

Notas Técnicas Versión 3.0



MultiBase/TransTOOLS
Notas Técnicas Versión 3.0
«Upgrade» 00 y 1.0

TransTOOLS, S.A.
<http://www.transtools.com>

© Copyright 1995. Reservados todos los derechos. TransTOOLS, S.A.

LA INFORMACIÓN CONTENIDA EN ESTE MANUAL ESTÁ SUJETA A CAMBIOS SIN PREVIO AVISO.
EN NUEVAS EDICIONES DE ESTE DOCUMENTO SE IRÁN INCORPORANDO DICHOS CAMBIOS

© Ninguna parte de este Manual puede ser reproducida ni transmitida por medio alguno sin autorización previa por escrito del titular del Copyright. MultiBase es marca registrada de TransTOOLS, S.A. Todos los productos citados en el presente volumen son marcas registradas de sus respectivos propietarios.

Indice General

1.	MultiBase 3.0.00: Mejoras Incorporadas.	4
1.1.	EasyReport.	4
1.2.	Editor Wedit.	4
1.3.	Comando ctlinf.	5
2.	Nuevas Funcionalidades y Sugerencias Implementadas.	6
2.1.	Lenguaje CTL.	6
2.2.	Gestor de Base de Datos (CTSOL).	10
2.3.	Enlace de Programas (Comando ctlink).	11
2.4.	Depurador («Debugger»).	11
2.5.	Entorno de Desarrollo (TRANS).	12
2.6.	Transacciones.	12
2.7.	EasyReport (MS-DOS y UNIX).	12
2.8.	Documentador Automático (Tdocu).	12
3.	Gateways.	13
3.1.	Gateway Oracle.	13
3.2.	Gateway Informix.	15
3.3.	Gateway Ingres.	16
4.	MultiBase 3.0.1.0: Mejoras Incorporadas.	19
4.1.	Nuevas Funciones Internas del CTL.	20
4.2.	EasyReport en Windows.	21
	4.2.1. Nuevas Características de EasyReport Comunes a UNIX, MS-DOS y Windows.	22
4.3.	Entorno de Desarrollo (TRANS).	23
4.4.	Actualización de Versiones Serializadas (UNIX).	23

1. MultiBase 3.0.00: Mejoras Incorporadas.

La versión 3.0 de MultiBase añade nuevas funcionalidades y prestaciones al producto. Especialmente significativas resultan las mejoras introducidas en la versión para Windows, que ahora incorpora el generador de informes EasyReport (disponible ya en UNIX y MS-DOS desde la versión 2.0.05), un nuevo editor «Wedit» y el comando `ctlinf`.

1.1. EasyReport.

Esta herramienta emplea la documentación interactiva dentro del propio producto, utilizando las opciones de ayuda («HELP») disponibles en cada programa. Básicamente, EasyReport está compuesto por los tres elementos siguientes:

- «**cdsedit.exe**»: Equivalente al **mbeasy** del EasyReport de las versiones para UNIX y MS-DOS. Este comando genera un ECD (Esquema Conceptual de Datos) a partir de una base de datos.
- «**confedit.exe**»: Este comando permite la definición de algunos parámetros de configuración del ECD, así como los distintos tipos de formato que podrá utilizar el usuario en ejecución para crear sus informes. Equivale a configurar la sección «.CONFIGURATION» del ECD.
- «**easyrep.exe**»: Equivalente al comando **trw** del EasyReport de las versiones de UNIX y MS-DOS, este comando permite crear diferentes tipos de listados, dependiendo de la información contenida en un ECD generado previamente con el comando **cdsedit.exe**.

1.2. Editor Wedit.

Nuevo editor que permite manejar ficheros de mayor tamaño que el incluido en versiones anteriores. Esta ampliación afecta tanto al número de caracteres por línea como al total de líneas. Estos límites se sitúan ahora aproximadamente en 20.000 caracteres por línea y 8.000 líneas de fichero.

Asimismo, este nuevo editor emplea diferentes colores para determinar palabras reservadas (negro), variables de usuario e internas (rojo), literales (azul), comentarios (verde), etc.

El comando encargado de la ejecución del editor es **wedit**, cuya sintaxis es la siguiente:

```
wedit [-v] | [-m] fichero [-l secuencia_a_buscar] [-x extensión]
```

Donde:

- v Cláusula opcional que indica la versión del comando, así como su «upgrade».
- m Cláusula opcional que ejecuta el editor maximizado.

fichero	Parámetro que indica el nombre del fichero que se desea crear o editar.
-l secuencia_a_buscar	Cláusula opcional que permite acceder a un texto en un punto determinado por una secuencia de caracteres («secuencia_a_buscar») indicada por el usuario, de forma que el cursor aparezca posicionado en el primer carácter de la primera secuencia encontrada. Esta opción sólo tiene sentido para acceder a ficheros ya existentes, y es especialmente útil en documentos muy extensos.
-x extensión	Esta opción permite indicar la extensión que tendrán los ficheros generados.

Nota: Las cláusulas «-m», «-l» y «-x» pueden indicarse indistintamente antes o después del nombre del «fichero».

1.3. Comando **ctlinf**.

Este comando —que ya existía en la versión 2.0.05 para UNIX y MS-DOS—, devuelve información sobre los programas y módulos que componen una aplicación realizada con MultiBase. Para más información consulte el apartado 12.1.3 del Manual del Administrador de MultiBase.

