Firebird™ Version 1.5

Notas de la Versión 1.5

5 de Febrero de 2004 – Versión de documento 1.08

Contenidos

Introducción

El motor de bases de datos Firebird™ ha sido desarrollado por un equipo independiente de desarrolladores voluntarios a partir del código fuente de Interbase™ que fue liberado por Borland bajo la licencia pública InterBase Public License v.1.0 el 25 de Julio de 2000.

El desarrollo del código Firebird 2 arranca inicialmente en el desarrollo de Firebird 1, con el traspaso del código C de Firebird 1 a C++ y la primera gran limpieza en el código. Firebird 1.5 es la primera versión del código Firebird 2. Ello supone haber cubierto una etapa muy importante para los desarrolladores y el propio proyecto Firebird, pero no es un fin en sí mismo. Cubierta la etapa de la liberación de Firebird 1.5, el viaje hacia Firebird 2 prosigue con importantes modificaciones.

Firebird 1.0.x continúa manteniéndose de forma activa, con la corrección de errores y la inclusión de mejoras tomadas de la versión 1.5

Dónde encontrar Firebird 1.5

Firebird puede ser descargado desde la web de Firebird -
http://sourceforge.net/project/showfiles.php?group_id=9028

Identificadores de versión para las versiones de Firebird 1.5

Win32: "WI-V1.5.0.nnnn Firebird 1.5"
Linux: "LI-V1.5.0.nnnn Firebird 1.5"
etc., donde nnnn es el número de reconstrucción (build)

Por favor, diríjase a la sección de documentación para saber dónde encontrar la documentación recomendada.

Nuevas Características

Nuevo código, mejor optimización

Esta versión ha sido construida a partir del código portado del original en C a C++, un proceso iniciado por Mike Nordell en el año 2000. La limpieza total del código y la corrección de errores ha continuado, completada por un nuevo gestor de memoria y mejoras en el lenguaje. No menos importantes han sido los cambios experimentados por el optimizador de consultas SQL durante el proceso de desarrollo de la v. 1.5, con mejoras y correcciones de la mano de Arno Brinkman y otros, cuyo resultado ha sido una mejora en la velocidad de entre un 30 y un 60 por ciento como mínimo.

Arquitectura

Dos nuevas aportaciones significativas a las plataformas Windows son las versiones Classic server y embedded server.

- No ha existido la versión Classic Server para Windows en los últimos ocho años. Esta versión puede utilizar varios procesadores, algo que todavía elude la versión Superserver para Windows. Aunque se puede utilizar, la versión Classic Server debe ser considerada como experimental.

- Embedded server es una dll que combina una única conexión cliente con un servidor Firebird Superserver para construir veloces y eficientes aplicaciones autónomas y de maletín.

Nuevas e importantes características han sido añadidas al lenguaje desde la versión 1.0.x, incluyendo las expresiones del estándar SQL-92 CASE, COALESCE y NULLIF. Para conocer la sintaxis de estas y otras de las nuevas características implementadas, diríjase a la sección Mejoras en el lenguaje, de este mismo documento.

Módulos instalados y seguridad

Si usted ha estado utilizando Firebird 1.0.x hasta ahora, notará grandes cambios en los nombres de los módulos y las reglas para acceder y encontrarlos. A continuación los más destacables; para encontrar información detallada sobre la instalación, situación en disco y configuración, mire la sección correspondiente en este documento.

1. La mayor parte de los modulos y constantes han sido renombrados. En la mayoría de los casos, los nuevos nombres se articulan en torno a “firebird” o “fb”. Por ejemplo, la API se encuentra ahora en una librería llamada “fbclient.dll” en windows y “libfbclient.so” en el resto de plataformas. La

excepción que rompe esta regla es la base de datos de seguridad, anteriormente llamada "ISC4.gdb", que ahora se llama "security.fdb".
2. Los ficheros externos utilizados por el servidor (UDF's, filtros BLOB, librerías de conjuntos de caracteres, tablas externas) están ahora sujetos a niveles de seguridad, que en algunos casos, se colocan a un nivel que será diferente a lo que usted tenía bajo Firebird 1.0.x o Interbase.
3. El nuevo fichero de configuración del servidor, firebird.conf, que reemplaza a ibconfig (Windows) e isc_config (en el resto de plataformas) contiene algunas nuevas características configurables además de auto-documentación y una mejor organización.
4. La versión 1.5 incluye la posibilidad de definir alias para las bases de datos. Ahora es posible esconder la localización física de la base de datos, utilizando en el código de su aplicación un alias para sustituir a la ruta de la base de datos en la cadena de conexión. Las rutas físicas se almacenan en el fichero de texto aliases.conf. El propósito principal de los alias, no obstante, es proteger sus rutas físicas de ser leídas maliciosamente en la red.
5. Por defecto (y una práctica del pasado) en los servidores windows era el usuario del sistema el que ejecutaba el programa de instalación de Firebird y el que iniciaba el servicio Firebird al arrancar el sistema. Esto podía provocar serios problemas de seguridad si el servidor era tomado, ya que proporcionaba una ventana a través de la cual un hacker podía obtener acceso total al servidor. La versión 1.5 de este programa (instsvc.exe) acepta un usuario común autenticado por windows para la instalación del servicio. Se recomienda encarecidamente crear un usuario Firebird para este propósito y hacer uso de la nueva característica si el servidor está conectado a internet.

Recorte de los campos varchar para protocolos remotos

Este trabajo ha sido retomado y completado para implementar esta difícil característica en el cliente 1.5 y ahora los varchars viajan recortados a la longitud utilizada más 2 bytes.
NOTA Como es el cliente el que pide los varchar recortados al servidor, el cliente 1.5 (fbclient.dll o libfbclient.so) lo hará aunque se conecte a una versión anterior. Igualmente, si se utiliza un cliente antiguo, no lo hará aunque se conecte a un servidor 1.5 o posterior.

Semántica de triggers polivalentes

Ahora es posible escribir condicionalmente acciones insert/update/delete en un trigger before o after para tener cubiertas en un solo lugar todas las acciones DML posibles en esa fase. Esto reduce el trabajo de composición y mantenimiento de triggers, sin detrimento de la posibilidad de tener múltiples triggers por fase.

Mejora de las restricciones (constraints) con nombre

Los índices asociados a restricciones de integridad con nombre, pueden ahora ser creados con nombres definidos por el usuario. Tenga cuidado: si utiliza esta característica, su base de datos no podrá ser utilizada por Firebird 1.0.x o Interbase®.

Se incrementa el número máximo de índices por tabla

Ahora—tanto en la versión 1.0 como en ésta—el máximo número de índices que se pueden definir en una tabla se ha incrementado de 64 a 256.

Bloqueo pesimista

Para las raras ocasiones en las que es necesario imponer un bloqueo pesimista, esta versión añade la sintaxis para colocar "bloqueos de lectura", en los registros que son enviados al cliente. Utilícelo con precaución.

Conexión a la base de datos de seguridad en caché

La conexión a la base de datos de seguridad se conserva en caché en la arquitectura SS.Esto significa que la conexión a security.fdb se realiza cuando el primer cliente accede y permanece activa hasta que todos los clientes se desconectan.

Mejoras en los mensajes de error

En la medida de lo posible, los mensajes de error indican la causa de los errores SQL al máximo nivel de detalle. Es IMPORTANTE indicar que encontrará mensajes de error extraños si utiliza un fichero interbase.msg o firebird.msg antiguo.

API de servicios en la versión Classic para Linux

Se dispone ahora de un soporte limitado para la API de servicios en el servidor Classic bajo Linux. Los servicios de gbak (backup/restore) y gfix (validación, apagado/disponible (shutdown/online) de las base de datos, etc) funcionan. Otros servicios (gstat, server logs, etc.) no han sido probados y probablemente no sean funcionales actualmente.

Cambios en las librerías cliente

Clientes Windows

La librería cliente se llama ahora "fbclient.dll". Todas las utilidades del servidor (gbak, gfix, etc) utilizan sólo la dll fbclient.dll. Conecte sus nuevas aplicaciones a fbclient.dll, sin utilizar gds32.dll (recomendado).
Por compatibilidad con aplicaciones existentes, es posible generar un "clon" de fbclient.dll con el nombre "gds32.dll" usando una nueva utilidad llamada instclient.exe. Los detalles exactos se dan en la sección de instalación y en las notas de instalación distribuidas con el paquete de Windows.

Clientes linux

La librería cliente Superserver se llama ahora "libfbclient.so". Por compatibilidad con aplicaciones existentes, se instala el enlace simbólico "libgds.so" que apunta a libfbclient.so. La librería cliente para aplicaciones embebidas que se conecten a ClassicServer se ha renombrado como libfbembed.so.

Módulos y ficheros que cambian de nombre

Plataforma	Módulo	Firebird 1.0	Firebird 1.5	Comentarios
Todas	Variables de entorno	INTERBASE	FIREBIRD	Ruta a la raíz de la instalación
		INTERBASE_LOCK	FIREBIRD_LOCK	Ubicación del archivo de bloqueos
		INTERBASE_MSG	FIREBIRD_MSG	Ubicación del archivo de mensajes
		INTERBASE_TMP	FIREBIRD_TMP	Directorio para espacio de ordenación
Todas	Base de datos de seguridad	Isc4.gdb	security.fdb	
Todas	Fichero de mensajes	Interbase.msg	firebird.msg	
Todas	Fichero log del Servidor	interbase.log	firebird.log	

Plataforma	Módulo	Firebird 1.0	Firebird 1.5	Comentarios
Todas	Version ODS	10	10.1	El nuevo ODS no provoca ninguna incompatibilidad con los anteriores ODS, pero la versión no se actualiza automáticamente. Tanto Firebird 1.0 como Firebird 1.5 pueden servir bases de datos con ODS 10.0 y ODS 10.1. No obstante, siempre se recomienda realizar backup y posteriormente restore al migrar bases de datos a una versión diferente del servidor.
Linux	Ejecutable del Classic server	gds_inet_server	fb_inet_server	
Linux	Gestor de bloqueos del Classic server	ib_lock_mgr	fb_lock_mgr	
Linux	Control del Superserver	ibmgr.bin	fbmgr.bin	
Linux	Ejecutable del Superserver	ibserver	fbserver	
Linux	Fichero de Configuración	isc_config	firebird.conf	
Linux	Librería cliente	libgds.so	libfbclient.so libfbembed.so	Cliente remoto (seguro en operaciones multihilo) y cliente local (con conexión TCP/IP local) para Superserver Cliente local (un solo usuario, no seguro en operaciones multihilo) para Classic
Linux	Enlace simbólico a la librería cliente para compatibilidad		libgds.so	
Windows	Guardian	ibguard.exe	fbguard.exe	
Windows	Ejecutable del Superserver	Ibserver.exe	fbserver.exe	No es capaz de utilizar varios procesadores
Windows	Ejecutable del Classic Server		fb_inet_server .exe	No permite conexiones locales en windows. Funciona con TCP/IP y NetBEUI. Es capaz de utilizar varios procesadores.
Windows	Librería cliente	gds32.dll	fbclient.dll	Las versiones 1.5 de las utilidades del servidor y las nuevas aplicaciones, sólo necesitan fbclient.dll. Consulte las notas más abajo respecto a la compatibilidad de gds32.dll para aplicativos antiguos

Plataforma	Módulo	Firebird 1.0	Firebird 1.5	Comentarios
Windows	Fichero de configuración	ibconfig	firebird.conf	
Windows	Puerto local IPC	InterBaseIPI	FirebirdIPI	Por defecto el servidor se configura de forma que no se pueden realizar conexiones locales desde las aplicaciones utilizando la vieja librería cliente (gds32.dll). Si es necesario, se puede configurar el servidor para utilizar el nombre antiguo para el puerto IPC, por medio del fichero firebird.conf.
Windows	Clave del Registro por defecto	HKLM\SOFTWARE\Borland\InterBase	HKLM\SOFTWARE\Firebird Project\Firebird Server\Instances	La ruta se guarda en el parámetro "DefaultInstance". Se elimina la clave "CurrentVersion", y "RootDirectory" se sustituye por "DefaultInstance".
Los nuevos nombres de servicio en Windows son "Firebird Guardian – DefaultInstance" y "Firebird Server – DefaultInstance"				

Compatibilidad con versiones antiguas

On-disk structure (ODS)

La estructura interna de disco (on-disk structure u ODS) para Firebird 1.5 se denomina ODS 10.1. Este cambio menor en el ODS ha sido necesario debido a:
- Tres nuevos índices para las tablas del sistema
- pequeños cambios en el BLR de dos triggers del sistema
- mejoras en la codificación de RDB$TRIGGER_TYPE.

Algunas mejoras que precisaban de cambios en el ODS, han sido pospuestas para la versión 2. Como contrapartida, se pueden migrar las bases de datos de Firebird 1.0 directamente. Realice backups de sus bases de datos de Firebird 1.0.x y verifíquelos antes de realizar la migración a 1.5.

Bases de datos de InterBase™

Si planea "jugar" con Firebird utilizando una base de datos de Interbase, con la intención de volver a utilizarla con Interbase posteriormente, debe tomar todas las precauciones haciendo backup de la misma utilizando la versión adecuada de gbak. Para comenzar a trabajar con ella en Firebird 1.5, utilice la versión de gbak de firebird 1.5 para hacer el restore.
El manual Operations Guide de la documentación de InterBase® 6.0 beta detalla la sintaxis de los comandos para hacer backup y restore con gbak.

Las bases de datos de IB 7.x y probablemente también las de IB 6.5 pueden trabajar incorrectamente tras la migración a FB 1.5 via backup/restore, si se han utilizados las nuevas características propias de IB.

Nombres de ficheros y su localización

En esta versión, una parte sustancial de los ficheros tienen nuevos nombres, como parte de un proceso gradual de sustitución de los nombres heredados de InterBase® 6. En la sección Nombres de ficheros y su localización, puede encontrar información detallada y recomendaciones al respecto.

Servidores ejecutándose de forma concurrente

Los cambios realizados a algunos de los objetos del sistema permiten instalar y utilizar FB 1.5 en un ordenador que ya tenga instalado Interbase o Firebird 1.0.x. En Windows, FB 1.5 también utiliza otras claves de registro. Si se configuran los servidores para utilizar diferentes puertos de red, es posible ejecutar varias instancias del servidor de forma concurrente, o ejecutar FB 1.5 mientras IB o FB 1.0.x está ejecutándose.

Retroceder a Firebird 1.0.x

A causa de la gran cantidad de errores corregidos, el buen comportamiento de las bases de datos puede cambiar si se traspasan de la versión 1.5 a la versión 1.0.x. Especialmente si se crean claves primarias, claves únicas o claves foráneas como restricciones con nombre, los nombres por defecto de los índices serán incompatibles con v.1.0.x. Puede que en un futuro se publique un LEAME (probablemente en inglés, README) detallando todas las incidencias que pueden suceder.

Compatibilidades en linux

A causa de los problemas con el compilador C++ GNU, Firebird 1.5 para linux necesita unas versiones más recientes de las librerías glibc que antes. Esto significa, desgraciadamente, que estamos en un periodo en el que la capacidad de algunas distribuciones en particular para instalar y ejecutar la 1.5 es algo difícil de predecir.

La siguiente tabla puede ayudar. No obstante, será bienvenida información adicional. Por favor, comparta sus experiencias con estas y otras distribuciones en el foro de desarrollo de firebird.

Distribución	Versión	Classic	Superserver
Red Hat	7.x	No	No
	8.0	Sí	Sí
	9.0	Sí	Sí
Mandrake	8.x	No	No
	9.0, 9.1, 9.2	Sí	Sí
SuSE	7.3	Sí – Instale los paquetes libgcc-3.2-44.i586.rpm y libstdc++-3.2-44.i586.rpm antes de instalar Firebird 1.5	Se desconoce
	8.0, 8.1	Sí	8.0 Sí (8.1 se desconoce)

Mejoras en el Lenguaje

TIPOS DE DATOS

(1.5) Nuevo tipo de dato Nativo SQL

BIGINT
Tipo de dato numérico exacto acorde a SQL-99, de 64 bit con signo, con escala cero. Disponible sólo en el dialecto 3.

Ejemplo(s)
i)
```
DECLARE VARIABLE VAR1 BIGINT;
```
ii)
```
CREATE TABLE TABLE1 (FIELD1 BIGINT);
```

METADATOS

(1.5) Mejoras en las restricciones (constraints) con nombre

Dmitry Yemanov

Los índices asociados a restricciones de integridad con nombre, pueden ahora ser creados con nombres definidos por el usuario.

Anteriormente, aunque era posible crear claves primarias, foráneas y únicas con nombre, el identificador del índice asociado que se creaba automáticamente era calculado por el sistema, por ejemplo RDB$FOREIGN13 y no podía ser cambiado. Este comportamiento permanece por defecto cuando no se utilizan restricciones con nombre.

En cualquier caso, se han añadido extensiones del lenguaje para permitir
a) A un índice generado por el sistema recibir automáticamente el mismo identificador que la restricción con nombre a la que está asociado
b) A un índice que se asocia a una restricción con o sin nombre serle asignado explícitamente un identificador en concreto y ser construido en orden descendente.
NOTA Ello no es posible actualmente utilizando un indice preexistente.

Sintaxis
```
...
[ADD] CONSTRAINT [<identificador_de_restriccion>]
<tipo_de_restriccion> <definicion_de_restriccion>
[USING [ASC[ENDING] | DESC[ENDING]] INDEX <nombre_del_indice_asociado>]
```

Advertencia: Cerciórese de que los índices de la foreign key y de la primary key utilizan el **mismo tipo de ordenación** (DESC | ASC).

Ejemplos

i) Restricción con nombre e índice con nombre concreto

```
CREATE TABLE ATEST (
  ID BIGINT NOT NULL,
  DATA VARCHAR(10));
COMMIT;
```

La siguiente sentencia creará una clave primaria llamada PK_ATEST y un índice asociado descendente llamado IDX_PK_ATEST:

```
ALTER TABLE ATEST
ADD CONSTRAINT PK_ATEST PRIMARY KEY(ID)
USING DESC INDEX IDX_PK_ATEST;
COMMIT;
```

ii) Alternativa a i):

```
CREATE TABLE ATEST (
  ID BIGINT NOT NULL,
  DATA VARCHAR(10),
  CONSTRAINT PK_ATEST PRIMARY KEY(ID)
  USING DESC INDEX IDX_PK_ATEST;
```

iii) Esta sentencia crea la tabla ATEST con la clave primaria PK_ATEST. El índice asociado se llama también PK_ATEST.

```
CREATE TABLE ATEST (
  ID BIGINT NOT NULL,
  DATA VARCHAR(10),
  CONSTRAINT PK_ATEST PRIMARY KEY(ID));
```

(1.5) Triggers polivalentes

Dmitry Yemanov

Los triggers se han mejorado para permitir manejar diferentes operaciones a nivel de registro de forma condicional.

Sintaxis

```
CREATE TRIGGER name FOR table
  [ACTIVE | INACTIVE]
  {BEFORE | AFTER} <multiple_action>
  [POSITION number]
AS trigger_body
```

<accion_multiple> ::= <accion_simple> [OR < accion_simple > [OR < accion_simple >]]
< accion_simple > ::= {INSERT | UPDATE | DELETE}

Ejemplos
i)
```
CREATE TRIGGER TRIGGER1 FOR TABLE1
ACTIVE BEFORE INSERT OR UPDATE AS
...;
```

ii)
```
CREATE TRIGGER TRIGGER2 FOR TABLE2
ACTIVE AFTER INSERT OR UPDATE OR DELETE AS
...;
```

Cambio en el ODS
La codificación del campo RDB$TRIGGER_TYPE (tabla RDB$TRIGGERS) ha sido extendido para permitir acciones complejas en los triggers. Para conocer todos los detalles, diríjase al documento readme.universal_triggers.txt[1] que se encuentra en la carpeta /docs/sql.extesions del CVS de Firebird.
Nota(s):
1. Los triggers simples son totalmente compatibles con el ODS de FB 1.0.
2. El campo RDB$TRIGGER_TYPE depende del orden, por ejemplo, BEFORE INSERT OR UPDATE y BEFORE UPDATE OR INSERT serán codificados con valores diferentes, aunque tienen la misma semántica y se ejecutarán exactamente igual.
3. En los triggers polivalentes se dispone de las variables contextuales OLD y NEW. Cuando el uso de alguna de ellas no tiene sentido (por ejemplo la variable OLD en una operación de INSERT), todos los campos de esa variable serán evaluados a NULL. Si son asignadas en un contexto inadecuado, se disparará una excepción en tiempo de ejecución.
4. Para evaluar el tipo de operación en tiempo de ejecución se dispone de las nuevas variables contextuales booleanas INSERTING/UPDATING/DELETING. (Ver más adelante)

(1.5) RECREATE VIEW

Hace exactamente lo mismo que CREATE VIEW si la vista no existe. Si existe, RECREATE VIEW intentará borrarla y crear de nuevo el objeto. RECREATE VIEW fallará si el objeto está en uso.
Utiliza la misma sintaxis que CREATE VIEW.

(1.5) CREATE OR ALTER {TRIGGER | PROCEDURE }

Sentencia que, o bien creará un nuevo trigger o procedure (si no existe) o lo modificará (si ya existe) y recompilará. La sintaxis CREATE OR ALTER respeta las dependencias y permisos existentes.

La sintaxis es como la de CREATE TRIGGER | CREATE PROCEDURE, respectivamente, excepto por las palabras adicionales "OR ALTER".

(1.5) Valores nulos (NULL) en índices y claves únicas (UNIQUE)

Dmitry Yemanov

Ahora es posible aplicar una clave unica o crear un indice unico (UNIQUE Constraint o UNIQUE Index) a una columna que no tiene la restricción de NOT NULL. Esto cumple con el estándar SQL-99. Sea precavido al utilizar esta característica si planea migrar su base de datos a Firebird 1.0.x o a cualquier version de Interbase.

[1] Nota del Traductor: Este documento se encuentra en inglés

```
<unique constraint definition or index definition> ::=
<unique specification> ( <unique column list UCL> )
<unique specification> ::=
{{[constraint-name]UNIQUE | UNIQUE INDEX index-name]} | [constraint-name]
PRIMARY KEY}
```

donde <unique column list> puede contener una o más columnas sin el atributo NOT NULL, si <unique specification> es UNIQUE o UNIQUE INDEX index-name. Note que todas las columnas de una clave primaria todavía deben ser declaradas con NOT NULL.

La restricción permite la existencia únicamente de aquéllos registros para los que la condición (i) o (ii) es evaluada como verdadera, de acuerdo con la siguiente lógica:

i) si <unique specification> especifica PRIMARY KEY, entonces la condición (serch condition) será

UNIQUE (SELECT UCL FROM TN) AND (UCL) IS NOT NULL

ii) en otro caso, <search condition> será

UNIQUE (SELECT UCL FROM TN)

En este caso, la condición UNIQUE puede no ser verdadera si se da el caso que (SELECT UCL FROM TN) devuelve dos filas donde todos los pares no nulos correspondientes son iguales.

La restricción permite la existencia solamente de aquellas filas para las cuales la condición <search condition> mencionada más arriba se evalúa verdadera. En un índice unique o bajo una restricción UNIQUE, dos conjuntos de valores pueden ser considerados diferentes y por lo tanto ser admitidos si:

a) ambos conjuntos contienen sólo valores nulos, o
b) existe al menos un par de valores correspondientes de los cuales uno es no nulo y los otros nulos o un valor no nulo diferente.

Ejemplos

Restricción UNIQUE:

```
CREATE TABLE t (a INTEGER, b INTEGER, CONSTRAINT pk UNIQUE (a, b));
```

o índice UNIQUE:

```
CREATE TABLE t (a INTEGER, b INTEGER);
COMMIT;
CREATE UNIQUE INDEX uqx ON t(a, b);
COMMIT;
INSERT INTO t VALUES (NULL, NULL); /* ok, se permiten los nulos */
INSERT INTO t VALUES (1, 2); /* al igual que los no nulos */
INSERT INTO t VALUES (1, NULL); /* y combinaciones */
INSERT INTO t VALUES (NULL, NULL); /* ok, todos los pares de nulos son diferentes */
```

no obstante,

```
INSERT INTO t VALUES (1, NULL); /* falla debido a que todos los valores no nulos coinciden */
```

Esto significa que ***la restricción PRIMARY KEY no permite nulos*** mientras que la restricción UNIQUE y los índices unique permiten un número arbitrario de nulos. Para conjuntos de resultados de varias columnas en (SELECT UCL FROM TN), se aplican las reglas normales para los nulos, por ejemplo (1,NULL) es distinto de (NULL,1) y un (NULL, NULL) es distinto de cualquier otro (NULL, NULL).

