MultiBase

Notas a la versión 3.0, Rel. 3.0

BASE100

BASE 100, S.A.
www.base100.com

Índice

1 Nuevas funcionalidades implementadas y errores corregidos

1.1 Instalador

1. Se ha incorporado el idioma alemán a la distribución. Se añaden los ficheros de mensajes y los diálogos en alemán de los programas asociados al EasyReport (EasyReport, Confedit, Cdsedit y Wedit).

2. Se incluyen en el directorio "msg" los ficheros "tdocu.int", "systmenu", e "inwords.dll" (la del idioma de instalación seleccionado), que por error no se incluían en la instalación de la versión 3.0 2.0. Además, se crea el directorio "c:\tmp" en caso de que no exista, cosa que tampoco se hacía en la versión 3.0 2.0.

1.2 CTL

1. Desconexiones dinámicas: Implementación de la función **sqldisconnect()**, mediante la cual se logra la desconexión de la base de datos en uso en el programa CTL, consiguiendo así un mecanismo de conexión/desconexión dinámica en un programa CTL. En el momento que desconectamos de una base de datos se pueden cambiar los valores de las variables relacionadas con las conexiones a nuevas bases de datos. Por ejemplo, cambiar las variables DBHOST, DBPATH, DBUSER, DBPASSWD y DBSERVICE para poder acceder a otras bases de datos residentes en otros ordenadores (cliente-servidor).

2. Función **getipaddress()**: Devuelve la dirección IP de la máquina cliente en la cual se está ejecutando el programa CTL.

3. Cambios realizados al optimizador: El optimizador da preferencia a las columnas de la cláusula WHERE frente a las del ORDER BY. Entre las columnas de la WHERE, las que más preferencia tienen son las que están condicionadas por el operador igual (=). Anteriormente, en caso de que varios índices de una tabla comenzasen por la(s) misma(s) columnas, en el momento que se condicionaba la primera columna de cualquiera de dichos índices, siempre optimizaba por el primer índice definido para la tabla. En estos momentos, el optimizador sigue analizando el resto de condiciones y selecciona el índice más parecido al total de condiciones establecidas en la cláusula WHERE. Por ejemplo: Una tabla compuesta por los siguientes índices:

```
Índice 1: empresa, cliente, fecha.
Índice 2: empresa, cliente, total_facturado.
Índice 3: empresa, cliente, cif
```

En las versiones anteriores de MultiBase, cuando se condicionaba en la cláusula WHERE la columna "empresa", cualquier índice era válido y, por tanto, siempre seleccionaba el "índice 1" siempre y cuando no hubiera condiciones por igual en el total de las columnas de otro índice. Por ejemplo:

```
empresa = 1 and cliente = 2 and cif = "A234343"
```

En este caso sí seleccionaba el índice 3. Sin embargo, si la condición era:

```
empresa = 1 and cliente > 2 and cif > "A"
```

El índice a seleccionar siempre era el "índice 1". En este caso, el programador sólo tenía la opción de poner como primera columna a condicionar en la cláusula WHERE aquella que fuera distinta en los índices. En el ejemplo anterior, para la versión 3.0 2.0 y anteriores habría que condicionar en primer lugar el "cif" y después las otras dos columnas para que seleccionara el "índice 3". Actualmente, en la versión 3.0 3.0 el optimizador sigue analizando hasta el final de las condiciones para evaluar qué índice es el más idóneo. Por tanto, en nuestro ejemplo, independientemente del orden de las condiciones, el índice por el que optimizará será "índice 3".

4. La optimización trataba mal las funciones de fecha que aparecían en la WHERE ("day", "month", etc.), dando lugar a errores que provocaban que una consulta sobre una tabla que tenía que devolver varias filas no devolviera ninguna. Este problema se ha solucionado en esta versión.

5. Cuando sobre un CURSOR se definían muchas instrucciones no procedurales (sección CONTROL), podía llegar a ocurrir que se acabaran corrompiendo los datos de ese CURSOR, generando comportamientos extraños del programa CTL, que al final acababan con un error de protección general del sistema. Este error ha sido corregido.

6. Variables de entorno pasadas al servidor SQL: Se ha hecho que en una conexión cliente-servidor con el SQL, el CTL le pase al servidor las variables de entorno OUTOPT2 y OUTOPT3 (que sirven para obtener información sobre la optimización), que en las versiones anteriores no se las pasaba.

7. Tratamiento de "Warnings" con la instrucción WHENEVER: Mediante la definición de la variable de entorno ERRWSQL con valor "ON", la instrucción WHENEVER ERROR, además de detectar los errores del SQL, también se activará con la siguiente lista de warnings del SQL:

```
-1117, -1196, -1202, -1214, -1261, -1299
```

(Consulte el **Manual del Administrador** para ver el mensaje de error que se asocia a cada uno.)