2. Nuevas Funcionalidades y Sugerencias Implementadas.

A continuación se indican las nuevas funcionalidades y mejoras implementadas en el producto respecto a la versión anterior en cada sistema operativo. Esta información se ha estructurado en los siguientes apartados:

- Lenguaje CTL.
- Gestor de base de datos (CTS QL).
- Enlace de programas (comando c tlink).
- Depurador («debugger»).
- Entorno de Desarrollo.
- Transacciones.
- EasyReport (UNIX y MS-DOS).
- Documentador.
- «Gateways».

Cuando alguna sugerencia o nueva funcionalidad es específica de un entorno concreto (UNIX, MS-DOS, Windows o red local), éste se indica entre corchetes antes del comentario correspondiente.

IMPORTANTE

El SQL y los «gateways» de la versión 3.0 son incompatibles con el CTL de versiones anteriores de MultiBase y viceversa. Por lo tanto, en instalaciones cliente-servidor ambas máquinas (cliente y servidor) tienen que disponer de la versión 3.0 del CTL y del CTS QL.

Todos los desarrollos realizados con versiones anteriores de MultiBase son compatibles con esta nueva versión.

2.1. Lenguaje CTL.

1. [Red local en MS-DOS y Windows]. En entornos de red local sobre MS-DOS y Windows se ha evitado el bloqueo de las máquinas que se originaba al emplear un FORM y bloquear una fila mediante la instrucción MODIFY (operación de actualización) cuando desde un nodo de la red se recorría sobre dicha fila un CURSOR definido con la cláusula «FOR UPDATE». Y lo mismo sucedía cuando se empleaba un CURSOR con la cláusula «FOR UPDATE» desde ambos nodos.
2. [UNIX]. Cuando se definía un CURSOR «FOR UPDATE» sin emplear ningún índice para construir la tabla derivada (acceso secuencial), se originaba el siguiente bloqueo al acceder desde distintos puestos de trabajo: Si el primer puesto que ejecutaba el programa leía un número de filas físicamente anteriores en la tabla a las que debería leer el segundo puesto, éste se bloqueaba hasta que el primero cerraba el CURSOR.

Por ejemplo: Valores de la tabla a actualizar mediante un CURSOR «FOR UPDATE»:

Grupo	Descripción
1	Valor1
1	Valor2
1	Valor3
2	Valor4
2	Valor5
2	Valor6

El primer puesto de trabajo leería las filas correspondientes al grupo «1»; si el segundo puesto leyese las correspondientes al grupo «2», éste quedaría bloqueado hasta que el primero cerrase el CURSOR después de leer la fila «1-Valor3».

3. En un mantenimiento (FORM) del tipo cabecera-líneas con ordenación de filas (cláusula «ORDER BY» en la sección JOINS), el programa se ralentizaba si existían muchas filas de detalle para una cabecera al tener que mostrar siempre la primera fila que cumpliera las condiciones de enlace. Este problema es debido al cambio del optimizador llevado a cabo en las versiones 2.0.xx de MultiBase. Para resolverlo basta con añadir en la cláusula «ORDER BY» de la sección JOINS las columnas de enlace entre ambas tablas.

Ejemplo:

```
f1 = cabecera.c1 = lineas.c1 required;
f2 = cabecera.c2 = lineas.c2 required;

f3 = lineas.linea noentry noupdate;

joins
  lineas lines of cabecera order by c1, c2, linea
  composites <cabecera.c1, cabecera.c2> <lineas.c1, lineas.c2>
end joins
```

Si la tabla «lineas» tiene muchas filas para una «cabecera», indicar la sección JOINS según se indica en el ejemplo. De no hacerlo así, el programa funcionará igualmente, aunque de modo más lento.

4. Al leer un valor nulo («NULL») en un CURSOR, la variable de la cláusula «INTO» o «BY NAME» recogía blancos en lugar del valor nulo.
5. [Windows]. La instrucción WINDOW con la cláusula «AS FILE» no realizaba bien el «scroll» de la última página si ésta no se encontraba completamente llena, mostrándose parte de la información correspondiente a la página anterior.

6. La instrucción CANCEL dentro de la sección CONTROL de un FORM dejaba la columna componente de la clave primaria de la tabla a mantener con valor nulo («NULL»). Por ejemplo:

```
database almacen
define
  form
  tables
    provincias
  end tables

  control
    before delete of provincias
      select "1" from clientes
      where clientes.provincia = $provincias.provincia
      if found = true then cancel
    end control
  end form
end define

main begin
  menu form
end main
```

En el ejemplo anterior, si el operador eligiese la opción de borrado de la provincia en curso, y en ésta hubiese clientes, el programa cancelaba la operación de borrado, dejando la variable «provincias.provincia» a nulo. Si inmediatamente después se volvía a realizar la misma operación, se producía el borrado indebido de la provincia al no existir integridad referencial entre las tablas «clientes» y «provincias».

