Firebird™ Versão 1.5

Notas da Versão v.1.5

5 Fevereiro 2004 – Versão do Documento 1.08

Conteúdo

Notas Gerais

A base de dados Firebird™ foi desenvolvida por um grupo independente de programadores a partir do código do InterBase™ que foi lançado pela Borland sobre a "InterBase Public License v.1.0" em 25 de Julho de 2000.

O desenvolvimento do código do Firebird 2 começou antes do término do desenvolvimento da primeira versão do Firebird 1, com o porte do código de C para C++ e a primeira limpeza geral do código. O Firebird 1.5 é o primeiro lançamento do código do Firebird 2. É um passo significativo para os programadores e para todo o projecto Firebird, mas ainda não é um fim. Enquanto o Firebird 1.5 avança para uma nova Versão, outras alterações de grande impacto são entretanto efectuadas para avançar no desenvolvimento do Firebird 2.

O Firebird 1.0.x continua em modo de manutenção – resolução de erros e alguns melhorias do Firebird 1.5 são transferidos para a versão 1.0.

Os Binários do Firebird 1.5

Os binários do Firebird podem ser obtidos a partir do website -
http://sourceforge.net/project/showfiles.php?group_id=9028

"Version Strings" do Firebird 1.5

Win32: "WI-V1.5.0.nnnn Firebird 1.5"
Linux: "LI-V1.5.0.nnnn Firebird 1.5"
E por aí adiante, sendo nnnn o número do "build"

Reporte ao capítulo de Documentação para obter informação sobre a documentação recomendada.

Novas Funções

Novo código, melhor optimização

Esta versão foi desenvolvida a partir do código rescrito do C original para C++, um processo iniciado em 2000 por Mike Nordell. O código foi extensamente limpo e revisto, o trabalho de remoção de erros foi continuado, foi desenvolvido um novo gestor de memória, e foram introduzidas novas funções na linguagem. Ainda durante o desenvolvimento desta versão 1.5, Arno Brickman e outros fizeram um excelente trabalho no "SQL query optimizer", com várias optimizações e correcções, que resultaram numa melhoria de velocidade na ordem dos 30 a mais de 60 por cento.

Arquitectura

Duas novidades importantes nas plataformas Windows são o "Classic server" e o "Embedded server".

- Não existia uma versão do modelo "Classic" para windows há mais de 8 anos. Este modelo pode utilizar multi-processadores, algo que ainda é limitado na versão "SuperServer" para Windows. Embora utilizável, a versão do modelo "Classic" deve ser olhada ainda como experimental.

- "Embedded server" (ou servidor embutido) é um dll que junta um cliente com o servidor SuperServer e que permite a construção rápida e eficiente de uma aplicação monoposto.

Várias e importantes extensões foram adicionadas à linguagem desde a versão 1.0.x, incluindo as expressões de SQL-92 CASE, COALESCE e NULLIF. Para informações sobre o detalhe destas e outras extensões à linguagem, refira-se ao capítulo Extensões à Linguagem deste mesmo documento.

Módulos Instalados e Segurança

Se tem estado a utilizar o Firebird 1.0.x até hoje, vai notar várias diferenças nos nomes dos módulos e nas regras para o seu acesso e localização. De seguida serão mostrados alguns detalhes sobre este assunto, mas informação mais detalhada sobre a instalação e configuração segue mais adiante neste artigo.

1. A maior parte dos módulos e constantes mudaram de nome. Na maior parte dos casos, os novos nomes possuem alguma variante das palavras "firebird" ou "fb". Por exemplo, a biblioteca API está agora localizada numa biblioteca partilhada chamada "fbclient.dll" em Windows e "libfbclient.so" nas outras plataformas. A excepção a esta regra é a base de dados de segurança, antigamente nomeada "isc4.gdb", e que agora tem o nome "security.fdb".
2. Os ficheiros externos utilizados pelo servidor (bibliotecas UDF, filtros BLOBs, bibliotecas de "character set", tabelas externas) estão agora sujeitas a níveis de protecção no sistema de

ficheiros que, em alguns casos, são por defeito diferentes dos utilizados nas versões 1.0.x e pelo Interbase.

3. O novo ficheiro de configuração do servidor, firebird.conf, que substitui o antigo ibconfig (Windows) e isc_config (outras plataformas) contém muitas mais opções de configuração, assim como uma nova organização e documentação integrada.
4. A possibilidade de criar 'aliases' para as bases de dados existe na versão 1.5. Agora é permitido, opcionalmente, esconder a localização da base de dados atribuindo-lhe um nome, ou "alias". A localização original dos ficheiros encontra-se noutro ficheiro, aliases.conf. O propósito principal desta função é, porém, esconder a localização física dos ficheiros de um "sniffer" que ausculte a rede.
5. Por defeito (e nas versões anteriores) para as plataformas Windows é necessário ter o utilizador "local system" a correr o programa que instala o servidor Firebird como serviço, no arranque do sistema. Esta vulnerabilidade de segurança pode ser aproveitada por um hacker, como uma porta para obter acesso a toda a máquina. A versão 1.5 deste instalador (instsvc.exe) aceita agora um nome de utilizador para a instalação do serviço. É altamente recomendável que se crie um utilizador Firebird para este propósito, e que se utilize esta nova função se o seu servidor está conectado à internet de alguma forma.

"Trimming" de Campos Varchar em protocolos remotos

Foi terminado o trabalho de completar esta função no cliente 1.5, e agora os campos Varchar são transferidos "right-trimmed" pela rede (i.e., os caracteres de "espaços" à direita não são transmitidos), ocupando apenas o número de caracteres utilizados mais 2 bytes.
NOTA: Como é o cliente que pede ao servidor para elaborar o "trim" dos varchars, o cliente Firebird 1.5 (fbclient.dll ou libfbclient.so) irá elaborar o trim, mesmo quando conectado a um servidor de versão anterior à 1.5. Se utilizador um cliente antigo, não obterá o "trimming", mesmo que esteja conectado a um servidor 1.5 ou posterior.

Semântica de Triggers para Multi-acções

Agora existem triggers Universais, o que permite elaborar um único trigger que irá agir em qualquer que seja a operação: "insert", "update" ou "delete". Esta nova semântica reduz a composição e a manutenção de Triggers sem eliminar a possibilidade de possuir múltiplos triggers por fase.

Melhoria na nomenclatura de "constraints"

Os índices que forçam a integridade podem agora possuir um nome definido pelo utilizador.
Um aviso, se usa esta funcionalidade, a base de dados não será compatível com a v.1.0.x ou com o InterBase®.

Número máximo de índices por Tabela aumentado

Agora - tanto na versão 1.0 como nesta versão — o número máximo de índices por tabela passou de 64 para 256.

"Pessimistic locking"

Para os casos raros em que se necessita de impor um "lock" pessimista, esta versão adiciona uma nova sintaxe para colocar um "reader's lock" em linhas enquanto são transferidas para o cliente. Use com cuidado.

"Cache de Conexão" à base de dados de Segurança

A conexão à base de dados de segurança é mantida em cache nos modelos SS. Isto é, a security.fdb é carregada quando a primeira conexão é feita, e é mantida em cache até que todas as conexões clientes desapareçam.

Melhores mensagens de erro

Sempre que possível, as mensagens que reportam erros de SQL são agora mais detalhadas. É de realçar que se podem encontrar mensagens bizarras ao utilizar um ficheiro interbase.msg ou firebird.msg antigo.

Serviços API na versão Classic do Linux

Está disponível o suporte limitado para os Serviços API na versão do "Classic server" para o Linux. Os Serviços disponíveis permitem usar o gbak (backup/restore) e gfix (validar base de dados, shutdown/online, etc). Outros (gstat, registos do servidor, etc.) não foram testados e provavelmente não funcionam.

Alterações nas bibliotecas do Cliente

Clientes Windows

A biblioteca cliente tem agora o nome "fbclient.dll". Todos os utilitários do servidor (gbak, gfix, etc) utilizam apenas esta biblioteca. É recomendado conectar as novas aplicações directamente a fbclient.dll, sem necessidade da gds32.dll.

Para manter a compatibilidade com as aplicações existentes, é possível criar uma cópia do fbclient.dll com o nome "gds32.dll" usando um utilitário chamado instclient.exe. Para informação mais detalhada, consulte a secção de instalação e as notas de instalação distribuidas com o seu kit windows.

Clientes Linux

A biblioteca cliente do Superserver chama-se agora "libfbclient.so". Para compatibilidade com aplicações existentes, é instalado um symlink "libgds.so" que aponta para libfbclient.so. A biblioteca local para aplicações embedded que se conectam ao servidor Classic foi renomeada para libfbembed.so.

Ficheiros e Módulos renomeados

Plataforma	Módulo	Firebird 1.0	Firebird 1.5	Notas especiais
Todas	Base de Dados de Segurança	Isc4.gdb	security.fdb	
Todas	Ficheiro de Mensagens	Interbase.msg	firebird.msg	
Todas	Ficheiro de Log	interbase.log	firebird.log	
Todas	Versão de ODS	10	10.1	Novo ODS (10.1). Não causa nenhuma incompatibilidade com versões anteriores de ODS mas a versão é agora actualizada automaticamente. Tanto o Firebird 1.0 como o 1.5 podem servir BD's de ODS 10.0 e 10.1. Porém, a operação de backup/restore é ainda o procedimento recomendado para migrar BD's para uma versão diferente do servidor.

Plataforma	Módulo	Firebird 1.0	Firebird 1.5	Notas especiais
Todas	Variáveis de Ambiente	INTERBASE INTERBASE_LOCK INTERBASE_MSG INTERBASE_TMP	FIREBIRD FIREBIRD_LOCK FIREBIRD_MSG FIREBIRD_TMP	Aponta para directoria root da instalação Aponta para o ficheiro lock Aponta para o ficheiro de mensagens Aponta para uma directoria usada para espaço de ordenação
Todas	Base de dados de Segurança	Isc4.gdb	security.fdb	
Linux	Binário do Classic server	Gds_inet_server	fb_inet_server	
Linux	Gestor de lock Classic	ib_lock_mgr	fb_lock_mgr	
Linux	Controlo Superserver	ibmgr.bin	fbmgr.bin	
Linux	Binário Superserver	ibserver	fbserver	
Linux	Ficheiro de Configuração	isc_config	firebird.conf	
Linux	Biblioteca Cliente	Libgds.so	libfbclient.so libfbembed.so	Cliente remoto "Thread-safe" e cliente TCP/IP local loopback para Superserver Cliente local (monoposto, não-thread-safe) para Classic
Linux	Symlink para Biblioteca cliente para Compatibilidade	N/A	libgds.so	
Windows	Guardian	ibguard.exe	fbguard.exe	
Windows	Binário Superserver	ibserver.exe	fbserver.exe	Não tem suporte a SMP.
Windows	Binário Classic	N/A	fb_inet_server.exe	Conexões locais Windows não disponíveis. TCP/IP, NetBEUI OK. Suporte a SMP.
Windows	Biblioteca Cliente	gds32.dll	fbclient.dll	Os utilitários do servidor de versão 1.5, e todas as novas aplicações, apenas necessitam da fbclient.dll. Veja as notas abaixo sobre compatibilidade do gds32.dll para aplicações antigas.
Windows	Ficheiro de Configuração	ibconfig	firebird.conf	

Plataforma	Módulo	Firebird 1.0	Firebird 1.5	Notas especiais
Windows	Local IPC port	InterBaseAPI	FirebirdAPI	Com as configurações por defeito não pode estabelecer conexões locais com a biblioteca cliente antiga (gds32.dll). Se necessário, pode configurar o servidor para usar o nome antigo de IPC map, via firebird.conf.
Windows	"Registry key" Por defeito	HKLM\SOFTWARE\Borland\InterBase	HKLM\SOFTWARE\Firebird Project\Firebird Server\Instances	A directoria é guardada no parâmetro "DefaultInstance". i.e., não existe mais a chave "CurrentVersion", e a chave "RootDirectory" é substituída por "DefaultInstance".
Os novos nomes dos serviços no Windows são "Firebird Guardian – DefaultInstance" e "Firebird Server – DefaultInstance".				

Compatibilidade

On-disk structure (ODS)

A On-Disk Structure do Firebird 1.5 é designada por ODS 10.1. Este pequeno upgrade de ODS foi necessário pelas seguintes razões:

- Três novos índices nas tabelas do sistema
- Pequenas alterações no BLR de dois triggers de sistema
- Melhorias na codificação do RDB$TRIGGER_TYPE.

Algumas outras novas funcionalidades que requerem alterações ao ODS foram adiadas para a versão 2. Até lá, deve ser possível o transporte directo de bases de dados Firebird 1.0.x. É recomendado o backup de bases Firebird 1.0.x antes de as transportar para um servidor 1.5.

Base de Dados InterBase™

Se tencionar "brincar" com o Firebird usando uma base de dados InterBase, com a intenção de reverter para o Interbase posteriormente, tome todas as precauções e faça um backup utilizando a versão correspondente do gbak do Interbase. Para começar a trabalhar com o Firebird 1.5, utilize o gbak deste para fazer o restore do seu backup.
O "Operations Guide" da Documentação do InterBase® 6.0 beta contém a sintaxe dos comandos do utilitário gbak que permite elaborar este backup e restore.

As bases de dados do IB 7.x e provavelmente do IB 6.5 poderão trabalhar incorrectamente depois de migrar para o FB 1.5 via backup/restore, se algumas das novas funcionalidades específicas do IB tiverem sido utilizadas.

Nomes de Ficheiros e Localização

Nesta versão, um número substancial de ficheiros possuem um novo nome, fruto do trabalho da substituição de nomes herdados pelo InterBase® 6. Leia a secção de Nomes de Ficheiros e Localização para obter descrições e algumas recomendações.

Servidores a correrem em Simultâneo

As alterações elaboradas aos nomes dos objectos de sistema permitem que o Firebird 1.5 seja instalado e utilizado em simultâneo numa máquina que possua o InterBase ou o Firebird 1.0.x instalado. Em Windows, o FB 1.5 ainda usa outra "Registry key". Se configurar o servidor para usar outras portas de rede, é possível correr várias instancias do servidor em simultâneo, ou correr a versão 1.5 ao mesmo tempo que o IB ou o FB 1.0.x.

Voltando ao Firebird 1.0.x

Devido ao enorme número de erros resolvidos, o comportamento da base de dados pode variar se fizer um "downgrade" de uma base v.1.5 para v.1.0.x. Concretamente, se criar uma "primary", "unique" ou "foreign keys" como "constraints", o valor por defeito do nome dos índices será incompatível com a v.1.0.x. Esteja atento a um futuro README detalhando estes casos à medida que vão aparecendo.

Compatibilidades em Linux

Devido a um histórico de problemas envolvendo o compilador GNU C++, as versões de Linux do Firebird 1.5 necessitam de versões glibc superiores às que eram usadas anteriormente. Isto faz com que, infelizmente, estejamos num período em que é difícil prever que distribuições Linux poderão facilmente instalar e correr os binários da versão 1.5. O quadro seguinte poderá ajudar. Porém,

agradecemos toda a informação que nos possa dar. Por favor partilhe a sua experiência com estas e outras distribuições linux no forum firebird-devel.

Distribuição	Nível	Classic	Superserver
Red Hat	7.x	Não	Não
	8.0	Sim	Sim
	9.0	Sim	Sim
Mandrake	8.x	Não	Não
	9.0, 9.1, 9.2	Sim	Sim
SuSE	7.3	Sim – pacotes de instalação libgcc-3.2-44.i586.rpm & libstdc++-3.2-44.i586.rpm antes de instalar o Firebird 1.5.	Desconhecido
	8.0, 8.1	Sim	8.0 Sim (8.1 Desconhecido)

Extensões à Linguagem

TIPOS DE CAMPOS

(1.5) Novos tipos de Campos SQL Nativos

BIGINT
Tipo numérico exacto, SQL99-compliant, com uma escala de zero. Disponível apenas no dialecto 3.

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

METADATA

(1.5) Melhorias a "named constraints"

Dmitry Yemanov

Os índices que forçam "named constraints" podem agora ter um nome atribuído, com identificadores definidos pelo utilizador.

Em versões anteriores, embora fosse possível atribuir um nome a "constraints" do tipo PRIMARY, FOREIGN KEY e UNIQUE, o identificador do índice gerado automaticamente era definido pelo sistema, p.ex. RDB$FOREIGN13 e não podia ser alterado. Esta continua a ser a regra quando esta nova funcionalidade não é utilizada.

Porém, foram adicionadas extensões à linguagem para permitir:
a) a um índice gerado automaticamente pelo sistema receber automaticamente como identificador o mesmo nome que a "constraint" que ele força.
b) a um índice que força uma "constraint", com ou sem nome, possuir ele mesmo um nome atribuído por um identificador e opcionalmente ser construído em ordem DESCENDING.
NOTA: Não é ainda possível utilizar um índice já existente.

Sintaxe
```
...
[ADD] CONSTRAINT [<identificador-da-constraint>]
<tipo-de-constraint > <definição-de-constraint>
[USING [ASC[ENDING] | DESC[ENDING]] INDEX <nome_do_indice>]
```

Aviso: Tem que garantir que os índices "foreign key" e "primary key" usam o **mesmo tipo de ordenação** (DESC | ASC).

Exemplos
i) "Named constraint" e atribuição de nome ao índice

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

A declaração seguinte irá criar uma "primary key constraint" com o nome PK_ATEST e um índice, descendente, com o nome IDX_PK_ATEST:

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

ii) Uma alternativa a i) seria:

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

iii) Esta declaração cria a tabela ATEST com uma primary key PK_ATEST. O índice correspondente terá o mesmo nome: PK_ATEST.

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

(1.5) Triggers de Multi-acção

Dmitry Yemanov

Os Triggers foram melhorados para permitir o uso condicional em várias operações.

Sintaxe

```
CREATE TRIGGER nome FOR tabela
  [ACTIVE | INACTIVE]
  {BEFORE | AFTER} <acção_múltipla >
  [POSITION número]
AS Corpo_Do_Trigger
```

<acção_múltipla> ::= <acção_simples> [OR <acção_simples> [OR <acção_simples>]]
<acção_simples> ::= {INSERT | UPDATE | DELETE}

Exemplos

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
...;
```

Alteração do ODS

O campo RDB$TRIGGER_TYPE (relação RDB$TRIGGERS) foi estendido para permitir estas operações de triggers complexas. Se desejar obter mais detalhes, leia o documento readme.universal_triggers.txt na directoria /doc/sql.extensions na árvore CVS do Firebird.
Nota (s):

1. Triggers de apenas uma acção são totalmente compatíveis ao nível do ODS com o FB 1.0.
2. O tipo de RDB$TRIGGER_TYPE depende da sua ordem, i.e., BEFORE INSERT OR UPDATE e BEFORE UPDATE OR INSERT serão codificados de forma diferente, embora partilhem a mesma semântica e serão executados exactamente da mesma forma.
3. As variáveis de contexto OLD e NEW estão disponíveis em triggers de múltipla acção. Se a forma de invocar o trigger proibir o seu uso (por exemplo, OLD numa operação de INSERT), todos os campos nesse contexto serão considerados como NULL. A tentativa de lhes dar um valor num contexto inválido irá provocar uma excepção.
4. Foram criadas novas variáveis de contexto - INSERTING/UPDATING/DELETING – podem ser utilizadas para verificar a operação em "runtime". (Veja mais à frente.)

(1.5) RECREATE VIEW

Exactamente o mesmo que CREATE VIEW se a view ainda não existir. Se existir, RECREATE VIEW irá tentar destruir e criar um objecto novo. RECREATE VIEW irá falhar se o objecto estiver a ser utilizado. Utiliza a mesma sintaxe que CREATE VIEW.

(1.5) CREATE OR ALTER {TRIGGER | PROCEDURE }

Esta declaração irá ou criar um novo trigger ou procedure (se ainda não existirem) ou alterá-lo (se existir) e voltar a recompilá-lo. A sintaxe CREATE OR ALTER preserva as dependências e permissões existentes.

A sintaxe é a mesma de CREATE TRIGGER | CREATE PROCEDURE, respectivamente, excepto a declaração "OR ALTER".

(1.5) NULLs em "unique constraints" e índices

Dmitry Yemanov

Agora é possível aplicar uma "constraint" UNIQUE ou um índice UNIQUE a colunas que não possuam a "constraint" NOT NULL. Este facto respeita o SQL-99. Seja cauteloso na utilização desta nova funcionalidade se planeia reverter a sua base de dados para Firebird 1.0.x ou qualquer versão do Interbase. A lógica utilizada é:

```
< definição de constraint ou definição de índice > ::=
< especificação única > ( < lista de colunas únicas UCL> )
< especificação única > ::=
{{[nome-constraint]UNIQUE | UNIQUE INDEX nome-índice]} | [nome-constraint]
PRIMARY KEY}
```

1) Se a <especificação única> especificar PRIMARY KEY, então a condição de procura (Search Condition) será:

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

ii) Caso contrário, a condição de procura (Search Condition) será:

```
UNIQUE ( SELECT UCL FROM TN )
```

Onde UNIQUE indica que se não existirem duas linhas em (SELECT UCL FROM TN) em que o valor de cada coluna não é NULL e não é distinto do valor correspondente na outra linha, então o resultado é verdadeiro; caso contrário o resultado é Falso.

Esta "constraint" permite a existência de apenas linhas em que a declaração SC mencionada for verdadeira. Isto indica que **uma PRIMARY KEY não permite NULLs** enquanto uma constraint UNIQUE permite um número arbitrário de NULLs. Para grupos de resultados de múltiplas colunas de (SELECT UCL FROM TN), são aplicadas as regras comuns para NULLs, i.e. (1, NULL) é distinto de (NULL, 1) e um (NULL, NULL) é distinto de qualquer outro (NULL, NULL).

DSQL

(1.5) Expressões e variáveis como argumentos de procedure

Dmitry Yemanov

Chamadas a EXECUTE PROCEDURE ProcName(<Lista-De-Argumentos >) e
SELECT <lista-Do-Output> FROM ProcName(<Lista-De-Argumentos >) agora podem aceitar como argumentos variáveis locais (em PSQL) e expressões (em DSQL e PSQL).

(1.5) Expressões CASE

Arno Brinkman

a) CASE

Permitem que o resultado de uma coluna seja determinado com base na avaliação de um grupo de condições exclusivas.

Sintaxe