DSQL

(1.5) Expresiones y variables como argumentos de procedimientos almacenados

Dmitry Yemanov

Las llamadas a EXECUTE PROCEDURE NombreProc(<Lista-Argumentos>) y
SELECT <Lista-Campos> FROM NombreProc(<Lista-Argumentos>) pueden ahora aceptar variables locales (en PSQL) y expresiones (en DSQL y PSQL) como argumentos.

(1.5) Nuevas construcciones para las expresiones CASE

Arno Brinkman

a) CASE

Permite que el resultado de una columna sea determinado por el resultado de evaluar un grupo de condiciones exclusivas.

Sintaxis

```
 <expresion case > ::=
     <abreviatura case >  | <especificacion case>

 <abreviatura case> ::=
     NULLIF <parentesis izdo> <expresion valor> <coma> <expresion valor> <parentesis dcho>
| COALESCE <parentesis izdo > < expresion valor > { <coma> < expresion valor > }... < parentesis dcho >

 < especificacion case > ::=
     <case simple>  | <case buscado>

 <case simple> ::=
   CASE < expresion valor >  <clausula when simple>...
     [ <clausula else> ]
   END

 <case buscado> ::=
   CASE <clausula when buscada>...
     [ <clausula else> ]
   END

 <clausula when simple> ::= WHEN <operador when> THEN <resultado>
 <clausula when buscada> ::= WHEN <condicion de busqueda> THEN <resultado>
 <operador when> ::= <expresion valor>
 <clausula else> ::= ELSE <resultado>
 <resultado> ::= <expresion resultado>  | NULL
 <expresion resultado> ::= <expresion valor>
```

Ejemplos

i) simple

```
SELECT
    o.ID,
    o.Description,
    CASE o.Status
      WHEN 1 THEN 'confirmed'
      WHEN 2 THEN 'in production'
      WHEN 3 THEN 'ready'
      WHEN 4 THEN 'shipped'
      ELSE 'unknown status ''' || o.Status || ''''
    END
FROM Orders o;
```

ii) buscado

```
  SELECT
    o.ID,
    o.Description,
    CASE
      WHEN (o.Status IS NULL) THEN 'new'
      WHEN (o.Status = 1) THEN 'confirmed'
      WHEN (o.Status = 3) THEN 'in production'
      WHEN (o.Status = 4) THEN 'ready'
      WHEN (o.Status = 5) THEN 'shipped'
      ELSE 'unknown status ''' || o.Status || ''''
    END
  FROM Orders o;
```

b) COALESCE

Permite al valor de una columna ser calculado por un número de expresiones, de forma que se toma como valor, el valor de la primera expresión que devuelva un valor no nulo.

Formato

<abreviatura casen> ::=
| COALESCE <parentesis izdo> <expresion valor> { <coma> <expresion valor> }… <parentesis dcho>

Reglas Sintácticas

i) COALESCE (V1, V2) es equivalente a la siguiente <especificacion case>:
CASE WHEN V1 IS NOT NULL THEN V1 ELSE V2 END

ii) COALESCE (V1, V2,…, Vn), para n >= 3, es equivalente a la siguiente:
<especificacion case>:
CASE WHEN V1 IS NOT NULL THEN V1 ELSE COALESCE (V2,…,Vn) END

Ejemplos

```
SELECT
    PROJ_NAME AS Projectname,
    COALESCE(e.FULL_NAME,'[> not assigned <]') AS Employeename
FROM
    PROJECT p
    LEFT JOIN EMPLOYEE e ON (e.EMP_NO = p.TEAM_LEADER);
```

```
SELECT
    COALESCE(Phone,MobilePhone,'Unknown') AS "Phonenumber"
FROM
    Relations
```

c) NULLIF

Devuelve NULL para una sub-expresión si tiene una valor específico; en otro caso devuelve el valor de la subexpresión.

Formato

<abreviatura case> ::=
NULLIF <parentesis izdo> <expresion valor> <coma> <expresion valor> <parentesis dcho>

Reglas Sintácticas

NULLIF (V1, V2) es equivalente a la siguiente <especificacion case>:
CASE WHEN V1 = V2 THEN NULL ELSE V1 END

Ejemplo

```
UPDATE PRODUCTS
    SET STOCK = NULLIF(STOCK,0)
```

(1.5) Puntos de rescate (savepoints) acordes a SQL99

Nickolay Samofatov

Los puntos de rescate (savepoints; como alternativa a las transacciones anidadas con nombre) proporcionan un metodo adecuado para manejar errores lógicos sin necesidad de hacer rollback de la transacción. Disponible solo en DSQL.

Utilice la sentencia SAVEPOINT para identificar un punto en la transacción al cual podrá volver posteriormente, haciendo un rollback parcial.

```
SAVEPOINT <identificador>;
```

<identificador> especifica el nombre del punto de rescate a crear. Tras ser creado, usted puede elegir entre continuar procesando, hacer commit, hacer rollback o hacer rollback hasta el punto de rescate Los nombres de los puntos de rescate deben ser distintos dentro de una transacción dada. Si se crea un segundo punto de rescate con el mismo nombre que un punto de rescate anterior, el punto de rescate anterior se elimina

```
ROLLBACK [WORK] TO [SAVEPOINT] <identificador>;
```

Esta sentencia realiza las siguientes operaciones:

- Hace Rollback de todos los cambios realizados en la transacción después del punto de rescate
- Borra todos los puntos de rescate creados con posterioridad al punto de rescate. El punto de rescate en cuestión se mantiene, por tanto es posible realizar rollback al mismo punto de rescate varias veces.Los puntos de rescate anteriores también se mantienen.
- Libera todos los bloqueos de registro realizados de forma implícita o explícita después del punto de rescate. Las transascciones que hayan pedido acceso a los registros bloqueados tras el punto de rescate, deben seguir esperando hasta que la transacción haga commit o rollback. Las transacciones que soliciten acceso y no lo hayan solicitado ya, pueden acceder a los registros de forma inmediata.
  Nota: este comportamiento puede cambiar en fututras versiones del producto.

El seguimiento necesario para volver a un punto de rescate puede consumir gran cantidad de memoria en el servidor, especialmente si se modifican los mismos registros en la misma transacción varias veces. Utilice la sentencia RELEASE SAVEPOINT para liberar los recursos consumidos en el mantenimiento del punto de rescate.

```
RELEASE SAVEPOINT <identificador> [ONLY];
```

La sentencia RELEASE SAVEPOINT elimina el punto de rescate <identificador> del contexto de la transacción. A no ser que se especifique la cláusula ONLY , todos los puntos de rescate establecidos tras el punto de rescate <identificador> son eliminados también.

Ejemplo del uso de puntos de rescate

```
create table test (id integer);
commit;
insert into test values (1);
commit;
insert into test values (2);
savepoint y;
delete from test;
select * from test; -- returns no rows
rollback to y;
select * from test; -- returns two rows
rollback;
select * from test; -- returns one row
```

Puntos de rescate internos

Por defecto, el motor utiliza un punto de rescate automático de sistema a nivel de transacción para realizar rollback de la misma. Cuando se lanza una sentencia ROLLBACK, todos los cambios realizados en esa transacción se deshacen via este punto de rescate y entonces se hace commit de la transacción. Esta lógica reduce la cantidad de basura provocada por las transacciones que hacen rollback.

Cuando la cantidad de cambios realizados en una transacción es demasiado grande (del orden de 10^4 a 10^6 registros afectados) el motor libera el punto de rescate a nivel de transacción y usa el mecanismo TIP para hacer rollback de la transacción si fuera necesario. Si usted espera que el volumen de cambios en su transacción va a ser muy grande, puede usar el flag TPB isc_tpb_no_auto_undo para evitar que el punto de rescate automático se cree.

Puntos de rescate y PSQL

Al implementar puntos de rescate en la capa del PSQL, se puede romper la regla de atomicidad para las sentencias, incluyendo las llamadas a los procedimientos almacenados. Firebird proporciona manejadotes de excepciones en el PSQL para deshacer los cambios realizados en los procedimientos almacenados y triggers. Cada sentencia SQL/PSQL se ejecuta bajo un punto de rescate de sistema automático, donde o bien toda la sentencia se ejecuta o bien TODOS los cambios se deshacen y se lanza una excepción. Cada bloque manejado por una excepción PSQL se enlaza con un punto de rescate automático del sistema.

(1.5) Bloqueos explícitos

Nickolay Samofatov

La introducción de la cláusula opcional WITH LOCK proporciona una limitada capacidad de hacer bloqueos pesimistas explícitos para uso prudente en las condiciones en las que el conjunto de registros afectados es a) extremadamente pequeño (mejor si es uno solo) y b) exactamente controlado por el código del aplicativo.

NOTA La necesidad de un bloqueo pesimista en Firebird es muy rara de hecho y debe ser bien entendido antes de plantearse utilizar esta extensión.

Sintaxis

```
SELECT ... FROM <algunatabla>
  [WHERE ...]
  [FOR UPDATE [OF ...]]
  [WITH LOCK]
...;
```

Si la cláusula WITH LOCK se ejecuta bien, se realizará un bloqueo de los registros seleccionados y se evitará que cualquier otra transacción obtenga acceso de escritura a dichos registros, o sus dependientes, hasta que su transacción termine.

Si se incluye la cláusula FOR UPDATE, el bloqueo se aplicará a cada registro, uno por uno, guardándose en la caché del servidor. Llega a ser posible, entonces, que un bloqueo que pareció funcionar cuando fue solicitado sin embargo falle posteriormente, cuando se intenta enviar un registro que está bloqueado por otra transacción.

Es esencial comprender los efectos del nivel de aislamiento de las transacciones y otros atributos de las mismas antes de intentar incluir bloqueos explícitos en las aplicaciones.

La construcción SELECT... WITH LOCK está disponible en DSQL y PSQL. Sólo funciona en el nivel más alto, con sentencias SELECT que afectan a una única tabla. No está disponible en subconsultas, ni para joins. No se puede especificar con el operador DISTINCT, la cláusula GROUP BY o cualquier otra operación de agregación de registros. No puede utilizarse con o en una vista, ni con tablas externas, ni con los resultados de un procedimiento almacenado llamado desde un select.

Comprendiendo la cláusula WITH LOCK

Como el motor considera, por turnos, cada registro que cae bajo una sentencia de bloqueo explícita, bien devuelve la versión del registro que ha sido confirmada lo más actualmente posible, sin importar el estado de la base de datos cuando la sentencia fue enviada, o bien devuelve una excepción.

El comportamiento esperado y la notificación de conflictos depende de los parámetros especificados a la transacción (modo TPB).

Modo TPB	*Comportamiento*
isc_tpb_consistency	Los bloqueos explícitos son sobreescritos por los bloqueos explícitos o implícitos a nivel de tabla y son ignorados
isc_tpb_concurrency + isc_tpb_nowait	Si un registro es modificado por una transacción confirmada después del comienzo de la transacción que esperaba conseguir el bloqueo explícito , o una transacción activa ha realizado una modificación de este registro, una excepción de conflicto de actualización se lanza inmediatamente
isc_tpb_concurrency + isc_tpb_wait	Si el registro es modificado por una transacción confirmada después del comienzo de la transacción que esperaba conseguir el bloqueo explícito, una excepción de conflicto de actualización se lanza inmediatamente. Si una transacción activa está pretendiendo apoderarse de este registro(vía bloqueo explícito o por un bloqueo de escritura optimista normal) la transacción que intenta el bloqueo explícito esperará el resultado del bloqueo de la otra transacción y, cuando acabe, intentará conseguir el bloqueo del registro otra vez. Esto significa que, si la otra transacción hizo commit de la versión modificada de este registro, se lanzará una excepción de conflicto de actualización.
isc_tpb_read_committed + isc_tpb_nowait	Si una transacción activa está pretendiendo apoderarse de este registro(vía bloqueo explícito o por un bloqueo de escritura optimista normal), se lanzará una excepción de conflicto de actualización.
isc_tpb_read_committed + isc_tpb_wait	Si una transacción activa está pretendiendo apoderarse de este registro(vía bloqueo explícito o por un bloqueo de escritura optimista normal) la transacción que intenta el bloqueo explícito esperará el resultado del bloqueo de la otra transacción y, cuando acabe, intentará conseguir el bloqueo del registro otra vez. Nunca se lanzarán excepciones por conflicto de actualización debidas a sentencias de bloqueo explícitas con este modo TPB.

Cuando una sentencia UPDATE intenta acceder un registro que está bloqueado por otra transacción, lanza una excepción por conflicto de actualización o espera a que la transacción que lo bloquea termine, dependiendo del modo TPB. El comportamiento del motor es el mismo que si este registro ya ha sido modificado por la transacción que lo bloquea. No se devuelven valores GDSCODE especiales por conflictos que involucren bloqueos explícitos.

El motor garantiza que todos los registros devueltos por una sentencia con bloqueo explícito están bloqueados y cumplen las condiciones impuestas en la cláusula WHERE, siempre y cuando estas condiciones no dependan de otras tablas vía joins, subconsultas, etc. Esto garantiza a su vez que los registros que no cumplen la condición del WHERE no serán bloqueados. Lo que no garantiza es que todos los registros que cumplen la condición sean bloqueados; algunos de ellos pueden no serlo. Esta situación puede surgir si otra transacción, paralelamente hace commit de sus cambios durante el curso de la ejecución de la sentencia de bloqueo.

El motor bloquea los registros en el momento de enviarlos. Esto tiene importantes consecuencias si se bloquean muchos registros a la vez. Algunos métodos de acceso a las bases de datos Firebird, por defecto envían los resultados en paquetes de unos cuantos cientos de registros ("fuffered fetches" o envíos por buffer). La mayor parte de los componentes de acceso a datos no pueden traerse los registros contenidos en el último paquete enviado, si se produce un error.

La cláusula FOR UPDATE proporciona la técnica para evitar el uso de los envíos por búfer, opcionalmente con la cláusula OF <nombres de columnas> para permitir modificaciones posicionadas. Como alternativa, se puede asignar el búfer de envío de los componentes de acceso a datos a 1. Esto le permitirá procesar el registro bloqueado actualmente antes de que el siguiente sea enviado y procesado, o manejar los errores sin hacer rollback de la transacción.

Al revertir (hacer rollback) un punto de rescate (savepoint) implícito o explícito se liberan los registros bloqueados que han sido utilizados bajo ese punto de rescate, pero no se notifica a las transacciones que se encuentran en espera. Las aplicaciones no deberían depender de este comportamiento, ya que puede cambiar en el futuro.

Mientras los bloqueos explícitos pueden ser usados para prevenir y/o manejar errores por conflictos de modificación poco comunes, el volumen de los errores por deadlock pueden crecer a menos que usted diseñe su estrategia de bloqueos cuidadosamente y controle esto rigurosamente. La mayoría de las aplicaciones no necesitan bloqueos explícitos en absoluto. Los principales objetivos de los bloqueos explícitos son (1) evitar las costosas manipulaciones de los errores por conflictos de modificación en aplicaciones sobrecargadas y (2) mantener la integridad de los objetos mapeados en una base de datos relacional en un entorno clúster. Si el uso que va a hacer del bloqueo explícito no entra dentro de estos dos casos, entonces está utilizando una mala forma de trabajo con Firebird.

El bloqueo explícito es una característica avanzada, ¡no haga mal uso de ella!. Mientras que soluciones de este tipo pueden ser interesantes para sitios web que manejen cientos de conexiones de escritura concurrentes, o para sistemas operativos ERP/CRM de grandes corporaciones, la mayor parte de los aplicativos no necesitan trabajar en semejantes condiciones.

Ejemplos

i) (simple)

```
SELECT * FROM DOCUMENT WHERE ID=? WITH LOCK
```

ii) (varios registros, procesados uno a uno por un cursor DSQL)

```
SELECT * FROM DOCUMENT WHERE PARENT_ID=? FOR UPDATE WITH LOCK
```

(1.5) Mejoras en el manejo de agregaciones

Arno Brinkman

Originalmente, los conjuntos agrupados sólo se podían construir con columnas con nombre. En Firebird 1.0, se hizo posible agrupar por una expresión de una UDF. En 1.5 se han añadido varias extensiones nuevas para manejar funciones agregadas y la cláusula GROUP BY permite ahora agrupaciones fabricadas en base al orden (función degree) de las columnas que devuelve la consulta (el orden va de izquierda a derecha comenzando por el 1, como en la cláusula ORDER BY) o en base a diversas expresiones.
NOTA: No todas las expresiones están disponibles actualmente dentro de GROUP BY. Por ejemplo, la concatenación no se permite.

Sintaxis de Group By

```
SELECT ... FROM .... [GROUP BY lista_group_by]

lista_group_by : elemento_group_by [, lista_group_by];

elemento_group_by : nombre_columna
                  | degree (ordinal)
                  | udf
                  | funcion_group_by;

funcion_group_by : funcion_de_valor_numerico
                 | funcion_de_valor_texto
                 | expresion_case
                 ;

funcion_de_valor_numerico : EXTRACT '(' parte_de_timestamp FROM valor ')';

funcion_de_valor_texto    :  SUBSTRING '(' valor FROM posicion ')'
                          | SUBSTRING '(' valor FROM posicion FOR longitud ')'
                          | KW_UPPER '(' valor ')'
                          ;
```

El elemento_group_by no puede ser una referencia a otra función agregada (incluídas las que están dentro de la expresión) del mismo contexto.

HAVING
La cáusula having sólo permite funciones agregadas o expresiones válidas que formen parte de la cláusula GROUP BY. Anteriormente permitía utilizar columnas que no formaban parte de la cláusula GROUP BY y utilizar expresiones no válidas.

ORDER BY
Cuando el contexto es una sentencia agregada, la cláusula ORDER BY sólo permite expresiones válidas que sean funciones agregadas o partes de las expresiones de la cláusula GROUP BY. Previamente se permitía utilizar expresiones no válidas.

Funciones agregadas dentro de subconsultas
Ahora es posible utilizar una función agregada o expresión contenida en una cláusula GROUP BY dentro de una subconsulta.

Ejemplos

```
SELECT
    r.RDB$RELATION_NAME,
    MAX(r.RDB$FIELD_POSITION),
    (SELECT
       r2.RDB$FIELD_NAME
     FROM
       RDB$RELATION_FIELDS r2
     WHERE
       r2.RDB$RELATION_NAME = r.RDB$RELATION_NAME and
       r2.RDB$FIELD_POSITION = MAX(r.RDB$FIELD_POSITION))
  FROM
    RDB$RELATION_FIELDS r
  GROUP BY
    1
```

```
SELECT
  rf.RDB$RELATION_NAME AS "Relationname",
  (SELECT
     r.RDB$RELATION_ID
   FROM
     RDB$RELATIONS r
   WHERE
     r.RDB$RELATION_NAME = rf.RDB$RELATION_NAME) AS "ID",
  COUNT(*) AS "Fields"
FROM
  RDB$RELATION_FIELDS rf
GROUP BY
  rf.RDB$RELATION_NAME
```

Mezclando funciones agregadas de diferentes contextos
Se pueden utilizar funciones agregadas de diferentes contextos dentro de una expresión.

Ejemplo
```
SELECT
    r.RDB$RELATION_NAME,
    MAX(i.RDB$STATISTICS) AS "Max1",
    (SELECT
       COUNT(*) || ' - ' || MAX(i.RDB$STATISTICS)
     FROM
       RDB$RELATION_FIELDS rf
     WHERE
       rf.RDB$RELATION_NAME = r.RDB$RELATION_NAME) AS "Max2"
  FROM
    RDB$RELATIONS r
    JOIN RDB$INDICES i on (i.RDB$RELATION_NAME = r.RDB$RELATION_NAME)
  GROUP BY
    r.RDB$RELATION_NAME
  HAVING
    MIN(i.RDB$STATISTICS) <> MAX(i.RDB$STATISTICS)
```

¡Nota!. Esta consulta devuelve resultados en Firebird 1.0, ¡pero son ERRÓNEOS!

Se permiten subconsultas dentro de una función agregada
Se permite utilizar, dentro de una función de agregación, un select que devuelva un sólo registro.

Ejemplo
```
SELECT
    r.RDB$RELATION_NAME,
    SUM((SELECT
           COUNT(*)
         FROM
           RDB$RELATION_FIELDS rf
         WHERE
           rf.RDB$RELATION_NAME = r.RDB$RELATION_NAME))
  FROM
    RDB$RELATIONS r
    JOIN RDB$INDICES i on (i.RDB$RELATION_NAME = r.RDB$RELATION_NAME)
  GROUP BY
    r.RDB$RELATION_NAME
```

Funciones agregadas anidadas
Se permite utilizar una función agregada dentro de otra función agregada si la función agregada de dentro es de un contexto más reducido (ver ejemplo más abajo).

Agrupar por degree (número de orden)
Cuando se utiliza el número de orden de salida de una columna en la cláusula GROUP BY, se 'copia' la expresión correspondiente de la cláusula del select (como hace la cláusula ORDER BY). Esto significa que, cuando el número de orden se refiere a un subquery, el subquery se ejecuta cada vez.

(1.5) La cláusula ORDER BY puede contener expresiones y la colocación de los nulos

Nickolay Samofatov

La cláusula ORDER BY permite especificar cualquier expresión válida para ordenar los resultados de las consultas. Si la expresión consiste en un número, se interpreta como el número de orden (degree) de la columna, igual que antes.
El orden en el que aparecerán los nulos en el resultado, puede ser controlado con la cláusula de colocación de nulos. Los resultados se pueden ordenar colocando los valores nulos al principio (NULLS FIRST) o al final (NULLS LAST) de los valores no nulos.
El comportamiento cuando no se especifica nada es NULLS LAST.

Sintaxis

```
SELECT ... FROM .... [ORDER BY order_list] ....;
order_list : order_item [, order_list];
order_item : <expression> [order_direction] [nulls_placement]
order_direction : ASC | DESC;
nulls_placement : NULLS FIRST | NULLS LAST;
```

Restricciones
- Si se especifica NULLS FIRST, no se utilizará ningún índice para la ordenación.
- Los resultados de una ordenación basada en los valores devueltos por una UDF o procedimiento almacenado serán impredecibles si los valores devueltos no pueden ser usados para determinar una secuencia de ordenación lógica.
- La carga de trabajo generada por una ordenación basada en una UDF o procedimiento almacenado será impredecible, a pesar de, o si el orden está especificado por la propia expresión o por un número ordinal que represente una expresión en la especificación de la lista de columnas.
- Una cláusula ORDER BY para ordenar el resultado de una consulta que contenga UNION, sólo puede usar el número de orden de las columnas para referirse a ellas.

Ejemplos
i)
```
SELECT * FROM MSG
ORDER BY PROCESS_TIME DESC NULLS FIRST
```

ii)
```
SELECT FIRST 10 * FROM DOCUMENT
ORDER BY STRLEN(DESCRIPTION) DESC
```

iii)
```
SELECT DOC_NUMBER, DOC_DATE FROM PAYORDER
UNION ALL
SELECT DOC_NUMBER, DOC_DATA FROM BUDGORDER
ORDER BY 2 DESC NULLS LAST, 1 ASC NULLS FIRST
```

(1.5) EXECUTE STATEMENT

Alex Peshkov

Extensión PSQL que admite una cadena de caracteres la cual es una consulta DSQL (dynamic SQL) y la ejecuta como si hubiera sido enviada a DSQL.
Disponible en triggers y procedimientos almacenados.

La sintaxis puede tener tres variantes.