Si ERRWSQL no está definida, o posee un valor distinto de "ON", estos "warnings" no son detectables desde el CTL.

8. Lentitud en conexiones cliente-servidor con cliente Windows: Existían problemas de velocidad en las conexiones cliente-servidor en las que la máquina cliente era Windows. Se ha detectado que el problema radicaba en el tratamiento de los "sockets" y en las conexiones TCP/IP que hace Windows ("winsock.dll"). A la máquina cliente CTL de Windows le costaba mucho tiempo recibir los datos del servidor (lectura del "socket" establecido con el servidor). Este problema ocurría sobre todo en los "unload" y en los "query". El problema se soluciona haciendo más grande el tamaño del "búffer" que el cliente le pide al servidor que le rellene con el resultado de la consulta. Así pues, se ha modificado el CTL para que en los "query" y en los "unload" el tamaño de ese "búffer" sea suficientemente grande (8 Kbytes). En versiones anteriores se trabajaba con tres tamaños de "búffer" (256, 512, y 1.024 bytes), escogiendo uno u otro según el número de bytes de cada "tupla" de la consulta a realizar.

Además, se ha observado que en los cursores del CTL también podía darse este problema, por lo que se ha añadido la posibilidad de que el usuario controle el tamaño del "búffer" que el cliente le solicita al servidor mediante una variable de entorno denominada MBIOROWS. En esta variable se puede indicar el número de filas o "tuplas" de la consulta que deseamos que el servidor nos prepare cada vez. El funcionamiento es el siguiente:

- Si MBIOROWS no está definida o tiene un valor no válido: funciona como en la versión antigua, solicitando un buffer dependiendo del número de bytes de la tupla de la consulta:

tupla <= 256 bytes	Se solicitan 512 bytes.
256 bytes < tupla <= 512 bytes	Se solicitan 1.024 bytes.
tupla > 512 bytes	Se solicita el número de bytes de la "tupla".

- Si MBIOROWS está definida con un valor numérico válido se solicita un número de bytes que es: "(nº de bytes de la tupla) * MBIOROWS". Si este valor supera los 8 Kbytes, se solicitan como máximo los 8 Kbytes.

La variable MBIOROWS se puede establecer desde un programa CTL con una llamada a la función interna "**putenv()**", pero no puede cambiarse para un mismo CURSOR salvo que hagamos una llamada a la función "**sqlreset()**", porque una vez que hayamos abierto el CURSOR se habrá quedado con el valor de MBIOROWS que estuviera fijado.

1.3 Ctlcomp

La corrección de los errores de esta nueva versión no implica la recompilación de módulos.

1.4 Servidor CTSQL-NT

Se ha arreglado el problema que acababa provocando un error de protección general en el que saltaba la dll del sistema "user32.dll".

Actualmente, no puede emplearse la palabra reservada NOPASSWD como valor de la variable de entorno DBPASSWD.

1.5 Gateway de Oracle para Windows NT

Se han incluido cambios y correcciones para que el gateway funcione con Oracle 8.

Se han corregido varios errores: problemas en el ALTER TABLE y en el tratamiento y creación de índices.

1.6 Tprocess y Tlisterr: errores año 2000

En el "Tprocess", al usar el comando "@d", que imprime la fecha, en el año 2000 ponía como año el 100. Esto se ha solucionado utilizando 4 dígitos para el año. Además, en diciembre, en vez de DIC, ponía DI, y esto también se ha corregido.

En el "Tlisterr" ocurría este mismo error con el año 2000 (no con los meses), al indicar la fecha de actuación sobre una tabla. Se ha corregido del mismo modo que en el "Tprocess".

1.7 Wedit

No se posicionaba en el primer error del fichero ".err", sino que lo hacía en la primera fila de éste.

Por cada sesión, el Wedit genera un fichero "wedit.ini" con la configuración local de dicha sesión. Por tanto, si el operador cambia la fuente, ésta queda permanentemente modificada en dicho fichero "*.ini".

1.8 EasyReport

Se le ha añadido la variable de entorno DBDATERANGE, con la misma funcionalidad que en el CTL para el tratamiento de las fechas.

Se ha corregido un error relacionado con el manejo de grupos.

1.9 ODBC

Se incluye la nueva versión del driver de ODBC, que corrige errores relacionados con el manejo de fechas y el acceso concurrente a tablas.

1.10 Utildll

Se han añadido nuevas funciones a esta dll. Para más información ver el fichero ".doc" que la acompaña.