```
<expressão case> ::=
    <abreviatura de case>  | <especificação case>

<abreviatura de case> ::=
    NULLIF <parent. esquerdo> <expressão valor> <vírgula> <expressão valor> <parent. direito>
  | COALESCE <parent. esquerdo> <expressão valor> { <vírgula> <expressão valor> }...<parent. direito>

<especificação case> ::=
    <case simples>  | <case com procura>

<case simples> ::=
  CASE <expressão valor>  <cláusula when simples>...
    [ <cláusula else> ]
  END

<case> ::=
  CASE <cláusula when com procura>...
    [ <cláusula else> ]
  END

<cláusula when simples> ::= WHEN <operador de when> THEN <resultado>
<cláusula when com procura > ::= WHEN <condição de procura> THEN <resultado>
<operador de when> ::= <expressão valor>
```

<cláusula else> ::= ELSE <resultado>
<resultado> ::= <expressão resultado> | NULL
<expressão resultado> ::= <expressão valor>

Exemplos

i) simples

```
SELECT
    o.ID,
    o.Description,
    CASE o.Status
      WHEN 1 THEN 'Confirmado'
      WHEN 2 THEN 'Em produção'
      WHEN 3 THEN 'Disponível'
      WHEN 4 THEN 'Enviado'
      ELSE 'Status Desconhecido: ''' || o.Status || ''''
    END
FROM Orders o;
```

ii) search

```
  SELECT
    o.ID,
    o.Description,
    CASE
      WHEN (o.Status IS NULL) THEN 'Novo'
      WHEN (o.Status = 1) THEN 'Confirmado'
      WHEN (o.Status = 3) THEN 'Em produção'
      WHEN (o.Status = 4) THEN 'Disponivel'
      WHEN (o.Status = 5) THEN 'Enviado'
      ELSE 'Status Desconhecido: ''' || o.Status || ''''
    END
  FROM Orders o;
```

b) COALESCE

Permite que o valor de uma coluna seja calculado por um número de expressões, onde a primeira destas expressões que devolva um valor não NULL é usada para determinar o valor a devolver.

Formato
<abreviatura de case> ::=
| COALESCE <parent. esquerdo> <expressão valor> { <vírgula> <expressão valor> }... <parent. direito>

Regras de Sintaxe
i) COALESCE (V1, V2) é equivalente às seguintes expressões <case >:
CASE WHEN V1 IS NOT NULL THEN V1 ELSE V2 END
ii) COALESCE (V1, V2,..., Vn), para n >= 3, é equivalente a um <case>:
CASE WHEN V1 IS NOT NULL THEN V1 ELSE COALESCE (V2,...,Vn) END

Exemplos

```
SELECT
    PROJ_NAME AS NomeDoProjecto,
    COALESCE(e.FULL_NAME,'[> Não Atribuído <]') AS NomeDoFuncionario
```

```
FROM
    PROJECT p
    LEFT JOIN EMPLOYEE e ON (e.EMP_NO = p.TEAM_LEADER);

SELECT
    COALESCE(Phone,MobilePhone,'Desconhecido') AS "NumeroTelefone"
FROM
    Relations
```

c) NULLIF

Devolve NULL para uma sub-expressão se esta possui um valor específico, caso contrário devolve o valor da sub-expressão.

Formato
<abreviatura de case> ::=
NULLIF <parent. esquerdo> <expressão valor> <vírgula> <expressão valor> <parent. direito>

Regras de Sintaxe
NULLIF (V1, V2) é equivalente ao <case >:
CASE WHEN V1 = V2 THEN NULL ELSE V1 END

Exemplo

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

(1.5) "Savepoints" compatíveis com SQL99

Nickolay Samofatov

"Savepoints" definidos pelo utilizador (também conhecidos por "*nested transactions*") disponibilizam um método conveniente para lidar com erros de lógica sem necessidade de fazer um rollback da transacção. Disponível apenas em DSQL.

Utilize uma declaração SAVEPOINT para identificar um ponto de transacção, ao qual poderá voltar posteriormente, fazendo um "Rollback" parcial.

```
SAVEPOINT <identificador>;
```

<identificador> especifica o nome do savepoint a criar. Depois de criado um "savepoint" você pode continuar o processamento, elaborar um commit, rollback de toda a transacção, ou apenas o rollback até ao "savepoint".

Os nomes dos "Savepoint" têm que ser distintos dentro de uma transacção. Se criar um segundo "Savepoint" com o mesmo identificador, o primeiro será destruído.

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

Esta declaração executa as seguintes operações:
- Rollback de alterações executadas na transacção após o "savepoint"
- Destrói todos os "savepoints" criados após este "savepoint". Este Savepoint é mantido, pelo que poderá efectuar rollback para o mesmo "savepoint" múltiplas vezes. Os savepoints anteriores a este serão também mantidos.

- Liberta todos os locks obtidos implícita e explicitamente após o "savepoint". Outras transacções que tenham requerido acesso a linhas que possuam um "lock" criado após o savepoint terão de continuar a aguardar até a transacção ser terminada, por commit ou rollback. Outras transacções que não tenham requerido as linhas, poderão requerer e aceder às linhas imediatamente. Nota: este comportamento pode ser alterado em futuras versões.

O "undo log" do "savepoint" pode consumir uma grande quantidade de memória no servidor, principalmente se fizer modificações sobre os mesmos registos múltiplas vezes na mesma transacção. Utilize a declaração RELEASE SAVEPOINT para libertar recursos consumidos pela manutenção do próprio "savepoint".

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

A declaração RELEASE SAVEPOINT destrói o "savepoint" <identificador> do contexto da transacção. A menos que especifique ONLY, todos os "savepoints" criados desde o "savepoint" <identificador> serão também eles destruídos.

Exemplos de utilização de "savepoints"

```
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; -- não devolve nenhuma linha
rollback to y;
select * from test; -- devolve duas linhas
rollback;
select * from test; -- devolve uma linha
```

"Savepoints" internos

Por defeito, o "engine" utiliza um savepoint de sistema automático, a nível da transacção, para efectuar um rollback de uma transacção. Quando se elabora um ROLLBACK, todas as alterações efectuadas na transacção são repostas via um "savepoint" a nível da transacção, e posteriormente é feito um COMMIT da transacção. Esta lógica permite reduzir a quantidade de "garbage collection" originada por transacções "rolled back".

Quando o volume de dados modificados sob um savepoint ao nível de uma transacção é demasiado grande (10^4-10^6 registos afectados) o "engine" liberta o "savepoint" a nível da transacção, e utiliza o mecanismo TIP para elaborar um Roll Back da transacção, se necessário. Se é esperado um grande volume de alterações dentro de uma transacção, pode utilizar o TPB flag isc_tpb_no_auto_para evitar a criação de um savepoint a nível da transacção.

Savepoints e PSQL

Ao implementar "savepoints" definidos pelo utilizador ao nível do PSQL podem-se quebrar as regras de atomicidade para as declarações, incluindo as declarações de chamadas a procedures. O Firebird permite o tratamento de excepções em PSQL para reverter alterações elaboradas em Stored Procedures e Triggers. Cada declaração SQL/PSQL é executada sobre um sistema de "savepoints" internos automáticos, pelo que ou toda a declaração é executada com sucesso, ou TODAS as alterações efectuadas são "rolled back" e uma excepção ocorre. Cada bloco de tratamento de excepções PSQL está incluído num sistema automático de savepoints.

(1.5) Locking Explicito

Nickolay Samofatov

O uso da cláusula adicional WITH LOCK permite a capacidade limitada de lock pessimista explícito, para um uso com cautela em condições em que as linhas afectadas sejam a) extremamente pequenas (idealmente apenas uma) e b) controladas com precisão a partir do código da aplicação.

NOTA: A necessidade de um lock pessimista no Firebird é rara, e deve ser compreendida antes de se considerar o uso desta extensão.

Sintaxe

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

Se a cláusula WITH LOCK for bem sucedida, irá estabelecer um lock nas linhas seleccionadas e impedir que outras transacções obtenham acessos de escrita em qualquer uma dessas linhas, ou dependentes, até que esta transacção termine.

Se a cláusula FOR UPDATE for usada, o lock será aplicado a todas as linhas, uma a uma, assim que é trazida para o cache do lado do servidor. É assim possível que um lock que pareça ser bem sucedido quando requerido, falhe subsequentemente, quando existir uma tentativa de trazer uma linha sobre a qual tenha sido feito um lock por uma outra transacção.

É essencial que se compreenda os efeitos do isolamento de uma transacção, assim como outros dos seus atributos, antes de se implementar um mecanismo de lock explicito numa aplicação.

SELECT... WITH LOCK está disponível em DSQL e PSQL. Pode ser bem sucedido apenas a nível de topo, num SELECT de uma única tabela. Não está disponível a nível de uma sub-query, nem em joins. Não pode ser utilizado com o operador DISTINCT, uma cláusula GROUP BY, ou qualquer outra agregação. Não pode ser utilizado nem numa view, nem com uma tabela externa, nem no output de uma stored procedure seleccionável.

Compreendendo a Clausula WITH LOCK

Como o "engine" considera cada registo contido numa declaração lock explícita, vai devolver ou a versão que foi mais recentemente "commit", independentemente do estado da base de dados quando a declaração foi submetida, ou uma excepção.

O comportamento do "Wait" e o retorno de conflitos irá depender dos parâmetros da transacção especificados no bloco TPB.

Modo TPB	***Comportamento***
isc_tpb_consistency	Locks explícitos são eliminados por locks implícitos ou explícitos a nível da tabela e são ignorados.
isc_tpb_concurrency + isc_tpb_nowait	Se um registo é modificado por alguma transacção que foi "committed" depois que a transacção que está a tentar adquirir o lock explicito começou, ou alguma transacção activa modificou o registo, uma excepção de conflito é imediatamente executada.

Modo TPB	*Comportamento*
isc_tpb_concurrency + isc_tpb_wait	Se um registo é modificado por alguma transacção que foi "committed" depois que a transacção que está a tentar adquirir o lock explicito começou, uma excepção de conflito é imediatamente executada.
	Se uma transacção activa é possuidora do registo (via um lock explicito ou por um lock normal optimista), a transacção que tenta o lock explicito espera pelo retorno desta e, quando termina, tenta de novo obter o lock do registo. Este facto indica que se a transacção bloqueadora tentar o "commit" de uma versão modificada do registo, uma excepção de conflito será criada.
isc_tpb_read_committed + isc_tpb_nowait	Se uma transacção activa é possuidora do registo (via um lock explicito ou por um lock normal optimista), uma excepção de conflito ocorre de imediato.
isc_tpb_read_committed + isc_tpb_wait	Se uma transacção activa é possuidora do registo (via um lock explicito ou por um lock normal optimista), a transacção que tenta o lock explicito espera pelo retorno desta e, quando termina, tenta de novo obter o lock do registo. Uma excepção de conflito nunca acontece neste modo de TPB.

Quando uma declaração UPDATE tenta aceder a um registo que está bloqueado por outra transacção, ou dispara uma excepção de conflito, ou espera que a transacção que possui o bloqueio termine, dependendo do modo TPB. O Comportamento do "engine" é o mesmo como se o registo já tivesse sido modificado pela transacção que estabeleceu o bloqueio.

O "engine" garante que todos os registos devolvidos por uma declaração de bloqueio específico estão efectivamente bloqueados e que efectivamente todos os parâmetros da cláusula "where" são satisfeitos, desde que a condição de busca não dependa de outras tabelas, via joins, subqueries, etc. Ainda garante que as linhas que não satisfazem a condição WHERE não serão bloqueadas pela transacção. Porém, não pode garantir que não existam linhas que, embora satisfaçam a cláusula "where", não estejam bloqueadas. Esta situação pode acontecer se alguma outra transacção paralela tiver elaborado um "commit" durante o curso de execução da declaração de bloqueio.

O "engine" bloqueia as linhas na altura do "fetch". Este facto tem bastante influência se se bloqueiam várias linhas de uma só vez. Muitos métodos de acesso ao Firebird utilizam processos para fazer "fetch" de apenas algumas linhas de cada vez ("buffered fetches"). Grande parte dos componentes de acesso não poderão devolver as linhas contidas no último pacote "fetched", onde o erro ocorreu.

A declaração FOR UPDATE estabelece uma técnica que previne a utilização de "buffered fetches", e ainda a opção OF <nome-das-colunas> para activar updates posicionais. Em alternativa, pode ser possível definir nos seus componentes de acesso o tamanho do buffer de "fetch" para 1. Isto irá permitir o processamento da linha, que está correntemente bloqueada, antes que a próxima linha seja "fetch" e bloqueada, ou ainda fazer um processamento de erros sem a necessidade de "rollback" da transacção.

O Rollback de um "savepoint" implícito ou explicito liberta os locks de registos que tenham sido adquiridos nesse savepoint, mas não notifica outras transacções que estejam a correr. As aplicações não devem tirar uso deste comportamento, pois é provável que seja modificado no futuro.

Embora os locks explícitos possam ser utilizado para prevenir ou processar erros de conflitos pouco comuns, o volume de erros de deadlock irá aumentar a menos que se tenha uma estratégia cuidadosa e que a controle rigorosamente. A maior parte das aplicações não irão necessitar de utilizar locks explícitos. O grande objectivo dos locks explicitos é (1) prevenir o processamento de um grande volume de erros de conflitos, com modificações, em aplicações "heavily loaded" e (2) manter a integridade de

objectos mapeados a base de dados relacional, em ambientes de clusters. Se pensar em utilizar locks explícitos fora destas duas categorias, então esta é a forma errada de executar essa tarefa no Firebird.

Os locks explícitos são uma "Funcionalidade Avançada", não a use inconscientemente! Embora estes casos sejam preciosos para web sites que lidam com milhares de Escritas em Simultâneo, ou para aplicações ERP/CRM utilizadas em grandes empresas, a maior parte das aplicações não necessitam de trabalhar nestas condições.

Exemplos

i) (simples)
```
  SELECT * FROM DOCUMENT WHERE ID=? WITH LOCK
```

ii) (múltiplas linhas, processamento com cursor DSQL um-por-um)
```
  SELECT * FROM DOCUMENT WHERE PARENT_ID=? FOR UPDATE WITH LOCK
```

(1.5) Melhorias no tratamento de agregados

Arno Brinkman

Originalmente, o agrupamento apenas era permitido em colunas nomeadas. No Firebird 1, já era permitido o agrupamento por uma expressão UDF. Na versão 1.5, várias extensões foram aplicadas para permitir a processamento de funções agregadas na cláusula GROUP BY, que permite agora o uso da posição da coluna no output (a posição base-1 da esquerda para a direita, como na cláusula ORDER BY), ou por uma variedade de expressões.
NOTA: Nem todas as expressões são permitidas dentro de uma declaração GROUP BY. Por exemplo, a concatenação não é permitida.

Sintaxe do Group By

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

lista_group_by: item_group_by [,lista_group_by];

item_group_by: nome_coluna
               | grau (ordinal)
               | udf
               | função_group_by;

função_group_by : função_valor_numérico
               | função_valor_texto
               | expressão_case
               ;

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

função_valor_texto      :  SUBSTRING '(' valor FROM posição_short_integer ')'
                        | SUBSTRING '(' valor FROM posição_short_integer FOR
longitude_short_integer ')'
                        | KW_UPPER '(' valor ')'
                        ;
```

O item_group_by não pode ser uma referência a uma função agregada (incluindo alguma que esteja localizada dentro da expressão) do mesmo contexto.

HAVING
A cláusula HAVING apenas permite funções agregadas ou expressões válidas que fazem parte da cláusula GROUP BY. Anteriormente era permitido utilizar colunas que não faziam parte da cláusula GROUP BY e utilizar expressões não-válidas.

ORDER BY
No contexto de uma declaração agregada, a cláusula ORDER BY permite apenas expressões válidas que sejam funções agregadas ou partes de expressões que estejam na cláusula GROUP BY. Antigamente, era permitido usar expressões não-válidas.

Funções Agregadas em "subqueries"
É agora possível utilizar em "subqueries" uma função agregada ou expressão contida numa cláusula GROUP BY.

Exemplos
```
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
```

Misturando funções agregadas de diferentes contextos
As funções agregadas de diferentes contextos podem ser utilizadas dentro de uma expressão.

Exemplo
```
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 query funciona em FB1.0, mas os resultados são errados!

Subqueries são suportadas numa Função Agregada
Utilizar um "singleton select" (que devolve apenas uma linha) numa função agregada é agora suportado.

Exemplo

```
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
```

Funções agregadas "Nested"
A utilização de uma função agregada dentro de outra função agregada (nested) é possível se a função agregada "inner" for de um contexto inferior (ver exemplo).

Grau de Group By (ordinal)
A utilização do número do grau da coluna de retorno na cláusula 'GROUP BY' "copia" a expressão da lista do select (à semelhança da cláusula ORDER BY). Isto faz com que, se o grau se refere a uma subquery, esta subquery irá correr pelo menos duas vezes.

(1.5) A Cláusula ORDER BY pode especificar expressões e localização dos NULL's

Nickolay Samofatov

A cláusula ORDER BY permite que se especifique qualquer expressão válida para ordenar os resultados da Query. Se a expressão consistir num número, é interpretada como o grau da coluna, como até aqui. A ordem dos NULL's no resultado pode ser controlada utilizando a cláusula de posicionamento dos nulls. Os resultados podem ser ordenados de forma que os NULL surgirão antes (NULLS FIRST) ou depois (NULLS LAST) dos resultados não-NULL. Quando nada é especificado, a posição dos NULLs é NULLS LAST.

Sintaxe

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

```
nulls_placement : NULLS FIRST | NULLS LAST;
```

Restrições

- Se for especificado NULLS FIRST, nenhum índice será utilizado na ordenação.
- Os resultados da ordenação de valores devolvidos por uma UDF ou Stored Procedure irão ser imprevisíveis se os valores devolvidos não poderem ser utilizados para determinar a sequência lógica de ordenação.
- O número de vezes que uma UDF ou Stored Procedure será chamada é indeterminado, independentemente da mesma ser chamada por uma expressão ou pela sua ordem de posicionamento.
- Uma cláusula de ordenação aplicada a uma query com union apenas pode utilizar o número de grau (posicionamento) para se referir às colunas.

Exemplos

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
```

PSQL (Linguagem de Stored procedures e triggers)

(1.5) EXECUTE STATEMENTg

Alex Peshkov

Extensão ao PSQL que processa uma string que contém uma declaração SQL válida e a executa como se esta tivesse sido submetida ao DSQL.
Disponível em triggers e stored procedures.

O Sintaxe pode assumir três formas.

Sintaxe 1
Executa <string> como uma operação SQL que não devolve dados, por exemplo INSERT, UPDATE, DELETE, EXECUTE PROCEDURE ou qualquer declaração DDL excepto CREATE/DROP DATABASE.

```
EXECUTE STATEMENT <string>;
```

Exemplo

```
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
```

Sintaxe 2

Executa <string> como uma operação SQL, devolvendo apenas uma linha. apenas um SELECT que devolva apenas uma única linha ("singleton") pode ser utilizado com esta sintaxe.

