MultiBase

Notas a la versión 3.2

La versión 3.2 de MultiBase incorpora un nuevo motor CTSQL cuyas características principales se detallan en este documento.

BASE100

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

Índice

1 Incompatibilidades con versiones anteriores

1.1 Formato de la base de datos

Este SQL trabaja por defecto:

- Sin límite de duplicados (DUPLIMITS=OFF).
- Las bases de datos que cree el CTSQL junto con las tablas del catálogo NO tendrán la limitación de 8 caracteres en su nombre (como ocurre en UNIX) (DOSNAME = OFF).

  Esta característica ya estaba implícita en el motor de MultiBase en su versión para Unix, pero al unificar los fuentes la variable DOSNAME permitirá elegir el tamaño de los nombres.

Una base de datos creada con versiones posteriores a la 3.0 2.x de CTSQL no se puede leer con versiones anteriores.

Para poder crear o utilizar bases de datos compatibles con versiones anteriores se deben definir las variables "DUPLIMITS=ON" y "DOSNAME=ON".

La variable DUPLIMITS admite dos valores: ON y OFF. Si toma el valor ON indica que estaremos trabajando con límite de duplicados (32.767 valores duplicados para un índice); si toma valor OFF no tendremos limitación de valores repetidos para un índice.

1.2 Acceso a tabla "syssynonyms"

Otra incompatibilidad es la de acceso por sinónimos. Ahora, el motor CTSQL accede a la tabla de sinónimos (syssynonyms) por el índice "synname", formado por los campos "synname" y "owner". En bases de datos antiguas, este índice no existe, por lo que mostrará el error "La Tabla (<tabname>) no existe en la base de datos". La manera de arreglarlo es creando el índice en cuestión:

```
create unique index synname on syssynonyms (synname, owner)
```

2 Nuevas funcionalidades

- **CTSQL como servicio:** Nuevo parámetro del CTSQL ("-port nnn") que ofrece la posibilidad de ejecutar el "ctsql.exe" como un servicio individual sin necesidad de instalar MultiWay en una máquina NT.

  Esta funcionalidad es útil para aplicaciones ligeras que se ejecuten sobre una máquina bajo Windows 9x. Por ejemplo:

  ```
  ctsql system 3.2 c:\multiway\etc\ctsql.ini NET -port 20010
  ```

- **No User validation:** Nueva parámetro del CTSQL ("-NP") para que no valide usuarios al conectarse a bases de datos. Por ejemplo:

  ```
  ctsql system 3.2 c:\multiway\etc\ctsql.ini NET -port 20010 -NP
  ```

- **Se permiten más ficheros abiertos.**
- Mecanismos de "log": **TABLOG** y **OPERLOG**. (Ver más adelante).

3 Errores corregidos

- Fichero de "collation". Se puede indicar tanto en el "path" absoluto como en el relativo.
- "Alter table" y "trepidx" modificaban grupo y propietario del fichero de la tabla. Ahora lo dejan tal como estaba.

  Cuando se producía una modificación de los ficheros de una tabla se asignaba como grupo el principal del usuario que hacía la modificación, en lugar de respetar el que tuvieran los ficheros (que podría ser el grupo secundario del usuario).

- "Rename table" no funcionaba bien cuando la tabla se había creado con la cláusula IN. En el campo "dirpath" de "systables" no se incluía el directorio.
- "Drop synonym". No lo borraba de la caché de la sesión SQL activa. No era efectivo para esa sesión hasta que se desconectaba y volvía a conectarse.
- Se ha corregido el error que permitía insertar un valor nulo en una columna de tipo fecha definida como "not null".
- Se ha corregido el error por el que una tabla interna temporal que era la salida de una "select" permitía que los campos fueran "not null". Esto provocaba errores de ejecución.

4 Mecanismo de “TABLOG”

Se le ha añadido al CTSQL una nueva funcionalidad que permite mantener un registro, o "log", de operaciones de inserción, borrado y modificación sobre tablas. Los datos relativos a esas operaciones se almacenan en tablas de la base de datos que deben ser creadas por el usuario exclusivamente para este fin.