7. La instrucción WHENEVER con la cláusula «INTERRUPT» no se ejecutaba la segunda vez que se pulsaba la tecla de «break» del sistema operativo.
8. La condición «MATCHES» en una instrucción embebida en un módulo CTL no empleaba el índice definido para la columna, por lo que el acceso era bastante lento, mientras que sí lo hacía desde CTSQL. El problema era debido al análisis de la variable «host» empleada como condición.
9. [UNIX]. Al realizar una consulta sobre una columna de tipo «char» con los metacaracteres de la cláusula «MATCHES» en el mantenimiento de una tabla en un FORM, no se devolvía las mismas filas que si se ejecutaba a través del CTSQL la instrucción SELECT con las mismas condiciones.
10. [Windows]. Al ejecutar la instrucción «START OUTPUT STREAM...» (con cláusula «APPEND») sobre un fichero existente que no tuviese permiso de lectura/escritura para el usuario que estuviese ejecutando el programa se producía el bloqueo de la máquina.
11. [Windows]. Al ejecutar la instrucción «START OUTPUT STREAM...» sobre un fichero cuyo nombre estuviese incluido en una variable de tipo «char» de gran longitud (por ejemplo 500), si no se indicaba la cláusula «CLIPPED» se producía un error de protección general en Windows al intentar generar un fichero con un nombre compuesto por 500 caracteres. Actualmente, en este caso se produce un «CLIPPED» automático en este tipo de variables.

12. La variable interna «sqlrows» se activa con el número de filas leídas en una instrucción SELECT con la cláusula «INTO TEMP», tanto si se encuentra preparada como si no (instrucciones PREPARE y EXECUTE). Por el contrario, dicha variable no se activará cuando dicha instrucción SELECT se encuentre en una TSQL.
13. [Windows]. El primer icono de un menú «Lotus» desaparecía al ejecutar por segunda vez consecutiva un mantenimiento sobre una tabla en un FORM.
14. Las instrucciones BREAK y EXIT PROGRAM, dependiendo del lugar donde se incluyesen, no liberaban totalmente la memoria reservada.
15. En instalaciones cliente-servidor, la línea a insertar en el fichero «/etc/inetd.conf» de la máquina servidor, relativa al gestor de base de datos a emplear, debe indicar necesariamente la versión 3.0.

Por ejemplo:

— En CTSQL:

```
ctsql stream tcp nowait root $TRANSDIR/lib/ctsql ctsql system 3.0 0.0 NET
```

— En Informix:

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

— En Oracle:

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

— En Ingres:

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

16. Al ejecutar repetidamente la instrucción INPUT FRAME con la cláusula «FOR UPDATE», si el FRAME incluía el atributo LOOKUP en alguna de sus variables, el cruce no se borraba de la pantalla.
17. [Error de Documentación]. La cláusula «LABEL» de la instrucción PROMPT FOR puede ser también una expresión.

2.2. Gestor de Base de Datos (CTSQL).

1. Si se empleaba una función interna de CTSQL (en una cláusula «WHERE») relativa al tipo de dato «date» no se seleccionaba ninguna fila, siempre y cuando la columna «fecha» empleada como parámetro de la función fuera parte de un índice. Por ejemplo, el siguiente ejemplo no seleccionaba ninguna fila:

```
create table tabla (campo1 char(11) not null,
  fecha date not null,
  descripcion char(20) upshift)
primary key (articulo, fecha);

select * from tabla
  where tabla.articulo = "valor"
     and year(fecha) between 1990 and 1995
```

2. Las siguientes instrucciones SELECT no devolvían la tabla derivada correcta:

```
select * from provincias
  where provincia not between 1 and 4
     and provincia not between 5 and 7

select a,b from tabla
  where a not in ("S","N") or b not in ("S","N")
```

3. La creación de una base de datos con la cláusula «COLLATING» determinaba que ciertas condiciones (de la cláusula «WHERE») no devolviesen las filas correctas.
4. [HP-UX]. La lectura de una tabla en la que se condicionaba una columna de tipo «DECIMAL» o «MONEY» no devolvía la tabla derivada correcta.
5. La negación de una condición «IS NOT NULL» de CTSQL no seleccionaba las filas correctas. Por ejemplo:

```
select * from provincias
  where not (prefijo is not null)
```

6. Para una correcta optimización mediante el empleo del «ROWID», éste debe encontrarse a la izquierda de una condición. Por ejemplo:

```
select * from provincias
  where rowid = 20
```

Por el contrario, si se hubiese indicado:

```
select * from provincias
  where 20 = rowid
```

No se optimizaría correctamente.

7. La integridad referencial entre dos tablas no se controlaba al actualizar (instrucción UPDATE) parte de la columna componente (substring) de la clave primaria de la tabla principal. Por ejemplo:

```
update provincias set col_primary[2,3] = "ab"
where ...
```

8. [SOLARIS]. Al ejecutar una instrucción SELECT que leía de una tabla temporal (cláusula «INTO TEMP») se producía un error de interrupción en la comunicación con el CTSQL.
9. [SOLARIS]. Al emplear la instrucción EDIT de una variable no se presentaba su contenido.
10. [UNIX]. La lectura del «ROWID» de una tabla con una condición «MATCHES» sólo devolvía la primera fila encontrada como válida. Sin embargo, si se leía alguna columna perteneciente a la tabla, sí devolvía la tabla derivada correcta. Este problema se detectaba entre la ejecución de la instrucción SELECT con la condición «MATCHES» y la consulta desde un FORM con la misma condición. Por ejemplo:

```
select * from articulos
where descripcion matches "M*e*"
```

En la base de datos de demostración incluida en su copia de MultiBase, esta instrucción SELECT devuelve varias filas. Si en lugar del asterisco seleccionábamos el «ROWID» sólo devolvía la primera fila que cumpliera la condición. Asimismo, si se ejecutaba el mantenimiento de «articulos» desde el FORM y se introducía la condición «M*e*» en el campo «descripcion» sólo se seleccionaba una fila.