Sintaxis 1
Ejecuta <cadena> como una operación SQL que no devuelve ningún registro, es decir INSERT, UPDATE, DELETE, EXECUTE PROCEDURE o cualquier sentencia DDL excepto CREATE/DROP DATABASE.

```
EXECUTE STATEMENT <cadena>;
```

Ejemplo

```
CREATE PROCEDURE DynamicSampleOne (Pname VARCHAR(100))
AS
DECLARE VARIABLE Sql VARCHAR(1024);
DECLARE VARIABLE Par INT;
BEGIN
   SELECT MIN(SomeField) FROM SomeTable INTO :Par;
   Sql = 'EXECUTE PROCEDURE ' || Pname || '(';
   Sql = Sql || CAST(Par AS VARCHAR(20)) || ')';
   EXECUTE STATEMENT Sql;
END
```

Sintaxis 2
Ejecuta <cadena> como una sentencia SQL que devuelve un solo registro. Solamente se pueden ejecutar consultas SELECT que devuelvan un único registro con esta variante de EXECUTE STATEMENT.

```
EXECUTE STATEMENT <cadena> INTO :var1, […, :varn] ;
```

Ejemplo

```
CREATE PROCEDURE DynamicSampleTwo (TableName VARCHAR(100))
AS
DECLARE VARIABLE Par INT;
BEGIN
   EXECUTE STATEMENT 'SELECT MAX(CheckField) FROM ' || TableName INTO :Par;
   IF (Par > 100) THEN
      EXCEPTION Ex_Overflow 'Overflow in ' || TableName;
END
```

Sintaxis 3
Ejecuta <cadena> como una sentencia SQL que devuelve varios registros. En esta variante de EXECUTE STATEMENT se puede ejecutar cualquier sentencia SELECT.

```
FOR EXECUTE STATEMENT <cadena> INTO :var1, …, :varn DO
   <conjunto_de_sentencias>;
```

Ejemplo

```
CREATE PROCEDURE DynamicSampleThree (
   TextField VARCHAR(100),
   TableName VARCHAR(100))
RETURNS (Line VARCHAR(32000))
AS
DECLARE VARIABLE OneLine VARCHAR(100);
BEGIN
Line = '';
FOR EXECUTE STATEMENT
   'SELECT ' || TextField || ' FROM ' || TableName INTO :OneLine
   DO
      IF (OneLine IS NOT NULL) THEN
         Line = Line || OneLine || ' ';
   SUSPEND;
END
```

Notas adicionales acerca de EXECUTE STATEMENT

La cadena DSQL 'EXECUTE STATEMENT' no puede incluir ningún parámetro en ninguna de las tres variantes. Todas las variables deben sustituirse en la parte estática de la sentencia SQL antes de la ejecución de EXECUTE STATEMENT.

Es deseable utilizar esta característica con mucho cuidado y debe ser utilizada tomando en cuenta todos los factores. Debería utilizarse sólo cuando resulta imposible utilizar otros métodos, o resultan peores que EXECUTE STATEMENT.

EXECUTE STATEMENT es potencialmente inseguro y peligroso por varias razones:

1. No hay forma de comprobar la sintaxis de la sentencia SQL.
2. No se comprueban las dependencias para descubrir qué tablas o columnas han sido borradas.
3. Las operaciones serán lentas ya que la sentencia embebida debe ser preparada cada vez que se ejecuta.
4. Se aplica un checkeo estricto a los valores que devuelve respecto del tipo de dato para evitar excepciones impredecibles de conversión de tipos. Por ejemplo, la cadena '1234' sería convertida en entero, 1234, pero 'abc' daría un error de conversión.
5. Si el procedimiento almacenado tiene privilegios especiales sobre algunos objetos, la sentencia enviada en EXECUTE STATEMENT no los hereda. Los privilegios se limitan a aquéllos que han sido concedidos al usuario que ejecuta el procedimiento.

(1.5) Nuevas variables contextuales

Dmitry Yemanov

CURRENT_CONNECTION

y

CURRENT_TRANSACTION

Cada una de estas variables contextuales devuelve el identificador del sistema de la conexión activa o del contexto de transacción actual, respectivamente. El tipo de dato que devuelven es INTEGER. Se encuentran disponibles en DSQL y PSQL. Puesto que estos valores se almacenan en la página de cabecera de la base de datos, serán reasignados tras hacer un restore de la misma.

Sintaxis

```
CURRENT_CONNECTION
CURRENT_TRANSACTION
```

Ejemplos

```
SELECT CURRENT_CONNECTION FROM RDB$DATABASE;
NEW.TXN_ID = CURRENT_TRANSACTION;
EXECUTE PROCEDURE P_LOGIN(CURRENT_CONNECTION);
```

ROW_COUNT

Devuelve un entero, el número de registros involucrados en la última sentencia DML. Disponible en PSQL, en el contexto del trigger o procedure. Actualmente devuelve cero tras una sentencia SELECT.

Sintaxis

```
ROW_COUNT
```

Ejemplo

```
UPDATE TABLE1 SET FIELD1 = 0 WHERE ID = :ID;
IF (ROW_COUNT = 0) THEN
   INSERT INTO TABLE1 (ID, FIELD1) VALUES (:ID, 0);
```

Nota: esta variable no puede ser utilizada para ver los registros involucrados en EXECUTE STATEMENT.

SQLCODE y GDSCODE

Cada variable contextual devuelve un entero que es el código de error numérico de la excepción activa. Se encuentra disponible en PSQL, dentro del ámbito del bloque de manejo de excepción concreto. Ambas se evalúan como cero fuera del bloque.

La variable GDSCODE devuelve una representación numérica del código de error GDS (ISC), por ejemplo '335544349L' devolverá 335544349.

Un bloque de excepción 'WHEN SQLCODE' o 'WHEN ANY' tomará un valor diferente de cero para SQLCODE y devolverá cero para GDSCODE. Sólo un bloque 'WHEN GDSCODE' tomará valores para la variable GDSCODE (y devolverá cero para SQLCODE). Si se lanza una excepción definida por el usuario, ambas variables (SQLCODE y GDSCODE) contendrán cero, independientemente del tipo de bloque de manejo de excepción en el que se encuentren.

Sintaxis

```
SQLCODE
GDSCODE
```

Ejemplo

```
BEGIN
   WHEN SQLCODE -802 DO
     EXCEPTION E_EXCEPTION_1;
   WHEN SQLCODE -803 DO
     EXCEPTION E_EXCEPTION_2;
   WHEN ANY DO
     EXECUTE PROCEDURE P_ANY_EXCEPTION(SQLCODE);
END
```

Puede ver también la parte MEJORAS EN EL MANEJO DE EXCEPCIONES, más abajo, y el documento README.exception_handling[2] que se encuentra en la carpeta firebird2/doc/sql.extensions del CVS de Firebird.

INSERTING

UPDATING

DELETING

Tres expresiones pseudo booleanas que pueden ser invocadas para saber el tipo de operación DML que está siendo ejecutada. Se encuentran disponibles en PSQL, pero únicamente en los triggers. Están pensadas para usarlas en los triggers polivalentes (ver METADATA, más arriba).

Sintaxis

```
INSERTING
UPDATING
DELETING
```

Ejemplo

```
IF (INSERTING OR DELETING) THEN
  NEW.ID = GEN_ID(G_GENERATOR_1, 1);
```

(1.5) Mejoras en el manejo de excepciones en PSQL

Dmitry Yemanov

La sintaxis normal de una sentencia EXCEPTION en PSQL es:

```
  EXCEPTION [nombre [valor]];
```

Las mejoras de la version 1.5 permiten
1) Definir un mensaje en tiempo de ejecución para una excepción con nombre.
2) Reiniciar (relanzar) una excepción tomada dentro del ámbito del bloque de excepción
3) Obtener el código de error numérico para una determinada excepción

1) Mensajes en tiempo de ejecución

Sintaxis

```
EXCEPTION <nombre_de_la_excepcion> <texto_del_mensaje>;
```

Ejemplos

i)

```
EXCEPTION E_EXCEPTION_1 'Error!';
```

ii)

```
EXCEPTION E_EXCEPTION_2 'Wrong type for record with ID=' || new.ID;
```

2) Relanzamiento de excepciones

Nota – esto no funciona fuera del bloque de excepción.

[2] Nota del Traductor: Este documento se encuentra en inglés

Sintaxis

```
EXCEPTION;
```

Ejemplos

i)

```
BEGIN
  ...
  WHEN SQLCODE -802 DO
    EXCEPTION E_ARITH_EXCEPT;
  WHEN SQLCODE -802 DO
    EXCEPTION E_KEY_VIOLATION;
  WHEN ANY DO
    EXCEPTION;
END
```

ii)

```
WHEN ANY DO
BEGIN
   INSERT INTO ERROR_LOG (...) VALUES (SQLCODE, ...);
   EXCEPTION;
END
```

3) Códigos de error en tiempo de ejecución

Mirar SQLCODE / GDSCODE (más arriba).

(1.5) Sentencia LEAVE | BREAK

Finaliza la ejecución de un bucle, continuando la ejecución en la sentencia que haya a continuación del END que marca el final del bucle. Está disponible en WHILE, FOR SELECT y FOR EXECUTE únicamente, en otro caso, al analizar la sentencia, se lanza un error. La definición de LEAVE que hace el estándar SQL-99, hace que break pierda valor. Está disponible en triggers y también en procedimientos almacenados.

Sintaxis

```
LEAVE;
```

Ejemplos

```
(i)
BEGIN
   <sentencias>;
   IF (<condiciones>) THEN
     LEAVE;
   <sentencias>;
END

(ii)
WHILE (<condición>) DO
  BEGIN
    <sentencias>;
    WHEN ... DO
      LEAVE;
  END
```

NOTA Las sentencias LEAVE | BREAK y EXIT se pueden utilizar ahora en los triggers

(1.5) Ahora es posible incluir en los triggers sentencias PLAN válidas

Ignacio J. Ortega

Hasta ahora, un trigger que contenía una cláusula PLAN en su interior, era rechazado por el compilador. Ahora se pueden utilizar planes en los triggers.

(1.5) Bloques BEGIN..END vacíos

Dmitry Yemanov

Los bloques BEGIN..END vacíos en triggers y procedimientos almacenados (PSQL) son legales ahora. Por ejemplo, se pueden hacer rutinas como

```
CREATE TRIGGER BI_ATABLE FOR ATABLE
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
END ^
```

(1.5) Declarar y definir variables locales en una sola línea

Claudio Valderrama

Simplifica la sintaxis y permite a las variables locales ser declaradas y definidas (o inicializadas) en una sóla sentencia.

Sintaxis

```
DECLARE [VARIABLE] nombre <tipo_de_la_variable> [{'=' | DEFAULT} valor];
```

Ejemplo

```
DECLARE my_var INTEGER = 123;
```

(1.0) SELECT [FIRST (<expresion entera m>)] [SKIP (<expresion entera n>)]

(1.5) Se puede pasar cero como argumento a SELECT FIRST

FB 1.5 permite cero como argumento de FIRST. En ese caso no se devuelven registros.

Devuelve los primeros m registros del conjunto seleccionado. La cláusula opcional SKIP provocará que los primeros n registros sean desechados y el conjunto de registros devuelto contendrá m registros empezando por el n+1. En la variante más simple, m y n son números enteros, pero se puede utilizar cualquier expresión que devuelva un entero. Un identificador que evalue a un entero puede utilizarse también en GDML, al contrario que en SQL o DSQL, donde no se puede.

Los paréntesis son obligatorios para los argumentos que sean expresiones, y opcionales en otro caso.

También se pueden utilizar variables, por ejemplo, SKIP ? * FROM ATABLE devuelve los registros que quedan después de descartar los n primeros, donde n se toma de la variable "?". SELECT FIRST ? COLUMNA, COLUMNB FROM ATABLE devuelve los primeros m registros y descarta el resto.

La cláusula FIRST es también opcional, es decir, se puede incluir SKIP en una sentencia sin FIRST para obtener todos los registros excepto los indicados por SKIP.

Disponible en SQL y DSQL excepto cuando se indica lo contrario.

Ejemplos:

```
SELECT SKIP (5+3*5) * FROM MYTABLE;

SELECT FIRST (4-2) SKIP ? * FROM MYTABLE;

SELECT FIRST 5 DISTINCT FIELD FROM MYTABLE;
```

Dos problemass con SELECT FIRST

1. Esto

```
delete from TAB1 where PK1 in (select first 10 PK1 from TAB1);
```

borrará todos los registros de la tabla.¡Vaya! La subconsulta se evalúa cada 10 registros candidatos a borrarse; borrados estos, se salta a los 10 siguientes... ad infinitum, hasta que no quedan más. ¡Cuidado!

2. Una consulta como:

```
...
WHERE F1 IN ( SELECT FIRST 5 F2 FROM TABLE2 ORDER BY 1 DESC )
```

No trabaja según lo esperado, ya que la optimización que realiza el motor transforma el WHERE… IN(SELECT…) en un predicado EXIST. Es obvio que en este caso FIRST N no tiene ningún sentido:

```
WHERE EXISTS (
   SELECT FIRST 5 TABLE2.F2 FROM TABLE2
   WHERE TABLE2.F2 = TABLE1.F1 ORDER BY 1 DESC )
```

Mejoras en los conjuntos de caracteres

Añadido en 1.5

- Añadido el orden de comparación (collation) WIN1251_UA (para Ucraniano y Ruso) al conjunto de caracteres WIN1251.
- Corregidas las mayúsculas por defecto en WIN1251
- Añadido ISO_HUN (para Húngaro) en el conjunto de caracteres ISO8859_2

Nuevos conjuntos de caracteres (sin orden de comparación -collation- no binario) añadidos
Blas Rodriguez Somoza

- DOS737 PC Griego
- DOS775 PC Báltico
- DOS858 Variante del Cp850 con el carácter del Euro (€)
- DOS862 PC Hebreo
- DOS864 PC Árabe
- DOS866 MS-DOS Ruso
- DOS869 IBM Modern Griego
- WIN1255 Hebreo para Windows
- WIN1256 Árabe para Windows
- WIN1257 Báltico para Windows
- ISO8859_3 Latin 3 (Esperanto, Maltés, Pinyi, Sami, Croata y otros)
- ISO8859_4 Latin 4 (Baltico, Groenlandés, Lapón)
- ISO8859_5 Ciríllico
- ISO8859_6 Árabe
- ISO8859_7 Griego
- ISO8859_8 Hebreo
- ISO8859_9 Turco
- ISO8859_13 Báltico

Añadido en 1.0

- Se añade un orden de comparación no sensible a mayúsculas para Húngaro, desarrollado y testeado por Sandor Szollosi (ssani@freemail.hu).

- Firebird ahora soporta el conjunto de caracteres ISO8859-2 (para Checo).

EXTENSIONES DEL LENGUAJE ARRASTRADAS DESDE FIREBIRD 1.0.x

Las siguientes extensiones del lenguaje, introducidas en Firebird 1.0.x, se exponen aquí para su comodidad.

(1.0) CURRENT_USER y CURRENT_ROLE

Estas dos variables contextuales nuevas han sido añadidas para referenciar el USER y (si es el caso[1]) el ROLE de la conexión actual.

```
CREATE GENERATOR GEN_USER_LOG;
CREATE DOMAIN INT_64 AS NUMERIC(18,0);
COMMIT;
CREATE TABLE USER_LOG(
  LOG_ID  INT_64 PRIMARY KEY NOT NULL,
  OP_TIMESTAMP TIMESTAMP,
  LOG_TABLE VARCHAR(31),
  LOG_TABLE_ID INT_64,
  LOG_OP CHAR(1),
  LOG_USER VARCHAR(8),
```

```
  LOG_ROLE VARCHAR(31));

COMMIT;

CREATE TRIGGER ATABLE_AI FOR ATABLE
ACTIVE AFTER INSERT POSITION 0 AS
BEGIN
  INSERT INTO USER_LOG VALUES(
    GEN_ID(GEN_USER_LOG, 1),
    CURRENT_TIMESTAMP,
    'ATABLE',
    NEW.ID,
    'I',
    CURRENT_USER,
    CURRENT_ROLE);
END
```

CURRENT_USER es un sinónimo DSQL de la palabra clave USER que aparece en el estándar SQL. Son idénticos. CURRENT_USER no supone ninguna ventaja sobre USER.

[1] Si usted insiste en utilizar una base de datos de las versiones 4.x o 5.1 de Interbase, ROLE no existirá en ella, por lo tanto current_role será NONE (como ordena el estandar SQL en ausencia de un ROLE explícito) aunque se haya pasado un rol en la conexión. Si se utiliza IB 5.5, IB 6 o Firebird, el ROLE pasado se verifica. Si el rol no existe, se asigna NONE, sin devolver ningún error.

Esto significa que en FB nunca se obtendrá un ROLE inválido al llamar a CURRENT_ROLE, ya que será asignado NONE. Esto contrasta con IB, donde el valor falso se arrastra internamente, aunque no es visible para el SQL.

(1.0) DROP GENERATOR

Permite eliminar de la base de datos los generadores que no tengan dependencias. El nombre podrá ser utilizado de nuevo tras el siguiente RESTORE. Disponible en SQL y DSQL.

```
DROP GENERATOR <nombre del generador>;
```

(1.0) GROUP BY UDF

Ahora es posible realizar consultas SELECT agregadas, agrupando por el resultado de una UDF.
Por ejemplo:

```
select strlen(rtrim(rdb$relation_name)), count(*) from rdb$relations
group by strlen(rtrim(rdb$relation_name))
order by 2
```

Un efecto colateral de los cambios que permiten agrupar por UDF's es que, mientras que antes no se podía llamar a las funciones internas de Firebird en la cláusula GROUP BY, ahora, escondiendo la operación dentro de una falsa UDF, usted puede hacer esto:

```
select count(*)
from rdb$relations r
group by bin_or((select count(rdb$field_name) from rdb$relation_fields f
where f.rdb$relation_name = r.rdb$relation_name),0)
```

(1.0) RECREATE PROCEDURE

Este nuevo comando DDL permite crear un nuevo procedimiento almacenado con el mismo nombre que uno ya existente, reemplazando al viejo, sin necesidad de borrarlo previamente. La sintaxis es idéntica a CREATE PROCEDURE.
Disponible en SQL y DSQL.

(1.0) RECREATE TABLE

Este nuevo comando DDL permite crear una nueva estructura para una tabla existente sin necesidad de borrarla primero. La sintaxis es identica a la de CREATE TABLE.

Observe que RECREATE TABLE no conserva los datos de la tabla vieja.

Disponible en SQL y DSQL.

(1.0) SUBSTRING(<*expresion texto*> FROM <*posicion*> [FOR <*longitud*>])

Función interna que implementa la función ANSI SQL SUBSTRING(). Devolverá todos los bytes comenzando por el que se encuentre en <posicion> hasta el final de la expresión texto. Si se especifica la opción FOR <longitud>, devolverá <longitud> bytes o los que haya hasta el final de la expresión texto.

El primer argumento puede ser cualquier expresión, constante o identificador que se evalúe como cadena de caracteres (string).
<*pos*> debe evaluarse como un entero (integer).
<*pos*> comienza en 1, como otros comandos SQL.
Ni <*pos*> ni <*length*> pueden ser ser parámetros de la consulta.

Puesto que <posicion> y <longitud> son posiciones de bytes, el identificador puede ser un blob binario, o un blob de texto sub_type 1 creado bajo un conjunto de caracteres donde cada carácter ocupe un byte. La función actualmente no maneja blobs de conjuntos de caracteres Chino (2 bytes por carácter como máximo) o Unicote (3 bytes por carácter como máximo). Para un argumento de tipo string (a diferencia de un blob), la función trabaja con CUALQUIER conjunto de caracteres.

Disponible en SQL y DSQL.

```
UPDATE ATABLE
SET COLUMNB = SUBSTRING(COLUMNB FROM 4 FOR 99)
WHERE ...
```

Por favor, diríjase también a la sección Funciones Externas (UDFs) a continuación, para conocer los detalles de los cambios y mejoras en las funciones externas substring de la librería UDF estándar.

(1.5) Mejoras en los comentarios de una sola línea

Dmitry Yemanov

Los comentarios de una sola línea, pueden empezar en cualquier posición de la misma, no sólo al principio.
Por tanto, en 1.5 el marcador "--" se puede usar para hacer un comentario al final de una sentencia en un script, procedimiento almacenado, trigger o sentencia DSQL. Puede ser usado de este modo para comentar partes de las sentencias. Todos los caracteres desde el marcador "--" hasta el siguiente retorno de carro o salto de línea serán ignorados.

...

```
WHERE COL1 = 9 OR COL2 = 99 -- OR COL3 = 999
```

(1.0) Nuevo marcador de comentarios

Claudio Valderrama

Para utilizarlo en scripts, DSQL, procedimientos almacenados y triggers.

Ejemplo

```
-- Esto es un comentario
```

Este nuevo marcador puede ser utilizado para comentar una línea de código en un script, sentencia DDL/DML, procedimiento almacenado o trigger.

La forma en la que se ignoran los caracteres es la siguiente:

1. Si la primera pareja de caracteres que encuentra a continuación de un final de línea (LF en Linux/Unix, CRLF en Windows) es '--', la ignora
2. Continua ignorando caracteres hasta el siguiente final de línea

Este marcador no está pensado para mezclarlo con los comentarios por bloques (/* un comentario */). En otras palabras, no use el estilo '--' de comentarios dentro de un comentario por bloques y no use el estilo de comentario por bloque dentro de una línea comentada con '--'.

SESIONES INTERACTIVAS ISQL : Guarde esto en mente cuando trabaje en una sesión interactiva con **isql**. **isql** acepta trozos de sentencias en diferentes segmentos, mostrando el prompt 'CON>' hasta que recibe el símbolo de final de sentencia (normalmente '**;**'). Si usted escribe '--' al inicio de la siguiente línea, la lógica para ignorar finalizará en la marca de fin de línea que escriba en la pantalla o fichero de salida cuando se pulse INTRO. Esto puede provocar errores si continúa escribiendo pensando que lo que escriba será ignorado.

El problema con ISQL surge porque tiene sus propios comandos que deben ser procesados solo por ISQL. Si no son reconocidos por una mala colocación de '--' entonces se envían al motor. Obviamente, el motor no entiende los comandos ISQL SET y SHOW y los rechaza.

(1.0) Alter Trigger ya no incrementa el número de cambios de la tabla

Cuando el contador de cambios del metadata para una tabla cualquiera alcanza el máximo de 255, la base de datos se vuelve inutilizable. Se necesita hacer Backup/Restore para resetear los contadores y volver a dejar la base de datos operativa. La intención de esta característica es forzar a la base de datos a reestructurarse cuando la estructura de las tablas ha sufrido gran cantidad de cambios, para no perjudicar el rendimiento del motor.

Previamente, cada vez que un trigger era activado o desactivado (set ACTIVE|INACTIVE) por una sentencia ALTER TRIGGER, el contador de cambios asociado a la tabla se incrementaba. Esto hacía que no se pudiera desactivar y reactivar el código de los triggers frecuentemente y de forma reiterada, ya que esto provocaba que el contador de cambios aumentase rápidamente.

Nuevas palabras reservadas

Las siguientes **palabras clave nuevas** deben ser añadidas a la lista de palabras reservadas publicadas por Interbase 6.0.1.

BIGINT (1.5) CASE (1.5) CURRENT_CONNECTION (1.5)

CURRENT_ROLE	CURRENT_TRANSACTION (1.5)	CURRENT_USER
RECREATE	ROW_COUNT (1.5)	RELEASE
SAVEPOINT		

Las siguientes palabras están reservadas para usarlas en el futuro:

ABS	BOOLEAN	BOTH
CHAR_LENGTH	CHARACTER_LENGTH	FALSE
LEADING	OCTET_LENGTH	TRIM
TRAILING	TRUE	UNKNOWN

Las siguientes palabras estaban reservadas en Firebird 1.0 y no lo están en Firebird 1.5:

BREAK	DESCRIPTOR	FIRST
IIF	SKIP	SUBSTRING

Las siguientes palabras no reservadas, son reconocidas en 1.5 como palabras clave en sus respectivos contextos estructurales:

COALESCE	DELETING	INSERTING
LAST	LEAVE	LOCK
NULLIF	NULLS	STATEMENT
UPDATING	USING	

Las siguientes **palabras clave nuevas en InterBase 6.5 y 7** (no reservadas en Firebird) deben ser también tratadas como si lo estuvieran, por compatibilidad:

BOOLEAN	FALSE	GLOBAL
PERCENT	PRESERVE	ROWS
TEMPORARY	TIES	TRUE

Características del ISQL

Capacidad en el shell del isql "tipo readline"

Mark O'Donohue

Se incorpora un histórico de comandos (similar a readline de Unix) en el shell de isql. Ahora se pueden utilizar las teclas flecha arriba y flecha abajo para moverse por los comandos enviados en las sesiones de isql.

