ESKER Tun® Plus

Tun SQL – Acceso a datos

ESKER®
Document Process Automation

Tun Plus 2009
Issued May 2008

Esker S.A., 10 rue des Émeraudes, 69006 Lyon, France
Tel: +33 (0)4.72.83.46.46 ♦ Fax: +33 (0)4.72.83.46.40 ♦ info@esker.fr ♦ www.esker.fr

Esker, Inc., 1212 Deming Way, Suite 350, Madison, WI 53717 USA
Tel: +1.608.828.6000 ♦ Fax: +1.608.828.6001 ♦ info@esker.com ♦ www.esker.com

Esker Australia Pty Ltd. (Lane Cove - NSW) ♦ Tel: +61 (0)2 8596 5100 ♦ info@esker.com.au ♦ www.esker.com.au
Esker GmbH (München) ♦ Tel: +49 (0) 89 700 887 0 ♦ info@esker.de ♦ www.esker.de
Esker Italia SRL (Milano) ♦ Tel: +39 02 57 77 39 1 ♦ info@esker.it ♦ www.esker.it
Esker Ibérica, S.L. (Madrid) ♦ Tel: +34 91 552 9265 ♦ info@esker.es ♦ www.esker.es
Esker UK Ltd. (Derby) ♦ Tel: +44 1332 54 8181 ♦ info@esker.co.uk ♦ www.esker.co.uk

PREFACIO

Tun SQL - Acceso a datos es una suite de aplicaciones y servidores que permite a los PCs trabajar en modo cliente/servidor con bases de datos remotas (Informix, Oracle, Sybase, DB2, Progress y C-ISAM). Tun SQL utiliza la arquitectura ODBC definida por Microsoft.

Tun SQL funciona en las siguientes plataformas: Windows 3.x, Windows 95, Windows 98, Windows NT 3.51, Windows NT 4.0, Windows 2000, Citrix WinFrame, Citrix MetaFrame et Windows NT TSE.

Tun SQL es una parte de la gama de productos de software **Tun** tal y como se muestra a continuación:

	Funcionalidades en Windows	**Componenetes en entorno multiusuario**
Esker TCP/IP Stack	Pila de comunicaciones TCP/IP para Windows 3.x (DLL)	N/D
Acceso a recursos de la red	Aplicaciones TCP/IP (NIS, Cliente y Servidor NFS, PING, Redirección y compartición de impresoras, Cliente y Servidor FTP, TELNET, Cliente y Servidor RSH, TAR, WALL, TFTP, TIME)	Aplicaciones TCP/IP (NIS, Cliente y Servidor NFS, PING, Redirección y compartición de impresoras, Cliente y Servidor FTP, TELNET VT320, Cliente RSH, TAR, WALL)
Acceso a aplicaciones	Emulador de terminal (asíncrono, emulación IBM3270 e IBM5250, impresoras 3287/3812)	Emulador de terminal (asíncrono, emulación IBM3270 e IBM5250, impresoras 3287/3812)
Acceso a datos	Drivers ODBC para modelo Cliente/Servidor TCP/IP (SGBD Oracle, Informix, Sybase, DB2, Progress y C-ISAM) y herramienta de conversión de bases de datos	Drivers ODBC para modelo Cliente/Servidor TCP/IP (SGBD Oracle, Informix, Sybase, DB2, Progress y C-ISAM) y herramienta de conversión de bases de datos
TCP/IP Network Services	Navegador NIS, redirección y compartición de impresoras	Redirección y compartición de impresoras

La mayoría de las funcionalidades y procedimientos que se describen en este manual se aplican igual en Windows 3.x, Windows 95, Windows 98, Windows NT 3.51, Windows NT 4.0, Windows 2000 o Citrix/ WindowsNT TSE. Sin embargo, algunas funcionalidades o procedimientos sólo se aplican a una o algunas de estas plataformas. En este caso, el párrafo o la sección en cuestión se marcará de la siguiente manera:

Windows 3.x

Windows 95 y Windows 98

Windows NT (Windows NT 3.51 y Windows NT 4.0 que incluyan entorno multiusuario, a no ser que se indique lo contrario) y Windows 2000

Windows NT 4.0 que incluyan entorno multiusuario, a no ser que se indique lo contrario

Windows NT 3.51 que incluyan entorno multiusuario, a no ser que se indique lo contrario

Windows 32 bits (Windows 95, Windows 98, Windows NT 3.51 y Windows NT 4.0 que incluyan entorno multiusuario, a no ser que se indique lo contrario y Windows 2000)

Entorno multiusuario

Exepto entorno multiusuario

Tun SQL para Windows también se incluye en **Tun PLUS** que además incluye todos los módulos mencionados con anterioridad. El procedimiento de instalación de **Tun PLUS** permite instalar **Tun SQL**. Excepto en la versión de **Tun PLUS** para Citrix/Microsoft NT TSE, se puede instalar **Tun SQL** independientemente de **Tun PLUS**.

Nota
En todo el documento, los características para Windows 98 equivaldran a los de Windows 95.

TABLA DE CONTENIDOS

PARTE PRIMERA
PRESENTACIÓN Y USO

1

INTRODUCCIÓN A TUN SQL

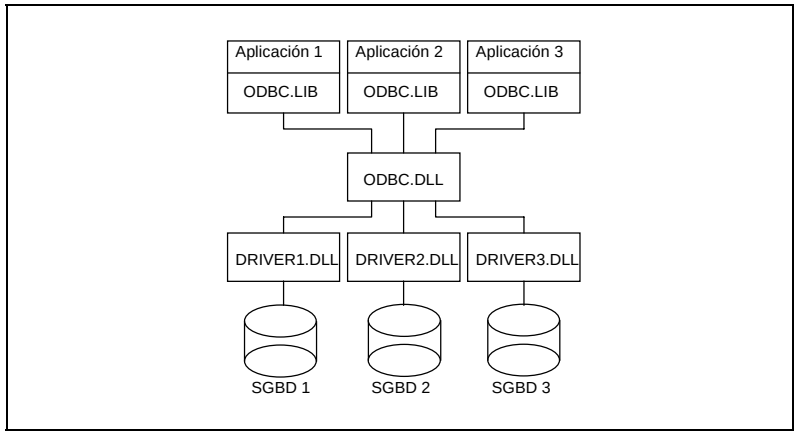
El mecanismo ODBC

En el mundo de las bases de datos, los programadores utilizan tradicionalmente un mecanismo llamado "SQL Embebido" para proporcionar un interface entre sus aplicaciones y una base de datos específica. El mecanismo SQL Embebido hace posible insertar peticiones en programas escritos en COBOL o C. Ofrece la ventaja de hacer las aplicaciones más portables a diferentes máquinas.

El mecanismo SQL Embebido, sin embargo, tiene varias desventajas:

- Hay tantos SQLs Embebidos como SGBD hay en el mercado. Las aplicaciones que utilizan SQL sólo se pueden comunicar con un SGBD al mismo tiempo. Deben ser reescritos o al menos modificados si se van a comunicar con otras bases de datos. Para aplicaciones que van a acceder a todas las bases de datos del mercado, es inconcebible la utilización del mecanismo SQL Embebido.
- El mecanismo SQL Embebido está relativamente sin desarrollar, ya que es restrictivo y difícil de utilizar. No permite sacar el máximo partido de las bases de datos, y a veces es preferible utilizar la API suministrada por el SGBD directamente.

Para paliar estas desventajas del mecanismo SQL Embebido, Microsoft concibió una nueva aproximación basada en el mecanismo ODBC (Open Database Connectivity) que está basado en WOSA (Windows Open System Architecture).



ODBC es un conjunto de funciones en C bien definidas que hacen posible recuperar o actualizar datos en un SGBD. Estas funciones han sido ensambladas en una DLL (Dynamic Link Library) que puede ser utilizada por cualquier aplicación Windows. Las funciones del DLL ODBC (ODBC.DLL) analizan las peticiones SQL y se procesan por los controladores ODBC cuyo trabajo es convertir las llamadas al API particular del SGBD que desea utilizar. El controlador ODBC permite ver el interface del SGBD y a la aplicación le permite utilizarlo como cualquier SGBD que cumpla ODBC.

Microsoft proporciona la librería ODBC.DLL y todas las herramientas necesarias para utilizarla; sin embargo, no proporciona los controladores ODBC para todos los SGBD del mercado. A Microsoft le basta con proporcionar los controladores para sus sistemas de almacenamiento propietarios (Access, Excel, Word...).

Los controladores ODBC específicos para las bases de datos pueden ser suministrados directamente por el propietario del SGBD o por terceras partes especializadas en este campo (Esker).

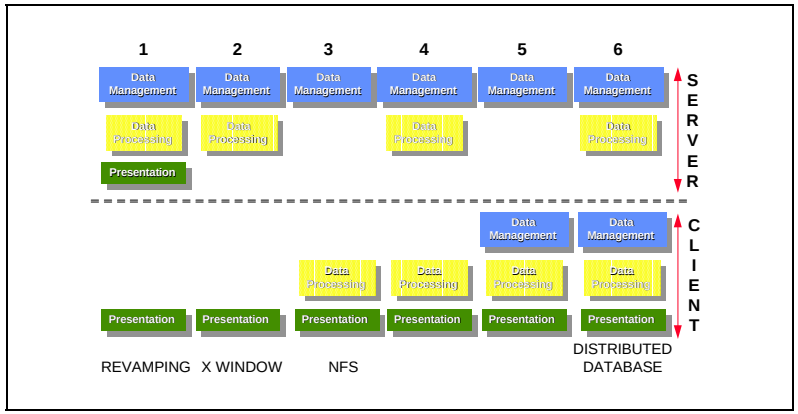
Finalmente, el mecanismo ODBC proporciona la máxima interoperabilidad. Una aplicación simple de Windows puede acceder a diferentes sistemas de gestión sin tener que haber sido diseñada necesariamente con esta idea. El ODBC permite a los desarrolladores programar, compilar y entregar sus programas sin tener que preocuparse del SGBD sobre el que se utilizará el programa. Los usuarios sólo tienen que poner el controlador adecuado para que la aplicación con la que están trabajando pueda cooperar con el SGBD que hayan elegido.

El ODBC es un mecanismo con valor para aplicaciones multi - dominio como las hojas de cálculo, aplicaciones de proceso de texto y herramientas de desarrollo que pueden manipular información de cualquier SGBD sin conocer a priori que SGBD se va a utilizar.

El modelo Cliente/Servidor

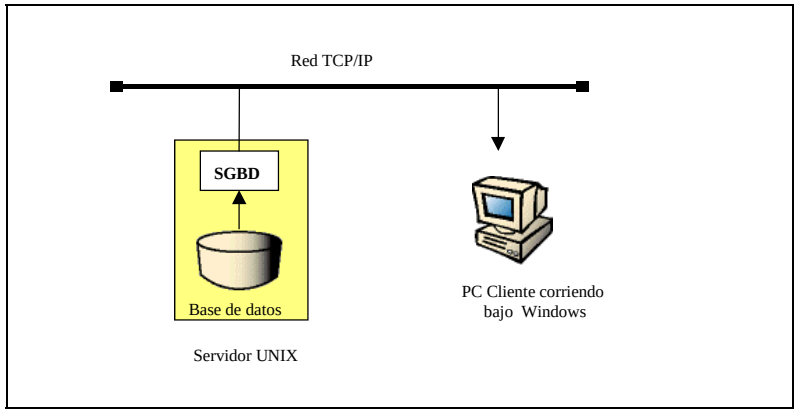
Durante algunos años, Cliente/Servidor ha sido el término que ha estado en los labios de todas las personas relacionadas con la informática. En su sentido más amplio, el modelo Cliente/Servidor es una concepción de comunicación en donde al menos dos unidades cooperan para dar un servicio en particular.

El "Gartner Group" ha identificado seis tipos principales de aplicaciones Cliente/Servidor que se han clasificado de acuerdo con el número de funciones realizadas por el cliente o el servidor:



Las aplicaciones que hacen las menores peticiones al "cliente" son aplicaciones de "conversión" o servidores X-WINDOWS. Las aplicaciones que hacen más peticiones al "cliente" son aquellas que utilizan bases de datos distribuidas.

Cuando la industria de la informática habla del modelo Cliente/Servidor, se suele referir a las aplicaciones relacionadas con bases de datos distribuidas. La ilustración más típica de este modo se muestra en el siguiente esquema:



Un PC "cliente" tiene un sistema de gestión tradicional que trabaja en modo gráfico. Los datos se almacenan centralizados en un servidor UNIX y se gestionan con un Sistema de Gestión de Bases de Datos Relacional (SGBDR). Para recuperar o actualizar datos, el cliente envía peticiones SQL al servidor el cual ejecuta y devuelve la respuesta al cliente a través de la red.

Las ventajas principales de esta arquitectura son las siguientes:

- El usuario final tiene un interface gráfico Hombre/Máquina amigable en su PC bajo sistema operativo Windows.
- El PC se puede utilizar para otras tareas además de las aplicaciones normales (aplicaciones de ofimática, cálculos, aplicaciones personales...).
- Mientras se tenga su propia máquina, el usuario puede tener acceso a los datos centralizados en el servidor al mismo tiempo que otros usuarios.

- El servidor queda relevado de la parte de proceso de aplicaciones y toda su potencia se puede concentrar en servir datos. Hay una distribución equilibrada de la carga de trabajo entre el cliente y el servidor.
- La red sólo se utiliza para enviar datos de la aplicación esenciales; no se sobrecarga con información de presentación.

El modelo Cliente/Servidor SQL tal y como se ha descrito es una versión moderna del modo transaccional que todavía se puede encontrar en entornos Mainframe - terminal síncrono.

ODBC y el modelo Cliente/Servidor SQL

Así como el mecanismo ODBC facilita el acceso a todos los sistemas de gestión de datos, también se puede utilizar en bases de datos remotas. En este contexto, ODBC permite que una aplicación Windows utilice cualquier SGBD esté donde esté.

También permite que aplicaciones multi - dominio como "Excel", "Word" o "Access" utilicen la información centralizada de la empresa. En un contexto Cliente/Servidor, el mecanismo ODBC le permite aumentar el número de aplicaciones que pueden utilizar los datos de la empresa.

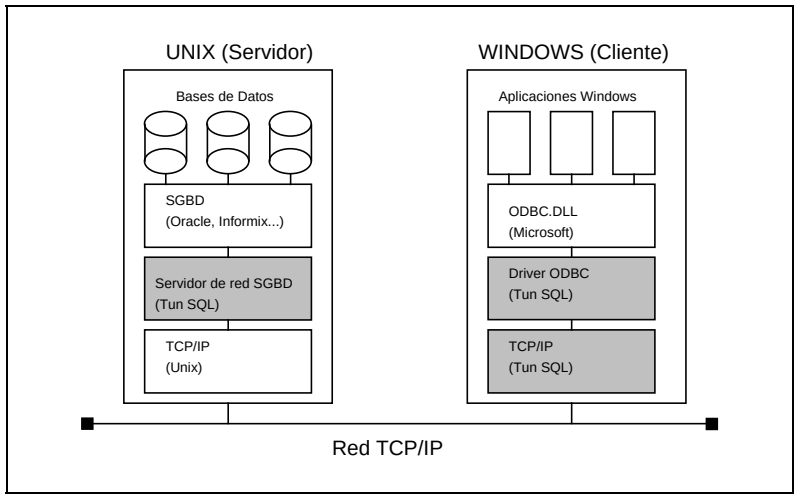
En cualquier caso, la implementación de esta arquitectura hoy en día no es algo sencillo. En verdad, un usuario que ya tenga una red de área local de PCs, un servidor UNIX y un SGBD debe también adquirir los siguientes componentes para establecer una arquitectura Cliente/Servidor SQL utilizando ODBC:

- La parte del servidor de red del SGBD (Informix Net, Sql Net...).
- La parte del PC cliente del SGBD (Informix Net PC, Sql Net PC...).
- El controlador ODBC apropiado.
- Las pilas de TCP/IP para el PC compatibles con **Winsock.**

Los propietarios de los SGBD siempre proporcionan los dos primeros componentes. Este no es siempre el caso del controlador ODBC y nunca el caso de las pilas de TCP/IP. Por lo tanto es necesario conseguir los dos últimos componentes en otro sitio prestando particular atención a la cuestión de la compatibilidad.

Tun SQL

Tun SQL se concibió para resolver los problemas definitivamente. Integra todos los componentes mencionados anteriormente, así como potente funcionalidad de revamping de base de datos y un controlador virtual ODBC para acceso a las bases de datos actualizadas en un único paquete de software homogéneo Dentro de **Tun SQL** se incluyen incluso los componentes de la red del SGBD (cliente y servidor). Esto representa un considerable ahorro cuando se va a equipar a varios PCs. Hace que las cosas sean especialmente muy sencillas cuando se va a implementar una arquitectura Cliente/Servidor SQL.



Las principales características de **Tun SQL** son las siguientes.

➢ Sólo un controlador ODBC para la mayoría de los SGBD del mercado

Para limitar el número de productos software necesarios para el PC Cliente, **Tun SQL** proporciona un sólo controlador ODBC para acceder a los siguientes SGBD indiferentemente:

- Oracle versión 7.
- Informix versiones 5 y 7.
- Sybase versión 10.
- DB2 versión 2.

- Progress versiones 6, 7 y 8.
- C-ISAM versiones 4 a 7.

➢ Revamping de base de datos

Tun SQL, con la ayuda de una aplicación integrada orientada al usuario, permite redefinir las tablas de una base de datos y hacerlas más accesibles al usuario final (mediante reorganización personalizada de tabla, cambio de nombres de tabla y campo y funciones predefinidas).

➢ Controlador virtual ODBC para bases de datos actualizadas

Para utilizar una base de datos redefinida mediante actualización, **Tun SQL** dispone de un controlador virtual ODBC que traduce las solicitudes realizadas a tablas virtuales en solicitudes que puede manejar un controlador ODBC normal.

➢ La parte del servidor del SGBD está incluida en Tun SQL

La parte del servidor de cada SGBD se instala bajo Unix y se entrega de forma estándar con **Tun SQL** para los siguientes sistemas operativos:

- ScoUnix 3.2x v.4.2 and 5.0.
- SunOs 4.1.3.
- Solaris 2.5.
- AIX 3.2 and 4.1.
- HP-UX 9.x and 10.x.
- OSF1 v.3.2.

Esta propiedad ahorra a los usuarios de **Tun SQL** el coste de la parte del servidor del SGBD que utilicen.

➢ Las pilas de TCP/IP se entregan de forma estándar

Igual que cualquier software de la gama **Tun**, **Tun SQL** se entrega sistemáticamente con las pilas de TCP/IP de Esker. Las pilas dan unas prestaciones excelentes y han sido probadas con todos los componentes de **Tun SQL**. La inclusión de las pilas en el paquete **Tun SQL** ahorra al usuario el problema de conseguirlas aparte.

➢ Sencillez de instalación y administración

El propósito de **Tun SQL** es facilitar la implementación de una arquitectura Cliente/Servidor basada en Windows y Unix. Consecuentemente, **Tun SQL** tiene un procedimiento simple de instalación para Unix y Windows y viene con una completa documentación.

Además del controlador ODBC, se suministran dos aplicaciones Windows para comprobar e implementar el Cliente/Servidor:

- **Tun DB Show** se usa para comprobar la conexión Cliente/Servidor de extremo a extremo. La aplicación puede encontrar desde el servidor Unix los SGBD instalados y preguntar a algunos SGBD por las bases de datos que contienen.
- **Tun DB Script** ejecuta archivos batch de SQL bajo Windows que crean bases de datos en los SGBD remotos.

Además de la seguridad que ofrece Unix y los diferentes SGBD, **Tun SQL** proporciona un mecanismo que puede denegar el acceso de algunas aplicaciones Windows a bases de datos particularmente delicadas.

Por último, la integración de NIS (Servicio de información de la red) en **Tun SQL** permite la gestión centralizada de los recursos de la red y facilita el acceso a recursos remotos. Para más información sobre NIS; consulte el manual de usuario de servicios de red TCP/IP.

➢ La conformidad del controlador de Tun SQL

El controlador de **Tun SQL** soporta todas las funciones de nivel 1. También soporta algunas funciones de nivel 2. Con el controlador se proporciona la "Microsoft ODBC Cursor Library". Aunque esta librería sólo soporta cursores estáticos y de "sólo avance", es suficiente para muchas aplicaciones.

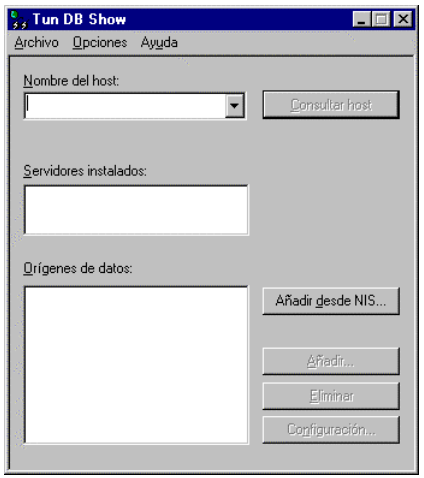
CONFIGURACIÓN Y USO BAJO WINDOWS

Comprobar el funcionamiento de Tun SQL

➢ Ejecutar Tun DB Show

Después de la instalación y configuración de **Tun SQL** bajo Windows y UNIX, es necesario verificar que funciona correctamente. Esto se puede hacer ejecutando la aplicación **Tun DB Show**.