11. Hacer un DELETE cuando existe un índice duplicado con muchas duplicidades resulta extremadamente lenta. Para agilizar esta operación hay que cambiar el índice duplicado por uno único compuesto por la columna o columnas duplicadas del índice inicial más la columna o columnas componentes de la clave primaria.

2.3. Enlace de Programas (Comando `ctlink`).

La creación de programas en UNIX y MS-DOS a partir de módulos fuente de CTL generados por MultiBase para Windows en los que se utilicen funciones específicas de este entorno, no provocará ningún error, asumiendo que dichas funciones también son internas de aquellos sistemas.

2.4. Depurador («Debugger»).

A la hora de consultar el valor de una variable desde la línea de comando mediante «?variable», si aquella superaba los 80 caracteres no se mostraba su valor completo. Actualmente dicho valor se muestra en tantas líneas como sea necesario.

2.5. Entorno de Desarrollo (TRANS).

Al activar la opción «Ventana de BD» de la persiana «Entorno» se eliminan de la pantalla las ventanas de información relativas a la base de datos en curso y a las teclas de ayuda, información y selección.

Por su parte, en instalaciones cliente-servidor, al ejecutar el Entorno de Desarrollo se solicita el nombre del usuario («DBUSER») con el que se conecta a la máquina servidor, así como su clave de acceso («DBPASSWD»).

Por último, cuando un módulo se define con la opción «Depuración», todas las compilaciones se realizarán con «debugger» hasta que se especifique lo contrario.

2.6. Transacciones.

La instrucción ROLLFORWARD producía el error «Fallo en rollforward» al intentar recuperar la ejecución provocada por la instrucción DELETE del SQL.

Todos los «cursores» definidos con la cláusula «FOR UPDATE» que se encuentren afectados por una transacción («BEGIN WORK») serán siempre «NOWAIT», teniendo que controlar el bloqueo de una fila mediante la variable interna «locked».

2.7. EasyReport (MS-DOS y UNIX).

Cuando se intentaba generar un listado que empleaba parámetros en su ejecución se producía un error no recuperable al intentar abrirlo.

2.8. Documentador Automático (Tdocu).

El documentador automático de MultiBase (Tdocu) producía un error no recuperable en las versiones para MS-DOS y Windows al generar el manual de usuario de un programa en el que se definía el atributo LOOKUP en una variable de un FRAME.

3. Gateways.

A partir de esta versión se ha implementado una nueva función interna, denominada «getsqlerror()», que devuelve el número de error («errno») producido por el gestor de base de datos que se esté empleando.

3.1. Gateway Oracle.

Variables de configuración: Se ha añadido la siguiente variable

ORACLE_PROC Indica al «gateway» si el gestor de Oracle permite la creación de procedimientos SQL. Por defecto es ON.

Variables de entorno: Las variables ORACLE_SID, ORACLE_HOME y ORACLE_UID funcionan como tal en entornos cliente-servidor. De esta manera puede obviarse el valor general que se les haya asignado en el fichero de configuración «gworacle.env», particularizándolo para cada cliente.

Foreign Keys: Al nombre de una «foreign key» ya no se le antepone el prefijo FK. (Si se quiere borrar una «foreign key» de antiguas versiones hay que destruirla directamente a través del gestor de Oracle y también, en modo mantenimiento, a través de MultiBase).

Instrucción INSERT: Se han eliminado todas las restricciones de funcionalidad que existían para esta instrucción y que aparecían cuando intervenía el tipo de dato SERIAL.

IMPORTANTE

La tabla interna «mbserial» utilizada para simular este tipo de dato ha cambiado de estructura. El programa CTL siguiente borra la tabla y la crea con el nuevo esquema, actualizando los datos que hubiera. Ha de ejecutarse tantas veces como bases de datos MultiBase existan:

```
define
  parameter[1] dbname char(20)
  parameter[2] user_pass char(40)
end define
main begin
  call putenv("ORACLE_UID", user_pass, 2)
  database $dbname
  tsql "execsql create table mbtserial(tabname char(30), lastserial integer)"
  tsql "execsql insert into mbtserial select tabname, lastserial '&&'
      "from mbserial,mbtables where mbserial.tabid = mbtables.tabid"
  tsql "execsql drop table mbserial"
  tsql "execsql rename mbtserial to mbserial"
end main
```

CASOS PRÁCTICOS:

IMPORTANTE

El campo «dirpath» en «mbtables» contiene el usuario donde se localiza la tabla. Si el campo tiene un valor nulo heredado de antiguas versiones, hay que rellenarlo como corresponda para evitar un funcionamiento erróneo.