Así pues, por cada tabla de la base de datos cuyas operaciones se deseen registrar deberá existir otra tabla en la base de datos en la cual se almacenará información sobre dichas operaciones. Nos referiremos a esta segunda tabla como tabla de movimientos.

4.1 Tablas de movimientos

Cada tabla de movimientos almacena información sobre una tabla concreta de la base de datos de la cual se desea llevar un registro de operaciones realizadas. Hay información que se guarda siempre e información opcional que el usuario puede configurar para que se guarde o no.

Una tabla de movimientos debe ser creada con la misma estructura de campos de la tabla a analizar, más una serie de campos por delante, que han de ser del siguiente tipo:

- **char(1):** Indicará el código de la operación que se ha realizado sobre la tabla a analizar. Sus posibles valores son:
  - "I": Se ha efectuado un "insert".
  - "U": Se ha efectuado un "update".
  - "D": Se ha efectuado un "delete".
- **serial:** Almacenará un número consecutivo.
- **date:** Almacenará la fecha en que se llevó a cabo la operación sobre la tabla a analizar.
- **time:** Almacenará la hora en que se llevó a cabo la operación sobre la tabla a analizar.
- **char(16):** Almacenará el usuario que realizó la operación sobre la tabla a analizar.

Los dos primeros campos son obligatorios, y además deben aparecer en ese orden por delante de los campos de la tabla a analizar. Los tres restantes son opcionales, pudiendo el usuario prescindir de ellos si lo desea.

El hecho de almacenar un "serial" en la tabla de movimientos responde a la necesidad de poder tener una clave en dicha tabla que identifique unívocamente cada una de sus filas, para que se pueda distinguir el orden en que se hicieron las operaciones sobre la tabla analizada. No obstante, esto hace que si en la tabla analizada tenemos un campo "serial", deberemos sustituirlo en la de movimientos por un "integer", ya que en MultiBase una tabla no puede tener dos campos de tipo "serial".

4.2 Configuración

Los pasos a seguir para activar este mecanismo son los siguientes:

1. Se utiliza un fichero de configuración que indicará para qué tabla o tablas de la base de datos se ha de utilizar este mecanismo (es decir, qué tablas se desea analizar). Este fichero, llamado "systablog", es un fichero de texto que deberá estar situado en el directorio donde se encuentran todos los ficheros de la base de datos (el catálogo y las tablas creadas por el usuario), y habrá de tener permiso de lectura para todos los usuarios.

La estructura de este fichero consta de 3 campos separados por espacios en blanco:

- El primer campo indica la tabla a analizar.
- El segundo indica el nombre de la tabla donde quedarán reflejadas las operaciones efectuadas sobre la primera.
- El tercero es una especificación de cuál de los campos opcionales de la tabla de movimientos se incluyen en ésta. Cada uno de esos campos se especifica mediante una letra: 'D'=fecha, 'T'=hora, 'U'=usuario. Así pues, este campo estará formado por la combinación de estas tres letras.

Los dos primeros campos son obligatorios, pero no el tercero, que puede no aparecer, en cuyo caso, y como opción por defecto, se asume que la tabla de movimientos incluye el campo de tipo fecha. Esto se ha hecho así por compatibilidad con una versión anterior de este mecanismo, en la cual no existían campos opcionales en la tabla de movimientos.

Veamos un ejemplo:

Supongamos que la base de datos se llama "test", que el DBPATH es "/home/datos" y que queremos activar este mecanismo para las tablas "clientes", "provincias" y "unidades" utilizando como tablas de movimientos "climov", "provmov" y "unimov", respectivamente. El fichero "systablog" deberá estar ubicado en "/home/datos/test.dbs" y sus entradas podrían ser así:

clientes	climov	DTU
provincias	provmov	UD
unidades	unimov	

Con esta configuración, la tabla de movimientos para clientes guardaría información acerca de la fecha, la hora y el usuario que ha realizado la operación; la tabla de movimientos para provincias guardaría información sobre el usuario y la fecha, y la tabla de movimientos para unidades guardaría información sobre la fecha (la información por defecto). Además, todas ellas guardarían el código de la operación, un "serial" y el registro sobre el que se ha actuado.