Ejecute el programa haciendo click en el icono **Tun DB Show** del grupo **Data Access** (menú de **Inicio**, **Programas**, **Esker Tun** en Windows 95/98/2000 y Windows NT).

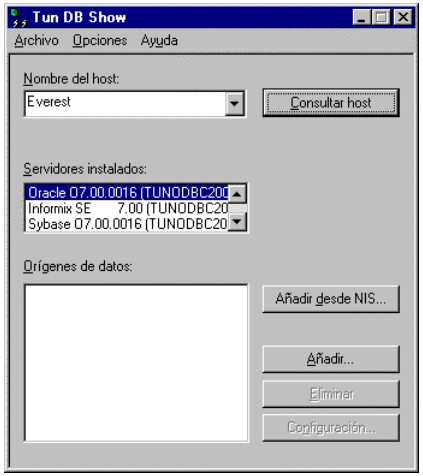


Esta utilidad se puede utilizar para consultar en un servidor UNIX de la red si hay uno o más servidores **Tun SQL** para UNIX presentes. Introduzca el nombre del servidor en el campo **Nombre del host** o seleccionar el servidor en la lista (esta lista muestra los servidores declarados en el archivo **hosts** y en el servidor NIS).

Para más información sobre la configuración de **Tun NIS**, consulte **TCP/IP Network Services** o **Tun NET**.

Pulse el botón **Consultar host**

Si hay instalados correctamente uno o más servidores **Tun SQL** para UNIX en la máquina remota, debe aparecer la lista **Servidores instalados** (con al menos una línea) de la siguiente manera:



Cada línea contiene la siguiente información:

- El nombre del SGBD con el que se comunica el servidor **Tun SQL** para UNIX.
- El número de versión del SGBD.
- El nombre del archivo ejecutable de UNIX que actúa de servidor (por ejemplo tunodbc200.ora)

Nota:
Con algunos SGBD (Informix On-Line, por ejemplo), al seleccionar el servidor en la lista se muestra en la columna correspondiente (Bases de Datos) la lista de bases de datos administradas por el SGBD.

Si no hay servidores **Tun SQL** en la lista, significa que ha habido algún problema durante la instalación. En este caso hay que ejecutar otra vez todas las operaciones y comprobaciones descritas en los capítulos anteriores.

➢ Parámetros

Usar el menú **Opciones** de **Tun DB Show** para:

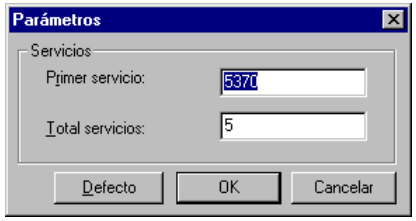
- Administrar servicios.
- Configurar el servidor proxy.

Administración de servicios

Se asocia un número a partir de 5370 con cada uno de los procesos de servidor de **Tun SQL**. El funcionamiento correcto de **Tun SQL** precisa la definición de un primer servicio y el número de servicios posibles, lo que le permite detectar los distintos sistemas de base de datos accesibles a un servidor **Tun SQL**. El valor predeterminado de **Primer servicio** es 5370 y el de **Total servicios** es de 5. La lista de servicios que pueden ser usados es la siguiente:

- Oracle 5370
- Informix 5371
- Sybase 5372
- DB2/RS6000 5373
- Progress 6 5374
- Progress 7 5375
- C-ISAM 5376
- DB2 for MVS 5377
- Progress 8 5378

Seleccionar **Opciones➔Parámetros...** en el menú principal. Se abre este cuadro de diálogo:

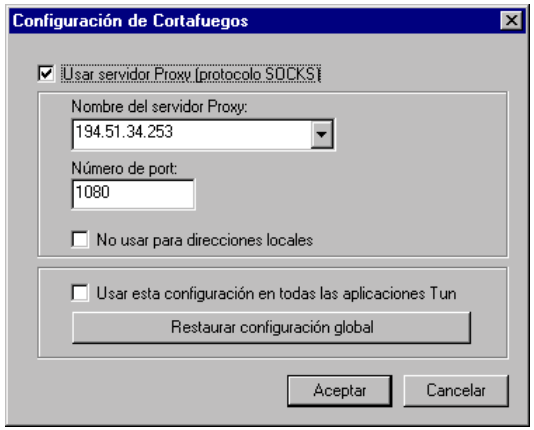


Uso de un servidor proxy como cortafuegos

Cuando se configura un servidor proxy en Tun DBRevamp, el acceso a servidores externos se realiza a través de un gateway proxy.

Para establecer los parámetros del cortafuegos (dirección IP, número de port, etc), seleccionar **Opciones➔Cortafuegos** en el menú principal.

Se abre este cuadro de diálogo:



Activar la casilla de verificación **Usar servidor proxy**.

Escribir el nombre o la dirección IP del servidor. Escribir sólo un nombre si se utiliza DNS. También se puede seleccionar un servidor en la lista desplegable (pulsar la flecha hacia abajo a la derecha del campo). La lista contiene los nombres de los servidores registrados en la tabla de servidores (hosttab) y en el servidor NIS (los recursos NIS tienen iconos amarillos).

También, escribir el número de port para el protocolo SOCKS (generalmente, el 1080, el valor por defecto).

Si no se quiere usar el gateway para comunicaciones en red local, activar la casilla de verificación **No usar para direcciones locales**.

Se puede aplicar la configuración del cortafuegos a todas las aplicaciones Tun de la máquina activando la casilla de verificación **Usar esta configuración en todas las aplicaciones Tun**. Para aplicar la configuración general a todas las aplicaciones **Tun** (después de utilizar configuraciones especiales en **Tun *NFS***, por ejemplo) pulsar en **Restaurar configuración global**.

Creación de una base de datos

Para familiarizar a los usuarios con la funcionalidad de ODBC en un entorno cliente/servidor, **Tun SQL** se entrega con ejemplos prácticos.

Para utilizar los ejemplos suministrados con el paquete de software **Tun SQL**, hay que crear una base de datos específica en uno de los SGBD disponibles. Esto se puede hacer con las herramientas del SGBD en cuestión. Debido a la dificultad para crear una base de datos en algunos SGBD (Oracle), se puede utilizar una base de datos que se entrega y que no contiene datos importantes. Es preferible llamar a la nueva base de datos **tunsqldemo** para una mejor comprensión de la siguiente parte de la documentación.

Creación de una fuente de datos

➢ Introducción a las fuentes de datos

Debido a que se puede utilizar un controlador y una base de datos en particular, las aplicaciones que cumplen ODBC tienen que reconocer una Fuente de Datos. **Fuente de Datos** es un término que se refiere al nombre del controlador ODBC utilizado (por ejemplo, **tunodb32.dll**) y a la información necesaria para hacerlo funcionar. Esta información es la siguiente:

- El nombre o la dirección IP del servidor UNIX remoto.
- El tipo de SGBD (Oracle, Informix, Sybase, DB2, Progress, C-ISAM).
- El nombre de la base de datos.
- Comentarios.
- Información adicional.

➢ Creación de una fuente de datos

Los ejemplos que se entregan con **Tun SQL** usan la misma fuente de datos. datos (excepto los ejemplos sobre bases de datos virtuales).
Es necesario crear esta fuente de datos antes de poder usar los ejemplos (consulte "**Bien démarrer avec Tun**").

Para crear una fuente de datos se puede utilizar:

- **Tun DB Show**.
- **Administrador ODBC** (utilidad de Windows).

Creación de una fuente de datos con Tun DB Show

Arranque **Tun DB Show** otra vez y realice las siguientes operaciones

- Escriba el nombre o la dirección IP del servidor donde está instalada la base de datos para la cual quiere crear la fuente de datos.
- Pinche en el botón **Buscar servidores** para mostrar los servidores **Tun SQL** instalados en la máquina UNIX.
- Seleccione el servidor apropiado para el SGBDR correspondiente a la base de datos con la que se quiere conectar.
- Pinche en el botón **Añadir...**.

Si **Tun NIS** está instalado en el PC y el administrador de red ha configurado las tablas NIS, se puede usar el botón **Añadir desde NIS…** para acceder a las fuentes de datos de la red. Para la configuración de **Tun NIS**, consulte el manual **TCP/IP Network Services** o **Tun NET**.

A continuación vea la sección "**Configuración de la fuente de datos**".

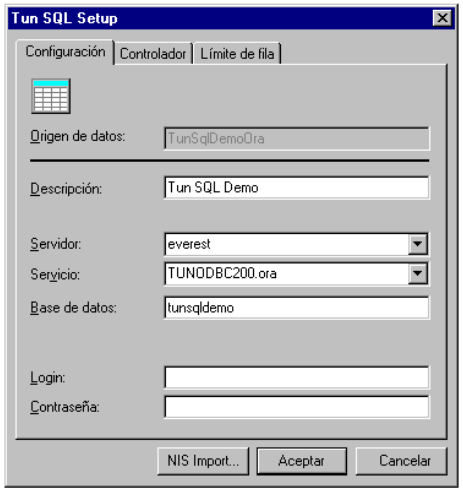
Creación de una fuente de datos con el Administrador ODBC

Abra el **Panel de control** y pinche en el icono **ODBC**. En el cuadro de diálogo, pulse en el botón **Agregar**.

Seleccione el controlador ODBC **Tun32 Driver**.

Configuración de la fuente de datos

Pinche en el botón **Añadir** en **Tun DB Show,** o a través del **Administrador ODBC** como se describe previamente, para mostrar este cuadro de diálogo:



➢ Configuración

Estos son los campos de la primera solapa:

Origen de datos

El icono  representa el campo que contiene el nombre de la fuente de datos utilizada por las aplicaciones ODBC. Para poder conectarse a la base de datos de ejemplo, es imprescindible que la fuente de datos se llame:

- **TunSqlDemoIfx** Para Informix On-Line.
- **TunSqlDemoIse** Para Informix SE.
- **TunSqlDemoOra** Para Oracle.
- **TunSqlDemoSyb** Para Sybase.
- **TunSqlDemoDB2** Para DB2.
- **TunSqlDemoPro** Para Progress

Descripción

Este campo contiene un comentario asociado con la Fuente de Datos.

Servidor

Este campo contiene la dirección IP o el nombre del servidor donde está instalada la base de datos que el usuario quiere utilizar.

Servicio

Este campo contiene el nombre del proceso del servidor **Tun SQL** asociado con el SGBD en donde se creó la base de datos requerida (por ejemplo, **tunodbc200.ora**).

Si se utilizan pilas de TCP/IP distintas de **TCP/IP Stack**, se deben rellenar los servicios de archivo en el software del TCP/IP con los siguientes valores:

```
tunodbc200.ora  5370/tcp      # Tun-SQL ORACLE
tunodbc200.ifx  5371/tcp      # Tun-SQL INFORMIX
tunodbc200.syb  5372/tcp      # Tun-SQL SYBASE
tunodbc200.db2  5373/tcp      # Tun-SQL DB2
tunodbc200.pro  5374/tcp      # Tun-SQL PROGRESS
```

Los otros servicios son los siguientes (no utilizados en el ejemplo):

```
tunodbc200.pro7 5375/tcp       # Tun-SQL PROGRESS7
tunodbc200.ism  5376/tcp       # Tun-SQL C-ISAM
tunodbc200.mvs  5377/tcp       # Tun-SQL DB2/MVS
tunodbc200.pro8 5378/tcp       # Tun-SQL PROGRESS8
```

Base de Datos

Escriba el nombre de la base de datos con la que se quiere conectar. Para usar la base de datos de ejemplo, escribir **tunsqldemo** (es la base de datos ejemplo que se entrega con **Tun SQL**).

Login

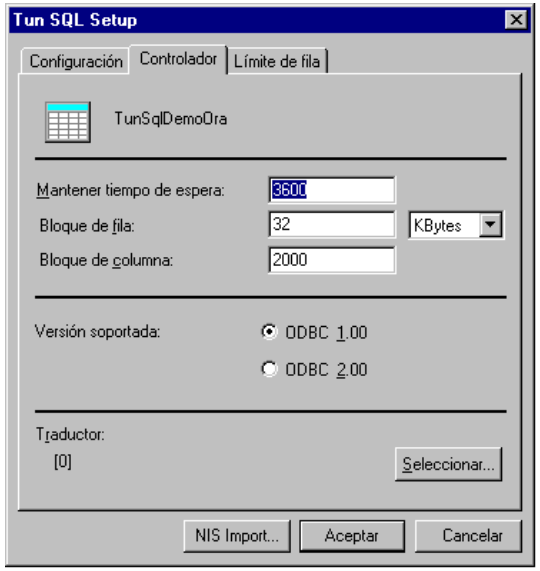
Este campo contiene el nombre de un usuario autorizado a acceder a la base de datos.

Contraseña

Escriba la contraseña asociada al usuario.

➢ Controlador

Haga clic en la solapa **Controlador** para acceder a la ventana de configuración del controlador ODBC.



Mantener tiempo de espera

Debido a que el PC es una máquina que está sujeta a frecuentes fallos de software y hardware, el servidor **Tun SQL** para UNIX tiene que comprobar regularmente que el PC sigue conectado. Para hacer esto, envía regularmente paquetes al PC. Si no le responde en un tiempo de **n** segundos el proceso se para. Este campo almacena el valor del tiempo máximo (por defecto es una hora).

Bloque de fila

Indica el tamaño de los paquetes de datos extraídos de una tabla durante una operación de selección SQL. El valor puede expresarse en kilobytes o en número de líneas.

Si el valor es 1, hay un paquete TCP por fila recuperada. Si el valor es 100, entonces la filas se agrupan en paquetes de 100 hasta alcanzar el número de filas que se recuperan. Este valor permite optimizar los intercambios en la red. El valor óptimo oscila ente 50 y 150. El valor predeterminado es de 32 KB.

Bloque de columna

Indica la unidad de fragmentación que se utilizará cuando hay que recuperar de la base de datos columnas excesivamente anchas (grandes cantidades de texto o imágenes). Si este valor es demasiado pequeño significa que el número de intercambios por la red se incrementará.

Versión soportada

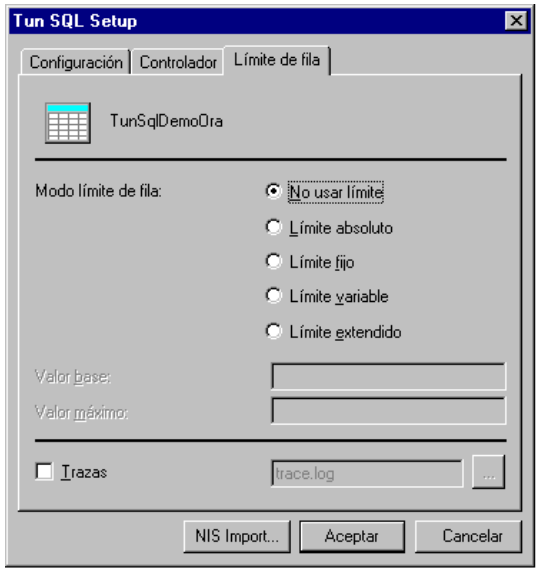
Por el momento hay dos versiones del API de ODBC, numeradas 1.00 y 2.00. Algunas aplicaciones sólo son compatibles con ODBC versión 1.00 (Access de Microsoft) y no funcionan con controladores de versiones mayores. El controlador de ODBC de **Tun SQL**, que es compatible con la versión 2.00, puede emular la versión 1.00 para que estas aplicaciones puedan ejecutarse. Marque la casilla adecuada para indicar al controlador de ODBC el nivel de compatibilidad requerido por una aplicación dada.

Traductor

Dadas las diferencias de notación con caracteres acentuados entre los entornos Windows (CP850) y UNIX (ISO8859), a veces hay es necesario establecer unas tablas de conversión de caracteres para el controlador de ODBC. El botón **Seleccionar...** permite al usuario elegir la tabla de conversión necesaria. En la base de datos de ejemplo se puede ignorar este campo ya que todo el texto que contiene está en inglés.

Nota:
Las tablas de conversión se pueden crear o editar mediante la aplicación **Tun DB Map**.

Haga clic en **Límite de fila** para acceder al cuadro de diálogo de límites:



Modo límite de fila

Algunas aplicaciones de ofimática permiten al usuario componer sus propias consultas SQL. En otros casos, ciertas aplicaciones tienden a leer los contenidos de una tabla completamente antes de visualizarla por la pantalla. Esto no supone ningún problema en particular si se trabaja con tablas pequeñas en bases de datos locales.

Por otra parte, puede haber innumerables problemas cuando se trata de tablas grandes en bases de datos centralizadas y remotas. En este caso, existe un considerable número de intercambios por la red y el PC no tiene suficiente memoria para almacenar los datos que recibe. El PC debe ser rearrancado frecuentemente después de una petición "select". Para compensar este problema, el controlador de ODBC de **Tun SQL** incorpora la noción de límites que se pueden definir a través del botón **Límites...**.

El controlador de ODBC de **Tun SQL** reconoce cinco clases de límites:

No usar límite	El controlador de ODBC no impone límites
Límite absoluto	El controlador de ODBC rechaza la carga de más de n filas durante una petición select. No se visualiza ningún mensaje que informe al usuario.
Límite fijo	El controlador de ODBC rechaza la carga de más de n filas durante una petición select. Se visualizará un mensaje que informa al usuario.
Límite variable	El controlador de ODBC rechaza la carga de más de n filas durante una petición select. Sin embargo se mostrará un mensaje proponiendo cargar más con el límite de Valor máximo.
Límite extendido	El controlador de ODBC rechaza la carga de más de n filas durante una petición select. Sin embargo se mostrará un mensaje para cargar más sin el Valor Máximo. En este caso el mensaje es un simple aviso.

Trazas

Puede seleccionar la casilla **Trazas** para guardar sus consultas SQL en un archivo "**.log**" que puede consultar más adelante. Esta opción le permite ver lo que hace el controlador ODBC cuando le emite una consulta SQL.

Haga clic en **...** para elegir el directorio en el que desea situar el archivo.

NIS

Si **Tun NIS** está instalado en el PC y el administrador de red ha configurado las tablas NIS, puede pinchar en el botón **NIS Import…** para acceder a las fuentes de datos de la red. Consulte en el manual **TCP/IP Network Services** (si no tiene **Tun NET**) cómo configurar **Tun NIS**, o el capítulo "**El visualizador NIS**" en el manual **Tun NET**.

Notas:

- En este capítulo se le pidió que creara una fuente de datos llamada **tunsqldemoXXX** que es a la que se refieren todos los ejemplos suministrados con el paquete de software **Tun SQL**.
- Si desea utilizar **Tun SQL** con otras aplicaciones y bases de datos, se debe crear una fuente de datos para cada vez. Como regla general, debe haber una Fuente de Datos particular para cada aplicación y para cada base de datos utilizada.
- Se puede crear una fuente de datos directamente utilizando la aplicación ODBC en el **Panel de Configuración**.

Transferencia de la base de datos de demo

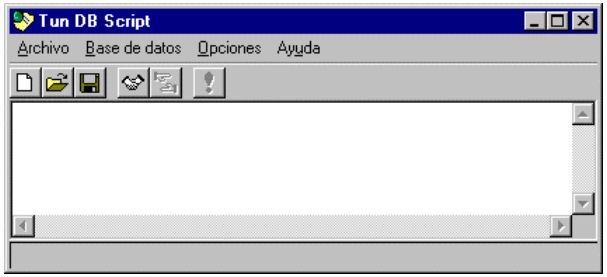
El paquete **Tun SQL** se entrega con una base de datos de ejemplo para ser utilizada con los ejemplos incluidos. Esta base de datos debe ser transferida del PC a la base de datos **tunsqldemoXXX** la cual se le pidió crear en las secciones precedentes.

Nota:
La variable XXX en el nombre de la Fuente de Datos puede tomar uno de los siguientes valores:

- **Ifx** para Informix On-Line.
- **Ise** para Informix SE.
- **Ora** para Oracle.
- **Syb** para Sybase.

Para realizar la transferencia, ejecute el programa haciendo click en el icono **Tun DB Script** del grupo **Data Access** (menú de **Inicio**, **Programas**, **Esker Tun** en Windows 95/98/2000 y Windows NT).

Cuando esta aplicación comienza a ejecutarse, aparece la siguiente ventana:



➢ Carga del archivo batch de SQL para crear la base de datos

La base de datos de su elección se puede cargar con la opción **Archivo➔Abrir...** o pinchando en el botón  . Para transferir la base de datos de ejemplo, es necesario cargar el archivo **\Demo\Db\Xxxcreat.SQL** desde el directorio de instalación de **Tun SQL**.