Cambio de base de datos MultiBase: Se realiza con la instrucción a tal efecto:

```
DATABASE nombre_de_base_de_datos
```

pero haciendo previamente un «putenv» de la variable de entorno «ORACLE_UID» con el valor correspondiente para esa base de datos:

```
call putenv("ORACLE_UID", user_password, 2)
```

Tablas temporales: Si a lo largo de la ejecución se ha cambiado sucesivamente de base de datos puede que al finalizar el proceso se hayan creado tablas «temporales» en diferentes usuarios Oracle. El «gateway» las borrará naturalmente para simular esa temporalidad, pero haciendo dos supuestos a la hora de conectarse a los usuarios: que no se haya cambiado de ORACLE_SID y que la PASSWORD de esos usuarios coincida con su nombre. En cualquier otro caso quedarían tablas sin borrar.

Tablas compartidas: Para compartir una tabla entre dos bases de datos (dos usuarios Oracle) se sigue un camino similar al realizado frente al gestor de MultiBase.

El usuario A, creador de la tabla a compartir, da los permisos que considere oportunos al usuario B sobre la misma y sobre la tabla «mbserial» si existe algún campo cuyo tipo de dato es SERIAL.

El usuario B crea el sinónimo:

```
CREATE SYNONYM nombre_de_tabla FOR usuario_A.nombre_de_tabla
```

para poder utilizar el mismo nombre de tabla. Además, crea la tabla en modo mantenimiento para que su catálogo la contenga y, con una instrucción UPDATE modifica el campo «dirpath» de la fila que haya aparecido en «mbtables» para esa tabla, cambiando el valor «usuarioB» por «usuarioA» (es en A donde realmente se halla la tabla).

Si en algún momento se altera la estructura de la tabla, hay que repetir el procedimiento anterior (ALTER en modo mantenimiento y cambio de «dirpath»).

Catálogo de MultiBase: Es posible romper la relación «biunívoca» existente entre bases de datos MultiBase y usuarios Oracle.

Si se quiere hacer corresponder una base de datos MultiBase con varios usuarios Oracle manteniendo un solo catálogo, éstos tendrían que compartir las tablas «mb*» siguiendo el procedimiento anterior. Uno de los usuarios sería el creador real del catálogo («dador» de permisos) y los demás compartirían las tablas (creando sinónimos).

La tabla «mbserial» no aparece en el catálogo, dada su naturaleza interna, pero sí han de poder manipularla todos, para lo cual, además de los permisos:

```
EXECSQL GRANT ALL PRIVILEGES ON mbserial TO PUBLIC
```

cada uno de los usuarios, excepto el creador del catálogo, necesita el siguiente sinónimo:

```
EXECSQL CREATE SYNONYM mbserial FOR usuario_creador.mbserial
```

Este procedimiento podría seguirse también con las tablas del entorno de programación (tablas «ep*») si así se desea.

3.2. Gateway Informix.

Tipos de Datos: Las conversiones:

Informix		MultiBase
VARCHAR(size)	——>	CHAR(size)
FLOAT	——>	DECIMAL(p,s)
SMALLFLOAT	——>	DECIMAL(p,s)

no están especificadas en el manual, pero también se realizan automáticamente.

Variables de entorno: La variable INFORMIXDIR funciona como tal en un entorno cliente-servidor. De esta manera puede obviarse el valor general que se le haya asignado en el fichero de configuración «gwinformix.env», particularizándolo para cada cliente.

Instrucción CREATE. Atributo DEFAULT: Para la versión 4 del gestor de Informix este atributo no existe. A partir de ahora el «gateway» lo simula en las operaciones INSERT cuyos valores en VALUES se introduzcan a través de constantes o variables «hosts» (esto se hace extensivo al FORM).

La simulación no funciona cuando los valores se introducen a través de una instrucción SELECT, ya que no puede controlarse su resultado (INSERT ... SELECT...).

Foreign Keys: Al nombre de una «foreign key» ya no se le antepone el prefijo FK. (Si se quiere borrar una «foreign key» de antiguas versiones hay que destruirla directamente a través del gestor de Informix y también, en modo mantenimiento, a través de MultiBase).

Instrucción START DATABASE: Esta instrucción únicamente introduce en el catálogo de MultiBase el registro «syslog», indicando que la base de datos es transaccional. No envía ninguna orden al gestor de Informix.

Tablas temporales: No se pueden compartir tablas temporales entre bases de datos cuando se está utilizando este gestor.

3.3. Gateway Ingres.

Variables de configuración: Aparecen tres nuevas variables:

INGRES_UID	Indica al «gateway» el nombre del usuario Ingres para la conexión a la base de datos. Excepcionalmente, esta variable puede leerse del entorno (instalación en local), siendo éste prioritario. Por defecto será el usuario CTL.
INGRES_DBA	Indica al «gateway» el nombre del usuario Ingres que es DBA. Excepcionalmente, esta variable puede leerse del entorno (instalación en local), siendo éste prioritario.
MBLCKTIMEOUT	Permite establecer el tiempo de respuesta en segundos a las peticiones de bloqueo al servidor Ingres. Si en ese tiempo el bloqueo no se facilita, el «query» que lo solicitó finaliza. Con esta variable se simula el comportamiento de los «cursores» NOWAIT (en Ingres son todos WAIT), si bien de forma limitada. El tiempo va de un segundo (ON o 1) a «n». Por defecto es OFF (WAIT).

