la ventana del debugger se puede ver la secuencia actual de llamadas a métodos. Sirve también para visualizar los argumentos pasados a una llamada de método.

En la parte superior del stack se incluye el último método llamado por el programa. Debajo se encuentra la entrada para el método previamente llamado. El listado continúa con la primera función llamada, al final de la lista.

Con el stack puede ver el código fuente incluido en una determinada llamada de función :

Para utilizar el stack :

1. Haga doble clic en la función que ha llamado a la función.
2. El editor de código se activa con el cursor situado en la posición de la llamada de función.

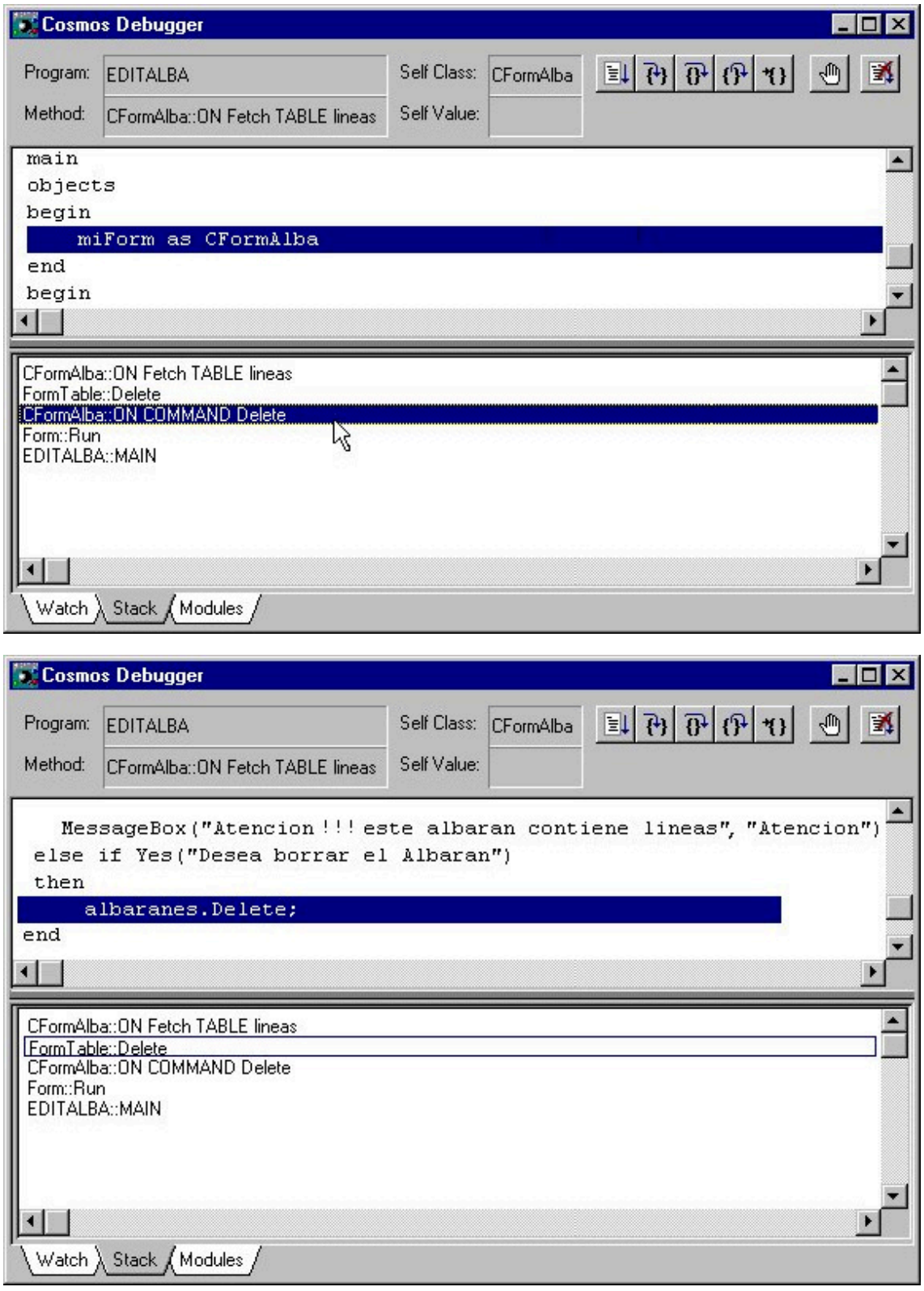


Figura 17.9. Utilización del Stack

En la figura 17.9. se puede ver que se esta ejecutando la llamada al método Run de la clase Form en la función Main del módulo. Si se hace doble clic en el stack sobre el método Run, la ventana de código se activa con el cursor situado en la llamada al método Run dentro del código de la función Main.

Capítulo 18
SQL Interactivo

Contenidos

1. Introducción

El SQL Interactivo permite editar y ejecutar ficheros con extensión *.sql* que contendrán instrucciones pertenecientes al CTSQL. Gracias a la utilización del módulo *MultiWay,* el SQL Interactivo permite trabajar no sólo con el gestor CTSQL propio de MultiBase, sino también con otros de los existentes en el mercado de distintos fabricantes (Informix, Ingres, Oracle, etc.), tanto en modo local como en instalaciones con arquitectura cliente-servidor.

Para ejecutar el SQL Interactivo, hay dos formas:

- ejecutando la instrucción csql, cuya sintaxis es la siguiente:

csql [fichero]

fichero Nombre del fichero *.sql* con el que se desea trabajar.

- mediante el icono *CSQL Interactivo* del grupo COSMOS 

2. Aspecto general del SQL interactivo

La ventana principal del SQL Interactivo tiene el aspecto que se muestra en la siguiente figura:

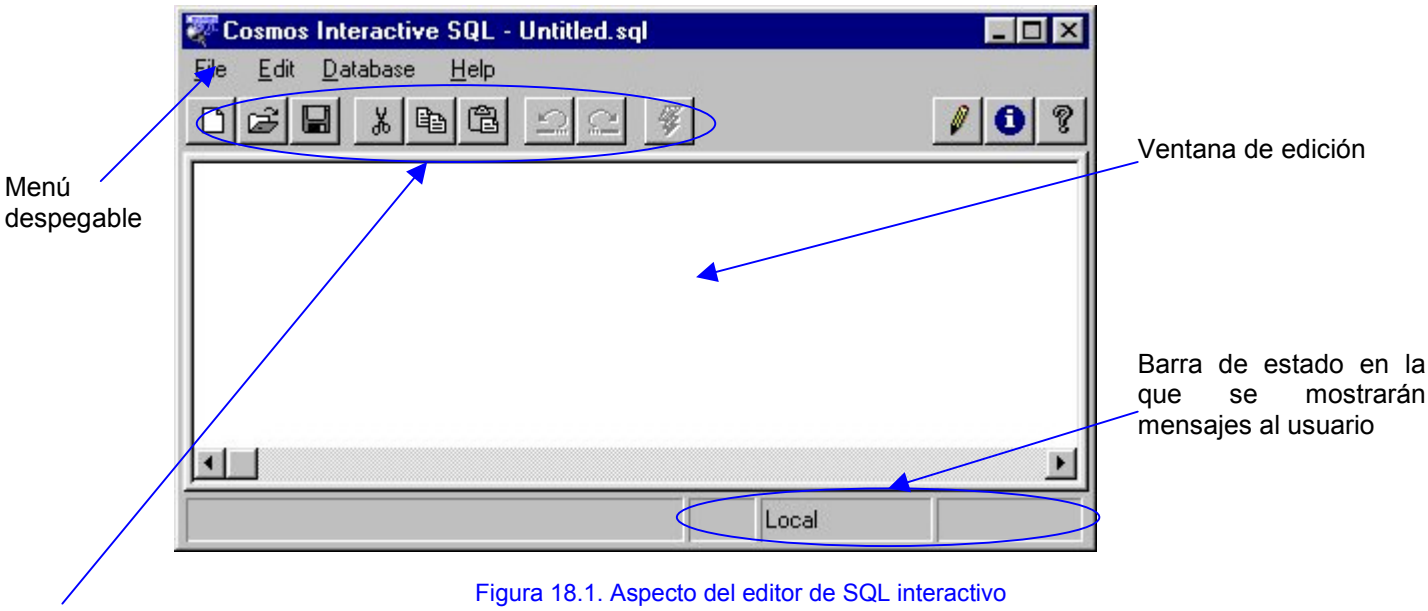


Figura 18.1. Aspecto del editor de SQL interactivo

El Sql Interactivo permite trabajar en dos modos de edición diferentes:

Inserción Inserta texto a partir del punto en que se encuentre situado el cursor y desplaza el texto existente.

Reemplazo Los caracteres que se escriban sustituirán al texto existente a partir del punto en que se encuentre situado el cursor.

Para cambiar el modo de operación de inserción a reemplazo o viceversa pulse la tecla *[Insert]*. El modo de operación por defecto al arrancar es el de inserción.

El modo de inserción en curso se reconoce por la forma del cursor de edición, ancho en modo reemplazo y estrecho en modo inserción.

Figura 18.2. Aspecto del cursos en modo reemplazo y en modo inserción

El SQL interactivo trabaja sobre ficheros con extensión *.sql* que contienen instrucciones pertenecientes al CTSQL.

En estos ficheros no se pueden incluir datos variables. Si un fichero incluye más de una instrucción CTSQL, cada una de ellas, y opcionalmente la última, tiene que finalizar en un punto y coma (;).

Por ejemplo:

```
CREATE TABLE cliente (
     numero         INTEGER        NOT NULL LABEL "numero" RIGHT,
    dni                CHAR(10)     LABEL "dni",
    nombre             CHAR(25)   LABEL "nombre" RIGHT,
    direccion          CHAR(25)   LABEL "Direccion",
    ciudad             CHAR(25)   LABEL "ciudad",
    telefono           CHAR(10)   LABEL "telefono" RIGHT,
    departamento       CHAR(10)   LABEL "Departamento",
    fijo               CHAR(1)    CHECK ($ in ("S","N")) DEFAULT "N"
                       UPSHIFT LABEL "fijo"
    )
PRIMARY KEY (numero);
CREATE INDEX numero ON cliente (numero DESC);


CREATE TABLE gest_clientes (
     Numero            INTEGER     NOT NULL LABEL "Número",
    Id                 CHAR(10)   DOWNSHIFT LABEL "Identificador",
    client          INTEGER         LABEL "Cliente" RIGHT,
    operacion       SMALLINT        LABEL "Tipo operación",
    fecha              DATE         LABEL "Fecha de la operación"
     )
PRIMARY KEY (numero)
FOREIGN KEY oper_clien (operacion)
REFERENCES operaciones
    ON UPDATE RESTRICT
    ON DELETE RESTRICT
;
```

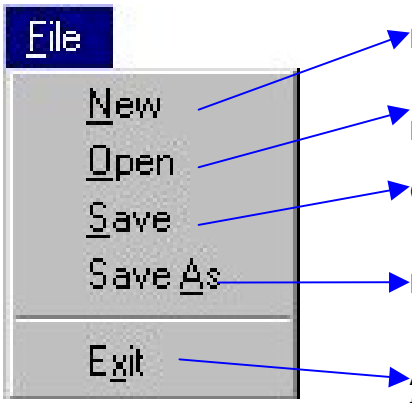
Además del SQL Interactivo de Cosmos, para ejecutar un fichero con extensión *.sql* podremos emplear también los métodos de la clase *SqlServer* predefinida en Cosmos. Estos métodos son los siguientes:

Métodos	Efecto
SqlFile	Este método ejecuta una o varias instrucciones del SQL almacenadas en un fichero cuya extensión sea *.sql*. En caso de que alguna instrucción SQL a ejecutar sea una SELECT, se mostrarán por defecto todas las filas de su tabla derivada en una ventana
SqlFileToFile	Este método ejecuta una o varias instrucciones del SQL almacenadas en un fichero cuya extensión sea *.sql*. En caso de que alguna instrucción SQL a ejecutar sea una SELECT, el resultado de su ejecución se obtendrá en un fichero

3. Elementos de menú del SQL interactivo

Menú File

Las opciones incluidas en el menú *File* permiten crear y guardar ficheros *.sql*, y acceder a ficheros *.sql* existentes. Estas opciones son las siguientes:



Permite crear un nuevo fichero *.sql* vacío.

Permite editar un fichero *.sql* existente.

Guarda el fichero *.sql* en edición.

Permite guardar el fichero *.sql* en edición con otro nombre.

Abandona el Sql Interactivo y retorna al punto desde el que fue invocado.

Crear un nuevo fichero .sql (Opción New)

La opción *New* del menú *File* permite editar un fichero *.sql* nuevo denominado *untitled.sql*.

El documento en edición se borrará sin pedir confirmación. Este sistema permite al usuario ejecutar sentencias de una en una, limpiando el área de edición con agilidad.

Esta opción se puede ejecutar pulsando el botón  del menú de iconos.

Abrir un fichero .sql (Opción Open)

La opción *Open* del menú *File* permite abrir un fichero *.sql* existente. Al igual que ocurría en la opción *New*, no se pide confirmación, y los cambios del documento en edición se perderán si no han sido salvados.

La ejecución de esta opción muestra un cuadro de diálogo con la lista de ficheros y directorios para permitir la selección del documento con el que se desea trabajar. El árbol de directorios se iniciará siempre en el directorio correspondiente a la variable de entorno DBTEMP.

Figura 18.3. Aspecto del cuadro de diálogo Abrir fichero sql

Esta opción se puede ejecutar pulsando el botón  en el menú de iconos.

Guardar un fichero .sql (Opción Save)

La opción *Save* del menú *File* guarda el contenido del fichero *.sql* activo. Si se trata de un documento nuevo, su ejecución será idéntica a la de la opción *Save As*, presentando un cuadro de diálogo para indicar el nombre que se desea asignar al documento y el directorio donde se desea guardar.

Guardar un fichero .sql con otro nombre (Opción Save As)

La opción *Save As* del menú *File* guarda tanto un fichero *.sql* de nueva creación como otro ya existente asignándole otro nombre. Para la selección del nombre se abre el cuadro de dialogo de Guardar como iniciado en el directorio correspondiente a la variable de entorno DBTEMP.

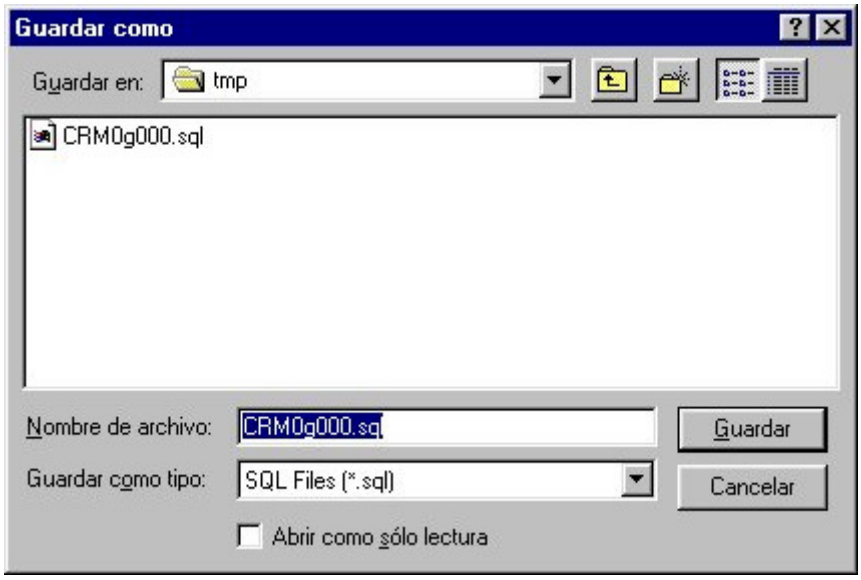


Figura 18.4. Cuadro de diálogo Guardar como

Si el nombre asignado al fichero *.sql* coincidiese con el de algún otro ya existente, el sistema avisaría al usuario presentando un mensaje en pantalla, en cuyo caso el usuario podrá elegir entre:

Figura 18.5. Mensaje de aviso

- Guardar el fichero *.sql* con el mismo nombre (*Sobre-escribir*), con lo cual se grabarían las modificaciones que hubieran podido realizarse sobre él. En este caso el sistema pediría confirmación avisando que se trata de un documento ya existente que se va a sobreescribir.
- Asignar un nombre distinto al actual, con lo cual estaríamos generando un fichero *.sql* nuevo.
- Cancelar la operación.

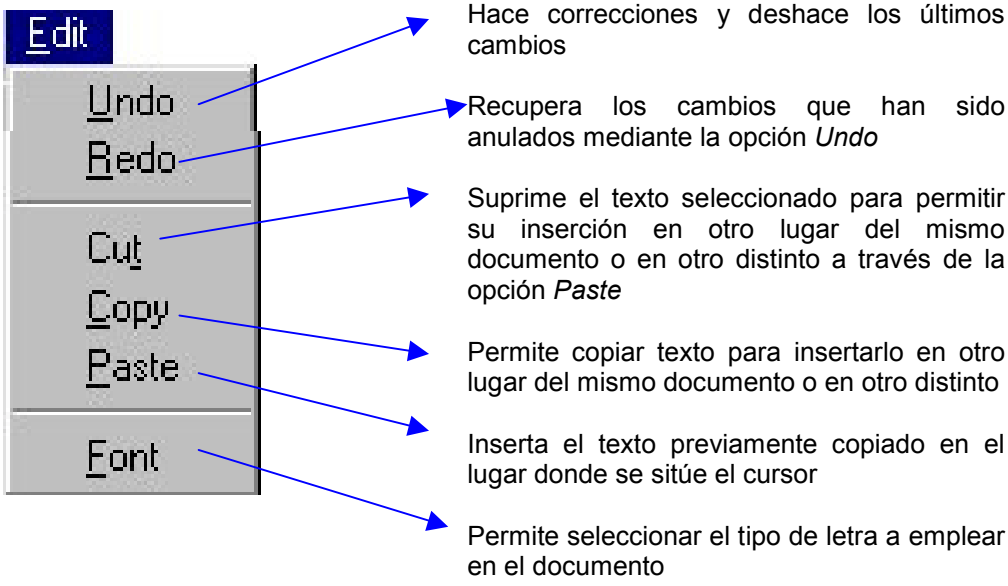
Esta opción se puede ejecutar pulsando el botón  del menú de iconos.

Salir del SQL interactivo (Opción Exit)

La opción *Exit* del menú *File* cierra el SQL Interactivo y retorna al punto desde el que fue invocado, sin pedir confirmación, los cambios del documento en edición se perderán si no han sido salvados.

Menú Edit

Las opciones incluidas en el menú *Edit* permiten realizar ciertas operaciones de edición sobre el texto del documento activo.



Seleccionar texto

Se pueden introducir cambios globales que afecten a bloques de texto con sólo seleccionarlos y ejecutar a continuación algunas de las opciones del menú *Edit* del menú principal. Para seleccionar texto en un documento podremos emplear indistintamente el ratón o el teclado:

Para seleccionar texto con el ratón proceda de la siguiente manera:

1. Sitúe el cursor delante del primer carácter del bloque de texto a seleccionar y pulse el botón izquierdo del ratón.
2. Arrastre el cursor hasta el último carácter que desee incluir en la selección.
3. Suelte el botón del ratón.

Para seleccionar una sola palabra haga doble click sobre ella. Para cancelar la selección haga clic en cualquier lugar del documento.

Para seleccionar texto con el teclado proceda de la siguiente manera:

1. Utilice las teclas de dirección para desplazar el cursor hasta el primer carácter del bloque a seleccionar.
2. Pulse la tecla [Mayúsculas] y, manteniéndola presionada, utilice las teclas de dirección para mover el cursor hasta el último carácter que desee seleccionar.

Si desea cancelar la selección pulse cualquiera de las teclas de dirección.

Deshacer los últimos cambios (Opción Undo)

El Sql Interactivo guarda un registro con algunas de las últimas modificaciones realizadas.

Si comete algún error o cambia de idea mientras esta escribiendo puede utilizar la opción *Undo* del menú *Edit* para hacer correcciones y deshacer los últimos cambios.

Para deshacer la última acción, pulse la tecla *[Ctrl]* + *[Z]* o el botón  del menú de botones.

Si desea deshacer más cambios, ejecute otra vez la opción *Undo*.

Rehacer modificaciones (Opción Redo)

El Sql Interactivo permite recuperar los cambios que han sido anulados mediante la opción *Undo* del menú *Edit*.

Para rehacer la última acción, pulse la tecla *[Ctrl]* + *[A]* o el botón  del menú de botones. Para deshacer los cambios realizados con esta opción, utilice la opción *Undo*.

Si desea rehacer más cambios, ejecute otra vez la opción *Redo*.

Copiar texto (Opción Copy)

La opción *Copy* del menú *Edit* copia el texto seleccionado del fichero *.sql* activo al portapapeles de Windows para permitir su inserción en otro lugar del mismo fichero *.sql*, en otro distinto o, incluso, en otra aplicación.

Para copiar texto:

1. Seleccione el texto que desee copiar.
2. Pulse la tecla *[Ctrl]* + *[C]* o el botón  del menú de iconos.

También puede copiar texto seleccionándolo y arrastrándolo con el botón izquierdo del ratón a la posición deseada.

Puede mover texto de sitio seleccionándolo y arrastrándolo con el botón izquierdo del ratón a la posición deseada manteniendo la tecla [Mayúsculas] pulsada.

Pegar texto (Opción Paste)

La opción *Paste* del menú *Edit* inserta el texto previamente copiado al portapapeles de Windows en el lugar donde se sitúe el cursor.

Para pegar texto:

1. Copie o corte el texto que desee insertar en otro lugar.
2. Sitúe el cursor en el lugar donde desee insertar el texto copiado.
3. Pulse el botón  del menú de iconos o la tecla *[Ctrl] + [V]*.

Cortar texto (Opción Cut)

La opción *Cut* del menú *Edit* elimina el texto seleccionado y lo almacena en el portapapeles de Windows para permitir su inserción en otro punto del fichero *.sql* activo, en otro distinto o, incluso, en otra aplicación.

Para cortar texto:

1. Seleccione el texto que desee cortar.
2. Pulse el botón  del menú de iconos o la tecla *[Ctrl] + [X]*.

Borrar texto

El SQL Interactivo permite eliminar texto del documento activo. Para borrar texto disponemos de varias posibilidades:

- Para borrar caracteres a la derecha del punto de inserción pulse la tecla *[Del] ([Supr])*.
- Para borrar caracteres a la izquierda del punto de inserción pulse la tecla *[Retroceso] ([Backspace])*.
- Para borrar palabras o bloques de texto selecciónelos previamente y pulse cualquiera de las teclas citadas en los dos puntos anteriores.

También puede borrar texto utilizando las teclas *[Del]* o *[Retroceso]* para eliminar caracteres uno a uno.

Seleccionar Fuente (Opción Font)

Mediante la opción *Font* del menú *Edit* el usuario podrá seleccionar la fuente de trabajo con la que se mostrará e imprimirá el documento.

Al ejecutarla se mostrará un cuadro de diálogo para elegir el tipo de letra, su tamaño y estilo (normal, negrita, cursiva, etc.).

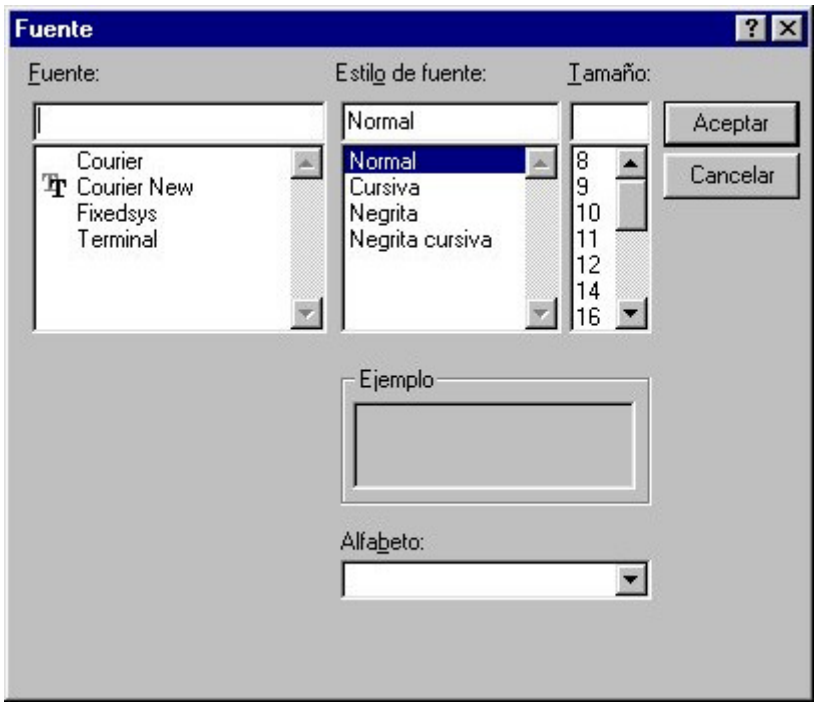
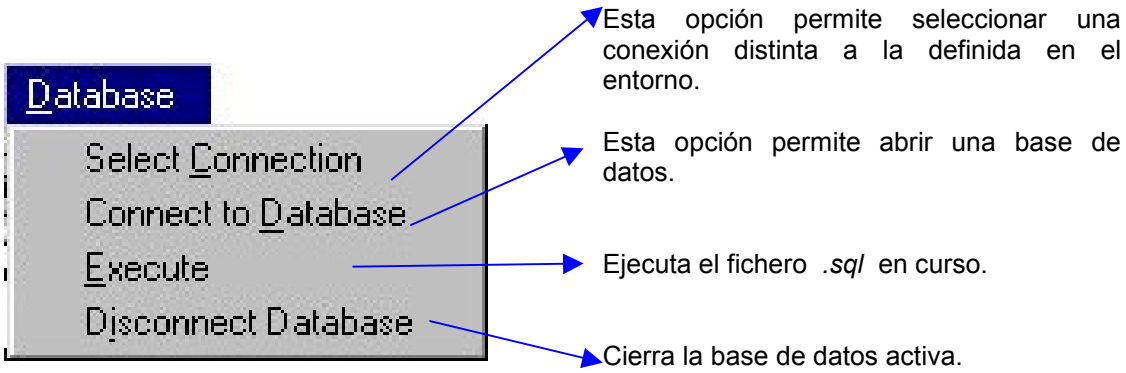


Figura 18.6. Cuadro de diálogo para seleccionar la fuente

Esta opción se puede ejecutar desde el menú de iconos pulsando el botón  .

Menú DataBase

Este menú tiene las opciones necesarias para que el SQL Interactivo trabaje con una base de datos determinada.



Seleccionar una conexión (Opción Select Connection)

Esta opción permite seleccionar un entorno de conexión distinto al definido en el entorno de usuario.

Mediante esta opción se podrá establecer una conexión local o remota (cliente-servidor), pudiendo acceder a otros gestores de bases de datos.

Al ejecutarla aparecerá un cuadro de diálogo con una lista desplegable, que incluye las conexiones definidas en el fichero de configuración COSMOS.INI, para seleccionar la conexión deseada. Si el entorno seleccionado se corresponde con una conexión remota y no tiene definida la variable de entorno DBPASSWD y/o DBUSER automáticamente se pedirá la contraseña y/o el nombre de usuario para poder establecer la conexión en el siguiente acceso al servidor de la base de datos.

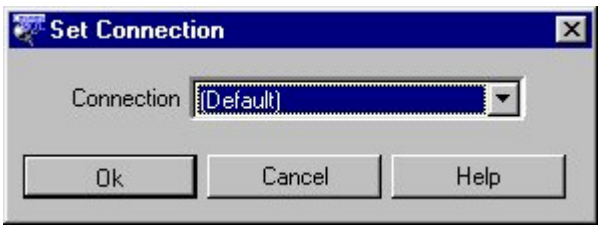


Figura 18.7. Cuadro de diálogo de selección de conexión

Seleccionar una Base de Datos (Opción Connect to Database)

Esta opción permite elegir la base de datos sobre la que desea trabajar.

Al ejecutar esta opción se muestra un cuadro de diálogo con una lista desplegable en la que aparecen todas las bases de datos detectadas por el sistema, es decir, las bases de datos existentes en las conexiones definidas dentro del fichero de Configuración COSMOS.INI.

Figura 18.8. Cuadro de diálogo de selección de Base de Datos

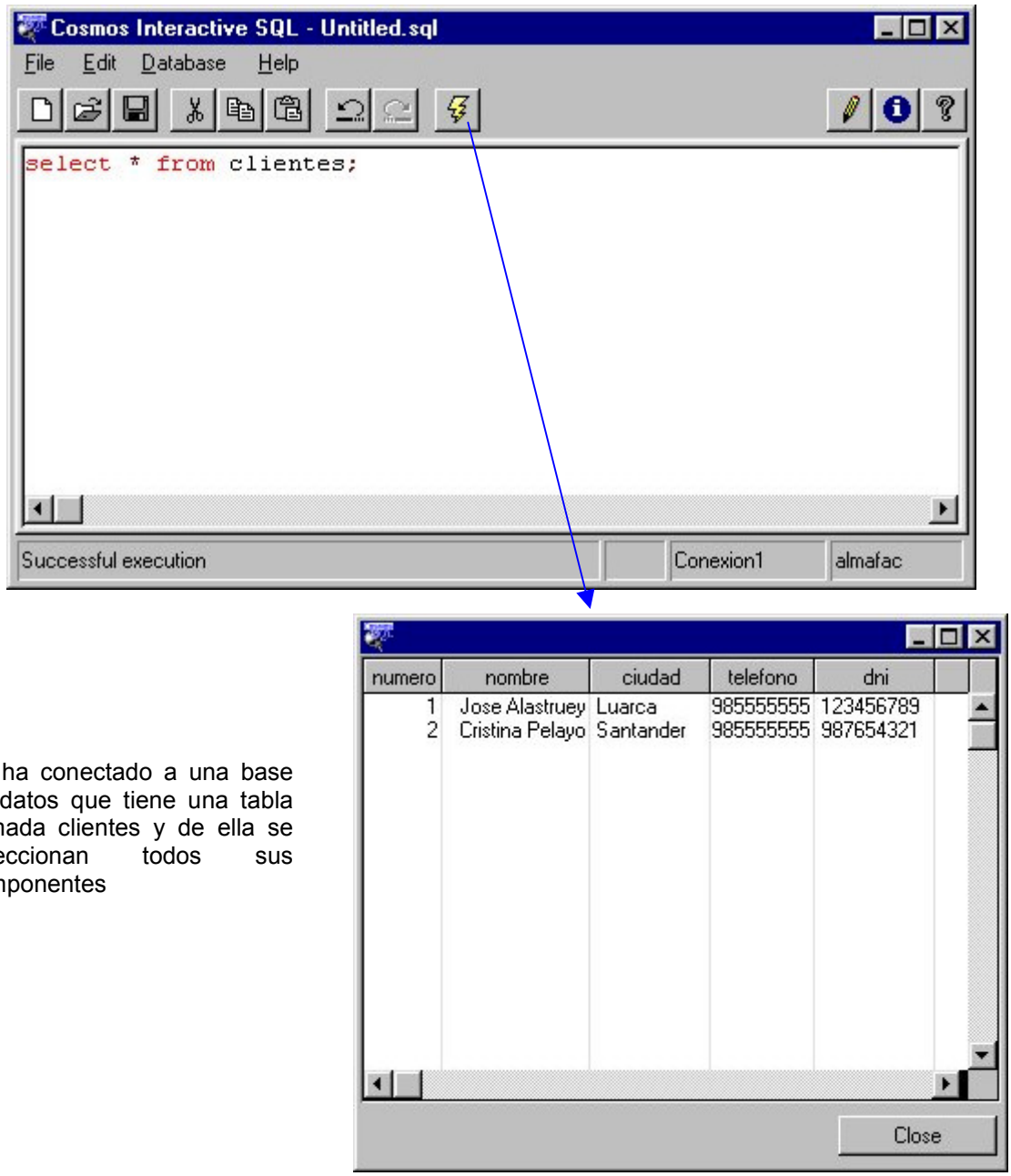
Seleccione la base de datos de la lista en curso y si no aparece escriba el nombre de la nueva base de datos que desea seleccionar (si la conexión no es vía ODBC, este nombre deberá ser un literal alfanumérico de 10 caracteres de longitud como máximo).