➢ Conexión con la fuente de datos

Antes de que pueda ejecutar el archivo batch de SQL, es necesario establecer una conexión con la Fuente de Datos. Para hacer esto pinche en el botón  o utilice la opción **Base de Datos➔Conectar...**. Esta operación muestra una caja de diálogo que le pide el nombre de la Fuente de Datos a utilizar. Si se puede establecer la conexión, se ejecuta el archivo batch.

Para transferir la base de datos de ejemplo, seleccione la Fuente de Datos **TunSqlDemoXXX** definida anteriormente.

➢ Ejecución

Para ejecutar el archivo batch de SQL, seleccione la opción **Base de Datos➔Ejecutar** del menú principal o pinche en el botón  . **Tun DB Script** enviará los comandos SQL consecutivamente a la base de datos que corresponda a la Fuente de Datos seleccionada. Si ocurre un error, la aplicación **Tun DB Script** se parará y mostrará un mensaje de error.

➢ Desconexión de la fuente de datos

Tras la ejecución, hay que desconectar la Fuente de Datos pinchando en el botón  o seleccionando la opción **Base de Datos➔Desconectar** en el menú principal. Al salir de la aplicación también se provoca la desconexión de la Fuente de Datos.

Nota:
Aunque el propósito principal de **Tun DB Script** es transferir la base de datos de ejemplo de **Tun SQL**, también tiene otros usos. En realidad, **Tun DB Script** puede ejecutar completas listas de comandos SQL para crear otras bases de datos o para actualizar o borrar completamente tablas extremadamente grandes.

Creación de una fuente de datos virtual

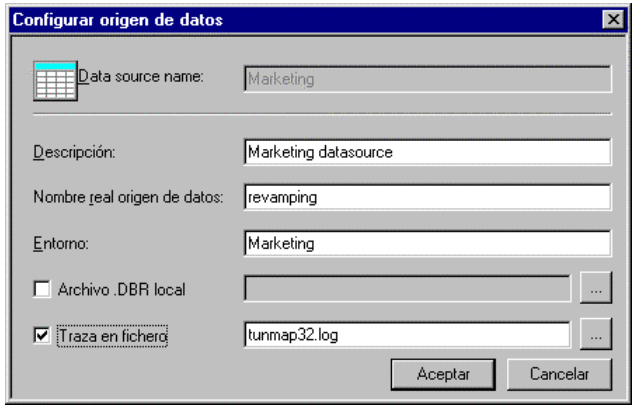
Tun DB Revamp puede vincular tablas virtuales, adaptadas a las necesidades de los usuarios finales, con una base de datos real. Para más información sobre esta aplicación, consulte la sección "**Tun DB Revamp**".

Cuando se crea una base de datos virtual (revamping), puede crear fuentes de datos asociadas a la misma, igual que si fuera una base de datos real. Los usuarios pueden usar el controlador ODBC virtual que se entrega con **Tun SQL** para acceder a la base de datos virtual que se ha creado para ellos.

Una fuente de datos virtual puede considerarse como la asociación de una fuente de datos real y un entorno.

La creación de una fuente de datos virtual se hace a través del **Administrador ODBC**, eligiendo el controlador **Tunmap32**. Consúltese "**Creación de una fuente de datos**".

Se abre este cuadro de diálogo:



Data source name

Escriba el nombre de la fuente de datos virtual.

Descripción

Este campo contiene un comentario asociado con la fuente de datos.

Nombre real origen de datos

Escriba el nombre de la fuente de datos correspondiente a la base de datos real a partir de la cual depende la bases de datos virtual.

Entorno

Escriba el nombre del entorno para el cual se quiere crear la fuente de datos virtual. Un entorno es un conjunto de tablas virtuales. Cada base de datos puede tener uno o más entornos. Pinche en el botón  para seleccionar el entorno entre todos los definidos en la base de datos.

Archivo .DBR local

En vez de escribir el nombre de una fuente de datos real, puede seleccionar el entorno a partir de un archivo local "**.dbr**". Para más información sobre el uso de este tipo de archivos, consultar la sección "**Tun DB Revamp**".

Active la casilla de verificación de **Archivo .DBR local**. Escriba la ubicación del archivo o pinche en el botón  para seleccionar el archivo. A continuación seleccione el entorno (campo **Entorno**).

Traza en fichero

A fin de guardar un registro de las peticiones SQL, active la casilla de verificación de **Traza en fichero**. Se guarda un registro de actividades que se puede consultar con un editor de texto. Con este fichero se puede comprobár qué hace el controlador ODBC cuando se le realiza una petición SQL.

Pinche en el botón  para dar un nombre al fichero de traza y para seleccionar el directorio en el que se quiere guardar.

Tablas de conversión de caracteres

Esta sección puede saltarse en la primera lectura.

➢ Diferencias en la notación utilizada por los distintos sistemas informáticos

Aunque todos los caracteres utilizados en inglés se han codificado perfectamente en la tabla ASCII (0 a 127), no ocurre lo mismo con los caracteres especiales o acentos utilizados por otros idiomas (como en francés, alemán, español, italiana, etc). A pesar de que existen normas (como ISO 8859), no siempre se implementan en los sistemas informáticos.

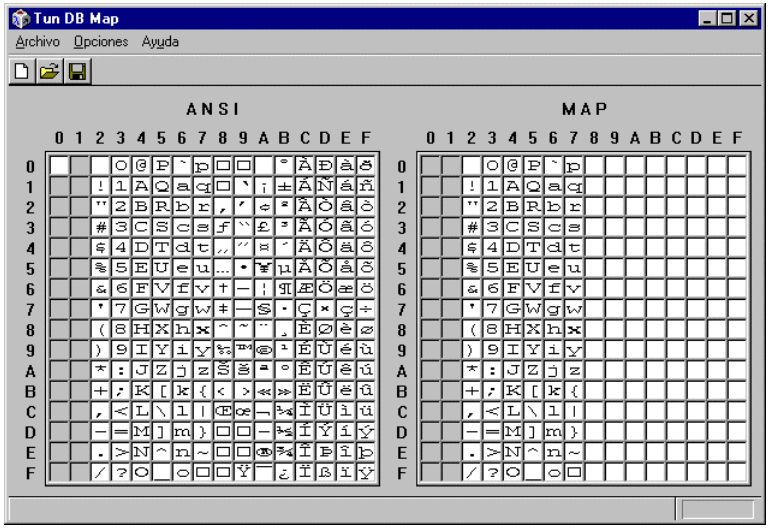
Ya que **Tun SQL** permite a un sistema informático (el PC) que acceda a los datos situados en otro sistema informático (un SGBD trabajando bajo UNIX), se debe incluir dentro de **Tun SQL** un mecanismo para controlar las diferencias de notación entre los caracteres nacionales de los diferentes sistemas. Este mecanismo permite al PC visualizar un carácter acentuado (por ejemplo una **é**) incluso si ha sido codificada de forma diferente en el SGBD bajo UNIX.

➢ Creación de tablas de conversión

El mecanismo que proporciona **Tun SQL** utiliza "tablas de conversión" que se pueden crear o modificar con la aplicación **Tun DB Map**.

Ejecute el programa haciendo click en el icono **Tun DB Map** del grupo **Data Access** (menú de **Inicio**, **Programas**, **Esker Tun** en Windows 95/98/2000 y Windows NT).

Se visualizará la siguiente pantalla:



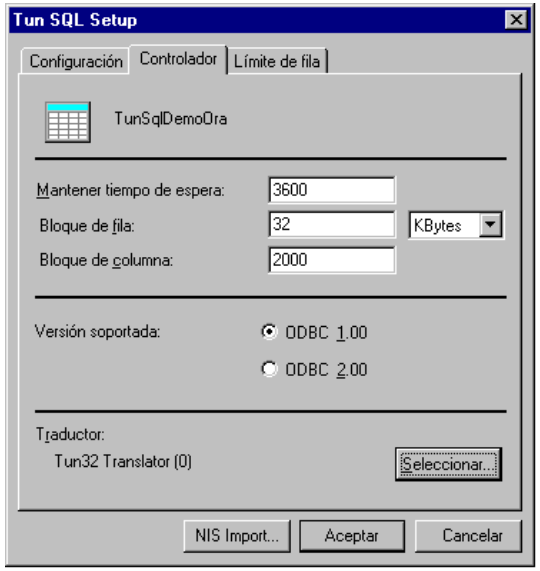
La tabla de la izquierda muestra todos los caracteres disponibles en el PC (ASCII y ASCII extendido). La tabla de la derecha debe tener los mismos caracteres pero en las posiciones que correspondan con la notación utilizada por el SGBD de la máquina UNIX. Las primeras 128 posiciones de la tabla de la derecha ya están ocupadas porque son iguales en ambos sistemas.

Para asignar un carácter a una de las posiciones de la tabla de la derecha, seleccione un carácter en la tabla de la izquierda y "arrástrelo" a uno de los cuadraditos de la tabla de la derecha manteniendo pulsado el botón del ratón.

Se pueden crear tantas tablas de conversión como sean necesarias mediante la opción **Archivo** del menú general y salvando los archivos con la extensión "**.ttt**".

➢ Implementación de las tablas de conversión

Para que se tengan en cuenta, hay que asociar las tablas de conversión con una Fuente de Datos utilizando la caja de diálogo que muestra **Tun SQL** a este efecto; introduzca los detalles en la sección **Traductor**:



o presione **Seleccionar...** para seleccionar el traductor ODBC.

3

C-ISAM

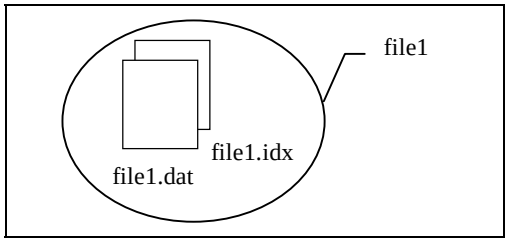
Introducción a C-ISAM

➢ El sistema de archivos C-ISAM

El C-ISAM (Método de Acceso Secuencial Indexado) es una librería de funciones C desarrollada por Informix. Permite la administración de archivos secuenciales indexados (crear e insertar archivos, operaciones de borrado y lectura). El C-ISAM incluye otras funciones como el soporte de bloqueos y transacciones que aseguren la integridad de los datos. Estas características garantizan que los datos sean accesibles, validados y se utilicen correctamente.

El C-ISAM utiliza tipos de datos similares a los que se utilizan en C. Como el C-ISAM implementa estos tipos independientemente del sistema UNIX utilizado, la forma de guardar los datos puede ser distinta a la de representación de los mismos en tiempo de ejecución. El C-ISAM incluye funciones de conversión para convertir el formato de los datos en tiempo de ejecución al formato de almacenamiento.

Un archivo C-ISAM en verdad es una combinación de dos archivos: uno que contiene los datos (file.dat) y otro con el índice para encontrar los datos en el archivo de datos (file.idx). Los dos archivos se utilizan siempre juntos como un único archivo C-ISAM.



➢ SGBDRs y C-ISAM

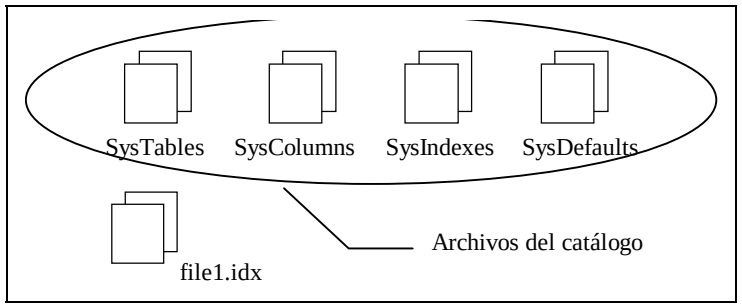
Como los archivos C-ISAM utilizan acceso secuencial indexado, los desarrolladores deben comprender la estructura del archivo y utilizar los archivos de índice para acceder a los datos.

Al construir un sistema de base de datos con archivos C-ISAM se libera a los desarrolladores de estas restricciones mediante el uso de una estructura de archivos indexados como tablas, columnas e índices en un catálogo.

➢ Tun SQL C-ISAM

El C-ISAM depende de funciones C para la consulta y actualización de archivos secuenciales. El controlador para C-ISAM que se suministra con Tun SQL permite visualizar archivos secuenciales como si fueran bases de datos relacionales estándar con tablas, campos y claves. Por tanto se pueden utilizar instrucciones SQL estándar para consultar o actualizar una base de datos creada con archivos C-ISAM. El controlador para C-ISAM traduce las instrucciones a funciones C que realizan las operaciones necesarias en los archivos secuenciales.

Para pasar de una vista de datos secuenciales a una vista de base de datos relacional, el controlador C-ISAM añade archivos descriptivos de bases de datos conocidos como **catálogo** a los archivos estándar de índice y de datos C-ISAM. Estos serán archivos de tipo C-ISAM (**.dat** archivo de datos y **.idx** archivo de índice): SysTables, SysColumns, SysIndexes y SysDefaults.



sqltools es una herramienta UNIX para crear y administrar bases de datos construidas bajo C-ISAM. Funciona de la misma forma: las instrucciones SQL se utilizan para construir la base de datos y se traducen a funciones C antes de que el sistema de archivos C-ISAM pueda leerlas. .

➢ Instalación del controlador C-ISAM

Para instalar **Tun SQL C-ISAM**, remítase a "**Instalación y configuración**".

Utilización de sqltools

➢ Utilización de sqltools

Sqltools se puede utilizar directamente o desde un entorno semi-gráfico (ventanas, menús).

Conéctese al servidor UNIX que contiene los archivos C-ISAM. Se recomienda crear un ID de usuario para acceder a los archivos C-ISAM

Sitúese en el directorio de instalación de **sqltools** y ejecute la aplicación introduciendo el siguiente comando:

`sqltools`
para ejecutar la aplicación en línea,
o
`sqtools -v`
para utilizar el entorno gráfico

➢ Creación de la base de datos

El primer paso es crear la base de datos que contenga los datos de los archivos C-ISAM. Para ello, utilice uno de los siguientes métodos:

- Elija **Database➔Create** e introduzca el nombre de la base de datos que quiere crear.

- Introduzca la siguiente instrucción en la ventana inferior (Ventana de entrada):

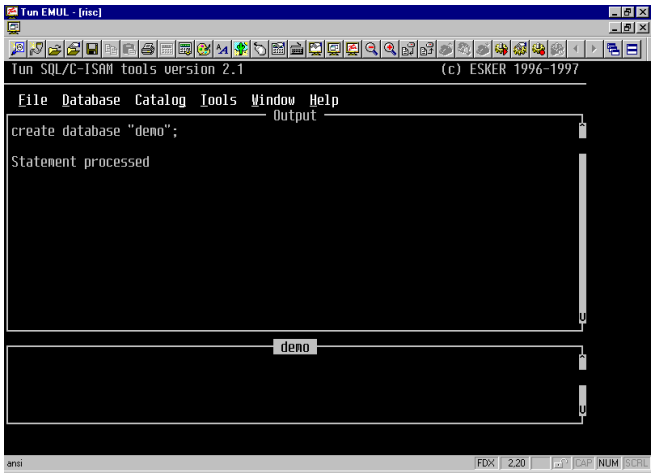
```
create database "basename";
```

Esta instrucción crea un directorio con el nombre de la base de datos y la extensión **.ism** en el directorio indicado en la variable ISAM-PATH. Vea "**Instalación del controlador C-ISAM**" en la "**Guía de Instalación Tun**" para más información acerca de la variable ISAM-PATH.

Ejemplo:
`Create database "demo";`
crea el directoriodemo.ism en el directorio ***/TunSql/bases suponiendo que ISAM-PATH=/TunSql/bases.***

Este directorio contiene los archivos C-ISAM .dat y .idx que describen las bases de datos: SysTables, SysColumns, SysIndexes y SysDefaults, un total de 4 archivos lógicos C-ISAM y 8 archivos del sistema operativo.

La ventana inferior (Ventana de entrada) tomará el nombre de la base de datos:



Nota:
Se pueden ejecutar comandos del shell directamente desde **sqltools**. Ponga el comando precedido de un signo de exclamación y seguido de un punto y coma en la ventana inferior (Ventana de entrada).
Ejemplo:
*! **ls -a** ;*

Para introducir caracteres en mayúsculas utilice comillas.
Ejemplo:
!ls "/TunSql/locisam";

➢ Conexión a una base de datos existente

Si utiliza una base de datos existente, se pueden hacer operaciones sobre las tablas tras la conexión.

Para ello, elija **Database➔Connect** e introduzca el nombre de la base de datos. La ventana inferior (Ventana de entrada) tomará el nombre de la base de datos a la que se conectó.

➢ Creación de tablas

Una vez creada la base de datos (el directorio **.ism** contiene los archivos descriptivos de la tabla), se pueden crear sus tablas. .

Se puede realizar este paso utilizando un par de archivos C-ISAM ya existentes (el archivo de datos **.dat** y el archivo de índice **.idx**), o creando unos nuevos. La instrucción que se utiliza en este segundo caso es diferente, así como las precauciones a tener en cuenta: Utilice la instrucción **create table** para crear una pareja de archivos C-ISAM a la vez que se crea la tabla y la instrucción **define table** para crear una tabla a partir de una pareja de archivos C-ISAM existentes.

Creación de tablas partiendo de cero

Para crear la pareja de archivos C-ISAM a la vez que la tabla, introduzca la siguiente instrucción en la ventana inferior (Ventana de entrada), que ahora tiene el nombre de la base de datos como título:

```
create table tablename (field1 type1, field2
type2,..., primary key(field1)));
```

Esta instrucción crea una tabla con el nombre "tablename" en la base de datos. La tabla contiene los campos field1, field2, etc. con los tipos type1, type2, etc. Ver lista de tipos.

El par de archivos C-ISAM de datos y de índice se crean en el directorio de la base de datos (**basename.ism**). Los nombres de estos archivos utilizan los siete primeros caracteres del nombre de la tabla y un número de identificación único que se asigna automáticamente: Al archivo de datos se le añade la extensión **.dat** y al de índice la extensión **.idx**. Si el nombre de la tabla es menor de siete caracteres, estos nombres de archivos se rellenarán con caracteres de subrayado ("_").

Ejemplo:

```
create table table1 (field1 longint, field2 char(25),
filler char (30), primary key(field1));
```

*crea los archivos **tabla1_100.dat** y **tabla1_100.idx**. La tabla contiene el campo **field1** del tipo **longint**, el campo **field2** con un máximo de 25 **char** (caracteres), y el campo **filler** con un máximo de 30 **char** (caracteres). La clave primaria para esta tabla es **field1**.*

Creación de tablas partiendo de la pareja de archivos C-ISAM

Para crear una tabla en una base de datos donde la pareja de archivos C-ISAM **.dat** y **.idx** ya existen, introduzca la siguiente instrucción en la ventana inferior (Ventana de entrada) que tiene de título el nombre de la base de datos

```
define table tablename file is filename (field1
type1, field2 type2,..., primary key(field1)));
```

Esta instrucción crea una tabla con el nombre "nombredetabla" a partir de loa archivos **filename.dat** y **filename.idx**.

Nota importante:
Antes de utilizar la tabla (por ejemplo, antes de utilizar instrucciones **select** en la tabla), deberá copiar los archivos especificados por la palabra clave **file is** en el directorio de la base de datos.
La instrucción **define**, puede ser ejecutada, sin embargo, aunque no se hayan copiado todavía los archivos a este directorio. Se recomienda **no** copiar los archivos hasta que se ejecute la instrucción **create index**, si se desea crear un índice para la tabla.

➢ Creación de un índice

Para crear un índice de una a ocho columnas en una tabla, introduzca la siguiente instrucción en la ventana inferior (Ventana de entrada), que tiene de título el nombre de la base de datos:

```
create unique index indexname on tablename (field1,
field3);
```

Esta instrucción crea el índice "indexname" para las columnas **field1** y **field3** en la tabla "tablename".

➢ Estructuras C

Cada instrucción que se pasa a **sqltools** se traduce a código C para el sistema de archivos C-ISAM. Para ver la estructura en C correspondiente para una tabla, elíjase **Catalog➔GetCStruct**.

Por ejemplo, la instrucción que crea **table1** produce la siguiente estructura en C:

```
struct root_table1              /* file "table1_100" */
 {long          lint_field1;    /* field1 longint */
 char           chr_field2[25]; /* field2 char(25) */
 char           chr_filler[30]; /* filler char(30) */
 unsigned char  null_flags[1];  /* reserved */
};
```

En este ejemplo, hay un campo reservado (un array de **unsigned char**s) de un byte de longitud. Este campo lo utiliza el sistema de administración de C-ISAM. Cuando se crea una tabla con la instrucción **create table**, **sqltools** añade automáticamente este campo reservado a la tabla.