Funciones definidas por el usuario

En ib_udf

rpad (*instring, length, padcharacter***)**

Juan Guerrero

Rellena por la derecha la cadena *instring* añadiendo *padcharacter* hasta alcanzar la longitud *length*. La cadena *instring* puede tener hasta 32766 bytes. La longitud no debe ser más grande de 32765 bytes.

Declaración

```
DECLARE EXTERNAL FUNCTION rpad
        CSTRING(80), INTEGER, CSTRING(1)
        RETURNS CSTRING(80) FREE_IT
        ENTRY_POINT 'IB_UDF_rpad' MODULE_NAME 'ib_udf';
```

lpad (*instring, length, padcharacter***)**

Juan Guerrero

Rellena por la izquierda la cadena *instring* añadiendo *padcharacter* hasta alcanzar la longitud *length*. La cadena *instring* puede tener hasta 32766 bytes. La longitud no debe ser más grande de 32765bytes.

Declaración

```
DECLARE EXTERNAL FUNCTION lpad
        CSTRING(80), INTEGER, CSTRING(1)
        RETURNS CSTRING(80) FREE_IT
        ENTRY_POINT 'IB_UDF_lpad' MODULE_NAME 'ib_udf';
```

log (*x, y***)**

Paul Vinkenoog

Esta función tiene un viejo error, consistente en que los argumentos x e y estaban por error al revés. Debería devolver el logaritmo en base x de y, pero devolvía el logaritmo en base y de x. Esto ha sido corregido. Si usted la utilizaba en sus aplicaciones anteriormente, ¡POR FAVOR, REVISE EL CÓDIGO DE LAS MISMAS! pueden devolver resultados incorrectos; o puede ser que usted ya haya invertido el orden de los parámetros para que el cálculo se haga correctamente.

En fbudf

1. Las funciones *NVL y *NULLIF permanecen por compatibilidad con versiones anteriores, pero han perdido utilidad debido a la introducción de las nuevas funciones internas CASE, COALESCE y NULLIF.
2. Debe tenerse en cuenta que fbudf no puede manejar cadenas de más de 32Kb – 1byte de longitud. Este límite puede tener efectos perjudiciales cuando las cadenas que se pasan como argumentos a las UDFs se concatenan antes de ser enviadas a la UDF. Si la suma de la longitud supera este límite, el comportamiento es indefinido. La función puede devolver un resultado absurdo o realizar una operación ilegal.
3. Si Ud. está migrando una base de datos creada en Firebird 1.0.x y ha declarado las funciones **truncate** y **Round** de fbudf, estas declaraciones no funcionarán con Firebird 1.5 debido a que los nombres de los puntos de entrada han cambiado. Deberá eliminar las funciones (drop) y redeclararlas, usando las declaraciones del script fbudf.sql que está en el directorio /UDF de Firebird 1.5.

Nuevo archivo de Configuración – firebird.conf

El directorio raíz de Firebird

El directorio raíz de su instalación de Firebird es un valor que tanto en la instalación como durante el funcionamiento del servidor es usado de distintas formas, también los parámetros de configuración y los clientes dependen de él. Dado que existen varias maneras de indicar al servidor cual es el valor de este atributo, los programadores y administradores de sistema deben tener en cuenta la lista de precedencia que sigue el servidor en su arranque, para determinarlo correctamente.

Win32 Superserver y Classic (tanto servidor como cliente):
1) Variable de entorno FIREBIRD
2) Parámetro RootDirectory en firebird.conf
3) Registro:
   HKLM\SOFTWARE\Firebird Project\Firebird Server\Instances\DefaultInstance
   busca la clave DefaultInstance.
4) El directorio padre del directorio donde se encuentra el ejecutable del servidor

Win32 Embedded (integrado):
1) Variable de entorno FIREBIRD
2) Parámetro RootDirectory en firebird.conf
3) El directorio donde se encuentra fbembed.dll (renombrada como fbclient.dll)

Linux Classic:
1) Variable de entorno FIREBIRD
2) Parámetro RootDirectory en firebird.conf
3) Ruta de instalación por defecto (/opt/firebird)

Linux Superserver:
1) Variable de entorno FIREBIRD
2) Parámetro RootDirectory en firebird.conf
3) El directorio padre del directorio donde se encuentra el ejecutable del servidor (en caso de ser soportado, se accede mediante el enlace simbólico "/proc/self/exe")
4) Ruta de instalación por defecto (/opt/firebird)

Parámetros

Para la mayoría de parámetros son aplicables valores por defecto. Los nombres de parámetros y sus valores son sensibles a mayúsculas en Linux pero no en Windows. Para establecer un parámetro a un valor distinto al valor por defecto, borre la señal de comentario (#) y modifique el valor. Puede modificar la configuración mientras el servidor está en funcionamiento. Para activar los cambios en la configuración, debe parar y reiniciar el servicio.
Las entradas son del tipo:

nombre_parámetro *valor*

- nombre_parámetro es una cadena sin espacios que indica una propiedad del servidor a configurar

- *valor* es un número, Booleano (1=Cierto, 0=Falso) o cadena que especifica el valor del parámetro

Parámetros relacionados con el Sistema de Archivos

RootDirectory

Cadena, la ruta absoluta al directorio raíz en el sistema de archivos local. Se debe mantener comentada a menos que desee forzar al proceso de arranque a sustituir la ruta al directorio raíz de la instalación del servidor Firebird, que de otra forma se detecta por si sola.

DatabaseAccess

Soporta la característica de *alias de bases de datos*. En versiones previas, el servidor podía conectar a cualquier base de datos en su sistema de archivos local y que fuera accedida por la aplicación pasando la ruta absoluta de su archivo en el sistema de archivos. Este parámetro proporciona la opción de restringir el acceso del servidor únicamente a las bases de datos con alias, o a las bases de datos que se encuentran localizadas dentro de un sub-árbol específico del sistema de archivos.
DatabaseAccess puede valer None, Restrict o Full.
Full (valor por defecto) permite que se pueda acceder a cualquier archivo de base de datos en el sistema de archivos local
None permite al servidor de conectarse únicamente a las bases de datos listadas en **aliases.conf**.
Restrict le permite configurar la ubicación de las bases de datos a las que se podrá conectar mediante una lista de sub-árboles del sistema de archivos. Debe proporcionar una lista de uno o más directorios raíz de sub-árboles, separados por punto y coma (;), para definir las posibles ubicaciones.
Por ejemplo,
Unix: `/db/databases;/userdir/data`
Windows: `D:\data`
Las rutas relativas se tratan como relativas a la ruta que el servidor reconoce como su directorio raíz. Por ejemplo, en Windows, si el directorio raíz es C:\Archivos de Programa\Firebird, entonces el siguiente valor restringirá el servidor a poder acceder únicamente a los archivos de base de datos que se encuentren bajo el directorio C:\Archivos de Programa\Firebird\DatosUsuario;

```
DatabaseAccess = Restrict DatosUsuario
```

NOTA Espejado (shadowing) de bases de datos – el manejo actual de DatabaseAccess contiene un error que implica que se debe usar la opción Restrict si se hace espejado de cualquier base de datos en el servidor.

ExternalFileAccess

Se llamaba *external_file_directory* en isc_config/ibconfig pero la sintaxis ha cambiado.

Proporciona tres niveles de seguridad acerca de los ARCHIVOS EXTERNOS (EXTERNAL FILES, archivos de texto en formato de ancho fijo que son accesibles como tablas de base de datos). El valor es una cadena, que puede ser None, Full o Restrict.
None (valor por defecto) inhabilita cualquier uso de archivos externos en su servidor.
Restrict Proporciona la posibilidad de restringir la ubicación de los archivos externos para los accesos de bases de datos dentro de unos sub-árboles específicos. Debe proporcionar una lista con uno o más directorios raíz de sub-árboles, separados por punto y coma (;), dentro de los que deben almacenarse los archivos externos.
Por ejemplo,
Unix: `/db/extern;/mnt/extern`
Windows: `C:\TablasExternas`
Las rutas relativas se tratan como relativas a la ruta que el servidor reconoce como su directorio raíz. Por ejemplo, en Windows, si el directorio que el servidor en funcionamiento reconoce como el directorio raíz de la instalación de Firebird es C:\Archivos de Programa\Firebird, entonces el siguiente valor restringirá el servidor a poder acceder únicamente a los archivos localizados dentro de C:\Archivos de Programa\Firebird\DatosUsuario\TablasExternas:

```
ExternalFileAccess = Restrict DatosUsuario\TablasExternas
```

Full permite que los archivos externos sean accesibles en cualquier lugar del sistema de archivos

Vea el **ATENCIÓN** bajo la próxima entrada, UdfAccess.

UdfAccess

Se llamaba *external_function_directory* en isc_config/ibconfig pero la sintaxis ha cambiado.

No solo se sustituye el nombre del parámetro anterior, sino también la forma en que se indican sus valores. Las modificaciones tienen el propósito de activar distintos niveles opcionales de protección para los módulos de las librerías de funciones definidas por el usuario, objetivos tradicionales de ataques externos maliciosos. UdfAccess debe ser None, Restrict o Full.

Restrict (valor por defecto) mantiene la funcionalidad proporcionada por el parámetro **external_function_directory** en Firebird 1.0, de restringir la ubicación de las librerías externas cargables a localizaciones específicas del sistema de archivos. Debe proporcionar una lista con uno o más directorios, separados por punto y coma (;), dentro y por debajo de esos directorios se deben almacenar las UDF, filtros BLOB y definiciones de conjuntos de carácter.
Por ejemplo,
Unix: /db/externos;/mnt/externos
Windows: C:\ModulosExternos
Las rutas relativas se tratan como relativas a la ruta que el servidor reconoce como su directorio raíz. Por ejemplo, en Windows, si el directorio raíz de la instalación de Firebird es C:\Archivos de Programa\Firebird, entonces el siguiente valor restringirá al servidor a acceder únicamente a las librerías externas ubicadas dentro de C:\Archivos de Programa\Firebird\userdata\ModulosExternos :

ExternalFileAccess = Restrict userdata\EsternalModules

None inhabilita cualquier uso de librerías externas de funciones definidas por el usuario.

Full permite que se pueda acceder a librerías externas ubicadas en cualquier lugar del sistema de archivos.

ATENCIÓN :: Evite configurar sub-árboles propios para UdfAccess y ExternalFileAccess que compartan directorios. La configuración por defecto es segura. Si realiza su propia configuración y no establece sub-árboles diferenciados para ellos, el servidor puede ser fácilmente forzado a ejecutar código no autorizado. Un ejemplo de lo que debe evitar :

UdfAccess = UDF; /dir_erroneo
ExternalFileAccess = /externos; /dir_erroneo/archivos

En esta configuración UdfAccess y ExternalFileAccess comparten un sub-árbol, /dir_erroneo/archivos, donde alguien puede almacenar su archivo externo /dir_erroneo/archivos/hackudf.so y ejecutar su propio código en el sistema atacado.

Parámetros relacionados con recursos

Máscara CpuAffinity

Se llamaba *cpu_affinity* en isc_config/ibconfig

En Firebird SuperServer para Windows, existe un problema con el sistema operativo que provoca el cambio continuo entre procesadores en sistemas SMP del proceso del Servidor. Esto degrada su rendimiento. Este parámetro se puede usar en sistemas SMP con Windows para fijar el proceso del Firebird SuperServer a una única CPU.

> ATENCIÓN Los Servidores Firebird hasta la versión 1.5 (ésta incluida) pueden no soportar la característica Hyperthreading presente en las placas madre más modernas. Para evitar problemas de balanceo, puede necesitar desactivar el hyperthreading a nivel de la BIOS.

CpuAffinityMask toma un valor entero, la máscara para indicar una CPU.

Ejemplo

CpuAffinityMask = 1
solo se ejecuta en la primera CPU (CPU 0).

CpuAffinityMask = 2
solo se ejecuta en la segunda CPU (CPU 1).

CpuAffinityMask = 3
se ejecuta sobre la primera y la segunda CPU.

Calculando el valor para la máscara
Puede usar este flag para fijar Firebird a un único procesador, o (en Classic server) a cualquier combinación de las CPUs instaladas en el sistema.
Considere las CPUs como un array M numerado de 0 a n-1, siendo n el número de procesadores instalados. Cada posición i en el array representa una CPU, un 1 indica que la CPU está seleccionada y un 0 que no. El valor A de la máscara correspondiente a esa selección, equivale al valor del número binario representado por el array M. Use la siguiente fórmula para calcular el valor A de la máscara:

$M_i = 2^I$
$A = M_1 + M_2 + M_3 . . .$

Por ejemplo, para seleccionar el primer y el cuarto procesador (el procesador 0 y el procesador 3) realice el siguiente cálculo:

$A = 2^0 + 2^3 = 1 + 8 = 9$

DeadlockTimeout

Se llamaba *deadlock_timeout* en isc_config/ibconfig

Número de segundos (entero) que el gestor de bloqueos esperará después de la aparición de un conflicto antes de purgar todos los bloqueos de procesos muertos y realizar un ciclo posterior de detección de deadlocks. Normalmente el motor detecta instantáneamente los deadlocks. El timeout de deadlocks solo se dispara cuando algo va mal.

El valor por defecto de 10 segundos es adecuado para la mayoría de las situaciones. Configurar un valor menor no aumenta necesariamente la velocidad a la que los deadlocks problemáticos devuelven una excepción de conflicto. Si es demasiado bajo, el efecto puede ser escaneos extra innecesarios que degraden el rendimiento del sistema.

DefaultDbCachePages

Se llamaba *database_cache_pages* en isc_config/ibconfig

Entero, el valor por defecto varia en función del Servidor. Número de páginas de la base de datos que se mantienen en memoria, para cada base de datos. El valor configurado puede ser sobrescrito a nivel de la base de datos.
El valor por defecto para SuperServer es 2048 páginas. Para Classic es 75.
SuperServer y Classic utilizan de forma distinta la caché. SS comparte su caché para ser usada por todas las conexiones; Classic asigna una caché estática para cada conexión.

EventMemSize

Entero, representa el número de bytes de memoria reservada para el gestor de eventos. El valor por defecto es 65536 (64 Kb).

LockAcquireSpins

Se llamaba anteriormente *lock_acquire_spins*

Solo se usa en maquinas SMP ejecutando el servidor Classic. En el servidor Classic, solo un proceso de cliente puede acceder en un momento dado a la tabla de bloqueos. Un mutex controla el acceso a la tabla de bloqueos. Los procesos de cliente pueden solicitar el mutex condicionalmente o incondicionalmente. En caso de ser condicional, una solicitud denegada debe ser reintentada. Si es incondicional, la solicitud esperará hasta ser satisfecha. LockAcquireSpins establece el número de reintentos que se harán en las solicitudes condicionales al mutex.
Entero. El valor por defecto es 0 (incondicional). No hay un valor mínimo o máximo recomendados.

LockHashSlots

Se llamaba *lock_hash_slots* en isc_config/ibconfig

Utilice este parámetro para ajustar la lista de hash de bloqueos. Bajo fuertes cargas, se puede mejorar el throughput (caudal de transferencia) elevando el número de slots en la lista hash para dispersar la lista en cadenas de hash más cortas. Se recomienda el uso de números enteros primos. El valor por defecto es 101.

LockGrantOrder

Se llamaba *lock_grant_order* en isc_config/ibconfig

Cuando una conexión quiere bloquear un objeto, obtiene un bloque de solicitud de bloqueo en el que se especifica el objeto y el nivel de bloqueo solicitado. Cada objeto bloqueado tiene un bloque de bloqueo. Las solicitudes de bloqueos se conectan a estos bloques de bloqueo ya sea como solicitudes que han sido aceptadas, o como solicitudes pendientes.
El parámetro LockGrantOrder es un Booleano. El valor por defecto (1=Cierto) indica que los bloqueos se aceptarán en base a un Primero-en-llegar-Primero-en-servirse. La configuración de Falso (0) emula el comportamiento de Interbase v3.3, aceptando el bloqueo tan pronto como esté disponible. Puede provocar "inanición" en solicitudes de bloqueo.

LockMemSize

Este parámetro entero representa el número de bytes de memoria compartida asignados para el gestor de bloqueos. Para el servidor Classic, LockMemSize indica la asignación inicial, que aumentará dinámicamente cuando se agote la memoria ("*Lock manager is out of room*"). Si hay muchas conexiones o grandes cachés de página, incrementar este parámetro evitará esos errores.
En SuperServer, la memoria asignada para el gestor de bloqueos no aumenta.
El tamaño por defecto en Linux y Solaris es 98304 bytes (96 kb). En Windows es 262144 (256 kb).

LockSemCount

Parámetro entero, especifica el número de semáforos disponibles para comunicaciones entre procesos (IPC). El valor por defecto es 32. Modifique este parámetro en entornos non-threading para elevar o disminuir el número de semáforos disponibles.

SortMemBlockSize

Este parámetro le permite configurar en bytes, el tamaño de cada bloque de memoria usado por el modulo de ordenación en memoria. El valor por defecto es 1 Mb, lo puede modificar a cualquier tamaño hasta el máximo definido por el valor del parámetro SortMemUpperLimit (vea más abajo).

SortMemUpperLimit

La cantidad máxima de memoria, en bytes, que puede ser asignada por el modulo de ordenación en memoria. El valor por defecto es 67108864 bytes (64 Mb) para SuperServer y 8388608 (8 Mb) para el servidor Classic.

> ATENCIÓN Para Classic tenga en cuenta que incrementar tanto el tamaño de bloque como el límite máximo, afecta a cada instancia de conexión de cliente, y en consecuencia, disparará el consumo de memoria del servidor.

Parámetros relacionados con las Comunicaciones

ConnectionTimeout

Se llamaba *connection_timeout* en isc_config/ibconfig

Número de segundos a esperar antes de abandonar un intento de conexión. Por defecto 180.

DummyPacketInterval

Se llamaba *dummy_packet_interval* en isc_config/ibconfig

Entero, es el número de segundos que el servidor esperará con una conexión de cliente sin tráfico, antes de enviar un paquete de control para comprobar que el cliente sigue activo.
NO UTILIZE ESTA OPCIÓN en un servidor Win32 con clientes TCP/IP. Provoca un incremento continúo en el uso de memoria no paginada del kernel que puede bloquear o colgar Windows en el lado cliente tal como se describe en:

http://support.microsoft.com/default.aspx?kbid=296265

Exceptuando TCP/IP sobre Win32, es la única forma de detectar y desconectar clientes inactivos cuando se usan los protocolos NamedPipes (NetBEUI), XNET o IPC. No se han detectado problemas en sistemas POSIX.

Normalmente Firebird utiliza la opción a nivel de socket SO_KEEPALIVE para seguir la pista de las conexiones activas. Si no le gusta su timeout por defecto de dos horas, puede ajustar apropiadamente su configuración del Sistema Operativo:

- En SO's tipo UNIX, modifique el contenido de /proc/sys/net/ipv4/tcp_keepalive_*.
- En Windows, siga las instrucciones de este artículo:

  http://support.microsoft.com/default.aspx?kbid=140325

El valor por defecto debe ser 0, y no 60 que era el antiguo valor por defecto en Firebird 1.0 y algunos de los 1.5 release candidates. Una configuración de 60 debe considerarse el valor por defecto en los sistemas en que necesite hacer uso de este envío de paquetes de control.

RemoteServiceName

Valor por defecto = gds_db

RemoteServicePort

Valor por defecto = 3050
Estos dos parámetros proporcionan la capacidad de sobrescribir ya sea el nombre de servicio TCP/IP o el puerto TCP/IP usado para escuchar solicitudes de conexiones de clientes de bases de datos, si uno de ellos difiere de los valores por defecto en la instalación (gds_db/tctp 3050)

Modifique una de las dos entradas, no las dos. Primero se verifica que exista una entrada con el valor RemoteServiceName en el archivo services. Si se encuentra, se usará el puerto indicado en el archivo services, en caso contrario se usará el puerto RemoteServicePort con un valor por defecto de 3050.

NOTA Si se proporciona un número de puerto en la cadena de conexión TCP/IP, siempre tendrá prioridad sobre RemoteServicePort.

RemoteAuxPort

El comportamiento heredado de Interbase, de enviar por la red mensajes de notificación de eventos a través de puertos TCP/IP seleccionados al azar, ha sido una fuente continua de errores de red y conflictos con firewalls, llegando al extremo de causar el cuelgue del servidor bajo ciertas condiciones. Este parámetro le permite configurar un único puerto TCP para todo el tráfico de notificación de eventos.
El valor por defecto (0) mantiene el comportamiento tradicional de puertos al azar. Para dedicar un puerto específico para la notificación de eventos, configure un entero que sea un número de puerto disponible.

RemoteBindAddress

Los clientes pueden conectarse por defecto desde cualquier interfaz de red cuyo tráfico sea aceptado por el host servidor. Este parámetro le permite enlazar el servicio Firebird a las solicitudes que lleguen por un único NIC, rechazando las solicitudes de conexión que lleguen por cualquier otra interfaz de red. Esto le permitirá evitar problemas en algunas subredes en las que el servidor gestiona tráfico a través de varias NICs.
Cadena, formando una dirección IP valida. El valor por defecto (no enlazado) es sin valor.

TcpRemoteBufferSize

El motor lee por delante del cliente y puede enviar varias filas de datos en un único paquete. Cuanto mas grande es el tamaño de paquete, más filas se envían por transferencia. Use este parámetro –con cautela y total conocimiento de sus efectos en el rendimiento de la red- si desea aumentar o reducir el tamaño del paquete TCP/IP usado para enviar y recibir buffers. Afecta tanto al cliente como al servidor.
Su valor es un entero (tamaño del paquete en bytes) entre 1448 y 32768. El valor por defecto es 8192.

Parámetros específicos de POSIX

LockSignal

Parámetro entero, signal de UNIX usada para comunicaciones entre procesos. Valor por defecto: 16

RemoteFileOpenAbility

USELA SOLO CON EXTREMA CAUTELA

Parámetro booleano, que si se establece a Cierto, permite al motor abrir archivos ubicados en particiones montadas de sistemas de archivos de red (NFS). Debido a que el sistema de archivos se encuentra más allá del control del sistema local, se trata de una *característica muy peligrosa* que no debe usarse con la intención de abrir para lectura/escritura una base de datos cuya integridad sea importante para usted.
El valor por defecto es 0 (Falso, desactivado) y lo debe mantener así a menos que sea muy consciente de sus efectos.

TcpNoNagle

Se llamaba *tcp_no_nagle* en isc_config/ibconfig

En Linux, por defecto, la librería de sockets optimiza las escrituras físicas mediante un buffering antes de enviar los datos, utilizando un algoritmo interno (implementado como la opción TCP_NODELAY de la conexión a socket) conocido como el Algoritmo de Nagle. Fue diseñado para evitar problemas con unos pequeños paquetes, llamados tinygramas, en redes lentas.
Por defecto, se activa TCP_NODELAY (valor 0) al instalar Firebird SuperServer en Linux. En redes lentas, desactivándolo puede aumentar la velocidad de red. Note la doble negación –configurar el parámetro a Cierto para desactivar TCP_NO_DELAY y a Falso para activarlo.
En las versiones hasta v.1.5 (ésta incluida), esta característica solo está activa en SuperServer.

Parámetros específicos de Windows

CreateInternalWindow

El protocolo local en Windows utiliza una ventana oculta para las comunicaciones entre procesos, entre el cliente y el servidor. Esta ventana IPC se crea en el arranque del servidor cuando CreateInternalWindow es Cierto (1, por defecto). Configúrelo a 0 (Falso) para ejecutar el servidor sin la

ventana con lo que se desactiva el protocolo local. Con el protocolo local desactivado es posible ejecutar varias instancias simultáneas del servidor.