Si la conexión definida en el entorno es local aparecerán los nombres de las bases de datos disponibles, mientras que si la conexión es remota sólo aparecerá la base de datos de la variable de entorno DBNAME, si esta se encuentra definida. Si la conexión es ODBC aparecerán las fuentes de datos ODBC definidas desde el panel de control.

La base de datos también puede abrirse ejecutando la sentencia DATABASE, si la admite el servidor de la base de datos.

Ejecutar un fichero .sql (Opción Execute)

Mediante esta opción se podrán ejecutar las instrucciones del CTSQL contenidas en el fichero *.sql* en edición. El resultado de la ejecución de instrucciones SELECT se mostrará en una ventana. El progreso de la ejecución se mostrará en la barra de estado.



Se ha conectado a una base de datos que tiene una tabla llamada clientes y de ella se seleccionan todos sus componentes

Figura 18. 9. Ejecución de una instrucción SELECT

Esta opción se puede ejecutar pulsando *[Ctrl] + [E]* o pulsando el botón  del menú de iconos.

Cerrar la base de datos (Opción Disconnect)

Esta opción cierra la base de datos activa. La base de datos también puede cerrarse ejecutando la sentencia CLOSE DATABASE, si la admite el servidor de la base de datos.

4. Ficheros generados y utilizados por el SQL interactivo

El SQL Interactivo de Cosmos utiliza y genera una serie de ficheros identificados por sus extensiones. Estas extensiones y su significado son las siguientes:

Extension	Significado
*.scs	Estos ficheros se encargan de componer el orden de caracteres en la tabla ASCII para utilizarla en las operaciones de comparación y ordenación, y son asignados a la base de datos cuando esta se crea con la instrucción CREATE DATABASE y mediante la cláusula COLLATING.
*.ocs	Ficheros objetos generados por el compilador de ficheros de ordenación (tcollcom).
*.sql	Ficheros ASCII que incluyen instrucciones del SQL..
*.unl	Ficheros ASCII generados al descargar una tabla. Estos ficheros contienen todas las filas de la tablas descargadas en formato ASCII, y su estructura es la generada por la instrucción UNLOAD, Cada línea del fichero corresponde con una fila de la tabla, diferenciando los campos mediante un separador que, por defecto es la barra vertical (carácter ASCII 124 |).
*.dbs	Directorio que representa la base de datos creada por el CTSQL.
*.dat	Fichero binario que contendrá los datos de una base de datos. La única forma de acceso a este fichero es mediante instrucciones del CTSQL. Este fichero tiene que estar siempre en el mismo directorio que su respectivo *.idx.
*.idx	Fichero binario que contendrá la estructura de todos los índices relativos a una tabla de la base de datos. La elección del índice por el cual se desea acceder a los datos depende de la optimización de la instrucción SELECT del CTSQL (cláusulas WHERE y ORDER BY).

No se debe editar nunca los ficheros de extensión **.dat* y **.idx*

Anexo A: Gramática del COOL

Contenidos

1. Introducción

Las convenciones empleadas en Cosmos para presentar la sintaxis de cualquier elemento son las siguientes:

Elemento	Significado
[ ]	Aquellos elementos que se encuentren entre corchetes son opcionales
{ }	Indica la obligatoriedad de incluir uno de los elementos encerrados entre llaves
|	(carácter ASCII 124). Este carácter se utiliza para separar los diferentes elementos opcionales u obligatorios de la sintaxis, dependiendo de si éstos van entre corchetes o entre llaves
...	El elemento anterior a los puntos puede aparecer más veces en la sintaxis
,...	Como el caso anterior, pero cada nueva aparición del elemento debe ir precedida por una coma
palabra	Las palabras en minúsculas corresponden con identificadores COOL o expresiones, etc
PALABRA	Las palabras en mayúsculas y en negrita son reservadas del lenguaje COOL por tanto invariables
<u>subrayado</u>	Si no se especifica otra opción, la palabra o palabras subrayadas se toman por defecto
negrita	Cualquier signo de las convenciones que se encuentre en negrita es obligatorio en la sintaxis

2. Tokens

token ::= identifier | keyword | literal | string | operator | punctuator

3. Identificador (identifier)

identifier ::= {letter | underscore} {{ underscore | letter | digit}...}

Elemento	Significado
letter	Puede ser mayúsculas o minúsculas. Rango a-z y A-Z
underscore	Carácter de subrayado _
digit	Carácter numérico. Rango 0-9

4. Literales

literal ::= number | string | **TODAY** | **NOW** | **NULL** | **TRUE** | **FALSE**
number ::= {digit}... [.{digit}...]
string ::= “character1... “ | ‘character2... ‘

Elemento	Significado
number	Es una secuencia de digitos que puede tener un caracter “.”, que representan un número en base 10. Si aparece el caracter “.”, se considera que number es un literal de la clase Decimal. En caso contrario, si number es menor que 32768 se considera como un literal de la clase Smallint. Si está comprendido entre 32768 y 2147483647 se considera como un literal de la clase Integer y en caso contrario, se considera de la clase Decimal
string	Es una cadena de caracteres, sin salto de línea entre ellos. character1 puede ser cualquier caracter distinto de “””, mientras que character2 puede ser cualquier caracter distinto de “”, La longitud máxima de un literal string es de 256 caracteres
TODAY	Es un literal dinámico de la clase Date cuyo valor es la fecha del sistema
NOW	Es un literal dinámico de la clase Time cuyo valor es la hora del sistema
NULL	Es un literal estático de la clase Object cuyo valor es “desconocido”
TRUE	Es un literal estático de la clase Boolean cuyo valor es “cierto”
FALSE	Es un literal estático de la clase Boolean cuyo valor es “falso”

5. Operadores

logical_operator ::= **OR | AND | XOR**
comparison_operator ::= == | <> | < | <= | > | >= | **MATCHES | LIKE**
aritmetic_operator ::= + | += | - | -= | * | ** | *= | / | /= | % | %=
unary_operator ::= **NOT** | ++ | -- | -

6. Expresiones

expression ::= primary_expression
| unary_operator expression
| expression = expression
| expression logical_operator expression
| expression comparison_operator expression
| expression aritmetic_operator expression
| expression **BETWEEN** expression **AND** expression
| expression IN (expression,...)
| expression **IS [NOT] NULL**

primary_expression ::= literal
| (expression)
| {**SELF | PARENT | MODULE**}
| [{**SELF | PARENT | MODULE**} .] simple_expression
| primary_expression . simple_expression
| primary_expression [expression,...]
| primary_expression ([expression,...])

simple_expression ::= identifier | **OPERATOR** op | **CONVERSOR** class
op ::= logical_operator|comparison_operator|aritmetic_operator| unary_operator

logical_operator ::= **OR | AND | XOR**
comparison_operator ::= == | <> | < | <= | > | >= | **MATCHES | LIKE**
aritmetic_operator ::= + | += | - | -= | * | ** | *= | / | /= | % | %=

unary_operator ::= **NOT** | ++ | -- | -

Elemento	Significado
literal	La clase asociada a la expresión es la del literal
simple_expression	Expresión simple que no se subdivide en otras expresiones
Identifier	La clase asociada a la expresión depende del elemento del lenguaje al que represente identifier: - Nombre de constante: la clase asociada es la de la constante. - Nombre de objeto: la clase asociada es la del objeto. - Nombre de método (función, dll o conversor): la clase asociada es la clase de retorno especificada en el prototipo del método
class	Es un identificador de una clase Simple. La clase asociada a la expresión es class

7. Condiciones

condition ::= expression

Elemento	Significado
expression	Expresión cuya clase asociada es la clase predefinida Boolean o bien su interfaz especifica un método conversor a la clase predefinida Boolean

8. Instrucciones

Sintaxis:

statement ::= call_statement

| null_statement
| compound_statement
| if_statement
| switch_statement
| do_statement
| for_statement
| forever_statement
| repeat_statement
| while_statement
| break_statement
| continue_statement
| return statement

call_statement ::= [**CALL**] expression ;

null_statement ::= ;

compound_statement ::= **BEGIN** statement… **END**

if_statement ::= **IF** condition **THEN** statement1 [**ELSE** statement2]

switch_statement ::=
SWITCH expression1
BEGIN
[{**CASE** switch_condition,... : statement1}...]
[**DEFAULT** : statement2]
END

switch_statement ::=
SWITCH
BEGIN
[{**CASE** condition,... : statement1}...]
[**DEFAULT** : statement2]
END

switch_condition ::= [switch_comparison_operator] expression
| **IN** (expression,...)
| **IS** [**NOT**] **NULL**
| **BETWEEN** expression **AND** expression

switch_comparison_operator ::= < | <= | > | >= **| MATCHES | LIKE**

do_statement ::= **DO** statement **WHILE** condition;

for_statement ::= **FOR** identifier = expression1 [**DOWN**] **TO** expression2
[**STEP** expression3] **DO** statement

for_statement ::= **FOR** identifier **AS** expression,... **DO** statement

forever_statement ::= **FOREVER** statement

repeat_statement ::= **REPEAT** statement **UNTIL** condition;

while_statement ::= **WHILE** condition **DO** statement

break_statement ::= **BREAK**;

continue_statement ::= **CONTINUE**;

return_statement ::= **RETURN** [expression];

Instrucción Call

Sintaxis:

call_statement ::= [**CALL**] expression ;

Elemento	Significado
Expression	Cualquier expresión cuya evaluación finalice con la invocación de un método

Instrucción Vacia

Sintaxis:

null_statement ::= ;

Instrucción Compuesta

Sintaxis:

compound_statement ::= **BEGIN** statement… **END**

Elemento	Significado
statement	Cualquier instrucción

9. Control de Flujo

Instrucción IF

Sintaxis:

if_statement ::= **IF** condition **THEN** statement1 [**ELSE** statement2]

Parámetro	Significado
Condition	Condición de bifurcación
statement1	Instrucción a la que se cede el control si el resultado de evaluar condition es TRUE
statement2	Instrucción a la que se cede el control si el resultado de evaluar condition es distinto de TRUE

Si condition evalúa a NULL, por tanto es distinta de TRUE, se ejecuta statement2, si se ha especificado.

Instrucción SWITCH

- **Formato1**

switch_statement ::=
SWITCH
BEGIN
[{**CASE** condition,... : statement1}...]
[**DEFAULT** : statement2]
END

Parámetro	Significado
Condition	Si evalúa a TRUE, provoca la ejecución de statement1
Statement1	Se ejecuta si alguna condición de las lista de condiciones de de su rama CASE evalúa a TRUE
Statement2	Se ejecuta si ninguna de las condiciones anteriores evalúa a TRUE

- **Formato2**

switch_statement ::=
SWITCH expression1
BEGIN
[{**CASE** switch_condition,... : statement1}...]
[**DEFAULT** : statement2]
END

switch_condition ::= [switch_comparison_operator] expression
| **IN** (expression,...)
| **IS** [**NOT**] **NULL**
| **BETWEEN** expression **AND** expression

switch_comparison_operator ::= < | <= | > | >= **| MATCHES | LIKE**

Parámetro	Significado
Expression1	Es la expresión a comparar
Expression	Es la expresión con la que se compara expresión1, aplicando el operador indicado
Statement1	Se ejecuta si alguna condición de las lista de condiciones de su rama CASE evalúa a TRUE
Statement2	Se ejecuta si ninguna de las condiciones anteriores evalúa a TRUE

Instrucción DO

Sintaxis:

do_statement ::= **DO** statement **WHILE** condition;

Parámetro	Significado
Condition	Si evalúa a TRUE, provoca una nueva iteración de statement. En caso contrario se cede el control a la siguiente instrucción a do_statement
statement	Se ejecuta mientras condition evalúe a TRUE

Instrucción FOR

- **Formato 1**

Sintaxis:

for_statement ::= **FOR** identifier = expression1 [**DOWN**] **TO** expression2
[**STEP** expression3] **DO** statement

Parámetro	Significado
Identifier	Referencia un objeto. Este puede ser de cualquier clase
expression1	Valor inicial que toma identifier
expression2	Valor que determina la continuación del bucle. Si se especifica DOWN, identifier ha de aceptar la operación >= con expresión2. En caso contrario ha de aceptar la operación <= con expresión2 Esta comparación se realiza al principio de cada iteración.Si el resultado la operación es TRUE, se ejecuta statement
expression3	Es opcional. Si no se especifica STEP, identifier ha de aceptar la operación –, si se ha especificado DOWN o la operación ++ si no se ha especificado DOWN. Si se especifica STEP, identifier ha de aceptar la operación -= con expression3, si se ha especificado DOWN o la operación += con expression3 si no se ha especificado DOWN. Esta operación se realiza al final de cada iteración
Statement	Instrucción que se itera

- **Formato 2**

Sintaxis:

for_statement ::= **FOR** identifier **AS** expression,... **DO** statement

Parámetro	Significado
identifier	Referencia un objeto. Este puede ser de cualquier clase
expression	Expresión que se asigna identifier al inicio de cada iteración. identifier ha de aceptar la operación = con expression
statement	Instrucción que se itera. Se ejecuta tantas veces como expresiónes aparezcan

Instrucción FOREVER

Sintaxis:

forever_statement ::= **FOREVER** statement

Parámetro	Significado
statement	Instrucción que se itera

Instrucción REPEAT

Sintaxis:

repeat_statement ::= **REPEAT** statement **UNTIL** condition;

Parámetro	Significado
statement	Se ejecuta mientras condition no evalúe a FALSE
condition	Si evalúa a FALSE, se produce una nueva iteración de statement. En caso contrario se cede el control a la siguiente instrucción a repeat_statement

Instrucción WHILE

Sintaxis:

while_statement ::= **WHILE** condition **DO** statement

Parámetro	Significado
condition	Si evalúa a TRUE., se produce una nueva iteración de statement. En caso contrario se cede el control a la siguiente instrucción a while_statement
statement	Se ejecuta mientras condition evalúe a TRUE

Instrucción BREAK

Sintaxis:

break_statement ::= **BREAK**

Instrucción CONTINUE

Sintaxis:

continue_statement ::= **CONTINUE**;

Instrucción RETURN

Sintaxis :

return_statement ::= **RETURN** [expression];

Parámetro	Significado
expression	Ha de evaluar a una clase Simple

10. Definición de Constantes

Sintaxis:

constant_section ::= [**CONSTANTS BEGIN** [constant_definition...] **END**]
constant_definition ::= identifier = literal

Parámetro	Significado
identifier	Identificador único en su ámbito

11. Definición de Clases

Sintaxis:

class_section ::= [**CLASSES BEGIN** [class_definition...] **END**]
class_definition::=[access]identifier1**IS** [**ABSTRACT]** identifier2 [class_interface]
access ::= **PUBLIC | PROTECTED | PRIVATE**

Parámetro	Significado	
access	Indica la visibilidad de la clase fuera del módulo en el que se ha declarado. Puede ser de tres tipos	
	PUBLIC	Una clase declarada como pública siempre es visible fuera del módulo
	PROTECTED	Una clase declarada como protegida siempre es visible fuera del módulo, pero no será derivable fuera del módulo
	PRIVATE	Una clase privada solamente puede ser utilizada dentro del módulo en el que se ha declarado
identifier1	Identificador único en su ámbito. Es el nombre de la nueva clase	
identifier2	Nombre de la clase padre de la que estamos definiendo. Si se especifica **ABSTRACT**, la clase que estamos definiendo se considera abstracta o incompleta. En este caso, la clase identifier2 ha de ser también abstracta y no se debe especificar class_interface	
class_interface	Completa la interfaz de la nueva clase, si no se ha especificado **ABSTRACT**	

12. Definición de Objetos

Sintaxis:

object_section ::= [**OBJECTS BEGIN** [object_definition...] **END**]
object_definition ::= [access] identifier... AS class [**DEFAULT** literal]
access ::= **PUBLIC | PROTECTED | PRIVATE**

Parámetro	Significado	
access	Nos limita el acceso a la interfaz del objeto. Puede ser de tres tipos	
	PUBLIC	La interfaz del objeto no está restringida. Si el módulo es de tipo include, el objeto es visible en cualquier otro módulo que dependa de éste
	PROTECTED	La interfaz del objeto está restringida al propio módulo y a todos los métodos de las clases que se definan en el módulo. El objeto no es visible en ningún otro módulo
	PRIVATE	La interfaz del objeto está restringida al propio módulo. El objeto no es visible en ningún otro módulo
identifier	Ha de ser único en la sección de objetos	
class	Puede ser cualquier clase no abstracta. Si es una de las clases simples Smallint, Integer, Decimal, Char, Time, Date o Boolean, se puede especificar literal como valor por defecto mediante la cláusula DEFAULT. Si la clase es Char, se puede especificar una longitud. Si la clase es Decimal o Money, se puede especificar una longitud y una precisión	

13. Definición de Métodos

Sintaxis:

method_definition ::= prototype object_section method_body

Parámetro	Significado
Prototipo	Es la interfaz del método. Describe las características del mensaje asociado a la operación que implementa el método. Contiene toda la información necesaria para invocarlo correctamente

Prototype

Sintaxis:

prototype ::= [access] method_identifier [([parameter,...])] [**RETURN** class1]
access ::= **PUBLIC | PROTECTED | PRIVATE**
parameter ::= [**VAR**] identifier... **AS** class2 [**DEFAULT** literal]

Parámetro	Significado	
access	Nos limita "quíen" puede invocar el método, es decir, restringe la clase del objeto emisor del mensaje asociado a la operación que implementa el método. Hay tres tipos de acceso:	
	PUBLIC	la clase del objeto emisor no está restringida
	PROTECTED	la clase del objeto emisor ha de ser la misma que la clase del objeto receptor, o descender de ésta
	PRIVATE	la clase del objeto emisor ha de ser la misma que la clase del objeto receptor
method_identifier	Tipo de método junto con un identificar adicional si es necesario. Especifica la operación asociada al método. Ha de ser único en el interfaz de la clase en la que se define el método. Los métodos se clasifican en: Función, Dll, Operador, Conversor, Notificación, Comando, Main.	
Class1	Es la clase del valor de retorno. Si se especifica, ha de ser de clase Simple y además el valor de retorno será un objeto temporal. En caso contrario el valor de retorno es el mismo objeto sobre el cual se ha invocado al método (SELF)	

- *Parameter*

Un parámetro es un objeto que describe los requisitos que ha de cumplir un argumento en la invocación de un método. Su identificador identifier ha de ser único en la lista de parámetros.

Es posible especificar un literal como valor por defecto para el parámetro mediante la cláusula DEFAULT. De esta manera el compilador insertará literal en la invocación del método, si no se especifica un argumento. La cláusula DEFAULT solo se puede especificar si class2 es una de las siguientes clases simples: Smallint, Integer, Decimal, Char, Time, Date o Boolean. Una vez que se especifica que un parámetro tiene un valor por defecto, se ha de especificar en los parámetros siguientes.

Es posible especificar que se desea recibir una lista variable de argumentos especificando la clase ArgList para el último parámetro. Por tanto, podemos subdvidir en tres partes la lista de parámetros:

- lista de parámetros sin valor por defecto
- lista de parámetros con valor por defecto
- parámetro de clase Arglist.

Cada una de estas partes es opcional, pero de aparecer, lo han de hacer en en el orden anterior.

Si no se especifica VAR, el runtime realiza una copia del argumento antes de la invocación del método y la destruye después. En caso contrario, el estado resultante de las operaciones realizadas sobre el argumento durante la invocación del método es permanente. Si la clase class2 del parámetro no es Simple, se considera que se ha especificado VAR. class2 puede ser cualquier clase, incluso una clase abstracta.

Object Section (Sección de objetos locales)

Sintaxis:

object_section ::= [**OBJECTS BEGIN** [object_definition...] **END**]
object_definition ::= identifier... **AS** class [**DEFAULT** literal]

Parámetro	Significado
Identifier	Ha de ser único tanto en la sección de objetos locales, como en la lista de parámetros
Class	Puede ser cualquier clase no abstracta. Si es una de las clases simples Smallint, Integer, Decimal, Char, Time, Date o Boolean, se puede especificar literal como valor por defecto mediante la cláusula DEFAULT. Si la clase es Char, se puede especificar una longitud. Si la clase es Decimal o Money, se puede especificar una longitud y una precisión

Method Body (Cuerpo del método)

Consta de una secuencia de instrucciones orientadas a realizar una tarea específica. La invocación del método finaliza cuando se ejecuta la última instrucción de la secuencia o cuando se ejecuta la instrucción RETURN. En este caso, es obligatorio especificar un valor de retorno, si el prototipo del método lo exige.

Sintaxis:

method_body ::= [compound_statement]

Si una clase redefine un método, el prototipo de éste ha de ser idéntico al del método que se está redefiniendo, salvo la clase de retorno, que puede ser descendiente de la clase de retorno del método original.

Un paso de mensaje, que implica la invocación de un método, se implementa mediante una expresión. Esta expresión es de la forma:

message ::= [expression .] operation [([argument,...])]
argument ::= expression

Parámetro	Significado
Expression	Es el receptor del mensaje. La evaluación de esta expresión determina el objeto receptor del mensaje. Su especificación es opcional, pues dicho objeto puede estar especificado implícitamente. Consulte el apartado ámbito y visibilidad de un objeto
Operation	Es el selector del mensaje. Es la única parte de éste cuya especificación es obligatoria
Argument,...	Es la lista de argumentos del mensaje. Cada argumento se valida con el parámetro que ocupa la misma posición en el prototipo. Para que esta validación sea correcta, se ha de cumplir una y solo una de estas condiciones: 1. El argumento ha de ser el literal NULL 2. La clase del argumento ha de ser la misma que la clase del parámetro, o descender de ésta 3. La interfaz de la clase del argumento ha de especificar un método conversor hacia la clase del parámetro 4. Si la clase del parámetro es igual o descendiente de la clase ArgList, el argumento y todos los que le siguen, han de cumplir una de las tres condiciones anteriores con respecto a la clase asociada a la clase ArgList

Dado que operation puede ser polimórfica, pues puede haber más de una clase cuya interfaz especifique esta operación, se determina sobre la clase asociada a expression.

Tanto expression como argument, se evalúan antes de la ejecución de la primera instrucción del método asociado a operation.

Todo mensaje aparece en un método, ya que éste es la implementación de una operación. El objeto que emite este mensaje se le llama “objeto emisor”. Nos podemos referir a él con la palabra reservada SELF.

14. Function

Sintaxis de definición:

function ::= function_prototype object_section function _body
function_prototype ::= access function_identifier[([parameter,...])]
[**RETURN** class1]
function_identifier ::= [**FUNCTION**] identifier
function _body ::= compound_statement

Parámetros	Significado
Identifier	Es el nombre de la operación que implementa el método. Este nombre ha de ser único, es decir no debe coincidir con el nombre de otra operación definida en la misma clase ni con el identificador de algún atributo de ésta

Sintaxis de invocación:

function_message ::= [expression .] identifier [([argument,...])]
argument ::= expression

Parámetros	Significado
Identifier	Identificador de la función que queremos invocar

15. DLL

Sintaxis de definición:

dll ::= function_prototype object_section dll_body
dll_prototype ::= access dll_identifier [([parameter,...])] [**RETURN** class1]
dll_identifier ::= [**SELF**] **DLL** dll_path identifier
dll_body ::=

Parámetros	Significado
Identifier	Es el nombre de la operación que implementa el método. Este nombre ha de ser único, es decir no debe coincidir con el nombre de otra operación definida en la misma clase ni con el identificador de algún atributo de ésta. Este identificador ha de coincidir además con el nombre del procedimiento exportado por la librería dinámica
Dll_path	Literal de clase Char. Es el path en el que se busca la dll, en la primera invocación al método. No es necesario indicar el path completo de la Dll, basta con indicar su nombre siempre que esta se vaya a ser accesible para Windows. Sólo se comprueba en tiempo de ejecución y no en tiempo de compilación

La clase de los parámetros ha de ser Simple. No obstante, no se da ningún error en caso contrario.

Sintaxis de invocación:

dll_message ::= function_message

La sintaxis de invocación de un método dll es idéntica a la de un método función. Podemos ver un método dll como un método función con implementación externa.

Si se especifica SELF en la definición del método, el objeto sobre el que se invoca se pasará como primer parámetro.

El paso de argumentos al procedimiento de la Dll se realiza mediante el convenio Pascal. El paso de argumentos se realiza de la siguiente manera:

Clase	Por Valor	Por referencia
Smallint	Entero de 16 bits	puntero de 32 bits
Integer	Entero de 32 bits	puntero de 32 bits
Decimal	no implementado	no implementado
Char	Puntero de 32 bits	puntero de 32 bits
Date	Entero de 32 bits	puntero de 32 bits
Time	Entero de 32 bits	puntero de 32 bits
Enum	Entero de 32 bits	puntero de 32 bits
EnumSet	Entero de 32 bits	puntero de 32 bits
Complex	no implementado	no implementado
Container	no implementado	no implementado

Si un argumento se pasa por valor y su estado es desconocido (IS NULL), el valor que se pasa es indeterminado.

Si un argumento con valor NULL se pasa por referencia, tras la invocación del método, se asume que su valor ha sido modificado, por lo que a partir de este momento, sea cual sea su valor se tomará como valor real (NOT NULL).

Si el argumento es el literal NULL, se pasa 0.

El valor de retorno se realiza de manera análoga al paso de argumentos por valor. Si la clase del valor de retorno es la clase Char y el valor de retorno es 0, se considera el valor de retorno como desconocido (IS NULL). En caso contrario el estado del valor de retorno no es desconocido (IS NOT NULL).

La librería dinámica se carga la primera vez que se invoca un procedimiento de ella, y se descarga cuando finaliza la ejecución de la aplicación.

16. Operator

Sintaxis de definición:

operator ::= operator_prototype object_section operator_body
operator_prototype::=access operator_identifier[([parameter])][**RETURN** class1]
operator_identifier ::= **OPERATOR** op
operator_body ::= compound_statement

Parámetros	Significado
op	Es el nombre de la operación que implementa el método. Este nombre ha de ser único, es decir no debe coincidir con el nombre de otra operación definida en la misma clase ni con el identificador de algún atributo de ésta

17. Conversor

Sintaxis de definición:

conversor ::= conversor_prototype object_section conversor_body
conversor_prototype ::= access conversor_identifier
conversor_identifier ::= **CONVERSOR** identifier
conversor_body ::= compound_statement

Parámetros	Significado
identifier	Es la clase del valor de retorno. Siempre debe ser un identificador de una clase Simple. Dicha clase no puede ser una clase padre de la misma que lo implementa, aunque si una clase hija

18. Main

main ::= main_prototype object_section main_body
main_prototype ::= main_identifier [([parameter,...])]
main_identifier ::= **MAIN**
main_body ::= compound_statement

19. Notificaciones

notification ::= form_notification | control_notification | table_notification

Notificaciones de Control

control_notification::=control_notification_prototype
object_section
notification_body
control_notification_prototype ::= **ON** event control [(idm **AS** Integer)]
| **ON** event (control **AS** Char[, idm **AS** Integer])
notification_body ::= compound_statement

Parámetros	Significado
event	Identificador de una de las notificaciones que puede mandar un objeto de clase Control a su Form, tales como Click, RClick, Dblclick, SelChange, etc..
control	Identificador del objeto de clase Control que envía la notificación al Form
idm	Indice del elemento seleccionado en un control de tipo: Lista, grupo de cajas, grupo de botones, grupo de botones radio

Notificaciones de Form

form_notification::= form_notification_prototype object_section notification_body
form_notification_prototype ::= notification_identifier
notification_body ::= compound_statement
notification_identifier ::= ON event

Parámetros	Significado
event	Identificador de una de las notificaciones que puede mandar un objeto de clase Form, tales como Open y Close

Notificaciones de Tabla

table_notification::=table_notification_prototype object_section notification_body
table_notification_prototype ::= ON event TABLE table
| ON event TABLE (table AS Char)
notification_body ::= compound_statement

Parámetros	Significado
event	Identificador de una de las notificaciones que puede mandar un objeto de clase FormTable a su Form definidas en Cosmos, puede ser una de las siguientes: Add, New, Update, etc..
tabla	Objeto de la clase FormTable afectado por el evento
identifier	Identificador del objeto de la clase FormTable afectado por el evento

20. Comandos

command ::= form_command | table_command

Comandos de Form

Sintaxis:

form_command ::= form_command_prototype object_section command_body
form_command_prototype ::= ON COMMAND cmd
| ON COMMAND (cmd AS Char)
command_body ::= compound_statement

Parámetros	Significado
cmd	Es un identificador de comando. Puede aparecer explícito, o como el valor de un objeto de clase Char

Comandos de Tabla

Sintaxis:

table_command ::= table_command_prototype object_section command_body
table_command_prototype ::= ON COMMAND cmd TABLE table
| ON COMMAND cmd TABLE (table AS Char)
| ON COMMAND table TABLE (cmd AS Char)
| ON COMMAND TABLE(tableASChar,cmd AS Char)
command_body ::= compound_statement

Parámetros	Significado
cmd	Es un identificador de comando. Puede aparecer explícito, o como el valor de un objeto de clase Char
table	Es un identificador de tabla receptora del comando. Puede aparecer explícito, o como el valor de un objeto de clase Char

Anexo B: Palabras Reservadas del Lenguaje COOL

Contenidos

1. Identificadores de las clases predefinidas

Arglist	Decimal	Enum	Form
Array	Integer	Enumset	
Boolean	Money	Struct	
Char	Smallint	Module	
Date	Time	Page	

Se reserva para un futuro uso los identificadores: Bag, Stack, List, Tree, Queue, Vector, Set.