➢ Validación de una tabla creada a partir de archivos C-ISAM

Cuando se crea una tabla a partir de una pareja de archivos C-ISAM (con la instrucción **define**), debe tener cuidado para que la estructura de la tabla cuadre con los archivos C-ISAM.

Por ejemplo, usted crea la tabla "table2" basada en los archivos C-ISAM **filename.dat** y **filename.idx**, escritos en C. Estos archivos tienen la siguiente estructura:

- Un campo compuesto por una variable longint,
- Un campo compuesto por un array de 25 variables char,
- Un campo compuesto por un array de 30 variables char.

Si define la tabla "table2" así:

```
define table table2 file is table1_100 (field1
longint, field2 char(25), filler char(25));
```

creará una tabla cuyos registros tendrán la estructura:

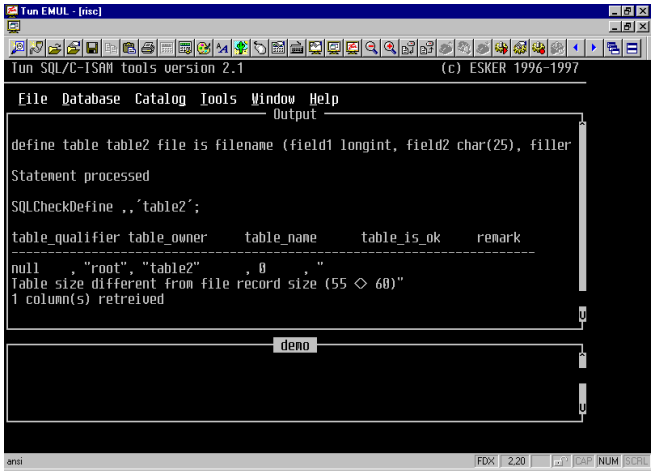
- Un campo de tipo longint,
- Dos campos array de caracteres de longitud 25,

los cuales no corresponden a la estructura generada por los archivos C-ISAM en los que están basados la tabla.

En este caso, existe una discordancia entre la tabla creada y los archivos C-ISAM en los que se basa.

Para verificar que la estructura de los datos de la tabla corresponde a los archivos C-ISAM originales, elija **Catalog➔CheckDefine**. Para verificar todas las tablas de la base de datos, elija **Tools➔Check Catalog**.

En nuestro ejemplo, la ventana de salida mostrará los siguientes resultados:



Se mostrará el siguiente mensaje:

```
Table size different from file record size (55 <>
60)"
```

El mensaje avisa de que la tabla **table2** se definió diferente a los archivos C-ISAM "filename" en los que está basada. La instrucción **define** debería ser así:

```
define table table2 file is filename (field1 longint,
field2 char(25), filler char(30));
```

➢ Ver la estructura C de la tabla

Se puede visualizar la estructura C de una tabla, esto es, las diferentes columnas creadas (nombre, tipo de datos y longitud), así como la información referente a la administración de los datos (claves primarias, por ejemplo).

Para ello, elija **Catalog➔GetCStruct** e introduzca el nombre de la tabla de la que quiere ver su estructura en C. Para ver la estructura C de todas las tablas de la base de datos, elija **Tools➔List Structures**.

El siguiente ejemplo muestra el resultado de este comando para una tabla creada con un clave primaria:

La siguiente instrucción crea la tabla:

```
create table customer
(cust_number longint,
 cust_name char(20),
 cust_address1 char(20),
 cust_address2 char(20),
 filler char(20),
 primary key (cust_number)
);
```

La estructura C que se genera es:

```
struct doc_customer            /* file "custome110" */
{long    lint_cust_number;     /* cust_number longint */
 char    chr_cust_name[20];    /* cust_name char(20) */
 char    chr_cust_address1[20];/* cust_address1 char(20) */
 char    chr_cust_address2[20];/* cust_address2 char(20) */
 char    chr_filler[20];       /* filler char(20) */
 unsigned char null_flags[1];  /* reserved */
};

struct keydesc idx_customer_1;
idx_customer_1.k_flags = ISNODUPS;
idx_customer_1.k_nparts = 1;
idx_customer_1.k_part[0].kp_start = 0;
idx_customer_1.k_part[0].kp_leng  = 4;
idx_customer_1.k_part[0].kp_type  = LONGTYPE;
```

La primera parte del código muestra la estructura de la tabla (los campos), la segunda muestra la clave primaria y sus asignaciones.

➢ Información del catálogo

Con el menú **Catalog** se obtiene información del catálogo. Las opciones de menú proporcionan información acerca de:

- **TypeInfo**: Cada tipo de datos se identifica con un número, tal y como define el estándar ODBC. Introduzca el número del tipo de datos que quiere verificar o introduzca un 0 para ver la lista de tipos de datos.
- **Tables**: Se puede consultar el catálogo para obtener información de sus tablas, utilizando el nombre del usuario que creó la tabla, el propio nombre de la tabla, o el tipo de tabla (tabla de sistema, sinónimo, etc.). Introduzca el carácter % para incluir todos los tipos de tablas en la consulta.
- **Columns**: Se puede consultar el catálogo para obtener información acerca de sus columnas, utilizando el nombre del usuario que creó la tabla, el propio nombre de la tabla, o el nombre de la columna. Introduzca el carácter % para incluir todas las tablas o columnas en la consulta.
- **Statistics**: Se pueden obtener estadísticas de los datos del catálogo.
- **PrimaryKeys**: Se puede consultar el catálogo por las claves primarias de la tabla, utilizando el nombre del usuario que creó la tabla o el propio nombre de la tabla. Introduzca el carácter % para incluir todas las tablas en la consulta.

Para más información acerca de las opciones del menú ODBC.

➢ Edición de una tabla

La longitud de los registros de una tabla se establece cuando esta se crea. Por lo tanto, no se pueden añadir o borrar registros si se afecta a la longitud total.

Si se desea cambiar la estructura de una tabla, primero hay que borrar la tabla tomando las precauciones descritas en "Borrado de una tabla".

➢ Borrado de una tabla

Hay dos formas de borrar una tabla del catálogo:

Si creó la tabla con la instrucción **create table**: Utilice la instrucción **drop table** para borrarla. **Nota**: Esta instrucción elimina todas las referencias a la tabla de los archivos del catálogo y borra la pareja de archivos C-ISAM asociados a la tabla.

Si definió la tabla con la instrucción **define table**: Utilice la instrucción **undefine table** para anular la instrucción **define**. Esta instrucción sólo elimina todas las referencias a la tabla de los archivos del catálogo.

Nota:
Se puede utilizar la instrucción **drop table** para una tabla definida con **define table**. Sin embargo, la instrucción **drop table** borrará la pareja de archivos C-ISAM especificados en la palabra clave **file is** si están en el directorio de la base de datos. Por tanto, debería tener cuidado al utilizar la instrucción **drop table** si generó la tabla a partir de archivos C-ISAM.

Si desea mantener la pareja de archivos C-ISAM asociados a la tabla que va a borrar con la instrucción **drop table**, primero tiene que copiar estos archivos a un directorio distinto. De esta forma, tras borrar la tabla podrá utilizar estas copias de seguridad.

➢ Mantenimiento de los archivos C-ISAM

Al crear tablas a partir de archivos C-ISAM existentes (con **define table**), deberá copiar estos archivos en el directorio de la base de datos para poder utilizar las tablas.

No obstante, si estos archivos se utilizan y actualizan con otras aplicaciones, es útil poder utilizar las actualizaciones en la base de datos C-ISAM. Para ello, se deben crear enlaces simbólicos con los archivos existentes C-ISAM desde la base de datos, mejor que hacer copias.

Ejemplo:
*Se crea la base de datos **dbtest** en el directorio **/TunSql**. Para las tablas de esta base de datos, se utilizan los archivos **filename.dat** y **filename.idx**, que están en el directorio **/data** y que son utilizados por otras aplicaciones.*

*Se crean enlaces simbólicos a estos archivos en el directorio **/TunSql/dbtest.ism** (el directorio de la base de datos) con el siguiente comando:*

```
ln -s /data/filename.* /TunSql/dbtest.ism
```

➢ Borrado de una base de datos

Para eliminar (borrar) una base de datos, elija **Database➔Drop** e introduzca el nombre de la base de datos a eliminar, o introduzca el siguiente comando en la ventana inferior (Ventana de entrada):

```
drop database databasename
```

➢ Guardar los resultados

Los resultados que aparecen en la ventana superior (ventana de salida) se pueden guardar en un archivo de texto con extensión **.res**.

Para ello, elija **File➔Save as...** y seleccione el lugar y el nombre del archivo a guardar.

➢ Ejecución de un programa

Con la opción **File➔Execute** se ejecutan programas SQL que utilicen instrucciones que soporte **sqltools**.

PARTE SEGUNDA
BASES DE DATOS VIRTUALES
(REVAMPING)

4

REVAMPING

Bases de datos virtuales

La mayoría del almacenamiento de datos estructurado actual está formado por Sistemas de gestión de bases de datos relacionales (SGBDR). Las bases de datos permiten almacenar los recursos de datos de la corporación, que pueden actualizarse utilizando aplicaciones especiales. El volumen de datos así recopilados interesa a grandes números de usuarios que desean extraer de estos almacenes la información que precisan para su trabajo (indicadores de eficacia, estadísticas, sistemas expertos). Para actualizar y consultar las bases de datos se utiliza el lenguaje SQL.

Sin embargo, la estructura de las bases de datos, que son el núcleo del sistema de información, puede complicar el acceso a la información que contienen a todos los niveles de la corporación:

- Hay un número considerable de tablas y registros en las bases de datos, demasiados para el usuario medio que frecuentemente está interesado sólo en parte de los datos.
- La estructura de base de datos siempre es compleja y requiere muchos conocimientos para recorrerla.
- El entorno informático de bases de datos no está orientado al usuario. Por ejemplo, los nombres de las tablas y registros rara vez se expresan en términos comprensibles.
- La manipulación y el uso de datos precisan conocimientos de lenguaje SQL para la consulta de la base de datos y obtener el resultado deseado.

Hay varios avances que reducen estos obstáculos y facilitan el acceso a bases de datos (p.ej.: la inclusión de interfaces gráficas en herramientas de interrogación de bases de datos).

El siguiente paso está dirigido a liberar por completo al usuario final del prerrequisito de conocimientos técnicos sobre bases de datos poniendo a su disposición sólo los datos que precise en la forma más apropiada para su entorno de trabajo.

Las consecuencias de este cambio son:

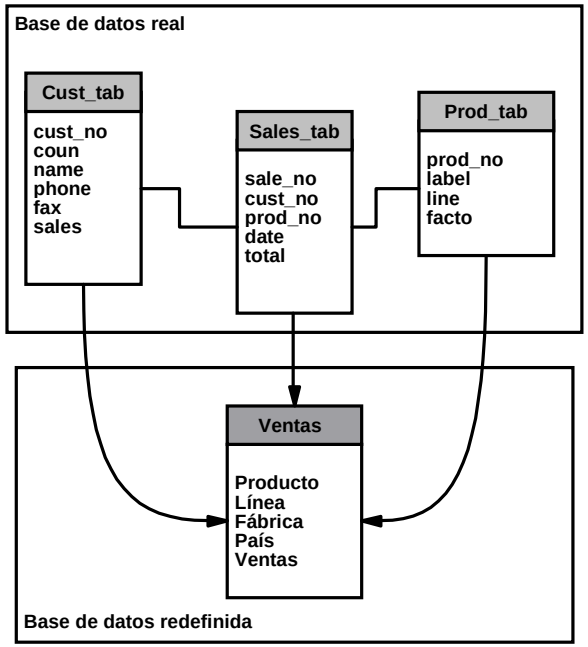
- Mejora de la productividad: el usuario final se autonomiza en el uso de datos, el análisis y la toma de decisiones son más rápidos porque se han facilitado.
- Información más relevante: Con sólo los datos que necesita y puede dominar, el usuario mejora su capacidad de análisis y síntesis y refina sus resultados.

➢ Revamping

El principio de revamping es la construcción de una base de datos virtual adaptada al entorno del usuario de la base de datos existente. Aunque no existe como base de datos real, el usuario ve la nueva estructura como una base de datos normal cuyas tablas y campos corresponden exactamente con sus necesidades: la base de datos sólo contiene la información que el usuario necesita de verdad para sus análisis en una forma apropiada para sus necesidades (nombres de datos y funciones predefinidas comprensibles).

La base de datos redefinida la perfecciona un administrador que reconfigura las tablas y campos de bases de datos reales.

Por ejemplo:



En el ejemplo anterior, la base de datos real contiene tres tablas: "Cust_tab" (tabla de clientes), "Sales_tab" (tabla de ventas) y "Prod_tab" (tabla de productos).

El administrador define una tabla virtual que presenta el resultado de ventas por producto, por línea de producto, por centro de producción y por país.

La tabla virtual "Ventas" contiene los campos siguientes:

- Producto (campo real: "prod_tab.label»).
- Línea (campo real: "prod_tab.line").
- Fábrica (campo real: "prod_tab.fact").
- País (campo real: "cust_tab.coun").
- Ventas (campo real: "sales_tab.total").

La tabla virtual construye una unión entre las tablas "Prod_tab" y "Sales_tab" mediante el campo común "prod_no", y una unión entre las tablas "Sales_tab" y "Cust_tab" mediante el campo común "cust_no".

Revamping en Tun SQL

Tun SQL puede realizar la administración y uso de bases de datos virtuales gracias a los dos componentes siguientes:

- El administrador de bases de datos **Tun DB Revamp**, que administra el revamping.
- El controlador virtual ODBC, que permite al usuario acceder a bases de datos puestas al día por el administrador.

➢ El administrador Tun DB Revamp

El objetivo de la base de datos virtual de **Tun SQL** es ofrecer al usuario final información redefinida en contacto para un "entorno" en particular.

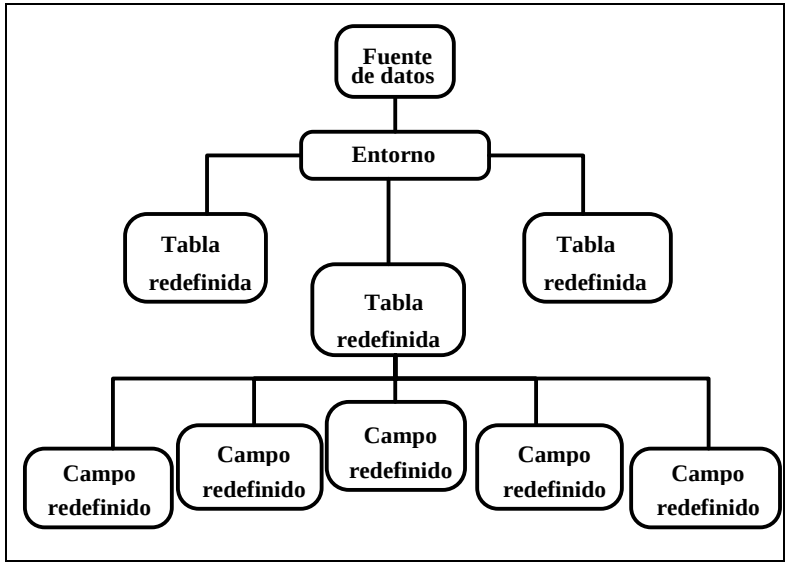
Gracias a una interfaz gráfica intuitiva, el administrador puede definir cuantos "entornos" desee para distintos usuarios o tipos de usuarios. El entorno se basa en "ocupación": un contable podría ver sólo las tablas relacionadas con contabilidad, el personal de ventas sólo verá las tablas relacionadas con su actividad principal.

Cada entorno puede verse con el componente frontal ODBC que realiza la consulta como fuente de datos especial (consulte el diagrama de arquitectura en la sección "**Controlador virtual ODBC**").

El modelo de base de datos virtual es como sigue:

- Uno o más entorno definidos a partir de una fuente de datos real que contienen una selección de tablas particulares en la base de datos real basadas en las necesidades del usuario.
- Las tablas definidas en un entorno son nativas de la base de datos real o el resultado de uniones entre una o más tablas (noción de vista).
- Cada tabla contiene sólo los campos que necesita el usuario final.
- Los campos puestos al día son campos existentes de tablas reales o campos recalculados que simplifican el uso de la base de datos.

- Las tablas y los campos puestos al día pueden renombrarse de forma más clara para el usuario final (por ejemplo: "Cust_tab" se convertiría en "Tabla de clientes" y "Cust_no" en "Cliente número").



Las tablas redefinidas en un entorno no existen físicamente en la base de datos. Sin embargo, la base de datos redefinida se almacena de forma indexada en tres tablas suplementarias creadas en la base de datos:

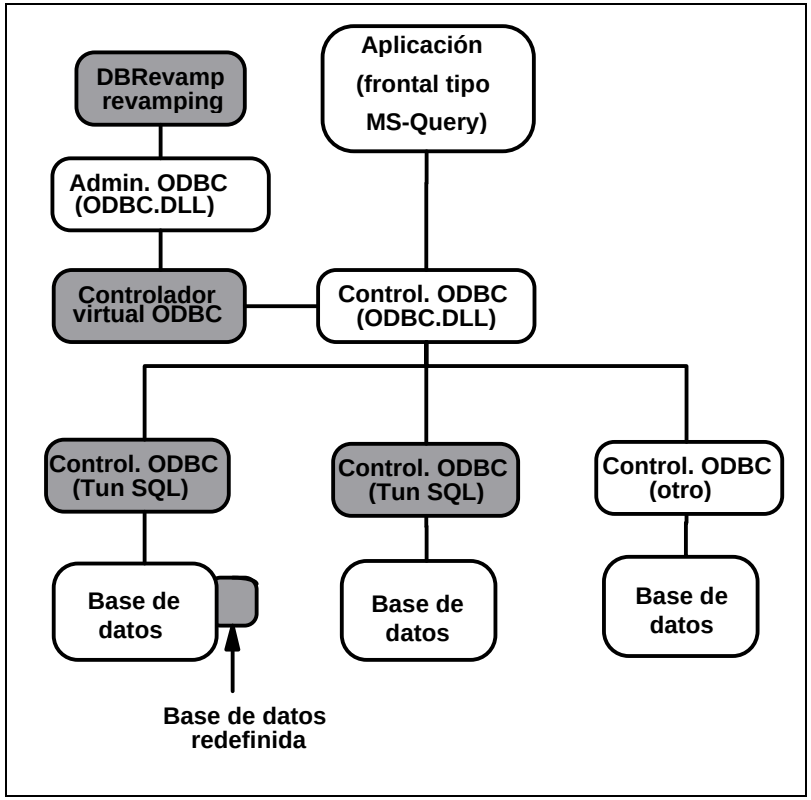
- La tabla de entorno con lista de entornos en la forma de "nombre de entorno" y "descripción".
- Una tabla con la lista de tablas redefinidas, cada una de las cuales se indexa por nombre, una descripción del nombre y el entorno al que pertenece.
- Una tabla de campos redefinidos, cada uno de los cuales se indexa por nombre, una descripción, un origen (campo existente, datos procesados, concatenación de datos) y el nombre de la tabla virtual a la que pertenece.

Tras una operación de revamping, la base de datos siempre contendrá estas tres tablas adicionales.

➢ Controlador virtual ODBC

Para el usuario final, la consulta de una base de datos virtual es similar a la de una real en modo real. Esta transparencia se debe a la integración en **Tun SQL** de un controlador ODBC especial para bases de datos virtuales.

Cuando el administrador de ODBC (**ODBC.DLL**) recibe consultas de un entorno específico, las transmite al controlador virtual ODBC que a su vez las traduce en consultas apropiadas para la base de datos real. A continuación, el controlador virtual ODBC devuelve las consultas traducidas al administrador ODBC que las dirige al controlador ODBC normal de la base de datos real.



5

UTILIZACIÓN GENERAL DE TUN DB REVAMP

En este capítulo se da una descripción de los principales comandos **de Tun DB Revamp**.

Opciones generales

➢ Selección del idioma de trabajo

Para elegir el idioma de interfaz que desea utilizar, haga clic en la opción **?➔ Idioma"** y seleccione el idioma en el que desea trabajar.

➢ Modificación de la presentación

Para modificar los controles que aparecen en la ventana principal de DB Revamp:

- Seleccione o cancele la opción **Ver➔Barra herramientas** del menú principal para ver o no la barra de herramientas.
- Seleccione o cancele la opción **Ver➔Barra estado** del menú principal para ver o no la barra de estado.
- Seleccione o cancele la opción **Ver➔Barra propiedades** del menú principal para ver o no la barra de propiedades del objeto.

➢ Copia de un objeto

Utilice uno de estos métodos para copiar un objeto:

1. Para utilizar el método de "**pinchar y arrastrar**", seleccione el objeto que desea copiar y arrastre el ratón hasta el lugar donde desea copiarlo con el botón del ratón presionado.