DeadThreadsCollection

Parámetro entero para configurar el planificador de procesos en Windows, establece el número de ciclos de cambios de prioridad (vea más abajo PrioritySwitchDelay) que el planificador ejecutará antes de eliminar (o cerrar) un thread (proceso, hilo de ejecución).
La eliminación (o cierre) inmediata de hilos de ejecución activos necesita un semáforo y una llamada de bloqueo, lo que conlleva una sobrecarga considerable. En cambio el planificador de procesos mantiene los threads en un pool. Cuando un thread completa su tarea, se marca como inactivo. El thread inactivo es eliminado (o cerrado) después de *n* iteraciones del bucle del planificador, siendo *n* el valor del parámetro DeadThreadsCollection.
Para servidores que gestionen cantidades muy elevadas de conexiones –del orden de miles- el valor del parámetro se debe aumentar respecto al valor por defecto de 50.

GuardianOption

Parámetro booleano utilizado por los servidores Windows para determinar si el Guardian debe reiniciar el servidor cada vez que éste finaliza anormalmente. El valor por defecto es que lo haga (1=Cierto). Para desactivar el reinicio, configure este parámetro a Falso (0).

IpcMapSize

Se llamaba *server_client_mapping* en ibconfig

Tamaño en bytes de una sección del archivo mapeado en memoria del cliente utilizado para las comunicaciones entre procesos (IPC) en las conexiones del tipo conexiones locales de Windows. No tiene equivalentes en otras plataformas. Entero, de 1024 a 8192. El valor por defecto es 4096. Incrementar el IpcMapSize puede aumentar el rendimiento cuando se recuperan grandes conjuntos de datos (con muchas filas o muchos datos por fila), así como los que devuelven BLOBs gráficos.

> NOTA Este valor ya no se puede modificar desde la ventana emergente que aparece al pulsar en del icono del Guardian que hay en la barra de tareas.

IpcName

Valor por defecto: FirebirdIPI
El nombre del área de memoria compartida utilizada como canal de transporte en el protocolo local. El valor por defecto en la versión 1.5 –FirebirdIPI- es incompatible con las versiones anteriores de Firebird y con Interbase®. En caso de ser necesario, configure el valor InterBaseIPI para restaurar la compatibilidad.

MaxUnflushedWrites

Este parámetro se introdujo en la versión 1.5 para corregir un bug en el sistema operativo de los servidores Windows, en los que las escrituras asíncronas nunca se pasaban al disco hasta ocurrir un shutdown controlado. (Las escrituras asíncronas no están soportadas en Windows 9x o ME). Por lo que en sistemas 24/7 (servicio ininterrumpido 24 horas 7 días), las escrituras asíncronas nunca se trasladaban a disco.
Este parámetro determina la frecuencia con la que las paginas en memoria se trasladan a disco cuando ForcedWrites está desactivado (las escrituras asíncronas están activadas). Su valor es un entero que establece el número de páginas que deben permanecer en memoria antes que se active una señal para que en el commit de la siguiente transacción se trasladen a disco dichas páginas. El valor por defecto es 100 en las instalaciones Windows y -1 (desactivado) en las instalaciones para otras plataformas.

Si se llega al final del ciclo del MaxUnflushedWriteTime (vea abajo) antes que el contador de páginas en memoria alcance el MaxUnflushedWrites, se activa inmediatamente la señal de flush (traslado de páginas) y el contador de páginas pendientes se reinicializa a cero.

MaxUnflushedWriteTime

Este parámetro determina el tiempo máximo que las páginas permanecen en memoria para escrituras asíncronas antes de ser trasladadas a disco cuando Forced Writes está desactivado (las escrituras asíncronas están activadas). Este parámetro es un entero que establece el intervalo, en segundos, entre el último flush a disco y la activación de una señal para realizar un flush en el siguiente commit de una transacción. El valor por defecto en las instalaciones Windows son 5 segundos y -1 (desactivado) en las instalaciones para otras plataformas.

PrioritySwitchDelay

Parámetro entero para la configuración del planificador de procesos en Windows, establece el tiempo, en milisegundos, que debe pasar antes de que la prioridad de un thread inactivo se reduzca a BAJA o la prioridad de un thread activo aumente a ALTA. Una iteración de esta secuencia de cambio representa un ciclo del planificador de tareas.
El valor por defecto son 100 ms, escogido en función de pruebas sobre procesadores PIII/P4. Para procesadores con velocidades de reloj inferiores, se necesitará un retraso superior.

PriorityBoost

Entero, establece el número de ciclos extra asignados a un thread cuando su prioridad se cambia a ALTA. El valor por defecto es 5.

ProcessPriorityLevel

Se llamaba *server_priority_class* en ibconfig

Nivel/tipo de prioridad para el proceso del servidor. Este parámetro reemplaza al parámetro server_priority_class de versiones anteriores a la 1.5 —vea abajo— con una implementación nueva. Los valores son enteros, con este formato:

- 0 – prioridad normal.
- valores positivos – alta prioridad (igual al modificador –B[oostPriority] de instsvc.exe en las opciones *configure* y *start*).
- valores negativos – baja prioridad.

Nota: Cualquier cambio de este valor debe ser cuidadosamente testeado para asegurar que el motor responde antes a las solicitudes.

RemotePipeName

Aplicable solo a las conexiones NetBEUI
Parámetro de tipo cadena, es el nombre de la pipe usada como canal de transporte en el protocolo NetBEUI. La named pipe es el equivalente a un número de puerto en TCP/IP. El valor por defecto —interbas— es compatible con las versiones anteriores de Firebird y con InterBase®.

Parámetros de configuración del espacio temporal para ordenaciones

Cuando el tamaño del buffer interno de ordenación es demasiado pequeño para contener las filas involucradas en una operación de ordenación, Firebird necesita crear archivos temporales de ordenación en el sistema de archivos del servidor. Por defecto usará la ruta especificada en la variable de entorno **FIREBIRD_TMP**. Si La variable no existe, intentará usar el directorio **/tmp** en Linux/UNIX, o **C:\temp** en Windows NT/2000/XP. Ninguna de estas ubicaciones se puede configurar por tamaño. Firebird proporciona un parámetro para configurar el espacio de disco que se usará para almacenar esos archivos temporales. Es prudente usarlo, para asegurar que habrá suficiente espacio disponible para ordenaciones en cualquier condición.

Todas las solicitudes CONNECT o CREATE DATABASE comparten la misma lista de directorios de archivos temporales y cada una crea su propio archivo temporal. Los archivos de ordenación son liberados cuando la ordenación finaliza o la solicitud es liberada.
En la versión 1.5, el nombre del parámetro ha cambiado de **tmp_directory** a **TempDirectories** así como la sintaxis del valor del parámetro también ha cambiado.

TempDirectories

Sustituye la entrada *tmp_directory* en isc_config/ibconfig

Proporciona una lista de uno o más directorios, separados por punto y coma (;), en los que se pueden almacenar ficheros de ordenación. Cada elemento puede incluir opcionalmente un argumento de tamaño, en bytes, para limitar su almacenamiento. Si se omite el argumento, o no es válido, Firebird usará el espacio en el directorio hasta que se agote, antes de pasar al siguiente directorio de la lista. Por ejemplo,
Unix: `/db/ordenaciones1 100000000;/firebird/ordenaciones2`
Windows: `E:\Ordenaciones 500000000`
Las rutas relativas se tratan como relativas a la ruta que el servidor reconoce como su directorio raíz. Por ejemplo, en Windows, si el directorio raíz es C:\Archivos de Programa\Firebird entonces el siguiente valor indicará al servidor que debe almacenar los archivos temporales de ordenación en C:\Archivos de Programa\Firebird\DatosUsuario\Ordenaciones, hasta un máximo de 500 Mb:

```
TempDirectories = DatosUsuario\Ordenaciones 500000000

No es necesario entrecomillar las rutas como en Firebird 1.0
```

Parámetros de Compatibilidad

CompleteBooleanEvaluation

Establece el método de evaluación de Booleanos (completa o perezosa. El valor por defecto (0=Falso) es "evaluación perezosa", la evaluación booleana de una expresión que involucre predicados AND o OR devolverá el resultado tan pronto se obtenga un valor de Cierto o Falso que ya no se pueda ver afectado por los resultados de la evaluación de un predicado posterior.
En algunas raras condiciones (normalmente evitables), puede ocurrir que una operación dentro de una condición AND u OR que no se haya evaluado debido al comportamiento perezoso, hubiese podido afectar el valor del resultado original. Si tiene la mala suerte de heredar una aplicación con estas características en su lógica SQL, puede desear modificar este parámetro para forzar la evaluación completa hasta que tenga la oportunidad de corregir ese funcionamiento. El tipo del parámetro es Booleano.

> No pase por alto el hecho de que este flag afecta a todas las evaluaciones booleanas efectuadas sobre cualquier base de datos del servidor.

OldParameterOrdering

La versión 1.5 ha corregido un viejo bug de Interbase que causaba que los parámetros de salida fueran devueltos al cliente en un orden idiosincrásico dentro de la estructura XSQLDA. Este bug es tan antiguo que varias aplicaciones, drivers y componentes de acceso existentes llevan un workaround (corrección) incorporado para solventar el problema en el lado cliente.
La versión 1.5 y posteriores reflejan las condiciones corregidas en el API y se instalan con OldParameterOrdering=0 (Falso). Establezca este parámetro booleano a Cierto si necesita volver a las condiciones antiguas para la compatibilidad con código existente.

Alias de archivos de Bases de Datos

La versión 1.5 de Firebird introduce alias de archivos de bases de datos para mejorar la portabilidad de aplicaciones y estrechar el control sobre los archivos de bases de datos internos y externos.

Aliases.conf

Configure los alias de archivos de bases de datos en el archivo de texto aliases.conf, ubicado en el directorio raíz de la instalación de su servidor Firebird. El archivo aliases.conf instalado será similar a:

```
#
# List of known database aliases
# ------------------------------
#
# Examples:
#
#   dummy = c:\data\dummy.fdb
#
```

Como en todos los archivos de configuración de Firebird, el símbolo '#' es una marca de comentarios. Para configurar un alias, simplemente borre el '#' y cambie la línea de ejemplo a la ruta apropiada del archivo de la base de datos.

```
# fbdb1 se encuentra en un servidor Windows:
fbdb1 = c:\Firebird\ejemplos\Employee.fdb
# fbdb2 se encuentra en un servidor Linux
fbdb2 = /opt/databases/killergames.fdb
#
```

Puede modificar aliases.conf mientras el servidor está en ejecución. No es necesario parar y reiniciar el servidor para que la nueva entrada en aliases.conf sea reconocida.

Conectar usando un alias de base de datos

Debe modificar la cadena de conexión de su aplicación cliente para que sea algo similar a:

```
Nombre_servidor:nombre_alias
```

Con el ejemplo anterior, la siguiente cadena de conexión realizará la solicitud al servidor Firebird que corre en la estación Linux llamada "miservidor" y buscará conectarse a la base de datos que tiene la ruta especificada en aliases.conf como "fbdb2":

```
miservidor:fbdb2
```

Nota: debido a que la utilidad **gstat** no utiliza una conexión para leer el archivo de base de datos, todavía se requiere la ruta completa al usar gstat (esto puede cambiar).

Nomenclatura de bases de datos en Windows

Observe que ahora se recomienda para los archivos de bases de datos en Windows ME y XP la extensión ".fdb" para evitar posibles conflictos con la característica "Restauración de Sistema" de Windows. Una omisión de este punto en esas plataformas le provocará la aparición del conocido problema del retraso en la primera conexión a una base de datos cuyo archivo primario y/o secundario tengan un nombre que use la extensión convencional ".gdb".

Equipo de Desarrollo de Firebird

Desarrollador	País	Principales tareas
Dmitry Yemanov	Rusia	Coordinador de versiones, mejoras en DSQL y PSQL, implementación del Servidor Integrado, numerosas mejoras del metadata, alias de bases de datos, triggers multi-acción, tipo de datos BigInt, nuevas variables de contexto, servidor Classic en Windows, numerosas correcciones de bugs
Nickolay Samofatov	Rusia	Diseñador/desarrollador de características SQL (Savepoints, bloqueos pesimistas), mejoras del metada, principales reimplementaciones del motor, localización y corrección de bugs, mejoras en arquitectura, activación del API de Servicios en Linux Classic, mejoras de rendimiento, binarios de Linux Classic
Arno Brinkman	Holanda	Mejoras del Optimizador, varias características nuevas en DSQL
Claudio Valderrama	Chile	Revisión de código, localización y corrección de bugs, mejoras PSQL, corrección de UDF, diseñador y desarrollador
Alex Peshkoff	Rusia	Nuevas características PSQL y DSQL, coordinador de características de seguridad y autor, corrección de código, binarios de Linux Superserver
Mike Nordell	Suecia	Traducción del código base de Firebird a C++, mejoras en rendimiento, características de portabilidad, localización y corrección de bugs
Blas Rodriguez Somoza	España	Desarrollador de nuevos conjuntos de caracteres, principal limpiador del código, binarios MinGW
Roman Rokytskyy	Alemania	Desarrollador y coordinador de JayBird
David Jencks	Estados Unidos	Diseñador y co-coordinador de JayBird, diseñador de las utilidades de documentación de Firebird
Carlos Guzmán Álvarez	España	Desarrollador y coordinador del provider .NET para Firebird
John Bellardo	Estados Unidos	Desarrollo del plug-in para gestión de conjuntos de caracteres, coordinador de binarios Darwin, desarrollador inicial del nuevo modelo de memoria
Erik Kunze	Alemania	Localización y corrección de bugs, limpieza de código, binarios SINIX-Z
Dmitry Sibiryakov	Rusia	Limpieza de código, binarios MinGW
Pavel Cisar	República Checa	Binarios Linux (versión 1.0), diseñador y coordinador de las utilidades QA
Ann Harrison	Estados Unidos	Corrección de bugs, consejera técnica, incremento del máximo de índices permitidos
Mark O'Donohue	Australia	Característica readline en isql, corrección de bugs, binarios de arranque (versión 1.0), corrección de código (versión 1.0)

Desarrollador	País	Principales tareas
Paul Reeves	Francia	QA; instaladores Win32; applet de panel de control estándar para Win32
Ignacio J. Ortega	España	Adición de la característica PLAN en triggers, limpieza de código
Konstantin Kuznetsov	Rusia	Binarios de Solaris Intel
Olivier Mascia	Bélgica	Re-desarrollo de los servicios de instalación Win32
Peter Jacobi	Alemania	Mejoras, actualizaciones de conjuntos de caracteres
Tilo Muetze	Alemania	Coordinador del proyecto de documentación de Firebird
Paul Vinkenoog	Holanda	Coordinador del proyecto de documentación de Firebird, correcciones de UDF
Artur Anjos	Portugal	Applets de panel de control Win32 mejorados, desarrollo del gestor de configuración de Firebird, internacionalización del mismo
Achim Kalwa	Alemania	Applets de panel de control Win32 mejorados
Sean Leyne	Canadá	Organizador del seguimiento de bugs, limpieza de código
Ryan Baldwin	Inglaterra	Desarrollador del driver Jaybird Type 2
Sandor Szollosi	Hungría	Desarrollador de collations para conjuntos de caracteres
Dmitry Kuzmenko	Rusia	Correcciones de bugs en GSTAT
Artem Petkevych	Ucrania	Correcciones de bugs en el tipo ARRAY
Vlad Horsun	Ucrania	Mejora drástica de velocidad del sweep, corrección de bugs en commit de 2 fases
Tomas Skoda	Eslovaquia	Correcciones de bugs
Evgeny Kilin	Rusia	Correcciones de bugs
Oleg Loa	Rusia	Correcciones de bugs
Erik S. La Bianca	Estados Unidos	Correcciones de bugs
Tony Caduto	Estados Unidos	Instaladores no oficiales para Win32
Juan Guerrero	España	Nuevas UDFs
Chris Knight	Australia	Binarios FreeBSD
Neil McCalden	Inglaterra	Binarios Solaris
Grzegorz Prokopsi	Hungría	Binarios Debian

Desarrollador	País	Principales tareas
Paul Beach	Inglaterra	Binarios HP-UX
Geoffrey Speicher	Estados Unidos	Binarios FreeBSD
Helen Borrie	Australia	Autora de las Notas de la versión, pruebas de campo y policía del pensamiento

"LOS HEROES DE LAS PRUEBAS DE CAMPO"

Pavel Kuznetsov Eugene Kilin Dmitry Kovalenko Vladimir Kozloff Barry Kukkuk Yakov Maryanov Christian Pradelli	Daniel Rail Volker Rehn David Ridgway David Rushby Pavel Shibanov Ruslan Strelba

N O T A S P A R A L A I N S T A L A C I Ó N

Instalación de Firebird para Windows 32

¡ LEA ESTO PRIMERO !

Con la introducción de dos nuevos modelos de servidor para plataformas Win32, las opciones de instalación de firebird han aumentado.

- Asegúrese de que ha iniciado sesión como administrador (esto no se aplica a Win9x o ME)
- Todos los modelos, superserver, clásico y embebido, además de las herramientas para clientes y servidores, pueden ser comodamente instalados usando el programa de instalación. Para la instalación completa de una nueva versión es muy recomendable usar el instalador correspondiente si existe alguno disponible.
- Utilice **gbak** para hacer una copia de seguridad de su base de datos de seguridad antigua **isc4.gdb** que mas adelante podrá restaurar en el nuevo formato como **security.fdb**
- Si tiene algún ajuste especial en **ibconfig** es posible que quiera transportarlo a los parametros equivalentes en **firebird.conf**. Revise la documentación disponible sobre firebird.conf para averiguar los parametros que podrá copiar directamente y los que necesitan una nueva sintaxis.

- En el caso de que en el directorio de instalación se encuentren ya ciertos archivos, éstos serán protegidos por el instalador pero SOBREESCRITOS si la instalación se realiza descomprimiendo un paquete comprimido (fichero zip) en la ubicación predeterminada. Estos archivos son:
  security.fdb
  firebird.log
  firebird.conf
  aliases.conf

- Cada modelo puede ser instalado desde un archivo comprimido zip. Este metodo puede ser extremadamente rapido si usted adquiere experiencia en este tipo de instalaciones. En cambio, este sistema puede resultarle complicado si es nuevo en Firebird.

- Se asume que

1. entiende como funciona su red
2. entiende por qué un sistema cliente/servidor necesita clientes y necesita servidores
3. ha leído el resto de las notas de introducción, o al menos ha ojeado lo suficiente para averiguar si algo resulta erróneo.
4. conoce la forma de acceder a la lista de soporte firebird para recibir asistencia en http://www.yahoogroups.com/groups/firebird-support

Si actualmente dispone de una versión anterior de Firebird o Interbase® en su servidor y piensa que puede necesitar regresar a ella de nuevo, antes de comenzar realice una copia de seguridad completa que le permita retornar a ella.

- Utilice la versión actual de GBAK para hacer copias de seguridad de sus bases de datos con formato transportable

- Haga una copia de seguridad del archivo gds32.dll de su carpeta de sistema. Quizá le interese renombrarla "gds32.dll.ib5" o "gds32.dll.fb103" o algo similarmente informativo. O simplemente copiarlo en otra carpeta distinta.

- También puede ser una buena idea hacer una copia de seguridad del runtime de Microsoft C++, el archivo msvcp60.dll. El instalador no debería sobreescribir este archivo, pero nunca se sabe lo que puede suceder en Windows.

- **DETENGA CUALQUIER SERVIDOR FIREBIRD O INTERBASE QUE ESTE EJECUTANDO**
  El instalador intentará detectar si alguna versión existente de Firebird o Interbase está instalada o ejecutándose. En una instalación manual el responsable de comprobar esto es usted.

- La ubicación por defecto de Firebird 1.5 será C:\Archivos de programa\Firebird\Firebird_1_5. Si su versión anterior está ubicada en esta misma carpeta y quiere utilizar la configuración por defecto de la versión 1.5, renombre la carpeta existente antes de comenzar la instalación.

- Para instalar Firebird como un servicio: Si quiere hacer uso de la nueva caracteristica **login seguro**, debe crear un "usuario de servicio firebird" con el nombre y contraseña que desee como usuario ordinario del sistema con los privilegios apropiados. Debería revisar la documentación contenida en el archivo README.instsvc.txt antes de continuar. Si dispone de un kit comprimido, puede encontrar este archivo en la carpeta /doc de la raiz del kit. Si no dispone de un kit comprimido, el fichero no estará disponible hasta después de la instalación, por eso puede acceder al mismo documento en la siguiente dirección:
  http://cvs.sourceforge.net/viewcvs.py/firebird/firebird2/doc/README.instsvc

¡ LEA ESTO TAMBIEN !

Uno de los objetivos de diseño de Firebird 1.5 es preparar el camino para múltiples instalaciones del servidor. Esto permitirá a los usuarios ejecutar diferentes versiones al mismo tiempo. Firebird 1.5 soporta esta característica, aunque no está bien documentada y requiere la intervención de un usuario avanzado. Las versiones futuras de Firebird harán este proceso mucho menos complicado. Mientras tanto, Firebird 1.5 necesita preparar el terreno. Esto nos fuerza a confrontar la cuestión de la instalación de librerías. Al mismo tiempo, Microsoft ha dado sus propios pasos para administrar la instalación de versiones diferentes de una librería. En conjunto, estas dos acciones por separado significan una nueva forma de instalación de librerías para Firebird 1.5 y siguientes.

Instalación de librerías de sistema de Microsoft

Los problemas asociados con la instalación de diferentes versiones de librerías de sistema de Microsoft son tan notorios que han adquirido el nombre de 'infierno de las DLL' (DLL Hell).

Desde el lanzamiento de Windows 2000 en adelante, Microsoft ha hecho prácticamente imposible la actualización de librerías de sistema. Para resolver esto Microsoft ahora recomienda que cada aplicación instale copias locales de cualquier librería de sistema que requiera.

Firebird 1.5 sigue esta práctica y ubica las librerías requeridas en el directorio \bin del servidor.

Instalación de fbclient.dll

A partir de la versión 1.5, Firebird no utiliza más gds32.dll como librería cliente. Ahora se llama fbclient.dll. Dados los problemas que Microsoft ha tenido con las DLL, no tendría mucho sentido continuar almacenando la librería cliente de Firebird en el directorio <system>. Además, si queremos permitir la instalación de múltiples servidores simultáneos, estaríamos provocando nuestro propio 'infierno' con las DLL si continuamos con la práctica de usar el directorio <system> para la librería cliente. Por lo tanto, a partir de Firebird 1.5, la librería cliente reside en el directorio \bin junto con los otros ejecutables.