2. Sección CONSTANTS de un módulo

Begin	End	NULL
Constants	FALSE	TRUE
Default	Now	Today

3. Sección CLASSES de un módulo

Abstract	Default	Now	Protected
As	End	NULL	Public
Begin	FALSE	Of	Today
Classes	Is	Private	TRUE

4. Para la definición de una clase Form

Form	OBJECTS
Interface	Table
Menu	Vartable

5. Sección Screen de una clase Form

Alignment	Columns	Foreground	Multipanel	Simple
All	Command	Frame	Multiple	Slider
Allowdrag	Comment	Frameautosize	Noborder	Smallint
Allowdrop	Control	Frameborder	Nolabel	Sort
As	Current	Grayscale	None	Spin
Attach	Cursortype	Grid	Number	Split
Autosize	Datatype	Horizontal	Operation	Status
Background	Date	Icon	Page	Store
Bar	Decimal	Integer	Palind	Stretch
Bartype	Default	Interface	Panel	String
Begin	Defpush	Invisible	Paneltype	Sysmenu
Bitmap	Design	Italic	Parentgrid	Tab
Black	Disabled	Label	Password	Table
Blackborder	Double	Labelleft	Percent	Tags
Bold	Down	Left	Point	Text
Border	Dropedit	Likevar	Position	Thincaption
Bottom	Droplist	Lines	Pulldown	Thinframe
Bottomright	Edit	List	Quadruple	Tickmarks
Box	Editmode	Listcols	Radio	Ticks
Boxgroup	End	Listtype	Range	Time
Button	Etched	Lowtab	Resizing	Top
Buttongroup	Etchedin	Manual	Rgb	Topleft
Bycols	Etchedout	Mask	Right	Treeview
Center	Extra	Maximize	Rows	User
Char	File	Menu	Scroll	Values
Check	Fixed	Message	Selecttype	Variable
Checked	Focus	Minimize	Separator	Vertical
Clicktype	Font	Multiline	Show	Width
Color				

6. Sección Tables de una clase Form

By	Lookup
Depend	On
Enumerated	Order
Get	References
Join	Required
Joining	Values

7. Palabras reservadas en la definicion de la sección Template de una clase Page

Alignment	Char	File	Margin	Size
All	Check	Fixed	Mask	Smallint
As	Checked	Font	Multiline	Solid
Attach	Color	Foreground	Nolabel	Status
Autosize	Columns	Frameautosize	Number	Store
Background	Control	Grayscale	Page	Stretch
Begin	Current	Grid	Palind	Tags
Bitmap	Datatype	Group	Percent	Text
Bold	Date	Icon	Position	Thickline
Border	Decimal	Integer	Range	Tickmarks
Bottom	Design	Invisible	Repeat	Time
Box	Disabled	Italic	Rgb	Top
Boxgroup	Double	Label	Rows	Upshadow
Buttongroup	Doubleline	Labelleft	Right	Var
Bycols	DownShadow	Left	Separator	Vertical
Center	End			

8. Palabras reservadas en la definicion de una clase Menu

Allowcheck	Default	Menu
Allowicon	Disabled	Nolabel
Begin	End	Nowait
Butfile	Help	Option
Check	Icon	Pulldown
Checked	Iconfile	Separator
Command		

9. Palabras reservadas en la sección OBJECTS de un módulo

As	OBJECTS
Begin	Private
End	Protected
Default	Public
FALSE	Today
Now	TRUE
NULL	

10. Palabras reservadas en las secciones de código

And	Conversor	If	Objects	Step
As	Default	In	On	Switch
Begin	Dll	Is	Operator	Table
Between	Do	Let	Or	Then
Break	Down	Like	Parent	To
Call	Else	Main	Public	Today
Case	End	Matches	Private	TRUE
Class	FALSE	Module	Protected	Until
Code	For	Not	Repeat	Var
Command	Forever	Now	Return	While
Continue	Function	NULL	Self	Xor

Anexo C:
Clases Predefinidas

Contenidos

1. Introducción

En Cosmos se definen tres tipos de clases:

Clases virtuales

En Cosmos se denomina clase virtual a las clases predefinidas que se caracterizan por:
- El usuario no puede crear una clase derivada directamente de una clase virtual.
- No se pueden instanciar (por lo tanto son clases abstractas).

Clases abstractas

Una clase abstracta es una clase que puede utilizarse solamente como clase base de otras clases. La idea es disponer de un mecanismo que soporte la noción de un concepto general tal como número (clase Numeric) del cual utilizaremos variantes concretos Decimal, Integer, Smallint.

Pensando en esta idea, no se pueden crear objetos de una clase abstracta, lo que crearemos será objetos de sus clases derivadas.

Todas las clases creadas por el usuario en un módulo son instanciables o abstractas.

Por ejemplo, se puede definir una clase abstracta derivada de la clase Form en la que se definirán todos los métodos comunes a los Form que necesitemos definir en nuestra aplicación. Estos Form derivarán de dicha clase abstracta para poder acceder a sus métodos.

En Cosmos toda clase abstracta es derivada de otra clase abstracta. Una clase abstracta no puede derivar de una clase instanciable. En una clase abstracta solo se pueden definir métodos.

Clases instanciables

En cosmos se denomina clase instanciable a todas las clases que no son abstractas ni virtuales. Una clase es instanciable cuando el usuario puede crear objetos de dicha clase.

Todas las clases creadas por el usuario en un módulo son instanciables o abstractas.

Jerarquía de Clases predefinidas

En la siguiente figura se muestra la jerarquía de las clases predefinidas en Cosmos. La Clase Object es la clase base de toda la jerarquía y es virtual.

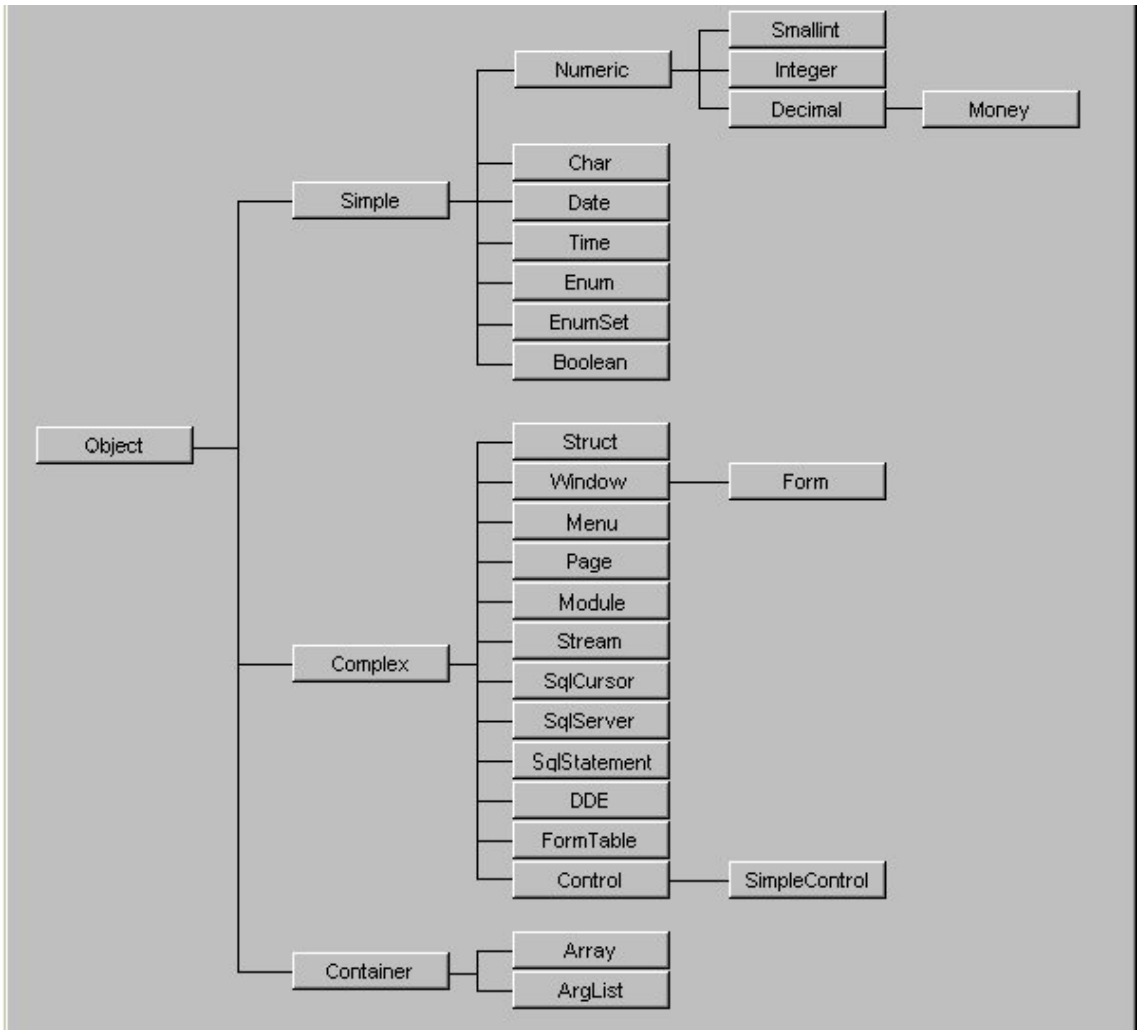


Figura C.1. Clases predefinidas de COSMOS

1. Clase Object

La Clase Object es la clase base de toda la jerarquía y es virtual. Cualquier clase deriva al final de esta clase, por tanto podemos decir que cualquier objeto posible en un programa cosmos “Es un Object”.

De la clase Object derivan:

Clase Simple	Esta clase deriva de la clase Object, es una clase virtual, de ella derivan las clases que almacenan los tipos de datos “simples” de Cosmos. Se define un tipo “simple” como aquel que su contenido puede estar representado por una expresión del lenguaje. Así mismo los objetos de tipo simple, comúnmente llamados variables en otros lenguajes de programación, son los únicos que pueden ser “pasados por copia” en la llamada a una función.
Clase Complex	Esta clase deriva de la clase Object, es una clase virtual, de ella derivan las clases que almacenan los objetos que tiene una estructura compleja como los Menus, Forms, Cursores SQL, etc.
Clase Container	Esta clase deriva de la clase Object. Es una clase virtual. De ella derivan las clases que son contenedores de objetos, como por ejemplo la Clase Array.

Operadores de la Clase Object

Operador	Funcionalidad
IS NULL	Sirve para comprobar si un objeto es nulo.
IS NOT NULL	Sirve para comprobar si un objeto es no nulo

Métodos de la Clase Object

Método	Funcionalidad
IsMe	Este método permite comprobar si dos objetos son idénticos
Name	Este método permite consultar el nombre de cualquier objeto

2. Clase Simple

Esta clase deriva de la clase Object, es una clase virtual, de ella derivan las clases que almacenan los tipos de datos “simples” de Cosmos. Se define un tipo “simple” como aquel que su contenido puede estar representado por una expresión del lenguaje. Así mismo los objetos de tipo simple, comúnmente llamados variables en otros lenguajes de programación, son los únicos que pueden ser “pasados por copia” en la llamada a una función.

De la clase simple derivan:

Clase Numeric	Esta clase deriva de la clase Simple, es una clase virtual, de ella derivan las clases que almacenan los tipos numéricos de datos.
Clase Char	Esta clase deriva de la clase Simple, es una clase instanciable. Una clase Char(n) permite almacenar un cadena de caracteres alfanuméricos de longitud máxima n. Siendo “n” un número entre 1 y 32.767. También se pueden definir objetos o clases Char sin indicar la longitud, en este caso se entiende que es un Char de longitud indefinida y se irá reservando memoria dinámicamente.
Clase Date	Esta clase deriva de la clase Simple, es una clase instanciable. Esta clase permite almacenar valores de fecha.

Clase Time — Esta clase deriva de la clase Simple, es una clase instanciable. Esta clase permite almacenar valores horarios desde 00:00:01 a 24:00:00.

Clase Boolean — Esta clase deriva de la clase Simple, es una clase instanciable. Esta clase permite almacenar objetos que solamente pueden tener valores TRUE o FALSE.

Clase Enum — Esta clase deriva de la clase Simple, es una clase abstracta. Esta clase permite almacenar objetos que pueden tener un solo valor entre una lista de valores posibles. El ejemplo clásico es la asignación de valores a los días de la semana.

Clase EnumSet — Esta clase deriva de la clase Simple, es una clase abstracta. Esta clase permite almacenar valores que representan la presencia de 0, 1 o mas elementos de un conjunto. En una clase EnumSet definida por el usuario se definiran los elementos que componen el conjunto. Un EnumSet no puede tener elementos repetidos y sus elementos no están ordenados. Un EnumSet puede tener como máximo 32 elementos. Sirve por ejemplo para almacenar los "flags" de apertura de un fichero.

3. Clase Numeric

Esta clase deriva de la clase Simple, es una clase virtual, de ella derivan las clases que almacenan los tipos numéricos de datos.

De la clase Numeric derivan:

Clase Smallint — Esta clase deriva de la clase Numeric, es una clase instanciable y permite almacenar números enteros. El rango de valores numéricos que admite está entre -32.767 y +32.767

Clase Integer — Esta clase deriva de la clase Numeric, es una clase instanciable y permite almacenar números enteros. El rango de valores numéricos que admite un objeto de esta clase es desde -2.147.483.647 hasta +2.147.483.647, ambos inclusive.

Clase Decimal — Esta clase deriva de la clase Numeric, es una clase instanciable. Una clase DECIMAL [(m[,n])] permite almacenar números decimales de coma flotante con un total de m dígitos significativos (precisión) y n dígitos a la derecha de la coma decimal (escala). m puede ser como máximo menor o igual a 32 (m = 32) y n menor o igual que m. Cuando se asignan valores a m y n, la variable decimal tendrá punto aritmético fijo. El segundo parámetro n es opcional, y si se omite se tratará como un decimal de coma flotante. Un elemento decimal(m) tiene una precisión m y un rango de valor absoluto entre 10^{-130} y 10^{125}. En caso de no especificar parámetros a un elemento DECIMAL, éste será tratado como decimal(16)

Operadores de la clase Numeric

- **Operadores aritméticos:**

Operador	Funcionalidad
+ (suma)	Este operador suma dos objetos de tipo Numeric
- (resta)	Este operador resta dos objetos de tipo Numeric
* (multiplicación)	Este operador multiplica dos objetos de tipo Numeric
/ (división)	Este operador divide dos objetos de tipo Numeric
% (módulo)	Este operador calcula el resto de la división de dos objetos numéricos enteros
** (exponenciación)	Este operador exponenciación eleva el valor de un objeto de tipo Numeric a una potencia indicada por otro
- (signo)	Operador de inversión de signo. Los objetos numéricos positivos se convierten en negativos y viceversa

- **Operadores relacionales**

Operador	Funcionalidad
== (igualdad)	Este operador de igualdad chequea si dos objetos numéricos tienen igual valor
(distinto)	Este operador de igualdad chequea si dos objetos numéricos no tienen igual valor
(mayor)	Este operador chequea si un objeto de tipo Numeric es mayor que otro
= (mayor o igual)	Este operador chequea si un objeto de tipo Numeric es mayor o igual que otro
(menor)	Este operador chequea si un objeto de tipo Numeric es menor que otro
= (menor o igual)	Este operador chequea si un objeto de tipo Numeric es menor o igual que otro
BETWEEN	Este operador chequea si un objeto de tipo Numeric esta comprendido o no en el rango dado por otros dos
IN	Este operador chequea si un objeto Numeric es igual o no a alguno de los valores incluidos en una lista de valores

Métodos de la clase Numeric

Método	Funcionalidad
Using	Este método realiza una conversión formateada de un objeto de tipo Numeric a Char
Trace	Este método muestra en una ventana el valor del objeto

Conversores

La clase Numeric tiene conversores a las siguientes clases:

- Boolean
- Smallint
- Integer
- Decimal
- Char

4. Clase Smallint

Esta clase deriva de la clase Numeric, es una clase instanciable y permite almacenar números enteros. El rango de valores numéricos que admite está entre -32.767 y +32.767.

Métodos de la clase Smallint

- **Operadores aritméticos de asignación**

Operador	Funcionalidad
= (asignación)	Este método permite asignar un valor a un objeto Smallint. Retorna el propio objeto
+= (suma)	Este método suma al valor de un objeto de tipo Smallint el valor de una expresión del mismo tipo. Retorna el primer operando, es decir, el objeto sobre el que se aplica el operador
-= (resta)	Este método resta al valor de un objeto de tipo Smallint el valor de una expresión del mismo tipo. Retorna el primer operando, es decir, el objeto sobre el que se aplica el operador
*= (multiplicación)	Este método multiplica el valor de un objeto de tipo Smallint el valor de una expresión del mismo tipo. Retorna el primer operando, es decir, el objeto sobre el que se aplica el operador
/= (división)	Este método divide el valor de un objeto de tipo Smallint entre el valor de una expresión del mismo tipo. Retorna el primer operando, es decir, el objeto sobre el que se aplica el operador
%= (módulo)	Este método calcula el resto de la división del valor del objeto entre el valor de una expresión del mismo tipo. Retorna el primer operando, es decir, el objeto sobre el que se aplica el operador

- **Operadores aritméticos**

Operador	Funcionalidad
++ (incremento)	El operador incremento suma una unidad al operando sobre el que se aplica
-- (decremento)	El operador decremento resta una unidad al operando sobre el que se aplica

Métodos de la clase Smallint

Método	Funcionalidad
Character	Devuelve el carácter correspondiente a la posición en la tabla ASCII que corresponde con el valor que tiene el objeto
MonthName	Este método devuelve el nombre completo del mes correspondiente al valor del objeto. El valor 1 retorna “January” y el valor 12 retorna “December”
MonthName3	Este método devuelve el nombre del mes con 3 letras, correspondiente al valor del objeto. El valor 1 retorna “Jan” y el valor 12 retorna “Dec”
WeekDayName	Este método devuelve el nombre completo del día de la semana correspondiente al valor del objeto. El valor 0 retorna “Sunday” y el valor 6 retorna “Saturday”
WeekDayName3	Este método devuelve el nombre de dia de la semana con 3 letras, correspondiente al valor del objeto. El valor 0 retorna “Sun” y el valor 6 retorna “Sat”

Conversor

La clase Smallint tiene conversores a la clase Char

5. Clase Integer

Esta clase deriva de la clase Numeric, es una clase instanciable y permite almacenar números enteros. El rango de valores numéricos que admite un objeto de esta clase es desde -2.147.483.647 hasta +2.147.483.647, ambos inclusive.

Operadores de la clase Integer

- **Operadores aritméticos de asignación**

Operador	Funcionalidad
= (asignación)	Este método permite asignar un valor a un objeto Integer. Retorna el propio objeto
+= (suma)	Este método suma al valor de un objeto de tipo Integer el valor de una expresión del mismo tipo. Retorna el primer operando, es decir, el objeto sobre el que se aplica el operador
-= (resta)	Este método resta al valor de un objeto de tipo Integer el valor de una expresión del mismo tipo. Retorna el primer operando, es decir, el objeto sobre el que se aplica el operador
*= (multiplicación)	Este método multiplica el valor de un objeto de tipo Integer el valor de una expresión del mismo tipo. Retorna el primer operando, es decir, el objeto sobre el que se aplica el operador
/= (división)	Este método divide el valor de un objeto de tipo Integer entre el valor de una expresión del mismo tipo. Retorna el primer operando, es decir, el objeto sobre el que se aplica el operador
%= (módulo)	Este método calcula el resto de la división del valor del objeto entre el valor de una expresión del mismo tipo. Retorna el primer operando, es decir, el objeto sobre el que se aplica el operador

- **Operadores aritméticos**

Operador	Funcionalidad
++ (incremento)	El operador incremento suma una unidad al operando sobre el que se aplica
-- (decremento)	El operador decremento resta una unidad al operando sobre el que se aplica

Conversor

La clase Integer tiene conversor a la clase Char

6. Clase Decimal

Esta clase deriva de la clase Numeric, es una clase instanciable. Una clase DECIMAL [(m[,n])] permite almacenar números decimales de coma flotante con un total de m dígitos significativos (precisión) y n dígitos a la derecha de la coma decimal (escala). m puede ser como máximo menor o igual a 32 (m = 32) y n menor o igual que m. Cuando se asignan valores a m y n, la variable decimal tendrá punto aritmético fijo. El segundo parámetro n es opcional, y si se omite se tratará como un decimal de coma flotante. Un elemento decimal(m) tiene una precisión m y un rango de valor absoluto entre 10^{-130} y 10^{125}. En caso de no especificar longitud a un DECIMAL, éste será tratado como decimal(16).

De esta clase deriva:

Clase Money Un "Money" es un "Decimal" que tiene un número de decimales por defecto igual a 2. Esta clase es instanciable.

Operadores de la clase Decimal

- **Operadores aritméticos de asignación**

Operador	Funcionalidad
= (asignación)	Este método permite asignar un valor a un objeto Decimal. Retorna el propio objeto
+= (suma)	Este método suma al valor de un objeto de tipo Decimal el valor de una expresión del mismo tipo. Retorna el primer operando, es decir, el objeto sobre el que se aplica el operador
-= (resta)	Este método resta al valor de un objeto de tipo Decimal el valor de una expresión del mismo tipo. Retorna el primer operando, es decir, el objeto sobre el que se aplica el operador
*= (multiplicación)	Este método multiplica el valor de un objeto de tipo Decimal el valor de una expresión del mismo tipo. Retorna el primer operando, es decir, el objeto sobre el que se aplica el operador
/= (división)	Este método divide el valor de un objeto de tipo Decimal entre el valor de una expresión del mismo tipo. Retorna el primer operando, es decir, el objeto sobre el que se aplica el operador
%= (módulo)	Este método calcula el resto de la división del valor del objeto entre el valor de una expresión del mismo tipo. Retorna el primer operando, es decir, el objeto sobre el que se aplica el operador

- **Operadores aritméticos**

Operador	Funcionalidad
++ (incremento)	El operador incremento suma una unidad al operando sobre el que se aplica
-- (decremento)	El operador decremento resta una unidad al operando sobre el que se aplica
- (signo)	Operador de inversión de signo. Los objetos Decimal positivos se convierten en negativos y viceversa

Métodos de la clase Decimal

Método	Funcionalidad
Round	Redondea el valor de un objeto Decimal con la precisión indicada
Truncate	Trunca el valor del objeto Decimal a la precisión indicada

Conversor

La clase Decimal tiene conversor a la clase Char

7. Clase Money

Una clase MONEY [(m[,n])] almacena números decimales de coma flotante con un total de m dígitos significativos (precisión) y n dígitos a la derecha de la coma decimal (escala), igual al tipo de dato DECIMAL. Los objetos de esta clase representan cantidades referentes a unidades monetarias.

Un "Money" es un "Decimal" que tiene un número de decimales por defecto igual a 2

8. Clase Char

Esta clase deriva de la clase Simple, es una clase instanciable. Una clase char(n) permite almacenar una cadena de caracteres alfanuméricos de longitud máxima n. Siendo "n" un número entre 1 y 32.767. También se pueden definir objetos o clases char sin indicar la longitud, en este caso se entiende que es un char de longitud indefinida y se irá reservando memoria dinámicamente.

Operadores de la clase Char

Operador	Funcionalidad
+ (suma)	Este operador concatena dos objetos de tipo alfanumérico
= (asignación)	Este operador permite asignar un valor a un objeto Char
+= (suma)	Este operador concatena una expresión alfanumérica (Char) a un objeto de tipo Char. Modifica el objeto
[ ] (substring)	Devuelve la cadena de caracteres de un objeto Char comenzando en un carácter dado y con una longitud dada

- **Operadores relacionales**

Operador	Funcionalidad
== (igualdad)	Este operador de igualdad chequea si dos objetos Char son iguales
(distinto)	Este operador chequea si dos objetos Char son distintos
(mayor)	Este operador chequea si un objeto Char es mayor que otro
= (mayor o igual)	Este operador chequea si un objeto Char es mayor o igual que otro
(menor)	Este operador chequea si un objeto Char es menor que otro
= (menor o igual)	Este operador chequea si un objeto Char es menor o igual que otro
BETWEEN	Este operador chequea si un objeto Char esta comprendido o no en el rango dado por otros dos

Operador	Funcionalidad
IN	Este operador chequea si un objeto Char es igual o no a uno de los valores incluidos en una lista de valores
LIKE	Condición de comparación de objetos Char. Devuelve TRUE o en caso de que el resultado de una expresión se ajuste o no un patrón
MATCHES	Condición de comparación de objetos Char. Devuelve TRUE o FALSE dependiendo de que el resultado de una expresión se ajuste o no a un patrón

Métodos de la clase Char

Método	Funcionalidad
AnsiToOem	Convierten el valor de un objeto Char, del conjunto de caracteres ANSI al conjunto de caracteres OEM
Ascii	Devuelve el código ASCII del primer carácter de un objeto Char
Count	Este método devuelve el número de veces que un patrón se encuentra en un objeto Char
DirName	Devuelve de un objeto Char que representa un path completo su valor eliminando el último elemento del path.
FileName	Devuelve la parte correspondiente al último elemento del path de un objeto Char que representa un path completo
GetWord	Devuelve la enésima palabra de un objeto Char
Length	Devuelve la longitud del valor de un objeto Char
L ocate	Este método busca un literal en otro
Lowcase	Devuelve el objeto Char (expresión alfanumérica) convertido a minúsculas
LTrim	Elimina los blancos situados a la izquierda en el objeto Char (expresión alfanumérica) y retorna dicho objeto modificado
NumWords	Este método cuenta las palabras de un objeto Char
OemToAnsi	Convierte el valor de un objeto Char, del conjunto de caracteres OEM al conjunto de caracteres ANSI
Replace	Este método sustituye un literal por otro en un objeto Char
RTrim	Elimina los blancos situados a la derecha en el objeto Char (expresión alfanumérica) y retorna dicho objeto modificado
StrRepeat	Este método repite el valor de un objeto Char, tantas veces como se indique. La longitud máxima del resultado es de 512 caracteres
SubString	Devuelve la cadena de caracteres de un objeto Char comenzando en un carácter dado y con una longitud dada
Trace	Este método muestra en una ventana el valor que tiene asignado el objeto
Trim	Elimina los blancos situados a la derecha y a la izquierda del objeto Char (expresión alfanumérica) y reduce cada grupo de blancos entre palabras a uno solo. Retorna el objeto modificado
Upcase	Convierte a mayúsculas el valor de un objeto Char (expresión alfanumérica) . Retorna el objeto modificado
VarLength	Devuelve la longitud con que ha sido definido un objeto Char (espacio en Bytes). Para una clase Char(20) devolverá siempre 20. Para una clase Char de longitud indefinida devolverá el espacio en Bytes que ocupa en ese momento el objeto

Conversores

La clase Char tiene conversores a las siguientes clases:

- Time
- Date
- Boolean
- Smallint
- Integer
- Decimal
- Numeric
- Money

9. Clase Date

Esta clase deriva de la clase Simple, es una clase instanciable. Esta clase permite almacenar valores de fecha.

Los tipos DATE son introducidos como una secuencia día, mes y año, con caracteres numéricos. El día puede representarse como el día del mes (1 ó 01, 2 ó 02, etc.). El mes se representa como un número (Enero: 1, Febrero: 2, etc.). El año se representa como un número de cuatro dígitos (0001 a 9999). En caso de mostrar dos dígitos para el año, CTSQL asume que el año es 19yy si la fecha del sistema es anterior al año 2000 y 20yy en caso contrario.

Se puede ordenar por una columna de tipo DATE y hacer una comparación cronológica entre dos columnas de tipo DATE.

Operadores de la Clase Date

- **Operadores aritméticos**

Operador	Funcionalidad
+ (suma)	Suma a una fecha un número indicado de días
- (resta)	Resta a una fecha un número indicado de días
++ (incremento)	El operador incremento suma un día a la fecha sobre la que se aplica
-- (decremento)	El operador decremento resta un día a la fecha sobre la que se aplica

- **Operadores aritméticos de asignación**

Operador	Funcionalidad
= (asignación)	Este método permite asignar un valor a un objeto Date
+= (suma)	Este método suma un número indicado de días a la fecha sobre la que se aplica
-= (resta)	Este método resta un número indicado de días a la fecha sobre la que se aplica

- **Operadores relacionales**

Operador	Funcionalidad
== (igualdad)	Este operador de igualdad chequea si dos objetos de tipo Date son iguales
(distinto)	Este operador chequea si dos objetos de tipo Date son distintos
(mayor)	Este operador chequea si un objeto de tipo Date es mayor que otro
= (mayor o igual)	Este operador chequea si un objeto de tipo Date es mayor o igual que otro
(menor)	Este operador chequea si un objeto de tipo Date es menor que otro
= (menor o igual)	Este operador chequea si un objeto de tipo Date es menor o igual que otro
BETWEEN	Este operador chequea si un objeto de tipo Date esta comprendido o no en el rango dado por otros dos
IN	Este operador chequea si un objeto de tipo Date es igual o no a alguno de los valores incluidos en una lista de valores

Métodos de la clase Date

Métodos	Funcionalidad
Day	Este método devuelve el día del mes correspondiente a una fecha determinada
DaysTo	Este método calcula el número de días comprendidos entre dos fechas
Month	Este método devuelve el mes en el que se encuentra una fecha determinada
Using	Este método realiza una conversión a Char formateada de expresiones de tipo fecha (Date)
Trace	Este método muestra en una ventana el valor que tiene asignado el objeto
WeekDay	Este método devuelve el día de la semana de una fecha determinada
Year	Este método devuelve el año en el que se encuentra una fecha determinada

Conversor

La clase Date tiene conversor a la clase Char

10. Clase Time

Esta clase deriva de la clase Simple, es una clase instanciable. Esta clase permite almacenar valores horarios desde 00:00:01 a 24:00:00.

A los objetos de esta clase se les asigna un valor como una secuencia de hora, minutos y segundos, sin separador entre ellos (tampoco espacios en blanco). Asimismo, permite ordenaciones, operaciones y comparaciones horarias entre dos objetos de la clase TIME.

Operadores de la clase Time

- **Operadores aritméticos**

Operador	Funcionalidad
+ (suma)	Suma a una hora un número determinado de segundos
- (resta)	Resta a una hora un número determinado de segundos
++ (incremento)	El operador incremento suma un segundo a la hora sobre la que se aplica
-- (decremento)	El operador decremento resta un segundo a la hora sobre la que se aplica

- **Operadores aritméticos de asignación**

Operador	Funcionalidad
= (asignación)	Este método permite asignar un valor a un objeto Time
+= (suma)	Este método suma un número determinado de segundos a la hora sobre la que se aplica
-= (resta)	Este método suma un número determinado de segundos a la hora sobre la que se aplica

- **Operadores relacionales**

Operador	Funcionalidad
== (igualdad)	Este operador de igualdad chequea si dos objetos de tipo Time son iguales
(distinto)	Este operador chequea si dos objetos de tipo Time son distintos
(mayor)	Este operador chequea si un objeto de tipo Time es mayor que otro
= (mayor o igual)	Este operador chequea si un objeto de tipo Time es mayor o igual que otro
(menor)	Este operador chequea si un objeto de tipo Time es menor que otro
= (menor o igual)	Este operador chequea si un objeto de tipo Time es menor o igual que otro
BETWEEN	Este operador chequea si un objeto de tipo Time esta comprendido o no en el rango dado por otros dos
IN	Este operador chequea si un objeto de tipo Time es igual o no a alguno de los valores incluidos en una lista de valores

Métodos de la clase Time

Método	Funcionalidad
Hour	Devuelve las horas del objeto Time
Minute	Devuelve los minutos del objeto Time
Second	Devuelve los segundos del objeto Time
SecondsTo	Este método calcula el número de segundos comprendidos entre dos objetos de la clase Time
Trace	Este método muestra en una ventana el valor que tiene asignado el objeto
Using	Este método realiza una conversión a Char formateada del objeto de tipo Hora (Time).

Conversores

La clase Time tiene conversor a la clase Char

11. Clase Boolean

Esta clase deriva de la clase Simple, es una clase instanciable. Esta clase permite almacenar objetos que solamente pueden tener valores TRUE o FALSE.

Operadores de la clase Boolean

- **Operadores lógicos**

Operador	Funcionalidad
AND	Devolverá TRUE si ambas condiciones son ciertas
OR	Devolverá TRUE si al menos una de las condiciones es cierta
XOR	Es el operador OR exclusivo entre dos condiciones. Devolverá TRUE si una de las condiciones es cierta y la otra falsa, devuelve FALSE en caso contrario
NOT	El operador negación invierte el resultado que se obtiene al evaluar una condición

- **Operadores relacionales**

Operador	Funcionalidad
== (igualdad)	Este operador de igualdad chequea si dos objetos de tipo Boolean son iguales
(distinto)	Este operador de igualdad chequea si dos objetos de tipo Boolean son distintos

- **Operador de asignación**

Operador	Funcionalidad
= (asignación)	Este operador permite asignar un valor a un objeto de tipo Boolean

Métodos de la clase Boolean

Método	Funcionalidad
Trace	Este método muestra en una ventana el valor que tiene asignado el objeto

12. Clase Enum

Esta clase deriva de la clase Simple, es una clase abstracta.. Esta clase permite almacenar objetos que pueden tener un solo valor entre una lista de valores posibles.