2. Utilice la opción del menú principal **Editar➔Copiar** para copiar el objeto seleccionado, y después la opción **Editar➔Pegar** para pegarlo en el lugar apropiado.
3. Utilice las opciones **Copiar** y **Pegar** del menú contextual (que aparece al presionar el botón derecho del ratón) para copiar el objeto seleccionado y pegarlo en la ubicación apropiada.
4. Utilice las teclas **Ctrl-C** (copiar y **Ctrl-V** (pegar) para realizar la operación.
5. Utilice los botones de la barra de herramientas,  (copiar)  (pegar), para realizar la operación.

➢ Eliminación de un objeto

Para eliminar un objeto, selecciónelo haciendo clic y después:

1. Utilice la opción **Editar➔Eliminar** del menú principal.
2. Utilice la opción **Eliminar** del menú contextual.
3. Utilice una de las teclas **Supr** del teclado.
4. Haga clic en  en la barra de herramientas.

➢ Cambio de nombre de un objeto

Para cambiar el nombre de un objeto, primero selecciónelo, y después:

1. Utilice la solapa **General** de la barra de propiedades.
2. Presione la tecla **F2** y sustituya el nombre anterior por el nuevo.
3. Vuelva a hacer clic en el objeto y continúe con el método 2.

➢ Almacenamiento de cambios

Para almacenar los cambios realizados a valores de propiedades, pulse **Entrar** con el cursor en el cuadro de diálogo correspondiente o utilice el botón **Aplicar**.

➢ Obtención de ayuda

Para acceder a la ayuda en línea u obtener más información sobre DB Revamp, haga clic en la opción **? ➔Acerca de DB Revamp...** del menú principal o utilice el botón  de la barra de herramientas.

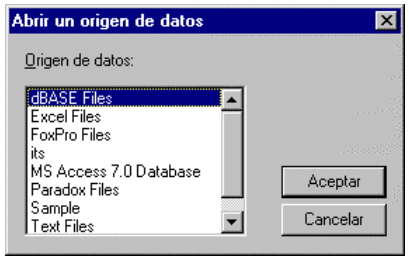
➢ Salida de Tun DB Revamp

Para salir de la aplicación, haga clic en **Archivo➔Salir**.

Importación de entornos de fuentes de datos

Para redefinir una base de datos real, ha de seleccionar la fuente de datos correspondiente. Para ello, utilice la opción **Archivo➔Importar...** o haga clic en el botón  de la barra de herramientas.

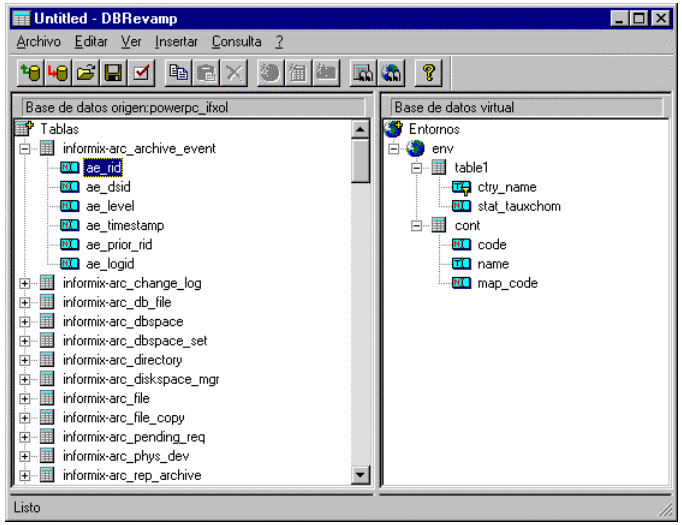
Se abre este cuadro de diálogo:



Muestra la lista de fuentes de datos declaradas en el PC. Para crear una fuente de datos, consulte "**Creación de una fuente de datos**". Como las fuentes de datos virtuales, obtenidas poniendo al día una base de datos real, no pueden redefinirse, no aparecen en esta lista. Sin embargo, sí que aparecen en la lista de fuentes de datos disponibles para el usuario final en cualquier aplicación de Windows que los utilice (p.ej. Microsoft Query).

Seleccione la fuente de datos que desee utilizar.

Aparece una ventana de **Tun DB Revamp** similar a la siguiente:



Las tablas de la base de datos real aparecen en la sección izquierda de la ventana.

Si la base de datos en cuestión no se ha redefinido con **Tun DB Revamp**, la sección derecha de la ventana contendrá un entorno vacío llamado "Nuevo entorno" que es el primer entorno que puede configurar.

Por otro lado, si la base de datos ya se ha redefinido con **Tun DB Revamp** (en cuyo caso se trata de actualizar la base de datos virtual), la sección derecha contendrá la lista de entornos ya creados y sus contenidos.

Creación de un entorno

Para definir un nuevo entorno para la base de datos, seleccione la raíz (denominada "Entornos") y seleccione en el menú principal **Insertar➔Nuevo entorno**. También puede pinchar en el botón  en la barra de herramientas.

Introduzca el nombre y opcionalmente una descripción para este entorno.

Creación de una tabla virtual

Para crear una tabla virtual en un entorno, seleccione el entorno y haga lo siguiente:

- Seleccione en el menú principal **Insertar➔Nueva tabla** o seleccione **Nueva tabla** en el menú contextual del entorno. También puede pinchar en el botón  en la barra de herramientas.
- Seleccione **Ver➔Cuadro de propiedades** para ver el cuadro de propiedades de la tabla que se acaba de crear (en caso de que no se estuviese mostrando).
- En la solapa **General, Cuadro de propiedades**, escriba un nombre y una descripción opcional de la tabla. Puede utilizar también la tecla de función **F2** para cambiar el nombre de una tabla seleccionada.

Si quiere que la tabla virtual contenga toda o parte de la tabla real, puede copiar la tabla real en el entorno de su elección: Todos los campos de la tabla real también se copiarán. Para hacer esto:

- Utilice uno de los métodos descritos en la introducción (**pinchar y arrastrar**, **copiar/pegar**, abreviatura de teclado o botón de la barra de herramientas) para seleccionar la tabla real en la base de datos fuente y copiarla al entorno destino.
- Elimine los campos que no necesite de la base de datos virtual o modifíquelos como se describe en la sección "**Creación de campos**".
- Si lo desea, puede cambiar los nombres de los objetos copiados (tablas y campos), y añadirles una descripción en la correspondiente solapa **General**.

Creación de campos

En una tabla virtual puede:

- Insertar un campo que ya existe en la base de datos real, sin cambiar su definición.
- Crear nuevos campos virtuales a partir de campos reales de la base de datos.

➢ Campos existentes

Puede copiar un campo existente de una tabla de la base de datos real directamente a la virtual, para ello:

- Use uno de los métodos descritos en la introducción (**pinchar y arrastrar**, **Copiar/Pegar**, atajo de teclado y botón de barra de herramientas) para seleccionar un campo de la tabla real de la base de datos fuente y copiarla a la tabla virtual de la base de datos redefinida.
- Si quiere, puede cambiar el nombre del campo y asignarle una descripción en la solapa **General** correspondiente (o presionar **F2** para renombrarlo).

➢ Campo nuevo

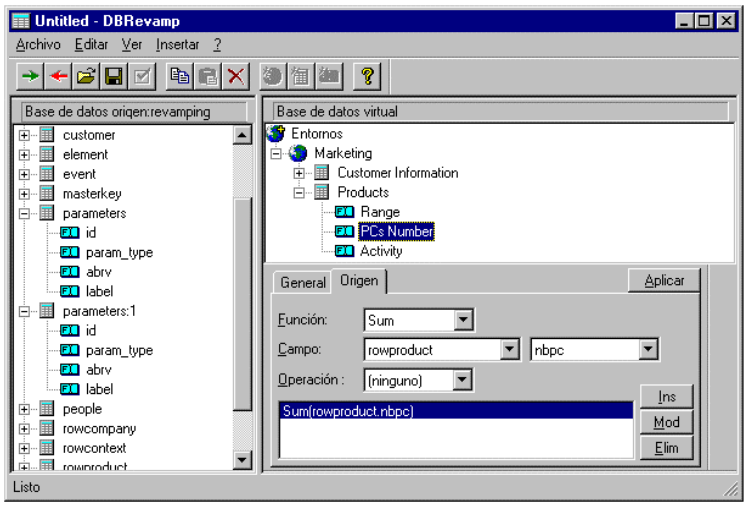
Para definir un nuevo campo en una tabla virtual, seleccione la tabla virtual y haga lo siguiente::

- Seleccione en el menú principal **Insertar➔Nuevo campo** o seleccione **Nuevo campo** en el menú contextual de la tabla. También puede pinchar en el botón  de la barra de herramientas.
- Seleccione **Ver➔Cuadro de propiedades** para ver el cuadro de propiedades del campo que se acaba de crear (en caso de que no se estuviese mostrando).
- Escriba un nombre y, opcionalmente, una descripción en la solapa **General**.

Haga clic en la solapa **Origen**. Ahora puede:

- Añadir una función al campo: seleccione la función que desee en la lista **Función**. Las funciones disponibles son: **Sum**, **Min**, **Max**, **Numero**, **Media**, o **Ninguno**.
- Añadir un valor de un campo existente de una base de datos real al campo nuevo o a la función antes seleccionada: seleccione la tabla real y el campo que desea en las dos listas **Campo**.
- Añadir una operación al campo antes seleccionado: elija el operador que desea emplear en la lista **Operación**. Las operaciones disponibles son: +, - , * , / , o ninguna. El operador + puede emplearse para concatenar caracteres.

- Tras ello, haga clic en **Ins** para añadir estas opciones a la definición de campo.



Ejemplo 1:
Tiene acceso a la tabla real "res_tab" que contiene cuatro campos res1, res2, res3 y res4 correspondientes a los resultados trimestrales de un año particular. Quiere definir el campo "Resultado" en una tabla de su base de datos virtual que contiene la suma de los cuatro campos reales.

*En la solapa **Origen,** del campo "Resultado":*

- *Seleccione la tabla "res_tab" y el campo "res1" en las listas **Campo**.*
- *Seleccione el operador **+** en la lista **Operación**.*
- *Haga clic en **Mod** para sustituir la entrada predeterminada. La nueva entrada es "res_tab.res1 +".*
- *Seleccione la tabla "res_tab" y el campo "res2" en las listas **Campo**.*
- *Seleccione de nuevo el operador **+** en la lista **Operación**.*
- *Haga clic en **Ins** para añadir la entrada recién creada "res_tab.res2 +".*
- *Realice el mismo procedimiento para "res3".*
- *Para el campo "res4", seleccione el operador **Ninguno** en lugar de **+**.*
- *El campo "Resultado" se define finalmente como sigue:*

  res_tab.res1+
  res_tab.res2+
  res_tab.res3+
  res_ btab.res4

  *lo que significa que el campo **Resultado** contiene la suma de los cuatro campos "res1"," res2", "res3" y "res4".*

Ejemplo 2:
Quiere que el campo "Resultado" muestre la suma de las ventas anuales. Para ello, hay que sumar todos los campos res1, res2,… y a continuación sumar estos cuatro resultados.

*En la solapa **Origen,** del campo "Resultado":*

- *Seleccione la función **Sum** en **Función**.*
- *Seleccione la tabla res_table y el campo res1 en las listas de opciones de **Campo.***
- *En **Operación**, seleccione el operador +.*
- *Pinche en el botón **Mod** para sustituir la entrada por defecto. La nueva entrada es "res_tab.res1 +".*
- *Haga lo mismo para los campos "res2" y "res3". Para el campo "res4",en **Operación** seleccione el operador "ninguno", en vez de "+".*

Puede cambiar un elemento en la definición de un campo con el botón **Mod** (el elemento resaltado se sustituye por los valores antes seleccionados). Haga clic en **Elim** para borrar el elemento seleccionado.

Tras definir el campo, haga clic en **Aplicar** para que las nuevas opciones tengan efecto.

En cuanto se haya definido un nuevo campo virtual, no olvide definir las uniones entre la(s) tabla(s) utilizadas para crear la tabla virtual, si las hay. Consulte la sección "**Vínculos entre tablas**".

Para comprobar que el cálculo asignado al campo creado hace lo que usted quiere, utilice la función de consulta de **Tun DB Revamp**. Ver "**Consulta de bases de datos reales y virtuales**".

Asignación de filtros a los campos

Se puede completar la definición de un campo virtual con un filtro, esto es, se puede definir una condición para calcular el valor del campo. El filtro corresponde a condiciones limitadas de consulta (como en **MS Query**).

Ejemplo:
Usted desea obtener la suma de los campos "res1" cuando el campo "res2" es mayor que un número dado. La condición dada en "res2" es un filtro.

Tun DB Revamp le permite asignar un filtro a campos virtuales que se utilizarán cuando el usuario final utilice el campo. Un filtro puede ser:

- Estático: El valor del filtro es fijo.
- Dinámico: Los usuarios introducen sus propios valores cuando realizan la consulta.

Para asignar un filtro a un campo virtual, seleccione el campo y haga clic en la solapa **Filtro** del **Cuadro de Propiedades**. A continuación:

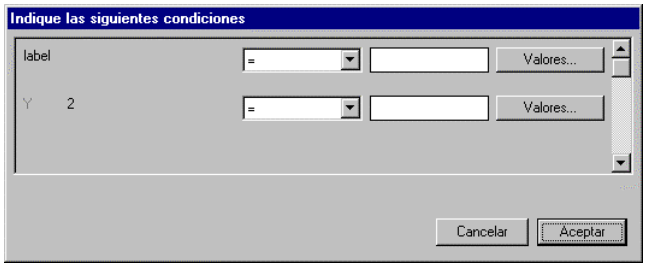
- Introduzca una etiqueta para el filtro en el campo **Etiqueta**. La etiqueta es opcional para filtros estáticos. Para filtros dinámicos, la etiqueta debe indicar el propósito del filtro de cara al pedirle el valor al usuario.

- Seleccione la tabla y el campo al que se le aplicará el filtro en la lista desplegable **Campo**.
- Seleccione el operador de comparación de la lista desplegable **Comp**.
- Para filtros estáticos, introduzca el valor del filtro en el campo **Valor**. Para filtros dinámicos, introduzca una interrogación (**?**).
- Haga clic en **Ins** para introducir el criterio definido.

Se puede crear un conjunto de condiciones o criterios: Defina los criterios como se ha indicado anteriormente. Seleccione **Y** o **O** para añadir más criterios.

Para comprobar que el filtro que ha asignado al campo creado es exactamente lo que usted quiere, utilice la función de consulta de **Tun DB Revamp**. Ver "**Consulta de bases de datos reales y virtuales**".

Si el filtro es dinámico, al consultar el campo virtual aparece una ventana como esta:



Introduzca el valor solicitado para aplicar el filtro al campo virtual. Haga clic en el botón **Valores...** para que se muestre la lista de posibles valores para el campo en cuestión.

Vínculos entre tablas

Los campos definidos en una tabla virtual se obtienen de una o más tablas de la base de datos real.

Para cada tabla virtual, es imprescindible definir los vínculos entre las tablas reales de las que se extraen sus componentes. La definición de estos vínculos permite crear las uniones entre las tablas reales cuando el usuario final consulta la base de datos virtual. Estos vínculos pueden ser directos o indirectos (p.ej. vínculos entre una tabla y otra o entre más de una tabla y otras tablas).

➢ Definición de vínculos

La forma más sencilla es definir los vínculos al mismo tiempo que los campos de la tabla virtual. Todas las tablas reales utilizadas para definir campos han de vincularse directa o indirectamente a las otras tablas reales utilizadas por la tabla virtual.

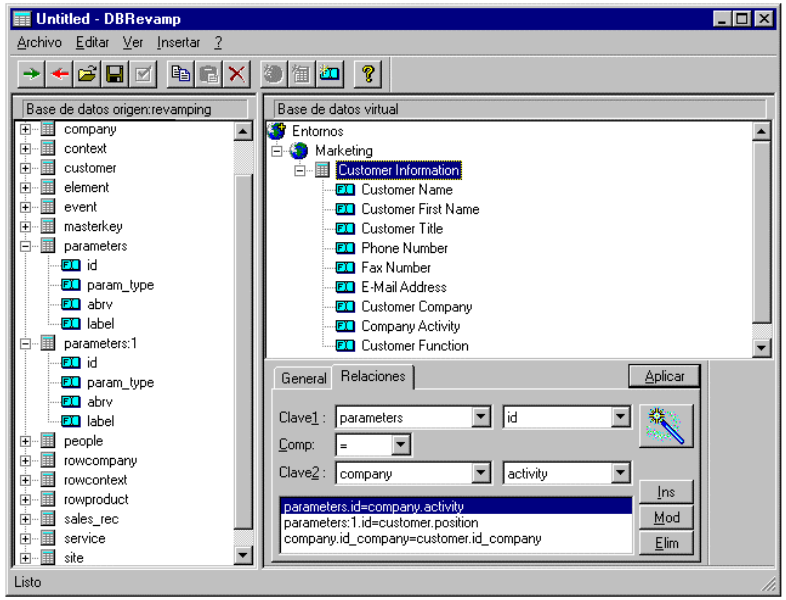
Para definir relaciones entre tablas reales en una tabla virtual, seleccione la tabla virtual y haga lo siguiente:

- Pinche en la ficha **Relaciones** de la tabla virtual.
- Para cada una de las tablas, seleccione el nombre de la tabla real y el campo que sirve de enlace con la otra tabla. Para ello, use los cuadros de lista **Clave1** para la primera tabla real y **Clave2** para la segunda tabla real.

Nota:
Los nombres de los dos campos que vinculan las tablas pueden ser distintos, aunque representen los mismos datos.

- Seleccione un operador de comparación en la lista **Comp**.

- Haga clic en **Ins** para añadir el vínculo a la lista de vínculos de la tabla virtual.



Puede cambiar un vínculo con el botón **Mod** tras cambiar los valores del elemento seleccionado. Haga clic en **Elim** para eliminar el elemento seleccionado.

Haga clic en **Aplicar** para que se valide la lista de vínculos definida.

➢ Comprobación de vínculos

Tun DB Revamp dispone de una función que verifica los vínculos definidos por el administrador entre las tablas reales utilizadas para crear la virtual.

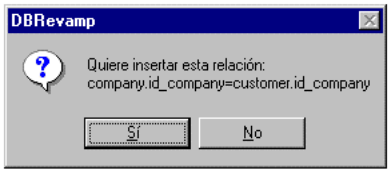
Puede comprobar fácilmente en cada tabla virtual si las tablas reales utilizadas están vinculadas y si los vínculos definidos forman un todo coherente.

Para ello, haga clic en el botón  de la solapa **Relaciones**.

Tun DB Revamp examina todos los vínculos definidos por el administrador y detecta los posibles vínculos directos o indirectos ausentes que aíslan tablas específicas del resto.

Cuando falta un vínculo entre dos tablas, **Tun DB Revamp** intenta vincularlos utilizando dos campos del mismo nombre.

Si existen los dos campos, **Tun DB Revamp** propone definir el vínculo de este modo:



En la mayoría de los casos, el vínculo propuesto será el correcto. Sin embargo, si cree que los dos campos propuestos por **Tun DB Revamp** no deben formar el vínculo entre las tablas, defina el vínculo manualmente como se describe en la sección "**Definición de vínculos**".

Por otro lado, si dos tablas no vinculadas no tienen campos del mismo nombre, **Tun DB Revamp** muestra una lista de las tablas no vinculadas:

En este caso, defina el vínculo manualmente como se explica en "**Definición de vínculos**".

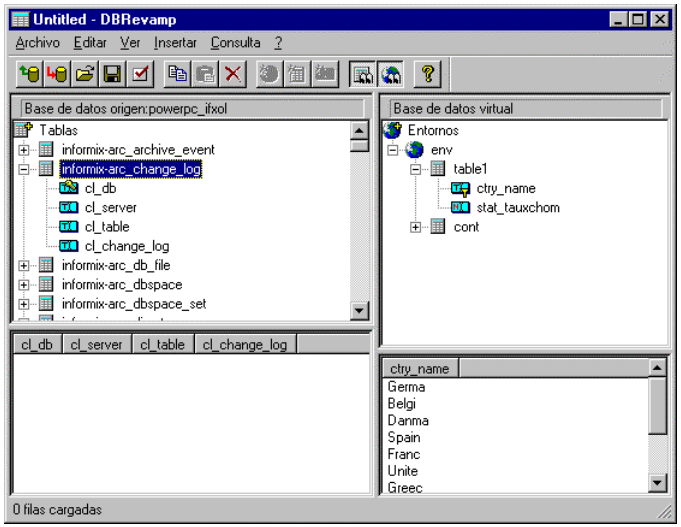
Consulta de bases de datos reales y virtuales

Tun DB Revamp incluye una función de consulta para tablas y campos de bases de datos reales y virtuales.

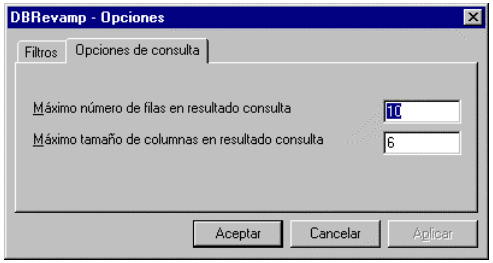
Se puede utilizar esta función para ver una tabla o un campo de la base de datos real o virtual directamente desde **Tun DB Revamp** sin tener que utilizar una herramienta de consulta como **MS Query**.