2. Las tablas de movimientos asociadas son tablas de la base de datos, **y han de ser creadas antes de activar este mecanismo.** La estructura de campos de una tabla de movimientos será la misma que la de la tabla a la que se refiere, pero por delante llevará los campos obligatorios que se han descrito anteriormente, y tras éstos, los opcionales, **siempre en el mismo orden en que aparecen en la especificación de "systablog".** Es decir, para el "systablog" del ejemplo anterior, si la estructura de las tablas "clientes", "provincias" y "unidades" es esta:

```
create table clientes
   (cli smallint, nom char(20), prov char(40))
create table provincias
   (codigo smallint, nombre char(20))
create table unidades
   (unidad serial, descripcion char(10))
```

Entonces la estructura de las tablas de movimientos habría de ser la siguiente:

```
create table climov
   (op char(1),
   ser serial,
   fecha date,
```

```
    hora time,
    usuario char(16),
    cli smallint, nom char(20), prov char(40))

create table provmov
    (op char(1),
    ser serial,
    usuario char(16),
    fecha date,
    codigo smallint, nombre char(20))

create table unimov
    (op char(1),
    ser serial,
    fecha date,
    unidad integer, descripcion char(10))
```

El nombre de los campos de la tabla de movimientos puede ser cualquiera, con tal de que se respeten los tipos de datos.

Tal y como se ha indicado, si la tabla que se va a analizar posee algún campo de tipo "serial", en la de movimientos éste tiene que sustituirse por uno de tipo "integer". Esto lo vemos ilustrado en el ejemplo, para el caso de la tabla "unimov".

Nótese también el comportamiento por defecto para el caso de no indicar un tercer campo en "systablog": en la tabla "unimov" incluimos un campo de tipo "date".

3. Por último, después de haber seguido los pasos anteriores de configuración, para poder activar el mecanismo habrá que definir una variable de entorno llamada MBTABLOG.

Esta variable de entorno indicará si queremos activar o no este mecanismo. Los valores que admite son ON y OFF; si queremos que se active habrá que ponerla a ON. Una vez hecho esto, y si se han seguido los pasos de configuración, el mecanismo quedará activado. En caso de que "systablog" esté mal construido, o de que no exista alguna de las tablas de movimientos, el mecanismo no se activará.

5 Mecanismo de “OPERLOG”

Este mecanismo es una versión alternativa a la gestión automática de auditoría sobre tablas con un enfoque más optimizado.

Con este nuevo mecanismo se mantendrá un registro, o "log", de operaciones de inserción, borrado y modificación sobre tablas, pero almacenando únicamente un registro de operación por cada registro modificado de una misma tabla. Es decir, si se añade un registro nuevo y posteriormente se modifica varias veces, en la tabla de operaciones sólo aparecerá un registro con el último tipo de operación relevante y la clave primaria del registro para poder obtenerlo directamente de la tabla original.

Los datos relativos a esas operaciones se almacenarán en una única tabla de la base de datos, la cual ha de ser creada por el usuario exclusivamente para este fin. A esta tabla la denominamos tabla de operaciones.

5.1 Tablas de operaciones

Las tablas de operaciones almacenan información de las operaciones realizadas sobre ciertas tablas de la base de datos. Una tabla de operaciones debe ser creada con la siguiente estructura:

- **char(1):** Indicará el código de la operación que se ha realizado sobre el registro. Sus posibles valores son:
  - "I": Se ha efectuado un "insert".
  - "U": Se ha efectuado un "update".
  - "D": Se ha efectuado un "delete".
- **date:** Almacenará la fecha en que se llevó a cabo la última operación sobre el registro.
- **time:** Almacenará la hora en que se llevó a cabo la última operación sobre el registro.
- **char(SQLFNAME):** Almacenará el nombre de la tabla a la que pertenece el registro.
- **integer:** Número de campos clave significativos.
- **char ((120-SQLFNAME)/8):** Valor del 1º campo clave del registro.
- **char ((120-SQLFNAME)/8):** Valor del 2º campo clave del registro.
- **char ((120-SQLFNAME)/8):** Valor del 3º campo clave del registro.
- **char ((120-SQLFNAME)/8):** Valor del 4º campo clave del registro.
- **char ((120-SQLFNAME)/8):** Valor del 5º campo clave del registro.
- **char ((120-SQLFNAME)/8):** Valor del 6º campo clave del registro.
- **char ((120-SQLFNAME)/8):** Valor del 7º campo clave del registro.
- **char ((120-SQLFNAME)/8):** Valor del 8º campo clave del registro.