Una nueva clave de registro ha sido agregada, y todas las aplicaciones compatibles con Firebird deben usarla para localizar la versión correcta de Firebird que se desea utilizar. La nueva clave es:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Firebird Project\Firebird Server\Instances
```

Firebird garantizará que siempre exista una entrada en esta clave. Se conocerá como

```
"DefaultInstance"
```

y almacenará la ruta al directorio raíz de la instalación por defecto (como habrá adivinado). Aquellos que no busquen una instalación particular pueden siempre usar la instancia por defecto para localizar la librería fbclient.dll.

Versiones futuras de Firebird incorporarán otras entradas bajo Instances. Las aplicaciones podrán enumerar las entradas del registro para determinar la librería de la instancia que deseen cargar.

Soporte a aplicaciones y controladores (drivers) realizados para versiones anteriores

Tradicionalmente, las aplicaciones que usan Interbase o Firebird han esperado encontrar la librería cliente gds32.dll en el directorio <system>. Firebird 1.5 incluye una utilidad llamada 'instclient.exe' que puede instalar un 'clon' de fbclient.dll al directorio de sistema de Windows (con nombre gds32.dll). Esta copia se modifica en la operación para que la información de versión del archivo comience con "6.3", para proveer compatibilidad con aplicaciones antiguas que verifican la versión de archivo de gds32.dll y no entenderían una cadena de versión como "1.5".

Durante el proceso de instalación el instalador verifica si existe un servidor Interbase o Firebird en el sistema. Si no encuentra ninguno, escribirá la gds32.dll en el directorio <system>. Si detecta que puede haber una versión de Interbase o Firebird instalada, no copiará la gds32.dll en el directorio <system>. No obstante, se puede usar la utilidad 'instclient.exe' para hacerlo a posteriori.

Se apunta a que versiones futuras de Firebird no intenten instalar gds32.dll en el directorio <system>, y por último a que sea completamente eliminada de la distribución.

La utilidad 'instclient.exe' también puede instalar la librería fbclient.dll misma al directorio de sistema de Windows, si se requiere. Esto serviría para las aplicaciones que necesiten cargarla desde esa ubicación.

La utilidad instclient.exe debería estar ubicada en el directorio 'bin' de su instalación Firebird y debe ser ejecutado desde allí.

Uso de instclient.exe:

```
instclient i[nstall] [ -f[orce] ] librería
           q[uery] librería
           r[emove] librería
```

donde *librería* es: `fbclient | gds32`

'-z' se puede usar con cualquier otra opción, para imprimir la versión.

Los contadores de información de versión y librería compartida son manejados automáticamente. Ud. puede agregar la opción –f[orce] para saltear las verificaciones de versión.

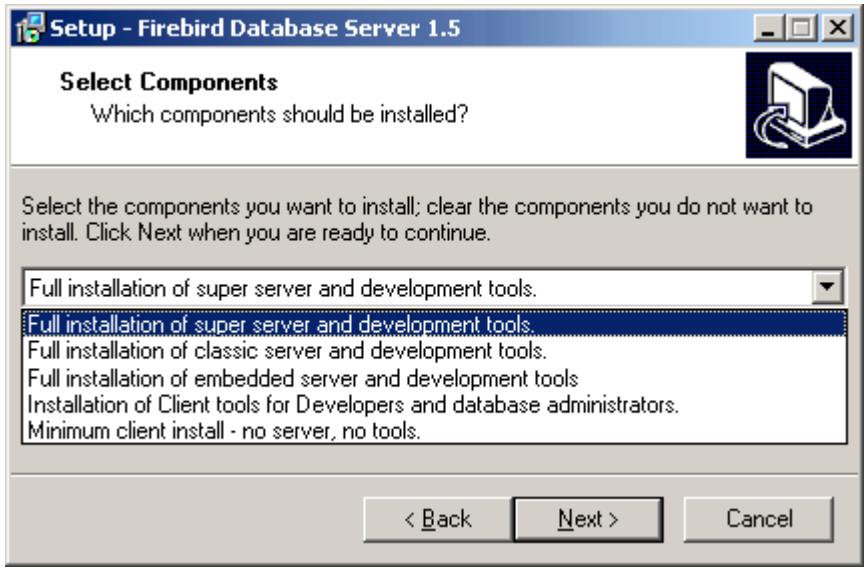
NOTA Si Ud. fuerza (-f) la instalación, puede interrumpir el funcionamiento de otra versión de Firebird o Interbase® ya instaladas. Debería reiniciar la máquina para finalizar la copia.

Para más detalles, vea el documento README.Win32LibraryInstallation.txt que está ubicado en la raíz de la instalación o en ..\doc.

Limpiando instalaciones de versiones candidatas (Release candidate)
Debería notarse que el instalador elimina fbclient.dll del directorio <system> si el fichero se encuentra ahí. El instalador también elimina la clave de registro HKLM\Software\FirebirdSQL.

Usando el instalador de Firebird para Win32.

Ejecute el programa y vaya respondiendo los dialogos. Después de cerca de cuatro dialogos, debería ver una lista desplegable similar a la siguiente. Este es el momento de seleccionar el tipo de instalación que desea hacer.



Seleccione el tipo de instalación y haga clic en "Next" para continuar la instalación.

Servicio o aplicación ?

Si elige instalar las versiones superserver o classic, y su sistema operativo soporta servicios, se le preguntará si desea ejecutar Firebird como un servicio o como una aplicación. A no ser que tenga algún motivo especial para hacer lo contrario, seleccione la ejecución como servicio.

Manual o automático ?
Con la opción automática, el motor Firebird se iniciará automáticamente cuando arrranque el servidor. Con la opción manual puede iniciar y detener el motor bajo demanda.

Guardian option
El Guardian es una utilidad que puede ejecutarse "por encima" del superserver y restaurarlo en caso de malfuncionamiento por cualquier motivo. No es demasiado interesante utilizarlo en entornos de desarrollo, pero en entornos de producción puede evitar algunas situaciones en las que el servidor deja de funcionar y nadie puede localizar al administrador que debe reiniciarlo.

Carpeta raiz de instalación
Si decide no utilizar la ubicacion por defecto, seleccione una ubicación creada con anterioridad o directamente escriba una ruta completa. Si la ruta introducida no existe, el instalador le pedirá confirmación y creará la nueva carpeta.

Eventualmente, los dialogos dejarán de aparecer y el servidor iniciará silenciosamente o se le solicitará que reinicie el ordenador. Será necesario reiniciar el ordenador en caso de que el instalador sobreescriba el archivo msvcp60.dll o una copia antigua de gds32.dll estuviera cargada en memoria al comenzar la instalación.

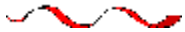
Instalación de la versión Superserver desde un kit comprimido

La instalación de FB 1.5 es en principio similar a la de las versiones anteriores.
Si no tiene un instalador especial (se distribuye por separado) los pasos son los siguientes:

- Descomprimir el archivo en un directorio aparte (puesto que los nombres de algunos archivos han cambiado, no tiene sentido descomprimir los archivos de la versión 1.5 en el directorio de IB/FB1)
- situarse en el directorio <root>\bin (a partir de ahora <root> será el directorio en el que están los ficheros de la versión 1.5 de Firebird)
- ejecutar instreg.exe:
-

  instreg.exe install <root>
  esto provoca que la ruta de instalación se grabe en el registro de windows
  (HKLM\Software\Firebird Project\Firebird Server\Instances\DefaultInstance)
- si se quiere registrar el servicio, ejecutar también instsvc.exe:
-

  instsvc.exe install
- de forma opcional, se pueden copiar los ficheros fbclient.dll y gds32.dll al directorio system del sistema operativo
-



Instalación de la versión Classic desde un kit comprimido

Para instalar el motor Classic, la única diferencia es la opción adicional que hay que pasarle al instsvc.exe:

instsvc.exe install <root> -classic

Note que esto significa que sólo se puede tener una arquitectura del motor—tanto fbserver.exe (Superserver) como fb_inet_server.exe (el proceso padre para Classic)—instalada como servicio.

El ***applet* del panel de control** deliberadamente no se instala con la versión Classic. No intente instalarlo ni usarlo. El concepto de terminar un servicio no se aplica al modelo del servidor Classic.

Instalación simplificada

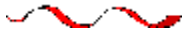
Si no necesita registrar el servicio, entonces puede evitarse ejecutar instreg.exe e instsvc.exe. En este caso puede simplemente descomprimir el fichero zip en un directorio aparte y ejecutar el servidor:

fbserver.exe -a

En este caso su directorio padre será considerado el directorio raiz de la instalación.

Desinstalación

Para eliminar FB 1.5 sin utilizar un desinstalador debe:
- parar el servidor
- ejecutar "instreg.exe remove"
- ejecutar "instsvc.exe remove"
- borrar el directorio de la instalación
- borrar los ficheros fbclient.dll y gds32.dll del directorio system del sistema operativo



Instalación del servidor Embebido desde un kit comprimido

Un servidor embebido es un cliente con un servidor totalmente funcional enlazado como una libreria dinámica (fbembed.dll). Tiene la misma capacidad que un superserver común y exporta los puntos de entrada del API estandar de Firebird.

Registro Las entradas del registro para firebird (es aquí donde normalmente el servidor busca la ubicación de la instalación) se ignoran. El directorio raiz del servidor embebido es el el directorio en el que se encuentra la dll.

Acceso a bases de datos Solo está permitido el "acceso local". El servidor embebido no ofrece soporte para protocolos remotos, por eso incluso el acceso a través de "localhost" no funciona.

Autentificación y seguridad La base de seguridad (security.fdb) no es utilizada por los servidores embebidos porque no se requiere. Cualquier usuario puede conectar con cualquier base de datos. Como el cliente y el servidor ejecutan en el mismo ambito local, la seguridad se reduce a una cuestión de acceso físico. De forma independiente, los privilegios SQL son chequeados, al igual que en el resto de modelos de servidor.

Compatibilidad Se puede ejecutar cualquier numero de aplicaciones usando el servidor embebido sin ningún tipo de conflicto. Disponer de un servidor IB/FB ejecutando no es ningún problema, pero en cualquier caso debe ser cauteloso al acceder a la misma base de datos desde multiples servidores embebidos simultaneamente, porque estos tienen arquitectura SuperServer y bloquean para uso exclusivo las bases de datos conectadas.

Estructura de ficheros para el Servidor Embebido
Simplemente copie el fichero fbembed.dll en el directorio de la aplicación. Renombrelo como fbclient.dll o gds32.dll, dependiendo de la forma de conexión de su software. Realice copias con los dos nombres si va a necesitar hacer uso de las herramientas del servidor (isql, gbak, etc.)

Debe también copiar los ficheros firebird.msg, firebird.conf (si lo necesita) e ib_util.dll al mismo directorio.

Si su aplicación utiliza UDFs o precisa soporte para INTL (fbintl.dll) o UDFs, debe colocarlas en otro directorio diferente. Para utilizarlas, colóquelas en un arbol de directorios similar al de un servidor Firebird, es decir, en unos subdirectorios llamados /intl y /udf colgando directamente del directorio en el que se encuentren los ficheros de firebird.

Ejemplo

```
D:\my_app\app.exe
D:\my_app\gds32.dll (fbembed.dll renombrada)
D:\my_app\fbclient.dll (fbembed.dll renombrada)
D:\my_app\firebird.conf
D:\my_app\aliases.conf
D:\my_app\isql.exe
D:\my_app\ib_util.dll
D:\my_app\gbak.exe
D:\my_app\firebird.msg
D:\my_app\intl\fbintl.dll
D:\my_app\udf\fbudf.dll
```

Entonces inicie su aplicación. Utilizará el servidor embebido como librería cliente y obtendrá acceso a bases de datos locales.

NOTA Asumiendo que ha configurado la estructura de directorios de acuerdo a estas reglas, no será necesario configurar el parámetro RoorDirectory en firebird.conf. No obstante, si Ud. decide instalar el servidor embebido y su aplicación en alguna estructura distinta de directorios, asegúrese de leer primero el documento README_embedded.txt de la distribución del servidor embebido, para obtener instrucciones sobre configuración adicional.

Desinstalación

Los desinstaladores de Firebird conservan y renombran los siguiente ficheros clave:

conserva security.gdb o lo renombra como security.fbnnnn
conserva firebird.log
conserva firebird.conf o lo renombra como firebird.confnnnn
conserva aliases.conf o lo renombra como aliases.confnnnn

"nnnn" es el número de build de la instalación vieja.
No se intenta desinstalar ningún archivo que no formara parte de la instalación original.
Los ficheros compartidos como fbclient.dll y gds32.dll se borran cuando el contador de uso indica que ninguna otra aplicación los utiliza.
Las claves del registro que hubieran sido creadas se eliminan.

Otras Notas

Winsock2

Firebird requiere WinSock2. Todas las plataformas Win32 deben tenerlo, excepto Win95. Se realiza una comprobación de la librería Winsock2 durante la instalación. Si no se encuentra, la instalación se detiene. Para encontrar la manera de actualizarse, visite este enlace:
http://support.microsoft.com/default.aspx?scid=kb;EN-US;q177719

Windows ME y XP

Windows ME y XP (Ediciones Personal y Professional) tienen una característica llamada System Restore, que provoca una modificación automática (¿copia de seguridad oculta?) de todos los ficheros con la

extensión gdb. Como consecuencia se retrasa el acceso a las bases de datos InterBase/Firebird ya que los archivos se copian cada vez que se produce una operación de lectura o escritura. (En XP no existe el System Restore en los servidores .NET)

En Windows ME, la extensión ".gdb" se encuentra en el fichero c:\windows\system\filelist.xml, que contiene los "tipos de fichero protegidos". Inicialmente, Charlie Caro recomendaba eliminar la extensión GDB de la sección "includes" de este fichero. Posteriormente, se ha demostrado que WinME puede reconstruir esta lista. En XP, no es posible siempre editar el fichero filelist.xml.

En ME, sugerimos una de estas formas de trabajo:
- use FDB (Firebird DB) como la extensión de los ficheros primarios de bases de datos
- mueva la base de datos a C:\Mis Documentos, para que sea ignorada por el System Restore
- desactive completamente el System Restore (consulte la documentación de Windows para ello).

En Windows XP (Personal y Professional) puede mover sus bases de datos a otra partición y excluir el System Restore de esa partición.

Windows XP utiliza smart copy, por tanto la bajada de rendimiento puede ser menor que para ME, sobre todo para ficheros pequeños. Para ficheros grandes (como las bases de datos firebird, ¡vaya!) no parece haber una respuesta mejor mientras los ficheros ".gdb" estén en el sistema de ficheros general.

Estamos intentando conseguir una descripción exacta del problema y una forma de trabajo probada para publicarla aquí. Si usted puede ayudar con la descripción y/o la forma de trabajo, envíe por favor un mensaje a la lista firebird-support o al grupo de noticias firebird-devel en news://news.atkin.com

El comportamiento de apagado de Windows XP es una conocida "área oscura". Hay una pausa perceptiblemente larga mientras se detiene el servicio del servidor. Durante este tiempo se indica que Firebird está ejecutándose como una aplicación. El problema parece afectar únicamente a Windows XP y sólo aparece si no se utiliza el Guardian para parar el servicio. Esta debe ser la forma de trabajo hasta que se encuentre la forma de corregirlo.

Instalación en UNIX / Linux

(Original de Mark O'Donohue, revisado para la 1.5)

El servidor Firebird viene en dos formas, versión Classic que se ejecuta como un servicio, y versión SuperServer que se ejecuta como un demonio en segundo plano. Classic es un servicio UNIX más tradicional, mientras que Superserver usa hilos (threads) en lugar de procesos. Para usuarios que se inician con Firebird, cualquiera de los dos servirá, aunque el servidor Classic parece proporcionar una plataforma mejor para sus primeros contactos con Firebird.

NOTAS – LEA ESTO PRIMERO

1) Necesitará validarse como usuario root para instalar Firebird.
2) La instalación en linux requiere un paquete glibc igual o mayor que glibc-2.2.5 y un libstdc++.so igual o mayor que libstdc++-5.0.
3) Para una evaluación estimada sobre la compatibilidad de su Linux, utilice esta tabla, pero no la tome como "la última palabra". Las distribuciones binarias de Linux pueden variar dependiendo de dónde y cuándo se han construido.
4) Asegúrese que los editores 'ed' y 'vim' están instalados en el sistema. Si el editor requerido no está presente, el paquete se instalará pero las secuencias de comandos de instalación (scripts) fallarán. (NOTA esta dependencia puede ser reemplazada en un futuro por 'sed').

INSTALACIÓN EN LINUX

Las siguientes instrucciones describen la instalación del servidor Classic. Para la instalación del Superserver, hay que reemplazar 'CS' en el nombre del paquete por 'SS'. Por ejemplo, el paquete FirebirdCS-1.5.0-nnnn.i686.rpm se convierte en FirebirdSS-1.5.0-nnnn.i686.rpm.

Instalación con un rpm de linux

```
$rpm -ivh FirebirdCS-1.5.0-nnnn.i686.rpm
```

Instalación a partir de linux .tar.gz

```
$tar -xzf FirebirdCS-1.5.0-nnnn.tar.gz
$cd FirebirdCS-1.5.0-nnnn.i686
$./install.sh
```

* o bien `FirebirdSS-1.5.0-nnnn`

Que hará la Instalación Linux

La instalación Linux va a

1. Intentar detener cualquier servidor en ejecución
2. Agregar el usuario 'firebird' y el grupo 'firebird' si no existen.
3. Instalará el software en el directorio /opt/firebird, creará enlaces para las librerias en /usr/lib e instalará los archivos de cabecera en /usr/include
4. Añadirá automaticamente gds_db en /etc/services asociado al puerto 3050, si la entrada no existe
5. Añadirá automaticamente localhost.localdomain y HOSTNAME en /etc/hots.equiv
6. El Superserver también instalará un script de arranque del servidor /etc/rc.d/init.d/firebird.
7. El servidor Classic instala un script de arranque /etc/xinetd.d/firebird o, para sistemas inetd antiguos, agrega una entrada en el archivo /etc/inetd.
8. Específico de SuSE: un nuevo enlace rcfirebird se crea en /usr/bin para el script init.d y una entrada Firebird se crea en /etc/rc.config
9. Arranca el servidor/servicio. Firebird debería arrancar automáticamente en los runlevel 2, 3 o 5.
10. Genera y fija una nueva contraseña aleatoria para el usuario SYSDBA, y la almacena en el archivo /opt/firebird/SYSDBA.password.
11. Agrega una entrada en aliases.conf para la base de datos de ejemplo, employee.fdb.

La instalación para Classic configura automáticamente la entrada xinetd si encuentra el directorio /etc/xinetd.d, en otro caso configurará una entrada inetd. Debido a que algunas distribuciones mantienen xinetd en una ubicación distinta al directorio /etc/xinetd.d, en esas condiciones se requerirá su configuración manual.

Comprobando su instalación Linux

Paso 1 – Accediendo a una base de datos

```
$cd /opt/firebird/bin
$isql -user sysdba -password <password*>

>connect localhost:employee.fdb; /* esto es un alias */

>select * from sales;
>select rdb$relation_name from rdb$relations;
>help;

>quit;
```

* una contraseña fue generada para Ud. durante la instalación. Puede ser obtenida del archivo /opt/firebird/SYSDBA.password.

Paso 2 – Creando una base de datos

A partir de la versión 1.5, el servidor firebird corre por defecto como el usuario 'firebird'. Aunque ésta ha sido siempre la configuración recomendada, el comportamiento por defecto anterior era ejecutar el servidor como usuario 'root'. Cuando se ejecuta como usuario root, el servidor tiene amplias facilidades para leer, crear y borrar archivos de bases de datos en cualquier lugar del sistema de archivos POSIX. Por razones de seguridad, el servicio debería tener más límites para leer/borrar y crear archivos.

Mientras que la nueva configuración es mejor desde una perspectiva de seguridad, requiere que se tomen en cuenta algunas consideraciones especiales cuando se crean nuevas bases de datos:

a) el usuario 'firebird' tiene que tener permisos de escritura en el directorio en el cual se desea crear la base de datos.
b) El valor recomendado para el parámetro DatabaseAccess en /opt/firebird/firebird.conf en None, para permitir el acceso solamente a través de entradas en el archivo aliases.conf.
c) Utilice entradas en aliases.conf para abstraer a los usuarios de la ubicación física real de las bases de datos. Puede encontrar más notas sobre alias aquí.

Los procedimientos para crear una nueva base de datos pueden variar si se seleccionan diferentes configuraciones pero los siguientes son los pasos que yo recomiendo con la configuración recomendada:

1) Si no existe un directorio cuyo *owner* sea el usuario 'firebird', cambie al usuario root y cree ese directorio:

```
$su – root
$mkdir –p /var/firebird
$chown firebird:firebird /var/firebird
```

2) Cree un nuevo archivo de base de datos y agregue una entrada de alias referenciándolo. Como usuario root o firebird, ejecute el siguiente script:

```
$cd /opt/firebird/bin
$./createDBAlias.sh text.gdb /var/firebird/test.fdb
```

(el script se utiliza en general de la siguiente manera: createDBAlias <nombre_BD> <ruta_al_archivo>

3) Como una alternativa (para el paso 2) los pasos en el script createDBAlias.sh se pueden realizar manualmente:

```
$vi /opt/firebird/aliases.conf
y agregue la siguiente línea al final del archivo:
test.fdb /var/firebird/test.fdb
```

4) Luego cree la base de datos:

```
$/opt/firebird/isql –u sysdba –p <password*>
SQL>create database 'localhost:test.fdb';
SQL>quit;
```

5) Si el valor de DatabaseAccess en /opt/firebird/firebird.conf está en Full o una ruta concreta (por ejemplo: DatabaseAccess=/var/firebird) otra alternativa al paso 2 es crear el archivo de base de datos directamente, usando la ruta absoluta con el nombre del archivo:

```
$/opt/firebird/isql –u sysdba –p <password*>
SQL>create database '/var/firebird/test.fdb';
SQL>quit;
```

Si utiliza esta configuración, el archivo de base de datos puede ser también accedido directamente sin una entrada en el archivo de alias:

```
$/opt/firebird/isql –u sysdba –p <password*>
SQL>create database '/var/firebird/test.fdb';
SQL>quit;
```

* una contraseña fue generada para Ud. durante la instalación. Puede ser obtenida del archivo /opt/firebird/SYSDBA.password.

Sugerencias y scripts de utilidad

Además de los archivos de instalación estándar, los siguientes scripts se proporcionan en el directorio bin de esta versión:

- **changeDBAPassword.sh**— Cambia la contraseña del usuario SYSDBA de Firebird. Para el Superserver, esto cambiará el script de inicio /etc/rc.d/init.d/firebird para que utilice la nueva contraseña.

- **createAliasDB.sh**- createAliasDB.sh <nombre_bd> <ruta_a_la_bd>.
  Este script crea un nuevo archivo de base de datos y agrega una entrada en el archivo aliases.conf.

- **fb_config**- un script que puede ser utilizado en archivos make para generar las rutas requeridas y los directorios de librerías (lib) para la versión instalada de Firebird. fb_config –help le brindará la lista completa de opciones.

- **changeGdsLibraryCompatibleLink.sh**- Sólo Classic- Cambia el enlace para libgds.so entre las librerías libfbclient.so (multihilo) y libfbembed.so (hilo único) que permite abrir directamente el archivo de la base de datos. Por compatibilidad con instalaciones previas, libgds.so referencia por defecto a libfbembed.so.

- **Acceso directo o embebido a los archivos de bases de datos**
  La instalación Classic ofrece un modo embebido de acceso que permite a los programas abrir archivos de bases de datos directamente. Para operar en este modo, un usuario habilitado en la base de datos requiere también acceso privilegiado a algunos de los archivos de configuración y estado de Firebird.

- **Conseguir acceso a bases de datos**: ahora que es el usuario 'firebird' y no root el usuario por defecto que ejecuta el software, necesita saber como **agregar un usuario al grupo firebird** para habilitar el acceso a las bases de datos. Está documentado en las notas *readme* (léame), pero los siguientes pasos deberían bastar para lograr lo principal:

  Para agregar un usuario (por ejemplo, skywalker) al grupo firebird, el usuario root debe hacer lo siguiente:

  ```
  $ usermod -G firebird skywalker
  ```

  La próxima vez que 'skywalker' se autentique en el sistema, podrá trabajar con bases de datos Firebird.

  Para listar los grupos a los que pertenece un usuario, escriba lo siguiente en la línea de comandos:

  ```
  $ groups
  ```

- **Problemas con NTPL en las últimas distribuciones de Linux:**

La nueva NPTL (Native POSIX Thread Library, Librería nativa de hilos POSIX) en Red Hat 9 (hasta ahora) causará problemas con el Superserver y programas compilados localmente, incluyendo las utilidades. Gbak, en particular, emitirá un error 'Broken Pipe'. Para corregirlo:

1. en /etc/init.d/firebird
```
LD_ASSUME_KERNEL=2.2.5
export LD_ASSUME_KERNEL
```

Esto se encarga de la instancia del servidor. También se debe tener la variable configurada para el entorno local, por lo tanto:

2. Agregue lo siguiente a /etc/profile, para asegurarse que cada usuario lo toma para las utilidades de línea de comandos.

A continuación de
```
  HISTSIZE=1000
```
agregue
```
  LD_ASSUME_KERNEL=2.25
```
En la línea siguiente, expórtelo:
```
 export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUT_RC LD_ASSUME_KERNEL
```

Desinstalación en Linux

Si necesita desinstalar, hágalo como usuario root. Los siguientes ejemplos usan CS o ClassicServer, pero lo mismo es válido para el Superserver reemplazando CS con SS.

Si se instaló desde paquetes rpm:

```
$rpm -e FirebirdCS-1.5.0
```

si se instaló con el archivo .tar.gz:

```
$/opt/firebird/bin/uninstall.sh
```

Instalación de Firebird Classic & SuperServer en Solaris 2.7 Sparc

No disponible actualmente. Por favor tome como referencia las notas de versión de la versión v.1 para las instalaciones de la versión 1.5.

Instalación de Firebird Classic en MacOS X / Darwin

No disponible actualmente. Por favor tome como referencia las notas de versión de la versión v.1 para las instalaciones de la versión 1.5.

Compilación o Instalación de Firebird en FreeBSD

No disponible actualmente. Por favor tome como referencia las notas de versión de la versión v.1 para las instalaciones de la versión 1.5.

Configurando el puerto del servicio en el cliente y el servidor

Por defecto un servidor Firebird escucha en el puerto 3050 las solicitudes de conexión de los clientes. El nombre registrado para el puerto del servicio es **gds_db**. Las buenas noticias son que, si estos valores por defecto son apropiados, no hay que hacer nada en el servidor o en los clientes para configurar el puerto del servicio.

Se puede usar un puerto diferente, un nombre de servicio diferente, o ambas cosas. Ud. debería hacer esto si el puerto 3050 es requerido por otro servicio, por ejemplo otro servicio gds_db corriendo concurrentemente configurado para una versión diferente de Firebird o para un servidor InterBase®. Tanto el servidor como el cliente deben ser configurados para sobrescribir el nombre del servicio, el nro. de puerto, o ambos, de al menos uno de estos modos:

1. en la cadena de conexión del cliente
- ❑ en el comando usado para arrancar el ejecutable del servidor
- ❑ activanddo los parametros RemoteServicePort o RemoteServiceName en firebird.conf (V.1.5 en adelante)
- ❑ en el demonio de configuración (para Classic sobre POSIX)
- ❑ por una entrada en el archivo Services

Antes de examinar cada una de estas técnicas, será conveniente ver la lógica utilizada por el servidor para inicializar el puerto de escucha y la que utiliza el cliente para inicializar el puerto que debería sondear para solicitudes.

Como el servidor establece el Puerto de escucha

El ejecutable del servidor tiene un parámetro de línea de comandos opcional (-p) por el cual se puede establecer el **número de puerto** en el que estará escuchando o el **nombre del servicio** que lo hará. En este punto, si el parámetro está presente, se reemplazará el número de puerto 3050 o el nombre de servicio (gds_db) con el argumento suministrado con la opción –p.

A continuación —o al principio, si no hay parámetro –p, un servidor v.1.5 chequea firebird.conf para ver los parámetros RemoteServiceName y RemoteServicePort:

- ❑ Si ambos están comentados con "#" entonces se asumen los valores por defecto y no ocurre ningún otro cambio. Cualquier opción –p es válida y el argumento que "falta" se mantiene en su valor por defecto.

- ❑ Si el parámetro RemoteServiceName no está comentado, pero sí RemoteServicePort, entonces el nombre del servicio se sustituye *únicamente* si éste no ha sido sobreescrito ya con la opción –p.

- ❑ Si el parámetro RemoteServicePort no está comentado, pero sí RemoteServiceName, entonces el número de puerto se sustituye *únicamente* si éste no ha sido sobreescrito ya con la opción –p.

- ❑ Si eliminamos los comentarios de ambos, RemoteServicePort y RemoteServiceName, entonces RemoteSerivceName tiene prioridad si no ha sido sobrescrito con la opción –p. Si ya ha sido reemplazado el nombre del servicio, el valor de RemoteServiceName es ignorado y el valor de RemoteServicePort reemplaza al 3050.

- ❑ En este punto, si se detecta un reemplazo ya sea del número de puerto o del nombre del servicio, ambos servidores (v.1.0 y v.1.5) proceden a chequear el archivo Services buscando una entrada con la combinación correcta de nombre de servicio y número de puerto. Si se encuentra una correspondencia, todo está bien; en caso contrario, si el nombre del servicio no es gds_db, el

servidor generará una excepción y no arrancará. Si el nombre del servicio es gds_db y no está asignado a ningún otro puerto, será mapeado automáticamente al puerto 3050.

Si el nombre por defecto del servicio es reemplazado, entonces deberá crear una entrada en el archivo services —vea el siguiente apartado, *Configurando el archivo services.*

Usando el parámetro –p para reemplazar los valores

Por favor note que este parámetro también se encontraba disponible en Firebird 1.0.x, pero no estaba documentado.

Arrancar el servidor con el parámetro opcional –p posibilita sobreescribir el puerto por defecto (3050) o el nombre del servicio por defecto (gds_db) que el servidor usa para escuchar por solicitudes de conexión. El parámetro puede sobreescribir uno, pero no ambos. De Firebird 1.5 en adelante, se puedes usar el parámetro –p en combinación con una configuración en firebird.conf para lograr el reemplazo tanto del puerto como el nombre del servicio.

Sintaxis para TCP/IP

```
Comando-servidor <otros parámetros> -p número-puerto | nombre-servicio
```

Por ejemplo, para arrancar el Superserver como una aplicación y sobreescribir el nombre de servicio gds_db con fb_db:

```
fbserver -a -p fb_db
```

O, para sobreescribir el Puerto 3050 al 3051:

```
fbserver -a -p 3051
```

Sintaxis para redirección en WNet

En una red WNet, reemplace la sintaxis del parámetro –p anterior con la siguiente sintaxis "contrabarra-contrabarra-punto-arroba":

```
fbserver -a -p \\.@fb_db
```

o

```
fbserver -a -p \\.@3051
```

Classic sobre POSIX: el demonio *inetd* o *xinetd*

Con el servidor Firebird Classic sobre Linux o UNIX, el demonio **inetd** o **xinetd** se configura para escuchar en el puerto por defecto y transmitir el nombre del servicio por defecto. El script de instalación escribirá la entrada apropiada en el archivo de configuración /etc/inetd.conf o /etc/xinetd.conf.

Se puede editar la configuración existente si fuera necesario. A continuación hay un ejemplo de lo que debería encontrar en xinetd.conf o inetd.conf después de instalar Firebird Classic sobre Linux:

```
# default: on
# description: FirebirdSQL server
#
service gds_db
{
        flags                   = REUSE KEEPALIVE
        socket_type             = stream
        wait                    = no
        user                    = root
#       user                    = @FBRunUser@
        log_on_success          += USERID
        log_on_failure          += USERID
        server                  = /opt/firebird/bin/fb_inet_server
        disable                 = no
}
```

Si ha configurado el puerto del servicio a un valor diferente del estándar, necesitará modificar xinetd.conf o inetd.conf en el mismo sentido. Reinicie xinetd (o inetd) con kill –HUP para estar seguro de que el demonio usará la nueva configuración.

NOTA Las solicitudes de conexión fallarán si ambos xinetd (o inetd) y fbserver (o ibserver) intentan escuchar en el mismo puerto. Si la máquina tiene esta doble configuración, será necesario establecer valores adecuados para que cada versión del servidor tenga su propio puerto de servicio.

Usando un parámetro del archivo de configuración

Ud. puede configurar RemoteServiceName o RemoteServicePort en firebird.conf para reemplazar los valores por defecto correspondientes.

El servidor usará uno de los parámetros RemoteService* , pero no ambos. Si configura los dos, se ignorará RemoteServicePort en todos los casos, excepto cuando el comando de arranque del servidor sea invocado con el parámetro –p proveyendo un reemplazo del nombre de servicio. De esta manera, se puede usar la opción –p y un parámetro RemoteService*, en combinación, para especificar tanto el puerto como el nombre del servicio.

Si el nombre del servicio por defecto va a ser modificado, entonces debe crear una entrada en el archivo services.

CUIDADO!! Si elimina el comentario a RemoteServiceName o RemoteServicePort, pero deja los valores intactos, serán tratados como reemplazos. Entonces será necesario crear una entrada explícita en el archivo services para establecer el puerto de servicio por defecto.

Configurar un cliente para encontrar el Puerto del servicio

Si configura el servidor con los valores de la instalación por defecto (servicio gds_db escuchando en el puerto 3050) entonces no se requiere ninguna configuración. Si el servidor está escuchando en un puerto diferente o si está usando un nombre de servicio diferente, la aplicación y/o equipo cliente necesita habilitar alguna configuración para ayudar a la librería cliente de Firebird a encontrar el puerto de escucha.

La cadena de conexión usada por un cliente puede incluir información para el sondeo del puerto de escucha del servidor de varios modos. Los clientes Firebird 1.5 pueden opcionalmente usar una copia local de firebird.conf.
También podrían ser necesarios cambios en el archivo services del cliente.

Usando la cadena de conexión

Si sólo el número de Puerto o el nombre de servicio ha sido reconfigurado, entonces debe incluir el puerto o nombre de servicio alternativo en la cadena de conexión. Esto funciona en todas las versiones de Firebird.

Sintaxis para conexiones TCP/IP

Para conectar a un servidor de base de datos llamado alice que está trasmitiendo en el Puerto 3050 con el nombre de servicio fb_db, la cadena de conexión sería:

Para POSIX:
```
alice/fb_db:/data/teaparty.fdb
```

O, si el nombre del servicio es gds_db y el Puerto es el 3051:
```
alice/3051:/data/teaparty.fdb
```

Para Windows:
```
alice/fb_db:D:\data\teaparty.fdb

alice/3051:D:\data\teaparty.fdb
```

Observe que el separador entre el nombre del servidor y el puerto es una barra, no dos puntos. Los dos puntos antes de la ruta física también son necesarios.

Sintaxis para conexiones WNet

En una red WNet, use notación estilo-UNC:

```
\\alice@fb_db\d:\teaparty.fdb
```

o

```
\\alice@3051\d:\teaparty.fdb
```

Si el número de puerto o el nombre de servicio está sobreescrito en el servidor, entonces debe crear una entrada en el archivo services.

Usando sintaxis de puerto con alias a bases de datos

Para conectar a través de un puerto no estándar con un alias de base de datos, debe añadir el puerto o el nombre del servicio al nombre del servidor, no al alias. Por ejemplo, suponiendo que el alias de base de datos está almacenado en aliases.conf como

```
rabbit = /data/teaparty.fdb
```

Su cadena de conexión en la aplicación para el servidor 'alice' sería:

```
alice/fb_db:rabbit
```

o

```
alice/3051:rabbit
```

Usando una copia de firebird.conf

A partir de Firebird 1.5, opcionalmente se puede incluir una copia de firebird.conf del lado del cliente en el directorio raíz de Firebird y configurar RemoteServiceName o RemoteServicePort para ayudar al cliente a localizar el puerto del servidor.

- Puede configurar *uno* de estos dos parámetros para extender el reemplazo proporcionado por el otro parámetro a través de la cadena de conexión (arriba); o sobreescribir sólo RemoteServiceName o RemoteServicePort sin usar la cadena de conexión.

- Si necesita evitar pasar el nombre de servicio o número de puerto en la cadena de conexión y el servidor no está usando los valores por defecto para ambos, puede configurar los dos - -RemoteServiceName y RemoteServicePort. Se usaría esta técnica si su aplicación cliente necesitara mantener la capacidad para conectar a servidores Interbase o Firebird 1.0.

Ubicación de elementos de Firebird en los clientes

Cuando se depende del archivo firebird.conf en las máquinas cliente, es importante que la librería cliente sepa donde encontrarlo. Necesitará configurar un directorio raíz e informar al sistema de su localización. Puede usar la variable de entorno FIREBIRD para esto. Los clientes Windows pueden alternativamente usar la clave de registro Firebird. Con una configuración correcta del cliente, puede hacer uso también de una versión local del archivo de mensajes.

Configurando el archivo services

No se necesita configurar una entrada para el puerto del servidor o clientes Firebird si el servidor usa los valores por defecto de la instalación, gds_db en el puerto 3050. Si gds_db es el nombre del servicio y este no puede ser asignado a otro puerto, éste se mapeará automáticamente al puerto 3050.

Si está configurando el puerto del servicio para un puerto o nombre de servicio diferente, tanto el **servidor** como el **cliente** deben ser explícitamente actualizados para reflejar la reconfiguración. En ambos Linux y Windows, esta información está almacenada en el archivo **services**.

Localizando el archivo services

- En Windows NT/2000/XP/S2003, este archivo es C:\windows\system32\drivers\etc\services.

- En Windows 95/98/ME, es C:\windows\services.

- En Linux/UNIX es /etc/services.

Una entrada de servicio es como esto:

```
gds_db 3050/tcp # Firebird Server 1.5
```

Abra el archivo usando un editor de texto y añada una línea o edite la entrada existente gds_db como sigue:

- para un servidor o cliente Firebird 1.0.x, modifique el nombre del servicio o el número de puerto para reflejar como arrancará el servidor.

- Para un servidor Firebird 1.5 o superior, edite o agregue una línea según corresponda. Si tiene un servidor Firebird 1.0 o InterBase instalado en la misma máquina, retenga las entradas que ellos necesitan y agregue una nueva reflejando como trabajará el servidor Firebird 1.5.

Información adicional

Se puede encontrar más información sobre el motor de bases de datos Firebird en
http://firebird.sourceforge.net

o en los sitios relacionados:
http://firebirdsql.org
http://www.ibphoenix.com
http://www.cvalde.net

Si Ud. está interesado en involucrarse en el desarrollo de Firebird, o desea informar de un posible error o comenzar una discusión, será bienvenido a sumarse a nuestra lista firebird-devel. Para suscribirse, envíe un email vacío a:

firebird-devel-request@lists.sourceforge.net

con la palabra 'subscribe' en el campo Asunto.

Por favor no utilice la lista firebird-devel para preguntas de soporte.

Si necesita soporte técnico, suscríbase a la lista firebird-support. Vaya a la dirección
http://www.yahoogroups.com/groups/firebird-support

Hay una lista especializada para soporte de desarrollo con Interclient y Java:
http://www.yahoogroups.com/groups/Firebird-Java

La lista firebird-support se ocupa de problemas técnicos con Firebird e Interbase ®. Por favor envíe sus preguntas sobre Delphi y otros entornos de desarrollo al foro apropiado.

La comunidad de Código Abierto opera algunas otras listas de discusión sobre varios aspectos del desarrollo de Firebird. Para más detalles, refiérase a las listas de correo y grupos de noticias del sitio de la comunidad de Firebird.

La lista de desarrolladores de Firebird y las listas generales de la comunidad, junto con otras listas de interés para desarrolladores Firebird e Interbase, se pueden acceder como grupos de noticias en
news://news.atkin.com

Se puede contratar **soporte** en cualquier lugar del mundo a través de IBPhoenix (los números y direcciones de contacto están en http://www.ibphoenix.com). Varios miembros del equipo Firebird también ofrecen soporte y consultoría. Por favor contacte con ellos directamente.

IBPhoenix ofrece **Servicios de Recuperación de Bases de Datos**. Para analizar y posiblemente reparar bases de datos corruptas usted mismo, intente con el producto IBSurgeon de IBase.ru (www.ibase.ru). IBase.ru también ofrece servicios de recuperación de datos en Rusia y Europa.

La fundación FirebirdSQL Foundation Inc. puede recibir **pedidos/ofertas para mejoras patrocinadas**. Envíe un email a foundation@firebirdsql.org. Tal vez prefiera contactar con los administradores del proyecto Firebird directamente al principio – envíe un email a firebirds@users.sourceforge.net.

Las **Discusiones sobre mejoras en general a FB** pueden ser dirigidas a la lista Firebird-priorities (http://www.yahoogroups.com/community/Firebird-priorities).

La lista IB-Architect (http://www.yahoogroups.com/community/ib-architect) es para **discusiones técnicas de diseño** exclusivamente. Las preguntas de soporte/conversión están definitivamente fuera de lugar aquí.

Herramientas y controladores

Programas cliente de administración

Se puede encontrar una selección de excelentes programas GUI de administración para Firebird en la página Contributed Downloads en http://www.ibphoenix.com. Algunas son Open Source, otras gratuitas, y otras son productos comerciales ya establecidos.

El programa IBConsole de Borland no se recomienda como un cliente de administración de bases de datos para Firebird 1.5.

Controladores y Componentes

JAVA: el proyecto Jaybird de controlador JDBC se desarrolla activamente como parte del proyecto Firebird. Se ha lanzado ya un controlador Tipo 4 JDBC/JCA y un controlador tipo 2 en beta. Las fuentes y ejecutables pueden descargarse desde las páginas de lanzamientos de Firebird, en
http://sourceforge.net/project/showfiles.php?group_id=9028

Por ayuda y/o participación en el desarrollo y prueba, suscríbase al foro Firebird-java en

http://www.yahoogroups.com/community/firebird-java.

.NET: hay un proyecto de controlador .NET para Firebird en marcha. Los fuentes y ejecutables pueden descargarse desde las páginas de lanzamientos de Firebird, en
http://sourceforge.net/project/showfiles.php?group_id=9028

Por ayuda y/o participación en el desarrollo y prueba, suscríbase al foro del proveedor .NET de Firebird:
http://lists.sourceforge.net/lists/listinfo/firebird-net-provider

Delphi y C++Builder: los usuarios tienen dos poderosas alternativas para una conectividad completa y directa a través de la API de Firebird 1.5, ambas con un buen soporte de los desarrolladores y otros usuarios:

- IB Objects de Jason Wharton: http://www.ibobjects.com
- FIBPLus: http://www.devrace.com

C++: la librería gratuita de acceso para C++ 'IBPP' (http://www.ibpp.org), bajo licencia MPL, alojada en sourceforge.net, completamente portable entre Win32 y Linux y posiblemente otras plataformas POSIX. Es útil cuando se necesita un acceso de bajo nivel, del tipo de una API de C, pero con una capa de liviana de abstracción C++ no limitada a ningún entorno particular de desarrollo.

ODBC: se puede ver un listado de controladores ODBC en la página Contributed Downloads en http://www.ibphoenix.com. El 'laboratorio' para el desarrollo de un controlador Open Source ODBC/JDBC está en el foro http://lists.sourceforge.net/lists/listinfo/firebird-odbc-devel.

PHP: un grupo está trabajando para actualizar la antigua extension PHP para Interbase a Firebird. Por consultas sobre este proyecto, suscríbase al foro Firebird-PHP en http://www.yahoogroups.com/community/firebird-php

PYTHON: KInterbasDB es un paquete de extensión para Python que implementa soporte compatible con Python Database API 2.0 para Firebird. Soporte nativo total de la API del cliente Firebird; en una versión estable y bajo desarrollo activo. Licencia de código abierto BSD. Descargas y novedades en http://kinterbasdb.sourceforge.net/

Documentación

La documentación para Interbase v 6.0 se aplica también a la versión actual de Firebird. Una versión beta de los manuales de Interbase 6 se puede descargar en formato Adobe Acrobat desde

ftp://ftpc.inprise.com/pub/interbase/techpubs/ib_60_doc.zip

Un índice estructurado de documentación se mantiene en el sitio de la comunidad Firebird, en

http://firebird.sourceforge.net/index.php?op=doc

Este es un trabajo en progreso, y todas las adiciones son bienvenidas – envíe un mensaje a firebird-docs@lists.sourceforge.net

En el área de documentación se pueden encontrar algunos lineamientos de instalación y otros escritos (HowTos), que se puede acceder desde
http://www.firebirdsql.org
o más directamente desde
http://sourceforge.net/projects/firebird

El repositorio principal para cuestiones de uso y técnicas es el sitio de IBPhoenix
http://www.ibphoenix.com

IBPhoenix también publica regularmente un CD por suscripción(también se pueden obtener ediciones solas), que contiene manuales creados por ellos – Using Firebird (Usando Firebird) y The Firebird Reference Guide (Guía de referencia de Firebird).

También se puede encontrar alguna documentación adicional visitando el área de publicaciones técnicas de Borland:
http://www.borland.com/techpubs/interbase/

Correcciones y agregados desde la versión 1.0

N° seguimiento	Descripción	Contribuyente
(sin N°)	Solucionadas algunas inconsistencias menores en el nombrado de juegos de caracteres	P. Jacobi
(sin N°)	GSTAT se cerraba con algunas combinaciones de parámetros	D. Yemanov
(sin N°)	Corregido el manejo de puntos de retorno (savepoints) en BREAK|LEAVE/EXIT.	D. Yemanov
(sin N°)	Modificación al optimizador para que prefiera índices simples antes que compuestos, y también índices únicos de concordancia completa.	A. Brinkman
(sin N°)	Mejoras en las herramientas de instalación para Win32 instsvc.exe e instreg.exe	O. Mascia
Mejora	Se incrementó el número máximo de índices por tabla de 64 a (DB_PAGE_SIZE/16)-2	A. Harrison, portado a v1.5 por N.Samofatov
721792	Una conexión de mucho tiempo causaba pérdida de memoria en el dispositivo del núcleo del OS	N. Samofatov
775003	La UDF log(x, y) en realidad devolvía log(y, x)	P. Vinkenoog, N. Samofatov
774987	Las UDFs ltrim(") y rtrim(") devolvían NULL; rtrim salteaba el 1er carácter	P. Vinkenoog, N. Samofatov
(sin N°)	Corregido: caída del servidor causada por un contexto de transacción perdido.	A. Peshkoff
(sin N°)	Corregido: caída del servidor debido a cualquier combinación de sub-select y Between	A. Peshkoff
736318	"<value> STARTING WITH <field>" falla cuando se usan índices	D. Yemanov
(sin N°)	Se produce un abrazo mortal (deadlock) que no corresponde después de la ejecución de triggers before (update/delete)	A. Peshkoff
Mejora	Hacer no reservadas las palabras INSERTING/UPDATING/DELETING	N. Samofatov
Mejora	Se agregaron nuevo mensajes de error (más específicos) para algunos de los cambios en v1.5	D. Yemanov, A. Brinkman, A. Peshkoff
Corrección de seguridad	Se agregó un parámetro –login a instsvc que permite instalar el servicio FB con una cuenta diferente de localsystem	A. Peshkoff

N° seguimiento	Descripción	Contribuyente
Mejora	Se Re-introdujo el recorte de campos VARCHAR en los protocolos remotos	D. Yemanov
(sin N°)	Caída aleatoria del servidor cuando se preparan consultas grandes	D. Yemanov
Mejora	Mejora de Configuración – hacer que el manejo de las rutas en firebird.conf siga los requerimientos del OS	A. Peshkoff
(sin N°)	Argumentos erróneos de tipos DATE/TIME (dialecto 3) en UDFs	Oleg Loa
(sin N°)	Posible violación de integridad referencial	Vlad Horsun, D. Yemanov
745090 y otras cuestiones de instalación en RC2	Problemas de permisos en firebird.conf (SF #745090) También genera aliases.conf al instalar; use rpmbuild para crear paquetes Linux	Erik S. LaBianca, N. Samofatov
(sin N°)	Permite un ajuste fácil de LockSemCount en las plataformas POSIX. No se necesita usar gds_drop o reiniciar la máquina para hacer que el nuevo valor tome efecto.	N. Samofatov
Mejora	Hacer que las palabras FIRST/SKIP no sean reservadas	N. Samofatov
(sin N°)	Referencia de conexión errónea después de una excepción en PSQL	A. Peshkoff
(sin N°)	Los comandos BREAK/LEAVE y EXIT están disponibles ahora para su uso en triggers	D. Yemanov
(sin N°)	Posible corrupción de índices durante recolección de basura	Vlad Horsun, D. Yemanov
(sin N°)	Resueltos los problemas con la administración de archivos temporales Fallo de seguridad en todas las plataformas POSIX excepto FREEBSD/OPENBSD, relativa al uso de mktemp (posibles ataques DoS o elevación de privilegios) Solamente se generan 27 nombres de archivo únicos en win32 (lo que puede causar un comportamiento impredecible en SS)	N. Samofatov
(sin N°)	Cambio en el manejador de eventos: desabilitar uso de un puerto aux definido en CS, debido a cuestiones conocidas.	D. Yemanov

N° seguimiento	Descripción	Contribuyente
(sin N°)	Se permite el uso de funciones de agregación de diferentes contextos dentro de otra función de agregación Ejemplo: SELECT MAX((SELECT COUNT(*) FROM RDB$RELATIONS)) FROM RDB$RELATIONS	A. Brinkman
(sin N°)	Posibles caídas del servidor al desconectar cuando se utiliza la notificación de eventos	D. Yemanov
(sin N°)	Cambios en el manejador de servicios: algunas características de GSTAT/GSEC no están disponibles via la API de servicios de Win32 CS (hasta el lanzamiento de la v1.6)	D. Yemanov
(sin N°)	Se reportan estadísticas de registro erróneas cuando una operación falla por alguna razón.	D. Yemanov
(sin N°)	stdin/stdout no se pueden usar para redireccionar la E/S de consola en la versión Win32 de GBAK	A. Peshkoff
(sin N°)	El redimensionamiento de la tabla de bloqueos no funcionaba bien. No más errores “lock manager out of room” (Win32 CS 1.5 RC1) o caídas (posible en todas las otras versiones CS de Interbase/Firebird)	N. Samofatov
Mejora	Mejora a INTL: hacer que la función UPPER trabaje con el juego de caracteres WIN1251 sin usar secuencias de ordenación (collations) explícitas	N. Samofatov, D. Yemanov
BUGCHECK(291)	Posible corrupción de base de datos cuando se modifica/borra en un trigger “antes” el mismo registro por el cual fue llamado el trigger	A. Peshkoff
(sin N°)	Sobrepasamiento de Buffer en llamada a isc_database_info()	Oleg Loa
(sin N°)	Cambio en el administrador de configuración: ahora el servidor aborta si falta/es erróneo el archivo firebird.conf, con un reporte de error en el log del sistema.	A. Peshkoff
(sin N°)	Arreglado en la API de servicios: habilitar API de servicios de estadísticas en las versiones POSIX	N. Samofatov
(sin N°)	Cambios en el analizador gramatical. 1) ROWS_AFFECTED se renombra como ROW_COUNT 2) CONNECTION_ID/TRANSACTION_ID se denominan ahora CURRENT_CONNECTION / CURRENT_TRANSACTION 3) Algunos de los nuevos símbolos se hacen no-reservados	D. Yemanov
(sin N°)	Arreglado en la API de servicios: se habilita parcialmente la API de servicios para las versiones CS de Win32	D. Yemanov

N° seguimiento	Descripción	Contribuyente
(sin N°)	Entrega errónea de eventos (utilización innecesaria de paquetes OOB)	Jim Starkey, Paul Reeves
(sin N°)	Manejador de bloqueos mejorado: los abrazos mortales (deadlocks) ahora se detectan y se reportan tan pronto como todos los procesos bloqueados reciben notificación, esto es, instantáneamente en todos los casos normales.	N. Samofatov
(sin N°)	Caída del servidor en algunas operaciones con la API de servicios	A. Brinkman
(sin N°)	Características avanzadas de seguridad: se implementa acceso configurable para bases de datos, tablas externas y librerías UDF	A. Peshkoff
(sin N°)	Corregidas pérdidas de recursos/memoria	Mike Nordell, A. Peshkoff, N. Samofatov, D. Yemanov
(sin N°)	Sobrepasamiento de buffer con arreglos multidimensionales	D. Yemanov
213460, 678718	Varias cuestiones con eventos usados en servidores con varias direcciones IP NOTA: ahora también es posible configurar un puerto definido para procesamiento de eventos.	D. Yemanov
(sin N°)	Arregladas algunas pérdidas de recursos	Mike Nordell, A. Peshkoff
(sin N°)	Arreglo en la API de servicios: se habilita la API de servicios para las versiones CS posix Notas: 1. Los cambios apropiados en la version CS Win32 no están listos todavía. 2. El servicio de respaldo/recuperación fue arreglado, probado y debería funcionar. 3. La validación de base de datos fue parcialmente corregida y debería funcionar. 4. Posiblemente otros servicios no sean funcionales todavía en las versiones CS.	N. Samofatov
(sin N°)	Mejora SQL: se permiten NULLs en restricciones de unicidad e índices (espec. SQL-99).	D. Yemanov, N. Samofatov
(sin N°)	Mejora de rendimiento: el log de deshacer VIO ahora usa un árbol B+ para almacenar los datos de registros de puntos de referencia (savepoints). Esto mejora un poco el rendimiento (usualmente 2-3 órdenes de magnitud para 100000 registros) cuando se hacen múltiples actualizaciones de registros en una sola transacción.	N. Samofatov

N° seguimiento	Descripción	Contribuyente
(sin N°)	Corrupción de la base de datos cuando de respalda un punto de referencia después de un gran número de operaciones DML (de manera que el punto de referencia se descarta) y un registro fue actualizado fuera del punto de referencia y borrado dentro del punto de referencia.	N. Samofatov
(sin N°)	Mejora a EXECUTE STATEMENT. Ahora es posible devolver valores desde el SQL dinámico. Syntaxis: EXECUTE STATEMENT <valor> INTO <lista_var>; (un solo registro) o FOR EXECUTE STATEMENT <valor> INTO <lista_var> DO <lista_stmt>;	A. Peshkoff
(sin N°)	El servidor se cae durante la desconexión después de actualizaciones masivas.	D. Yemanov
(sin N°)	Mejora al optimizador: las subconsultas en la cláusula SET de un UPDATE pueden usar índices ahora.	A. Brinkman
(sin N°)	Error "Context already in use" en el caso de DISTINCT con subconsultas.	A. Brinkman
(sin N°)	Mejora a isc_database_info: la lista de transacciones activas actualmente está disponible ahora vía una llamada a isc_database_info.	N. Samofatov
(sin N°)	Mejora de rendimiento: circuito corto para evaluación booleana. NOTA este comportamiento es controlado por la opción "CompleteBooleanEvaluation" de firebird.conf. El valor por defecto es 0 (evaluación en circuito corto).	Mike Nordell
(error en Beta 2)	Sobrepasamiento de pila durante preparación de comandos	D. Yemanov, Mike Nordell
(sin N°)	Mejora de rendimiento para los CPU de arquitectura IA32: aceleración de operaciones con índices	Mike Nordell
(sin N°)	Cambio en triggers universales: se permite el acceso a ambos contextos (OLD y NEW) en los triggers universales.	D. Yemanov
(sin N°)	Mejora en el optimizador: cuando un nodo 'igual' y otros nodos (geq, leq, between…) están disponibles para recuperación por índice, entonces usa el nodo 'igual' siempre en lugar de los otros.	A. Brinkman
(sin N°)	Grandes retrasos durante conexión/desconexión en WinXP.	A. Brinkman
(sin N°)	Limpieza general: se removió un montón de código sin uso.	Blas Rodriguez Somoza, Erik Kunze

N° seguimiento	Descripción	Contribuyente
523589	Una vista afecta el resultado de una consulta. Comentario: el problema era que los RSE's (dentro de una vista) no se marcaban como invariantes.	A. Brinkman
(sin N°)	Se cambió el comportamiento del modo de escritura forzada (forced writes): ahora, si está deshabilitado, se puede controlar la frecuencia de volcado de páginas usadas a disco (permite una mejor confiabilidad cuando FW es deshabilitado en las plataformas Win32).	Blas Rodriguez Somoza
(sin N°)	La base de datos de seguridad fue renombrada a security.fdb.	D. Yemanov
(sin N°)	Nuevo archivo de configuración: se publica finalmente firebird.conf	D. Yemanov
(sin N°)	Se agregan nuevas funciones LPAD y RPAD a la librería IB_UDF.	Juan Guerrero
(sin N°)	Algunas veces, GFIX no permitía especificar los parámetros "-user" y "-password" (error "incompatible swiches").	D. Yemanov
(sin N°)	Cache de la conexión a la base de datos de seguridad: ahora se mantiene en reserva la conexión a la base de datos de seguridad, permitiendo bajar el tiempo de las siguientes conexiones.	D. Yemanov
Mejoras	1. Se redujo la utilización de memoria del servidor 2. Cuando no hay memoria disponible para el ordenamiento, se hace E/S externa directa. Se incrementó el número de corrientes y predicados soportados por el optimizador.	D. Yemanov
508594	LEFT JOIN con vistas: un LEFT JOIN simple sobre una vista con una sola cláusula ON no usaba un índice aún si era posible.	A. Brinkman
(sin N°)	Nuevos juegos de caracteres: DOS737, DOS775, DOS858, DOS862, DOS864, DOS866, DOS869, WIN1255, WIN1256, WIN1257, ISO8859_3, ISO8859_4, ISO8859_5, ISO8859_6, ISO8859_7, ISO8859_8, ISO8859_9, ISO8859_13 NOTA las secuencias de ordenación (collations) para los juegos de caracteres anteriores no están disponibles todavía.	Blas Rodriguez Somoza
(sin N°)	Cambios en CREATE VIEW: Se deshabilita la subcláusula PLAN.	D. Yemanov
(sin N°)	Cambios en el comportamiento del seguimiento de agregados -- se introdujo compatibilidad hacia atrás dentro de los agregados. El campo más profundo dentro de un agregado determina adónde debería pertenecer el contexto del agregado.	A. Brinkman
(sin N°)	Mejoras en el optimizador: mejor optimización de consultas de conjunto "complejas" (LEFT JOIN, vistas, SPs, etc)	A. Brinkman
(error en alpha 5)	Las pérdidas de memoria más importantes están corregidas.	D. Yemanov

N° seguimiento	Descripción	Contribuyente
(sin N°)	Nuevas funciones en la API: funciones compatibles con IB7 para devolver la versión de la librería cliente -- isc_get_client_version(), isc_get_client_major_version(), isc_get_client_minor_version()	D. Yemanov
(sin N°)	Mejora a Sort/merge: la combinación (planes SORT MERGE) es hecha ahora vía un módulo de ordenamiento en memoria.	D. Yemanov
(sin N°)	Se ha cambiado el manejador interno de memoria para obtener mejor rendimiento.	N. Samofatov
(sin N°)	Cambios en la versión Win32: 1. Cambios en los nombres de objetos USER32 para permitir al servidor correr simultáneamente con IB/FB1. 2. El nombre del mapa para protocolo local (IPC) se ha cambiado de manera que la librería cliente v1.5 no es más compatible con las versiones previas vía IPC 3. Todos los nombres de protocolos de transporte (puerto y servicio INET, tubería WNET, mapa IPC) son ahora configurables vía firebird.conf.	D. Yemanov
(sin N°)	RDB$FIELD_LENGTH toma valores incorrectos para vistas que contienen concatenaciones de campos CHAR/VARCHAR largos.	D. Yemanov

Mejora	Mejora en triggers: se agregaron comprobaciones de acción (predicados INSERTING/UPDATING/DELETING)
	Ejemplo:
	if (INSERTING) then
	new.OPER_TYPE = 'I';
	else
	new.OPER_TYPE = 'U';
		D. Yemanov

(sin N°)	Los cursores (cláusula WHERE CURRENT OF) no podrían ser usados en triggers.	D. Yemanov
221921	ORDER BY no tiene efecto	A. Brinkman
213859	Subconsulta conectada con la cláusula 'IN'.	A. Brinkman
Mejora	Se permiten expresiones arbitrarias en la cláusula ORDER BY	N. Samofatov
(sin N°)	El servidor se caía cuando se usaba UNION en una vista y esa vista era usada en la cláusula WHERE dentro de una subconsulta.	A. Brinkman
(sin N°)	Limpieza genérica de código: estructuras dentro de la válvula Y	A. Peshkoff, N. Samofatov

N° seguimiento	Descripción	Contribuyente
Mejora	Los comentarios de línea única (--) se permiten ahora en cualquier posición de una sentencia SQL.	D. Yemanov
(sin N°)	"Request sycnhronization error" en sentencias BREAK.	D. Yemanov
625899	Bugcheck 291.	A. Peshkoff
(sin N°)	Cambio en PSQL: EXECUTE VARCHAR se renombra a EXECUTE STATEMENT.	A. Peshkoff
521952	Error "No current record for fetch operation"	A. Brinkman
(sin N°)	QLI no entiende el tipo de dato BIGINT	D. Yemanov
(sin N°)	El largo de variables de texto dentro de procedimientos/triggers no era copiado a la estructura del descriptor.	A. Brinkman
(sin N°)	Cambios en FIRST/SKIP y ORDER BY -- 1. Se implementó la cláusula ORDER BY en subconsultas. 2. Se deshabilitó FIRST/SKIP para vistas 3. Se permite cero como argumento para FIRST	D. Yemanov
(sin N°)	Sobrepasamiento de buffer (MAXPATHLEN) y nombre de directorio de función local reescrito.	Erik Kunze
(sin N°)	Hacer el mapeo de parámetros SQLDA consistente con el orden y número de parámetros en la cadena SQL de origen. NOTA Se puede habilitar el antiguo comportamiento de mapeo (para compatibilidad hacia atrás) usando el parámetro de configuración "OldParameterOrdering".	N. Samofatov
Mejora	Mejora del optimizador: permitir que subconsultas también usen índices cuando su consulta principal es un procedimiento almacenado.	A. Brinkman
(sin N°)	Se eliminó la limitación de tamaño en los pedidos (request).	D. Yemanov
(sin N°)	Manejo de colaciones y orden de nulos (principio/final) en la cláusula "order by" de uniones	N. Samofatov
(sin N°)	Limpieza genérica de código: cambios de nombre, nuevas macros seguras, soporte para mingw.	Erik Kunze, Ignacio J. Ortega, D. Sibiryakov
(sin N°)	Finalizada la implementación de bloqueo explícito de registros. Debería ser estable y consistente ahora.	N. Samofatov
Mejora	Mejora en el optimizador: mejor manejo de los nodos AND dentro de un nodo OR.	A. Brinkman

N° seguimiento	Descripción	Contribuyente
(sin N°)	Las excepciones dentro de bucles for/while en triggers no son manejados correctamente.	A. Peshkoff
623992	Doble barra en la cadena de conexión.	Paul Reeves, Mark O'Donohue
(sin N°)	Abrazo mortal durante algunas operaciones de base de datos.	A. Peshkoff
Mejora	Mejoras en el optimizador: si unos pocos índices con muy distinta selectividad podrían ser usados para recuperación de registros, solamente los mejores se usan mientras que los demás son ignorados.	D. Yemanov
(sin N°)	Problema con los identificadores con comillas en expresiones de plan.	N. Samofatov
Mejora	La arquitectura CS es soportada ahora en Win32, pero todavía no puede ser considerada estable, por lo que cualquier observación es bienvenida.	D. Yemanov
(sin N°)	Los procedimientos almacenados no son recompilados antes del borrado.	N. Samofatov
(sin N°)	Nueva secuencia de ordenación (collation) para el juego de caracteres WIN1251: WIN1251_UA para los lenguajes Ucraniano y Ruso.	D. Yemanov
(sin N°)	Cambio en la librería cliente: las rutinas de la API ya no se exportan por ordinales.	D. Yemanov
Mejora	Nuevo manejador de configuración: habilita la misma configuración basada en un archivo plano de texto para todas las plataformas soportadas.	D. Yemanov
Mejora	Mejoras del optimizador: mejor soporte para usar índices con "OR". Selecciona el mejor índice compuesto para los nodos "AND"	A. Brinkman
Mejora	Se agrega soporte para manejo explícito de puntos de referencia (savepoints) en DSQL.	N. Samofatov
(sin N°)	Limpieza de protocolo: el protocolo de red IPX/SPX ya no es soportado.	Sean Leyne
(sin N°)	Limpieza de plataformas obsoletas: algunas plataformas dejan de ser soportadas por el código fuente actual DELTA, IMP, DG_X86, M88K, UNIXWARE, Ultrix, NeXT, ALPHA_NT, DGUX, MPE/XL, DecOSF, SGI, HP700, Netware, MSDOS, SUN3_3	Sean Leyne
Mejora	Mejora al optimizador: se agrega soporte para detectar el uso de índices con subconsultas en consultas de agregado.	A. Brinkman
Mejora	Organizador de hilos mejorado para Win32 SS: ahora el servidor debería responder mejor bajo una carga grande.	A. Peshkoff

N° seguimiento	Descripción	Contribuyente
Mejora	Se agrega soporte para bloqueo explícito. El comportamiento de espera en los modos de transacción isc_tpb_wait no es estable todavía. Sintaxis: SELECT <...> [FOR UPDATE [OF col [, col ...]] [WITH LOCK]]	N. Samofatov
558364	Los triggers no compilan si se usa PLAN.	Ignacio J. Ortega
(sin N°)	Una transacción distribuida (2PC) no puede ser cancelada correctamente debido a errores de red.	Vlad Horsun, Erik Kunze
(sin N°)	Limpieza genérica: macros ISC_STATUS_LENGTH y MAXPATHLEN	Erik Kunze
496784	Cuando el optimizador encuentra índices para LEFT JOIN, trabaja como INNER JOIN. Se arregló un problema que causaba que los encuentros externos complejos produjeran resultados erróneos.	N. Samofatov
(sin N°)	El subtipo de BLOB se ignora en dominios de sistema generados para campos de expresión en vistas.	D. Yemanov
(sin N°)	Error de instalación corregido: instreg.exe no crea el valor de registro "GuardianOptions"	D. Yemanov
(sin N°)	Pérdida de recursos en el manejo de procedimientos recursivos DDL que causa la falla de algunas instrucciones DDL.	N. Samofatov
(sin N°)	Una restricción Check que refiera sólo a un campo se borra automáticamente cuando ese campo es eliminado.	N. Samofatov
451927	Nueva variable de sistema ROWS_AFFECTED en PSQL: devuelve el número de filas afectadas por la última sentencia INSERT/UPDATE/DELETE. Para cualquier otra sentencia distinta de INSERT/UPDATE/DELETE, el resultado es siempre cero.	D. Yemanov
446240	Mensajes de excepción dinámicos: permite generar una excepción con un mensaje diferente al que se asignó al crear la excepción. Sintaxis: EXCEPTION nombre [mensaje];	D. Yemanov
547383	Nuevas variables de sistema SQLCODE y GDSCODE, proveen acceso al código de un error capturado dentro de un bloque WHEN en PSQL. Fuera del bloque WHEN, devuelve 0 (indicando correcto).	D. Yemanov

N° seguimiento	Descripción	Contribuyente
(sin N°)	Semántica de relanzamiento de excepciones: permite lanzar nuevamente una excepción capturada en un bloque WHEN en PSQL. Sintaxis: EXCEPTION; No tiene efecto fuera de un bloque WHEN.	"Digitman"
(sin N°)	El servidor se cae durante la recolección de basura bajo gran carga.	N. Samofatov
Mejora	Compilación de metadatos diferida: resuelve muchas de las causas del error "object in use".	N. Samofatov
Mejora	Nuevo manejo de ordenamiento de valores NULL: permite que el ordenamiento de los nulos sea definido por el usuario.	N. Samofatov
(sin N°)	Gstat mostraba un valor erróneo para el elemento maxdup.	D. Kuzmenko
(sin N°)	Se utiliza una nueva clave de registro en Win32: SOFTWARE\FirebirdSQL\Firebird.	--
451925	Nombres de índices de restricciones definidos por el usuario: permiten que el nombre del índice que implementa una restricción sea el mismo que el de la restricción o definido por el usuario.	D. Yemanov
Mejora	Nueva sentencia RECREATE VIEW: atajo para el par de sentencias DROP VIEW / CREATE VIEW. Sintaxis: RECREATE VIEW nombre <definición_de_vista>;	D. Yemanov
(sin N°)	Un trigger cuyo nombre empieza con 'RDB$' no puede ser alterado o eliminado.	D. Yemanov
(sin N°)	Se renombraron los archivos de la distribución para distinguir que somos Firebird. Ahora son fbserver, fbclient, firebird.msg, etc. La librería cliente es ahora fbclient y debería ser utilizada por todos los nuevos proyectos basados en Firebird. Gds32 contiene solamente funciones que redireccionan las llamadas a fbclient.dll y se mantiene sólo por compatibilidad.	Varios
(Minor ODS upgrade)	Se agregan nuevos índices de sistema (RDB$INDEX_41, RDB$INDEX_42, RDB$INDEX_43), la versión ODS es ahora 10.1.	D. Yemanov, N. Samofatov
451935	Nueva sentencia CREATE OR ALTER para triggers y procedimientos almacenados, permiten crear o alterar un objeto de la base de datos de acuerdo a si éste ya existe o no. Sintaxis: CREATE OR ALTER nombre <definición_de_objeto>;	D. Yemanov

N° seguimiento	Descripción	Contribuyente
(sin N°)	Luego de algunos cambios de metadata en la base de datos se pierden dependencias (como DB$34).	D. Yemanov
(sin N°)	Declaración mejorada de variables locales: se simplifica la sintaxis y permite declarar y definir una variable al mismo tiempo Sintaxis: DECLARE [VARIABLE] nombre <tipo_de_variable> [{'=' | DEFAULT} valor]; Ejemplo: DECLARE my_var INTEGER = 123;	Claudio Valderrama
(sin N°)	Se deshabilita la sentencia BREAK para triggers (como EXIT) debido a limitaciones internas.	D. Yemanov
555839, 546274	Agrupamiento mejorado: permite agrupar por funciones internas y subconsultas. También permite agrupar por ordinal (posición de columna, o sea grado de columna en el conjunto de salida).	A. Brinkman
451917	Nueva función interna COALESCE que permite que el valor de una columna sea calculado a partir de un número de expresiones, el valor de la columna es la primera expresión que devuelva un valor no nulo.	A. Brinkman
451917	Nueva función interna NULLIF devuelve NULL para una sub-expresión si tiene un valor específico, de otra manera devuelve el valor de la sub-expresión.	A. Brinkman
451917	Nueva función interna CASE permite que se determine el resultado de una columna por el resultado de una expresión de selección case.	A. Brinkman
545725	La operación sweep automática/en segundo plano falla.	A. Peshkoff
(sin N°)	El servidor se cae cuando las estructuras XSQLDA no son preparadas para todos los parámetros de la sentencia.	D. Yemanov
(sin N°)	PSQL: se habilita soporte para bloques BEGIN...END vacíos.	D. Yemanov
567931	Agujero de seguridad en metadata parcialmente solucionado.	D. Yemanov
(sin N°)	Los arreglos de BigInt no funcionan	Artem Petkevych
437859	Se implementa execute procedure y concatenación de cadenas, permitiendo que cualquier expresión sea usada como un parámetro a un SP	D. Yemanov
562417	Un agregado concatenaba un caracter vacío.	D. Yemanov
Mejora	Se agregó soporte de historia de comandos a ISQL	M. O'Donohue

N° seguimiento	Descripción	Contribuyente
446206	Nuevo tipo de datos BIGINT permite el uso de números exactos de 64 bits en forma nativa en SQL (solamente dialecto 3)	D. Yemanov
451922	Triggers universales permiten que un solo trigger sea disparado por varios tipos de acciones.	D. Yemanov
446238, 446243	Nuevas variables de sistema CONNECTION_ID y TRANSACTION_ID en PSQL. Devuelve el identificador interno apropiado, almacenado en la página de cabecera de la base de datos.	D. Yemanov
446180	Alias de servidor para bases de datos: permite la conexión a cualquier base de datos usando el nombre del alias en lugar de la ruta física. La lista de los alias conocidos de bases de datos se almacena en el archivo aliases.conf en la raíz de instalación del servidor. Ejemplo: Entrada de alias en el archivo de configuración: my_database = c:\dbs\my\database.gdb cadena de conexión en la aplicación: localhost:my_database	D. Yemanov
(sin N°)	Nuevo manejador de agregados (plugins) e interfaz INTL	John Bellardo
Mejora	Ordenamiento en memoria: si se usa un plan SORT en una sentencia SQL, el ordenamiento se hace en memoria. Si no hay memoria suficiente para la operación, se vuelve al método anterior de uso de un archivo temporal.	D. Yemanov
538201	Caída del servidor con extract from cuando la fecha es nula.	Claudio Valderrama
446256	Nueva sentencia PSQL: EXECUTE VARCHAR permite la ejecución de sentencias SQL dinámicas en SPs/triggers (posteriormente renombrado como EXECUTE STATEMENT)	A. Peshkoff
(sin N°)	Limpieza importante de código.	Sean Leyne, Erik Kunze
(sin N°)	Nuevo manejador de memoria	John Bellardo
(sin N°)	Nueva lógica de manejo de excepciones	Mike Nordell, John Bellardo
(sin N°)	Nueva configuración basada en autoconf	John Bellardo, M. O'Donohue, Erik Kunze
(sin N°)	Código portado de C a C++.	Mike Nordell, John Bellardo, M. O'Donohue

Equipo de traducción: *Antonio Sala, Ernesto Cullen, Marc Guillot, Juan Tendero y José Velasco*