Un Enum no puede tener elementos repetidos y sus elementos están ordenados. Un Enum permite asignar identificadores nemónicos a valores enteros. El ejemplo clásico es la asignación de valores a los días de la semana:

Cada valor incluido en los corchetes en una declaración de tipo enumerado contiene un valor entero subyacente, determinado por la posición del valor en la lista.

Operador de la clase Enum

Método	Funcionalidad
= (asignación)	Este método permite asignar un valor a un objeto Enum

Método de la clase Enum

Método	Funcionalidad
Trace	Este método muestra en una ventana el valor que tiene asignado el objeto

13. Clase EnumSet

Esta clase deriva de la clase Simple, es una clase abstracta. Esta clase permite almacenar valores que representan la presencia de 0, 1 o mas elementos de un conjunto. En una clase EnumSet definida por el usuario se definirán los elementos que componen el conjunto.

Un EnumSet no puede tener elementos repetidos y sus elementos no están ordenados. Un EnumSet puede tener como máximo 32 elementos.

Sirve por ejemplo para almacenar los 'flags' de apertura de un fichero. o por ejemplo para almacenar el conjunto de permisos que tiene un usuario sobre las tablas de la base de datos: añadir, borrar, modificar y consultar.

Operador de la clase EnumSet

- **Operador de asignación**

Operador	Funcionalidad
= (asignación)	Este método permite asignar un valor a un objeto de tipo EnumSet
+ (suma)	Calcula la unión de dos objetos de tipo EnumSet

Método de la clase EnumSet

Método	Funcionalidad
Trace	Este método muestra en una ventana el valor que tiene asignado el objeto

14. Clase Complex

Esta clase es abstracta y virtual, de ella derivan las clases que almacenan los objetos que tiene una estructura compleja.

De la clase Complex derivan:

Clase Struct — Esta clase permite derivar clases con una estructura definida por el programador. Esta clase es abstracta.

Clase Window — De ella derivan todas las clases que pueden usar ventanas para comunicarse con el usuario final de la aplicación. Esta clase es virtual.

Clase Menu — Esta clase define toda la funcionalidad necesaria para el manejo menús de tipo “Pull-Down”, “Popup” o “Button-Menus” de una aplicación “Windows”. Esta clase es abstracta.

Clase Page — Esta clase define la funcionalidad de todos los objetos de tipo Page de una aplicación. Un objeto Page contiene fundamentalmente un componente gráfico denominado Template que define el formato gráfico de una pagina de un impreso. Un listado de una aplicación no es mas que un conjunto de formatos de pagina impresos en papel o mostrados por pantalla. Esta clase es abstracta.

Clase Module — Esta clase define la funcionalidad básica de un modulo Cosmos (programas, librerías, includes). Un modulo Cosmos define en sí mismo una clase derivada de esta clase virtual Module. Cuando se carga un módulo de una aplicación en memoria, el Runtime de Cosmos crea un objeto de su clase modulo.

Clase Stream — Esta clase define la funcionalidad necesaria para establecer canales de Entrada y/o Salida para leer y escribir respectivamente en ficheros o dispositivos del sistema operativo. Así mismo permite la conexión mediante TCP/IP con procesos remotos.

Clase SqlCursor — Esta clase permite obtener información almacenada en una Base de Datos. Un cursor define una tabla virtual mediante una instrucción SELECT de SQL. La Clase SqlCursor tiene métodos asociados que permite definir la SELECT y recorrer una a una las filas que retorna como resultado el Gestor de Base de Datos asociado al Cursor.

Clase SqlStatement — Esta clase permite ejecutar instrucciones SQL sobre una Base de Datos.

Clase SqlServer — Esta clase permite establecer comunicación con un SGBD (Sistema Gestor de Bases De Datos). Permite establecer conexiones locales o remotas (cliente-servidor) para acceder a una base de datos. Permite por lo tanto utilizar distintos gestores de bases de datos:

- Utilización del gestor de base de datos CSQL incluido en el propio producto. Esta posibilidad permite el acceso a Base de Datos locales, en red local o bien en cliente-servidor contra un servidor Unix o NT.
- Utilización de otros gestores de base de base de datos en cliente-servidor, mediante la tecnología Multiway de TransTOOLs.
- Acceso a base de datos utilizando tecnología ODBC.

Clase PrnDocument	Esta clase define la funcionalidad necesaria para imprimir documentos a través de una impresora definida en el sistema. Esta clase tiene los métodos necesarios para indicar la impresora que se desea utilizar, mandar un documento a un buffer de impresión, hacer una presentación preliminar, mandar el documento al administrador de impresión de Windows, cancelar la impresión de un documento en el administrador de impresión si este se encuentra activo, etc.
Clase DDE	Esta clase define la funcionalidad necesaria para establecer comunicaciones con otras aplicaciones Windows usando el protocolo DDE ("Dynamic Data Exchange") .
Clase Control	Esta clase encapsula las propiedades y métodos genéricos para manejar los controles de una screen de un Form y los controles de impresión de una página. Esta clase es abstracta.
Clase FormTable	Esta clase permite manipular los datos de las tablas del Form. Una tabla del Form suele estar asociada con una tabla de la base de datos, por tanto, esta clase encapsula la funcionalidad necesaria para agregar, modificar, hacer consultas a dichas tablas, etc. Esta clase es abstracta.

15. Clase Struct

Esta clase deriva de la clase Complex, es abstracta. Esta clase permite derivar clases con una estructura definida por el programador.

Para referenciar un miembro de una estructura en una expresión, se emplea una construcción de la forma:

nombre_estructura.miembro

El operador miembro "." permite seleccionar un miembro de una estructura.

Las estructuras ayudan a organizar datos complicados. Permiten tratar como unidad un conjunto de objetos relacionados lógicamente unos con otros, en lugar de tratarlos como entidades independientes.

16. Clase Window

Esta clase deriva de la clase Complex, es virtual. De ella derivan todas las clases que pueden usar ventanas para comunicarse con el usuario final de la aplicación.

De la clase Window derivan:

Clase Form	Esta clase deriva de la clase Window, es abstracta. Esta clase define la funcionalidad de todos los objetos de tipo Form de una aplicación. Un Form contiene fundamentalmente un componente gráfico denominado Screen que permite dialogar con el usuario final mostrando información de la aplicación y recogiendo la respuesta de éste. Así mismo un Form de Cosmos tiene predefinida toda la funcionalidad necesaria para la edición directa de tablas de una Base de Datos.

17. Clase Form

Esta clase deriva de la clase Window, es abstracta. Esta clase define la funcionalidad de todos los objetos de tipo Form de una aplicación. Un Form contiene fundamentalmente un componente gráfico denominado Screen que permite dialogar con el usuario final mostrando información de la aplicación y recogiendo la respuesta de éste. Así mismo un Form de Cosmos tiene predefinida toda la funcionalidad necesaria para la edición directa de tablas de una Base de Datos .

La clase Form será la clase padre de las diferentes clases de formato de pantalla que defina el programador. Una vez definida una clase de tipo Form, el programador podrá definir objetos de esta clase en su programa si esta no ha sido definida como abstracta.

Esta clase tiene predefinidos una serie de métodos que permiten realizar tareas sencillas, como añadir, borrar o modificar filas de una tabla sin necesidad de programarlas. También permite realizar tareas más complejas, por ejemplo, el mantenimiento de tablas de la base de datos enlazadas mediante una relación “1 a N”.

Por lo general se hará uso de un Form siempre que se desee establecer un dialogo con el usuario.

Métodos de la clase Form

Método	Funcionalidad
AcceptEdit	Este método da validez a los datos introducidos en el formulario de la tabla activa del Form y inhabilita la edición en dicha tabla. (Sólo en modo edición)
AttachServer	Este método permite asociar un servidor al Form, cualquier operación automática que se ejecute en el Form sobre una base de datos, se hará contra el servidor asociado
CancelEdit	Este método cancela la edición, no da validez a los datos introducidos en el formulario de la tabla activa del Form y desactiva la edición en dicha tabla. (Sólo en modo edición).
Close	Este método permite cerrar un Form
Control	Este método retorna por referencia el control del Form que tiene de identificador el indicado como parámetro
CurrentFocus	Este método devuelve el control de la screen del Form que tiene el foco
DisableCommand	Este método habilita o inhabilita un comando en el Form.
DrawMenuBar	Este método repinta la barra de menú del Form. Por ejemplo, será necesario utilizarlo cuando se añade dinámicamente una opción al menú raíz del Form si este se presenta como “Pulldown”.
EditingTable	Este método retorna la tabla en edición del Form. (Sólo en modo edición).
EditQueryLike	Este método activa la edición de la tabla maestra del Form, sin mostrar los valores por defecto, y permite al usuario introducir los valores en los campos sobre los que se quiere realizar una consulta por condiciones de igualdad (QueryLike). (Sólo en modo edición)
Exit	Este método permite cerrar un Form ejecutado mediante el método “Run” (Form Modal) y devolver un valor de retorno.
FocusField	Este método devuelve una la variable asociada al control de la screen del Form que tiene el foco.
FocusTable	Este método devuelve la tabla del Form (objeto FormTable) que contiene el control que tiene el foco.
Frame	Este método devuelve el control asociado a la ventana del Form.

Método	Funcionalidad
GetQueryByForm	Muestra un cuadro de diálogo que permitirá al usuario construir una condición de búsqueda sobre la tabla maestra del Form
GetQueryLike	Este método permite generar una condición de búsqueda sobre la tabla maestra del Form, tomando como condiciones de igualdad los valores de los controles de la screen asociados a dicha tabla.
Hwnd	Este método devuelve un número entero (INTEGER o DWORD), que se corresponde con el manejador (handle) para Windows de la ventana principal del Form.
InEditQueryLike	Este método indica si se ha esta en edición para hacer un QueryLike en la tabla maestra del Form mediante el método "EditQueryLike". (Sólo en modo edición).
IsCommandDisabled	Este método indica si un comando esta habilitado o inhabilitado en el Form
IsOpen	Este método indica si el Form esta abierto
LosingFocus	Este método permite consultar qué control de la screen está perdiendo el foco cuando se esta procesando el evento On Enter de otro control.
MasterTable	Este método retorna la tabla maestra del Form
Maximize	Amplia la ventana del Form hasta su tamaño máximo
MessageBox	Este método muestra un cuadro de mensaje para pedir una respuesta al usuario e inhabilita el Form hasta que se cierra dicho cuadro de mensaje
Minimize	Minimiza la ventana del Form
NextFocus	Este método devuelve una referencia al siguiente control del Form que admite el foco, o al anterior si se le indica en el parámetro
Open	Este método muestra en pantalla la screen de un Form de forma no modal.
Query	Este método permite la consulta de filas sobre la tabla maestra del Form
QueryByForm	Este método permite la consulta de filas sobre la tabla maestra del Form. Muestra un cuadro de diálogo que permitirá al usuario construir la condición de búsqueda.
QueryLike	Este método permite la consulta de filas sobre la tabla maestra del Form, tomando como condiciones de igualdad los valores de los controles de la screen asociados a dicha tabla
ReceivingFocus	Este método permite consultar qué control de la screen esta recibiendo el foco cuando se esta procesando el evento On Exit de otro control.
Reset	Este método permite borrar la lista en curso de todas las tablas del Form. Pone la tabla maestra en estado New.
Restore	Este método restaura la ventana del Form a su tamaño y posición anteriores a la maximización o minimización.
Run	Este método permite realizar una ejecución modal de un Form, mostrando en pantalla su screen.
SendCommand	Este método permite ejecutar un comando del Form
SetEditMode	Este método permite activar o desactivar el modo edición
SetMenu	Este método permite cambiar de menú del Form dinámicamente
Sql	Este método retorna el objeto SqlServer que tiene asociado el Form.
SqlErrMsg	Devuelve el string correspondiente al código de error actual del servidor SQL asociado al Form.
SqlError	Devuelve el código de error actual del servidor SQL asociado al Form.
Table	Este método retorna por referencia la tabla del Form que tiene de nombre el indicado como parámetro.

18. Clase Menu

Un menú Cosmos es un objeto gráfico que permite al usuario de una aplicación elegir entre una serie de opciones disponibles.

Métodos de la clase Menu

Método	Funcionalidad
Add	Permite añadir opciones o submenús a un menú existente mientras se ejecuta la aplicación, para proporcionar más información u opciones al usuario. Este método permite añadir ítems a un menú de forma dinámica.
Delete	Este método permite borrar una opción o submenú de un menú
Option	Este método retorna el submenú u opción que tiene un determinado identificador
Track	Este método muestra un menú en pantalla como “Popup Menu”.

19. Clase Page

En la mayoría de las aplicaciones, gran parte de los resultados que éstas proporcionan se realizan mediante informes impresos, como por ejemplo facturas, etiquetas, cartas, listas, etc.

Para la generación de estos informes, Cosmos cuenta con dos clases especializadas Page y PrnDocument que definen la funcionalidad necesaria para su generación.

La Clase Page permite diseñar en su definición, un formato de impresión mediante un editor gráfico. Un informe impreso se construirá enviando uno o varios de estos formatos a una impresora por medio de un objeto PrnDocument.

Métodos de la clase Page

Método	Funcionalidad
Calculate	Recalcula las dimensiones de la página y de sus controles, así como el número de elementos visibles que admite cada control de impresión de tipo grupo. Este método debe usarse si el tamaño de la página al ser editada varía respecto al de la impresora seleccionada en ejecución.
Clear	Inicializa los datos de la página a sus valores por defecto
LoadVars	Este método actualiza las variables de la página de impresión con el valor que tengan sus controles asociados.
PrintVars	Este método actualiza los controles de la página de impresión con los valores de sus variables asociadas

20. Clase Module

Un módulo es una unidad de ejecución que contiene componentes (clases, objetos, etc.) que definen el funcionamiento de la aplicación. Todos los módulos de programación de Cosmos tienen la misma estructura definida. Sin embargo podemos agrupar los módulos de un proyecto basándonos en su funcionalidad. Existen tres tipos de módulos Cosmos: Programas, librerías e includes.

- Library: Una librería es un módulo que exporta (permite su uso desde otros módulos) sus métodos públicos. Sus métodos privados y protegidos, no se exportan y por tanto no pueden ser utilizados en el módulo que incluya la librería.
- Include: Un Include es un módulo que exporta sus métodos públicos, clases públicas, objetos globales públicos y constantes. Así mismo un include exporta los métodos, clases, etc, públicos que incluye a su vez.
- Program: Un programa no exporta nada de su contenido. Desde otro módulo sólo se puede acceder a él cargándolo en memoria por medio de las instrucciones Run y Load. La comunicación entre programas se establece a través de los parámetros pasados a la función Main del módulo llamado.

En Cosmos los módulos en sí son objetos que pertenecen a una clase derivada de la clase predefinida Module. Esta clase es virtual.

Métodos de la clase Module

Método	Funcionalidad
Beep	Este método hace sonar una alarma correspondiente a un nivel de alerta dado. El sonido para cada uno de los niveles de alerta está identificado en el panel de control de Windows.
Copy	Este método permite copiar (duplicar) un fichero del con otro nombre
ChDir	Este método cambia el directorio en curso
CosmosDir	Este método indica el directorio donde se encuentra instalado Cosmos
Delete	Este método permite borrar un fichero
DesktopSize	Este método retorna las dimensiones del escritorio de Windows en pixeles
Ending	Este método puede ser definido en cualquier módulo COOL y se ejecutará cuando se descargue el módulo de memoria
ErrorLevel	Este método permite modificar el nivel de error de la aplicación asociada al módulo
Exec	Ejecuta un programa Windows o MS-DOS sin esperar a que termine
Exit	Este método permite abortar la ejecución de la aplicación o del módulo de la aplicación que se esté ejecutando en ese momento
FileSize	Este método devuelve el tamaño, en bytes, de un fichero
GetCursorPos	Este método devuelve la posición del cursor en la pantalla
GetCwd	Este método devuelve el directorio en curso
GetEnv	Este método devuelve el valor de la variable de entorno que se le pasa como parámetro
GetIni	Este método permite obtener el valor correspondiente a una entrada de una sección de un fichero de configuración
Hinstance	Este método retorna la instancia Windows del programa Cosmos que se está ejecutando
HMS	Este método devuelve un objeto de tipo Time. Recibe como parámetros los tres componentes de la hora
IsLoaded	Este método permite consultar si un módulo Cosmos esta cargado en memoria

Método	Funcionalidad
License	Este método retorna el número de licencia de Cosmos
Load	Este método permite cargar en memoria un módulo Cosmos que pertenezca al proyecto en curso
MDY	Este método devuelve un objeto de tipo Date. Recibe como parámetros los tres componentes de la Fecha
MkDir	Este método permite crear un directorio
Move	Este método permite renombrar y mover de directorio un fichero
MsgText	Devuelve el texto del mensaje del fichero de mensajes que se le indica
MsgTextDef	Devuelve el texto del mensaje del fichero de mensajes que se le indica.
MessageBox	Este método muestra un cuadro de mensaje para pedir una respuesta al usuario
Ok	Este método muestra un cuadro de mensaje con los botones Ok y Cancel para pedir una respuesta al usuario
ProjectDir	Este método indica el directorio donde se encuentra el proyecto en curso
PutEnv	Este método permite añadir o modificar el valor de la variable de entorno que se le pasa como parámetro
Random	Devuelve un número aleatorio dentro de un intervalo que se pasa como parámetro
Rgb	Este método calcula el valor entero correspondiente a un color dado en formato RGB
RmDir	Este método permite borrar un directorio
Run	Este método permite ejecutar otro programa Cosmos. Este método carga en memoria el programa y al finalizar su ejecución lo descarga
ScreenSize	Este método retorna las dimensiones de la pantalla en pixeles
SetCursorPos	Este método mueve el cursor del ratón a la posición indicada
SetIni	Este método permite modificar el valor correspondiente a una entrada de una sección de un fichero de configuración
Sql	Cosmos tiene definido un servidor por defecto: Es un Objeto de la clase SqlServer al que se accede por medio de este método que puede ser utilizado en cualquier momento por el programador.
Start	Este método puede ser definido en cualquier módulo COOL y se ejecutará cuando se cargue el módulo en memoria
System	Ejecuta un programa Windows o MS-DOS y espera a que termine.
TestFile	Este método comprueba si un fichero es de un tipo determinado o bien si tiene unos permisos concretos
TreeWalk	Este método muestra un cuadro de diálogo que permite cambiar de directorio para elegir un fichero. Este cuadro de diálogo le permite moverse por los diferentes directorios y unidades de disco para seleccionar el fichero deseado
UnLoad	Este método permite descargar de memoria un módulo Cosmos
WindowsDir	Este método indica el directorio donde se encuentra instalado Windows
Yes	Este método muestra un cuadro de mensaje con los botones Yes y No para pedir una respuesta al usuario
Yield	Este método cede el control a Windows para que procese los mensajes pendientes. Permite el repintado de ventanas que han sido modificadas previamente. Se debe usar por ejemplo para refrescar en un bucle los controles modificados en el mismo si se quieren visualizar antes de su finalización (Control de tipo porcentaje, simulación de controles dinámicos, etc.)

21. Clase SqlCursor

Esta clase derivada de la clase Complex es derivable e instanciable. permite declarar y ejecutar cursores del SQL sobre una base de datos determinada. Un cursor es una tabla virtual asociada al programa que lo define, cuyo contenido será el resultado de una instrucción SELECT utilizada en su definición.

Cuando en un programa se emplea una instrucción SELECT para realizar una consulta (query), pueden ocurrir dos cosas: que el resultado (es decir, la tabla derivada) incluya una única fila de la base de datos, o bien que incluya varias filas. En este último caso es donde se hace necesaria la utilización de un cursor, en el que el SQL incluirá todas las filas resultado de la instrucción SELECT. Evidentemente, también se dispone de los métodos necesarios para poder moverse a través de las distintas filas del cursor.

Métodos de la clase SqlCursor

Método	Funcionalidad
AddGroup	Este método se utiliza para definir de uno en uno los agrupamientos en una instrucción SELECT previamente preparada. Este método puede utilizarse en lugar del método GroupBy o después de este
AddTotal	Este método se utiliza para definir de una en una las columnas a totalizar en los grupos. Puede utilizarse en lugar del método Totalize, o después de este
AttachServer	Asocia un servidor a la instrucción SQL, cuando se ejecute el cursor se hará contra el servidor asociado
BreakLevel	Este método se utiliza para conocer el estado de la ruptura de grupos en un momento determinado. Se utiliza después de la ejecución del método Fetch
Close	Este método se utiliza para cerrar un cursor abierto previamente por medio del método Open.
Error	Indica el código del último error producido en el servidor.
ErrMsg	Muestra el string asociado al último error producido en el servidor
Execute	Ejecuta el cursor previamente preparado con el método Prepare. Es una instrucción para el tratamiento de instrucciones SQL de forma dinámica
Fetch	Permite hacer desplazamientos secuenciales dentro de un cursor. Este método lee una fila de la tabla derivada devuelta por un cursor abierto previamente por medio de una instrucción Open
FetchFirst	Este método lee la primera fila de la tabla derivada devuelta por un cursor abierto previamente por medio de una instrucción Open
FetchLast	Este método lee la última fila de la tabla derivada devuelta por un cursor abierto previamente por medio de una instrucción Open
FetchNext	Este método lee la fila siguiente a la fila en curso de la tabla derivada devuelta por un cursor abierto previamente por medio de una instrucción Open
FetchPrev	Este método lee la fila anterior a la fila en curso de la tabla derivada devuelta por un cursor abierto previamente por medio de una instrucción Open
Found	Indica si el cursor a procesado alguna fila o ninguna
Free	Libera la memoria alocada por el cursor para la sentencia en curso. Es conveniente liberar la memoria cuando no se va a utilizar más esta sentencia
GroupAverage	Este método se utiliza para obtener la media aritmética de todos los elementos que tiene un grupo hasta el momento

Método	Funcionalidad
GroupBy	Este método se utiliza para definir una lista de agrupamientos en una instrucción SELECT previamente preparada
GroupCount	Este método se utiliza para obtener el número de filas que tiene un grupo hasta el momento
GroupMaximun	Este método se utiliza para obtener el máximo de todos los elementos que tiene un grupo hasta el momento
GroupMinimun	Este método se utiliza para obtener el mínimo de todos los elementos que tiene un grupo hasta el momento
GroupSum	Este método se utiliza para obtener la suma de todos los elementos que tiene un grupo hasta el momento
Into	Permite indicar la lista de variables globales del programa que se utilizarán para recoger los valores de las columnas del cursor
Locked	Este método devuelve TRUE si la fila en curso de un cursor esta bloqueada por otro usuario. En caso contrario su valor será FALSE. Dicho cursor tiene que declararse con la cláusula NOWAIT.
Open	Este método abre un cursor previamente declarado con el método prepare, es decir, ejecuta la instrucción SELECT asociada a el cursor
Prepare	Este método permite construir una instrucción de tipo SELECT del SQL de forma dinámica
Putname	Este método permite dar un nombre al curso para que el CTSQL pueda identificar el cursor y ejecute la instrucción update sin errores
ScrollFetch	Permite hacer desplazamientos relativos dentro de un cursor. Este método lee una fila de la tabla derivada devuelta por un cursor abierto previamente por medio de una instrucción Open.
Totalize	Este método se utiliza para definir la lista de columnas que se desean totalizar en los grupos

22. Clase SqlStatement

Esta clase derivada de la clase Complex es derivable e instanciable es la que permite la comunicación con el SQL. En esta clase se almacenan los siquientes objetos:

- Intrucciones SQL de consulta : SELECT.
- Instrucciones para el tratamiento dinámico del SQL. Son instrucciones de tipo SQL construidas mediante literales y expresiones para posteriormente ejecutarlas. Entre todas las instrucciones propias del SQL existen dos que permiten realizar esta operación: Prepare, Execute.

Métodos de la clase SqlStatement

Método	Funcionalidad
AttachServer	Asocia un servidor a la instrucción SQL, cuando se ejecute la instrucción se hará contra el servidor asociado
Error	Indica el código del último error producido.
ErrMsg	Muestra el string asociado al último error producido en el servidor
Execute	Ejecuta una instrucción del SQL previamente preparada con el método Prepare. Es un método para el tratamiento de instrucciones SQL de forma dinámica
Free	Libera la memoria alocada para la sentencia en curso. Es conveniente liberar la memoria cuando no se va a ejecutar más esta sentencia

Método	Funcionalidad
Prepare	Este método permite construir una instrucción SQL de forma dinámica (posibilidad de concatenar expresiones) para la comprobación de su sintaxis
Rows	Retorna el número de filas procesadas en la última operación de INSERT, DELETE, UPDATE, SELECT, LOAD realizada
SqlExec	Prepara y ejecuta una instrucción SQL. Sirve para el tratamiento de instrucciones SQL de forma dinámica

23. Clase SqlServer

Esta clase derivada de la clase Complex es derivable e instanciable permite establecer conexiones locales o remotas (cliente-servidor) para acceder a una base de datos.

Permite por lo tanto utilizar distintos gestores de bases de datos:

- Utilización del gestor de base de datos CTSQL incluido en el propio producto.Esta posibilidad permite la utilización del producto tanto en modo monousuario, en una red de PCs y en arquitectura cliente servidor.
- Utilización de los gestores de base de datos de Informix, Oracle, Ingres y Sybase, mediante la tecnología Multiway de TransTOOLs. Las aplicaciones son instaladas en modo cliente servidor.

Esta clase tiene los métodos necesarios para ejecutar cualquier instrucción SQL: carga y descarga de datos desde/hacia ficheros ASCII (Load/Unload), etc. También permite la ejecución de una instrucción SQL previamente almacenada en un fichero o producida como resultado de una expresión.

Métodos de la clase SqlServer

Método	Funcionalidad
AttachConnection	Este método permite indicar la conexión que se desea establecer con el servidor
Connect	Establece la conexión con el servidor y abre la base de datos. Si se obvia la llamada a este método la conexión se realizará automáticamente al realizar la primera operación contra la base de datos
DettachConnection	Desconecta el servidor y elimina la definición del servidor, pasando a ser la definición del servidor la del entorno actual
Disconnect	Fuerza el cierre y la desconexión de la base de datos, sin perder la definición del servidor
Error	Indica el código del último error producido
ErrMsg	Muestra el string asociado al último error producido.
LastSerial	Este método devuelve el último valor que se le ha dado a una columna de tipo SERIAL en una tabla. Será el valor que ha tomado dicha columna en la última instrucción INSERT del SQL que se ha ejecutado sobre una tabla
Load	Este método carga datos provenientes de un fichero ASCII sobre una tabla de la base de datos.
Putenv	Este método permite cambiar el valor de las variables de entorno del servidor en conexiones remotas en el sistema operativo UNIX
Rows	Retorna el número de filas procesadas en la última operación de INSERT, DELETE, UPDATE, SELECT, LOAD realizada en el servidor
Select	Este método permite ejecutar una instrucción SELECT del SQL recogiendo la información de la primera fila encontrada

Método	Funcionalidad
SelectWindow	Este método muestra una ventana con información procedente de una instrucción SELECT permitiendo extraer valores de dicha ventana sobre una o varias variables del programa
SqlDescribe	Dada una instrucción SELECT del SQL, devuelve un literal con la lista de columnas, sus tipos y, opcionalmente, sus atributos, tal y como están definidos en la base de datos. La utilidad principal de este método es poder crear a partir del valor retornado una tabla temporal del mismo tipo que la SELECT que recibe como parámetro
SqlExec	Prepara y ejecuta una instrucción SQL. Sirve para el tratamiento de instrucciones SQL de forma dinámica
SqlFile	Este método ejecuta una o varias instrucciones del SQL almacenadas en un fichero cuya extensión sea .sql. En caso de que la instrucción SQL a ejecutar sea una SELECT, se mostrarán por defecto todas las filas de su tabla derivada en una ventana.
SqlFileToFile	Este método ejecuta una o varías instrucciones del SQL almacenadas en un fichero cuya extensión sea .sql. En caso de que la instrucción SQL a ejecutar sea una SELECT, el resultado de su ejecución se obtiene en un fichero
SelectToFile	Este método ejecuta cualquier instrucción SELECT del SQL. El resultado de su ejecución se obtiene en un fichero
Unload	Este método permite la descarga de información recogida en las tablas de la base de datos. Dicha información se almacena en un fichero ASCII (de texto).

24. Clase PrnDocument

La clase PrnDocument nos proporciona los servicios necesarios para crear informes conectando con una impresora del sistema y enviándole las páginas que correspondan.

Un PrnDocument se asocia a una impresora del sistema mediante el método Setup que muestra un diálogo permitiendo elegir una impresora definida en WINDOWS y el método AttachPrinter que recibe como parámetro el nombre de esta impresora.

Inicialmente un PrnDocument está asociado a la impresora predeterminada de WINDOWS.

La clase PrnDocument permite así mismo mostrar un informe por pantalla en modo “preview”.

Métodos de la clase PrnDocument

Método	Funcionalidad
AbortPrinter	Cancela la impresión del documento si este se encuentra en el administrador de impresión. Limpia el buffer de páginas y cierra la ventana de preview si esta se encuentra activa.
AttachPrinter	Permite asociar el dispositivo de impresión (impresora) que se desea utilizar
Clear	Elimina las páginas de buffer del documento sin enviarlas al administrador de impresión e inicializa el contador de páginas
ClientSize	Este método retorna las dimensiones del área de trabajo (altura y anchura) del papel seleccionado en la impresora asociada al documento
ClosePrinter	Cierra el documento abierto previamente con OpenPrinter

Método	Funcionalidad
Flush	Vuelca sobre el documento de impresión el buffer de páginas de impresión, en el cual se han ido almacenando las páginas mediante el método SendPage
GetNumCopies	Este método permite consultar el número de copias que se han pedido de cada página del documento
GetPageNum	Indica el número de página que se esta procesando del documento de impresión
HideView	Esconde la ventana de Preview si esta se encuentra activa.
NumCopies	Permite indicar el número de copias que se desean de cada página del documento. Por defecto es 1
OpenPrinter	Abre el documento de impresión para su salida por impresora
Orientation	Este método permite consultar el tipo de orientación de la página que se tiene seleccionado para la impresora
PaperSize	Este método retorna las dimensiones (altura y anchura) del papel seleccionado en la impresora asociada al documento
Preview	Muestra la presentación preliminar del documento de impresión. Se muestra la ventana de preview de forma “modal”, esto es, el programa se detiene hasta que el usuario la cierra
SendPage	Envía una página al buffer del documento de impresión donde se va almacenando antes de enviarlo a la impresora o antes de hacer la presentación preliminar. Refresca la ventana de Preview si esta se encuentra visible
SetBufferPages	Indica el número máximo de páginas que puede alojar el buffer del documento de impresión
SetDocName	Asigna un nombre al documento de impresión para su identificación en el administrador de impresión de Windows. Este nombre se utiliza también como título de la ventana de la presentación preliminar
SetOrientation	Este método permite modificar el tipo de orientación de la página que se tiene seleccionado para la impresora
SetPageNum	Permite iniciar la cuenta del número de página a un número determinado
Setup	Este método muestra el dialogo de selección de impresoras y asocia al documento el dispositivo de impresión (impresora) que se seleccione
ShowView	Muestra la ventana de presentación preliminar del documento de impresión de forma “no modal”, esto es, no se detiene la ejecución del programa
WritePage	Equivale a SendPage y Flush. Si el documento ha sido abierto, no almacena la página en el buffer de almacenamiento de páginas sino que la envía directamente al dispositivo de impresión asociado al PrnDocument, esto es, la envía directamente al administrador de impresión de Windows

25. Clase DDE

El protocolo de comunicaciones entre diferentes aplicaciones Windows (DDE, Dynamic Data Exchange) permite a los programadores acceder desde el lenguaje de cuarta generación de Cosmos (COOL) a cualquier servidor DDE de Windows.