NOTA: En realidad se espera que las tablas no tengan nunca más de 7 campos clave, porque la clave de la tabla de operaciones incluye el nombre de la tabla en su clave primaria y no puede haber más de 8 campos en un índice. Así mismo, se ha ajustado el tamaño de los campos para que la clave no exceda de 120 caracteres.

La sentencia SQL para la creación de la tabla sería:

```
Create table sysoperlog(
  change_code  char(1),
  change_dt    date,
  change_tm    time,
```

```
  name          char(18) not null default "",
  nkeys         smallint,
  key1_str      char(14) not null default "",
  key2_str      char(14) not null default "",
  key3_str      char(14) not null default "",
  key4_str      char(14) not null default "",
  key5_str      char(14) not null default "",
  key6_str      char(14) not null default "",
  key7_str      char(14) not null default "",
  key8_str      char(14) not null default "")
primary key
(
  name, key1_str, key2_str, key3_str, key4_str,
  key5_str, key6_str, key7_str
);
```

NOTA: La columna "key8_str" no se podrá añadir al índice, así que se puede o no crear.

Esta tabla almacenará únicamente un registro por cada registro modificado y la clave del registro en la tabla destino. Esto significa que no habrá una secuencia de las modificaciones realizadas sobre el mismo registro, sino tan sólo la última operación. Por eso, al registrar una operación en la tabla de operaciones habrá que tener en cuenta los siguientes casos:

Última operación	Nueva operación	Operación resultante o mensaje de error
D	U	ERROR - El registro se borró con anterioridad
U	I	I
U	U	U
D	I	U
I	I	ERROR - El registro ya existe
U	I	ERROR - El registro ya existe
D	D	ERROR - El registro se borró con anterioridad
I	D	D
U	D	D

5.2 Configuración

Los pasos a seguir para activar este mecanismo son los siguientes:

1. Se utiliza un fichero de configuración que indica para qué tabla o tablas de la base de datos se ha de utilizar este mecanismo (es decir, sobre qué tablas se desean registrar las operaciones). Este fichero, llamado "sysoper-log", es un fichero de texto que deberá estar situado en el directorio donde se encuentran todos los ficheros de

la base de datos (el catálogo y las tablas creadas por el usuario), y habrá de tener permiso de lectura para todos los usuarios. Contendrá los nombres de las tablas sobre las que se va ha guardar el registro de operaciones.

Veamos un ejemplo:

Supongamos que la base de datos se llama "test", que el DBPATH es "/home/datos" y que queremos activar este mecanismo para las tablas "clientes", "provincias" y "unidades". El fichero "sysoperlog" debería estar ubicado en "/home/datos/test.dbs" y contener lo siguiente:

```
Clientes
Provincias
Unidades
```

2. La tabla de operaciones es una tabla de la base de datos, y ha de ser creada antes de activar este mecanismo.

3. Por último, después de haber seguido los pasos anteriores de configuración, para poder activar el mecanismo habrá que definir una variable de entorno llamada MBOPERLOG.

Esta variable de entorno indicará si queremos activar o no este mecanismo. Los valores que admite son ON y OFF; si queremos que se active habrá que ponerla a ON. Una vez hecho esto, y si se han seguido los pasos de configuración, el mecanismo quedará activado. En caso de que "sysoperlog" esté mal construido, o de que no exista alguna de las tablas de movimientos, el mecanismo no se activará.

Si se produce algún error en la actualización de la tabla de operaciones, éste quedará reflejado en un fichero de texto, llamado "sysoperlog.err", en el directorio de la base de datos. Este fichero se abre siempre para añadir, de forma que habrá que borrarlo externamente cuando se crea oportuno.