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

Exemplo

```
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
```

Sintaxe 3

Executa <string> como uma operação SQL, devolvendo várias linhas. Qualquer operador SELECT pode ser utilizado nesta forma.

```
FOR EXECUTE STATEMENT <string> INTO :var1, …, :varn DO
   <declaração composta>;
```

Exemplo

```
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 adicionais ao uso de EXECUTE STATEMENT

A string DSQL do 'EXECUTE STATEMENT' não pode conter nenhum parâmetro em qualquer uma das suas formas de sintaxe. Todas as substituições de variáveis pelo seu formato estático devem ser elaboradas antes de executar a declaração EXECUTE STATEMENT.

Esta funcionalidade requere um uso cauteloso, e deve ser utilizada tomando em conta todos os factores. Deve ser tomado como regra que se deve utilizar um EXECUTE STATEMENT apenas quando outros métodos não são possíveis, ou se estes têm ainda mais desvantagens que o EXECUTE STATMENT.

A declaração EXECUTE STATEMENT é potencialmente insegura nas seguintes condições:

1. Não existe forma de verificar a sintaxe correcta da declaração contida na string;
2. Não existem "Dependency checks" que possam descobrir quais as tabelas e colunas que foram destruídas;
3. As operações serão mais lentas porque a declaração embutida será preparada de cada vez que é executada.
4. Os valores devolvidos são verificados a nível do seu "data type", para que não existam excepções de 'type cast'. Por exemplo, a string '1234' será convertida para um inteiro, 1234, mas 'abc' irá gerar um erro de conversão.
5. Se a 'stored procedure' tem privilégios especiais em alguns objectos, a declaração dinâmica submetida na string do EXECUTE STATEMENT não os irá herdar. Os privilégios estão restritos aos atribuídos ao utilizador que manda executar a stored procedure.

(1.5) Novas variáveis de contexto

Dmitry Yemanov

CURRENT_CONNECTION

e

CURRENT_TRANSACTION

Cada uma destas novas variáveis de contexto devolve um identificador de sistema da conexão activa e da transacção corrente, respectivamente. O tipo devolvido é INTEGER. Disponível em PSQL e DSQL. Como estes valores são guardados no "header" da base de dados, os seus valores são objecto de um "reset" a cada "restore" da base de dados.

Sintaxe

```
CURRENT_CONNECTION
CURRENT_TRANSACTION
```

Exemplos

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

ROW_COUNT

Retorna um inteiro, com o número de linhas afectadas pela última declaração DML. Disponível em PSQL, no contexto de uma trigger ou stored procedure. Na sua forma corrente, devolve zero para uma declaração SELECT.

Sintaxe

```
ROW_COUNT
```

Exemplo

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

```
   INSERT INTO TABLE1 (ID, FIELD1) VALUES (:ID, 0);
```

Nota: esta variável não pode ser utilizada para verificar o número de linhas afectadas por uma declaração EXECUTE STATEMENT.

SQLCODE e GDSCODE

Cada variável de contexto devolve um número inteiro que é o código correspondente da excepção activa. Disponível em PSQL, dentro do contexto de um bloco de excepção. Fora do bloco, ambas devolvem zero.

A variável GDSCODE devolve uma representação numérica do código de erro GDS (ISC), ie, '335544349L' irá devolver 335544349.

Um bloco de excepção 'WHEN SQLCODE' ou 'WHEN ANY' irá obter um valor não-zero para a variável SQLCODE e devolver zero para GDSCODE. Apenas um bloco 'WHEN GDSCODE' irá obter um valor não-zero para a variável GDSCODE (e devolverá zero em SQLCODE). Se uma excepção criada pelo utilizador for gerada, as variáveis SQLCODE e GDSCODE serão ambas zero, independentemente do tipo de bloco de excepção.

Sintaxe

```
SQLCODE
GDSCODE
```

Exemplo

```
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
```

Veja ainda Melhorias no tratamento de excepções, em baixo, e o documento README.exception_handling em firebird2/doc/sql.extensions da árvore CVS do Firebird.

INSERTING

UPDATING

DELETING

Estas três expressões pseudo-Booleanas podem ser utilizadas para testar ou determinar o tipo de operação DML que está a ser executado. Disponível em PSQL, apenas em triggers. Criadas para serem utilizadas com triggers universais (ver METADATA, abaixo).

Sintaxe

```
INSERTING
```

```
UPDATING
DELETING
```

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

(1.5) Melhorias no Tratamento de Excepções em PSQL

Dmitry Yemanov

A Sintaxe comum para uma declaração EXCEPTION em PSQL é:

```
  EXCEPTION [nome | [valor]];
```

As melhorias na versão 1.5 permitem:

1) Definir uma mensagem em "run-time" para uma excepção;
2) Re-iniciar ("re-raise") uma excepção dentro do contexto do bloco da excepção
3) Obter um erro numérico da excepção despoletada

1) Mensagens de Excepção em "Run-time"

Sintaxe
```
EXCEPTION <nome_excepção> <valor_mensagem>;
```

Exemplos
i)
```
EXCEPTION E_EXCEPTION_1 'Erro!';
```
ii)
```
EXCEPTION E_EXCEPTION_2 'Tipo errado para o registo com o ID=' || new.ID;
```

2) "Re-raising" de uma excepção

Nota – não tem qualquer efeito fora de um bloco de excepção.

Sintaxe
```
EXCEPTION;
```

Exemplos
i)
```
BEGIN
  ...
  WHEN SQLCODE -802 DO
    EXCEPTION E_ARITH_EXCEPT;
  WHEN SQLCODE -802 DO
    EXCEPTION E_KEY_VIOLATION;
  WHEN ANY THEN
    EXCEPTION;
END
```
ii)
```
WHEN ANY DO
BEGIN
   INSERT INTO ERROR_LOG (...) VALUES (SQLCODE, ...);
   EXCEPTION;
END
```

3) Códigos de Erro em "Run-time"
Ver SQLCODE / GDSCODE (acima).

(1.5) Declarações LEAVE | BREAK

Termina o processamento de um loop, fazendo o processamento deslocar-se para a declaração que segue o END que termina o loop. Disponível apenas para WHILE, FOR SELECT e FOR EXECUTE, de outra forma um erro de parse será gerado. O uso do standard SQL-99 LEAVE irá tornar obsoleto o BREAK existente. Disponível em triggers e stored procedures.

Syntaxe

```
LEAVE;
```

Exemplos

```
(i)
BEGIN
   <declarações>;
   IF (<condições >) THEN
     LEAVE;
   <declarações>;
END
(ii)
WHILE (<declarações>) DO
  BEGIN
    <declarações>;
    WHEN ... DO
      LEAVE;
  END
```

NOTA: LEAVE | BREAK e EXIT podem agora ser utilizados em triggers

(1.5) Declarações válidas de PLAN podem ser incluídas em Triggers

Ignacio J. Ortega

Até agora, um trigger que continha uma declaração PLAN poderia ser rejeitado pelo compilador. Agora, uma declaração válida de PLAN pode ser incluída e utilizada.

(1.5) Blocos BEGIN..END vazios

Dmitry Yemanov

Blocos BEGIN..END vazios são agora válidos em módulos PSQL. Por exemplo, pode-se agora escrever módulos "stub" como:

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

(1.5) Declarar e definir variáveis locais numa única declaração

Claudio Valderrama

Simplificação da sintaxe e extensão da operacionalidade para que as variáveis locais possam ser declaradas e inicializadas com uma única declaração.

Sintaxe
```
DECLARE [VARIABLE] nome <tipo_variável> [{'=' | DEFAULT} valor];
```

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

(1.0) SELECT [FIRST (<integer expr m>)] [SKIP (<integer expr n>)]

(1.5) SELECT FIRST pode agora ter zero como argumento

O FB 1.5 permite zero como argumento de FIRST. Será devolvido um resultado vazio.

Devolve as primeiras m linhas do resultado da selecção. A cláusula opcional SKIP faz com que as primeiras n linhas sejam ignoradas, devolvendo as m linhas, do resultado da selecção, começando com a n + 1. Na forma mais simples, m e n são inteiros mas também serve qualquer expressão Firebird que devolva um inteiro. Um identificador que devolva um inteiro pode também ser usado no GDML, apesar de não fazer parte do SQL ou DSQL.

O parêntesis é obrigatório para os argumentos das expressões e opcionais nas outras situações.

Podem também ser variáveis cegas, i.e. SKIP ? * FROM ATABLE devolve os restantes dados após ignorar as n linhas iniciais, onde n é passado na variável "?". SELECT FIRST ? COLUMNA, COLUMNB FROM ATABLE devolve as primeiras n linhas e ignora as restantes.

A cláusula FIRST também é opcional, i.e. é possível incluir SKIP numa declaração sem o FIRST para obter um conjunto de dados que exclua as linhas apontadas por SKIP.

Disponível em SQL e DSQL excepto onde for indicado o contrário.

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

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

SELECT FIRST 5 DISTINCT FIELD FROM MYTABLE;
```

Dois Inconvenientes com o SELECT FIRST

1. Isto

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

irá eliminar todas as linhas da tabela. Ouch! O “sub-select” avalia as primeiras 10 linhas candidatas para apagar, apaga-as, avalia as próximas 10, apaga-as, e assim sucessivamente até que não existam mais linhas disponíveis na tabela.

Cuidado!

2. Queries como este:

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

não funcionam como esperado, por causa da optimização processada pelo servidor ao transformar os predicados correlacionados WHERE...IN (SELECT...) em predicados correlacionados EXISTS. É óbvio que neste caso FIRST N não faz qualquer sentido:

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

Melhorias em "Character set"

Adicionadas na versão 1.5

- Adicionada a "collation" WIN1251_UA (para as linguagens russa e ucraniana) ao "character set" WIN1251.
- Corrigido o valor por defeito para Maiú/Minúsculas de WIN1251
- Adicionada a ISO_HUN (para a linguagem Húngara) ao "character set" ISO8859_2.

Novos "character sets" ("no non-binary collations") adicionados
Blas Rodriguez Somoza

- DOS737 PC Greek
- DOS775 PC Baltic
- DOS858 Variante do Cp850 com o caracter do Euro (€)
- DOS862 PC Hebrew
- DOS864 PC Arabic
- DOS866 MS-DOS Russian
- DOS869 IBM Modern Greek
- WIN1255 Windows Hebrew
- WIN1256 Windows Arabic
- WIN1257 Windows Baltic
- ISO8859_3 Latin 3 (Esperanto, Maltese, Pinyi, Sami, Croatian e outras)
- ISO8859_4 Latin 4 (Baltic, Greenlandic, Lappish)
- ISO8859_5 Cyrillic
- ISO8859_6 Arabic
- ISO8859_7 Greek
- ISO8859_8 Hebrew
- ISO8859_9 Turkish
- ISO8859_13 Baltic

Adicionadas na versão 1.0

- Foi adicionado o "collation set" Húngaro, insensível a maiúsculas, desenvolvido e testado por Sandor Szollosi (ssani@freemail.hu).

- O Firebird suporta agora o character set ISO8859-2 (para a linguagem Checa)

EXTENSÔES DE LINGUAGEM TRANSPORTADAS DO FIREBIRD 1.0.x

As seguintes extensões de linguagem, introduzidas no Firebird 1.0.x, são reproduzidas aqui.

(1.0) CURRENT_USER e CURRENT_ROLE

Estas duas novas variáveis de contexto foram adicionadas para referenciar o USER e (se implementado[1]) o ROLE do contexto corrente da conexão.

```
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 é um sinónimo DSQL de USER que aparece no SQL standard. Eles são idênticos. Não existe qualquer vantagem em usar um ou outro.

[1] Se insistir em usar uma base de dados InterBase v.4.x ou 5.1 com o Firebird, ROLE não é suportado, como tal current_role será NONE (tal como definido pelo standard SQL na ausência de um role explicito) mesmo que o utilizador passe um nome de role. Se está usando IB 5.5, IB 6 ou Firebird, o ROLE passado é verificado. Se o role não existir, é reposto com o valor NONE sem devolver qualquer erro.

Isto significa que no FB não é possível obter um ROLE inválido devolvido pelo CURRENT_ROLE, porque seria atribuído a NONE. Este comportamento contrasta com o IB, onde o valor é transportado internamente, sem que seja visível para o SQL.

(1.0) DROP GENERATOR

Permite que generators não usados sejam removidos da base de dados. O armazenamento será libertado para reutilização no RESTORE seguinte. Acessível em SQL e DSQL.

```
DROP GENERATOR <nome generator>;
```

(1.0) GROUP BY UDF

É agora possível agregar um "SELECT by grouping" à saída de uma UDF.
P.ex.

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

Um efeito colateral do agrupamento por UDFs é que, onde antigamente não era permitido a utilização de funções internas do Firebird, agora tal é permitido pela utilização de uma "dummy" UDF:

```
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),1)
```

(1.0) RECREATE PROCEDURE

Esta nova declaração DDL permite criar uma "Stored Procedure" com o mesmo nome de uma já existente, substituindo a anterior, sem necessidade de fazer primeiro "drop" à antiga. A sintaxe é idêntica a CREATE PROCEDURE.

Disponível em SQL e DSQL.

(1.0) RECREATE TABLE

Esta nova declaração DDL permite criar uma nova estrutura para uma tabela existente, sem necessidade de fazer primeiro "drop" à tabela antiga. A sintaxe é idêntica a CREATE TABLE.

Note que RECREATE TABLE **não preserva** os dados da tabela antiga.

Disponível em SQL e DSQL.

(1.0) SUBSTRING (*<string expr>* FROM *<pos>* [FOR *<length>*])

Função interna que implementa a função ANSI SQL SUBSTRING(). Irá devolver uma "string" que consiste no byte na posição <*pos*> e todos os bytes subsequentes até ao final da "string". Se a opção FOR <*length*> estiver especificada, irá devolver o menor destes dois valores: o tamanho <length> especificado, ou o número de bytes até ao final da "string".

O primeiro argumento poderá ser uma expressão qualquer, constante ou identificador, que possa ser avaliada para uma "string".
<*pos*> necessita de ser avaliado para um inteiro.
<*pos*> começa em 1, como outras declarações SQL.
Nem <*pos*> nem <*length*> podem ser parâmetros de Querys.

Dado que <pos> e <length> são posições de byte, o identificador pode ser um blob binário, ou um blob "sub_type 1 text blob" com um charset de um-byte-por-caracter. A função não admite de momento blobs de texto com character sets Chinese (2 byte/char máximo) ou Unicode (3 byte/char máximo). Para um argumento do tipo "string" (ao contrário de um blob), a função trabalha com qualquer charset.

Disponível em SQL e DSQL.

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

Consulte também a secção de "Funções Externas" (UDFs) e seguinte, para detalhes de alterações e adições à biblioteca standard de UDF.

(1.5) Aperfeiçoamento do Marcador de Comentário por Linha

Dmitry Yemanov

Os comentários de uma linha podem ser colocados em qualquer posição na linha, não apenas na primeira.
Assim, na versão 1.5, o marcador "--" pode ser utilizado para comentar a linha no final de uma declaração num script, stored procedure, trigger ou declaração DSQL. Pode ainda ser utilizado para "remover passando a comentário" partes de uma declaração. Todos os caracteres desde o marcador '— ' até ao fim da linha (CR ou LF) serão ignorados.

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

(1.0) Novo marcador de Comentários

Claudio Valderrama

Para ser utilizado em "scripts", DSQL, "stored procedures" e "triggers".

Exemplo

```
-- Este é um comentário
```

Este novo marcador pode ser utilizado para comentar uma simples linha de código num "script", numa declaração DDL/DML, "stored procedure" ou "trigger".

A lógica é ignorar todos os caracteres que o sigam.

1. Saltar os caracteres "--" se este símbolo for o primeiro par de caracteres depois de uma marca de fim-de-linha "EOL" (LF em Linux/Unix, CRLF em Windows)
2. Continuar a ignorar os caracteres até à próxima marca de fim-de-linha (EOL)

Esta lógica não é feita para se misturar com um bloco de comentário lógico (/* um comentário */). Por outras palavras, não utilize os marcadores "--" dentro de um bloco de comentário, e não utilize um bloco de comentário numa linha que utiliza os marcadores "--".

SESSÕES ISQL INTERACTIVAS: Tenha atenção quando trabalhar numa sessão ISQL interactiva. O ISQL aceita partes de uma declaração em segmentos contínuos, mostrando a "prompt" CON> até que receba o símbolo de terminação (normalmente ;). Se utilizar "--" no início de uma linha de continuação, a lógica de ignorar irá continuar até encontrar a marcação de fim-de-linha (EOL) que está impressa no visor ou no seu ficheiro de "output" quando premir ENTER. Existe uma fonte potencial de erros se continuar a adicionar uma continuação, esperando que esta seja ignorada.

O problema com o isql surge devido aos seus comandos especiais que deverão ser objecto de um "parse" apenas pelo ISQL. Se não são reconhecidos devido a uma colocação inesperada de "--", então são enviados ao "engine". Obviamente, o "engine" não compreende os comandos de isql SET e SHOW e rejeita-os.

(1.0) Alter Trigger deixou de incrementar o "Contador de Alterações" na Tabela

Quando o contador das alterações de "metadata" alcança o valor máximo de 255, a base de dados fica indisponível. Uma operação de "Backup/Restore" é necessária para elaborar um "reset" do contador e tornar a base de dados disponível de novo. A intenção é forçar a uma limpeza da base de dados (cleanup) quando a estrutura das tabelas já sofreram bastantes alterações, e não para inibir as capacidades do "engine".

Antigamente, cada vez que um "trigger" fosse colocado ACTIVE|INACTIVE por uma operação de ALTER TRIGGER, o Contador de Alterações da tabela associada era incrementado. Em consequência, o uso destas funções em operações regulares não era possível, pois o número atingia o seu máximo rapidamente.

Novas Palavras Reservadas

As seguintes palavras do Firebird devem ser adicionadas à lista de Palavras Reservadas publicadas para o 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		

As seguintes palavras estão reservadas para o futuro:

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

As seguintes palavras estavam reservadas no Firebird 1.0, mas já não o são no Firebird 1.5:

BREAK	DESCRIPTOR	FIRST
IIF	SKIP	SUBSTRING

As seguintes palavras não-reservadas são reconhecidas pelo Firebird 1.5, se utilizadas nos seus respectivos contextos estruturais:

COALESCE	DELETING	INSERTING
LAST	LEAVE	LOCK
NULLIF	NULLS	STATEMENT
UPDATING	USING	

Estas palavras novas no **InterBase 6.5 e 7** (não reservadas no Firebird) devem também ser tratadas como reservadas, para manter a compatibilidade:

BOOLEAN	FALSE	GLOBAL
PERCENT	PRESERVE	ROWS
TEMPORARY	TIES	TRUE

Funcionalidades no ISQL

Capacidade de "readline" na shell do isql

Mark O'Donohue

Suporte a "command history" (como o readline do Unix) foi adicionado ao isql shell. Agora pode utilizar as teclas Up e Down para navegar nos comandos submetidos na sessão isql.

Funções Definidas pelo Utilizador (UDF's)

No ib_udf

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

"Right-pad" da string fornecida, por adição do caracter *padcharacter* até a string resultante ter o tamanho *length*. A string de entrada pode ter um tamanho até 32766 bytes. "Length" não pode exceder 32765 bytes.

Declaração
```
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

"Left-pad" da string fornecida, por adição do caracter *padcharacter* no seu início, até a string resultante ter o tamanho *length*. A string de input pode ter um tamanho até 32766 bytes. "Length" não pode exceder 32765 bytes.

Declaração
```
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 função tinha um erro muito antigo, em que os argumentos x e y estavam invertidos. Deveria devolver o logaritmo base x de y, mas de facto devolvia o log base y de x. Foi corrigido.
Se usava a função previamente nas suas aplicações, VERIFIQUE O SEU CODIGO! Ou estava a devolver os resultados incorrectos, ou alguém reverteu os valores deliberadamente de forma a obter o resultado correcto.

Em fb_udf

1. As funções *NVL e *NULLIF foram mantidas para compatibilidade com as versões anteriores, mas estão obsoletas pela introdução das novas funções internas CASE, COALESCE e NULLIF.
2. Deverá ser levado em linha de conta que a fbudf não pode processar campos string com um tamanho maior do que 32Kb - 1 bytes. Esta limitação poderá ter efeitos colaterais onde as strings sejam concatenadas antes de serem passadas como argumentos às UDFs. Se a soma dos campos estiver além dos limites, o comportamento será indefinido. A função poderá devolver resultados sem sentido ou o código do fbudf poderá executar uma operação ilegal.
3. Se a base de dados com a qual pretende evoluir foi criada pelo Firebird 1.0.x e possui declarações das funções fbudf **truncate** e **round**, essas declarações não irão mais funcionar com o Firebird 1.5 porque os nomes "entry_point" foram modificados. Terá que eliminar essas funções e redeclará-las, usando as declarações do script fbudf.sql na directoria UDF do 1.5.

Novo Ficheiro de Configuração – firebird.conf

A Directoria Raiz do Firebird

A directoria raiz da sua instalação de Firebird é utilizada de várias formas, não apenas na instalação, mas como atributo que as rotinas do servidor, parâmetros de configuração e clientes dependem. Como existem várias formas de indicar ao servidor onde encontrar o valor para este atributo, programadores e administradores de sistema devem ter a noção sobre a procedência que o servidor toma no arranque, para a determinar correctamente.

Win32 Superserver e Classic (tanto o servidor como o cliente):
1) Variável do "environment" FIREBIRD
2) Parâmetro RootDirectory no firebird.conf
3) Registo:
   HKLM\SOFTWARE\ SOFTWARE\Firebird Project\Firebird Server\Instances\DefaultInstance
   onde procura o Campo RootDirectory.
4) A directoria um nível acima aquela de onde o binário do servidor está localizado.

Win32 Embedded (Servidor Embutido):
1) Variável do "environment" FIREBIRD
2) Parâmetro RootDirectory no firebird.conf
3) A directoria onde fbembed.dll (renomeado fbclient.dll) está localizado

Linux Classic:
1) Variável do "environment" FIREBIRD
2) Parâmetro RootDirectory no firebird.conf
3) Path por defeito da instalação (/usr/local/firebird)

Linux Superserver:
1) Variável do "environment" FIREBIRD
2) Parâmetro RootDirectory no firebird.conf
3) A directoria um nível acima daquela onde o binário do servidor está localizado (obtido via symlink "/proc/self/exe", se suportado)
4) Path por defeito da instalação (/usr/local/firebird)

Parâmetros

Parâmetros por defeito ("Default") são aplicáveis a quase todos os parâmetros. Os nomes dos parâmetros e os seus valores são "case-sensitive" no Linux mas não em Windows. Para definir algum parâmetro para um valor "não-default", elimine o marcador de comentário (#) e edite o valor. Pode editar o ficheiro de configuração enquanto o servidor está activo. Para activar as alterações, é necessário parar e reiniciar o serviço.

As entradas são do tipo:

Nome_do_parâmetro *valor*

- Nome_do_parâmetro é uma string que não contém quaisquer espaços em branco e dá nome a uma propriedade do servidor a ser configurada.

- *valor* é um número, Booleano (1=Verdadeiro, 0=Falso) ou uma string que especifica o valor do parâmetro.

Parâmetros do Sistema de Ficheiros

RootDirectory

Uma String, a path absoluta em relação à directoria raiz do sistema de ficheiros local. Deve permanecer comentada, a menos que se pretenda forçar o procedimento de arranque para definir uma directoria para a instalação do servidor Firebird, que caso contrário será automaticamente detectada.

DatabaseAccess

Suporta a nova funcionalidade de Aliases. Nas versões anteriores, o servidor podia conectar-se a qualquer base de dados no seu sistema de ficheiros local, e era acedido pelos clientes que usavam a path absoluta para o ficheiro da base de dados. Este parâmetro implementa a opção de restringir o acesso ao servidor apenas através do seu Alias, ou apenas a bases de dados localizadas em determinadas árvores de directorias do sistema de ficheiros.
DatabaseAccess pode ser None, Restrict ou Full.
Full (o valor por defeito) permite que os ficheiros de bases de dados sejam acedidos em qualquer local do sistema de ficheiros local.
None permite que o servidor só utilize bases de dados que se encontrem definidas em **aliases.conf**.
Restrict permite a configuração das localizações de directorias que o servidor poderá utilizar para abrir ficheiros de bases de dados. Pode-se definir uma ou mais directorias, separadas por 'ponto e vírgula', para definir um ou mais locais permitidos.
Por exemplo,
Unix: `/db/databases;/userdir/data`
Windows: `D:\data`
Paths relativos são tratados como relativos ao path que o servidor reconhece como sendo a sua directoria raiz. Por exemplo, no Windows, se a raiz do Firebird é a directoria C:\Programas\Firebird, então o valor seguinte irá restringir o servidor a aceder a ficheiros que estejam localizados apenas em C:\Programas\Firebird\userdata:

```
DatabaseAccess = Restrict userdata
```

NOTA: "shadow" da base de dados – o tratamento actual do DatabaseAccess possui um erro, o que motiva o uso da opção Restrict se está a executar o "shadow" de alguma base de dados no servidor.

ExternalFileAccess

Foi *external_file_directory* em isc_config/ibconfig mas a sintaxe mudou.

Providencia três níveis de segurança no que respeita a EXTERNAL FILES (ficheiros de texto de formato fixo que são acedidos como tabelas da base de dados). O valor é uma string, mas pode ser None, Full ou Restrict.
None (o valor por defeito) desactiva o uso de EXTERNAL FILES no servidor;
Restrict permite restringir a localização das EXTERNAL FILES a directorias específicas. Deverá indicar uma directoria, ou uma lista de directorias separadas por ´ponto e vírgula', onde as EXTERNAL TABLES possam estar localizadas.
Por exemplo,
Unix: `/db/extern;/mnt/extern`
Windows: `C:\ExternalTables`
Paths relativos são tratados como relativos ao path que o servidor reconhece como sendo a sua directoria raiz. Por exemplo, no Windows, se a raiz do Firebird é a directoria C:\Programas\Firebird, então o valor seguinte irá restringir o servidor a aceder a EXTERNAL FILES que estejam localizadas apenas em C:\Programas\Firebird\ExternalTables:

```
ExternalFileAccess = Restrict userdata\ExternalTables
```

Full permite que os EXTERNAL FILES sejam acedidas em qualquer local do sistema.

Veja o **Aviso de Precaução** depois da próxima entrada, UdfAccess.

UdfAccess

Foi *external_function_directory* em isc_config/ibconfig mas a sintaxe mudou.

Substitui não só o nome dos parâmetros, mas também a forma como os valores são apresentados. O propósito das alterações foi activar opcionalmente níveis de protecção para módulos de biblioteca externos definidos pelo utilizador, um alvo reconhecido para ataques maliciosos. UdfAccess pode ser None, Restrict ou Full.

Restrict (o valor por defeito) mantêm a funcionalidade que era disponibilizada pelo parâmetro **external_function_directory** no Firebird 1.0, para restringir a localização das bibliotecas externas para localizações específicas do sistema de ficheiros. Defina uma ou mais directorias, separadas por 'ponto-e-vírgula', onde as bibliotecas de UDF, Filtros BLOB e definições de character set possam estar localizadas.
Por exemplo,
Unix: `/db/extern;/mnt/extern`
Windows: `C:\ExternalModules`
Paths relativos são tratados como relativos ao path que o servidor reconhece como sendo a sua directoria raiz. Por exemplo, no Windows, se a raiz do Firebird é a directoria C:\Programas\Firebird, então o valor seguinte irá restringir o servidor a aceder a EXTERNAL FILES que estejam localizadas apenas em C:\Programas\Firebird\ExternalModules:

```
ExternalFileAccess = Restrict userdata\ExternalModules
```

None desabilita o uso de bibliotecas externas.

Full permite que as bibliotecas externas possam ser acedidas em qualquer local do sistema.

Aviso de Precaução :: Evite definir directorias comuns para UdfAccess e ExternalFileAccess, de forma que as mesmas possuam uma mesma árvore de raiz. Os parâmetros por defeito são seguros. Se tiver a personalizar as mesmas, e não estabelecer directorias separadas para ambas, o servidor pode facilmente ser 'hackado' para executar código não autorizado. Um exemplo a evitar:

```
UdfAccess = UDF; /bad_dir
ExternalFileAccess = /external; /bad_dir/files
```

UdfAccess e ExternalFileAccess estão numa sub-arvore comum, `/bad_dir/files`, onde qualquer um poderá criar a sua EXTERNAL FILE `/bad_dir/files/hackudf.so` e executar o seu código no sistema comprometido.

Parâmetros dos Recursos

CpuAffinityMask

Foi *cpu_affinity* em isc_config/ibconfig

Com a versão SuperServer do Firebird em Windows, e em máquinas SMP, existe um problema que faz com que o sistema operativo faça um swap constante do processo entre processadores. Este facto arruína a performance. Este parâmetro pode ser utilizado em sistemas SMP em Windows para fazer o processo SuperServer do Firebird correr em apenas um CPU.

CpuAffinityMask é um inteiro, a CPU mask.

Exemplo

`CpuAffinityMask = 1`
Corre apenas no primeiro CPU (CPU 0).

`CpuAffinityMask = 2`
Corre apenas no segundo CPU (CPU 1).

`CpuAffinityMask = 3`
Corre no primeiro e segundo CPU.

Calculando o valor da "affinity mask"
Você pode usar esta opção para modificar o affinity do Firebird para qualquer processador singular ou (no servidor Classic) para qualquer combinação de CPUs instalada no sistema.
Considere os CPUs como uma matriz numerada de 0 a *n*-1, onde *n* é o número de processadores instalados e *i* é o número da matriz de um CPU. M é outra matriz, contendo a "MaskValue" de cada CPU seleccionado. O valor de A é a soma dos valores de M.
Use a fórmula seguinte para chegar a M e calcular o MaskValue A:

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

Por exemplo, para seleccionar o primeiro e quarto processador (processador 0 e processador 3) calcule da seguinte forma:

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

> ATENÇÃO: servidores Firebird, até à Release 1.5 (inclusive), não suportam a funcionalidade Hyperthreading presente nalgumas motherboards recentes. Para evitar problemas de balanceamento, poderá ter que desactivar o hyperthreading ao nível da BIOS do sistema.

DeadlockTimeout

Era *deadlock_timeout* em isc_config/ibconfig

Número de segundos (inteiro) que o lock manager irá aguardar se um conflito acontecer, antes de purgar os locks de processos "dead" e elaborar um novo ciclo de "deadlock". Normalmente, o "engine" detecta os deadlocks instantaneamente. Este timeout apenas surge quando algo não corre bem.

O "default" de 10 segundos é bom para quase todas as condições. Baixando este valor não vai necessariamente reduzir o tempo em que um deadlock confirma uma excepção de conflito. Se é baixo demais, o efeito pode levar a "scans" extras desnecessários que podem degradar o desempenho do sistema.

DefaultDbCachePages

Era *database_cache_pages* em isc_config/ibconfig

Número de datapages alocadas em memória pelo servidor, por base de dados. O valor configurado pode ser redefinido a nível da base de dados.
O valor por defeito para o SuperServer é de 2048 páginas, para o Classic é de 75.
A cache é utilizada de forma diferente pela versão SuperServer e pela Classic. No SS a cache é utilizada por todas as conexões, a Classic aloca uma cache estática para cada conexão.

EventMemSize

Inteiro, representa o número de bytes de memória reservado para o "event manager". O valor por defeito é de 65536 (64 Kb).

LockAcquireSpins

Era *lock_acquire_spins*

Relevante apenas em equipamentos SMP a correr o servidor Classic. Num servidor Classic, apenas um processo cliente pode aceder à tabela de locks de cada vez. Este processo é controlado por um mutex. Os processos clientes podem requerer o mutex condicionalmente ou incondicionalmente. Se é condicional, o "request" falha e deve ser repetido. Se é incondicional, o "request" espera até ser satisfeito. O parâmetro LockAcquireSpins" estabelece o número de tentativas que serão efectuadas se o mutex for condicional.
Inteiro. O "default" é 0 (incondicional). Não existe qualquer recomendação de valor mínimo e máximo.

LockHashSlots

Era *lock_hash_slots* em isc_config/ibconfig

Utilize este parâmetro para ajustar a lista de "lock hash". Em grande carga, o "throughput" (caudal de transferência) pode ser melhorado aumentando o número de "hash slots" para dispersar a lista em cadeias de "hash" mais pequenas. Um número primo é recomendado. O valor por defeito é 101.

LockGrantOrder

Era *lock_grant_order* em isc_config/ibconfig

Quando uma conexão pretende obter o lock de um objecto, obtém um "lock request block" que especifica o objecto e o nível de lock pretendido. Os blocos requeridos são conectados a estes blocos de locks como um "request" que foi atendido, ou como um "request" pendente.
O parâmetro LockGrantOrder é um Booleano. O default (1=True) indica que os locks devem ser requeridos numa base "first-come-first-served" (primeiro a chegar, primeiro a ser servido). Definir como falso (0), emulando o comportamento do InterBase v3.3, assegura o lock, logo que este esteja disponível. Pode resultar em "requests" que ficam "encravados".

LockMemSize

Este inteiro representa o número de bytes alocados na memória partilhada para o gestor de locks. Para um servidor Classic, LockMemSize indica o valor inicial alocado, que irá crescer dinamicamente até que a memória fique exausta (*"Lock manager is out of room").* Se existirem muitas conexões ou caches de páginas muito grandes, aumente este valor, para evitar estes erros.

No SuperServer, a memória alocada pelo gestor não aumenta dinamicamente.
O valor por default em Linux e Solaris é de 98304 bytes (96 Kb). Em Windows, é de 262144 (256 Kb).

LockSemCount

Parâmetro inteiro que especifica o número de semáforos disponíveis para a comunicação inter processos (IPC). O valor por defeito é 32. Use este parâmetro em ambientes que não utilizam "threads" para aumentar ou reduzir o número de semáforos disponíveis.

SortMemBlockSize

Este parâmetro permite a configuração, em bytes, do tamanho de cada bloco de memória utilizado pelas rotinas de ordenação em memória. O valor por defeito é de 1Mb, mas pode ser configurado para qualquer tamanho até ao valor máximo estabelecido pelo parâmetro SortMemUpperLimit (ver abaixo).

SortMemUpperLimit

O valor máximo de memória, em bytes, que pode ser alocado pelas rotinas de ordenação em memória. O valor por defeito é de 67108864 bytes (64 Mb) no SuperServer e 8388608 (8 Mb) em Classic.

> CUIDADO: Em Classic, tenha em atenção que aumentar o SortMemBlockSize ou o SortMemUpperLimit afecta cada conexão/instância do servidor e irá aumentar drasticamente a memória utilizada pelo servidor.

Parâmetros de Comunicação

ConnectionTimeout

Era *connection_timeout* em isc_config/ibconfig

Número de segundos de espera antes de abandonar uma tentativa de conexão. O valor por defeito é 180.

DummyPacketInterval

Era *dummy_packet_interval* em isc_config/ibconfig

Número de segundos (inteiro) que o servidor irá aguardar antes de enviar um "dummy packet" para reconhecer a conexão.
NÃO USE ESTA OPÇÃO num servidor Win32 com clientes TCP/IP. Causa um aumento de utilização da memória não paginada que pode fazer o SO crashar no lado do cliente como explicado aqui:
http://support.microsoft.com/default.aspx?kbid=296265

Problemas de Win32-com-TCP/IP à parte, esta é a única forma de detectar e desconectar clientes inactivos quando NamedPipes (NetBEUI), XNET ou IPC são utilizados. Não existem nenhumas restrições conhecidas em sistemas POSIX.

Normalmente, o Firebird utiliza a opção do socket SO_KEEPALIVE para controlar as conexões activas. Se não pretender ter um timeout de 2 horas (valor por defeito) ajuste os parâmetros do seu sistema operativo de acordo com:

- Em SO's do tipo UNIX, modifique o conteúdo de /proc/sys/net/ipv4/tcp_keepalive_*.
- Em Windows, siga as instruções deste artigo:

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

Default deve ser 0 – não 60 que era o default no Firebird 1.0 e na maior parte das RC do 1.5. Uma definição de 60 deve ser assumida como o valor por defeito em sistemas em que se pretende fazer uso deste "dummy packet polling".

RemoteServiceName

Valor por Defeito = gds_db

RemoteServicePort

Estes dois parâmetros permitem alterar o nome do serviço TCP/IP ou a porta TCP/IP utilizada para a conexão dos clientes, se algum deles for diferente dos instalados por defeito (gds_db/tcp 3050).

Mude um destes parâmetros, não os dois. Primeiro é verificada a presença de RemoteServiceName no ficheiro de serviços. Se existir, a porta aí especificada é utilizada. Se não existir, então usa a porta por defeito, 3050.

NOTA: Se uma porta de conexão for indicada na string de conexão TCP/IP, esta irá ter precedência sobre RemoteServicePort.

RemoteAuxPort

O comportamento herdado do InterBase, de passar mensagens de notificação de eventos de volta à rede através de portas TCP/IP seleccionadas de forma aleatória, tem sido uma fonte persistente de erros de rede e conflitos com firewalls, por vezes ao ponto de causar estouros do servidor em certas condições. Este parâmetro permite configurar uma única porta TCP para o tráfego de todas as notificações de eventos.
O valor de instalação por defeito (0) conserva o tradicional comportamento aleatório das portas. Para dedicar uma porta específica para a notificação de eventos, use um número inteiro correspondente a uma porta disponível.

RemoteBindAddress

Por defeito, os clientes podem conectar-se a qualquer interface de rede através da qual o alojador do servidor aceita o tráfego de rede. Este parâmetro permite ligar o serviço Firebird a pedidos recebidos através de um NIC e rejeitar conexões de quaisquer outras interfaces de rede. Isto deve ajudar a ultrapassar problemas nalgumas sub-redes onde o servidor tenha que lidar com o tráfego através de múltiplos NICs.
String, num formato válido IP. Valor por defeito (sem limitação) é o valor nulo.

TcpRemoteBufferSize

As leituras em antecipação pelo "engine" permitem enviar várias linhas de dados num único pacote. Quanto maior o tamanho do pacote, mais linhas serão enviadas por cada transferência. Use este parâmetro – com cuidado e compreensão profunda dos efeitos no desempenho da rede! - se precisar aumente ou reduza o tamanho do pacote TCP/IP usado para enviar e receber buffers. Afecta tanto o cliente como o servidor.
O valor é um inteiro (tamanho do pacote em bytes) dentro dos limites de 1448 até 32768. O valor por defeito da instalação é de 8192.

Parâmetros específicos POSIX

LockSignal

Parâmetro inteiro, sinal do UNIX para usar em comunicações inter-processos. Valor por defeito: 16

RemoteFileOpenAbility

USAR APENAS COM EXTREMA CAUTELA

Parâmetro Booleano o qual, se colocado a True, permite que o "engine" abra ficheiros de bases de dados os quais residam em partições montadas em sistema de ficheiros de rede (NFS). Porque o sistema de ficheiros se encontra fora do controlo do sistema local, isto é uma funcionalidade muito arriscada que não deve ser activada para propósitos de ler/escrever bases de dados de cuja "sobrevivência" você dependa.
O valor por defeito é 0 (Falso, desactivado) e deve mantê-lo assim a menos que esteja bem ciente dos seus efeitos.

TcpNoNagle

Era *tcp_no_nagle* em isc_config/ibconfig

No Linux, por defeito, a biblioteca de "socket" minimizará as escritas, guardando as escritas antes do envio dos dados, usando um algoritmo interno (implementado como opção TCP_NODELAY da conexão

"socket") conhecido como o Algoritmo de Nagle. Foi desenhado para evitar problemas, em redes lentas, com pacotes pequenos, chamados "tinygrams".
Por defeito, TCP_NODELAY é activado (valor 0) quando o Firebird Superserver é instalado no Linux. Em redes lentas, a sua desactivação poderá aumentar a velocidade. Tenha em atenção ao antagonismo—colocar o parâmetro a True para desactivar o TCP_NODELAY e colocá-lo a Falso para activá-lo.
Em versões até à v.1.5 inclusive, esta funcionalidade só está activa para o Superserver.

Parâmetros específicos do Windows

CreateInternalWindow

O protocolo "Windows local" usa uma janela escondida para a comunicação inter-processos entre o cliente local e o servidor. Esta janela IPC é criada no arranque do servidor quando o CreateInternalWindow está a true (1, valor por defeito). Coloque-o a 0 (desactivado) para correr o servidor sem a janela e desactivar o protocolo local. Com o protocolo local desactivado, é possível correr múltiplas instâncias do servidor simultaneamente.

DeadThreadsCollection

Uma parametrização para o "thread scheduler" no Windows, este parâmetro inteiro define o número de ciclos de mudança de prioridade (veja PrioritySwitchDelay, em baixo) que o "scheduler" executará antes de destruir um thread (ou fechá-lo).
A imediata destruição (ou fecho) de threads activas requereria um semáforo e uma chamada de bloqueio, gerando uma sobrecarga considerável. Em vez disso, um "thread scheduler" mantêm os threads numa fila de gestão. Quando o thread tiver completado a sua tarefa, será marcado como inactivo. O thread inactivo é destruído (ou fechado) após *n* iterações da ciclo do scheduler, onde *n* é o valor do parâmetro do DeadThreadsCollection.
Para um servidor poder gerir um grande número de conexões—na ordem das centenas ou mais—o valor do parâmetro terá que ser aumentado para cima do seu valor por defeito, que é 50.

GuardianOption

Parâmetro Booleano usado em servidores de Windows para determinar se o Guardian dever reiniciar o servidor de cada vez que este termine anormalmente. O valor por defeito de instalação (1=True), assim o impõe. Para desactivar o comportamento descrito, coloque o parâmetro off (0=False).

IpcMapSize

Era *server_client_mapping* em ibconfig

Tamanho em bytes de uma secção do arquivo mapeado em memória de um cliente usado para a comunicação inter-processos (IPC) no modelo de conexão usado pela conexão "Windows local". Não possui qualquer equivalência noutras plataformas. Inteiro, de 1024 até 8192. O valor por defeito é 4096.
Ao aumentar o tamanho do mapa poderá melhorar o desempenho quando estiver em causa leituras de grande dimensão ou número de dados, tal como acontece com o uso de BLOBs com gráficos.

> NOTA: Este valor já não é passível de ser alterado pelo diálogo gerado pelo ícone do Guardian localizado na Área de Notificação.

IpcName

Valor por defeito: FirebirdIPI
O nome da área de memória partilhada usada como canal de transporte no protocolo local.
Na Versão 1.5 o valor por defeito—FirebirdIPI—não é compatível com as versões anteriores do Firebird nem com o InterBase®. Use o valor InterBaseIPI para repor a compatibilidade, se necessário.

MaxUnflushedWrites

Este parâmetro foi introduzido na Versão 1.5 para lidar com o erro nos sistemas operativos servidores Windows, onde as escritas assíncronas nunca eram gravadas para o disco excepto quando o servidor Firebird era sujeito a um shutdown controlado. (Escritas assíncronas não são suportadas no Windows 9x ou ME.) Como tal, em sistemas 24/7, as escritas assíncronas nunca eram gravadas de todo.
Este parâmetro determina quão frequentemente as páginas retidas são gravadas para o disco quando o "Forced Writes" é desactivado (escrita assíncrona activada). O seu valor é um inteiro, o qual define o número de páginas a reter antes de ser despoletada uma gravação para disco e a qual será executada da próxima vez que for efectuado um commit da transacção. O valor por defeito é 100 nas instalações do Windows e -1 (desactivado) nas instalações das outras plataformas.
Se o final do ciclo MaxUnflushedWriteTime (ver em baixo) é atingido antes do número de páginas retidas ser atingido, a gravação é requerida imediatamente e o número de páginas retidas é reposto a zero.

MaxUnflushedWriteTime

Este parâmetro determina o tempo máximo de retenção das páginas, para a escrita assíncrona, antes de serem gravadas para disco quando o Forced Writes é desactivado (escrita assíncrona activada). O seu valor é um inteiro, o qual define o intervalo, em segundos, entre a última gravação para disco e a activação de um sinal para ser executada a gravação, na próxima vez que for efectuado um commit da transacção. O valor por defeito é de 5 segundos nas instalações Windows e -1 (desactivado) nas instalações das outras plataformas.

PrioritySwitchDelay

Uma configuração do "thread scheduler" no Windows, este valor inteiro estabelece o tempo, em milissegundos, que decorre antes de a prioridade de uma thread inactiva ser colocada em LOW ou a prioridade de uma thread activa ser colocada em HIGH. Uma iteração desta sequência de mudança representa um ciclo do "thread scheduler".
O valor por defeito é 100 ms, baseado em avaliação experimental nos processadores Intel PIII/P4. Para processadores com relógios de velocidade mais baixa, é necessário um atraso maior.

PriorityBoost

Inteiro, define o número de ciclos extra dados a uma thread quando a sua prioridade é colocada em HIGH. O valor por defeito é 5.

ProcessPriorityLevel

Era *server_priority_class* no ibconfig
Classe/Nível da prioridade para o processo do servidor. Este parâmetro substitui o parâmetro server_priority_class das versões pre-1.5, — ver em baixo— com uma nova implementação.
Os valores são inteiros, como a seguir:

- 0 – prioridade normal,
- valor positivo – prioridade elevada (o mesmo que a opção -B[oostPriority] no instsvc.exe nas opções *configure* e *start*)
- valor negativo – prioridade baixa.

Nota: Todas as modificações deste valor deverão ser cuidadosamente testadas para assegurar que elas conseguem que o "engine" seja efectivamente mais rápido a responder.

RemotePipeName

Aplicável apenas para conexões NetBEUI

Parâmetro String, o nome da "pipe" usada como canal de transporte no protocolo NetBEUI. O nome da "pipe" é equivalente ao número da porta para o TCP/IP. O valor por defeito—interbas— é compatível com as versões mais antigas do Firebird e do InterBase®.

Parâmetros para configurar o espaço temporário usado para ordenações

Quando o tamanho do buffer interno é demasiado pequeno para acomodar as linhas envolvidas numa operação de ordenação, o Firebird necessita de criar ficheiros temporários de ordenação no sistema de ficheiros do servidor. Por defeito, irá procurar por um caminho especificado na variável de ambiente **FIREBIRD_TMP**. Se tal variável não estiver presente, tentará usar a raiz do **/tmp** do sistema de ficheiros do Linux/UNIX, ou **C:\temp** no Windows NT/2000/XP. Nenhum destes locais podem ser configurados quanto ao tamanho.
O Firebird providencia um parâmetro para configurar o espaço em disco que será usado para guardar estes ficheiros temporários. É prudente usá-lo, para assegurar que existirá espaço suficiente para ordenação em todas as condições.
Todos os pedidos CONNECT ou CREATE DATABASE partilham a mesma lista de ficheiros temporários e cada um cria os seus próprios ficheiros temporários. Os ficheiros de ordenação são libertados após o fim da ordenação ou após o pedido ser libertado.
Na Versão 1.5, o nome do parâmetro mudou de **tmp_directory** para **TempDirectories** e a Sintaxe do parâmetro também mudou.

TempDirectories

Substitui as ocorrências de *tmp_directory* em isc_config/ibconfig

Fornece uma lista de uma ou mais directorias, separadas por ponto-e-vírgula (;), sob as quais estes ficheiros temporários poderão ser armazenados. Cada item poderá incluir um argumento de tamanho opcional, em bytes para limitar o seu armazenamento. Se o argumento for omitido, ou for inválido, o Firebird irá usar essa directoria até que ela esteja esgotada, antes de passar para a próxima directoria da lista.
Por exemplo,
Unix: `/db/sortfiles1 100000000;/firebird/sortfiles2`
Windows: `E:\sortfiles 500000000`
Paths relativos são tratados como relativos ao path que o servidor em execução reconhece como directoria da raiz da instalação do Firebird. Por exemplo, no Windows, se a directoria da raiz for C:\Programas\Firebird, então o valor seguinte dirá ao servidor para armazenar os ficheiros temporários em C:\Programas\Firebird\userdata\sortfiles, até ao limite de 500 Mb:

```
TempDirectories = userdata\sortfiles 500000000
```

Nota: Sem aspas ao contrário de como era requerido no Firebird 1.0

Parâmetros de Compatibilidade

CompleteBooleanEvaluation

Estabelece o método de avaliação dos Booleanos (completo ou por atalho). O valor por defeito (0=False) corresponde a seguir um "atalho" na avaliação duma expressão dum Booleano envolvendo os predicados AND ou OR, devolvendo logo que um resultado de True ou False seja obtido e que não possa ser afectado pelos resultados de qualquer avaliação posterior.
Sob certas raras condições (habitualmente evitáveis), poderá acontecer que uma operação dentro de uma condição OR ou AND que não tenha sido avaliada devido a um comportamento de atalho, tem o potencial de afectar o resultado original. Se tiver o infortúnio de herdar uma aplicação que tenha tais características na sua lógica de SQL, poderá desejar usar este parâmetro para forçar a completa avaliação até que tenha a oportunidade de o rectificar. O tipo de parâmetro é Booleano.

Não subestime o facto de que esta opção afecta todas as avaliações de Booleanos em qualquer base de dados do servidor.

OldParameterOrdering

A versão 1.5 abordou e resolveu um velho erro do InterBase que motivava que parâmetros de saída fossem devolvidos ao cliente com uma idiossincrática ordenação na estrutura XSQLDA. O erro tinha tão grande longevidade que muitas aplicações, drivers e interfaces existentes possuíam soluções para contornar o problema no lado do cliente.
As versões 1.5 e posteriores reflectem a correcção na API e são instaladas com o OldParameterOrdering=0 (Falso). Mude este parâmetro Booleano para True se necessitar de reverter, por razões de compatibilidade, ao velho comportamento.

Aliasing dos Ficheiros BD

O Firebird versão 1.5 introduziu o aliase de ficheiros de base de dados para melhorar a portabilidade das aplicações e para apertar o controlo quer interno quer externo do acesso às bases de dados.

Aliases.conf

Configure os alias de ficheiros de base de dados no ficheiro texto aliases.conf, localizado na raiz da directoria da sua Instalação do servidor Firebird. O aliases.conf instalado tem o seguinte aspecto:

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

Tal como em todos os ficheiros de configuração do Firebird, os símbolos '#' são marcadores de comentários. Para configurar um alias, apague simplesmente o '#' e mude a linha com o "dummy" para o valor correcto do path da base de dados:

```
# fbdb1 está no servidor Windows:
fbdb1 = c:\Firebird\sample\Employee.fdb
# fbdb2 está no servidor Linux
fbdb2 = /opt/databases/killergames.fdb
#
```

Você pode editar o aliases.conf enquanto o servidor se encontra a correr. Não existe qualquer necessidade de parar e reiniciar o servidor para que as novas alterações no aliases.conf sejam assumidas.

Conexão usando um path com alias

A string de conexão modificada na aplicação do seu cliente terá o seguinte aspecto:

```
Nome_servidor:nomealias
```

Com o exemplo abaixo, a seguinte string de conexão irá pedir ao servidor Firebird em execução numa "caixa" Linux de nome "meuservidor" para procurar e conectar o cliente à base de dados localizada no path identificado em aliases.conf como "fbdb2":

```
meuservidor:fbdb2
```

Nota: porque a ferramenta **gstat** não usa uma ligação à base de dados para ler o ficheiro da base de dados, é necessário providenciar um "path" completo para fazer uso do gstat. (Poderá ser alterado).

Nomear bases de dados no Windows

Note que agora a extensão recomendada para as bases de dados no Windows ME e XP é ".fdb" para evitar possíveis conflitos com a funcionalidade "System Restore" do Windows. Se não se tiver tal cuidado nessas plataformas, irá despertar os conhecidos problemas de atraso na primeira conexão à base de dados cujo ficheiro primário e/ou secundário usem a extensão convencional ".gdb".

Equipas de Desenvolvimento Firebird

Programador	País	Tarefas principais
Dmitry Yemanov	Federação Russa	Coordenador das Versões; melhorias no DSQL e PSQL; implementação do Embedded Server, numerosas melhorias no metadata, aliasing da base de dados, trigger multi-acção, tipo de dados BigInt, novas variáveis de contexto, servidor Classic para Windows; numerosas resoluções de erros
Nickolay Samofatov	Federação Russa	Desenho e implementação de funcionalidades SQL (Savepoints, locking pessimista); melhorias no metadata; re-implementações do "engine"; pesquisa e correcção de erros; problemas de arquitectura; Serviços API no Linux Classic activados; melhoria do desempenho; builds Classic para Linux
Arno Brinkman	Países Baixos	Melhorias do Optimizador; muitas novas funcionalidades DSQL
Claudio Valderrama	Chile	Escrutínio do Código; pesquisa e correcção de erros; melhorias do PSQL; Correcção da UDF, desenho e implementação
Alex Peshkoff	Federação Russa	Novas funcionalidades PSQL e DSQL; autor e coordenador de funcionalidades de segurança; correcção do código; builds Superserver para Linux
Mike Nordell	Suécia	Conversão do código do Firebird para C++; melhoria do desempenho; transporte de funcionalidades; pesquisa e correcção de erros
Blas Rodriguez Somoza	Espanha	Programador de novos character sets; limpeza do código e remodelação da árvore; builds MinGW
Roman Rokytskyy	Alemanha	Programador coordenador do Jaybird
David Jencks	U.S.A.	Desenhador e coordenador do JayBird; desenhador de ferramentas de documentação do Firebird
Carlos Guzman Alvarez	Espanha	Desenhador e coordenador do .NET provider para o Firebird
John Bellardo	U.S.A.	Implementou interface plug-in para o character sets; coordenador de builds Darwin; implementação inicial do novo modelo de memória
Erik Kunze	Alemanha	Pesquisa e correcção de erros; limpeza do código; builds SINIX-Z
Dmitry Sibiryakov	Federação Russa	Limpeza do código; builds MinGW
Pavel Cisar	República Checa	Builds Linux (Versão 1.0); desenhador/coordenador de ferramentas QA
Ann Harrison	U.S.A.	Correcção de erros; conselheira técnica; aumento do numero máximo de índices
Mark O'Donohue	Austrália	Funcionalidades de linha do isql; correcção de erros; boot builds (Versão 1.0); correcção do código (Versão 1.0)

Programador	País	Tarefas principais
Paul Reeves	França	QA; instaladores Win32; controlo de painel standard Win32
Ignacio J. Ortega	Espanha	Adicionadas funcionalidades de PLAN para triggers; limpeza de código
Konstantin Kuznetsov	Federação Russa	Builds Intel Solaris
Olivier Mascia	Bélgica	Re-implementação de serviços de instalação no Win32
Peter Jacobi	Alemanha	Aperfeiçoamentos, actualizações de character sets
Tilo Muetze	Alemanha	Coordenador do projecto de documentação Firebird
Paul Vinkenoog	Países Baixos	Coordenador, projecto de documentação do Firebird; correcções do UDF
Artur Anjos	Portugal	Melhorias no painel de controlo Win32; programador do Gestor de Configuração do Firebird; Internacionalização do mesmo
Achim Kalwa	Alemanha	Melhorias no programa do painel de controlo do Win32
Sean Leyne	Canada	Organizador do Bugtracker; limpeza do código
Ryan Baldwin	U.K.	Programador do driver Jaybird Tipo 2
Sandor Szollosi	Hungria	Implementador de character set collations
Dmitry Kuzmenko	Federação Russa	Resolução de erros de GSTAT
Artem Petkevych	Ucrânia	Resolução de erros do tipo de dados ARRAY
Vlad Horsun	Ucrânia	Aumento de velocidade do sweep; corrigido o commit de 2-Fases
Tomas Skoda	Eslováquia	Resolução de erros
Evgeny Kilin	Federação Russa	Resolução de erros
Oleg Loa	Federação Russa	Resolução de erros
Erik S. La Bianca	U.S.A.	Resolução de erros
Tony Caduto	U.S.A.	Instaladores Win32 não-oficiais
Juan Guerrero	Espanha	Novas UDFs
Chris Knight	Austrália	Builds FreeBSD
Neil McCalden	U.K.	Builds Solaris
Grzegorz Prokopsi	Hungria	Builds Debian
Paul Beach	U.K.	Builds HP-UX

Geoffrey Speicher	U.S.A.	Builds FreeBSD
Helen Borrie	Austrália	Autora das “Notas da Versão”; testes de campo e “Thought Police”

"OS HERÓIS DOS TESTES 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 D E I N S T A L A Ç Ã O

Instalar o Firebird 1.5 em Windows 32

LEIA ISTO PRIMEIRO!

Com a introdução de dois novos modelos de servidor no Win32 as escolhas para instalação do Firebird aumentaram.

- Garanta que está conectado como Administrador (não se aplica ao Win9x ou ME)

- Todos os modelos—Superserver, Classic e Embedded Server tal como ferramentas para servidor e para cliente—podem ser instalados usando o programa de instalação para Windows. Para uma instalação completa, recomenda-se vivamente que se use o instalador se existir algum.

- Use o **gbak** para efectuar cópias de segurança da sua base de dados de segurança **isc4.gdb**. Pode repô-la mais tarde com o nome **security.fdb**

- Se possuir configurações especiais no **ibconfig** poderão existir alguns valores que queira transferir para os parâmetros equivalentes no **firebird.conf**. Estude as notas sobre o firebird.conf para poder avaliar o que pode ser copiado directamente e o que necessita de assumir uma nova Sintaxe.

- Se certos ficheiros de configuração existirem na directoria de instalação estes serão preservados se executar o programa de instalação e SOBREPOSTOS se descomprimir o kit (ficheiro zip) para o local por defeito. Os ficheiros são
  security.fdb
  firebird.log
  firebird.conf
  aliases.conf

- Este modelo pode ser instalado a partir de um ficheiro zip. Este método será mais rápido do que o programa de instalação se possuir experiência suficiente em instalar Firebird 1.5 a partir de ficheiros zip. Todavia esta será uma tarefa desesperante se for um iniciado no Firebird.

- É assumido que
  1 Você compreende como funciona a sua rede.
  2 Você compreende porque um sistema cliente/servidor necessita simultaneamente de servidor e cliente
  3 Você leu as restantes notas da versão—ou pelo menos tem consciência de que necessita de lê-las se algo correr mal
  4 Você sabe que pode ir à lista do firebird-support se se encontrar em dificuldades. Inscreva-se em http://www.yahoogroups.com/groups/firebird-support

Se já possuir uma versão anterior do Firebird ou InterBase® no seu servidor e pensa que poderá retornar a ela, salvaguarde a sua situação actual antes começar.

- Use a versão existente do GBAK para efectuar uma cópias de segurança dos seus ficheiros de base de dados no formato transportável

- Vá à sua directoria de Sistema e faça uma cópia de backup do gds32.dll. Sugerimos que nomeie o backup como "gds32.dll.ib5" ou "gds32.dll.fb103", ou algo igualmente informativo; ou esconda-o noutra directoria

- Poderá também ser uma boa ideia efectuar um backup da biblioteca de runtime do Microsoft C++, msvcp60.dll. O programa de instalação não deverá sobrepor a sua versão deste ficheiro, mas algo de anormal pode acontecer.

- **PARE QUALQUER SERVIDOR FIREBIRD OU INTERBASE QUE SE ENCONTRE EM EXECUÇÂO**
  O programa de instalação irá tentar detectar qualquer versão existente do Firebird ou InterBase já instalada e/ou em execução. Numa instalação sem programa de instalação, você está por contra própria!

- O local por defeito da raiz do Firebird 1.5 será C:\Programas\Firebird. Se a sua versão anterior já estiver instalada nessa directoria e você pretende usar as localizações por defeito do 1.5, renomeie a directoria existente

- Para instalar o Firebird como serviço: se pretende fazer uso na nova funcionalidade **login seguro**, crie um "utilizador de serviço firebird" no sistema— com nome e password à sua escolha— como utilizador normal com privilégios apropriados. Você deverá ler primeiro o documento de nome README.instsvc.txt. Se possuir um kit comprimido, encontrá-lo-à na directoria /doc da raiz do ficheiro zip. Se não possuir um kit comprimido disponível, o ficheiro não será acessível até que finalize a instalação. Você pode ler o mesmo documento no seguinte endereço:
  http://cvs.sourceforge.net/viewcvs.py/firebird/firebird2/doc/README.instsvc

LEIA ISTO A SEGUIR!

Um dos objectivos do Firebird 1.5 é preparar o caminho para múltiplas instalações do servidor. Isto irá permitir que os utilizadores executem diferentes versões lado a lado. O Firebird 1.5 já o suporta, apesar de não estar bem documentado e exigir a intervenção de um utilizador com conhecimentos avançados. Futuras versões do Firebird irão tornar este processo menos complicado. Entretanto o Firebird 1.5 precisa de preparar o terreno. Isto coloca-nos perante a questão da instalação de diferentes bibliotecas. Ao mesmo tempo, a Microsoft contempla de forma própria a instalação de diferentes versões de bibliotecas. Encarando em conjunto estas duas questões separadas implica uma nova abordagem da instalação de bibliotecas para o Firebird 1.5 e seguintes.

Instalação de bibliotecas dos sistemas Microsoft

O problema associado com a instalação de diferentes versões das bibliotecas do sistema Microsoft são tão importantes que mereceram o nome de 'Inferno DLL'.

A partir do lançamento do Windows 2000 inclusive a Microsoft tornou quase impossível a actualização dos dll do sistema. Para resolver isto a Microsoft agora recomenda que cada aplicação instale cópias locais das bibliotecas de sistema que sejam necessárias.

Firebird 1.5 segue esta prática e coloca as bibliotecas necessárias na directoria \bin juntamente com o servidor.

Instalação do fbclient.dll

Para o Firebird 1.5 e seguintes já não mais será usado o gds32.dll como biblioteca cliente. Agora foi renomeado para fbclient.dll. Dados os problemas que a Microsoft teve com o inferno DLL não faria muito sentido se nós continuássemos a guardar a biblioteca cliente do Firebird na directoria de <sistema>. E como nós pretendemos que sejam permitidas a instalação simultânea de múltiplos motores de servidor, estaríamos então a criar o nosso inferno DLL se continuássemos com a prática de usar a directoria de <sistema> para as bibliotecas do cliente. Como tal, a partir do Firebird 1.5 e seguintes, a biblioteca do cliente passa a residir na directoria \bin conjuntamente com os restantes binários.

Uma nova chave de registo foi adicionada e todas as aplicações Firebird compatíveis com o Firebird devem passar a usá-la para localizar a versão correcta do Firebird que pretendam usar. A nova chave é:

HKEY_LOCAL_MACHINE\SOFTWARE\Firebird Project\Firebird Server\Instances

O Firebird garantirá que pelo menos exista uma entrada nesta chave. Será conhecida como

"DefaultInstance"

e conterá o path para a directoria da raiz (sim, o seu palpite está correcto) da instalação por defeito. Aqueles que não se importam com instalações particulares podem sempre usar a instancia por defeito para localizar o fbclient.dll.

Futuras versões do Firebird não verão mais quaisquer outras entradas abaixo de Instances. As aplicações serão capazes de analisar as entradas de registo para determinar qual a biblioteca que pretendem carregar.

Suportando aplicações legacy e drivers

Tradicionalmente, as aplicações que usam o Interbase ou o Firebird contavam com carregar a biblioteca cliente gds32.dll a partir da directoria de <sistema>. O Firebird 1.5 fornece uma ferramenta de nome 'instclient.exe' que pode instalar uma cópia do fbclient.dll na directoria de Sistema do Windows. Esta cópia é modificada por forma que a informação da versão do ficheiro comece com "6.3", para providenciar compatibilidade com aplicações mais antigas que dependam da versão do ficheiro e não consigam interpretar números de versão tais como "1.5".

Durante o processo de instalação o programa de instalação verifica se já existe uma instalação do InterBase ou Firebird. Se nenhuma estiver já instalada irá instalar o gds32.dll na directoria de <sistema>. Se porventura detectar qualquer possível versão do Firebird ou InterBase possa já ter sido instalada não procederá à instalação do gds32.dll na directoria de <sistema>. Contudo, a ferramenta 'instclient.exe' pode ser usada mais tarde para efectuar tal procedimento.

É suposto que em futuras versões do Firebird não será feita qualquer tentativa de instalar o gds32.dll na directoria de <sistema> e por último será completamente removida da distribuição.

Esta ferramenta 'instclient.exe' pode também instalar o FBCLIENT.DLL na directoria de sistema do Windows, se necessário. Isto resolverá a questão colocada por algumas ferramentas ou aplicações que dependem da sua localização nessa directoria.

O utilitário instclient.exe deverá estar localizado na directoria 'bin' da sua instalação do Firebird e deverá ser executado a partir daí.

Utilização do instclient.exe:

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

onde *biblioteca* significa: `fbclient | gds32`

'-z' pode ser usado com qualquer outra opção, imprime a versão.

A versão da informação e o contador das bibliotecas partilhadas são geridos automaticamente. Pode no entanto utilizar a opção -f[orce] para sobrepor o teste da versão.

NOTA Se utilizar o -f[orce] na instalação, poderá afectar outra versão Firebird ou InterBase® previamente instalada. Poderá ter que reinicializar a máquina por forma a finalizar a cópia.

Para mais detalhes, consulte a documentação *README.Win32LibraryInstallation.txt* a qual está localizada ou na raiz da instalação ou em ..\doc.

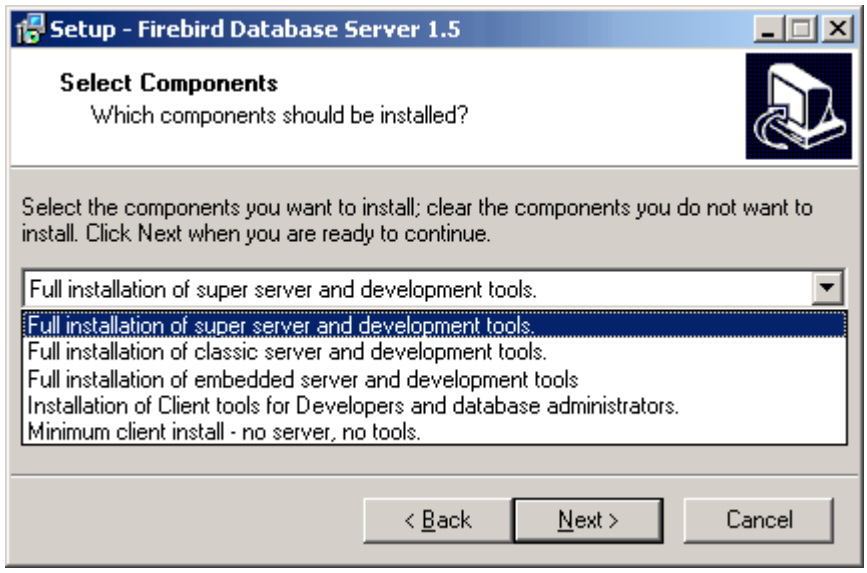
Limpeza de instalações prévias de release candidate

Deverá ser considerado que o programa de instalação removerá o fbclient.dll de directoria de <sistema> se o encontrar aí. O programa de instalação também irá remover a chave de registo igualmente abandonada HKLM\Software\FirebirdSQL.

Usando o programa de Instalação Firebird para o Win32

Esta é a instalação mais fácil.

Basta apenas correr o executável e responder aos diálogos. Após ter respondido a aproximadamente 4 diálogos, deverá visualizar uma caixa com uma lista "drop-down" a qual— quando aberta— apresenta um visual idêntico ao mostrado em baixo. Esta é a última hipótese que tem para escolher qual o tipo de instalação que pretende.



Escolha a instalação que pretende e clique em "Next" para avançar nos diálogos.

Serviço ou aplicação?

Se você seleccionar a versão de instalação Superserver ou Classic, e a versão do seu SO suportar serviços, será questionado a escolher se pretende correr o Firebird como serviço ou como aplicação. A menos que tenha uma necessidade específica para correr o servidor como aplicação, opte por serviço.

Manual ou automático?

Com a opção automática, o Firebird irá arrancar sempre que ligue a sua máquina. Com a opção manual você irá poder iniciar o servidor sempre que o pretenda e ordene.

Uso opcional do Guardian

O Guardian é um utilitário que pode correr "em cima" do Superserver e reiniciá-lo sempre que ele estoure por qualquer razão. Num ambiente de desenvolvimento, poderá optar por não o usar. Num ambiente de trabalho, a sua execução poderá evitar uma situação em que o servidor pare de servir e o DBA (administrador) não esteja disponível para o reiniciar.

Directoria de Instalação (Root)

Se decidir não usar a directoria raiz por defeito, navegue para o local que previamente tenha criado para tal; ou simplesmente digite o caminho completo pare ele. O caminho que digitou não tem necessariamente que existir: o programa de instalação, se este não existir, irá pedir a sua confirmação e criá-lo por si.

Eventualmente, os diálogos irão parar e o servidor irá, ou arrancar silenciosamente ou pedir a sua permissão para reiniciar a máquina—a tarefa de reinicialização será necessária se o programa de instalação tiver que sobrepor a sua msvcp60.dll, ou um gds32.dll mais antigo estava carregado quando o programa de instalação se iniciou.

Instalando o Superserver a partir de um kit-comprimido

A instalação do FB 1.5 é similar em princípio ao das versões anteriores.
Se não possui um programa especial de instalação (é distribuído em separado) os passos são os seguintes:

- Use o unzip para descomprimir o arquivo para uma directoria separada (como alguns, poucos, nomes foram mudados, não faz qualquer sentido efectuar o unzip dos ficheiros v1.5 para a directoria que contém o IB/FB1)
- Mude a directoria corrente para <raiz>\bin (aqui e em baixo <raiz> é a directoria onde os ficheiros da v1.5 estão localizados)
- execute instreg.exe:
  instreg.exe install
  faz com que o path de instalação da directoria abaixo seja guardado no registo (HKLM\Software\Firebird Project\Firebird Server\Instances\DefaultInstance)
- se pretende registar um serviço, execute também instsvc.exe:
  instsvc.exe install
- opcionalmente, deverá efectuar uma cópia de fbclient.dll e gds32.dll para a directoria de sistema do OS

Instalando o Classic Server a partir de um kit-comprimido

Para instalar o "engine" CS, a única diferença é uma opção adicional para o instsvc.exe:
instsvc.exe install -classic

Repare que isto significa que só pode ter uma única arquitectura do "engine" --ou fbserver.exe (Superserver) ou fb_inet_server.exe (o processo parente para o Classic)—instalada como serviço.

O **applet do Painel de Controlo** não é instalado com a versão Classic—deliberadamente. Não tente instalá-lo e usá-lo. O conceito de terminar um serviço não se aplica ao modelo Classic.

Instalação simplificada

Se não precisar de um serviço registado, então deverá evitar correr o instreg.exe e o instsvc.exe simultaneamente.

Neste caso deverá simplesmente descomprimir com o unzip o arquivo numa directoria à parte e correr o servidor:

fbserver.exe -a

Neste caso tal significa que a directoria parente será considerada como directoria raiz.

Desinstalação

Para remover o FB 1.5 sem um desinstalador Windows deverá:

- parar o servidor
- executar "instreg.exe remove"
- executar "instsvc.exe remove"
- apagar a directoria de instalação
- apagar os clientes fbclient.dll e gds32.dll na directoria de sistema do SO

Instalando o servidor Embedded a partir de um kit-comprimido

O servidor embedded é um cliente com todas as funcionalidades de um servidor ligado como uma biblioteca dinâmica (fbembed.dll). Este tem exactamente as mesmas funcionalidades do que o Superserver e exporta a interface API standard do Firebird.

Registo As entradas no Registo para o Firebird (onde o servidor normalmente tenta procurar pela localização da directoria da raiz) são ignoradas. A directoria raiz do servidor embedded é a directoria abaixo de onde o ficheiro binário (biblioteca) está localizado.

Acesso à Base de Dados Apenas o acesso "local verdadeiro" é permitido. O servidor "embedded" não possui suporte para protocolos remotos, logo até o acesso via "localhost" não funcionará.

Autenticação e segurança A base de dados de segurança (security.fdb) não é usada no servidor "embedded" e como tal não é necessária. Qualquer utilizador pode conectar-se a qualquer base de dados. Como quer o servidor quer o cliente correm no mesmo espaço de endereço (local), a segurança resume-se a uma questão de acesso físico.

Os privilégios SQL são validados, tal como nos outros modelos.

Compatibilidade Pode correr qualquer número de aplicações com o servidor "embedded" sem quaisquer conflitos. Correr ao mesmo tempo o servidor IB/FB também não constitui qualquer problema.

Mas deverá ter em atenção que não pode aceder simultaneamente à mesma base de dados a partir de múltiplos servidores "embedded", porque eles possuem uma arquitectura SuperServer e necessitam de efectuar um lock exclusivo às bases de dados.

Estrutura de ficheiros para o Servidor Embedded

Basta copiar o fbembed.dll para a directoria onde a sua aplicação está localizada. E então renomeá-lo para fbclient.dll ou para gds32.dll, dependendo de como o seu programa se conecta à base de dados. Efectue cópias com os dois nomes se necessitar de usar as ferramentas de servidor (isql, gbak, etc.)

Também deverá efectuar cópias do firebird.msg, firebird.conf (se necessário) e de ib_util.dll para a mesma directoria.

Se as bibliotecas externas, forem requeridas pela sua aplicação, p.ex. suporte INTL (fbintl.dll) ou bibliotecas UDF, então estes deverão estar localizados à parte da directoria da aplicação. Para as poder usar, coloque-as na numa árvore de directório que emule a do servidor Firebird, i.e., em directorias com o nome /intl e /udf directamente abaixo da directoria onde estão os ficheiros da raiz do Firebird.

Exemplo

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

Então, inicie a sua aplicação. Irá usar o servidor "embedded" como uma biblioteca cliente e poderá aceder a bases de dados locais.

NOTA Desde que a estrutura de directorias esteja em conformidade com estas regras, não será necessário configurar explicitamente a RootDirectory em firebird.conf. Contudo, se decidir distribuir a sua aplicação com um servidor "embedded" e a sua aplicação possuir uma estrutura de directorias diferente, deverá ler primeiro o documento README_embedded.txt da distribuição do Servidor "Embedded", para instruções acerca de configurações adicionais.

Desinstalação

A rotina de desinstalação do Firebird preserva e renomeia os seguintes ficheiros:

preserva security.gdb ou renomeia-o para security.fbnnnn
preserva firebird.log
preserva firebird.conf ou renomeia-o para firebird.confnnnn
preserva aliases.conf ou renomeia-o para aliases.confnnnn

"nnnn" é o número da versão da instalação anterior.
Nenhuma tentativa é feita de remover ficheiros que não tenham feito parte da instalação original. Ficheiros partilhados como é o caso do fbcliente.dll e do gds32.dll serão removidos se o contador de partilha ("share count") indicar que nenhuma outra aplicação o está a utilizar.
As chaves do Registro que foram criadas são removidas.

Outras Notas

Winsock2

O Firebird necessita do WinSock2. Todas as plataformas Win32 deverão já possuir o WinSock2, excepto o Win95. É feito um teste à presença das livrarias Winsock2 durante a instalação. Se não for detectado a instalação irá falhar. Para fazer a actualização do seu sistema consulte este link:
http://support.microsoft.com/default.aspx?scid=kb;EN-US;q177719

Windows ME e XP

No Windows ME e XP (Edições Home e Professional) existe uma "Funcionalidade" chamada "System Restore", que faz uma cópia de segurança automática ("backup caching"?) de todos os ficheiros do sistema que possuam uma terminação ".gdb". O efeito retarda o acesso a bases de dados InterBase/Firebird porque é elaborado um "backup" destes ficheiros sempre que ocorre uma operação de E/S. (No XP não existe "System Restore" em Servidores .NET).

Um ficheiro da directoria Windows do ME, c:\windows\system\filelist.xml, contém os "tipos de ficheiros protegidos ", e os tipos ".gdb" são aí referidos. Charlie Caro de início sugeriu que se removesse a extensão GDB da secção "includes" deste ficheiro. Porém, desde que se execute esta operação, ficou demonstrado que o WinME pode restaurar esta lista. Em XP, não é possível editar o ficheiro filelist.xml.

Em ME, sugere-se uma alternativa que consiste em:

- Utilizar a extensão FDB (Firebird DB) para os ficheiros Base de Dados
- Mover a Base de Dados para C:\Meus Documentos, que é ignorada pelo "System Restore"
- Desactivar por completo a opção "System Restore" (consulte a documentação do Windows para instruções).

No Windows XP Edições Home e Professional poderá mover as suas bases de dados para uma partição em separado, e configurar o "System Restore" para excluir esse volume.

O Windows XP utiliza uma "smart copy", pelo que o "overhead" visto em Windows ME não deve ser tão notável como no XP, pelo menos para ficheiros pequenos. Para ficheiros maiores (como qualquer Base de Dados Firebird tem tendência a ser!) não parece existir uma solução melhor se tiver os seus ficheiros ".gdb" localizados no sistema de ficheiros geral.

Estamos a tentar obter uma descrição acurada do problema e uma solução para o mesmo será colocado aqui. Se pode ajudar com uma descrição do problema ou com uma solução alternativa, por favor envie uma mensagem para a lista ib-support ou para a lista firebird-devel em news://news.atkin.com

O comportamento do encerramento no Windows XP constitui uma área sobre a qual pouco se sabe. Existe uma significativa pausa enquanto o servidor tenta parar o serviço. Durante este momento o écran indica que o Firebird está a correr como aplicação.
O problema aparenta só afectar o Windows XP e só acontece se o Guardian não se encontrar a ser usado para parar o serviço do servidor. Esta constitui uma forma de contornar o problema até uma melhor solução ser encontrada.

Instalação no UNIX / Linux

(Originalmente escrito por Mark O'Donohue, revisto para 1.5)

O servidor Firebird é fornecido em dois formatos, "Classic" que corre como um serviço, ou "SuperServer" que corre como um "background daemon". Aconselha-se o utilizador que está a iniciar com o Firebird a utilizar o "Classic", pois com certeza será uma melhor plataforma para experimentar o Firebird.

NOTAS – LEIA ISTO PRIMEIRO

1) Tem de possuir direitos "root" para instalar o Firebird.
2) A instalação em sistemas Linux requer que o pacote glibc instalado seja igual ou maior do que o glibc-2.2.5 e o libstdc++.so igual ou maior do que o libstdc++-5.0.

3) Para uma avaliação "tosca" da compatibilidade Linux, consulte esta tabela, mas não a considere como a sendo "a última palavra" sobre este assunto. Os binários das distribuições Linux podem variar dependendo de quem as construiu.
4) Para as instalações usando rpm, garanta que os editores de pacotes 'ed' e 'vim' estão instalados no seu sistema. Se o editor requerido não estiver presente, o pacote será instalado mas os scripts de instalação irão falhar.

INSTALAÇÂO NO LINUX

As instruções seguintes descrevem os procedimentos para uma instalação da versão Classic. Para instalar a versão Superserver o "CS" deverá ser substituido por "SS" no nome do pacote. Por exemplo, o pacote FirebirdCS-1.5.0-nnnn.i686.rpm é substituído por FirebirdSS-1.5.0-nnnn.i686.rpm.

Para uma instalação rpm linux

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

Para uma instalação .tar.gz linux

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

* ou `FirebirdSS-1.5.0-nnnn`

O que faz a instalação Linux

A instalação Linux irá:

1. Tentar parar algum servidor que esteja em execução;
2. Se existir uma instalação previa do Firebird, então esta e todos os ficheiros associados em /usr/lib /usr/include serão arquivados no ficheiro /opt/interbase_<datetimestamp>.tar.gz e de seguida serão eliminados.
3. Instalar o software na directoria /opt/firebird, as bibliotecas em /usr/lib e as "headers" em /usr/include
4. Automaticamente adicionar gds_db para a porta 3050 em /etc/services
5. Automaticamente adicionar localhost.localdomain e HOSTNAME a /etc/host.equiv
6. A versão SuperServer ainda instala um script de arranque em /etc/rc.d/init.d/firebird.
7. A versão do servidor Classic instala um script de arranque /etc/xinetd.d/firebird ou, para versões antigas de sistemas inetd, adiciona uma entrada no ficheiro /etc/inetd
8. Particularmente para o SuSE, um novo "link" rcfirebird é criado em /usr/bin para o script init.d e é criada uma entrada Firebird em /etc/rc.config.
9. Arrancar o servidor/serviço. O Firebird deverá arrancar automaticamente no nível 2, 3 ou 5
10. Gerar de forma aleatória uma nova password SYSDBA a qual é guardada no ficheiro /opt/firebird/SYSDBA.password.
11. Adiciona uma entrada em alias.conf para a base de dados exemplo, employee.fdb

A instalação "Classic" automaticamente configura a entrada xinetd se a directoria /etc/xinetd.d for encontrada, senão cria uma entrada inetd. Como algumas distribuições colocam o xinetd numa localização diferente da directoria /etc/xinetd.d, uma configuração manual é requerida nestas condições.

Testes da instalação Linux

Passo 1 – Acedendo à base de dados

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

SQL>connect localhost:employee.fdb /* isto é um caminho alias */

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

>quit;
```