Para consultar una tabla (o un campo) de una base de datos real, elija **Consulta➔Origen** (o **Consulta➔Entorno**) en el menú principal, o haga clic en el botón  de la barra de herramientas.

Se abrirá una ventana bajo la ventana de la base de datos real o virtual dependiendo de la opción elegida. Se pueden elegir ambas opciones al mismo tiempo.



Para limitar el número de registros mostrados así como el ancho de la columna al consultar una tabla o un campo, elija **Ver➔Opciones...** en el menú principal y haga clic en la solapa **Opciones de Consulta**:



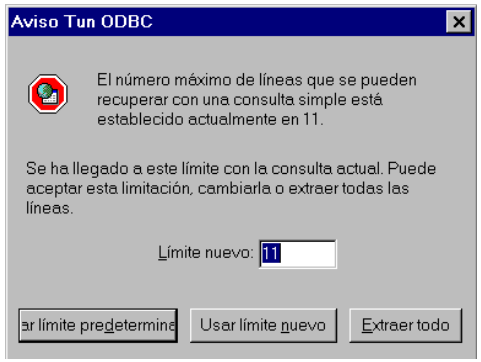
Introduzca el número máximo de registros que se visualizarán en **Máximo número de filas en resultado consulta**.

Introduzca el máximo ancho de columna que se visualizará en **Máximo tamaño de columnas en resultado consulta**.

Nota:
Si los límites fueron definidos una vez creada la fuente de datos (ver "**Creación de fuentes de datos**"), estos tendrán una prioridad más alta que las **Opciones de Consulta**.

Ejemplo:
Supongamos que al configurar la fuente de datos real se estableció un límite variable de 11 a 15 líneas.

Si se consulta una base de datos real que contenga más de 11 registros, aparecerá un mensaje de aviso como este:



El mensaje de aviso exacto dependerá del tipo de límite utilizado.

Validación de un entorno

Tun DB Revamp dispone de una función para verificar la coherencia entre el contenido de los entornos creados por el administrador y el contenido de la base de datos real utilizada. Esta función resulta especialmente útil cuando se realizan cambios en la estructura de la base de datos real (p.ej. se modifica o elimina un campo), que el administrador no ha tenido en cuenta en la base de datos virtual.

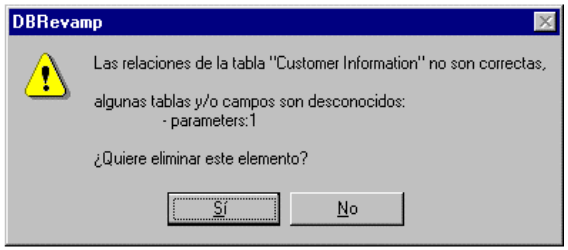
Para utilizar esta función, hay que validar el entorno antes de exportarlo y así evitar incoherencias. Utilice la opción **Archivo➔Validar entornos** del menú principal o haga clic en el botón  de la barra de herramientas.

Ejemplo:
Tome como ejemplo el caso en que la tabla "parameters:1" y sus campos se han copiado en el entorno "Marketing". Cada campo de la tabla virtual tiene por tanto la tabla "parameters:1" como punto de origen.

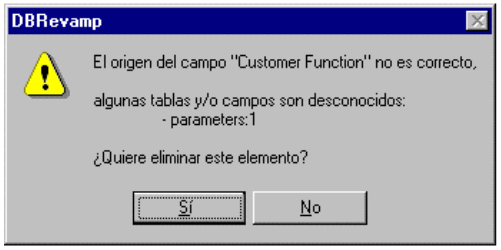
Si la tabla lógica se elimina en la base de datos real, los campos de la base de datos real, los campos de la tabla virtual no tendrán ya punto de origen. De forma similar, todas las tablas con referencia a la tabla "parameters:1" en sus relaciones serán incoherentes.

Cuando se solicita una validación de entorno, **Tun DB Revamp**:

- indica las tablas virtuales en que una o más relaciones afectan a la tabla eliminada y propone borrarlas. En este caso, es preferible no borrarlas, sino eliminar los campos cuyo punto de origen es la tabla eliminada.



- indica los campos virtuales cuyos orígenes están en la tabla borrada y propone eliminarlos.



Si no se detectan incoherencias, aparece la ventana siguiente:

Exportación de entornos de fuentes de datos

Para que la base de datos redefinida por sus entornos esté disponible para los usuarios, se debe exportar del PC al servidor.

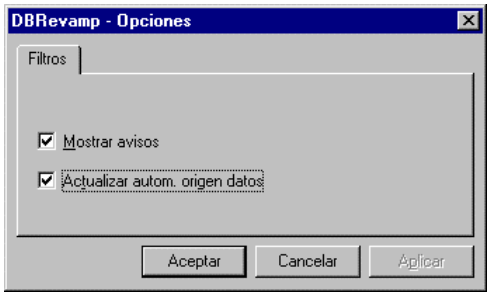
Para exportar entornos de fuentes de datos desde el PC al servidor, elegir **Archivo➔Exportar...** en el menú principal o haga clic en el botón  de la barra de herramientas.

Esta operación creará o actualizará las tres tablas que contienen la información de la base de datos redefinida. Ver el capítulo "**Redefinición en Tun SQL**".

Actualización de una fuente de datos virtual

Cuando cambia el nombre de un entorno, puede actualizar la fuente de datos virtual correspondiente. Esta función es automática si activa la casilla **Actualizar autom. origen datos** en **Ver➔Opciones**.

Se abrirá la siguiente ventana:



Si se selecciona esta opción, hay dos posibilidades:

- si se selecciona la casilla **Mostrar avisos**, **Tun DB Revamp** pedirá confirmación siempre que vaya a realizarse la actualización automática.
- si no se activa esta casilla, la actualización se efectúa automáticamente sin confirmación.

Por otro lado, si no se activa la opción **Actualizar autom. origen datos**, utilice la opción **Insertar➔Crear/Actualizar origen de datos** del menú principal o la opción **Actualizar origen de datos** del menú contextual del entorno correspondiente para actualizar la fuente de datos oportuna.

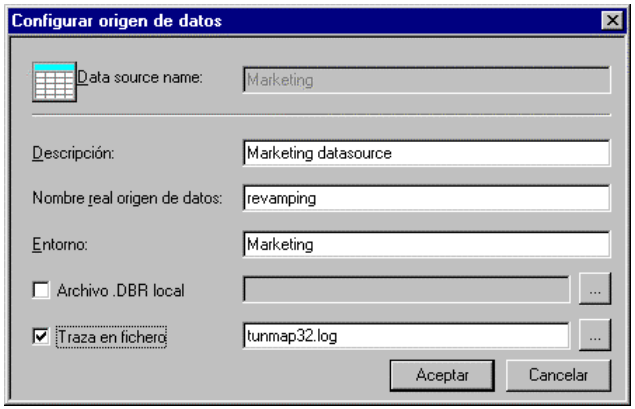
Si quiere cortar temporalmente el vínculo entre un entorno y su fuente de datos, utilice la opción **Insertar➔Eliminar origen de datos** o la opción **Eliminar origen de datos** del menú contextual. El vínculo puede volverse a crear posteriormente con **Actualizar origen de datos**.

Creación de una fuente de datos virtual

El capítulo "**Configuración y Uso**" describe cómo crear una fuente de datos virtual con el **Administrador ODBC**.

Esto se puede también hacer con **Tun DB Revamp**. Seleccione **Crear origen de datos asociado...** en el menú contextual del entorno.

Se abre este cuadro de diálogo:

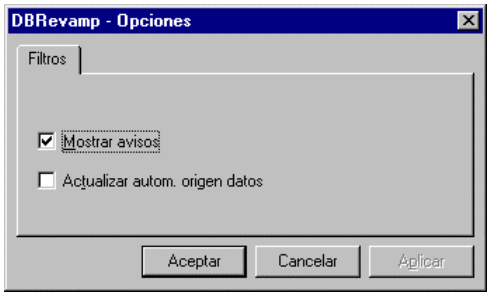


Escriba una descripción de la fuente de datos en el campo **Descripción**. Active la casilla de verificación de **Traza en fichero** e indique el nombre de fichero en el que se guardará un registro de actividades para la fuente de datos virtual. Para más información sobre este cuadro de diálogo, consúltese la sección "**Creación de una fuente de datos virtual**" en el capítulo "**Configuración y Uso**".

Mostrar avisos

Cuando el administrador utiliza **Tun DB Revamp** pueden presentársele diversas advertencias. Estas advertencias ofrecen información, por ejemplo, sobre incoherencias que pueden darse durante la redefinición de la base de datos o sobre elementos ausentes en la nueva estructura.

Tun DB Revamp muestra estas advertencias de forma predeterminada. Sin embargo, puede cancelar la presentación de cuadros de mensaje dejando sin marcar la casilla Mostrar avisos (**Ver➔Opciones**).



Administración de fuentes de datos redefinidas locales

Puede almacenar y abrir localmente la descripción de la base de datos redefinida. Esto puede resultar útil si no quiere exportar inmediatamente la base de datos redefinida, o si desea conservar versiones anteriores. La descripción se almacena en un archivo con extensión "**.dbr**".

➢ Almacenamiento local

Para almacenar localmente la descripción de una base de datos redefinida creada con DB Revamp, utilice la opción **Archivo➔Guardar** (o **Archivo➔Guardar como...** para almacenarla con otro nombre), o haga clic en el botón de la barra de herramientas.

También se almacena la vía de acceso a la fuente de datos real, lo que significa que la fuente de datos y su entorno pueden exportarse más adelante sin necesidad de indicar su destino.

➢ Abrir una fuente de datos local

Para abrir una fuente de datos redefinida que ha sido guardada en un archivo .dbr en la máquina local, seleccione **Archivo➔Abrir** en el menú principal o pulse el botón en la barra de herramientas.

Seleccione la fuente de datos redefinida que desea (un archivo "**.dbr**")

Puede abrir las fuentes de datos usadas más recientemente a través del menú **Archivo**.

➢ Nueva carga de la estructura de la base de datos

Al abrir un "**.dbr**" guardado localmente, podrá actualizar la estructura de la base de datos desde donde se creó el entorno. Esto es útil cuando la base de datos real se ha cambiado desde la última vez que se guardó el archivo **.dbr**. Para hacerlo, elija **Archivo➔Nueva carga de la estructura de la B.D.** en el menú principal.

Después de esta operación, se recomienda validar los entornos creados previamente, especialmente si los campos reales utilizados para definir los campos virtuales se han movido o eliminado. Ver "**Validación de un entorno**" para más detalles.

Identificación de campos

➢ Iconos de campos

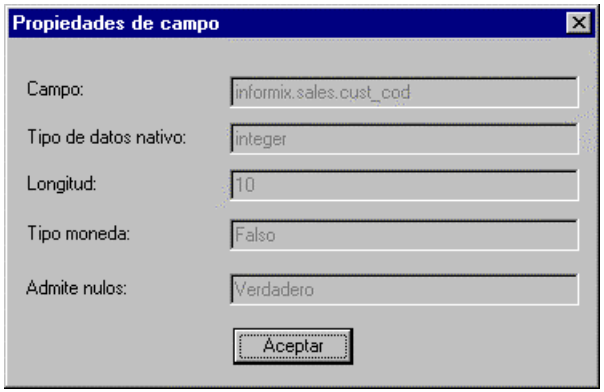
Tun DB Revamp utiliza iconos de campos para hacer más fácil el leer las tablas reales o virtuales.

Iconos	Significado
	campo de texto
	campo de fecha
	campo numérico
	campo binario
	clave de tabla primaria

➢ Propiedades de los campos reales

Se puede seleccionar **Propiedades** en el menú de contexto de los campos de tablas reales (ventana izquierda) para obtener información adicional del campo.

La ventana **Propiedades del campo** se abrirá:



Nota:
El tipo de campo del que se habla corresponde al tipo nativo y por tanto depende del SGBD utilizado.

PARTE TERCERA
APÉNDICES

REFERENCIA

Índice

Nota:
xxx equivale a la extensión de archivo relativa de una base de datos específica. Las extensiones de archivo son las siguientes:

- **ifx** Informix
- **ora** Oracle
- **syb** Sybase
- **db2** DB2
- **pro** Progress
- **pro7** Progress7
- **pro8** Progress8
- **ism** C-ISAM
- **mvs** DB2 / MVS

config.XXX	Archivo que contiene los parámetros de funcionamiento y de seguridad del servidor **Tun SQL** para UNIX
DBMAP	Aplicación Windows para crear o editar tablas de conversión de caracteres.
DBSCRIPT	Aplicación Windows que interpreta y ejecuta archivos batch de SQL
DBSHOW	Aplicación Windows de comprobación y configuración
param.XXX	Archivo que contiene los parámetros de arranque del servidor **Tun SQL** para UNIX
tunodbc200.XXX	Servidor **Tun SQL** para UNIX

Archivo que contiene los parámetros de funcionamiento y de seguridad del servidor **Tun SQL** para UNIX.

➢ Descripción

Los archivos **config.XXX** pueden proporcionar un cierto número de parámetros al servidor **Tun SQL** para UNIX. Al contrario que los archivos **param.XXX**, los parámetros no conciernen al total funcionamiento del servidor pero se refieren a una base de datos en particular. Por ejemplo. un archivo **config** tiene este aspecto:

```
#Optional declaration for databases
#Example :
#[base_name]
#Define=ENV_VARIABLE:value
#RowLimitMode=None|Absolute|Fixed|Variable|Extended|1
|2|3|4|5
#RowLimitValue=value
#RowLimitMax=value
#DbmsName=DatabaseName
#Version=DatabaseVersion

# In this section, list allowed configuration
(base,user,product)
# Base_Name|*,User_Name|*,Product_Name|*
[Allowed]

# In this section, list denied configuration
(base,user,product)
# Base_Name|*,User_Name|*,Product_Name|*
[Denied]
```

Este archivo puede contener tantas secciones como bases de datos administradas por el SGBD haya asociadas con el servidor **Tun SQL** para UNIX. No es necesario incluir todas las bases de datos si no se quiere definir algún parámetro en especial.

Cada sección tiene el nombre de la correspondiente base de datos entre corchetes a modo de título (por ejemplo **[tunsqldemo]**). Se pueden definir los siguientes parámetros en cada sección:

Define=END_VARIABLE:value

Asigna el valor **valor** a una variable de entorno **ENV_VARIABLE** antes de abrir la base de datos (directorio de instalación de la base de datos, fecha...). Esta opción se puede utilizar tantas veces como sea necesario en el archivo **config**.

RowLimitMode=None|Absolute|Fixed|Variable|Extended|1|2|3|4|5
RowLimitValue=value
RowLimitMax=value

Algunas aplicaciones comerciales dejan al usuario la libertad de componer sus propias peticiones SQL. En otros casos, las aplicaciones tienden a leer una tabla completamente antes de mostrarla en pantalla. Esto no supone ningún problema en particular si se trabaja con tablas pequeñas en bases de datos locales. Por otra parte, puede haber innumerables problemas cuando se trata de tablas grandes en bases de datos centralizadas y remotas. En este caso, existe un considerable número de intercambios por la red y el PC no tiene suficiente memoria para almacenar los datos que recibe. El PC debe ser rearrancado frecuentemente después de una petición "select". Para compensar este problema, el controlador de ODBC de **Tun SQL** incorpora la noción de límites que se pueden definir a través del parámetro **RowLimitValue**. Si se define este parámetro tiene prioridad respecto a los valores que se pudieron definir en la Fuente de Datos del PC.

El parámetro puede tener cinco diferentes valores:

None	El controlador de ODBC no impone límites
Absolute	El controlador de ODBC rechaza la carga de más de **RowLimitValue** filas durante una petición **select**. No se visualiza ningún mensaje que informe al usuario.
Fixed	El controlador de ODBC rechaza la carga de más de **RowLimitValue** filas durante una petición **select**. Se visualizará un mensaje que informa al usuario.
Variable	El controlador de ODBC rechaza la carga de más de **RowLimitValue** filas durante una petición **select**. Sin embargo se mostrará un mensaje proponiendo cargar más con el límite del valor máximo (**RowLimitMax**).
Extended	El controlador de ODBC rechaza la carga de más de **RowLimitValue** filas durante una petición **select**. Sin embargo se mostrará un mensaje para cargar más sin el Valor Máximo. En este caso el mensaje es un simple aviso.

DbmsName=DatabaseName

Este parámetro es para definir o cambiar el nombre de la base de datos a transmitir al controlador de ODBC.

Version=DatabaseVersion

Este parámetro es para definir o cambiar el número de versión de la base de datos a transmitir al controlador de ODBC.

Además de las secciones correspondientes a cada base de datos, el archivo **config** puede tener las secciones **[Allowed]** y **[Denied]** que se pueden utilizar para accesos de seguridad a ciertas bases de datos. Estas secciones funcionan así:

[Allowed]

Esta sección debe contener series de términos triples como:

```
base_name,user_name,product_name
```

donde

- **base_name** es el nombre de la base de datos
- **user_name** es el nombre de un usuario de la base de datos
- **product_name** es el nombre de una aplicación Windows que utilizará el servidor.

Cada serie de tres indica si un usuario (**user_name**) está autorizado a utilizar la base de datos (**base_name**) con la aplicación Windows (**product_name**). Cada parámetro se puede reemplazar por el carácter genérico *****.

Por ejemplo, la serie **tunsqldemp,*, excel** significa que todos los usuarios pueden utilizar la base de datos **tunsqldemo** desde la aplicación **excel** excepto aquellos nombres que estén en la sección similar a esta, llamada **[Denied]**.

[Denied]

Esta sección debe contener series de términos triples del mismo tipo que las de la sección **[Allowed]**.

Cada serie de tres indica si un usuario (**user_name**) no está autorizado a utilizar la base de datos (**base_name**) con la aplicación Windows (**product_name**). Cada parámetro se puede reemplazar por el carácter genérico *****.

Por ejemplo, la serie ***,juan,excel** significa que el usuario **juan** no puede utilizar ninguna base de datos desde la aplicación **excel** excepto para aquellos nombres que estén contenidos en una serie similar y contradictoria de la sección **[Allowed]**.

Nota:
Para que un servidor **Tun SQL** para UNIX tenga en cuenta el archivo **config**, se le debe indicar al servidor utilizando la opción de la línea de comandos **-c**.

Pueden coexistir dos motores de base de datos Informix distintos (Informix versión 5 e Informix versión 7). Por ejemplo, puede accederse a una de ellas mediante el motor 1 desde el directorio /u/informix1 y a la otra mediante el motor 2 desde el directorio /u/informix2. En este caso, el archivo config.ifx contendrá lo siguiente:

```
[database1]
Define=INFORMIXDIR:/u/informix1
Define=DBPATH:/u/database1
Version=5.01
[database2]
Define=INFORMIXDIR:/u/informix2
Define=DBPATH:/u/database2
Version=7.01
```

➢ Ver también

param.XXX, tunodbc200.XXX

DBMAP

Aplicación Windows para crear o editar tablas de conversión

➢ Sintaxis

```
DBMAP [-fnombre_archivo]
```

➢ Descripción

Tun DB Map se puede utilizar para crear o editar tablas de traducción de caracteres. Las tablas de traducción evitan los problemas causados por las diferencias en la codificación de caracteres acentuados que pueden ocurrir entre el PC y el SGBD remoto. Para que se tengan en cuenta, las tablas de traducción deben ser declaradas en la definición de la correspondiente Fuente de Datos.

-fnombre_archivo

Se utiliza para indicar el nombre de una tabla de traducción existente.

DBSCRIPT

Aplicación Windows para ejecutar archivos batch de SQL.

➢ Sintaxis

```
DBSCRIPT  [-dfuente_datos][nombre_archivo]
```

➢ Descripción

Tun DB Script permite ejecutar una lista completa de peticiones SQL en una única operación. Interpreta loas comandos SQL uno tras otro y se para cuando encuentra un error. **Tun DB Script** también se puede utilizar para modificar y salvar archivos batch de SQL.

Tun DB Script es útil para transferir el contenido de una base de datos de un PC que corre bajo Windows a un SGBD remoto. También se puede utilizar para actualizaciones a gran escala o borrado de bases de datos.

nombre_archivo

Se utiliza para indicar el nombre del archivo que contiene los comandos SQL que se cargarán cuando la aplicación arranque.

-dfuente_datos

Se utiliza para indicar el nombre de la Fuente de Datos con la que trabajará el archivo batch de SQL. **Tun DB Script** no crea automáticamente la conexión con la Fuente de Datos. Por tanto hay que hacerlo "a mano".

DBSHOW

Aplicación Windows de comprobación y configuración.