Frente a la utilización del portapapeles y de las librerías dinámicas (que cumplen igualmente este cometido), la incorporación del mecanismo DDE supone una gran mejora en el intercambio dinámico de datos entre aplicaciones.

DDE es un protocolo, basado en mensajes, para intercambiar datos entre distintos programas Windows. Gracias a esta funcionalidad los programadores podrán acceder desde el lenguaje COOL a cualquier servidor DDE de Windows, integrando en una arquitectura cliente-servidor las aplicaciones desarrolladas con Cosmos y los diferentes paquetes del mercado que incluyen el DDE entre sus características (Microsoft Excel, Microsoft Word, Lotus, etc.).

Para obtener rendimiento del protocolo DDE hay que conocer perfectamente la aplicación servidor a la cual enviamos y solicitamos información.

Métodos de la clase DDE

Método	Funcionalidad
Initiate	Este método abre dinámicamente un canal de comunicación para el intercambio dinámico de datos
Execute	Este método provoca la ejecución de un comando en la aplicación con la que previamente se ha iniciado la comunicación
Poke	Este método se encarga de enviar los datos a la aplicación con la que mantenemos la comunicación
Terminate	Finaliza de la comunicación con la aplicación

26. Clase FormTable

Esta clase deriva de la clase Complex. Esta clase virtual permite manipular los datos de las tablas del Form. Una tabla del Form suele estar asociada con una tabla de la base de datos, por tanto, esta clase encapsula la funcionalidad necesaria para agregar, modificar, hacer consultas a dichas tablas, etc.

Una tabla es un conjunto de columnas no necesariamente distintas. Una tabla se define con un nombre y la definición de sus correspondientes columnas. Una tabla representa una entidad de la base de datos.

Un Form puede mantener más de una tabla, siempre y cuando existe una relación entre ellas (join).

La clase FormTable tiene los métodos necesarios para:

- Grabar filas nuevas.
- Modificar las filas de la tabla que componen la lista en curso del Form.
- Borrar la fila en curso de la tabla.
- Recorrer la lista en curso del Form.
- Consultar el estado de las tabla del Form.
- Consultar el número de filas de la lista en curso.

Métodos de la clase FormTable

Método	Funcionalidad
Add	Este método permite añadir una fila a la tabla con los datos que se hayan introducido previamente en la Screen.
ChildTable	Este método retorna la tabla de líneas de una tabla
CurrentRow	Este método retorna el número de la fila en curso de la lista activa del Form
Delete	Este método permite borrar la fila activa de la lista en curso del Form

Método	Funcionalidad
DisableCommand	Este método habilita o inhabilita un comando de una tabla
EditNew	Este método entra en modo edición para añadir, Limpia los valores de los campos asociados a la tabla activa del Form mostrando los valores por defecto
EditTable	Este método entra modo edición para la tabla que llama al método. Si esta editando para modificar se bloquea la fila en curso
EditUpdate	Este método entra en modo edición para modificar y bloquea la fila en curso de la tabla activa del Form
First	Este método permite seleccionar como fila activa la primera de la lista en curso del Form
GetSqlName	Este método retorna el nombre SQL de la tabla
IsCommandDisabled	Este método indica si un comando esta habilitado o inhabilitado para la tabla que llama al método.
Last	Este método permite seleccionar como fila activa la última de la lista en curso del Form
LockRow	Este método permite bloquear o desbloquear la fila en curso de la tabla
Modified	Este método permite consultar si determinados campos de la tabla en la Screen han sido modificados.
Next	Este método permite seleccionar como fila activa la siguiente a la actual de la lista en curso del Form.
NextTable	Este método retorna la tabla del Form siguiente a la que llama al método, siguiendo el orden en el que están definidas las tablas en la estructura en árbol del Form
New	Este método permite limpiar los datos mostrados en la lista activa del Form y los inicializa con los valores por defecto
NumRows	Este método devuelve el número de filas de la lista en curso del Form
OrderBy	Este método permite modificar la lista de columnas de ordenación definida en las propiedades de la tabla.
ParentTable	Este método retorna la tabla de cabecera de una tabla determinada
Prev	Este método permite seleccionar como fila activa la inmediata anterior a la actual en la lista en curso del Form
Query	Este método permite la consulta de filas sobre la tabla del Form, permitiendo el refresco de una tabla dependiente, cuando el contenido de esta ha sido modificado por programa
Recall	Este método restaura los valores de la fila en curso con los que tuviera antes de entrar en edición
SendCommand	Este método permite ejecutar un comando de tabla
SetSqlName	Este método permite modificar el nombre SQL de la tabla
SqlModified	Este método permite consultar si alguna de las columnas definidas como Is column de la fila en curso de la tabla han sido modificadas
Status	Este método indica el estado de la tabla
Update	Este método permite modificar la fila activa de la lista en curso de un Form

27. Clase Control

Cosmos ha sido diseñado con el fin de proporcionar al programador acceso a todos los controles de Windows de forma sencilla, sin requerir el conocimiento de los cientos de funciones de interfaz gráfico de las librerías de Windows ni de su complejo funcionamiento.

Esta clase derivada de la clase Complex es virtual y encapsula las propiedades y métodos genéricos para manejar:

- Los controles de una screen de un Form.
- Los controles de impresión de una página.

De esta clase derivan:

Clase SimpleControl Esta clase encapsula las propiedades y métodos para el manejo de controles gráficos de Form y Page.

Métodos de la clase Control

Método	Funcionalidad
CoordToScreen	Este método convierte coordenadas relativas al control en coordenadas absolutas de pantalla
GetFramePos	Este método devuelve las coordenadas del control relativas al área de cliente de la ventana principal del Form
GetPos	Este método retorna las coordenadas del control respecto al área de cliente de su padre
GetScreenPos	Este método devuelve las coordenadas absolutas del control en la pantalla
GetSize	Este método retorna las dimensiones (altura y anchura) del control
Move	Este método permite cambiar la posición de un control dentro del área de cliente de su padre.
Name	Este método devuelve el identificador del control.
Next	Este método sirve para recorrer la estructura de controles de un Form o una página de impresión. Dependiendo del tipo de relación que se le pasa como parámetro, devuelve una referencia al control padre, hijo, siguiente o anterior.
ScreenToCoord	Este método convierte coordenadas absolutas de pantalla en coordenadas relativas al control que llama al método
Size	Este método permite modificar el tamaño de un control
Type	Este método retorna el tipo del control

28. Clase SimpleControl

Esta clase virtual derivada de la clase Control encapsula las propiedades y métodos para el manejo de controles gráficos de Form y Page.

Métodos de la clase SimpleControl

- **Métodos de la clase Control aplicables a cualquier control:**

Método	Funcionalidad
Count	Este método permite obtener el número de elementos que tiene un control de tipo lista (Drop edit, Drop list., List box) o el número de filas que tiene un control Group de una página de impresión o un control Grid de una screen
Hwnd	Este método devuelve un objeto Integer, que se corresponde con el manejador (handle) para Windows de la ventana del control
SetFocus	Este método permite asignarle el foco al control. Solo es aplicable en los controles de Form.

- **Métodos para los controles Edit Field multilinea**

Método	Funcionalidad
SetEditSel	Este método permite modificar la selección en el texto de un control Edit Field multilínea

- **Métodos para los controles Edit Field y Drop edit**

Método	Funcionalidad
GetEditSel	Este método indica los caracteres que hay seleccionados en el texto de un control Edit Field o DropEdit

- **Métodos para manejar los controles de tipo lista (List box, Drop edit y Drop list):**

Método	Funcionalidad
AddString	Este método permite añadir un elemento a un control de tipo lista: Drop edit, Drop list o List box que no tenga estructura en árbol
AddTreeString	Este método permite añadir un elemento a los controles de tipo List box con estructura en forma de árbol.
DeleteString	Este método permite borrar un elemento de un control de tipo: Drop edit, Drop list o List box.
FindCol	Este método retorna el índice del primer elemento de un control de tipo Drop edit, Drop list o List box en el que una columna determinada es igual al texto determinado
FindString	Este método sólo se puede aplicar a un control de tipo Drop edit, Drop list y List box. Retorna el primer elemento de la lista que empiece por el conjunto de caracteres especificado
FindStringExact	Este método solo se puede aplicar a un control de tipo Drop edit, Drop list y List box. retorna el índice del primer elemento de la lista que coincida exactamente con el conjunto de caracteres especificado
GetColumnAlign	Este método devuelve el tipo de alineamiento de una columna determinada en una lista (List box) de tipo Column list.

Método	Funcionalidad
GetColumnChars	Este método devuelve el ancho de presentación en caracteres (valor medio) de una columna determinada en una lista (List box) de tipo Column list.
GetListColumnInto	Este método permite recoger el valor de una columna determinada de un elemento de una lista (List box).
GetListColumnText	Este método devuelve como objeto Char el valor de una columna determinada de un elemento de una lista (List box).
GetListInfo	Un elemento de una lista puede almacenar, además de un literal, le valor de un objeto Integer para información adicional. Este método permite consultar este valor para un elemento determinado de un control de tipo List box
GetListInto	Este método permite cargar una fila de un control de tipo List box de tipo Column list en las variables into que se pasan como parámetro
GetListText	Este método permite obtener el texto asociado a un elemento de un control: Drop edit, Drop list, List box
GetTreeFirstChild	Este método solo se puede aplicar a un control List box con estructura en árbol. Devuelve el índice del primer elemento de un nodo determinado
GetTreeNext	Este método solo se puede aplicar a un control de tipo List box con estructura en árbol. Devuelve el índice del siguiente elemento de un nodo determinado
GetTreeParent	Este método devuelve el índice del nodo padre de un elemento de un control List box con estructura en árbol
InsertString	Este método permite insertar un elemento a un control de tipo lista: Drop edit, Drop list, List box
LoadSelect	Este método carga en un control List box de tipo Sql el resultado de una determinada instrucción SELECT
SetColumnAlign	Este método permite modificar el tipo de alineamiento de una determinada columna en un control List box de tipo Column list.
SetColumnChars	Este método permite modificar el ancho de presentación en caracteres (valor medio) de una determinada columna en un control List box de tipo Column list.
SetListInfo	Un elemento de un control de tipo List box puede almacenar, además de un literal, el valor de un objeto Integer con información adicional. Este método permite modificar este valor para un elemento determinado de la List box.
Sort	Este método permite ordenar los elementos de un control de tipo: List box, Drop edit, Drop list
Reset	Este método permite borrar todos los elementos de un control de tipo : List box, Drop edit, Drop list
TreeSort	Este método permite ordenar los elementos de un control de tipo List box con estructura en árbol.

- **Métodos para manejar los controles de tipo Box Group y Button Group:**

Método	Funcionalidad
CheckItem	Este método permite modificar el estado (activado o desactivado) de un elemento de un control de tipo Box Group o Button Group
IsItemChecked	Este método permite consultar el estado (activado o desactivado) de un elemento de un control de tipo Box Group o Button Group

- **Métodos para manejar los controles de tipo Box Group:**

Método	Funcionalidad
GetItemBackGround	Permite consultar el color de fondo de un elemento del control
SetItemBackGround	Permite modificar el color de fondo de un elemento del control.

- **Métodos para manejar los controles de tipo List Box, Box Group y Button Group:**

Método	Funcionalidad
GetListIcon	Este método permite consultar el índice del icono que tiene asignado un elemento de un control List Box, Box Group, Button Group
SetListIcon	Este método permite asignar un icono a un elemento de un control List Box, Box Group, Button Group

- **Métodos para manejar los controles de tipo Slider y Percentage box:**

Método	Funcionalidad
Max	Este método permite obtener el valor máximo del rango que abarca el control
Min	Este método permite obtener el valor mínimo del rango que abarca el control
Pos	Este método permite obtener la posición de la aguja del Slider o de la marca de un control de tipo Percentage box
SetPos	Este método permite modificar la posición de la aguja del Slider o de la marca de un control de tipo Percentage box
SetRange	Este método permite modificar el valor máximo y mínimo del rango que abarca el control

- **Métodos para el manejo de controles de impresión de tipo Group:**

Método	Funcionalidad
Truncate	Este método modifica el número de filas que tiene un control Group de una página de impresión

.

- **Métodos para el manejo de controles de impresión:**

Método	Funcionalidad
LoadVars	Este método actualiza la variable de la página de impresión asociada al control con el valor que tenga dicho control. Si se aplica a un control contenedor (Group, Box, Bitmap) se actualizarán todas las variables asociadas a sus controles hijos
PrintVars	Este método actualiza el control de la página de impresión con el valor de su variable asociada. Si se aplica a un control contenedor (Group, Box, Bitmap) se actualizarán todos sus controles hijos

- **Métodos para el manejo de controles de impresión de tipo Variable:**

Método	Funcionalidad
WrappedLine	Este método retorna la porción del texto que corresponde a la línea indicada como parámetro. El resultado es dependiente del dispositivo sobre el que se aplica , por tanto se debe pasar como parámetro el PrnDocument, al que va destinado el texto, para garantizar el funcionamiento con su dispositivo de contexto. Si el control no es de impresora, o se utiliza en modo preview, no es necesario indicar este parámetro, porque se asume la pantalla como dispositivo por defecto
WrappedLinesNeeded	Este método calcula cuantas líneas serían necesarias para encajar el texto en el control. El resultado es dependiente del dispositivo sobre el que se aplica , por tanto se debe pasar como parámetro el PrnDocument, al que va destinado el texto, para garantizar el funcionamiento con su dispositivo de contexto. Si el control no es de impresora, o se utiliza en modo preview, no es necesario indicar este parámetro, porque se asume la pantalla como dispositivo por defecto

- **Métodos para el manejo de controles de usuario:**

Método	Funcionalidad
SetUserProc	Este método permite indicar la función de una DLL que manejará un control de usuario

- **Métodos para el manejo de controles de tipo Tab :**

Método	Funcionalidad
DisablePage	Este método permite modificar el estado (habilitado o inhabilitado) de una página de un control de tipo Tab de una Screen
IsPageDisabled	Este método permite consultar el estado (habilitado o inhabilitado) de una página de un control de tipo Tab de una screen

29. Clase Container

Esta clase deriva de la clase Object. Es una clase virtual. De ella derivan las clases que son contenedores de objetos.

De la clase Container derivan:

Clase Array — Esta clase es abstracta. Un Array es un conjunto de objetos de la misma clase a los cuales se accede por un índice. El índice de un Array es una expresión que evalúa a un valor entero, indicando a que elemento del Array se va a acceder. Es importante recordar que siempre se debe utilizar un índice de Array con origen en uno (1). Esto significa que el primer elemento del Array es el elemento 1 y no el elemento 0. Cuando recorra un Array mediante un bucle, asegúrese de que inicia el elemento de Array a uno (1). Los Arrays del lenguaje COOL de Cosmos son unidimensionales, pero se puede extender a más dimensiones definiendo un Array de Arrays. Se hace referencia a cada elemento de un Array mediante el nombre del Array, seguido de su índice encerrado entre corchetes ([.]).

Clase ArgList Esta clase abstracta permite almacenar listas variables de argumentos. Una lista de argumentos es un conjunto de objetos del mismo tipo, es decir, todos los objetos de la lista tiene la misma clase base. A los elementos de un lista de argumentos se accede por su índice. Esta clase se utiliza para definir métodos que tiene un número variable de parámetros.

Métodos de la clase Container

Método	Funcionalidad
Size	Este método devuelve el número de elementos que tiene un objeto de la clase Container (Array, ArgList).

Operadores de la clase Container

Operador	Funcionalidad
[ ]	Este operador permite acceder a un elemento de un objeto de la clase Container (Array, ArgList).

30. Clase Array

Esta clase deriva de la clase Container. Esta clase es abstracta. Un Array es un conjunto de objetos de la misma clase a los cuales se accede por un índice. El índice de un Array es una expresión que evalúa a un valor entero, indicando a que elemento del Array se va a acceder. Es importante recordar que siempre se debe utilizar un índice de Array con origen en uno (1). Esto significa que el primer elemento del Array es el elemento 1 y no el elemento 0. Cuando recorra un Array mediante un bucle, asegúrese de que inicia el elemento de Array a uno (1).

Los Arrays del lenguaje COOL de Cosmos son unidimensionales, pero se puede extender a más dimensiones definiendo un Array conjunto de Arrays.

Se hace referencia a cada elemento de un Array mediante el nombre del Array, seguido de su índice encerrado entre corchetes ([.]).

31. Clase ArgList

Esta clase deriva de la clase Container. Esta clase abstracta permite almacenar listas variables de argumentos.

Una lista de argumentos es un conjunto de objetos del mismo tipo, es decir, todos los objetos de la lista tiene la misma clase base. A los elementos de un lista de argumentos se accede por su índice. Esta clase se utiliza para definir métodos que tiene un número variable de parámetros.

Una lista variable de argumentos siempre se pasa por referencia. Cuando en un método se define como parámetro una lista variable de argumentos, este debe indicarse en último lugar.

Anexo D: Variables de Entorno

Contenidos

1. Introducción

Estas variables sólo se pueden definir en el fichero de configuración COSMOS.INI mediante el Editor de Configuración.

En este anexo se explica la terminología y la sintaxis para la administración de las variables de entorno propias de Cosmos, así como el objeto de cada una de ellas para el manejo de la aplicación. Las variables de entorno específicas de cada gestor de base de datos que puede ser utilizado a través del módulo MultiWay se explican en el anexo de Conexiones si bien su configuración deberá llevarse a cabo también a través del Editor de Configuración.

- **De Conexión**.

DBCHARSET	DBNAME	DBPASSWD	DBUSER
DBHOST	DBPATH	DBSERVICE	STRANSDIR
XDBTEMP			

- **De Formato**

DBDATE	DBMONEY	DBTIME

- **De EasyReport**

CRWPATH	GRWPATH	TRWPATH	TRWPRIV

- **CTSQL**

ISAMBUFS	MBISFILES	OUTOPT	OUTOPT2
OUTOPT3	SORTMEM		

- **Gateways**

DBEMBED	DBLONGCHAR	DBSERVER	DBSQL
DBPATH	DBSYN	II_SYSTEM	INFORMIXDIR
MBCOMMIT	MBLCKTIMEOUT	DB2INSTANCE	DB2_UID
INGRES_UID	INGRES_DBA	MBLOADCOMMIT	ORACLE_DECIMAL
ORACLE_HOME	ORACLE_PROC	ORACLE_SID	ORACLE_UID

- **Diversas**

DBDELIM	DBEDIT	DBQUOTED	DBTEMP
LANG			

2. Variables de entorno de Conexión

Las variables de entorno de conexión definen como establecer una conexión con un servidor y permiten personalizarlo. Se pueden definir en un entorno global del fichero de configuración COSMOS.INI con el editor de configuración, aunque lo normal será definirlas en un entorno de conexión.

Las variables de entorno de conexión son las que se indican a continuación:

DBCHARSET

Esta variable de entorno permite especificar el juego de caracteres que se empleará tanto en Cosmos como en las aplicaciones con él desarrolladas.

Sus posibles valores son OEM y ANSI, para trabajar respectivamente con el conjunto de caracteres OEM (conjunto extendido de caracteres de IBM para aplicaciones en modo carácter bajo MS-DOS) o ANSI (caracteres propios del entorno Windows). Su valor por defecto es ANSI.

DBHOST

Esta variable es utilizada en instalaciones cliente-servidor, y su finalidad es indicar el nombre de la máquina cliente donde se encuentra el gestor de la base de datos.

```
DBHOST=nombre_servidor
```

Dado que en la instalación podrá existir más de un servidor, con el simple mantenimiento de esta variable de entorno será posible comunicar con uno u otro.

Esta variable no tiene valor por defecto.

DBNAME

En esta variable de entorno se indica el nombre de la base de datos que se desea utilizar por defecto en caso de no indicar una determinada al establecer una conexión.

Ejemplo:

```
DBNAME=alumnos
```

Este ejemplo indica que se va a utilizar por defecto la base de datos alumnos.

DBPATH

Esta variable contiene la lista de directorios en los cuales podrán existir o generarse bases de datos que utilicen dicha variable.

El separador entre directorios es el punto y coma (;). El examen de esta variable se realiza desde el primer path o directorio hasta el último. Su valor por defecto es siempre el directorio en curso.

Ejemplo:

```
DBPATH=c:\datos\alumnos;c:\datos\asignaturas
```

En este caso, podrán existir o crearse bases de datos en los directorios c:\datos\alumnos y c:\datos\asignaturas.

La variable DBPATH deberá contener todos los directorios donde se puedan encontrar las bases de datos antes de establecer la conexión.

Por facilidad de administración se recomienda que cada aplicación se encuentre en un directorio distinto.

Esta variable de entorno sólo puede utilizarse con los gestores de bases de datos propios de TransTOOLs (TransTOOL 3.x, MultiBase y Cosmos), así como por el de Informix.SE.

DBPASSWD

Esta variable se emplea únicamente en instalaciones con arquitectura cliente-servidor. Su finalidad es definir la palabra clave (password) para conectarse a una base de datos residente en la máquina servidor.

Si el usuario del servidor al que se va a conectar la aplicación tiene definida una password, la variable de entorno DBPASSWD deberá tener igualmente definida dicha palabra clave antes de establecer la comunicación con el SQL remoto en la máquina servidor.

Ejemplo:

```
...
{ getpasswd() es una función que pide la "password" por pantalla }
...
let pass = getpasswd()
call putenv ("DBPASSWD", pass clipped)
...
database base_datos
...
```

DBSERVICE

Esta variable indica el nombre del gateway de Cosmos como gestor de la base de datos en instalaciones cliente-servidor.

La configuración de esta variable sólo procede en las máquinas cliente. Sus posibles valores son los siguientes:

Valor	Gestor de BD
Ctsql	CTSQL (Cosmos)
Gwinformix	Informix
Gworacle	Oracle
Gwingres	Ingres

Ejemplo:

```
DBSERVICE=gwinformix
```

Este ejemplo indica que el gestor de base de datos es Informix.

El valor por defecto de esta variable es ctsql.

DBUSER

Esta variable se mantiene únicamente en versiones para red local o en instalaciones con arquitectura cliente-servidor con o sin gateways. En la versión para red local su objetivo es identificar a cada cliente de la red con un nombre de usuario, simulando un entorno multiusuario. Por su parte, en instalaciones cliente-servidor sólo deberá configurarse en las máquinas cliente indicando el nombre del usuario del que se tomarán los permisos para acceder a la base de datos.

Si no se define esta variable tomará por defecto el valor system, mientras que en instalaciones cliente-servidor su valor dependerá de los nombres de usuarios a los que el administrador de la base de datos haya concedido acceso.

A la hora de acceder a la base de datos, el error más común producido por una mala configuración de esta variable es el siguiente:

Permiso CONNECT requerido

Por ejemplo:

```
DBUSER=cris
```

El valor que se asigne a esta variable podrá tener una longitud máxima de ocho caracteres.

STRANSDIR

Esta variable de entorno sólo tiene que configurarse en instalaciones de red local. Su objetivo es la gestión de bloqueos entre usuarios.

La variable STRANSDIR indica el path o directorio del servidor de la base de datos que utiliza un cliente de la red, es decir, el directorio en la máquina servidor donde se instaló Cosmos.

La configuración de esta variable sólo tiene sentido cuando Cosmos reside en la máquina servidor. Todos los nodos de la red que accedan a una misma base de datos tendrán que definir el mismo directorio.

Por ejemplo:

```
STRANSDIR=c:\cosmos
```

Si no se define esta variable se asume el valor relativo al path donde se haya instalado Cosmos.

XDBTEMP

Esta variable determina el path o directorio de la máquina servidor para la creación de ficheros temporales del CTSQL en instalaciones de red local o en cliente-servidor con o sin gateways. Los ficheros temporales que se generarán en el correspondiente directorio serán los relativos al gestor de la base de datos.

Esta variable tiene que configurarse en la máquina cliente de la instalación. Su valor por defecto será siempre el directorio tmp del servidor.

El valor asignado a esta variable deberá ser un directorio existente en el servidor.

Cada vez que se arranque el servidor es conveniente limpiar el contenido del directorio temporal al que hace referencia esta variable.

3. Variables de entorno de Formato

Estas variables permiten definir el formato de presentación de los datos de tipo CHAR,DATE,TIME. Las variables de formato son las que se indican a continuación:

DBDATE

Indica el formato a utilizar para la representación de la fecha (tipos de datos DATE). Sus objetivos son los siguientes:

- Especificar el orden del mes, día y año.
- Indicar si el año se mostrará con dos o cuatro dígitos: Y2 o Y4, respectivamente.
- Determinar el carácter de separación entre mes, día y año.

Valor	Significado
D	Representa el día
M	Representa el mes
Y	Representa el año
2 ó 4	Indica el número de dígitos para el año
Guión(-) Punto(.) Barra(/)	Carácter de separación entre los distintos elementos de una fecha. Cualquier otro carácter que se indique será sustituido siempre por la barra.

Ejemplo:

```
DBDATE=DMY4/
```

Si no se define esta variable de entorno se asumirá como valor por defecto para las fechas el establecido a través del Panel de Control de Windows.

Si se insertan o modifican los valores sobre variables de tipo DATE, esta variable tendrá que indicar obligatoriamente formatos de fecha completos. Esto significa que con una variable de tipo DATE no podremos controlar la entrada de años sin día ni mes. Así, el siguiente supuesto produciría un resultado erróneo:

```
DBDATE=Y4
export DBDATE
```

DBMONEY

Esta variable indica el formato que se utilizará en los valores de tipos de datos MONEY, tanto para especificar los separadores de decimales como el símbolo o literal de la unidad monetaria correspondiente. Los valores posibles son:

Valor	Significado
Front	Símbolo o literal opcional que precede al valor MONEY
Coma (,) Punto (.)	Símbolos opcionales que separan la parte entera de la decimal
Back	Símbolo o literal opcional que sigue al valor MONEY

El formato es :

[" [front] , | . [back] "]

Los símbolos o literales front y back podrán tener una longitud máxima de 7 caracteres y contener cualquier carácter, a excepción de la coma y el punto.

Ejemplo:

```
DBMONEY=$.
```

En este caso, 1.430,12 se representará como $1,430.12.

Si no se define esta variable de entorno se asumirá como valor por defecto para los formatos de moneda el establecido a través del Panel de Control de Windows.

DBTIME

Esta variable indica el formato que se utilizará para la representación de la hora (tipos de datos TIME). Los valores posibles son:

Valor	Significado
H	Se muestra únicamente la hora
M	Se muestra la hora y los minutos
S	Se muestra la hora, los minutos y los segundos
MT	Aparece la hora respecto al meridiano (AM ó PM)

Ejemplo:

```
DBTIME=MMT
```

En este ejemplo las 16:35 se representaría como 04:35 PM.

Si no se define esta variable de entorno se asumirá como valor por defecto para los formatos horarios el establecido a través del Panel de Control de Windows.

4. Variables de entorno de EasyReport

Estas variables de entorno son utilizadas por EasyReport para la generación de informes.

La idea general respecto a los directorios especificados en las variables de entorno CRWPATH, GRWPATH y TRWPATH es la siguiente:

1. Todos los esquemas conceptuales de datos (ECD's) de una aplicación, al ser objetos que deben ser mantenidos por el administrador de la misma o por un programador, pueden estar en un mismo directorio (el definido en TRWPATH).
2. Cada usuario tendrá su propio catálogo y sus propios informes, que estarán contenidos en el directorio indicado en la variable de entorno CRWPATH o, en su defecto, en el indicado en TRWPATH o, en el caso de que no estuviese definida ninguna de estas variables, en el directorio en curso.
3. Dado que pueden existir informes creados por el administrador de la aplicación que sean globales para todos los usuarios, el administrador podrá editar los catálogos de los usuarios e incluirles los nombres y las descripciones de dichos informes. Para ello, en lugar de copiar los informes en los directorios definidos en la variable CRWPATH de cada usuario, los podrá almacenar en el directorio indicado en GRWPATH, que es común a todos los usuarios. Si un usuario edita un informe contenido en GRWPATH y lo guarda con el mismo nombre, no estará modificando el informe global contenido en GRWPATH, sino que estará creando un informe nuevo en su CRWPATH. Un usuario sólo podrá borrar los informes contenidos en CRWPATH.

Como ya se ha mencionado anteriormente las variables de entorno de EasyReport, son las que se indican a continuación:

CRWPATH

En esta variable de entorno se deberá definir el nombre del directorio en el que se almacenarán los informes generados por un usuario mediante EasyReport (ficheros con extensión .orw), así como el catálogo con los nombres y las descripciones de dichos informes (ficheros con extensión .crw).

En caso de no definir esta variable se asumirá el directorio indicado en TRWPATH, y si ésta tampoco estuviese definida se empleará el directorio en curso.

Al ejecutar EasyReport por vez primera se genera en el directorio indicado un fichero para catalogar los informes, incluyendo el nombre de cada informe y una breve descripción. Dicho fichero se denomina *ecd_nombre.crw*.

GRWPATH

Esta variable es utilizada por EasyReport, y su objeto es definir el nombre del directorio en el que se almacenarán los ficheros de catálogo.

El fichero de catálogo (fichero con extensión .crw) es un fichero secuencial ASCII que es mantenido automáticamente por EasyReport. No obstante, dicho fichero podrá ser editado por el programador para incluir otros informes. Si en su ejecución EasyReport no encuentra un informe (fichero con extensión .orw) en el directorio definido en la variable CRWPATH, o en su defecto en el directorio en curso, lo buscará en el indicado en GRWPATH.

Ejemplo:

```
GRWPATH=c:\informes
```

TRWPATH

Esta variable permite definir el directorio en el que se almacenarán los esquemas conceptuales de datos (ECD's) de una aplicación (ficheros con extensión .trw) generados por la utilidad EasyReport.

Si no se especifica esta variable se asumirá por defecto el directorio en curso.

Ejemplo:

```
TRWPATH=c:\ecd
```

TRWPRIV

Esta variable de entorno es utilizada por el EasyReport para definir un privilegio al usuario sobre las tablas y columnas de una base de datos.

Para definir un privilegio a un usuario se podrá emplear también cualquiera de los siguientes procedimientos:

- Incluyendo el parámetro -i al ejecutar el comando ceasyrep para arrancar EasyReport.
- Utilizando la cláusula PRIV en la sección .CONFIGURATION del ECD.

El usuario sólo tendrá acceso a aquellas tablas y columnas que tengan un privilegio igual o inferior al suyo.

El valor máximo de un privilegio, ya sea de usuario, de tabla o de columna es 32.767.

Respecto a los privilegios de usuario conviene tener en cuenta las siguientes puntualizaciones:

- En caso de indicar el privilegio con el parámetro -i del comando ceasyrep, éste será el privilegio del usuario, independientemente del valor de la variable TRWPATH o del privilegio indicado en la sección .CONFIGURATION del ECD.
- Si no se indica el privilegio con el parámetro -i la variable de entorno TRWPATH tendrá prioridad sobre el privilegio indicado en la sección .CONFIGURATION.
- En caso de tener varias cláusulas PRIV en la sección .CONFIGURATION, o varias secciones .CONFIGURATION con diversas cláusulas PRIV, el privilegio del usuario será el de la primera de dichas cláusulas encontrada en el ECD.
- El valor por defecto del privilegio de un usuario, al igual que el de una tabla o columna, es cero.