Variables de entorno: Las variables II_SYSTEM, INGRES_UID e INGRES_DBA funcionan como tal en entornos cliente-servidor. De esta manera puede obviarse el valor general que se les haya asignado en el fichero de configuración «gwingres.env», particularizándolo para cada cliente.

Instrucción CREATE. Atributo CHECK: El atributo CHECK funciona a partir de esta versión. Se traduce a una instrucción «CREATE INTEGRITY ON...» equivalente.

Instrucción INSERT: Se han eliminado todas las restricciones de funcionalidad que existían para esta instrucción y que aparecían cuando intervenía el tipo de dato SERIAL.

IMPORTANTE

La tabla interna «mbserial» utilizada para simular este tipo de dato ha cambiado de estructura. El programa CTL siguiente borra la tabla y la crea con el nuevo esquema, actualizando los datos que hubiera. Ha de ejecutarse tantas veces como bases de datos MultiBase existan:

```
define
  parameter[1] dbname char(20)
  parameter[2] user char(20)
end define
```

```

main begin
  call putenv("INGRES_UID", user, 2)
  database $dbname
  tsq! "execsql create table mbtserial(tabname char(30), lastserial integer)"
  tsq! "execsql insert into mbtserial select tabname, lastinteger "&&
      "from mbserial,mbtables where mbserial.tabid = mbtables.tabid"
  tsq! "execsql drop table mbserial"
  tsq! "execsql rename mbtserial to mbserial"
end main

```

CASOS PRÁCTICOS:

IMPORTANTE

El campo «dirpath» en «mbtables» contiene el usuario donde se localiza la tabla. Si el campo tiene un valor nulo heredado de antiguas versiones, hay que rellenarlo como corresponda para evitar un funcionamiento erróneo.

Cambio de base de datos MultiBase: Se realiza con la instrucción a tal efecto:

```
DATABASE nombre_base_de_datos
```

pero haciendo previamente un «putenv» de la variable de entorno «INGRES_UID» con el valor correspondiente para esa base de datos:

```
call putenv("INGRES_UID", user, 2)
```

Siempre que no se cambie de base de datos Ingres, el valor de la variable INGRES_DBA sigue siendo válido. De lo contrario debería hacerse otro «putenv» de la misma.

Tablas temporales: El «gateway» simula la funcionalidad que estas tablas tienen en MultiBase creando todas desde el usuario INGRES_DBA y dando permisos desde éste sobre ellas a todos los usuarios. Al finalizar el proceso las borrará naturalmente para simular esa temporalidad, conectándose a ese usuario INGRES_DBA. Esto se hace bajo el supuesto de que no se haya cambiado de base de datos Ingres. En cualquier otro caso quedarían tablas sin borrar.

Tablas compartidas: Para compartir una tabla entre dos o más bases de datos (usuarios Ingres) se deberá proceder como sigue:

Desde el gestor de Ingres ha de cambiarse el propietario de la tabla a compartir por el usuario INGRES_DBA, si es que no lo es ya. Éste ha de dar los permisos correspondientes a los que van a compartir la tabla.

Desde MultiBase, esa tabla ha de crearse en modo mantenimiento para los que no la tenían en su catálogo (todos excepto el propietario original). Y todos tienen que modificar con UPDATE el campo «dirpath» de la fila que haya en «mbtables» para la tabla, cambiando el valor «usuarioX» por el valor de INGRES_DBA (propietario real).

Si la tabla tiene algún campo cuyo tipo de dato es SERIAL tendrá que existir una tabla «mbserial» en el usuario INGRES_DBA y con permisos para los usuarios que comparten la tabla.

Si en algún momento se altera la estructura de la tabla (por el INGRES_DBA), hay que repetir el procedimiento anterior (ALTER en modo mantenimiento y cambio de «dirpath»). Y lo mismo sucede si se renombra la tabla (RENAME TABLE).

Catálogo de MultiBase: Es posible romper la relación «biunívoca» existente entre bases de datos MultiBase y usuarios Ingres. Se creará entonces una relación entre base de datos MultiBase y base de datos Ingres o varios usuarios Ingres.

Para ello se mantendrá un solo catálogo compartido entre todos los usuarios Ingres siguiendo un poco el procedimiento anterior, es decir, el propietario será el usuario INGRES_DBA, siendo éste quien dará permisos a todos los usuarios Ingres sobre las tablas del catálogo. La tabla «mbserial» no aparece nunca en el catálogo, pero sí han de poder manipularla todos, con lo que pertenecerá también al usuario INGRES_DBA y todos tendrán permiso.

Este procedimiento podría seguirse también con las tablas del entorno de programación (tablas «ep*») si así se desea.