➢ Sintaxis

```
DBSHOW [-hnombre_servidor]
```

➢ Descripción

Esta aplicación puede ser utilizada para consultar en un servidor UNIX de la red si hay presente uno o más servidores **Tun SQL** para UNIX.

Introduzca el nombre del servidor en el campo **Nombre del host**, a continuación pinche en el botón **Consultar host** para obtener esta información. Si hay correctamente instalados uno o más servidores **Tun SQL** para UNIX en la máquina remota, **Tun DB Show** devolverá sus nombres y los nombres de los SGBD con los que se relacionan.

Esta aplicación es particularmente útil para comprobar que la instalación es correcta.

-hnombre_servidor

Indica el nombre del servidor que se quiere consultar.

Archivo que contiene los parámetros de arranque del servidor **Tun SQL** para UNIX.

➢ Descripción

En vez de ejecutar los servidores **Tun SQL** para UNIX con un gran número de opciones en la línea de comando, es preferible escribir todas las opciones una tras otra en un archivo y dar el nombre del archivo al servidor a continuación de la opción **-f**.

El procedimiento de instalación de **Tun SQL** utiliza este mecanismo escribiendo las opciones en archivos **param.XXX** donde el carácter ***** es el nombre abreviado del SGBD (**param.ora**, **param.syb**, **param.ifx**).

He aquí un ejemplo de este tipo de archivos:

```
-output=/dev/null
-output2=/dev/null
-DORACLE_HOME=/home3/oracle/7.1.4
-DORACLE_SID=odbc
-config=/usr/tunsql/config.ora
```

El significado de las diferentes opciones se explica en la sección **tunnodbc.***.

➢ Progress

Algunos campos de Progress pueden contener varios valores: estos campos se conocen como “array fields”. Para ver estos valores desde aplicaciones como MS Query y MS Access, la siguiente opción debe aparecer en el fichero param.pro.X (donde X es el número de la versión de Progress que se utiliza).

```
-arrayfields=*
```

donde * se debe sustituir por uno de los siguientes caracteres:

$, &, #, %, - ,_.

El carácter por defecto es _.

Las columnas requeridas para ver los valores de los diferentes campos array se denominaran:

columnane*n*

donde * es el carácter escogido en el fichero param.proX (por defecto "_") y n la posición del valor en la tabla..

Ejemplo:
El segundo valor en el campo array aparece en la columna col_2_.

Si utilizar el carácter "_" causase problemas al generar la columna (otras columnas podrían tener el mismo nombre), deberá escoger uno de los otros cuatro caracteres.

➢ Ver también

config.XXX, tunodbc200.XXX

TUNODBC200.XXX

Servidor **Tun SQL** para UNIX.

> **Nota:**
> Los distintos servidores UNIX de **Tun SQL** tienen varias opciones en común. La lista de opciones distintas puede obtenerse utilizando el ejecutable **tunodbc200.xxx** con la opción **-a[ll]**, donde **xxx** representa la extensión relativa de la base de datos en cuestión.

➢ Sintaxis

```
tunodbc200.XXX
-a[ll]
-c[config]=config_file
-Dname=value
 -db[ms]=DBMS_name
 -de[bug]
 -f[ile]=param_file
 -h[old]
 -i[nter]
 -l[owercase]
 -n[opassword]
 -nor[owcount]
 -o[utput]=file_name
 -o[utput]2=file_name
 -ow[ner]
 -p[rogress]=XX
 -s[electby]
 -sv[archar]
 -sy[scolumns]
 -t[imer]=xx
 -u=user1,user2...
 -v[ersion]=DBMS_version_number
 -x=user1,user2...
```

➢ Descripción

-a

Muestra todas las opciones que soporta **tunodbc200.XXX.**

-c=config_file

Asocia un archivo de configuración (**config.XXX**) con el servidor (Ver config.XXX).

-db=name

Se utiliza para asociar un nombre de SGBD con el servidor **Tun SQL** para UNIX. Este es el valor que se mostrará en la lista al ejecutar **Tun DB Show**.

-de

Indica al servidor que funcione en modo de traza. Los mensajes se visualizarán por defecto en el dispositivo **/dev/console**.

-Dname=value

Establece la variable de entorno "**name**" con el valor "**value**" antes de ejecutar el servidor (directorio de instalación de la base de datos, formato de la fecha...). Esta opción se puede repetir en la línea de comandos tantas veces como sea necesaria. Es absolutamente indispensable definir algunas variables para que el servidor **Tun SQL** para UNIX pueda funcionar con ciertos SGBD. Esta definición se lleva a cabo por el procedimiento de instalación. Con el propósito de tener una referencia, se necesitan las siguientes variables para los siguientes SGBD:

Oracle:

ORACLE_HOME: Directorio de instalación de Oracle.
ORACLE_SID: Base de datos por defecto.

Informix

INFORMIX_DIR: Directorio de instalación de Informix.
DBPATH: Directorio que contiene la base de datos (sólo SE).

Sybase

SYBASE: Directorio de instalación de Sybase.
SYBSERVNAME: Identificador del archivo de arranque del servidor (opcional). Este valor es equivalente a la variable DSQUERY que se define y utiliza con SYBASE.

-f=param_file

Indica el nombre de un archivo donde están escritas consecutivamente las opciones definidas para el programa. Esta opción se puede utilizar para ejecutar sin tener que dar docenas de opciones diferentes.

-i

Utiliza el modo de prueba interactiva para verificar si el servidor funciona correctamente.

-o=file

Indica el nombre del archivo o el dispositivo donde se escribirán los mensajes de la traza. Sólo funciona en modo **debug**.

-o2=file

Indica el nombre del archivo o el dispositivo donde se escribirán los mensajes de la traza Watchdog. Sólo funciona en modo **debug**.

-t=xx

Asigna un tiempo máximo. Este es el tiempo después del cual cualquier respuesta del PC se considera como una parada. El servidor **Tun SQL** para UNIX por lo tanto también dejará de funcionar. Este valor es de menor prioridad que el asignado al PC cuando se definió la Fuente de Datos.

-u

Se utiliza con el siguiente parámetro para definir la lista de usuarios autorizados ("*****" para todos). El valor por defecto es *****. Por ejemplo:

```
-x = *
-u = bill          (sólo "bill" está autorizado)
```

-v=XX

Asocia un número de versión de un SGBD con el servidor **Tun SQL** para UNIX. Es el valor que se mostrará en la lista al ejecutar **Tun DB Show**.

-x

Se utilice para definir la lista de usuarios sin acceso ("*****" para todos). Por ejemplo:

```
-u = *
-x = bill          (sólo "bill" no tiene acceso)
```

➢ Opciones de Informix

-h

De forma predeterminada, Informix no mantiene los cursores abiertos durante la ejecución de los comandos **commit** y **rollback**. La opción -h asegura que se mantienen abiertos tras uno de estos comandos si las aplicaciones que utilizan este tipo de servidor no son capaces de hacerlo por sí solas.

-n

Se aplica sólo a SCO UNIX 5. Si el sistema SCO UNIX 3.2 versión 5 no puede comprobar correctamente contraseñas de usuario, esta opción cancela la comprobación de contraseñas.

-s

De forma predeterminada, Informix no puede efectuar comandos **select** con una opción de ordenación (**group by** u **order by**) en una columna que no se haya incluido en el comando **select**. Puede utilizarse la opción **-s** para compensar esta carencia si no está previsto en las aplicaciones.

➢ Opciones de Oracle

-l

Si se han creado tablas, columnas, índices o vistas en el catálogo con caracteres minúsculos, la opción -l asegura que las funciones de catálogo devuelven datos entre comillas. Las aplicaciones que utilicen este servidor crearán consultas con nombres entre comillas.

➢ Opciones de Progress

-n

Se aplica sólo a SCO UNIX 5. Si el sistema SCO UNIX 3.2 versión 5 no puede comprobar correctamente contraseñas de usuario, esta opción cancela la comprobación de contraseñas.

-nor

La base de datos Progress no puede indicar cuántas líneas se han modificado o suprimido cuando se ejecuta un comando **update** o **delete**. De forma predeterminada, este servidor compensa dicha carencia. Sin embargo, hay una penalización de tiempo cada vez que se emite un comando **update** o **delete**. Si las aplicaciones que utilizan el servidor no precisan saber el número de líneas modificadas, la aplicación **-nor** cancela la compensación de servidor, ahorrando tiempo.

-ow

De forma predeterminada, el servidor no soporta la noción de propietario de objeto en la base de datos. De hecho, algunas aplicaciones intentan añadir un prefijo de propietario si se devuelven los propietarios con los objetos, provocando así errores de ejecución. Esta opción, añadir propietarios, es útil si la aplicación ha de conocer el propietario y los propietarios se administran correctamente.

-p=XX

Si han de utilizarse opciones específicas a Progress (por ejemplo, la opción -Q que fuerza a la base de datos a respetar la norma ANSI), debe utilizarse -p=XX, sustituyendo **XX** con una cadena que contenga las opciones apropiadas entrecomilladas.

-sv

La base de datos Progress utiliza sólo un tipo de cadena de carácter. De forma predeterminada, todas las cadenas se consideran de longitud fija (SQL_CHAR en ODBC). Si se utiliza esta opción, las cadenas se tratarán como si fueran de longitud variable (**SQL_VARCHAR** en ODBC).

-sy

Progress puede definir columnas de sistema u ocultas que no se devuelven al realizar una búsqueda con caracteres de sustitución. Por ello, el servidor no describe las columnas en su catálogo de forma predeterminada. Sin embargo, si se precisan las columnas ocultas, esta opción fuerza su retorno.

➢ Ver también

param.XXX, config.XXX

B

INSTRUCCIONES SQL EN C-ISAM

Instrucciones Principales

Sintaxis de las instrucciones SQL

Las instrucciones SQL se presentan utilizando la siguiente sintaxis:

- Los nombres reservados están en mayúsculas (INSERT, UNIQUE, etc.). Se puede, sin embargo, utilizar minúsculas al introducirlos en la línea de comandos.
- Los nombres de variables están en cursiva (*Databasename*, *etc.*).
- Los corchetes indican parámetros o entradas opcionales ([optional]).
- Las llaves y la expresión **xor** indican una elección única ({A xor B xor C}).
- El exponente n (^{n}) indica una secuencia que puede ser repetida desde 0 a n veces (sequence^{n}).

Las marcas de puntuación y los paréntesis son símbolos literales que deben ser introducidos tal y como aparecen.

➢ Propósito

Crea una base de datos nueva.

➢ Sintaxis

CREATE DATABASE *Basename*

Nota: El nombre de la base de datos debe tener menos de 18 caracteres.

➢ Uso

La base de datos creada se convierte en la base de datos actual.

Esta instrucción sólo se puede utilizar con **sqltools** (herramienta **Tun SQL**).

Tras la creación de la base de datos, se crea un direcC-ISAM extra que componen el catálogo (SysTables, SysColumns, SysIndexes, SysDefaults). Es un subdirectorio del directorio designado por la variable de entorno ISAM-PATH si se definió, o del directorio actual.

➢ Ejemplo

```
CREATE DATABASE TEST;
```

Crea el directorio test.ism con los archivos SysTables.dat, SysTables.idx, SysColumns.dat, SysColumns.idx, SysIndexes.dat, SysIndexes.idx, SysDefaults.dat, y SysDefaults.idx.

➢ Propósito

Crea una tabla nueva en la base de datos actual y pone las restricciones de la integridad de datos en sus columnas o en un grupo de columnas.

➢ Sintaxis

CREATE TABLE *tablename* (Column definition [,Column definition]^{n}) [,Constraint definition] ^{n}

Nota: El nombre de la tabla debe ser de menos de 18 caracteres.

➢ Uso

Los nombres de tablas de la misma base de datos deben ser únicos. Cada columna de una misma tabla debe tener un nombre diferente.

El nombre de la tabla puede ser precedido por el nombre de un usuario UNIX que será el propietario de la tabla. Si no se especifica ningún nombre se utiliza el ID actual.

➢ Ejemplo

```
CREATE TABLE TABLE_TEST (c1 char);
```

Esta instrucción crea la tabla *table1* en la base de datos relacional mediante la creación de los archivos C-ISAM asociados (*table1_100.dat* y *table1_100.idx*, por ejemplo).

Los nombres de archivos C-ISAM se crean cogiendo los primeros siete caracteres del nombre de la tabla y añadiendo un valor único. Si el nombre de la tabla tiene menos de siete caracteres, los nombres nuevos de los archivos se rellenan con caracteres de subrayado (_). El valor único para la primera tabla creada es 100: Este valor se irá incrementando de uno en uno para cada tabla nueva.

➢ Propósito

Define una tabla nueva en la base de datos actual, con restricciones de integridad de datos en sus columnas o en parte de ellas y opciones condicionadas a la existencia de archivos.

➢ Sintaxis

DEFINE TABLE *tablename* [File is option] (Column definition [,Column definition]^{n}) [,Constraint definition] ^{n}

Nota: El nombre de la tabla debe ser de menos de 18 caracteres.

➢ Uso

Los nombres de tablas de la misma base de datos deben ser únicos. Cada columna de una misma tabla debe tener un nombre diferente.

El nombre de la tabla puede ser precedido por el nombre de un usuario UNIX que será el propietario de la tabla. Si no se especifica ningún nombre se utiliza el ID actual.

➢ Ejemplo

```
DEFINE TABLE TABLE1 file is file_1 (c1 char);
```

En este ejemplo, los archivos C-ISAM *file_1* (*file_1.idx* y *file_1.dat*) ya existen: Sólo hay que definir la tabla.

Utilizado en las instrucciones **CREATE TABLE** y **DEFINE TABLE**.

➢ Propósito

La opción "colomn definition" de la instrucción **DEFINE TABLE** (**CREATE TABLE**) lista el nombre, el tipo, valor por defecto y restricciones para una única columna.

➢ Sintaxis

Columnname Data type [Default clause] [Not null clause] [Constraint definition subset]

➢ Ejemplo

```
CREATE TABLE PETS
(name char (20),
 race char (25),
 sex  char(1));
```

Esta tabla se compone de las columnas ***name***, ***race*** y ***sex***.

Utilizado en la opción **COLUMN DEFINITION**.

➢ Sintaxis

DEFAULT [{Literal xor NULL xor Currentl xor Today xor User}]

➢ Uso

El valor por defecto se inserta en la columna cuando no se especifica un valor explícito. Si no se especifica un valor por defecto y la columna permite dejarse en blanco, por defecto será NULL.

LITERAL	Caracteres o números constantes definidos por el usuario
NULL	Vacío (en blanco)
CURRENT	Fecha y hora actual (sólo para el tipo TIMESTAMP)
TODAY	Fecha actual (sólo para el tipo DATE)
USER	Nombre del usuario actual (sólo con tipos VAR o VARCHAR)

➢ Ejemplo

```
CREATE TABLE PETS
(name char (20),
 race char(25),
 sex char(1) DEFAULT 'M')
```

En esta tabla el valor por defecto para ***sex*** es un literal: 'M'.

Utilizado en la opción **COLUMN DEFINITION**.

➢ Propósito

Si no se indica un valor por defecto para una columna, por defecto se deja en blanco a no ser que se ponga NOT NULL tras el tipo de datos de la columna. En este caso no existirá un valor por defecto para la columna.

➢ Sintaxis

NOT NULL

➢ Ejemplo

```
CREATE TABLE INVOICE
(invoice_id longint NOT NULL,
 customer_name char (30))
```

Si la columna se definió con NOT NULL (y no se especificó un valor por defecto), tendrá que introducir un valor en esta columna al insertar una fila o al actualizar esa columna en una fila. De lo contrario, el servidor de la base de datos devolverá un error.

Utilizado en la opción **COLUMN DEFINITION**.

➢ Propósito

El subconjunto de definición de restricciones permite crear restricciones para una columna.

➢ Sintaxis

{UNIQUE xor PRIMARY KEY} [CONSTRAINT *Constraint name*]

Nota: El nombre de la restricción no debe tener más de 18 caracteres y debe ser único en la base de datos.

➢ Uso

UNIQUE	El campo tiene que ser único
PRIMARY KEY	El campo debe ser único y además la clave primaria de la tabla

Si no se especifica nombre para la restricción, se asigna un nombre por defecto.

➢ Ejemplo

```
CREATE TABLE INVOICE
(invoice_number longint UNIQUE CONSTRAINT un_invoice,
 customer_name char (30))
```

La restricción de unicidad del número de factura se llama ***un_invoice***.

Utilizado en las instrucciones **CREATE TABLE** y **DEFINE TABLE**.

➢ Propósito

La opción de definición de restricciones permite crear restricciones para un conjunto de columnas (de 1 a 8 columnas).

➢ Sintaxis

{UNIQUE xor PRIMARY KEY} (*Columnname* [,*Columnname*]^{n})[CONSTRAINT *Constraint name*]

➢ Uso

UNIQUE	El campo debe ser único para el conjunto de columnas
PRIMARY KEY	El campo debe ser único y la clave primaria para el conjunto de columnas

Si no se especifica nombre para la restricción, se asigna un nombre por defecto.

Cada columna nombrada en una restricción debe ser una columna de la tabla y no puede aparecer en la restricción mas que una vez.

➢ Ejemplo

```
CREATE TABLE FAMILY
(name char (20),
 surname char (20),
 birth_date date,
 PRIMARY KEY (name, surname) CONSTRAINT pk_family)
```

La restricción de clave primaria ***pk_family*** afecta a los campos ***name*** y ***surname***.

Utilizado en la instrucción **DEFINE TABLE**.

➢ Propósito

Define el uso de la tabla de acuerdo a la existencia o no de un archivo.

➢ Sintaxis

FILE IS *filename*

➢ Uso

Si se utiliza esta opción, no se crean archivos C-ISAM. Tun SQL deberá encontrar los archivos *filename.dat* y *filename.idx* en el directorio de la base de datos actual antes de poder utilizar la tabla.

Si no se utiliza la opción, se asigna un nombre por defecto a los archivos creados en el directorio de la base de datos actual.

➢ Ejemplos

```
CREATE DATABASE TEST;
DEFINE TABLE TABLE1 FILE IS FIC1 (C1 CHAR);
```

En este ejemplo no se crean los archivos. Los archivos *foo1.dat* y *foo1.idx* deben ser añadidos al directorio *test.ism*.

```
DEFINE TABLE TABLE1 (C1 CHAR);
```

En este ejemplo, se crean los archivos *table_100.dat* y *table_100.idx* (nombres por defecto) en el directorio *test.ism*.

➢ Propósito

Crea un índice para una o más columnas de una tabla (de 1 a 8 columnas).

➢ Sintaxis

CREATE {UNIQUE xor DISTINCT} INDEX *indexname* ON *tablename* (*Columnname* [,*Columnname*] ^{n})

Notas: El nombre del índice no puede tener más de 18 caracteres. El número de columnas puede variar de 1 a 8.

➢ Uso

UNIQUE	Restringe que el índice sea único
DISTINCT	Sinónimo de UNIQUE

Para tablas definidas con la opción FILE IS, la instrucción CREATE INDEX debe ser ejecutada antes de copiar los archivos filename.dat y filename.idx.

➢ Ejemplo

```
CREATE DISTINCT INDEX ix_name ON Table1 (name,
birth_date) ;
```

Esta instrucción crea el índice ***ix_name*** para las columnas ***name*** y ***birth_date*** de la tabla ***Table1***.

➢ Propósito

Atribuye un sinónimo a una tabla.

➢ Sintaxis

CREATE SYNONYM *synonymname* FOR *tablename*

> **Nota:**
> El nombre del sinónimo no debe exceder de 18 caracteres.

➢ Uso

El nombre de sinónimo debe ser precedido por el nombre de un usuario UNIX que se convierte en el propietario del sinónimo. Si no se indica un nombre, se toma por defecto el ID actual.

➢ Ejemplo

```
CREATE SYNONYM empleados FOR Table1;
```

Esta instrucción crea el sinónimo empleados para la tabla Table1.

COMMENT

➢ Propósito

Atribuye un comentario a una tabla o sinónimo.

➢ Sintaxis

COMMENT ON {*tablename* xor *synonymname*} IS 'comment string'

➢ Ejemplo

```
COMMENT on TABLE1 IS 'Employees' Table'
```

DROP DATABASE

➢ Propósito

Elimina (borra) la base de datos completa, incluyendo catálogos, índices y datos.

➢ Sintaxis

DROP DATABASE *basename*

Nota: El nombre de la base de datos debe ser de menos de 18 caracteres.

➢ Uso

Una base de datos que está siendo utilizada por otro usuario no puede ser borrada.

La instrucción DROP DATABASE no elimina el directorio de la base de datos si contiene otros archivos además de los creados para las tablas de la bases de datos y los índices.

➢ Ejemplo

```
DROP DATABASE DBTEST;
```

Borra la base de datos ***DBTEST***.

CONNECT DATABASE

➢ Propósito

Se conecta con una base de datos distinta. No se pueden realizar operaciones en una base de datos si no se está conectado a ella.