El objeto de este mecanismo de privilegios es poder ocultar tablas y/o columnas a ciertos usuarios o, incluso, a todos ellos.

5. Variables de entorno de CTSQL

Estas variables definen como establecer una conexión con el servidor CTSQL (SQL de Cosmos) y permiten personalizarlo. Se pueden definir en un entorno global del fichero de configuración COSMOS.INI con el editor de configuración, aunque lo normal será definirlas en un entorno de conexión.

Estas variables son las que se indican a continuación:

ISAMBUFS

Esta variable indica el número de buffers de 1 Kbyte utilizados por el X/OPEN-ISAM de Cosmos. Se puede sintonizar esta variable en el caso de tener problemas de memoria con programas que realicen búsquedas en tablas de gran tamaño, con índices compuestos grandes o enlaces complejos.

Su valor por defecto es 16, siendo 4 su valor mínimo.

Ejemplo:

```
ISAMBUFS=16
```

MBISFILES

Esta variable permite modificar el número máximo de ficheros abiertos por ISAM, siendo 20 su valor por defecto.

El formato de definición de esta variable es:

MBISFILES=n[,m]

Valor	Significado
n	Indica el número máximo de tablas de la base de datos que pueden estar abiertas simultáneamente. Al alcanzar este número el SQL buscará una tabla que pueda ser cerrada, produciéndose un error en caso de no encontrar ninguna.
m	Si se define este parámetro el proceso de cierre comenzará a partir de m (el SQL optimizará cerrando aquellas tablas que no se estén utilizando). El valor por defecto de m es n en una base de datos no transaccional y 0 en bases de datos transaccionales.

El rango de valores válidos para m y n es de 16-100, siendo m siempre menor o igual que n.

Ejemplo:

```
MBISFILES=30
```

Con independencia del valor asignado a esta variable, el límite máximo de ficheros abiertos nunca podrá exceder de 100.

OUTOPT

Esta variable obtiene información sobre el optimizador de SQL. El valor que adquiere es el nombre de un fichero válido.

Por ejemplo:

```
OUTOPT=c:\fichero
```

Siendo fichero el path absoluto o relativo donde se generará el fichero con la información devuelta por el optimizador. En caso de no indicar ningún path se asumirá el directorio en curso.

Cuando el gestor de la base de datos (SQL) realice cualquier operación, ésta se reflejará en el fichero especificado. Para comprobar el resultado habrá que listar dicho fichero. La información que se graba en este fichero ASCII es la siguiente:

Sentence:
Frase SQL
Optimization: num **Tables** número de tablas implicadas
Table [tabname] num Ranges para la tabla "tabname"
se han definido "num" rangos.

Range[n]: key [**start/length]** ((s1/l1)(s2/l2)…)

número de rango _____/

(s1/l1) Descripción de la clave usada para este rango:
() indica acceso por filas(rowid).

Con el fin de obtener información más detallada sobre el proceso de optimización de consultas (queries) de CTSQL se pueden especificar hasta tres niveles del detalle que se desea obtener en el fichero de salida devuelto por el optimizador. El nivel deseado se indica definiendo las siguientes variables de entorno:

Variable	Significado
OUTOPT	Nivel 1 de detalle: devuelve en el fichero indicado la información sobre el orden de selección de las tablas de la cláusula FROM y, en su caso, el índice utilizado para acceder a cada una de ellas.
OUTOPT2	Nivel 2 de detalle: además de la información del nivel anterior, devuelve información sobre los criterios empleados para obtener el orden de selección de tablas.
OUTOPT3	Nivel 3 de detalle: devuelve en el fichero indicado, además de la información anterior, información sobre los criterios empleados para seleccionar cada índice de una tabla.

El orden en que se listan las tablas implicadas en la sentencia SQL (en el caso de usar más de una tabla) es el que el optimizador selecciona para acceder a ellas.

El número de rangos indica el número de porciones de una tabla que son seleccionadas en la cláusula WHERE.

Para cada rango se especificará, en su caso, la clave por la que se va a acceder a la tabla.

SORTMEM

Esta variable define en bytes el espacio de memoria destinado a ordenaciones por CTSQL. Su valor por defecto es 32.000 bytes.

6. Variables de entorno diversas

Estas variables de entorno permiten definir otras características. Estas variables son las que se indican a continuación:

DBDELIM

Indica el carácter que servirá como separador de campos por las instrucciones LOAD y UNLOAD (importación y exportación de ficheros ASCII.). Su valor por defecto es la barra vertical (carácter ASCII 124: |).

Ejemplo:

```
DBDELIM=,
```

En este caso el separador de campos será la coma (,).

A continuación, en el siguiente ejemplo de fichero ASCII se incluye el carácter \ precediendo a algunas comas. Este carácter indica que la coma que le sigue no es un separador de campos, sino parte del valor:

```
1,MOTOROLA,Duque de Medinacelli\, 13,
2,RTC S.A.,Profesor Waksman\, 49,
3,GUYON,Palencia\, 37,
4,P.P.B.,Narvaez\, 31,
```

Como puede observarse, todas las comas que no son separadores de campos van precedidas por el carácter \ .

El carácter \t indica que el separador de campos será un tabulador.

DBEDIT

Esta variable indica el editor textos que utilizará EasyReport para la generación y/o modificación de Esquemas Conceptuales de Datos (comando coscds). Su valor por defecto es CEdit.

Ejemplo:

```
DBEDIT=cedit
```

DBQUOTED

Esta variable de entorno indica si los campos de tipo CHAR en el fichero ASCII utilizado por los métodos LOAD y UNLOAD (carga y descarga de datos) van entrecomillados o no.

Esta variable de entorno sólo puede tomar 2 valores: FALSE, que indica que los campos de tipo CHAR no van entre comillas (valor por defecto), o TRUE, en cuyo caso los campos sí van entre comillas.

DBTEMP

Esta variable de entorno determina el directorio donde se almacenarán los ficheros temporales generados por Cosmos.

Estos ficheros se generan por consultas del SQL, por creación de tablas temporales, etc.

Su valor por defecto es el directorio c:\tmp. Si éste no existiese ni se hubiese definido la variable de entorno DBTEMP, se emplearía en su lugar el subdirectorio temp del directorio donde se hubiese instalado Windows.

Por ejemplo:

```
DBTEMP=c:\temporal
```

El directorio indicado en esta variable deberá haber sido creado antes de empezar a trabajar con Cosmos.

LANG

Esta variable determina el sufijo del directorio del que se tomarán los mensajes, cuadros de diálogo, textos del sistema, etc.

Dependiendo del valor de LANG, Cosmos dialogará en uno u otro idioma.

Ejemplo:

```
LANG=english
```

En este ejemplo, los cuadros de diálogo (*.drw) de Cosmos se buscarán en el directorio de instalación de Cosmos concatenado con drw\english; los ficheros de mensajes en el directorio del proyecto concatenado con msg\english; el fichero de configuración del proyecto en el directorio del proyecto concatenado con etc\english, y así sucesivamente.

7. Variables de entorno de GateWays

Las variables de entorno específicas de cada gestor de base de datos que puede ser utilizado a través del módulo MultiWay se explican en el anexo dedicado a Conexiones a una base de datos , si bien su configuración deberá llevarse a cabo también a través del Editor de Configuración.

Anexo E:
Conexión a una Base de Datos

Contenidos

1. Introducción

El gestor de base de datos de Cosmos (CTSQL) puede funcionar tanto en modo local como en cliente-servidor, disponiendo en ambos casos de capacidad multiservidor. Es decir, una misma instancia del lenguaje de cuarta generación COOL puede mantener abiertas de manera simultánea varias bases de datos a través de una única instancia del CTSQL.

Cosmos dispone de dos mecanismos diferentes de conectividad. El primero de ellos, a través del estándar ODBC de Microsoft, asegura un intercambio de datos con otras aplicaciones en los entornos gráficos más actuales. El segundo, MultiWay, permite la conexión en cliente-servidor tanto con bases de datos propias de TransTOOLs (TransTOOL 3.x, MultiBase y Cosmos) como con las bases de datos con interfaz SQL más extendidas del mercado. Este tipo de conexión a través de MultiWay presenta la ventaja de no necesitar modificación alguna en los programas fuentes.

MultiWay es una tecnología desarrollada por TransTOOLs que dota a Cosmos de unas extraordinarias prestaciones en instalaciones con arquitectura cliente-servidor.
MultiWay es una capa de software que permite al lenguaje de cuarta generación de Cosmos (COOL), y por tanto a cualquier aplicación desarrollada con él, el acceso transparente a diferentes gestores de bases de datos. Dicho acceso puede hacerse en modo local (COOL y servidor de base de datos en una misma máquina) o en modo cliente-servidor (COOL y servidor de base de datos en máquinas distintas).

Para ello, MultiWay se apoya en diferentes estándares del mercado, el SQL como lenguaje de interfaz con bases de datos relacionales, el TCP/IP como protocolo de comunicaciones y la librería WINSOCK.DLL como estándar para TCP/IP en el caso del Windows (máquina cliente).

La arquitectura del sistema es sencilla y muy ligera, no exigiendo, entre otras cosas, el soporte de red proporcionado por el correspondiente fabricante del gestor de base de datos.
Esta capa de software, que se sitúa al lado del servidor de la base de datos, hace a Cosmos independiente de otras capas de software de red de otros fabricantes de bases de datos, permitiendo la conexión remota y el enlace lógico entre el lenguaje de cuarta generación COOL y el SQL de distintos fabricantes.

MultiWay se encarga de proporcionar una visión homogénea de cualquier SQL que se conecte al lenguaje de cuarta generación de Cosmos, ofreciendo un único set de tipos de datos entre los diferentes sets de los distintos gestores de base de datos y una única gramática, al tiempo que se encarga de que cualquier conversión de tipo sea realizada de forma ortogonal.
Por último, MultiWay se encarga de la traducción de sentencias de SQL entre los distintos gestores de bases de datos soportados.

Los gestores de bases de datos soportados por MultiWay son los siguientes:

- CTSQL de TransTOOLs.
- Oracle (versiones 6 y 7).
- Informix-OnLine e Informix-SE (versiones 4 y 5 y 7).
- Ingres (versión 6).
- DB2/6000 de IBM (versión 2.1).

En este anexo se tratará las particularidades y diferencias existentes en Cosmos cuando se utilice cualquiera de los gestores de bases de datos anteriormente citados.

2. Configuración en Red Local

La configuración de Cosmos en red local precisa la definición de las siguientes variables de entorno:

Variable de entorno	Funcionalidad
DBPATH	Lista de directorios del servidor de red donde existen o se pueden crear bases de datos. Si no se define esta variable se asumirá por defecto el directorio de trabajo
DBUSER	Nombre de usuario de la base de datos que será utilizado para otorgar o revocar privilegios de acceso a la misma
STRANSDIR	Directorio del servidor de la red donde está instalado Cosmos

Defina con el Editor de Configuración (cosconf.exe) las variables de entorno indicadas anteriormente en un entorno de conexión del fichero de configuración COSMOS.INI.

Ejemplos:

- Ejemplo del fichero AUTOEXEC.BAT en instalaciones de red local en Windows: para instalaciones en Windows 3.1x

```
@ECHO OFF
set COMSPEC=c:\dos\command.com
VERIFY OFF
SHARE
```

- Ejemplo del fichero de configuración COSMOS.INI en las máquinas clientes:

```
[connections]
local=local
[environment local]
DBTEMP=h:\tmp
DBPATH=h:\apli1\almacen
DBUSER=pedro
STRANSDIR=h:\cosmos
```

Estos ficheros indican lo siguiente:

- El nombre de usuario para acceder a la base de datos se denomina pedro.
- Cosmos se encuentra instalado en el servidor en el directorio cosmos de la partición h:.
- Para que funcionen los bloqueos entre distintos usuarios de la red debe estar activada la utilidad SHARE en el fichero AUTOEXEC.BAT (sólo en Windows 3.x).

3. Configuración en Cliente-Servidor. MultiWay

El lenguaje de cuarta generación de Cosmos, COOL, dispone de toda la gramática del SQL, de forma que cuando encuentra una sentencia (estática o dinámica) de SQL la envía al gestor de la base de datos, el cual ejecutará la sentencia y devolverá los datos (el resultado) a COOL. En el caso de tener instalado MultiWay en cliente-servidor, COOL enviará la sentencia de SQL a través de la red al servidor que contenga la base de datos indicada. Dicha sentencia la recibirá MultiWay, quien localmente se pondrá en comunicación con el correspondiente gestor de base de datos, le pasará la sentencia SQL y, una vez ejecutada por el gestor, recibirá el resultado y lo enviará por la red al COOL que realizó la petición. Un esquema del funcionamiento del módulo MultiWay es el que se muestra en la Figura E-1.

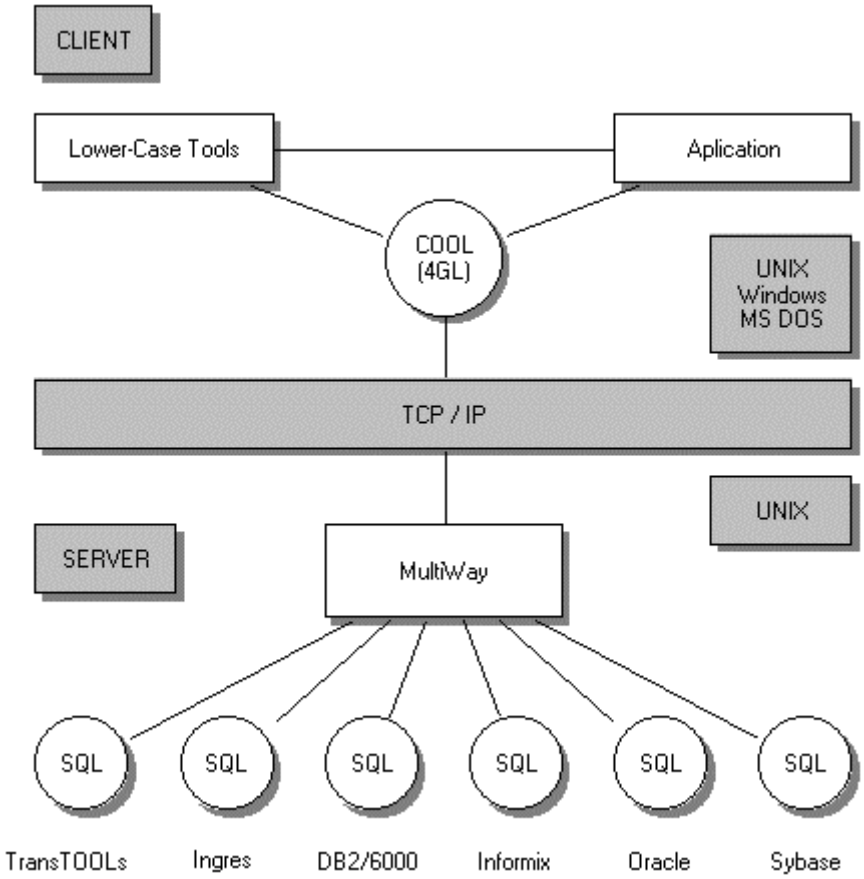


Figura E.1. Esquema de funcionamiento del módulo MultiWay.

MultiWay está compuesto por una serie de gateways, uno para cada gestor de base de datos. Estos gateways siempre ejecutan en máquinas UNIX y deberán estar dados de alta (los que se vayan a utilizar) como servicios de red de la máquina correspondiente (ver instalación gateways).

Básicamente, las tareas que realiza MultiWay son las siguientes:

Traducción

Las gramáticas de los distintos SQL del mercado, si bien todas ellas conforman los estándares ANSI e ISO, suelen tener peculiaridades y extensiones que las hacen diferir unas de otras, en unos casos ligeramente y en otros de una forma considerable. COOL utiliza la gramática de SQL que le proporciona el CTSQL. Estas sentencias son traducidas por MultiWay antes de ser enviadas al correspondiente gestor de base de datos.

Conversión de tipos: Lo mismo que sucede con las gramáticas de los distintos SQL del mercado ocurre con los tipos de datos que soportan. Por ejemplo, mientras que CTSQL soporta un tipo DATE y un tipo TIME, otros SQL soportan un tipo DATETIME que engloba ambos. MultiWay se encargará de las conversiones de tipos adecuadas en ambas direcciones (COOL-gestor y gestor-COOL). Para ello, MultiWay utiliza un catálogo propio que crea en la misma base de datos que se va a utilizar. Este catálogo será gestionado por MultiWay y está compuesto por una serie de tablas cuyos nombres comienzan por mb (mbtables, mbcolumns, mbindexes, etc.). Estas tablas guardan total simetría con las tablas de catálogo del CTSQL de Cosmos (systables, syscolumns, sysindexes, etc.).

Por lo tanto y como resumen, puede decirse que MultiWay se encarga de que cualquier SQL del mercado sea visto por COOL como si fuese CTSQL.

De esta forma, MultiWay proporciona la absoluta portabilidad de cualquier aplicación desarrollada con Cosmos. El que una aplicación funcione en modo cliente-servidor o local y que funcione sobre un determinado servidor de SQL u otro ubicado en una máquina u otra dependerá únicamente de los valores que tomen determinadas variables de entorno.

Ficheros de Configuración en MultiWay

Los ficheros de configuración en cada uno de los gateways definen las variables de entorno y de configuración específicas de cada uno de los gestores de bases de datos (Oracle, Informix, Ingres o DB2/6000). Asimismo, este fichero puede contener también información relativa al mapping de caracteres a realizar entre Cosmos y el servidor de base de datos, así como las funciones propias del servidor que pueden utilizarse en instrucciones CTSQL desde Cosmos. Por lo tanto, estos ficheros tienen la siguiente estructura:

- **Definición de variables de entorno y configuración:**

Al principio de este fichero, sin indicar ningún tipo de sección ni palabra reservada, se especifican los nombres de las variables de configuración de cada gestor de base de datos. Opcionalmente, también se podrán especificar aquellas variables de entorno propias del gestor.

Ejemplo:

```
ORACLE_HOME=/usr/oracle     #  Oracle está instalado en /usr/oracle
ORACLE_SID=contabilidad     #  La base de datos Oracle se
                            #  denomina "contabilidad"
ORACLE_UID=jose/jose123     #  Usuario y password Oracle.
DBSYN:conta=contabilidad    #  Base de datos "conta" en MultiWay y
                            #  "contabilidad" en Oracle.
```

- **Mapping de caracteres: Sección .TRANSLATIONS:**

Esta sección del fichero de configuración de los gateways permite trabajar con una base de datos que no utilice el juego de caracteres de Cosmos (el GCS#2 de IBM). Para ello se define en el fichero de configuración del gateway correspondiente (Informix, Oracle o Ingres) la sección .TRANSLATIONS, donde se especificarán las traducciones necesarias. En cada línea deben figurar dos parámetros: el carácter del juego de caracteres utilizado en la base de datos y su equivalente en el GCS#2 de IBM. Ambos parámetros pueden especificarse en cualquiera de los siguientes formatos: carácter entre comillas simples ('a'), hexadecimal (0x61) u octal (\141). Cualquier texto escrito después de esta definición en la misma línea será considerado como un comentario.

Ejemplo:

```
.TRANSLATIONS
\341    \240   a acentuada
\351    \202   e acentuada
\355    \241   i acentuada
\363    \242   o acentuada
\372    \243   u acentuada
\361    \244   eñe minúscula
\321    \245   eñe mayúscula
\344    \204   a diéresis
\353    \211   e diéresis
\357    \213   i diéresis
\366    \224   o diéresis
\374    \201   u diéresis
\272    \247   `
\252    \246   &
```

- **Funciones del servidor: Sección .FUNCTIONS:**

Existe también la posibilidad de utilizar funciones propias del servidor no reconocidas por Cosmos en frases SQL. Para ello habrá que definir en el fichero de configuración del gateway correspondiente (Informix, Oracle o Ingres) una sección encabezada con .FUNCTIONS en la que se especifican las funciones del servidor que Cosmos ha de reconocer y el tipo de dato que devuelven. En cada línea debe figurar el nombre de la función y el tipo de dato CTSQL del valor devuelto, así como la longitud de éste en el caso del tipo de dato CHAR. Si el tipo de dato del valor devuelto coincide con el de algún parámetro enviado a dicha función se especificará el número de orden de ese parámetro precedido por el carácter $.

Ejemplo:

```
.FUNCTIONS
decode $3
func   integer
subs   char 20
```

Instrucciones Específicas de MultiWay

Las instrucciones que se enumeran a continuación han de ejecutarse a través del método SqlExec de la clase SqlServer predefinida en Cosmos.

Ejemplo:

```
Sql.SqlExec ("create multibase catalogs")
```

- **CREATE MULTIBASE CATALOGS**

Esta instrucción crea el catálogo de MultiWay para la base de datos en curso. Como ya se ha indicado anteriormente, MultiWay utiliza un catálogo propio, compuesto por una serie de tablas cuyos nombres comienzan por mb (mbtables, mbcolumns, mbindexes, etc.). Estas tablas guardan total simetría con las tablas de catálogo del CTSQL de Cosmos (systables, syscolumns, sysindexes, etc.).

- **DROP MULTIBASE CATALOGS**

Al contrario que la anterior, esta instrucción se encarga de borrar el catálogo de la base de datos en curso. Como ya se ha indicado, estas tablas en los gateways se identifican porque su nombre comienza por mb.

- **BEGIN MAINTENANCE MODE**

A partir de la ejecución de esta directiva, las instrucciones SQL que se ejecuten a continuación no tendrán ningún efecto sobre la base de datos, salvo el que respecta a los catálogos de MultiWay.

Esta directiva es obligatoria para actualizar el catálogo de MultiWay cuando se han ejecutado instrucciones que afectan a la definición de la base de datos, utilizando el servidor directamente y no desde Cosmos.

- **END MAINTENANCE MODE**

Esta instrucción finaliza el modo de mantenimiento abierto mediante la instrucción BEGIN MAINTENANCE MODE.

BEGIN MAINTENANCE MODE y END MAINTENANCE MODE se utilizarán para alimentar las tablas del catálogo de MultiWay (tablas mb) con la información de las tablas creadas anteriormente con el propio servidor de la base de datos. Por ejemplo, si ya existe una base de datos en cualquiera de los gestores de base de datos y se instala el módulo MultiWay, habrá que alimentar las tablas del catálogo para que éste pueda manejar dicha base de datos.

- **EXECSQL "instrucción_sql_servidor"**

Esta instrucción ejecuta en modo transparente la frase SQL especificada en *instrucción_sql_servidor*. Esto significa que dicha instrucción es enviada sin analizar ni traducir al servidor de la base de datos correspondiente. Es muy necesaria cuando se quiere ejecutar desde Cosmos alguna instrucción específica del servidor, y por lo tanto no perteneciente al CTSQL (la *instrucción_sql_servidor* nunca puede ser una sentencia SELECT).

4. Configuración en Cliente-Servidor contra el Gestor CTSQL de TransTOOLs.

- **Configuración de las máquinas clientes**

Variable de entorno	Funcionalidad
DBHOST	Nombre del servidor donde se encuentra la base de datos
DBUSER	Nombre de usuario para la base de datos que será utilizado para otorgar o revocar privilegios de acceso a la misma
XDBTEMP	Directorio en el servidor para la creación de ficheros temporales
DBPATH	Directorio en el servidor para la localización de la base de datos
DBPASSWD	Palabra clave (password) del usuario del servidor indicado en la variable de entorno DBUSER
DBSERVICE	(Opcional). Nombre del gestor de base de datos en el servidor. Su valor por defecto es CTSQL

Defina con el Editor de Configuración (cosconf.exe) las variables de entorno indicadas en un entorno de conexión del fichero de configuración COSMOS.INI.

Ejemplo del fichero de configuración COSMOS.INI en la máquina cliente:

```
[Connections]
ctsql=remote

[Environment ctsql]
DBSERVICE=ctsql
DBHOST=nombre_host
DBUSER=user1
DBPATH=/usr/almacen
XDBTEMP=/tmp
DBPASSWD=clave
```

- Este fichero indica las siguientes características:
- El gestor de la base de datos es CTSQL.
- El servidor donde se encuentran tanto la base de datos como el gestor se llama nombre_host.
- El usuario user1 debe tener permiso de acceso a la base de datos que se encuentre en el directorio indicado por DBPATH.
- La variable de entorno DBPATH contiene un directorio de la máquina servidor denominado /usr/almacen.
- El directorio donde se crearán los ficheros temporales es el /tmp de la máquina servidor.
- La clave del usuario user1 es clave.

- **Configuración del servidor:**

Para trabajar con el CTSQL sólo será necesario configurar los ficheros de comunicaciones inetd.conf y /etc/services. La forma de configurar estos ficheros se ha explicado en el capítulo Procedimiento de Instalación.

Configuración en Cliente-Servidor contra el Gestor de Oracle.

- **Configuración de las máquinas clientes:**

Variable de entorno	Funcionalidad
DBSERVICE	Nombre del gateway para el gestor de la base de datos Oracle. Por ejemplo: gworacle
DBHOST	Nombre de la máquina servidor. Por ejemplo: unix
DBUSER	Nombre de usuario para la base de datos que será utilizando para otorgar o revocar privilegios de acceso a la misma. Por ejemplo: user1
XDBTEMP	Directorio en el servidor UNIX para la creación de ficheros temporales. Por ejemplo: /tmp
DBPASSWD	Palabra clave (password) del usuario UNIX incluido en la variable de entorno DBUSER

Defina con el Editor de Configuración (cosconf.exe) las variables de entorno indicadas en un entorno de conexión del fichero de configuración COSMOS.INI.

Ejemplo del fichero COSMOS.INI en las máquinas clientes:

```
[Connections]
oracle=remote

[Environment oracle]
DBSERVICE=gworacle
DBHOST=nombre_host
DBUSER=user1
XDBTEMP=/tmp
DBPASSWD=clave
```

Este fichero indica las siguientes características:

- El gestor de la base de datos es Oracle.
- El servidor donde se encuentran tanto la base de datos como el gestor se llama nombre_host.
- El usuario user1 debe tener permiso de acceso a la base de datos.
- El directorio donde se crearán los ficheros temporales en el servidor es /tmp.
- La clave del usuario UNIX indicado en la variable DBUSER es clave.

- **Configuración del servidor**

La configuración en el servidor se realiza a través del fichero de configuración de Oracle gworacle.env. En este fichero podrán definirse tanto las variables de configuración como, opcionalmente, las variables de entorno que se indican a continuación, siendo en este caso prioritario el valor del entorno de usuario.

Todas las variables que se definan en el servidor tendrán que configurarse en el fichero gworacle.env. Asimismo, las variables ORACLE_HOME, ORA-CLE_UID y ORACLE_ SID se pueden configurar desde un programa COOL en la máquina cliente mediante el método de la clase Module putenv.

Variable de configuración	Funcionalidad
ORACLE_PROC	Indica al gateway si el gestor de Oracle permite o no la generación de procedimientos SQL. Sus posibles valores son ON y OFF, siendo el primero su valor por defecto
ORACLE_DECIMAL	Indica el separador decimal utilizado por Oracle. Su valor por defecto es el punto decimal
DBLONGCHAR	Esta variable indica el número máximo de caracteres que considerará Cosmos para el tipo de datos LONG de Oracle. El valor por defecto asignado a esta variable es 2.000 caracteres
MBCOMMIT	Indica si ha de hacerse o no un COMMIT después de cada instrucción SQL siempre que no haya cursores FOR UPDATE abiertos. Los únicos valores que puede tomar son ON y OFF (por defecto ON)
MBLOADCOMMIT	Indica que se producirá un COMMIT por cada una de las filas que se inserten con la instrucción de carga de datos LOAD. Sus posibles valores son ON, OFF y n (por defecto OFF). En el caso de asignar un número n a esta variable significará que se hará un COMMIT por cada n filas que se inserten con la instrucción LOAD
DBSYN	Define sinónimos del nombre de la base de datos existente en Oracle. Establece la equivalencia entre el nombre de la base de datos de MultiWay con la del gestor de Oracle
DBSERVER	Indica la versión del servidor a utilizar. Sus posibles valores son OR60 y OR70, siendo el primero el valor por defecto

Variable de entorno	Funcionalidad
ORACLE_UID	Indica el nombre del usuario Oracle junto con su correspondiente clave de acceso (password)
ORACLE_SID	Indica el nombre de la base de datos Oracle Para poder utilizar sinónimos de la base de datos habrá que configurar la variable DBSYN tantas veces como sinónimos existan. Si no se define ninguna de las dos variables, se considerará como valor de ORACLE_SID el especificado en la instrucción correspondiente de conexión a la base de datos del programa Cosmos que vaya a dialogar con Oracle
ORACLE_HOME	Indica el directorio donde se encuentra instalado el servidor de la base de datos Oracle (obligatoria)

Ejemplo:
Contenido del fichero gworacle.env para funcionar en cliente-servidor:

```
ORACLE_HOME=/usr/oracle    # Oracle está instalado en /usr/oracle
ORACLE_SID=contabilidad    # La base de datos Oracle se
                           # denomina "contabilidad"
ORACLE_UID=jose/jose123    # Usuario y password Oracle
DBSERVER=OR70              # Versión 7.0 del gestor Oracle
DBSYN:conta=contabilidad   # Base de datos "conta" en MBCosmos y
                           # "contabilidad" en Oracle
```

Portabilidad Cosmos-Oracle.

Este apartado es una pequeña guía práctica de portabilidad de aplicaciones construidas con Cosmos en entornos con CTSQL a entornos con servidor Oracle versiones 6 ó 7.

❖ *Tipos de datos:*

Las conversiones entre los tipos de datos del CTSQL de Cosmos y los del servidor de base de datos de Oracle se realizan de forma automática según se indica en la siguiente tabla:

CTSQL		Oracle
Versión 6 y size < 256	CHAR(size)	CHAR(size)
Versión 7 y size < 2000	CHAR(size)	VARCHAR2
Versión 6 y size > 255 o Versión 7 y size > 2000	CHAR(size)	LONG
	SMALLINT	SMALLINT
	INTEGER	INTEGER
	DECIMAL(p,s)	DECIMAL(p,s)
	MONEY(p,s)	DECIMAL(p,s)
	SERIAL	INTEGER
	DATE	DATE
	TIME	CHAR(11)
	ROWID	ROWID

Oracle	CTSQL
NUMBER(p)	INTEGER
NUMBER(p)	SMALLINT
NUMBER(p,s)	DECIMAL(p,s)
NUMBER(p,s)	MONEY(p,s)
CHAR(size)	CHAR(size)
VARCHAR2	CHAR(size)
LONG	CHAR(DBLONGCHAR)
DATE (sin TIME)	DATE

DBLONGCHAR es la variable de configuración que limita la longitud del LONG para su manejo en programas Cosmos.

El programador tendrá que tener en cuenta estas tablas si quiere actualizar el catálogo de MultiWay con una base de datos ya existente en Oracle, creando un esquema acorde con las conversiones automáticas que realiza el gateway.

❖ *Funciones*

Restricciones de funcionalidad: Con la versión 6 del gestor de Oracle no se pueden utilizar las funciones contempladas por Cosmos en frases SQL, al no existir equivalencia de las mismas en dicho gestor ni posibilidad de generarlas.

Por el contrario, la versión 7 del gestor de Oracle sí permite la creación de funciones SQL (previa instalación de la utilidad correspondiente de Oracle) que se almacenan en la misma base de datos. En este momento, y siguiendo este procedimiento, MultiWay sólo crea la función typelength de forma automática, siempre y cuando la variable ORACLE_PROC se encuentre activada. El resto podrían ser simuladas sin dificultad siguiendo el mismo mecanismo.