4. MultiBase 3.0.1.0: Mejoras Incorporadas.

- El operador lógico «OR» implicado en la cláusula «WHERE» de una instrucción de SQL (SELECT, UPDATE o DELETE) no devolvía en ciertas circunstancias la tabla derivada correcta.
- La edición de un FRAME «multiscreen» (con varias páginas) producía un error, asignando valores ilegibles a las variables y cancelando posteriormente el programa.
- [MS-DOS]. El comando **trepidx** admite la cláusula «-y» para que no se realice ninguna pregunta al usuario. Esta cláusula ya se encontraba implementada en versiones anteriores para UNIX y Windows.
- A la hora de imprimir el manual generado por el comando **tdocu** de MultiBase, el salto de página no se realizaba correctamente. El problema residía en el comando **tprocess**, que es el que interpreta los comandos generados por la documentación de usuario o programador de MultiBase.
- La instrucción ROLLFORWARD DATABASE no funcionaba cuando en una transacción se empleaba la cláusula «WHERE CURRENT OF...» dentro de la lectura de un CURSOR declarado con la cláusula «FOR UPDATE».
- La instrucción WINDOW de una SELECT que no devolvía ninguna fila producía un error al encontrarse dentro de un bucle cuando alguna columna de la «lista_selección» era de tipo DECIMAL o MONEY.
- Una instrucción SELECT no declarada como CURSOR que incluyese una cláusula de recepción de valores («INTO» o «BY NAME») producía que el módulo CTL consumiese gran cantidad de memoria si devolvía más de una fila («ambiguous=true»), lo que originaba un error cuando dicha instrucción se ejecutaba varias veces.
- Al generar la lista en curso en un FORM, si se seleccionaba la opción de nueva consulta y se cancelaba, al recuperar la lista anterior la fila que quedaba en curso era siempre la última, produciéndose el mensaje «No hay más filas» cuando se intentaba seleccionar la siguiente. Sin embargo, si se seleccionaba la opción de modificación o borrado, la fila afectada por cualquiera de estas operaciones era la primera de la lista y no la última.
- La creación masiva de tablas temporales (CREATE TEMP TABLE) producía el mensaje «Imposible abrir la tabla de la base de datos...».
- La función «evalfun(función, parámetros)» no operaba correctamente con ciertas funciones internas de CTL.
- Si a una variable de tipo TIME se le asignaba una hora superior a las 24 horas devolvía como valor un cero «0». Actualmente, la expresión «let hora=24:01:10» devolverá el valor «NULL».

- La cláusula «DISTINCT» de la «lista_selección» de una instrucción SELECT con más de 8 campos devolvía resultados diferentes dependiendo de si se incluía o no la cláusula «INTO TEMP».
- El comando **thelpcomp** no emplea la variable MSGDIR, si bien el entorno de programación simula esta posibilidad. En la «release» anterior no se incluía dicha variable en la línea de comando del **thelpcomp**.
- [WINDOWS]. La selección de una segunda impresora mediante la función específica de Windows «setprtsetup("name", expresión)» no actualizaba el tamaño del papel a la hora de emplear el Administrador de Impresión («DBPRINT=PRINTMAN»).
- [WINDOWS]. En un mantenimiento (FORM) con «scroll» lateral (izquierda-derecha), al pulsar la combinación de teclas [CTRL]+[Q] para realizar el «scroll» hacia la izquierda se producía un error. Dicha combinación es la que MultiBase define por defecto para dicha acción.
- [UNIX y MS-DOS]. Se ha incorporado una nueva variable de entorno, WINFN, para que el compilador ignore o no las funciones específicas de Windows y que la ejecución del programa no produzca los mensajes de error típicos de que dicha función no existe en ningún módulo componente del programa. Los posibles valores de esta variable son «Y» y «N». El valor «Y» puede simularse mediante el parámetro «-wfn» del comando **ctlcomp**, en cuyo caso su sintaxis quedaría de la siguiente forma:

```
ctlcomp -wfn módulo
```
- En una base de datos definida con secuencia de ordenación, cuando se condicionaba una columna de tipo DECIMAL en la cláusula «WHERE» de una instrucción SELECT, UPDATE o DELETE, la tabla derivada que devolvía era correcta o incorrecta según fuese índice o no.
- [UNIX]. En versiones estándares 4.0 de UNIX se pueden definir grupos suplementarios. Esta versión de MultiBase contempla dichos grupos para los permisos relativos al UNIX sobre programas y bases de datos.

4.1. Nuevas Funciones Internas del CTL.

lastrowid()

Devuelve el número de «ROWID» de la última fila insertada mediante un FORM (instrucciones ADD, ADD ONE o INTERCALATE), o bien mediante la instrucción INSERT del sublenguaje DML del SQL.

lastserial()

Devuelve el número asignado a la columna SERIAL después de realizar una inserción mediante un FORM (instrucciones ADD, ADD ONE o INTERCALATE), o bien mediante la instrucción INSERT del sublenguaje DML del SQL.

licence()

Esta función, implementada en versiones anteriores, no funcionaba correctamente en la versión de Windows, devolviendo siempre el valor cero. Actualmente, el valor que devuelve es el de la Licencia sobre la que se ejecuta CTL.

DisableFormVar(expresión1,expresión2)**EnableFormVar(expresión1,expresión2)**

Desactiva/Activa la edición de la variable indicada en «expresión2» durante la ejecución de un FORM en el que se mantenga la tabla especificada en «expresión1». Ejemplo:

```
option "Altas"
  call DisableFormVar("clientes","total_facturado")
  add
  ...
```

DisableFrameVar(expresión1,expresión2)**EnableFrameVar(expresión1,expresión2)**

Desactiva/Activa la edición de la variable indicada en «expresión2» perteneciente al FRAME indicado en «expresión1».