➢ Sintaxis

CONNECT DATABASE *basename*

➢ Ejemplo

```
CONNECT DATABASE DBTEST2;
```

Se conecta con la base de datos ***DBTEST2***.

DISCONNECT DATABASE

➢ Propósito

Desconectarse de la base de datos actual.

➢ Sintaxis

DISCONNECT DATABASE *basename*

➢ Ejemplo

```
DISCONNECT DATABASE DBTEST2;
```

Se desconecta de la base de datos ***DBTEST2***.

➢ Propósito

Borra un índice.

➢ Sintaxis

DROP INDEX *indexname*

➢ Ejemplo

```
DROP INDEX ix_name ;
```

Borra el índice ***ix_name***.

➢ Propósito

Borra un sinónimo previamente definido.

➢ Sintaxis

DROP SYNONYM *synonymname*

➢ Uso

Si una tabla se borra, el sinónimo permanece hasta que se borre específicamente con la instrucción DROP SYNONYM.

➢ Ejemplo

```
DROP SYNONYM empleados;
```

Borra el sinónimo **empleados** atribuido a la tabla **TABLE1**. La tabla no se borra con esta sentencia.

➢ Propósito

Borra una tabla, además de sus índices y datos asociados.

➢ Sintaxis

DROP TABLE {*tablename* xor *synonymname*}

➢ Uso

Si un sinónimo es borrado por una sentencia **DROP TABLE,** la tabla también es borrada.

Si la sentencia **DROP TABLE** se aplica a una tabla, los sinónimos de la tabla solo se borraran si se usa la sentencia **DROP SYNONYM**.

➢ Ejemplo

```
DROP TABLE TABLE1;
```

Borra la tabla TABLE1, sus índices y sus datos.

UNDEFINE TABLE

➢ Propósito

Borra una tabla definida con la instrucción DEFINE pero no borra los archivos de datos e índices asociados.

➢ Sintaxis

UNDEFINE TABLE {*tablename* xor *synonymname*}

➢ Ejemplo

```
UNDEFINE TABLE TABLE1;
```

Borra la tabla TABLE1 creada con la instrucción DEFINE TABLE TABLE1.

➢ Propósito

Consultar una base de datos.

➢ Sintaxis

SELECT Select clause From clause [Where clause] [Group by clause] [Having clause] [Order by clause]

➢ Uso

Se pueden consultar las tablas de la base de datos actual u otra diferente.

➢ Ejemplo

```
SELECT customer_name
FROM customers
WHERE turn_over > 250
ORDERBY country ;
```

Esta consulta lee los clientes con una facturación anual de más de 250 de la tabla *customers* y devuelve sus nombres ordenados por país.

Utilizado en las instrucciones **SELECT** y **INSERT**.

➢ Sintaxis

[{ALL xor DISTINCT xor UNIQUE}] {Expression [[AS] *Display label*] [Expression [[AS] *Display label*]] [n]

En la instrucción SELECT, se especifica exactamente qué datos seleccionar y si se quiere omitir valores duplicados.

➢ Uso

ALL	Se devolverán todos los valores seleccionados, incluso si están repetidos.
DISTINCT	Borra las filas repetidas de los resultados de la consulta.
UNIQUE	Sinónimo de DISTINCT.

➢ Ejemplo

```
SELECT customer_name
FROM customers
WHERE turn_over > 250
ORDERBY country ;
```

Esta consulta lee los clientes con una facturación anual mayor de 250 en la tabla *customers* y devuelve sus nombres ordenados por país.

```
SELECT order_date, COUNT(*), paid_date - order_date
FROM orders
GROUP BY 1, 3
```

Esta consulta devuelve la fecha de pedido, el número de pedidos y la diferencia entre las fechas de pedido y de pago, agrupados por fecha de pedido y tiempo de demora (diferencia entre las fechas).

Utilizado en la instrucción **SELECT**.

➢ Sintaxis

{ [{*tablename* xor *synonymname* xor *tablealias*}.] *columnname*
xor NULL
xor Literal number
xor Quoted string
xor User
xor Aggregate expression}

➢ Ejemplo

```
'Cordwainer'
```

La secuencia de caracteres ***'Cordwainer'*** es una subexpresión.

Utilizado en las instrucciones **SELECT** y **DELETE**.

➢ Propósito

Listar la tabla o tablas de donde se seleccionan los datos.

➢ Sintaxis

FROM {Table name xor Synonym name } [[AS] *Table alias*]
[,{Table name xor Synonym name} [[AS] *Table alias*]] [n]

➢ Ejemplo

```
SELECT customer_name, order_num
FROM customers c, orders o
WHERE c.customer_num = o.customer_num ;
```

En esta instrucción, los datos se extraen de las tablas ***customers*** y ***orders*** y las tablas usan alias.

Utilizado en las instrucciones **SELECT**, **DELETE** y **UPDATE**.

➢ Propósito

Especificar las condiciones de búsqueda y de unión para los datos seleccionados.

➢ Sintaxis

WHERE Condition [AND Condition] ^{n}

➢ Ejemplo

```
SELECT customer_name
FROM customers
WHERE last_order_date < '28/07/1993'
ORDERBY country ;
```

En este ejemplo, la condición de búsqueda es la última fecha de pedido.

Utilizado en la instrucción **SELECT**.

➢ Propósito

Produce una única fila de resultados para cada grupo.

➢ Sintaxis

GROUP BY {Table name xor Synonym name } . *Column name* xor *Select number*}
[,{Table name xor Synonym name} . *Column name* xor *Select number*}]^{n}

➢ Uso

Un grupo es un conjunto de filas que tienen el mismo valor para cada columna listada.

La variable "select number" es un entero que representa la posición de la columna en la instrucción SELECT.

➢ Ejemplo

```
SELECT order_date, COUNT(*), paid_date - order_date
FROM orders
GROUPBY order_date, 3 ;
```

Los resultados se agrupan por ***order_date*** y ***paid_date - order_date***.

Utilizado en la instrucción **SELECT**.

➢ Propósito

Aplicar una o más condiciones a grupos.

➢ Sintaxis

HAVING Condition

➢ Ejemplo

```
SELECT customer_num, call_dtime, call_code
FROM cust_calls
GROUP BY call_code, 2 , 1
HAVING customer_num < 42 ;
```

Esta consulta devuelve las tablas call_code, call_dtime y customer_num y las agrupa por call_code para todas las llamadas de clientes cuyo número de cliente sea menor de 42.

Utilizado en la instrucción **SELECT**.

➢ Propósito

Ordenar los resultados de la consulta por los valores contenidos en una o más columnas.

➢ Sintaxis

ORDER BY {*Table name*. xor *Synonym name* .} *Column name* xor *Select number* xor *Display label*} [,{*Table name*. xor *Synonym name* .} *Column name* xor *Select number* xor *Display label*}]^{n}

➢ Uso

La variable "select number" es un entero que representa la posición de una columna en la instrucción SELECT.

➢ Ejemplo

```
SELECT customer_name
FROM customers
WHERE turn_over > 250
ORDERBY country ;
```

Esta consulta devuelve el resultado ordenado por país.

DELETE

➢ Propósito

Borra una o más filas de una tabla.

➢ Sintaxis

DELETE FROM {*Table name* xor *Synonym name* }
[WHERE {Condition xor CURRENT OF *Cursor name*}]

➢ Ejemplo

```
DELETE FROM customers WHERE last_order_date<1992;
```

Esta instrucción borra las filas de la tabla *customers* cuya última fecha de pedido sea anterior a 1992.

➢ Propósito

Inserta una o más filas en la tabla.

➢ Sintaxis

INSERT INTO {*Table name* xor *Synonym name*} [(Column name [,Column name] ^{n})] {Values clause xor Select clause}

➢ Ejemplo

```
INSERT INTO Pets VALUES ('Socks', 'Cat', 'M') ;
```

Esta instrucción inserta los valores ***'Socks'***, ***'Cat'*** y ***'M'*** en la tabla ***Pets***.

Utilizado en la instrucción **INSERT**.

➢ Sintaxis

VALUES ({*Variable name* : *Indicator variable* xor NULL xor Literal number xor Quoted string xor Literal Timestamp xor Literal date xor Literal time} [,{*Variable name* : *Indicator variable* xor NULL xor Literal number xor Quoted string xor Literal Timestamp xor Literal date xor Literal time}]^{n})

➢ Uso

Al utilizar la instrucción VALUES, sólo se puede insertar una fila de cada vez. Cada valor que siga a la palabra VALUES se asigna a la correspondiente columna listada con la instrucción INSERT INTO (o a las columnas en orden si no se especificó una lista de columnas).

➢ Ejemplo

```
INSERT INTO Pets VALUES ('Socks', , 'M') ;
```

Esta instrucción inserta el valor *'**Socks**'* en la primera columna y *'**M**'* en la tercera columna de la tabla ***Pets***. La segunda columna no se toca.

➢ Propósito

Cambiar el valor de una o más columnas o filas de una tabla.

➢ Sintaxis

UPDATE {*Table name* xor *Synonym name*} SET Set clause [WHERE {Condition xor CURRENT OF *Cursor name*}]

➢ Ejemplo

```
UPDATE Catalog SET item_price = 10 WHERE item_type =
'L' ;
```

Esta instrucción cambia a 10 el precio de todos los artículos del tipo ***'L'*** de la tabla ***Catalog***.

Utilizado en la instrucción **UPDATE**.

➢ Sintaxis

{Column name = {Constant xor Select statement xor NULL} [,Column name = {Constant xor Select statement xor NULL}]^{n}
xor {(*Column name* [,*Column name*]^{n} xor *} = (Select statement)

➢ Ejemplo

```
UPDATE Catalog SET item_price = 10
```

En esta instrucción, la clausula SET cambia el precio item_price a 10.

Utilizado en la instrucción **SELECT** o un expression.

➢ Sintaxis

{COUNT(*)
xor
{MIN xor MAX xor SUM xor AVG xor COUNT} ([{DISTINCT xor UNIQUE}] {*Table name* xor *Synonym name* xor *Table alias*} . *Column name*)
}

➢ Ejemplo

```
SELECT COUNT (DISTINCT item_type) FROM Catalog ;
```

Esta instrucción devuelve el número de tipos diferentes de un artículo en la tabla ***Catalog***.

Tipos de datos

Esta sección describe los tipos de datos soportados por el lenguaje SQL con sus definiciones en C equivalentes. Las descripciones corresponden a los tipos SQL utilizados por CREATE TABLE. Están en modo nativo equivalente. Para la importación de archivos creados fuera de la base de datos C-ISAM con el comando DEFINE TABLE, se indican las diferencias.

En la siguiente tabla, los tipos en cursiva de la columna "Tipo en lenguaje C" sólo se aplican a la utilización de CREATE TABLE.

Tipo SQL en la base de datos	Tipo SQL en ODBC	Tipo en lenguaje C
bit	SQL_BIT	*unsigned char notnull_data;* unsigned char data;
byte	SQL_TINYINT	*unsigned char notnull_data;* unsigned char data;

char(maxlength) 1 <= maxlength <= 32511	SQL_CHAR	char data[maxlength];
varchar(maxlength) 1 <= maxlength <= 32511	SQL_VARCHAR	char data[maxlength];
binary(maxlength) 1 <= maxlength <= 32511	SQL_BINARY	*unsigned char notnull_data;* unsigned char data[maxlength];
varbinary(maxlength) 1 <= maxlength <= 32511	SQL_VARBINARY	*unsigned char size_data[2];* *unsigned char data[maxlength];*
smallint	SQL_SMALLINT	short data; /* 2 bytes */
longint	SQL_INTEGER	long data; /* 4 bytes */
real	SQL_FLOAT	float data; /* 4 bytes */
double	SQL_DOUBLE	double data; /* 8 bytes */
decimal(prectot, precdec) 1<=prectot<=32 et 0<=precdec<=prectot	SQL_DECIMAL	char data[(prectot+1)/2+1];
date	SQL_DATE	unsigned long data;
time	SQL_TIME	unsigned long data;
timestampa	SQL_TIMESTAMP	unsigned long data;

➢ El tipo bit

Este tipo corresponde al tipo SQL_BIT en ODBC. Guarda los valores binarios 0 y 1, o el valor nulo (null).

Si este tipo de datos se utiliza con CREATE TABLE, se reserva un bloque de dos bytes en el archivo creado para diferenciar entre el valor nulo y 0 ó 1. Si se utiliza un campo para la definición de la clave, los dos bytes se utilizan en la clave.

Si el tipo de datos bit se utiliza con DEFINE TABLE, sólo se reserva un byte. El valor 0 no se diferencia totalmente del valor nulo. Por tanto con DEFINE TABLE este tipo de campo no puede ser nulo.

La siguiente tabla muestra los valores que se guardan. Los datos en cursiva sólo se aplican a CREATE TABLE.

Campo tipo bit	***notnull_data***	**Valor de los datos**
0	*1*	0
1	*1*	1
Null	*0*	*0*

➢ El tipo byte

Este tipo corresponde al tipo SQL_TINYINT en ODBC. Guarda valores desde 0 hasta 255, o nulo.

Si este tipo de datos se utiliza con CREATE TABLE se reserva un bloque de dos bytes (como en el tipo **bit**) para diferenciar el valor nulo de los demás valores. Si este tipo de campo se utiliza para la definición de la clave, los dos bytes se utilizan en la clave.

Si el tipo de datos se utiliza con DEFINE TABLE, sólo se reserva un byte. El valor 0 no se puede diferenciar del nulo. Con DEFINE TABLE este tipo de campo no puede ser nulo.

La siguiente tabla muestra los valores que se guardan. Los datos en cursiva sólo se aplican a CREATE TABLE.

campo tipo byte	***notnull_data***	**valor de los datos**
0	*1*	0
1	*1*	1
...	*1*	...
255	*1*	255
null	*0*	*0*

➢ El tipo char

Este tipo corresponde al tipo SQL_CHAR en ODBC. Puede almacenar de 1 a 32511 caracteres.

Cuando este tipo de datos se inserta en un campo, todos los espacios no significativos al final de la secuencia se borran y los bytes correspondientes del archivo se ponen a '\0' (código ASCII 0). Cuando se leen los datos de estos campos, la secuencia leída se completa automáticamente con espacios (código ASCII 32) hasta su tamaño máximo. Si se guarda una secuencia vacía en un campo de tipo **char**, se escribe un único espacio en el campo para distinguirlo del nulo.

campo tipo char(10)	**valor de los datos**
‘’	0x32000000000000000000
‘ ‘	0x32000000000000000000
‘A ‘	0x41000000000000000000
‘AaBb’	0x41614262000000000000
null	0x00000000000000000000

➢ El tipo varchar

Este tipo corresponde al tipo SQL_VARCHAR en ODBC. Se utiliza prácticamente de la misma forma que el tipo **char**. También almacena de 1 a 32511 caracteres.

Se diferencia del tipo **char** en que al insertar los datos en este tipo de campo, no se borran ninguno de los espacios no significativos al final de la secuencia pero los correspondientes bytes en el archivo se rellenan con el carácter '\0'. Cuando se leen los datos de estos campos, el ODBC devuelve la secuencia sin modificar. Igual que para el tipo **char**, una secuencia vacía se guarda como un carácter de espacio para distinguirlo del nulo.

campo tipo varchar(10)	valor de los datos
‘’	0x32000000000000000000
‘ ‘	0x32000000000000000000
‘A ‘	0x41202020202000000000
‘AaBb’	0x41614262000000000000
null	0x00000000000000000000

➢ El tipo binary

Este tipo corresponde al tipo SQL_BINARY en ODBC. Puede almacenar de 1 a 32511 caracteres.

Cuando se utiliza este tipo de datos con CREATE TABLE, se utiliza un byte extra (notnull_data). Este byte puede ponerse a 0 o a 1 para distinguir el valor nulo de la secuencia vacía. Si un campo de tipo **binary** se utiliza en la definición de una clave, el byte notnull_data se incluye con los bytes de datos en la clave.

Con DEFINE TABLE no se añade el byte extra al utilizar el tipo binary. Sólo se guardan los bytes significativos. El valor 0 no se puede distinguir del valor nulo. Por esta razón este tipo de campo no puede ser nulo.

Cuando se insertan datos en este tipo de campo, los bytes correspondientes en el archivo se completan con caracteres '\0'. Cuando se lee este tipo de campo, ODBC recupera todos los bytes.

La siguiente tabla muestra los valores que se guardan. Los datos en cursiva sólo se aplican a CREATE TABLE.

campo tipo binary(10)	*notnull_data*	valor de los datos
0x	*1*	0x00000000000000000000
0x00	*1*	0x00000000000000000000
0x1234	*1*	0x12340000000000000000
null	*0*	*0x00000000000000000000*

➢ El tipo varbinary

Este tipo corresponde al tipo SQL_VARBINARY en ODBC. Puede almacenar de 1 a 32511 caracteres.

Este tipo sólo se puede utilizar con el comando CREATE TABLE. Con DEFINE TABLE no se pueden utilizar campos del tipo **varbinary**. Se utilizan dos bytes extra con este tipo de datos. Los bytes guardan la longitud de los datos binary más un entero. El valor 0 corresponde a un binary nulo y el valor 1 a un binary de longitud cero.

Igual que en el tipo **binary**, al insertar datos en este tipo de campo, el bloque correspondiente en el archivo se completa con caracteres '\0'. Cuando se lee este tipo de campo, el ODBC sólo recupera los bytes con la longitud binary. Si un campo de tipo **varbinary** se utiliza para definir una clave, los bytes size_data no se incluyen con los bytes de datos en la clave.

campo de tipo varbinary(10)	size_data	valor de los datos
0x	0x0001	0x00000000000000000000
0x00	0x0002	0x00000000000000000000
0x1234	0x0003	0x12340000000000000000
null	0x0000	0x00000000000000000000

➢ El tipo smallint

Este tipo corresponde al tipo SQL_SMALLINT en ODBC. Almacena enteros en el rango -32767 a 32767 en 2 bytes. El valor –32768 se reserva para el valor nulo.

campo tipo smallint	valor de los datos
-32767	**-32767**
0	**0**
30000	**30000**
null	**-32768**

➢ El tipo longint

Este tipo corresponde al tipo SQL_INTEGER en ODBC. Almacena valores enteros en el rango -134217727 a 134217727 en 4 bytes. El valor –134217728 se reserva para el valor nulo.

campo tipo longint	valor de los datos
-32767	-32767
0	0
134217	134217
null	-134217728

➢ El tipo real

Este tipo corresponde al tipo SQL_REAL y SQL_FLOAT en ODBC. Almacena números con coma flotante en formato máquina de 4 bytes.

campo tipo real	valor de los datos
-25	(float)-25.0
0	(float)0.0
3.1415	(float)3.1415
null	valor no significativo

➢ El tipo double

Este tipo corresponde al tipo SQL_DOUBLE en ODBC. Almacena números con coma flotante en formato máquina de 8 bytes.

campo tipo double	valor de los datos
-25	(double)-25.0
0	(double)0.0
3.1415	(double)3.1415
null	valor no significativo

➢ El tipo decimal

Este tipo corresponde al tipo SQL_DECIMAL en ODBC. Guarda números con coma fija.

El tipo decimal debe ser seguido de dos parámetros entre corchetes, "c1 decimal(n, m)", donde n es el número total de dígitos y m el número de decimales. Si no se especifica la m se toma por defecto el valor 0.

Las funciones C-ISAM stdecimal() y lddecimal() se utilizan para leer/escribir campos de este tipo.

➢ El tipo date

Este tipo corresponde al tipo SQL_DATE en ODBC. Almacena una fecha como el número de días transcurridos desde el 1 de Enero del año 0 en un entero de 4 bytes. Para insertar o comprobar una fecha con comandos SQL se utiliza la notación ODBC ({d 'AAAA-MM-DD' }).

campo tipo date	valor de los datos
{ d '0000-01-01' }	0
{ d '1997-02-17' }	729438
null	-134217728

➢ El tipo time

Este tipo corresponde al tipo SQL_TIME en ODBC. Almacena la hora como el número de segundos en el día en un entero de 4 bytes. Para insertar o comprobar una hora con comandos SQL se utiliza la notación ODBC ({ t 'hh :mm :ss' }).

campo tipo time	valor de los datos
{ t '00 :00 :00' }	0
{ t '13 :40 :10' }	49210
null	-134217728

➢ El tipo timestamp

Este tipo corresponde al tipo SQL_TIMESTAMP en ODBC. Almacena la hora como el número de segundos transcurridos desde el 1 de Enero de 1970 (similar a la función time() en C). El valor máximo corresponde a la fecha 5 de febrero del 2036, 00.00 h. Para insertar o comprobar timestamp con comandos SQL se utiliza la notación ODBC ({ ts 'AAAA-MM-DD hh :mm :ss' }).

campo tipo timestamp	valor de los datos
{ ts '1970-01-01 00 :00 :00' }	0
{ ts '1997-02-17 13 :40 :10' }	856186810
null	-134217728

ÍNDICE

U

V

W