Por lo que respecta a las funciones SQL propias de Oracle, éstas podrán utilzarse si se especifican adecuadamente en la sección .FUNCTIONS del fichero de configuración gworacle.env.

❖ *Instrucciones*

Primera fase de traducción:

Algunas de las diferencias sintácticas existentes en instrucciones del CTSQL respecto a las del servidor de Oracle se resuelven por el gateway de forma transparente al programador en una primera fase de traducción. Por ejemplo:

- Las comillas dobles se traducen por comillas simples.
- La cláusula MATCHES y los metacaracteres *? por LIKE %_.
- La cláusula UNIQUE por DISTINCT.
- La cláusula WHERE CURRENT OF ‘cursor_name’ por WHERE ROWID = ‘rowid’.
- DECLARE CURSOR FOR UPDATE por DECLARE CURSOR FOR UPDATE OF <columnas>.
- En subindicaciones: variable[n1,n2] por substr(varname,n1,n2-n1+1).
- Un literal vacío (“”) por un blanco (“ “).
- La palabra reservada TEXT por MBTX.
- Las expresiones TODAY y NOW por sysdate y sysdate con el formato HH24:MI:SS.

Las fases subsiguientes resuelven, de manera particularizada, más diferencias de este tipo, así como el resto de incompatibilidades existentes entre las instrucciones del CTSQL de Cosmos y las del servidor de Oracle.

• CREATE DATABASE, DROP DATABASE, START DATABASE y DATABASE.

Estas instrucciones suponen la ejecución automática por parte del gateway de los siguientes pasos:

CREATE DATABASE	=>	CONNECT + CREATE MULTIBASE CATALOGS
DROP DATABASE	=>	DISCONNECT + DROP MULTIBASE CATALOGS
START DATABASE	=>	CONNECT
DATABASE	=>	CONNECT

Siendo CREATE MULTIBASE CATALOGS y DROP MULTIBASE CATALOGS instrucciones nuevas de MultiWay (disponibles para el programador), y CONNECT y DISCONNECT instrucciones internas para el gestor de Oracle.

Como puede observarse, la creación como tal de la base de datos ha de realizarse previamente a través del administrador de Oracle.

Cada una de las instrucciones anteriores asigna a la variable de entorno ORACLE_SID el nombre de la base de datos de conexión (considerando los posibles valores de la variable de configuración DBSYN existente en gworacle.env). El usuario/password al que se conecta viene determinado por la variable de entorno ORACLE_UID (definida en el entorno o en gworacle.env).

- **CREATE TABLE.**

Frente a la versión 6 del gestor de Oracle:

- El atributo NOT NULL se mantiene directamente por el SQL de Oracle.
- La claúsula PRIMARY KEY se simula de forma automática por el gateway a través de la creación de un índice único sobre las columnas especificadas y cuyo identificador es el nombre de la tabla precedido por PK.
- El atributo DEFAULT es simulado también por el gateway en las operaciones de INSERT cuyos valores en VALUES se introduzcan a través de constantes o variables hosts (esto se hace extensivo al FORM).

Restricciones de funcionalidad:

- Respecto al atributo DEFAULT, la simulación no funciona cuando los valores se introducen a través de una instrucción SELECT, ya que no puede controlarse su resultado (INSERT… SELECT…).
- El resto de atributos (entre ellos CHECK) y la clave referencial se aceptan gramaticalmente, pero no funcionan a nivel de SQL.
- No obstante, la definición de cualquier atributo queda almacenada en los catálogos de Cosmos, con lo que, a nivel de programa (FORM), muchos de ellos siguen teniendo validez (RIGHT, ZEROFILL, etc.) ya que se utiliza esta información como base.

Frente a la versión 7 del gestor de Oracle:

El SQL mantiene directamente los atributos CHECK y DEFAULT, además de NOT NULL. También las claúsulas PRIMARY KEY y FOREIGN KEY son manejadas automáticamente, con lo que el gateway ya no tiene que realizar ninguna labor de simulación para ellas.
Restricción de funcionalidad: El resto de atributos no funcionan a nivel de SQL.

- **CREATE TEMP TABLE.**

El empleo y manejo de tablas temporales desde programas COOL no varía a pesar de que internamente son tablas normales de usuario (y no temporales) identificadas con el número de proceso y el prefijo MBT. Dado que en Oracle no existe este tipo de tablas, éstas son simuladas automáticamente por el gateway, resultando invisible al programador esta carencia. Y lo mismo sucede con la tabla temporal de la instrucción SELECT… INTO TEMP….

Si a lo largo de la ejecución se ha cambiado sucesivamente de base de datos, puede que al finalizar el proceso se hayan creado tablas temporales en diferentes usuarios Oracle. En estos casos, el gateway las borrará naturalmente para simular esta temporalidad, pero haciendo dos supuestos a la hora de conectarse a los usuarios: Que no se haya cambiado de ORACLE_SID y que la password de estos usuarios coincida con su nombre. En cualquier otro caso quedarían tablas sin borrar.

- **CREATE INDEX.**

Restricción de funcionalidad: El atributo DESC no es efectivo, ya que los índices son siempre ascendentes.

- Las tablas y los índices han de tener identificadores distintos, dado que así lo exige Oracle, y 18 caracteres como máximo (límite de CTSQL).
- Si la base de datos existía en Oracle con anterioridad a su manejo con MultiWay, puede suceder que haya identificadores que superen esa longitud de 18 caracteres. El problema se soluciona fácilmente creando sinónimos en Oracle para todos esos identificadores con la longitud admitida por CTSQL. Estos nombres serán los que se manejen desde programas COOL.
- Si al ejecutar una frase SQL que genera, modifica o borra un objeto de la base de datos se produce un error actualizando el catálogo de Cosmos, éste puede quedar inconsistente, ya que la acción sobre el objeto no se deshace. En este caso es necesario rectificarlo para que cualquier aplicación Cosmos funcione correctamente.

- **ALTER TABLE.**

Dado que esta instrucción es desconocida por el SQL de Oracle, el gateway la simula para que, de forma transparente, el programador pueda seguir usándola.

- **RENAME COLUMN.**

No funciona en esta versión.

- **GRANT CONNECT.**

La sintaxis de esta instrucción en Oracle difiere de la del CTSQL. Para evitar esta incompatibilidad, el gateway añade al final de la misma la claúsula IDENTIFIED BY, y como password sigue el criterio de utilizar el mismo identificador empleado en el nombre de usuario.

- **GRANT SELECT.**

No funciona la opción GRANT SELECT (<columna1>, …, <columnan>) …, sino únicamente GRANT SELECT…, sin especificación de columnas.

- **INSERT.**

La inserción automática para el tipo de dato SERIAL (inexistente en Oracle) se simula en parte gracias a una tabla creada a tal efecto (MBSERIAL), de donde el gateway lee el último valor que asignó de esta forma (por cada tabla con una columna de este tipo, existe una fila con dicho valor).

- **FETCH CURSOR.**

Restricciones de funcionalidad:

- En Cosmos todos los cursores permiten el acceso a las filas siguiente, anterior, primera y última (NEXT, PREVIOUS, FIRST, LAST).
- Aunque Oracle no tiene cursores SCROLL, el gateway hace que esto se siga cumpliendo de forma transparente al programador, pero restringiéndolo a los cursores que no sean FOR UPDATE. Para ello emplea un fichero secuencial (SCRnnnnpid.GW) donde realiza el scroll necesario de los registros seleccionados.

- **LOCK TABLE.**

La instrucción LOCK TABLE será capaz de bloquear una tabla si la variable de configuración MBCOMMIT tiene el valor OFF (en el fichero gworacle.env), ya que este COMMIT automático después de cada instrucción SQL desbloquea las tablas cuando Cosmos funciona frente al SQL de Oracle. Lo normal es encerrar esta instrucción dentro de una transacción controlada por el programador, de manera que cuando se quiera desbloquear de verdad se finalice la transacción con COMMIT.

- **OPEN CURSOR.**

Los identificadores de CURSOR tendrán como máximo 15 caracteres en lugar de 18 (máximo permitido por CTSQL para cualquier identificador de la base de datos frente a cualquier SQL). Esta restricción permite distinguir cursores con igual nombre en diferentes módulos COOL, aspecto éste que sólo sabe manejar el CTSQL.

Si el CURSOR tiene la cláusula FOR UPDATE, el comportamiento de esta instrucción es algo diferente frente al servidor de Oracle:

- Este SQL ejecuta la frase SELECT en el momento de abrir el CURSOR, bloqueando todas las filas obtenidas en la selección hasta el COMMIT de la transacción que la incluye.
- Si el usuario no controla las transacciones desde su programa, es la variable de configuración MBCOMMIT=ON (del fichero gworacle.env) la que puede provocar este COMMIT.
- Tanto en un caso como en otro, es necesario cerrar previamente el CURSOR, ya que el COMMIT sólo es enviado al SQL de Oracle por el gateway, si no existe ningún CURSOR abierto FOR UPDATE.
- Si, al abrir el CURSOR, Oracle encuentra problemas para bloquear todas las filas seleccionadas, el CURSOR permanecerá cerrado. La ejecución del programa COOL sigue su curso siempre y cuando el CURSOR haya sido definido FOR UPDATE… NOWAIT, quedando registrado este incidente en la variable locked, que tomará el valor TRUE. Esto implica que, para que un programa COOL se comporte de manera equivalente cuando funcione frente al servidor de Oracle, es necesario controlar también en el OPEN los problemas de interbloqueo de cursores, y no sólo en el FETCH, como sucede en Cosmos con CTSQL.

- **SELECT… FROM…**

El gateway está simulando el comportamiento de CTSQL en un SELECT {MIN,MAX,AVG,SUM} sin GROUP BY (ya que Oracle devuelve una fila con valor NULL si el número de filas seleccionadas es cero, mientras que CTSQL no devuelve nada), con la salvedad de que el programador no puede distinguir el caso de una selección de filas con valores nulos (aunque esto no es habitual).

Restricciones de funcionalidad:

- La restricción DISTINCT/UNIQUE NO se puede incluir junto con la claúsula FOR UPDATE.
- NO se puede emplear el outer en la cláusula FROM. El gateway no realiza la traducción debido a la falta de equivalencia con el outer definido por columnas de Oracle.
- NO se pueden utilizar alias en las cláusulas ORDER BY y GROUP BY. La solución para aquellos casos en que sea necesario su empleo (referencias a columnas compuestas: expresiones, agregados, etc.) es utilizar en su lugar los números de orden dentro de la lista de campos de la SELECT.

- **ROLLFORWARD DATABASE y UPDATE STATISTICS.**

Al no tener equivalencia en Oracle, estas instrucciones no son enviadas a su SQL por el gateway, con lo que sus efectos son nulos aunque se utilicen en programas COOL, favoreciendo de esta manera la compatibilidad de fuentes frente a distintos servidores de bases de datos.

- **UNLOCK TABLE.**

Esta instrucción es ignorada por el gateway, ya que Oracle no la contempla. Para conseguir el desbloqueo de una tabla ha de realizarse un COMMIT de la transacción dentro de la cual se hizo el bloqueo.

❖ ***Otros***

- **Tablas compartidas**

Para compartir una tabla entre dos bases de datos (dos usuarios Oracle) se sigue un camino similar al realizado frente al gestor CTSQL.

El usuario A, creador de la tabla a compartir, da los permisos que considere oportunos al usuario B sobre la misma y sobre la tabla mbserial si existe algún campo cuyo tipo de dato es SERIAL.

El usuario B crea el sinónimo:

```
CREATE SYNONYM nombre_de_tabla
FOR usuario_A.nombre_de_tabla
```

para poder utilizar el mismo nombre de tabla. Además, crea la tabla en modo mantenimiento para que su catálogo la contenga y, con una instrucción UPDATE modifica el campo dirpath de la fila que haya aparecido en mbtables para esa tabla, cambiando el valor usuarioB por usuarioA (es en A donde realmente se halla la tabla).

Si en algún momento se altera la estructura de la tabla, hay que repetir el procedimiento anterior (ALTER en modo mantenimiento y cambio de dirpath).

- **Catálogo de Cosmos**

Es posible romper la relación biunívoca existente entre bases de datos Cosmos y usuarios Oracle.

Si se quiere hacer corresponder una base de datos Cosmos con varios usuarios Oracle manteniendo un solo catálogo, éstos tendrían que compartir las tablas mb* siguiendo el procedimiento anterior. Uno de los usuarios sería el creador real del catálogo (dador de permisos) y los demás compartirían las tablas (creando sinónimos).

La tabla mbserial no aparece en el catálogo, dada su naturaleza interna, pero sí han de poder manipularla todos, para lo cual, además de los permisos:

```
EXECSQL GRANT ALL PRIVILEGES ON mbserial TO PUBLIC
```

cada uno de los usuarios, excepto el creador del catálogo, necesita el siguiente sinónimo:

```
EXECSQL CREATE SYNONYM mbserial
FOR usuario_creador.mbserial
```

Configuración en Cliente-Servidor contra el Gestor de Informix.

- **Configuración de las máquinas clientes**:

Variable de entorno	Funcionalidad
DBSERVICE	Nombre del gateway para el gestor de la base de datos Informix. Por ejemplo: gwinformix
DBHOST	Nombre de la máquina servidor. Por ejemplo: unix
DBUSER	Nombre de usuario para la base de datos que será utilizado para otorgar o revocar privilegios de acceso a la misma. Por ejemplo: user1
XDBTEMP	Directorio en el servidor UNIX para la creación de ficheros temporales. Por ejemplo: /tmp
DBPATH	Directorio en el servidor UNIX para la localización de la base de datos en el caso de emplear Informix SE. Por ejemplo: /usr/apli1/almacen
DBPASSWD	Palabra clave (password) del usuario UNIX indicado en la variable DBUSER

Defina con el Editor de Configuración (cosconf.exe) las variables de entorno indicadas en un entorno de conexión del fichero de configuración COSMOS.INI:

Ejemplo del fichero COSMOS.INI en las máquinas clientes:

```
[Connections]
informix=remote

[Environment informix]
DBSERVICE=gwinformix
DBHOST=nombre_host
DBUSER=user1
DBPASSWD=clave
DBPATH=/usr/almacen                (sólo en Informix SE)
XDBTEMP=/tmp
```

Este fichero indica las siguientes características:

- El gestor de la base de datos es Informix.
- El servidor donde se encuentran tanto la base de datos como el gestor se llama nombre_host.
- El usuario user1 debe tener permiso de acceso a la base de datos que se encuentre en el directorio indicado en DBPATH.
- La variable de entorno DBPATH contiene un directorio de la máquina servidor denominado /usr/almacen.
- El directorio donde se crearán los ficheros temporales en el servidor es /tmp.
- La clave del usuario UNIX indicado en la variable DBUSER es clave.

- **Configuración del servidor**

La configuración en el servidor se realiza a través del fichero de configuración de Informix gwinformix.env. En este fichero podrán definirse tanto las variables de configuración como, opcionalmente, las variables de entorno que se indican a continuación, siendo en este caso prioritario el valor del entorno de usuario.

Todas las variables que se definan en el servidor tendrán que configurarse en el fichero gwinformix.env. Asimismo, la variable INFORMIXDIR se puede configurar desde un programa COOL en la máquina cliente mediante el método de la clase Module putenv.

Variable de Configuración	Funcionalidad
DBSERVER	Indica la versión del servidor a utilizar. Sus posibles valores son IX40, IX50 e IX70, siendo el primero el valor por defecto
DBEMBED	Indica la versión del embedded utilizado por el gateway de Informix. Sus posibles valores son: EIX40, EIX50 e EIX70. El valor por defecto dependerá del que tenga la variable de entorno DBSERVER.
DBSQL	Indica si el servidor de la base de datos es Informix On-Line o Informix-SE. Sus posibles valores son SE o ninguno, siendo este último el valor por defecto, lo que significa que el gestor es Informix On-Line
DBSYN	Define sinónimos del nombre de la base de datos existente en Informix. Establece la equivalencia entre el nombre de la base de datos de Cosmos con la del gestor de Informix

Variable de entorno	Funcionalidad
INFORMIXDIR	Esta variable indica el path donde se encuentra instalado el servidor de la base de datos de Informix. Su definición es obligatoria

Configuración en Cliente-Servidor contra el Gestor de Ingres

- **Configuración de las máquinas clientes**

Variable de entorno	Funcionalidad
DBSERVICE	Nombre del gateway para el gestor de la base de datos Ingres. Por ejemplo: gwingres
DBHOST	Nombre de la máquina servidor. Por ejemplo: unix
DBUSER	Nombre de usuario para la base de datos que será utilizado para otorgar o revocar privilegios de acceso a la misma. Por ejemplo: user1
XDBTEMP	Directorio en el servidor UNIX para la creación de ficheros temporales. Por ejemplo: /tmp
DBPASSWD	Palabra clave (password) del usuario UNIX indicado en la variable DBUSER

Defina con el Editor de Configuración (cosconf.exe) las variables de entorno indicadas en un entorno de conexión del fichero de configuración COSMOS.INI.

Ejemplo del fichero COSMOS.INI en las máquinas clientes:

```
[Connections]
ingres=remote

[Environment ingres]
DBSERVICE=gwingres
DBHOST=nombre_host
DBUSER=user1
XDBTEMP=/tmp
DBPASSWD=clave
```

Este fichero indica las siguientes características:

- El gestor de la base de datos es Ingres.
- El servidor donde se encuentran tanto la base de datos como el gestor se llama nombre_host.
- El usuario user1 debe tener permiso de acceso a la base de datos.
- El directorio donde se crearán los ficheros temporales en el servidor es /tmp.
- La clave del usuario indicado en la variable de entorno DBUSER es clave.

- **Configuración del servidor**

La configuración en el servidor se realiza a través del fichero de configuración de Ingres gwingres.env. En este fichero podrán definirse tanto las variables de configuración como, opcionalmente, las variables de entorno que se indican a continuación, siendo en este caso prioritario el valor del entorno de usuario.

Todas las variables que se definan en el servidor tendrán que configurarse en el fichero gwingres.env. Asimismo, la variable II_SYSTEM se puede configurar desde un programa COOL en la máquina cliente mediante el método de la clase Module putenv.

Variable de Configuración	Funcionalidad
MBCOMMIT	Indica si ha de hacerse o no un COMMIT después de cada instrucción SQL siempre que no haya cursores FOR UPDATE abiertos. Los únicos valores que puede tomar son ON y OFF (por defecto ON).
MBLOADCOMMIT	Indica que se producirá un COMMIT por cada una de las filas que se inserten con la instrucción de carga de datos LOAD. Sus posibles valores son ON, OFF y n (por defecto OFF). En el caso de asignar un número n a esta variable significará que se hará un COMMIT por cada n filas que se inserten con la instrucción LOAD
MBLCKTIMEOUT	Permite establecer el tiempo de respuesta en segundos a las peticiones de bloqueo al servidor Ingres. Si en ese tiempo el bloqueo no se facilita, el query que lo solicitó finaliza. Con esta variable se simula el comportamiento de los cursores NOWAIT (en Ingres son todos WAIT), si bien de forma limitada. El tiempo va de un segundo (ON o 1) a n. Por defecto es OFF (WAIT)
DBSYN	Define sinónimos del nombre de la base de datos existente en Ingres. Establece la equivalencia entre el nombre de la base de datos de Cosmos con la del gestor de Ingres

Variable de entorno	Funcionalidad
II_SYSTEM	Esta variable indica el path donde se encuentra instalado el servidor de la base de datos de Ingres. Su definición es obligatoria
INGRES_UID	Indica al gateway el nombre del usuario Ingres para la conexión a la base de datos
INGRES_DBA	Indica al gateway el nombre del usuario Ingres que actúa como administrador de la base de datos

Ejemplo:

Contenido del fichero gwingres.env para funcionar en cliente servidor:

```
II_SYSTEM=/usr/ingres     # Ingres está instalado en /usr/ingres
DBSYN:conta=contabilidad # Base de datos "conta" en Cosmos y
                          # "contabilidad" en Ingres
```

Configuración en Cliente-Servidor contra el Gestor de DB2/6000

- **Configuración de las máquinas clientes**

Variable de entorno	Funcionalidad
DBSERVICE	Nombre del gateway para el gestor de base de datos DB2/6000. Por ejemplo: gwdb2.
DBHOST	Nombre de la máquina servidor. Por ejemplo: AIX
DBUSER	Nombre de usuario para la base de datos que será utilizado para otorgar o revocar privilegios de acceso a la misma. Por ejemplo: user1
XDBTEMP	Directorio en el servidor UNIX para la creación de ficheros temporales. Por ejemplo: /tmp
DBPASSWD	Palabra clave (password) del usuario UNIX indicado en la variable DBUSER

Defina con el Editor de Configuración (cosconf.exe) las variables de entorno indicadas en un entorno de conexión del fichero de configuración COSMOS.INI.

Ejemplo del fichero COSMOS.INI en las máquinas clientes:

```
[Connections]
db2=remote

[Environment db2]
DBSERVICE=gwdb2
DBHOST=AIX
DBUSER=user1
XDBTEMP=/tmp
DBPASSWD=clave
```

Este fichero indica las siguientes características:

- El gestor de la base de datos es db2.
- El servidor donde se encuentran tanto la base de datos como el gestor se llama AIX.
- El usuario user1 debe tener permiso de acceso a la base de datos.
- El directorio donde se crearán los ficheros temporales en el servidor es /tmp.
- La clave del usuario indicado en la variable de entorno DBUSER es clave..

- **Configuración del servidor**

La configuración en el servidor se realiza a través del fichero de configuración de db2, gwdb2.env. En este fichero podrán definirse tanto las variables de configuración como, opcionalmente, las variables de entorno que se indican a continuación, siendo en este caso prioritario el valor del entorno de usuario.

Todas las variables que se definan en el servidor tendrán que configurarse en el fichero gwdb2.env. Asimismo, las variables DB2INSTANCE y DB2_UID se pueden configurar desde un programa COOL en la máquina cliente mediante el método Putenv de la clase Module.

Variable de Configuración	Funcionalidad
DBLONGCHAR	Número que indica el máximo de caracteres a considerar en COSMOS para el tipo de dato LONG VARCHAR de DB2
MBCOMMIT	Indica si ha de hacerse o no un COMMIT después de cada instrucción SQL siempre que no haya cursores abiertos. Los únicos valores que puede tomar son ON y OFF (por defecto ON).
MBLOADCOMMIT	Indica si se producirá un COMMIT por cada una de las filas que se inserten con la instrucción de carga de datos LOAD. Sus posibles valores son ON , OFF y n (por defecto OFF). En el caso de asignar un número n a esta variable significará que se hará un COMMIT por cada n filas que se inserten con la instrucción LOAD
DBSYN	Define sinónimos de los nombres de bases de datos existentes en DB2. Establece la equivalencia entre los nombres de bases de datos de Cosmos y las del gestor de DB2/6000. Pueden crearse varias bases de datos Cosmos para una sola de DB2/6000 (una por cada usuario DB2 que se conecte a ella)
DBSERVER	Indica la versión del servidor a utilizar. Su valor es DB21 por defecto y el único válido por el momento
DB2_DECIMAL	Indica el separador decimal que utiliza DB2. Por defecto es el punto decimal

Variable de entorno	Funcionalidad
DB2INSTANCE	Esta variable indica la instancia activa para el gestor de DB2. Por defecto NULL.
DB2_UID	Indica el nombre del usuario DB2 bajo cuya identificación se va a hacer la conexión a las bases de datos DB2 (junto con su correspondiente clave: user/password).

Ejemplo:

Contenido del fichero gwdb2.env para funcionar en cliente servidor:

```
DB2INSTANCE=db2001              # La instancia activa para el gestor
                                # db2 es db2001.
DBSYN:conta=contabilidad        # Base de datos "conta" en Cosmos y
                                # "contabilidad" en db2
DB2_UID=user1/pass1             # El nombre del usuario es user1 y su
                                # password es pass1
```

Casos Prácticos

Suponiendo que se haya desarrollado una aplicación en Cosmos frente al gestor de base de datos CTSQL, los pasos a seguir para que dicha aplicación funcione con ORACLE son los que se exponen:

1. Generar previamente sobre el servidor CTSQL de Cosmos el esquema de la base de datos (fichero con extensión sql de igual nombre al de la base de datos. Por ejemplo: almacen.sql), para poder reproducir después la misma estructura frente al servidor de Oracle.

2. Confeccionar las variables del fichero de configuración del servidor de Oracle (gworacle.env).
3. En instalaciones cliente-servidor la variable DBSERVICE debe tener el valor gworacle. Junto a esta variable hay que configurar también las variables de entorno DBHOST, DBUSER y DBPASSWD.
4. Crear la base de datos a través del administrador de bases de datos de Oracle.
5. Crear un usuario de conexión a la base de datos a través del administrador de la base de datos Oracle.
6. A continuación existen dos posibilidades:
   - Mediante un programa COOL ejecutar, sobre la base de datos de Oracle, el esquema generado anteriormente sobre el fichero almacen.sql.
     Ejemplo:

```
main begin
Sql.SqlExec ("create database almacen");
Sql.SqlFile ("almacen");
End
```

     La primera instrucción crea la base de datos almacen en Cosmos. Esta operación en MultiWay implica la generación del catálogo de la base de datos. Dicho catálogo está compuesto por las tablas mb* que se crearán en la base de datos de Oracle.
     La segunda instrucción crea las tablas, índices, claves primarias, etc. que estuviesen definidas en el fichero almacen.sql. Estas tablas se crean en la base de datos de Oracle. Asimismo, el catálogo de la base de datos de MultiWay se carga con la información que genera dicha creación de elementos SQL.

   - Ejecutar el esquema desde el entorno Oracle con instrucciones específicas de este servidor. Utilizando este procedimiento sería necesario crear después el catálogo de MultiWay y, entrando en modo mantenimiento, el resto del esquema actualizando dicho catálogo.
     Ejemplo:

```
main begin
Sql.SqlExec ("create database almacen");
Sql.SqlExec ("begin maintenance mode");
Sql.SqlFile ("almacen");
Sql.SqlExec ("end maintenance mode");
End
```

     Este ejemplo es similar al anterior, pero con la diferencia de que las tablas ya están creadas en la base de datos de Oracle. Por lo tanto, la única operación será actualizar el catálogo de MultiWay (tablas mb*). Esta actualización se realiza al ejecutar el esquema almacen.sql pero en modo mantenimiento (BEGIN MAINTENANCE MODE y END MAINTENANCE MODE).
     Ésta es la operación que hay que realizar en Cosmos cuando se desea importar desde el CTSQL una base de datos creada en Oracle.

7. Cargar los datos si procede.

Mensajes de Error

Existe un cierto número de mensajes de error devueltos por el servidor que se esté utilizando (Informix, Oracle, DB2/6000 o Ingres) que se traducen a mensajes del servidor CTSQL de Cosmos. El resto aparecerán bajo el texto genérico:

```
ERROR número DEVUELTO POR EL SERVIDOR SQL
```

Siendo número el número de mensaje de error devuelto por Informix, Oracle DB2 o Ingres. Para este último, en algunos casos aparece también junto al número el texto del mensaje de error correspondiente.

Anexo F: Notificaciones

Contenidos

1. Introducción

Como se vió en el capítulo del Lenguaje COOL existen tres tipos de notificaciones. En este anexo se muestran las listas completas de notificaciones de cada tipo.

2. Notificaciones de Control

Los eventos mandados por un Control de la screen y procesados por su Form son los que se indican a continuación:

Evento	Efecto
Click	El control manda al Form la notificación Click cuando se pulsa el botón izquierdo del ratón sobre el
DblClick	El control manda al Form la notificación DblClick cuando se pulsa dos veces el botón izquierdo del ratón sobre el
Enter	Esta notificación la manda un control de la screen del Form cuando obtiene el foco de entrada
Exit	Esta notificación la manda un control de la screen del Form cuando pierde el foco de entrada
RClick	El control manda al Form la notificación RClick cuando se pulsa el botón derecho del ratón sobre el
SelChange	Una lista manda al Form la notificación SelChange cuando se modifica el elemento o elementos seleccionados
SpinUp	Esta notificación la manda un control de tipo Spin cuando se pulsa su botón 
SpinDown	Esta notificación la manda un control de tipo Spin cuando se pulsa su botón 
SpinTop	Esta notificación la manda un control de tipo Spin cuando se pulsa su botón 
SpinBottom	Esta notificación la manda un control de tipo Spin cuando se pulsa su botón 

En la siguiente tabla se muestra una lista de las notificaciones que puede mandar un control a su Form.

Control	Evento	Control	Evento
Bitmap	Click DblClick RClick	Grid	Enter Exit
Box	Click DblClick RClick	List Box	DblClick Enter Exit RClick SelChange
Box Group	Click Enter Exit RClick	Menú	Enter Exit
Button Group	Click Enter Exit RClick	Push Button	Click Enter Exit RClick
Check Box	Click Enter Exit RClick	Radio Button	Click RClick Enter Exit
Drop Edit	Enter Exit SelChange	Slider	Click Enter Exit RClick
Drop List	Enter Exit SelChange	Spin	Enter Exit SpinUp SpinDown SpinTop SpinBottom
EditField (no multilínea)	Click	Tab	Click Enter Exit

En la siguiente tabla se muestra una lista de las notificaciones que puede mandar un control a su Form:

Evento	Control	Evento	Control
Click	Bitmap Box Box Group Button Group Check Box Edit Field * Push Button Radio Button Slider Tab	Rclick	Bitmap Box Box Group Button Group Check Box List Box Push Button Radio Button Slider
DblClick	Bitmap Box List Box	Selchange	Drop Edit Drop List List Box

Evento	Control	Evento	Control
Exit	Box Group Button Group Check Box Drop Edit Drop List Grid List Box Menu Push Button Radio Button Slider Spin Tab	SpinDown	Spin
SpinUp	Spin	SpinTop	Spin

* El evento Click se puede aplicar a controles de tipo EditField que no son multilínea.

3. Notificaciones de Form

Son eventos mandados y procesados por el Form. Puede ser uno de los indicados a continuación.

Close Esta notificación se recibe cuando se cierra el Form.
Open Esta notificación se le manda al Form cuando se desea ejecutar un conjunto de instrucciones antes de que el Form sea mostrado en pantalla.

4. Notificaciones de Tabla

Las instrucciones incluidas en el código de una notificación mandada por una tabla a su Form se ejecuta después de activarse una operación de edición sobre el Form: Add, Update, Delete, First, Last, Next, Prev, etc.

Las notificaciones que puede mandar una tabla a su Form son:

Notificación	Efecto
AcceptEdit	Esta notificación la manda la tabla activa a su Form cuando se da validez a los datos introducidos en el formulario de dicha tabla. Las instrucciones de esta notificación se ejecutan después de llamar al método AcceptEdit
Add	Esta notificación la manda una tabla a su Form después de agregar una fila a la tabla activa del Form. Las instrucciones de esta notificación se ejecutan después de llamar al método Add
CancelEdit	Esta notificación la manda la tabla activa a su Form cuando se cancela la edición sin dar validez a los datos introducidos en el formulario de dicha tabla. Las instrucciones de esta notificación se ejecutan después de llamar al método CancelEdit
Delete	Esta notificación la manda una tabla a su Form después de borrar una fila de la tabla activa del Form. Las instrucciones de esta notificación se ejecutan después de llamar al método Delete
Edit	Esta notificación la manda la tabla activa a su Form cada vez que se activa la edición de dicha tabla

Notificación	Efecto
Fetch	Esta notificación la manda una tabla a su Form después de hacer cualquier tipo de Fetch sobre la tabla activa del Form.
New	Esta notificación la manda una tabla a su Form después de poner dicha tabla en estado de New. Las instrucciones de esta notificación se ejecutan después de llamar al método New
QueryBegin	Esta notificación la manda una tabla a su Form al inicializar la lista en curso generada por medio de una consulta (Query). Cada vez que se provoque una consulta sobre la tabla en curso se inicializará la lista en curso
QueryEnd	Esta notificación la manda una tabla a su Form al finalizar la generación de la lista en curso por medio de una consulta (Query)
RowChanged	Esta notificación la manda una tabla a su Form cuando cambia la fila activa de la lista en curso del Form
Update	Esta notificación la manda una tabla a su Form después de modificar una fila de la tabla activa del Form. Las instrucciones de esta notificación se ejecutan después de llamar al método Update

Anexo G: Atributos del CTSQL

Contenidos

Introducción

Un atributo es una característica asignada a las columnas que integran las tablas de la base de datos.