CountFormVars(expresión1)

Devuelve el número de variables a editar en el FORM que mantenga la tabla especificada en «expresión1».

CountFrameVars(expresión1)

Devuelve el número de variables a editar pertenecientes al FRAME indicado en «expresión1».

GetFormVarName(expresión1,expresión2)

Devuelve el nombre de la variable de la tabla «expresión1» que se está manteniendo en un FORM y que corresponda con el número de orden indicado en «expresión2».

GetFrameVarName(expresión1,expresión2)

Devuelve el nombre de la variable del FRAME indicado en «expresión1» que se está manteniendo y que corresponda con el número de orden indicado en «expresión2».

4.2. EasyReport en Windows.

**Conexión dinámica del servidor de base de datos.**

Se ha implementado un nuevo parámetro en la línea de comando («-con») y una nueva opción en la persiana «Archivo» del menú principal de EasyReport («Cambiar Conexión») para seleccionar una conexión al servidor de la base de datos diferente de las previamente definidas en el Editor de Configuración («Confedit»). Las variables de entorno implicadas en este nuevo concepto de conexión dinámica que reciben un tratamiento especial son DBHOST, DBUSER y DBPASSWD.

Los nuevos parámetros de la línea de comando del EasyReport para el manejo del entorno son los siguientes:

-env	Selecciona un entorno definido en el fichero de configuración. El fichero de configuración por defecto es «COSMOS.INI», aunque puede generarse un fichero auxiliar, denominado «EASYREP.INI», cuyo nombre y localización pueden modificarse en la entrada «EASYREP.INI» de la sección [MultiBase] del fichero «WIN.INI».
-var	Define una variable de entorno con su correspondiente valor.
-con	Define un entorno de configuración.
-pass	Define la contraseña («password») para la conexión del DBUSER definido previamente.

En la sección [MultiBase] del fichero «WIN.INI» se pueden definir las entradas «COSMOSDIR» y «EASYREP.INI». «COSMOSDIR» identifica el directorio («path») de la aplicación para localizar los comandos propios de EasyReport (**easyrep**, **cdsedit** y **confedit**). Dicho directorio debe contener, al menos, los subdirectorios «bin», «drw», «msg» y «etc» con los ficheros propios del EasyReport. Por su parte, la entrada «EASYREP.INI» define el directorio y nombre («path completo») del fichero auxiliar de configuración de EasyReport. Este fichero tiene la misma estructura que el fichero «COSMOS.INI», sobre el cual tiene preferencia. Si no se especifica «EASYREP.INI» el fichero auxiliar de configuración estará definido como «COSMOSDIR\etc\easyrep.ini», no siendo obligatoria su existencia.

IMPORTANTE

EasyReport incluye un sistema de ayuda interactiva donde se explica su funcionamiento y características. Para cualquier aclaración sobre las nuevas posibilidades de la herramienta consulten este fichero.

4.2.1. Nuevas Características de EasyReport Comunes a UNIX, MS-DOS y Windows.

La línea de cabecera de un listado puede incluir caracteres especiales («-\$Pn») para mostrar los valores que se le asignan a los posibles parámetros de ejecución del listado.

El parámetro «-par» de la línea de comando del EasyReport sólo tendrá que especificar los parámetros diferentes que intervienen en la ejecución del listado.

La cláusula «USING» dentro de un Esquema Conceptual de Datos producía un error si no se indicaba, ya que no es obligatorio especificar un valor variable en la instrucción SELECT que se define.

4.3. Entorno de Desarrollo (TRANS).

- La opción «Descargar tablas» producía un error cuando se trabajaba con el «gateway» de Oracle.
- La opción de «Seleccionar» una base de datos no funcionaba al trabajar con «gateways».
- [Windows]. La opción «Salida» de la persiana «SQL» producía un error cuando la variable de entorno DBPRINT contenía el valor «PM» o «PRINTMAN» (empleo del administrador de impresión Windows).

4.4. Actualización de Versiones Serializadas (UNIX).

La actualización a la versión 3.0.1.0 de versiones 3.0.00 y 2.0.xx serializadas por software, es decir, por número de serialización facilitado por el Departamento de Soporte de TransTOOLS, deberá realizarse copiando la nueva versión sobre la antigua mediante el siguiente comando:

```
# cpio -iBduv -l /dev/r...
```

o bien:

```
# cpio -iBduv < /dev/r...
```

La cláusula «u» del comando **cpio** ordena la actualización incondicional de todos los ficheros de MultiBase.

IMPORTANTE

Si ha personalizado su copia de MultiBase en la versión 3.0.00, antes de proceder a la instalación de la nueva haga una copia de los ficheros «\$TRANSDIR/etc/termcap» y «\$TRANSDIR/etc/tprinter».

A continuación, y dependiendo de si la Licencia es de Desarrollo o Run Time, ejecute desde superusuario el comando **installctl** o **installrt**, respectivamente, con lo que la copia de MultiBase quedará actualizada automáticamente a la nueva versión.

La instalación de una Licencia nueva de la versión 3.0.1.0 (no una actualización) se realizará según lo explicado en el capítulo 3 del Manual del Administrador.

Si su Licencia de MultiBase es en «depósito» (por tiempo limitado), podrá ampliar la fecha de caducidad previa autorización del Departamento Comercial de TransTOOLS.



www.transtools.com