*Uma password foi gerada na sua instalação. Pode ser obtida a partir do ficheiro /opt/firebird/SYSDBA.password.

Passo 2 – Criando uma base de dados

A partir da versão 1.5 e seguintes, o servidor firebird será executado por defeito pelo utilizador 'firebird'. Embora tal configuração tenha sempre sido recomendada, o comportamento por defeito era que o servidor corresse como utilizador 'root' . Ao correr como utilizador root, o servidor possuía um amplo leque de permissões para ler, criar e eliminar ficheiros em qualquer parte de um sistema de ficheiros POSIX. Por razões de segurança, o serviço deverá ter permissões mais limitadas para ler/apagar e criar ficheiros.

Enquanto a nova configuração é melhor do ponto de vista da segurança, requer no entanto algumas considerações a serem levadas em linha de conta ao criar novas bases de dados:

a) O utilizador 'firebird' tem que possuir permissão de escrita na directoria na qual pretende criar a base de dados.
b) O valor recomendado do atributo DatabaseAccess no ficheiro /opt/firebird/firebird.conf deverá ser modificado para None, para permitir acesso apenas através de entradas no ficheiro aliases.conf.
c) Utilize entradas no aliases.conf para os utilizadores dos locais físicos da localização das bases de dados. Mais notas acerca de aliases podem ser consultados aqui.

Os procedimentos para criar uma nova base de dados podem variar em função das diferentes configurações por si escolhidas mas os passos seguintes são os que eu recomendo com a configuração sugerida:

1) Se uma directoria cujo dono é o utilizador 'firebird' não existir, mude para o utilizador "root" e crie essa directoria:

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

2) Crie uma nova base de dados e insira uma entrada "alias" para ela. Como utilizador root ou firebird, execute o seguinte:

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

(Como usar: createDBAlias.sh <nome da bd> <caminho para a bd>)

3) Como alternativa (para o passo 2) os passos no script createDBAlias.sh poderão ser executados manualmente do seguinte modo:

```
$vi /opt/firebird/aliases.conf
e adicione a seguinte linha no fim do ficheiro:
test.fdb /var/firebird/test.fdb
```

4) A seguir crie a base de dados:

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

5) Se o valor DatabaseAccess em /opt/firebird/firebird.conf estiver como Full ou com um valor de caminho restringido (por exemplo: DatabaseAccess=/var/firebird) outra alternativa ao passo 2 será criar o ficheiro de base de dados directamente, usando o caminho absoluto com o nome do ficheiro:

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

Se usar esta configuração, o ficheiro da base de dados pode também ser acedido directamente sem que exista uma entrada no ficheiro de aliases:

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

*Uma password foi gerada automaticamente durante a instalação. Poderá ser obtida lendo o ficheiro /opt/firebird/SYSDBA.password.

Dicas e Scrips Auxiliares

Adicionalmente aos ficheiros standard de instalação, são fornecidos na directoria bin do Firebird os seguintes scripts:

- **changeDBAPassword.sh**— Altera a password do utilizador Firebird SYSDBA. Para o Superserver, este script modifica também o "init script" /etc/rc.d/init.d/firebird com a nova password.

- **createAliasDB.sh**— Como usar: createAliasDB.sh <nome bd> <caminho para bd>.
  Este script cria uma nova base de dados e adiciona uma entrada no ficheiro de alias.conf.

- **fb_config**— Um script que poderá ser usado em makefiles para gerar os paths de include necessários e incluir as directivas de lib para a versão instalada do Firebird. fb_config -help dar-lhe-á uma lista completa das opções.

- **changeGdsLibraryCompatibleLink.sh**— Apenas para a versão Classic—Muda o link da biblioteca do cliente para libgds.so entre a libfbclient.so multithreaded e a biblioteca libfbembed.so single threaded que permite uma abertura "embedded" directa do ficheiro bd. Par razões de compatibilidade com instalações prévias, libgds.so por defeito aponta para libfbembed.so.

- **Acesso directo ou "Embedded" a ficheiros de bases de dados**
  A instalação da versão Classic permite um acesso em modo embedded para que um programa possa abrir os ficheiros de base de dados directamente. Para operar neste modo, o utilizador que acede à base de dados deverá possuir privilégios para aceder a alguns dos ficheiros de configuração e estado do Firebird.

- **Obtendo acesso a Bases de Dados:** Agora que o utilizador por defeito que executa o software é o utilizador 'firebird' (e não o root), é necessário saber como **registar um utilizador no grupo firebird** para obter acesso às bases de dados. Está documentado nas notas *readme*, mas os seguintes passos deverão ajudá-lo a conseguir o pretendido:
  Para adicionar um utilizador (p.ex. skywalker) ao grupo firebird, o utilizador root precisa de efectuar o seguinte:

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

Da próxima vez que o 'skywalker' efectue um log in, poderá de imediato trabalhar com as bases de dados firebird.

Para listar os grupos a que um utilizador pertence, insira o seguinte comando de linha:

```
$ groups
```

- **Problemas NTPL em versões recentes do Linux:**
  O novo NPTL (Native POSIX Thread Library) no Red Hat 9 (pelo menos até esta versão) irá provocar problemas à versão SuperServer e a programas compilados localmente, incluindo os utilitários. E em particular o Gbak, irá despoletar um erro de Broken Pipe. Para corrigir:

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

Isto deverá resolver o problema do servidor. Tem também que possuir uma variável de ambiente definida no seu ambiente local, como tal terá que:

2. Adicionar o seguinte /etc/profile, para garantir que todo o utilizador o use para poder executar os utilitários de comando de linha.

após
```
  HISTSIZE=1000
```
adicione
```
  LD_ASSUME_KERNEL=2.25
```
E na linha seguinte:
```
 export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUT_RC LD_ASSUME_KERNEL
```

Desinstalando no Linux

Se precisar de efectuar uma desinstalação, faça-o como utilizador root. Os seguintes exemplos usam a versão CS ou ClassicServer mas o mesmo será válido para o SuperServer substituindo CS por SS.

Para pacotes rpm:

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

ou para instalações com .tar.gz:

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

Instalação do Firebird Classic & SuperServer no Solaris 2.7 Sparc

Não está actualmente disponível. Por favor consulte as Notas da Versão de v.1 como uma referência para as instalações 1.5.

Instalação do Firebird Classic no MacOS X / Darwin

Não está actualmente disponível. Por favor consulte as Notas da Versão de v.1 como uma referência para as instalações 1.5.

Build ou Instalação do Firebird no FreeBSD

Não está actualmente disponível. Por favor consulte as Notas da Versão de v.1 como uma referência para as instalações 1.5.

Configurando a porta de serviço no cliente e no servidor

Por defeito um servidor Firebird lê os pedidos dos clientes na porta 3050. O nome do registo do serviço da porta é **gds_db**. A boa novidade é que, se funcionar tudo bem com estes valores por defeito não terá que efectuar qualquer configuração quer no servidor quer no cliente para configurar o serviço à porta.

Poderá no entanto usar uma porta diferente, um nome de serviço diferente, ou ambos. Isto poderá ser um imperativo se a porta 3050 estiver requisitada por outro serviço, ou por exemplo, estiver em execução um gds_db concorrente configurado para uma versão diferente do servidor Firebird ou InterBase® . Existem diversas alternativas para modificar os valor por defeito. Tanto o servidor como o cliente deverão ser configurados por forma a atribuir novos valores ao nome da porta de serviço ou ao seu número, ou a ambos, pelo menos através de um dos métodos seguintes:

1. na string de conexão do cliente
no comando efectuado pelo utilizador para iniciar o executável do servidor
activando os parâmetros RemoteServicePort ou RemoteServiceName no firebird.conf(V.1.5 e seguintes)
na configuração daemon (para a versão Classic no POSIX)
inserindo uma entrada no ficheiro de Serviços

Antes de examinar cada uma destas técnicas, será útil observar a lógica usada pelo servidor para definir a porta a auscultar, e pelo cliente para definir a porta a procurar usada pelo servidor para aguardar por pedidos.

Como o servidor define a porta de auscultação

O executável do servidor possui um switch opcional de linha-de-comando (-p) através do qual é possível especificar ou o **número da porta** a auscultar ou o **nome do serviço à porta** que irá efectuar a auscultação. Assim, se o switch for usado, ou o número da porta 3050 ou o nome do serviço (gds_db) será substituído pelo argumento fornecido pelo switch -p.

Em seguida—ou em primeiro lugar, se não existir nenhum switch -p—um servidor v.1.5 verifica o firebird.conf em busca dos parâmetros RemoteServiceName e RemoteServicePort:

- Se ambos estiverem como comentários com "#" então serão assumidos os valores por defeito e nenhuma alteração será efectuada. Qualquer argumento –p, mantém-se e caso não exista serão usados os valor por defeito.