Los atributos del CTSQL sólo podrán utilizarse en las instrucciones CREATE TABLE y ALTER TABLE dentro del Lenguaje de Definición de Datos (DDL).

Atributo Check

Este atributo se utiliza para definir una expresión de tipo booleana (condición). Todos los valores de la variable que hagan que dicha expresión evalúe a FALSE serán rechazados.

Su sintaxis es la siguiente:

CHECK (condición)

Parámetro	Significado
Condición	Lista de una o más subcondiciones separadas por operadores AND, OR y NOT. Estas condiciones pueden incluir el signo dólar ($) para sustituir el nombre de la variable a la que se aplica este atributo. Las condiciones válidas son las siguientes: expresión operador_relación expresión

Valor de operador_relación	Significado
=	Igual
!= o <>	Distinto
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que

Valor de operador_relación	Significado		
[NOT]expresiónBETWEENexpr1 AND expr2	Devuelve verdadero o falso en caso de que el resultado de una expresión $ esté o no comprendido en el rango dado por otras dos (ambas inclusive). La partícula NOT invierte el resultado		
[NOT] expresión IN (lista_valores)	Devuelve verdadero o falso en caso de que el resultado de una expresión $ sea igual o no a uno de los valores incluidos en lista_valores. La partícula NOT invierte el resultado.		
	lista_valores	Lista de valores separados por comas	
[NOT] expresión LIKE "literal"	Devuelve verdadero o falso en caso de que el resultado de una expresión se ajuste o no al patrón definido en literal. La partícula NOT invierte el resultado		
	literal	Expresión alfanumérica que puede contener dos metacaracteres que tienen un significado especial:	
		%	Indica cero o más caracteres en la posición donde aparezca
		_	Indica cualquier carácter (uno solo) en la posición indicada
[NOT]expresión MATCHES "literal"	Devuelve verdadero o falso en caso de que el resultado de una expresión se ajuste o no al patrón definido en literal. La partícula NOT invierte el resultado		
	literal	Expresión alfanumérica que puede contener varios metacaracteres que tienen un significado especial:	
		*	Indica cero o más caracteres en la posición indicada
		?	Indica cualquier carácter (uno solo) en la posición donde aparezca.
		l[ista]	Indica que cualquiera de los caracteres incluidos en la «lista» puede aparecer en esa posición. Se puede indicar un rango de caracteres separando el inicial y el final por medio de un guión. Si el primer carácter de lista es ^, indica que los caracteres o rangos no pueden aparecer en esa posición
		\	Indica que el carácter al que precede no debe ser considerado como especial (metacarácter). Siempre precederá a cualquiera de los metacaracteres indicados anteriormente.
expresión IS [NOT] NULL	Devuelve verdadero o falso en caso de que el resultado de una expresión sea o no un valor nulo. La partícula «NOT» invierte el resultado.		

Todas las condiciones indicadas pueden enlazarse mediante los siguientes operadores lógicos:

Operador	Efecto
AND	Devolverá TRUE si ambas condiciones evalúan a TRUE condición AND condición
OR	Devolverá TRUE si al menos una de las condiciones evalúa a TRUE condición OR condición

Ejemplo:

```
create table albaranes (
      albaran integer not null label "Num. Albaran",
      cliente integer not null label "Cod. Cliente",
      fecha_albaran date label "Fecha Albaran",
      fecha_envio date label "Fecha Envio",
      fecha_pago date label "Fecha Pago",
      formpago char(2) label "Cod. F.Pago",
      estado char(1) upshift check ($ in ("S", "N"))
      default "N" label "Facturado");
```

En este ejemplo, la columna *estado* incluye el atributo CHECK indicando que los dos valores posibles que admite son: «S» y «N» (en mayúsculas).

Los paréntesis agrupan operandos y operaciones alterando la precedencia de evaluación de los operadores.

En todos estos puntos, expresión puede sustituirse por el signo «$», lo que indicará que se trata del elemento (variable) que se está definiendo.

Una expresión que devuelva el valor NULL sólo hará cierta la condición IS NULL.

En los casos anteriores sería igualmente válido haber especificado en las condiciones el operador lógico NOT antes de la primera expresión.

Atributo DEFAULT

Este atributo se utiliza para asignar un valor por defecto a la columna de la tabla, en el momento de iniciar la edición en modo agregar para provocar *inser*t de filas. Asimismo, en caso de utilizar la instrucción INSERT del DML y no asignar un valor a la columna con este atributo, por defecto siempre se grabará el valor especificado en el mismo.

Su sintaxis es la siguiente:

DEFAULT valor

Parámetro	Significado
valor	Constante numérica o alfanumérica que se tomará por defecto. Si la variable es de tipo DATE, TIME o CHAR, este valor tiene que ir entre comillas

En caso de no utilizar este atributo, todas las columnas toman por defecto el valor nulo.

En las columnas de tipo DATE y TIME pueden utilizarse las variables internas de CTSQL para asignar valores por defecto. Estas variables son TODAY y NOW, respectivamente.

Ejemplo:

```
create table albaranes (
     albaran integer not null label "Num. Albaran",
     cliente integer not null label "Cod. Cliente",
     fecha_albaran date default today label "Fecha Albaran",
     fecha_envio date default today label "Fecha Envio",
     fecha_pago date label "Fecha Pago",
     formpago char( 2) label "Cod. F.Pago",
     estado char( 1) upshift check ($ in ("S", "N")) default "N" label
     "Facturado");
```

En este ejemplo, tanto la columna *fecha_albaran* como *fecha_envio* tienen asignadas el atributo DEFAULT con el valor de la fecha del sistema (TODAY).

Asimismo, la columna *estado* incluye el atributo DEFAULT indicando que por defecto siempre se grabará el valor N en mayúsculas (UPSHIFT).

Atributo DOWNSHIFT

Este atributo convierte las letras mayúsculas de los valores de una columna de tipo CHAR a minúsculas en el momento de su inserción o actualización (INSERT o UPDATE), respectivamente. En caso de emplear el lenguaje de cuarta generación COOL como interface para el mantenimiento de la tabla en concreto, en la edición de los valores de esta columna también convertirá las mayúsculas a minúsculas.

La sintaxis del atributo es la siguiente:

DOWNSHIFT

Si la tabla ya está creada y se han grabado en ella valores con letras en mayúsculas, al incorporar este atributo a dicha columna mediante una alteración de la tabla (ALTER TABLE) no se convertirán dichos valores.

Si se especifican los atributos DOWNSHIFT y UPSHIFT, CTSQL considerará únicamente el último que encuentre en la definición de la columna.

Ejemplo:

```
create table proveedores(
     proveedor integer not null label "Codigo Proveedor",
     empresa char(25) downshift label "Empresa",
     apellidos char(25) label "Apellidos",
     nombre char(15) label "Nombre",
     direccion1 char(25) label "Direccion1",
     direccion2 char( 25) label "Direccion2",
     poblacion char( 15) label "Poblacion",
     provincia smallint label "Provincia",
     distrito integer label "Distrito",
     telefono char( 13) label "Telefono")
primary key (proveedor) ;
```

En este ejemplo, la columna *empresa* se define con el atributo DOWNSHIFT, lo que significa que no podrá contener ningún carácter en mayúsculas. Si intentásemos insertar o actualizar un valor con caracteres en mayúsculas, éstos se convertirán automáticamente a minúsculas.

Atributo FORMAT

Este atributo determina el formato en el que se presentarán los valores de aquellas columnas en las que se asigne. El atributo FORMAT podrá asignarse a todos los tipos de datos, excepto a CHAR (para éste habrá que utilizar el atributo PICTURE).

Su sintaxis es:

FORMAT "formato"

Parámetro	Significado
Formato	Cadena de caracteres entre comillas que indica el formato

Dependiendo del tipo de dato, los posibles formatos son:

- Numéricos (SMALLINT, INTEGER, DECIMAL, SERIAL y MONEY). Los caracteres que se podrán emplear son los siguientes:

Formato	Significado
#	Representa cada dígito. No cambia los espacios en blanco por ningún otro carácter
. (punto)	El punto indica la posición de la coma decimal. Aparecerá una coma o un punto, dependiendo de la definición de la variable de entorno DBMONEY
, (coma)	Permite indicar la separación de miles, millones, etc. Se imprimirá una coma si la variable de entorno DBMONEY tiene como valor un punto, y un punto si aquél es una coma
*	Rellena los espacios en blanco con asteriscos
&	Rellena los espacios en blanco con ceros
-	Imprime un signo menos cuando la expresión sobre la que se aplica es menor que cero. Cuando se especifican varios signos -, sólo aparecerá uno lo más a la derecha posible
+	Imprime el signo + o -, dependiendo de que la expresión sobre la que se aplica sea mayor o igual a cero (+) o menor que cero (-)
(	Imprime un signo abrir paréntesis antes de un número negativo. Cuando se especifican varios signos (, sólo aparecerá uno lo más a la derecha posible
)	Imprime el signo cerrar paréntesis al final de un formato en el que se ha especificado)
$	Literal que se imprime como un signo de moneda. Cuando se indican varios en la plantilla, aparecerá sólo uno lo más a la derecha posible. Si se ha definido la parte front de la variable de entorno DBMONEY, será esto último lo que incluya el resultado y no el signo $
%	Este literal se imprimirá como tal en la posición donde se especifique. En caso de emplear más de un %, éstos aparecerán en las posiciones asignadas. Estos signos no tienen ningún efecto sobre la expresión a evaluar

El siguiente cuadro recoge de forma resumida algunos ejemplos sobre la utilización de diferentes tipos de formatos:

Números	Formato	Resultado
-13423,41	"##,###.##"	13.423,41
13423,41	"##,###.##"	13.423,41
30	"##,###.##"	30,00
2478	"&&,&&&"	02.478
2076	"**,***"	*2.076
-23453	"+++,+++"	-23.453
-1	"+##,###"	-1
23453	"+++,+++"	+23.453
1	"+##,###"	+1
-6498	"---,---"	-6.498
-25	"-##,###"	-25
-3712	"(++,+++)"	(-3.712)
-50	"---,---%"	-50%
-24	"(---,---%)"	(-24%)

- Fechas (DATE). En este caso los caracteres que se evalúan con un significado para la plantilla de formato son *d, m* e *y*, en las combinaciones que se indican a continuación, representándose cualquier otro carácter como tal.

Formato	Significado
dd	Dos dígitos para el día del mes (01 a 31).
ddd	Tres letras significativas del nombre del día de la semana (Lun a Dom)
mm	Dos dígitos para el mes (01 a 12)
mmm	Tres letras significativas del nombre del mes (Ene a Dic)
yy	Dos dígitos para el año desde 1900 (00 a 99)
yyyy	Cuatro dígitos para el año (0000 a 9999)

Casos prácticos:

Fecha	Formato	Resultado
01/07/1994	dd-mm-yyyy	01-07-1994
01/07/1994	dd ddd / mm mmm / yyyy	01 Vie / 07 Jul / 1994
01/07/1994	dd / mmm / yyyy	01 / Jul / 1994
01/07/1994	dd-mm-yy	01-07-94

- Horas (TIME). En este caso los caracteres que se evalúan con un significado para la plantilla de formato son h, m y s, en las combinaciones que se indican a continuación:

Formato	Significado
Hh:mm:ss	Dos dígitos para la hora, dos para los minutos y dos para los segundos
Hh:mm	Dos dígitos para la hora y dos dígitos para los minutos
hh	Dos dígitos para la hora

Casos prácticos:

Hora	Formato	Resultado
15:10:20	hh:mm:ss	15:10:20
15:10:20	hh:mm	15:10
15:10:20	hh	15

Ejemplo:

```
create table lineas(
   albaran integer not null format"###"label"Num. Albaran",
   linea smallint not null format "##" label "Num. Linea",
   articulo smallint label "Cod. Articulo",
   proveedor integer label "Cod. Proveedor",
   cantidad smallint format "###" label "Cantidad",
   descuento smallint label "Descuento",
   precio money(9,2) label "Precio")
primary key (albaran,linea)
```

Atributo LABEL

Este atributo en CTSQL define por defecto una etiqueta (título) para la columna que se está definiendo. Esta etiqueta será mostrada con las instrucciones de entrada/salida del lenguaje de cuarta generación COOL. Por ejemplo: el método SelectWindow de la clase SqlServer, etc.

La sintaxis de este atributo es la siguiente:

LABEL "literal"

Parámetro	Significado
"literal"	Cadena de caracteres alfanumérica entre comillas

Ejemplo:

```
create table provincias (
     provincia smallint not null label "Cod. Provincia",
     descripcion char(20) label "Provincia",
     prefijo smallint label "Prefijo");
```

En este ejemplo, todas las columnas tienen asignadas una etiqueta (label). Estas etiquetas aparecerán a la hora de hacer una lectura (SELECT) de los datos de la tabla, o bien en cualquier operación de entrada/salida desde el lenguaje de cuarta generación COOL.

Atributo LEFT

Alínea el campo a la izquierda. Su sintaxis es la siguiente:

LEFT

Si no se especifica ningún atributo de alineación, por defecto los datos numéricos se ajustan a la derecha y los tipos de datos CHAR a la izquierda.

En caso de especificar los atributos LEFT y RIGHT, CTSQL considerará únicamente el último que encuentre en la definición de la columna.

Atributo NOENTRY

Este atributo impide insertar un valor en aquella columna donde se define mediante la instrucción INSERT del CTSQL. Sin embargo, sí admitirá dicho valor cuando se actualice una fila mediante la instrucción UPDATE del CTSQL.

Su sintaxis es la siguiente:

NOENTRY

En caso de asignar a una columna los atributos NOT NULL y NOENTRY habrá que definir también el atributo DEFAULT.

Ejemplo:

```
create table albaranes (
     albaran integer not null label "Num. Albaran",
     cliente integer not null label "Cod. Cliente",
     fecha_albaran date label "Fecha Albaran",
     fecha_envio date label "Fecha Envio",
     fecha_pago date label "Fecha Pago",
     formpago char( 2) label "Cod. F.Pago",
     estado char( 1) upshift check ($ in ("S", "N")) noentry
     default "N" label "Facturado");
```

En este ejemplo, cada vez que se dé de alta una fila en la tabla *albaranes*, la columna *estado* no admitirá la asignación de valor alguno. No obstante, el valor que se grabará en todas las filas será *N*, ya que es su valor por defecto. Dicha columna en actualización (UPDATE) podrá admitir los valores *S* y *N*.

Atributo NOT NULL

Este atributo obliga a que la columna tenga un valor distinto de nulo (NULL) al insertar o modificar un fila de una tabla.

Su sintaxis es la siguiente:

NOT NULL

CTSQL utiliza NULL como valor por defecto en la inserción de nuevas filas sobre una tabla. A la hora de agregar una columna con este atributo a una tabla ya existente, es aconsejable asignar un valor por defecto. Esta operación la realiza el atributo DEFAULT. Esta recomendación se debe a que si la tabla ya contiene filas, la columna nueva se agregará con el valor por defecto en cada una de las filas existentes. Sin embargo, en caso de no especificar valor por defecto se producirá un error en dicha operación.

Ejemplo:

```
create table provincias (
     provincia smallint not null label "Cod. Provincia",
     descripcion char(20) label "Provincia",
     prefijo smallint label "Prefijo")
     primary key (provincia));
```

En este ejemplo, se asigna el atributo NOT NULL a la columna que integrará la clave primaria de la tabla *provincias*.

Todas las columnas que integran la clave primaria de una tabla han de tener asignado el atributo NOT NULL.

Atributo NOUPDATE

Este atributo impide actualizar un valor en aquella columna donde se define mediante la instrucción UPDATE del CTSQL. No obstante, sí admitirá dicho valor cuando se inserte una fila mediante la instrucción INSERT del CTSQL.

Su sintaxis es la siguiente:

NOUPDATE

Este atributo se asignará a aquellas columnas cuyo valor insertado (INSERT) no se pueda modificar. En caso de asignar este atributo a la(s) columna(s) que integran la clave primaria (primary key), su acción será similar a los controles originados por la programación de la integridad referencial entre dos tablas.

Ejemplo:

```
create table formpagos (
     formpago  char(2)  not  null  noupdate  label  "Cód.  F.  Pago",
descripcion char(20) label "Forma de pago")
primary key (formpago);
create table albaranes (
     albaran integer not null label "Num. Albaran",
     cliente integer not null label "Cod. Cliente",
     fecha_albaran date label "Fecha Albaran",
     fecha_envio date label "Fecha Envio",
     fecha_pago date label "Fecha Pago",
     formpago char( 2) label "Cod. F.Pago",
     estado char( 1) upshift check ($ in ("S", "N")) default "N" label
     "Facturado")
primay key (albaran)
foreign key for2_alb (formpago) references formpagos
     on update restrict
     on delete restrict;
```

En este ejemplo, una vez asignado un valor en la inserción de la fila, la columna *formpago* de la tabla *formpagos* nunca podrá actualizarse por dos razones:

- Tiene asignado el atributo NOUPDATE.
- En la base de datos existen tablas, por ejemplo «albaranes», que tienen integridad referencial con la tabla *formpagos* mediante la columna *formpago*. Esta integridad referencial especifica que en actualizaciones no puede existir cambio de valor de la columna *formpago* en la tabla *formpagos* (RESTRICT), siempre y cuando existan filas compartiendo dicho valor en la tabla de *albaranes*. Y lo mismo ocurre en la operación de borrado (DELETE).

Atributo PICTURE

Este atributo especifica la máscara de edición de una columna de tipo CHAR.

Su sintaxis es la siguiente:

PICTURE "máscara"

Parámetro	Significado
máscara	Cadena de caracteres que especifica el formato deseado. Esta cadena tiene que ir necesariamente entre comillas

Los valores posibles de máscara son:

Símbolo	Representación
A	Cualquier carácter alfabético
#	Cualquier carácter numérico
X	Cualquier carácter alfanumérico

Cualquier otro carácter se tratará como un literal y aparecerá en el campo en la misma posición en que se especifique en la máscara.

Ejemplo:

```
alter table clientes add
     (cif char(15) picture "A##/###########" upshift);
```

En este ejemplo, la columna *cif* tiene una máscara que indica lo siguiente:

- El primer carácter debe ser una letra (A-Z) en mayúsculas, ya que también incluye el atributo UPSHIFT.
- El segundo y tercer carácter tiene que ser un número (0-9).
- El cuarto carácter siempre será una barra (/).
- A partir del cuarto carácter hasta el final tienen que ser números (0-9).

Atributo RIGHT

Ajusta el campo a la derecha. Su sintaxis es la siguiente:

RIGHT

Si no se especifica ningún atributo de alineación, por defecto los datos numéricos se ajustan a la derecha y los tipos de datos CHAR a la izquierda.

En caso de especificar los atributos RIGHT y LEFT, CTSQL considerará únicamente el último que encuentre en la definición de la columna.

Atributo UPSHIFT

Este atributo convierte las letras minúsculas de los valores de una columna de tipo CHAR en mayúsculas en el momento de su inserción o actualización (INSERT o UPDATE), respectivamente.

Su sintaxis es la siguiente:

UPSHIFT

Si la tabla ya está creada y en ella se han grabado valores con letras en minúsculas, al incorporar este atributo a dicha columna mediante una alteración de la tabla (ALTER TABLE) no se convertirán dichos valores.

En caso de coincidir el atributo UPSHIFT con DOWNSHIFT, CTSQL obedecerá al último que se encuentre en la definición de la columna.

Ejemplo:

```
create table proveedores(
     proveedor integer not null label "Codigo Proveedor",
     empresa char(25) upshift label "Empresa",
     apellidos char(25) label "Apellidos",
     nombre char(15) label "Nombre",
     direccion1 char(25) label "Direccion1",
     direccion2 char( 25) label "Direccion2",
     poblacion char( 15) label "Poblacion",
     provincia smallint label "Provincia",
     distrito integer label "Distrito",
     telefono char( 13) label "Telefono")
primary key (proveedor);
```

En este ejemplo, la columna *empresa* se define con el atributo UPSHIFT. Esto significa que no podrá contener ningún carácter en minúsculas. En el caso de que intentásemos insertar o actualizar un valor con caracteres en minúsculas, éstos se convertirán automáticamente a mayúsculas.

Atributo ZEROFILL

Este atributo alinea a la derecha el valor, sin tener en cuenta el tipo de dato (numérico o alfanumérico), y rellena con ceros por la izquierda. Este atributo produce automáticamente RIGHT.

Su sintaxis es la siguiente:

ZEROFILL

Ejemplo:

```
create table clientes(
     cliente integer not null label "Codigo Cliente",
     empresa char( 25) upshift label "Empresa",
     apellidos char( 25) label "Apellidos",
     nombre char( 15) label "Nombre",
     direccion1 char( 25) label "Direccion1",
     direccion2 char( 25) label "Direccion2",
     poblacion char( 15) label "Poblacion",
     provincia smallint label "Provincia",
     distrito integer label "Distrito" zerofill,
     telefono char( 13) label "Telefono",
     formpago char( 2) label "Forma de Pago",

total_factura money( 11, 2) label "Total Facturado")
primary key (cliente)
```

En este ejemplo, la columna *distrito* tiene asignado el atributo ZEROFILL. En caso de no incorporar este atributo, los códigos postales que comiencen por cero (aunque se tecleen los ceros de la izquierda), se perderían al tratarse de un tipo de dato numérico (INTEGER). Por ejemplo: El distrito *08031* de Barcelona aparecería como *8031*. Sin embargo, si se incluye el atributo ZEROFILL como en nuestro ejemplo, no será necesario teclear los ceros de la izquierda, ya que dicho atributo los incluye automáticamente.

Anexo H:
Aceleradores de Teclado

Contenidos

1. Introducción

Los aceleradores del teclado son combinaciones de teclas que pueden utilizarse en los editores de COSMOS para moverse por el documento, seleccionar, copiar y pegar texto, ...

2. Teclas de menú

Las teclas o combinaciones de teclas que se indican a continuación pueden utilizarse en Editor de Código para seleccionar menús y sus opciones.

Teclas	Efecto
[ALT] o [F10]	Activa o cancela la primera persiana del menú principal
[ALT]+Carácter subrayado	Activa la persiana del menú principal correspondiente al carácter subrayado indicado.
[Return]	Ejecuta el comando u opción seleccionado
[Esc]	Cancelar la última operación realizada
[F. izquierda] y [F. derecha]	Moverse de un menú a otro
[F. arriba] y [F. abajo]	Moverse de una opción de menú a otra dentro de una persiana
[F3]	Salva el fichero
[F1]	Salva el fichero y cierra el editor
[Esc]	Cierra el Editor

3. Teclas de edición

Las siguientes combinaciones de teclas permiten realizar ciertas operaciones de edición con el texto del documento:

Teclas	Efecto
[Mayúsculas]+[Del] o Ctrl]+[X]	Borra el texto seleccionado y lo guarda en el portapapeles
[Ctrl]+[Insert] o [Ctrl]+[C]	Copia el texto seleccionado y lo guarda en el portapapeles
[Mayúsculas]+[Insert] o [Ctrl]+[V]	Pega el contenido del portapapeles
[Supr]	Borra el texto seleccionado
[TAB]	Si se ha seleccionado una o más líneas completas indenta dichas líneas hacia la derecha
[Mayúsculas]+[TAB]	Si se ha seleccionado una o más líneas completas indenta dichas líneas hacia la izquierda
[Ctrl] + [A]	Recupera los cambios que han sido anulados mediante la opción *Undo*
[Ctrl] + [Z]	Anula la última operación de edición
[F. izquierda] y [F. derecha]	Mueve el control seleccionado hacia la izquierda o la derecha

4. Teclas de movimiento del cursor

Las teclas que se indican a continuación pueden utilizarse para mover el cursor o punto de inserción dentro de un cuadro de texto o en cualquier otro lugar donde se puede escribir texto.

Teclas	Efecto
[Inicio] u [Home]	Mueve el cursor al principio de la línea
[Fin] o [End]	Mueve el cursor al final de la línea
[RePág]	Una página arriba
[AvPág]	Una página abajo
[F. arriba]	Desplaza el cursor una línea hacia arriba
[F. abajo]	Desplaza el cursor una línea hacia abajo
[F. derecha]	Mueve el cursor un carácter hacia la derecha
[F. izquierda]	Mueve el cursor un carácter hacia la izquierda
[Ctrl]+[F. derecha]	Mueve el cursor una palabra hacia la derecha
[Ctrl]+[F. izquierda]	Mueve el cursor una palabra hacia la izquierda
[Ctrl]+[Inicio]	Sitúa el cursor al principio del documento
[Ctrl]+[Fin]	Sitúa el cursor al final del documento

5. Teclas de selección de texto

Las selecciones que se describen a continuación comienzan siempre a partir del punto en que se encuentre situado el cursor:

Teclas	Efecto
[Ctrl]+[Mayúsculas]+[F. izquierda]	Selecciona la palabra anterior
[Ctrl]+[Mayúsculas]+[F. derecha]	Selecciona la palabra siguiente
[Ctrl]+[Mayúsculas]+[Inicio]	Selecciona el texto hasta el principio del documento
[Ctrl]+[Mayúsculas]+[Fin]	Selecciona el texto hasta el final del documento
[Mayúsculas]+[Inicio]	Selecciona el texto hasta el principio de la línea
[Mayúsculas]+[Fin]	Selecciona el texto hasta el final de la línea
[Mayúsculas]+[RePág]	Selecciona el texto una pantalla hacia arriba
[Mayúsculas]+[AvPág]	Selecciona el texto una pantalla hacia abajo
[Mayúsculas]+[F.derecha/izquierda]	Selecciona carácter a carácter hacia derecha o izquierda
[Mayúsculas]+[F. Arriba/abajo]	Selecciona el texto línea a línea hacia arriba o hacia abajo

Títulos de la Colección

Ingeniería Informática

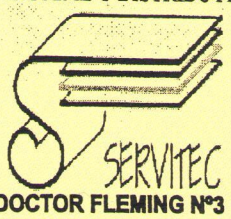
Nº CUADERNO	TITULO	ISBN
1	ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS	84-8497-802-8
2	DISEÑO DE PÁGINAS WEB USANDO HTML	84-8497-803-6
3	FORMATOS GRÁFICOS	84-8497-804-4
4	PROGRAMACIÓN EN LENGUAJE C	84-8497-805-2
5	ADMINISTRACIÓN DE WINDOWS N 4.0	84-8497-899-0
6	GUÍA DE USUARIO DE DERIVE PARA WUIDOWS	84-8497-919-9
7	GUÍA DE REFERÊNCIA DE MULTIBASE COSMOS	84-8416-001-7
8	GESTIÓN DE UN TALLER MULTIBASE CÓSMOS	84-8416-034-3
9	EJERCICIOS DE LÓGICA INFOMÁTICA	84-8416-357-1
10	CONEPTOS BÁSICOS DE PROCESADORES DE LENGUAJE	54-8416-889-1
11	INTRODUCCIÓN A LA ADMINISTRACIÓN DE UNIX	84-8416-570-1
12	LÓGICA PROPOSICIONAL PARA LA INFOMÁTICA	84-8416-613-9
13	PROGRAMACIÓN PRÁCTICA EN PROLOG	84-8416-612-0
14	LA HERRAMIENTA CASE META COSMOS	84-699-0054-4
15	GUIA DE LENGUAJE COOL DE MULTIBASE COSMOS	84-699-0053-6
16	GUÍA DE REFERÊNCIA PROGRESS	84-699-2083-9
17	GESTIÓN DE DEPOSITO DENTAL CON PROGRESS	84-699-2082-0
18	GUIA DE DISEÑO Y CONS. REPOSICIÓN MUL. COSMOS	84-699-2081-2
19	GESTIÓN Y ADMIN. DE UN COLEGIO MAYOR MUL-COSMOS	84-699-2080-4
20	MODELADO DE SOFWARE EN UML	84-699-2079-0
21	EL LENGUAJE DE PROGRAMACIÓN JAVA	84-699-3880-0
22	PRINCIPIOS DE ALGORITMIA	84-699-6537-9
23	LISTAS, PILAS Y COLAS	84-699-6536-0
24	RECURSIVIDAD	84-699-6535-2
25	ARBOLES	84-699-6849-1
26	CONJUNTOS Y TABLAS DE DISPERSIÓN	84-699-7398-3
27	ALGORITMOS DE ORDENACIÓN	84-699-7397-5
28	TEORÍA DE GRAFOS	84-699-8362-8
29	IINTR. AL PROCES. EFEC. DE TEXTOS CON MICROSOFT WORD	84-699-9618-5
30	ORACLE SQL	84-688-0420-7
31	TÉCNICAS DE DISEÑO DE ALGORITMOS	84-688-1764-3
32	LA PLATAFORMA . NET	84-688-3449-1
33	EJERC. DE HOJAS DE CÁL. EXCEL APLICADOS A LA GES.EMPRES.	84-688-3450-5
34	TIPOS ABSTRACTOS DE DATOS	84-688-4209-5
35	INTERPRETES T DISEÑO DE LENGUAJES DE PROGRAMACIÓN	84-688-4210-9
36	LENGUAJES Y AUTOMATAS EN PROCESADORES DE LENGUAJE	84-688-4211-7
37	METODOLOGIA DE LA PROGRAMACIÓN: GUIA DEL ALUMNO	84-688-5901-4
38	ANÁLISIS SEMÁNTICO EN PROCESADORES DE LENGUAJE	84-688-6208-8
39	ARQUITECTURA WEB EN APLICACIONES JAVA/J2EE	84-688-6580-9
40	ALGORITMICA CON FORTRAN 90	84-688-7250-4
41	TABLAS DE SÍMBOLOS EN PROCESADORES DE LENGUAJES	84-688-7631-3
42	INTRODUCCIÓN A LA COMINICACIÓN PERSONA MÁQUINA	84-688-8362-X
43	INTRODUC. A LA PROG. ORIENTADA A OBJETOS CON JAVA	84-688-8826-6
44	DESARR. APLIC. EN SISTEM. DISTRIBUIDOS E INTERNET	84-689-3379-1
45	LÓGICA DE PREDICADOS	84-689-3380-5
46	LENGUAJE C#	84-689-5893-X
47	INTEGRACIÓN DE APLICACIONES OFIMÁTICAS	84-689-5892-1
48	INFORMÁTICA GENERAL	84-689-7033-6
49	PROYECTOS INFORMÁTICOS	

IMPRIME Y DISTRIBUYE

C/DOCTOR FLEMING Nº3
33005 OVIEDO
TLF-FAX 985 250581
E-mail:copisteriaservitec@fade.es

Consultor Editorial
Juan Manuel Cueva Lovelle
cueva@lsi.uniovi.es

IMPRIME Y DISTRIBUYE



C/DOCTOR FLEMING Nº3
33005 OVIEDO
TLF-FAX 985 250581
www.fade.es/copisteriaservitec
E-mail:copisteriaservitec@fade.es