- se RemoteServiceName tiver sido des-comentado, mas não o RemoteServicePort, então o nome do serviço à porta será substituído *apenas* se não tiver já sido sobreposto por um switch -p.

- se RemoteServicePort tiver sido des-comentado, mas não o RemoteServiceName, então a número da porta será *apenas* substituído senão tiver já sido sobreposto por um switch -p.

- se ambos RemoteServicePort e RemoteServiceName tiverem sido des-comentados, então o RemoteServiceName tomará precedência, senão tiverem já sido substituídos por argumentos –p. Se já houver um nome de serviço à porta igual, o valor do RemoteServiceName é ignorado e o valor RemoteServicePort sobrepõe o 3050.

- Neste ponto, se uma sobreposição de quer o número da porta ou nome do serviço tiverem sido assinalados, ambos os servidores v.1.0 e v.1.5 efectuarão uma verificação no ficheiro de Serviços em busca de uma entrada com uma combinação correcta do nome de serviço e do número da porta. Se forem encontradas correspondentes, tudo estará bem. Senão, e o nome de serviço à porta não for gds_db, o servidor lançará uma excepção e não concluirá o arranque. Se o gds_db for o nome do serviço à porta e não puder ser atribuído a qualquer outra porta, será mapeado automaticamente à porta 3050.

Se o nome de serviço à porta tiver sido sobreposto, então irá necessitar de efectuar uma entrada no ficheiro de Serviços—ver o tópico em baixo, *Configurando o ficheiro de serviços*.

Usando o switch -p

Tenha em atenção que este switch também está acessível no Firebird 1.0.x, mas não fora previamente documentado.

Iniciando o servidor com o switch opcional –p dá-lhe a possibilidade de sobrepor o número da porta (3050) ou o nome por defeito do serviço à porta (gds_db) usado pelo servidor para auscultar pedidos das conexões. O switch pode sobrepor um, mas não ambos. Para o Firebird 1.5 e seguintes, pode usar o switch –p em combinação com a configuração no firebird.conf para permitir uma sobreposição do número da porta e do nome do serviço à porta em simultâneo.

Syntaxe para o TCP/IP

```
Comando-servidor <outros switches> -p número-porta | nome-serviço
```

Por exemplo, para iniciar o Superserver como aplicação e sobrepor o nome do serviço gds_db por fb_db:

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

Ou, para substituir a porta 3050 por 3051:

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

Syntaxe para a redirecção em WNet

Numa rede Wnet, substitua o argumento do switch –p na sintaxe em cima pela seguinte sintaxe "barra-barra-ponto-et ":

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

ou

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

Classic em POSIX: o daemon *inetd* ou *xinetd*

Com o servidor Firebird Classic no Linux ou UNIX, o daemon **inetd** ou **xinetd** é configurado para auscultar no porto por defeito e transmitir para o nome de serviço por defeito. O script de instalação irá escrever uma entrada apropriada no ficheiro de configuração /etc/inetd.conf ou /etc/xinetd.conf.

Pode editar a configuração corrente se necessitar. Em seguida demonstra-se um exemplo do que poderá observar no xinetd.conf ou inetd.conf após uma instalação do Firebird Classic no 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
}
```

Se tiver configurado o serviço à porta para ser diferente dos valores por defeito, terá que alterar o xinetd.conf ou inetd.conf em concordância. Reinicie o xinetd (ou inetd) com kill -HUP para ter a certeza que o daemon fará uso da nova configuração.

NOTA Pedidos de conexão fracassarão se ambos o xinetd (ou inetd) e o fbserver (ou ibserver) tentarem auscultar a mesma porta. Se a máquina de alojamento tiver esta dupla configuração, será necessário efectuar modificações por forma a que cada versão de servidor tenha a sua própria porta de serviço.

Usando um parâmetro de configuração em ficheiro

Pode configurar quer o RemoteServiceName quer o RemoteServicePort no firebird.conf para substituir quer o número da porta por defeito (3050) quer o nome de serviço à porta (gds_db) que o servidor usa para auscultar os pedidos de conexão.

O motor do servidor fará uso de um parâmetro RemoteService*, mas não de ambos. Se configurar ambos, ele irá ignorar o RemoteServicePort em todos os casos, excepto quando o servidor tenha sido iniciado com o switch –p impondo uma mudança do nome do serviço. Assim, pode usar o switch –p e o parâmetro RemoteService*, em conjunto, para sobrepor simultaneamente o número da porta e do nome do serviço.

Se o nome por defeito do serviço tiver que ser modificado, então irá precisar de inserir uma entrada no ficheiro de Serviços.

CUIDADO! Se des-comentar o RemoteServiceName ou o RemoteServicePort, mas deixar os valores por defeito inalterados, eles serão tratados como sobreposições. Terão então que ser inseridas entradas explicitas no ficheiro de serviços para as definições por defeito do serviço à porta.

Configurando um cliente para procurar a porta de serviço

Se configurar um servidor com os valores por defeito da instalação (serviço gds_db auscultando na porta 3050) nenhuma configuração será necessária. Se o servidor estiver a auscultar numa porta

diferente ou estiver a usar um nome de serviço à porta diferente, a aplicação do cliente e/ou a sua máquina da alojamento necessitarão de algumas configurações para ajudar a biblioteca do cliente Firebird a encontrar o porto de auscultação.
A string de conexão usada pelo cliente pode incluir informação para procurar a porta de auscultação do servidor de diferentes formas. Os clientes Firebird 1.5 podem opcionalmente usar uma cópia local do firebird.conf. Modificações poderão também ser necessárias no ficheiro de serviços do cliente.

Usando a string de conexão

Se apenas o número da porta ou nome do serviço tiver sido reconfigurado, então inclua o numero da porta ou nome do serviço alternativo na string de conexão. Isto funciona para todas as versões do Firebird.

Syntaxe para conexões TCP/IP

Para se conectar a uma base de dados num servidor de nome alice que transmita na porta 3050 com o nome de serviço fb_db, a string de conexão deverá ser:

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

Ou, se o nome do serviço for gds_db e o número da porta for 3051:
```
alice/3051:/data/teaparty.fdb
```

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

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

Repare que o separador entre o nome do servidor e a porta é uma barra e dois pontos. Os dois pontos antes do nome do caminho físico são também necessários.

Sintaxe para conexões WNet

Numa rede Wnet, use notação estilo-UNC:

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

Se o número configurado da porta ou do nome do serviço tiver sido modificado, então terá que inserir uma entrada no ficheiro de Serviços.

Usando sintaxe da porta com “aliases” de bases de dados

Para estabelecer uma conexão através de uma porta com valores diferentes dos instalados por defeito com um alias de base de dados, ligue o número da porta ou nome do serviço ao nome do servidor e não ao alias. Por exemplo, suponha que o alias da base de dados está definido no aliases.conf como

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

A string de conexão da sua aplicação ao servidor 'alice' seria como:

```
alice/fb_db:rabbit
```

ou
```
alice/3051:rabbit
```

Usando uma cópia do firebird.conf

A partir do Firebird 1.5 e seguintes, pode opcionalmente incluir uma cópia do lado do cliente do firebird.conf na directoria root do firebird e configurar o RemoteServiceName ou RemoteServicePort para ajudar o cliente a localizar a porta do servidor.

- Pode configurar *um* dos dois parâmetros para estender a sobreposição fornecida para o outro através da string de conexão (em cima); ou para sobrepor apenas o RemoteServiceName ou o RemoteServicePort sem ter que usar a string de conexão para tal.

- Se necessitar de evitar passar o nome do serviço da porta ou o número da porta na string de conexão e o servidor está a usar valores diferentes dos instalados por defeito para ambos, pode configurar ambos os RemoteServiceName e RemoteServicePort. Teria que usar esta técnica se o sua aplicação cliente precisasse de manter a capacidade de se conectar aos servidores InterBase ou Firebird 1.0.

Localização dos artefactos do Firebird nos clientes

Quando dependa do firebird.conf nas máquinas clientes, é importante que a biblioteca do cliente saiba onde o procurar. Precisará de configurar uma directoria root do Firebird e informar o sistema da sua localização. Para isso use a variável de ambiente FIREBIRD. Os clientes Windows podem alternativamente usar a chave de registo Firebird. Com uma configuração correcta no cliente, poderá sempre usar uma cópia local do ficheiro de mensagens.

Configurando o ficheiro de serviços

Você não precisa de configurar uma entrada da porta de serviço para o seu servidor ou clientes Firebird se o servidor usar os valores por defeito da instalação, gds_db na porta 3050. Se gds_db for o nome do serviço à porta e não estiver associado com qualquer outra porta, será automaticamente mapeado para a porta 3050.

Se estiver a configurar a porta de serviço para um número de porta ou nome de serviço diferente, ambos o **servidor** e o **cliente** deverão ser explicitamente actualizados por forma a reflectir a reconfiguração. Em ambos os sistemas Linux e Windows, esta informação é armazenada no ficheiro de **serviços**.

Localizando o ficheiro de serviços

- No Windows NT/2000/XP/S2003, este ficheiro está em C:\windows\system32\drivers\etc\services.

- No Windows 95/98/ME, encontra-se em C:\windows\services.

- No Linux/UNIX encontra-se em /etc/services.

Uma entrada no ficheiro de serviços terá a seguinte forma:

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

Abre este ficheiro com um editor de texto e ou adicione uma linha ou edite a entrada existente do gds_db, da seguinte forma:

- Para um servidor ou cliente Firebird 1.0.x, altere ou o nome do serviço ou o número da porta para reflectir a forma como o servidor arranca.

- Para um servidor Firebird 1.5 ou mais recente, edite ou adicione uma linha, como requerido. Se tiver um servidor Firebird 1.0 ou InterBase instalado no mesmo alojador, mantenha as entradas que estes requerem e adicione uma nova entrada para reflectir a forma como o servidor Firebird 1.5 arranca.

Mais Informação

Mais informação sobre o Firebird pode ser obtida em:
http://firebird.sourceforge.net

Ou nos sites afiliados:
http://firebirdsql.org
http://www.ibphoenix.com
http://www.cvalde.com

Se está interessado em participar no desenvolvimento do Firebird, ou se pretende levantar uma discussão sobre algum possível erro, junte-se à lista de desenvolvimento firebird-devel. Para subscrever, envie uma mensagem em branco para:

firebird-devel-request@lists.sourceforge.net

Com a palavra 'subscribe' no campo do assunto.

Por favor não utilize a lista firebird-devel para questões de suporte.

Para suporte técnico, junte-se à lista firebird-support em
http://www.yahoogroups.com/groups/firebird-support

Para suporte do InterClient e desenvolvimento Java, existe a lista:
http://www.yahoogroups.com/groups/Firebird-Java

A lista ib-support abarca questões técnicas sobre o Firebird e o InterBase(R). Por favor não traga as suas questões de Delphi ou outras linguagens de programação para este fórum.

A comunidade "open-source" trabalha com várias listas de discussão no desenvolvimento do Firebird. Por favor refira às Mail Lists e Newsgroups em Firebird community site.

A lista de programadores Firebird e a lista geral da comunidade, assim como outras listas de interesse para programadores Firebird e InterBase, são espelhadas como "newsgroups" em
news://news.atkin.com

Suporte Pago ao Firebird pode ser obtido através do IBPhoenix (os endereços de contacto assim como os números de contactos estão disponíveis em http://www.ibphoenix.com). Alguns membros da equipa Firebird estão também disponíveis para suporte ou consultoria. Por favor contacte-os directamente.

Serviços de Recuperação de Bases de Dados em Firebird ou InterBase podem ser executados por IBPhoenix. Para análise e possível reparação de base de dados corrompidas, executada por si, tente IBSurgeon em IBase.ru (www.ibase.ru)
IBase.ru também providencia serviços de recuperação na Rússia e Europa.

Ofertas para patrocinar melhoramentos ao Firebird devem ser enviados directamente para a Fundação FirebirdSQL Inc. Envie um email para foundation@firebirdsql.org. Se preferir o contacto com os administradores do projecto Firebird – envie um email para firebirds@users.sourceforge.net.

Discussão Geral sobre melhorias ao FB podem ser obtidas na lista Firebird-priorities:
http://www.yahoogroups.com/community/Firebird-priorities.

IB-Architect (http://www.yahoogroups.com/community/ib-architect)é apenas para **discussões técnicas de design** APENAS. Questões de Suporte/conversão não são bem vindas aqui.

Ferramentas e Drivers

Programas Desktop Clientes de Bases de Dados

Existem excelentes escolhas para administradores desktop GUI para o Firebird e estão referenciadas na página Contributed Downloads em http://www.ibphoenix.com. Alguns são open source, alguns são freeware, outros representam propostas comerciais.

O programa IBConsole da Borland não é recomendado como cliente de administração para o Firebird 1.5.

Drivers e Componentes

JAVA: O projecto de driver Jaybird JDBC é desenvolvido activamente no âmbito do projecto Firebird. Foram lançados inicialmente como driver Tipo 4 JDBC/JCA e um driver beta Tipo 2. O Código fonte e os binários podem ser transferidos da páginas de lançamentos Firebird -
http://sourceforge.net/project/showfiles.php?group_id=9028

Para suporte e participação no desenvolvimento e testes, inscreva-se no fórum Firebird-java em http://www.yahoogroups.com/community/firebird-java.

.NET: Firebird possui um projecto driver .NET em desenvolvimento. Código fonte e binários poderão ser transferidos da página de lançamentos do Firebird em -
http://sourceforge.net/project/showfiles.php?group_id=9028

Para suporte e participação no desenvolvimento e testes, inscreva-se no fórum Firebird .NET provider: http://lists.sourceforge.net/lists/listinfo/firebird-net-provider

Delphi e C++Builder: Os utilizadores possuem ao seu dispor duas alternativas, para conexão directa com a API do Firebird, ambas com excelente suporte por parte de programadores e suporte individual:

- Jason Wharton's IB Objects em http://www.ibobjects.com
- FIBPLus em http://www.devrace.com

C++: a biblioteca de acesso C++ freeware 'IBPP' (http://www.ibpp.org), licença MPL, alojada em sourceforge.net, completamente portável entre Win32 e Linux e provavelmente outras plataformas POSIX. É útil quando pretenda acesso do tipo low-level a C-API, mas com um nível de abstracção C++ independente de qualquer ambiente particular de desenvolvimento.

ODBC: A lista dos drivers ODBC pode ser encontrada na página Contributed Downloads http://www.ibphoenix.com. Um observatório do desenvolvimento do driver ODBC/JDBC open source encontra-se neste fórum: http://lists.sourceforge.net/lists/listinfo/firebird-odbc-devel

PHP: Um grupo encontra-se a trabalhar para converter a "velhinha" extensão de Interbase para o standard Firebird. Para informações sobre este projecto, inscreva-se no fórum Firebird-PHP em http://www.yahoogroups.com/community/firebird-php

PYTHON: KInterbasDB é uma extensão Python que implementa o suporte para Firebird de compatibilidade com Python Database API 2.0. Suporte completo nativo da API para o cliente Firebird;

em versões estáveis e em desenvolvimento activo. Licença open source BSD. Downloads e novidades em http://kinterbasdb.sourceforge.net/

Documentação

A documentação do InterBase v 6.0 também se aplica a esta versão do Firebird. Uma versão beta dos manuais do Interbase(tm) 6 em formato Adobe Acrobat pode ser obtida em

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

Um índice da Documentação é mantido no site da comunidade Firebird em

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

Isto é "work-in-progress" e todas as contribuições são bem vindas – envie uma mensagem para firebird-docs@lists.sourceforge.net

Alguns guias de instalação e alguns "HowTos" podem ser obtidos na área de documentação em:
http://www.firebirdsql.org
Ou mais directamente em
http://sourceforge.net/projects/firebird

O principal local para conselhos de utilização e técnicos no site da IBPhoenix -
http://www.ibphoenix.com
IB Phoenix também publica uma subscrição de um CD com alguma assiduidade (artigos avulso também estão disponíveis) o qual contém manuais publicados por eles – Using Firebird e The Firebird Reference Guide.

E alguma documentação adicional pode ser obtida visitando a área técnica da Borland:
http://www.borland.com/techpubs/interbase/

"Correcção de Erros" e Novidades desde a versão 1.0

Tracker #	Descrição	Contribuidor
(sem #)	Corrigidas algumas incoerências nos nomes dos charsets.	P. Jacobi
(sem #)	GSTAT estourava com algumas combinações de opções.	D. Yemanov
(sem #)	Correcção no tratamento de savepoint em BREAK|LEAVE/EXIT.	D. Yemanov
(sem #)	Optimizador corrigido para preferir índices únicos em relação a compostos e preferir índices únicos com semelhança total.	A. Brinkman
(sem #)	Melhoria em ferramentas de instalação do Win32, instsvc.exe e instreg.exe	O. Mascia
Melhoria	Aumento do número máximo de índices por tabela de 64 para (DB_PAGE_SIZE/16)-2	A. Harrison, transportado para 1.5 por N.Samofatov
721792	Uma conexão longa causava um "mem leak" no dispositivo do kernel do SO	N. Samofatov
775003	UDF log(x, y) devolvia log(y, x)	P. Vinkenoog, N. Samofatov
774987	UDFs ltrim(") e rtrim(") devolviam NULL; rtrim esquecia o 1º char	P. Vinkenoog, N. Samofatov
(sem #)	Resolvido o estouro do servidor causado pela perda do contexto da transacção.	A. Peshkoff
(sem #)	Resolvido o estouro do servidor para qualquer combinação de sub-select & between.	A. Peshkoff
736318	"<valor> STARTING WITH <campo>" falha quando se usam índices.	D. Yemanov
(sem #)	Deadlocks não-existentes ocorriam após a execução de triggers pre-(update/delete).	A. Peshkoff
Melhoria	Definir INSERTING/UPDATING/DELETING como palavras não reservadas.	N. Samofatov
Melhoria	Adicionadas novas (mais específicas) mensagens de erro para algumas modificações na v1.5.	D. Yemanov, A. Brinkman, A. Peshkoff
Correcção de Segurança	Adicionada opção –login a instsvc permitindo instalar o serviço FB com uma conta diferente de localsystem.	A. Peshkoff
Melhoria	Reintroduzido o "trimming" dos campos VARCHAR nos protocolos remotos.	D. Yemanov

Tracker #	Descrição	Contribuidor
(sem #)	Estouro aleatório do servidor quando da preparação de queries de grande dimensão.	D. Yemanov
Melhoria	Aperfeiçoamento da Configuração – fazer com que o tratamento do path no firebird.conf esteja em conformidade com os requisitos do SO.	A. Peshkoff
(sem #)	Argumentos da UDF do tipo DATE/TIME (dialecto 3) errados.	Oleg Loa
(sem #)	Possível violação de integridade referencial.	Vlad Horsun, D. Yemanov
745090 e outras questões da instalação RC2	Problema com permissões para firebird.conf (SF #745090). Geração de aliases.conf na instalação; uso de rpmbuild para criar os pacotes Linux	Erik S. LaBianca, N. Samofatov
(sem #)	Permitir fácil ajustamento de LockSemCount em plataformas POSIX. Sem necessidade de usar gds_drop ou reiniciar a máquina para que as modificações sejam assumidas	N. Samofatov
Melhoria	Definir FIRST/SKIP como palavras não reservadas.	N. Samofatov
(sem #)	Referência de conexão errada após exception em PSQL.	A. Peshkoff
(sem #)	Declarações BREAK/LEAVE e EXIT estão agora disponíveis para uso em triggers.	D. Yemanov
(sem #)	Possível corrupção de índices durante a execução de garbage collection.	Vlad Horsun, D. Yemanov
(sem #)	Resolvidos problemas com tratamento de ficheiros temporários: 1) Falha de segurança em todas as plataformas POSIX excepto FREEBSD/OPENBSD relacionada com o uso de mktemp (possíveis ataques DoS ou aumento de privilégios) Só eram gerados 27 nomes de ficheiros únicos em win32 (o que poderia provocar comportamentos inesperados nas versões SS)	N. Samofatov
(sem #)	Alterações no gestor de eventos: desactivado o uso de portas definitivas nas versões CS devido a problemas conhecidos.	D. Yemanov
(sem #)	Permitido o uso de funções agregadas de diferentes contextos dentro de outras funções agregadas. Exemplo: SELECT MAX((SELECT COUNT(*) FROM RDB$RELATIONS)) FROM RDB$RELATIONS	A. Brinkman
(sem #)	Possíveis estouros ao desligar quando era usada a notificação de eventos.	D. Yemanov

Tracker #	Descrição	Contribuidor
(sem #)	Modificações no gestor de Serviços: funcionalidades de GSTAT/GSEC não são acessíveis via Serviços API na win32 CS (até à v1.6).	D. Yemanov
(sem #)	Estatísticas erradas de registo são reportadas quando a operação falha por algum motivo.	D. Yemanov
(sem #)	stdin/stdout não pode ser usado para redireccionar a E/S para a consola, na versão win32 do GBAK.	A. Peshkoff
(sem #)	Redimensionamento da tabela de locks com problemas em CS. Resolvido o "lock manager out of room" (Win32 CS 1.5 RC1) ou estouros (possível em todas as outras versões CS do Interbase/Firebird).	N. Samofatov
Melhoria	Aperfeiçoamento do INTL: permitir que a função UPPER funcione para WIN1251 charset sem collations explícitos.	N. Samofatov, D. Yemanov
BUGCHECK(291)	Possível corrupção da base de dados quando era modificado/eliminado o mesmo registo num pre-trigger para o qual este trigger fora chamado.	A. Peshkoff
(sem #)	Buffer excedido na chamada a isc_database_info().	Oleg Loa
(sem #)	Modificação do Gestor de Configuração: o servidor termina na ausência /erro do firebird.conf gerando um relatório erros no registo de sistema.	A. Peshkoff
(sem #)	Correcção nos Serviços API: activadas as estatísticas dos Serviços API para plataformas POSIX versão CS.	N. Samofatov
(sem #)	Modificações no parser. 1) ROWS_AFFECTED foi renomeado para ROW_COUNT 2) CONNECTION_ID/TRANSACTION_ID foi renomeado para CURRENT_CONNECTION/CURRENT_TRANSACTION 3) Alguns dos novos símbolos foram definidos como não-reservados	D. Yemanov
(sem #)	Correcção nos Serviços API: serviços API parcialmente activados para as versões CS win32.	D. Yemanov
(sem #)	Entrega errada de eventos (uso desnecessário de pacotes OOB).	Jim Starkey, Paul Reeves
(sem #)	Melhoria no tratamento de locks: os deadlocks são agora detectados e reportados logo que sejam recebidas todas as notificações dos processos em bloqueio, i.e. quase instantaneamente na maior parte dos casos	N. Samofatov
(sem #)	O Servidor estourava nalgumas operações de Serviços API.	A. Brinkman
(sem #)	Capacidades de segurança avançadas: implementado o acesso configurável a bases de dados, tabelas externas e bibliotecas UDF.	A. Peshkoff
(sem #)	Resolvidas algumas perdas de memória/recursos.	Mike Nordell, A. Peshkoff, N. Samofatov,D. Yemanov

Tracker #	Descrição	Contribuidor
(sem #)	Buffer excedido com arrays multidimensionais.	D. Yemanov
213460, 678718	Várias questões com eventos usados em servidores com vários endereços IP. NOTA: Agora é possível configurar uma determinada porta para o processamento de eventos.	D. Yemanov
(sem #)	Resolvidas algumas perdas de recursos.	Mike Nordell, A. Peshkoff
(sem #)	Corrigidos os Serviços API: activados os serviços API para plataformas POSIX nas versões CS. Notas: 1. Modificações apropriadas no Win32 CS ainda não estão prontas 2. Serviço de Backup/restore foi corrigido, testado e deverá funcionar 3. Validação da Base de dados foi parcialmente corrigida e deverá funcionar 4. Outros serviços provavelmente ainda não funcionarão nas versões CS	N. Samofatov
(sem #)	Melhorias SQL: permitir NULLs em unique constraints e índices (SQL-99 spec).	D. Yemanov, N. Samofatov
(sem #)	Melhoria de Desempenho: o log de desfazer VIO agora usa uma árvore B+ para armazenar os dados de registos dos savepoint. O que permitiu melhorar um pouco o desempenho ao efectuar múltiplas modificações do registo numa única transacção (qualquer coisa como 2-3X para 100000 registos).	N. Samofatov
(sem #)	Corrupção da Base de Dados ao efectuar uma salvaguarda do savepoint após grande número de operações DML (por forma a libertar o savepoint da transacção) e o registo fora modificado fora do savepoint e eliminado dentro do savepoint.	N. Samofatov
(sem #)	Melhorado o EXECUTE STATEMENT. Agora é possível devolver valores a partir do SQL dinâmico. Sintaxe: EXECUTE STATEMENT <valor> INTO <lista_variáveis>; (um só registo) ou FOR EXECUTE STATEMENT <valor> INTO <lista_var> DO <lista_declarações>;	A. Peshkoff
(sem #)	O servidor bloqueava durante a desconexão após modificações massivas.	D. Yemanov
(sem #)	Melhorias no optimizador: Subselects na cláusula SET do UPDATE agora podem usar índices.	A. Brinkman

Tracker #	Descrição	Contribuidor
(sem #)	Erro "Context already in use" no caso de DISTINCT com subqueries.	A. Brinkman
(sem #)	Melhorias nas capacidades de isc_database_info: lista das transacções actualmente activas é agora acessível via chamada a isc_database_info.	N. Samofatov
(sem #)	Melhoria de Desempenho: avaliação de atalho do booleano. NOTA: o comportamento é controlado pela opção "CompleteBooleanEvaluation" do firebird.conf. Por defeito é 0 (avaliação de atalho).	Mike Nordell
(Beta 2 erro)	Estouro da Pilha Interna durante a preparação da declaração.	D. Yemanov, Mike Nordell
(sem #)	Melhoria de Desempenho para a arquitectura IA32 CPU: aumento de velocidade para as operações com índices	Mike Nordell
(sem #)	Mudança nos triggers universais: permitido o acesso aos contextos (OLD e NEW) nos triggers universais.	D. Yemanov
(sem #)	Melhorias no Optimizador: quando um nó-semelhante e outros nós (geq, leq, between...) estão disponíveis para a obtenção por índice, então é usado o nó-semelhante em vez dos outros.	A. Brinkman
(sem #)	Longos atrasos durante a conexão/desconexão no WinXP.	A. Brinkman
(sem #)	Limpeza genérica: removido muito do código não usado.	Blas Rodriguez Somoza, Erik Kunze
523589	A View afectava o resultado de um query. Comentário: O problema era que RSE's (dentro de um view) não era sinalizado como variante.	A. Brinkman
(sem #)	Modificado o comportamento do modo "forced writes": agora, se FW=off (desactivado), você pode controlar quão frequentemente as "dirty pages" são gravadas para o disco (permite maior fiabilidade quando o FW está desactivado nas plataformas Win32).	Blas Rodriguez Somoza
(sem #)	A base de dados de segurança foi renomeada para security.fdb.	D. Yemanov
(sem #)	Novo ficheiro de configuração: firebird.conf foi finalmente publicado.	D. Yemanov
(sem #)	Novas funções definidas-pelo-utilizador LPAD e RPAD adicionadas para a biblioteca IB_UDF.	Juan Guerrero
(sem #)	Por vezes GFIX não permitia usar as opções "-user" e "-password" (erro "incompatible swiches").	D. Yemanov
(sem #)	Cache da conexão à BD de segurança: conexão à base de dados de segurança é agora colocada na cache, permitindo assim diminuir o tempo para as conexões seguintes à base de dados.	D. Yemanov

Tracker #	Descrição	Contribuidor
Melhorias	1. A Memória usada pelo servidor foi reduzida. 2. E/S externa directa quando não existe memória disponível para ordenação. Aumentado o número de streams e predicados suportados pelo optimizador.	D. Yemanov
508594	LEFT JOIN com VIEWs: simples LEFT JOIN num VIEW com apenas uma cláusula ON não usava um índice quando era isso era possível.	A. Brinkman
(sem #)	Novos character sets: 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: Collations para os charsets acima não estão ainda disponíveis.	Blas Rodriguez Somoza
(sem #)	Modificações em CREATE VIEW: proibida a sub cláusula PLAN.	D. Yemanov
(sem #)	Alterado o comportamento do rastreamento de agregados -- introduzida a compatibilidade para traz dentro dos agregados. O campo mais profundo dentro do agregado determina onde um contexto de agregado pertence.	A. Brinkman
(sem #)	Melhorias no optimizador: melhores optimizações de queries JOIN "complexos" (LEFT JOIN, views, SPs, etc).	A. Brinkman
(alpha 5 erro)	As maiores perdas de memória foram resolvidas.	D. Yemanov
(sem #)	Novas funções API: funções IB7-compliant devolviam versão da biblioteca cliente -- isc_get_client_version(), isc_get_client_major_version(), isc_get_client_minor_version()	D. Yemanov
(sem #)	Melhorias no Sort/merge: merging (planos SORT MERGE) é agora feito pelo módulo de ordenação em-memória.	D. Yemanov
(sem #)	Novo gestor de memória interno foi modificado para permitir um melhor desempenho.	N. Samofatov
(sem #)	Alterações no build Win32: 1. Mudados os nomes dos objectos USER32 para permitir correr o servidor ao mesmo tempo do que o IB/FB1. 2. Nome do Mapa para o protocolo local (IPC) foi alterado, como tal a biblioteca cliente da v1.5 já não é compatível com as versões prévias via IPC. 3. Todos os nomes de protocolos de transporte (porta e serviço INET, pipe WNET, mapa IPC) são agora configuráveis via firebird.conf.	D. Yemanov
(sem #)	RDB$FIELD_LENGTH para views que continham concatenações de campos long CHAR/VARCHAR, assumia valores incorrectos.	D. Yemanov

Tracker #	Descrição	Contribuidor

Melhoria	Melhoria nos Triggers: adicionadas acções de validação em runtime (predicados INSERTING/UPDATING/DELETING).
	
	Exemplo:
	
	if (INSERTING) then
	
	new.OPER_TYPE = 'I';
	
	else
	
	new.OPER_TYPE = 'U';
		D. Yemanov

(sem #)	Os cursors (cláusula WHERE CURRENT OF) não podiam ser usados em triggers.	D. Yemanov
221921	ORDER BY não tinha qualquer efeito.	A. Brinkman
213859	Subquery ligado a cláusula com 'IN'.	A. Brinkman
Melhoria	Permitidas as expressões arbitrárias na cláusula ORDER BY.	N. Samofatov
(sem #)	O "engine" estourava quando eram usadas UNIONS numa VIEW e essa VIEW era usada numa cláusula WHERE dentro de um subquery.	A. Brinkman
(sem #)	Limpeza genérica de código: estruturas dentro de Y-valve.	A. Peshkoff, N. Samofatov
Melhoria	Comentários de Linha-Única (--) são agora permitidos em qualquer posição da declaração SQL.	D. Yemanov
(sem #)	"Request sycnhronization error" com a declaração BREAK.	D. Yemanov
625899	Bugcheck 291.	A. Peshkoff
(sem #)	Alteração PSQL: EXECUTE VARCHAR foi renomeado para EXECUTE STATEMENT.	A. Peshkoff
521952	"No current record for fetch operation"-nenhum registo encontrado para leitura.	A. Brinkman
(sem #)	QLI não compreendia o tipo de dados BIGINT.	D. Yemanov
(sem #)	Tamanho de variáveis de texto dentro de procs/triggers não era copiado para a estrutura do descritor.	A. Brinkman
(sem #)	Alterações FIRST/SKIP e ORDER BY -- 1. Implementada a cláusula ORDER BY em subqueries. 2. Proibido FIRST/SKIP para views. 3. Permitido o valor zero como argumento válido para FIRST.	D. Yemanov

Tracker #	Descrição	Contribuidor
(sem #)	Buffer excedido (MAXPATHLEN) e nome local da função directório reescrito.	Erik Kunze
(sem #)	Foi modificado o mapeamento de parâmetros do SQLDA para ser consistente com a ordem e número de parâmetros da string SQL de origem. NOTA: Você pode activar o comportamento anterior de mapeamento (por motivo de retro-compatibilidade) usando o parâmetro do gestor de configuração "OldParameterOrdering".	N. Samofatov
Melhoria	Melhorias no Optimizador: deixar que os subqueries também usem índices quando o seu parente for uma stored procedure.	A. Brinkman
(sem #)	Removida a limitação de tamanho do pedido.	D. Yemanov
(sem #)	Tratamento de Nulls first/last e collation na cláusula "order by" das unions	N. Samofatov
(sem #)	Limpeza genérica de código; renomeações, novas “safe macros”, suporte para mingw.	Erik Kunze, Ignacio J. Ortega, D. Sibiryakov
(sem #)	Implementação de “Explicit record locking” finalizada. Deverá estar estável e consistente.	N. Samofatov
Melhoria	Melhoria no Optimizador: melhor tratamento dos nós AND dentro dos nós OR.	A. Brinkman
(sem #)	Excepções dentro de um ciclo for/while em triggers não eram tratados correctamente.	A. Peshkoff
623992	Duplas barras na string de conexão.	Paul Reeves, Mark O’Donohue
(sem #)	Deadlock durante algumas operações de base de dados.	A. Peshkoff
Melhoria	Melhoria no Optimizador: se alguns índices com grande diferença de selectividade puderem ser usados para obtenção de dados, apenas os melhores deles serão usados enquanto os outros serão ignorados.	D. Yemanov
(sem #)	Problema com identificadores com aspas em expressões de “plan”.	N. Samofatov
Melhoria	A arquitectura CS é agora suportada no Win32, mas ainda não pode ser considerada estável, assim toda a informação será bem vinda.	D. Yemanov
(sem #)	As Stored procedures já não são recompiladas antes de serem apagadas.	N. Samofatov
(sem #)	Novo collation para WIN1251 charset: WIN1251_UA para a linguagem Ucraniana e Russa.	D. Yemanov
(sem #)	Alteração na biblioteca cliente: as rotinas API já não são mais exportadas por ordinals.	D. Yemanov

Tracker #	Descrição	Contribuidor
Melhoria	Novo gestor de configuração: activada a mesma configuração baseada em ficheiro de texto para todas as plataformas suportadas.	D. Yemanov
Melhoria	Melhoria no Optimizador: foi acrescentado melhor suporte para usar índices com "OR". E obter o melhor índice composto de todos os nós "AND".	A. Brinkman
Melhoria	Adicionado o suporte para tratamento de "explicit savepoint" em DSQL.	N. Samofatov
(sem #)	Limpeza de Protocolo: O protocolo de rede IPX/SPX já não é suportado.	Sean Leyne
(sem #)	Limpeza de plataformas obsoletas: algumas plataformas já não eram suportadas pelo presente código. DELTA, IMP, DG_X86, M88K, UNIXWARE, Ultrix, NeXT, ALPHA_NT, DGUX, MPE/XL, DecOSF, SGI, HP700, Netware, MSDOS, SUN3_3	Sean Leyne
Melhoria	Melhoria no Optimizador: adicionado o suporte para a detecção do uso de índice com sub-selects em select agregado.	A. Brinkman
Melhoria	Melhoria no "thread scheduler" para Win32 SS: agora o servidor deverá ser mais rápido na resposta sob "carga pesada".	A. Peshkoff
Melhoria	Adicionado suporte para locking explicito. O comportamento do Wait nos modos de transacção isc_tpb_wait ainda não está estável. Sintaxe: SELECT <...> [FOR UPDATE [OF col [, col ...]] [WITH LOCK]]	N. Samofatov
558364	Os Triggers fracassavam na tentativa de compilar se o PLAN fosse usado.	Ignacio J. Ortega
(sem #)	Uma Transacção Distribuída (2PC) não podia ser "rolled back" devido a erros de rede.	Vlad Horsun, Erik Kunze
(sem #)	Limpeza genérica: ISC_STATUS_LENGTH e macros MAXPATHLEN.	Erik Kunze
496784	Quando o optimizador encontra índices para o LEFT JOIN, comporta-se como o INNER JOIN. Resolvido o problema que fazia com que "outer joins" complexos produzissem resultados errados.	N. Samofatov
(sem #)	O subtipo BLOB é ignorado em domínios de sistemas gerados para campos de expressão em views.	D. Yemanov
(sem #)	Erro de instalação resolvido: instreg.exe não criava o valor de registo "GuardianOptions".	D. Yemanov
(sem #)	Perda de Recursos no tratamento recursivo de procedimentos DDL o que fazia com que alguns DDL fracassassem.	N. Samofatov
(sem #)	Check constraint que usem apenas um campo de tabela são agora eliminadas automaticamente quando este campo é eliminado.	N. Samofatov

Tracker #	Descrição	Contribuidor
451927	Novas variáveis de sistema ROWS_AFFECTED em PSQL: devolve o número de linhas afectadas pela última declaração INSERT/UPDATE/DELETE. Para quaisquer outras declarações além de INSERT/UPDATE/DELETE, o resultado será sempre zero.	D. Yemanov
446240	Mensagens de exceptions dinâmicas: é permitido despoletar uma exception com uma mensagem diferente daquela com que a exception foi criada. Sintaxe: EXCEPTION nome [valor];	D. Yemanov
547383	Novas variáveis de sistema SQLCODE e GDSCODE fornecem acesso ao erro ocorrido dentro dum bloco-WHEN em PSQL. Fora do bloco-WHEN, devolve 0 (sucesso).	D. Yemanov
(no #)	Semântica de reinicialização da Exception: permite que uma exception já tratada em PSQL seja relançada num bloco WHEN. Sintaxe: EXCEPTION; Nenhum efeito fora do bloco-WHEN.	"Digitman"
(no #)	O servidor estourava durante a garbage collection sob carga intensa.	N. Samofatov
Melhoria	Compilação de metadata deferida: resolvido grande número de causas do conhecido erro; "objecto em uso".	N. Samofatov
Melhoria	Novo tratamento da ordenação NULL: permitir ordenação de NULLs definida pelo utilizador.	N. Samofatov
(no #)	Gstat mostrava valor errado para elemento maxdup.	D. Kuzmenko
(no #)	É usada uma nova chave de registo no win32: SOFTWARE\FirebirdSQL\Firebird.	--
451925	Nomes de índices de restrição definidos pelo utilizador: é permitido usar nomes definidos pelo utilizador ou o nome da restrição, para índices que imponham restrições.	D. Yemanov
Melhoria	Nova declaração RECREATE VIEW: substituto para o par de declarações DROP VIEW / CREATE VIEW. Sintaxe: RECREATE VIEW nome <definição_view>;	D. Yemanov
(no #)	Trigger cujo nome começa por 'RDB$' não pode ser alterado ou eliminado.	D. Yemanov

Tracker #	Descrição	Contribuidor
(no #)	Ficheiros da distribuição renomeados de acordo com o nome Firebird. Agora são fbserver, fbclient, firebird.msg etc. A nova biblioteca cliente é o fbclient e deverá ser usada pelos novos projectos baseados no FB. gds32 apenas redirecciona para o fbclient e é fornecido por razões de retro-compatibilidade.	Vários
(Pequeno incremento no ODS)	Adicionados novos índices de sistema (RDB$INDEX_41, RDB$INDEX_42, RDB$INDEX_43), agora a versão do ODS é a 10.1.	D. Yemanov, N. Samofatov
451935	Nova declaração CREATE OR ALTER para triggers e strored procedures, permitindo criar ou alterar um objecto da base de dados dependendo de este existir ou não. Sintaxe: CREATE OR ALTER nome <definição_objecto>;	D. Yemanov
(no #)	Apareciam dependências inválidas na base de dados (tal como DB$34) após alterações de metadata.	D. Yemanov
(no #)	Declaração de variáveis locais melhorada: simplificada a sintaxe e permitido declarar e definir variáveis ao mesmo tempo. Sintaxe: DECLARE [VARIABLE] nome <tipo_de_variável> [{'=' | DEFAULT} valor]; Exemplo: DECLARE my_var INTEGER = 123;	Claudio Valderrama
(no #)	Desactivada a declaração BREAK para triggers (como EXIT) devido a limitações internas conhecidas.	D. Yemanov
555839, 546274	Agrupamento melhorado: permite usar GROUP BY para funções internas e subqueries, bem como para valores ordinais (isto é posições de colunas, ou seja o grau da coluna no conjunto de dados devolvidos).	A. Brinkman
451917	Nova função interna COALESCE permite que uma coluna possa ser calculada por mais do que uma expressão, em que a primeira expressão que devolva um valor não NULL é usada para devolver o valor da coluna.	A. Brinkman
451917	Nova função interna NULLIF para sub-expressões devolve NULL se for igual a um valor específico, senão devolve o valor da sub—expressão.	A. Brinkman
451917	Nova função interna CASE permite que o resultado de uma coluna possa ser determinado através do resultado de uma expressão do tipo “case”.	A. Brinkman
545725	Sweep automático em segundo plano bloqueava.	A. Peshkoff
(no #)	O servidor estourava quando estruturas XSQLDA não eram preparadas para todos os parâmetros da declaração.	D. Yemanov
(no #)	PSQL: suporte activado para blocos BEGIN...END vazios.	D. Yemanov

Tracker #	Descrição	Contribuidor
567931	Resolvido parcialmente um problema de segurança do metadata.	D. Yemanov
(no #)	BigInt arrays não funcionavam.	Artem Petkevych
437859	Implementada execute procedure e concatenação de string, visando permitir que qualquer expressão possa ser usada como parâmetro de SP.	D. Yemanov
562417	Concatenação de caracteres vazios para Agregados.	D. Yemanov
Melhoria	Suporte de Readline (histórico de comandos) para ISQL.	M. O'Donohue
446206	Novo tipo de dados BIGINT que permite o uso SQL nativo de números exactos de 64-bit (Apenas para Dialecto 3).	D. Yemanov
451922	Permitir que um Trigger universal seja disparado para um número de tipos de acções.	D. Yemanov
446238, 446243	Novo CONNECTION_ID e TRANSACTION_ID de sistema acessível em PSQL. Devolve identificador interno apropriado armazenado na página de cabeçalho da base de dados.	D. Yemanov
446180	Alias de Base de Dados no servidor: ligue-se a qualquer base de dados usando o "alias" em vez da localização física da mesma. A lista de aliases, de base de dados, reconhecidos está armazenada no ficheiro aliases.conf localizado na raiz da instalação do servidor. Exemplo: Entrada alias no ficheiro de configuração: minha_base_dados = c:\dbs\my\database.gdb Texto para ligação à BD na aplicação: localhost:minha_base_dados	D. Yemanov
(no #)	Novo gestor de plugin e interface INTL.	John Bellardo
Melhoria	Ordenação em memória: se o plano SORT for usado numa declaração SQL, a ordenação é processada em memória. Se não houver memória disponível para esta operação, recorre ao método anterior com utilização de ficheiro temporário.	D. Yemanov
538201	Estouro com extracção de nulo como data.	Claudio Valderrama
446256	Nova extensão da declaração PSQL EXECUTE VARCHAR permite a execução de declarações de SQL dinâmico em SPs/triggers. (Posteriormente renomeado para EXECUTE STATEMENT).	A. Peshkoff
(no #)	Profunda limpeza e revisão do código.	Sean Leyne, Erik Kunze
(no #)	Novo gestor de memória.	John Bellardo
(no #)	Nova lógica de tratamento de excepções.	Mike Nordell, John Bellardo
(no #)	Nova configuração de compilação baseada no autoconf.	John Bellardo, M. O'Donohue,

		Erik Kunze
(no #)	A transferência de código C para C++.	Mike Nordell, John Bellardo, M. O'Donohue

Equipa de Tradução: Artur Anjos
Carlos Mação