Firebird™ Versão 1.5

Notas da Versão v.1.5

08 de Fevereiro de 2004 – Versão 1.08

Conteúdo

Notas Gerais

O Servidor de Banco de dados Firebird™ foi desenvolvido por um grupo independente de programadores a partir do código fonte original do InterBase™ que foi disponibilizado pela Borland sob a "InterBase Public License v.1.0", em 25 de Julho de 2000.

O desenvolvimento do código do Firebird 2 começou antes mesmo do término do desenvolvimento da primeira versão do Firebird 1, com a adaptação do código original de C para C++ e a primeira limpeza geral. O Firebird 1.5 é o primeiro lançamento do código do Firebird 2. É um passo significativo para os programadores e para todo o projeto Firebird, mas ainda não é um fim. Simultaneamente ao

lançamento da versão 1.5, novas alterações estão sendo continuamente efetuadas para o desenvolvimento do Firebird 2.

A manutenção do Firebird 1.0.x prossegue normalmente – as correções de erros e melhorias importantes do Firebird 1.5 foram transportadas para a versão 1.0.

Os Executáveis do Firebird 1.5

Os executáveis do Firebird podem ser obtidos no site da Web:
http://sourceforge.net/projet/showfiles.php?group_id=9028

Descrições de Versão do Firebird 1.5

Win32: "WI-V1.5.0.nnnn Firebird 1.5"
Linux: "LI-V1.5.0.nnnn Firebird 1.5"
E assim por adiante, sendo nnnn o número da compilação.

Veja a Seção de Documentação para obter informação sobre a documentação recomendada.

Novas Funcionalidades

Novo código, melhor otimização

Esta versão foi desenvolvida a partir da transposição do código original em C para C++, um processo iniciado em 2000 por Mike Nordell. Além do processo contínuo de revisão, limpeza e remoção de erros, foi desenvolvido um novo gerenciador de memória, e foram introduzidas novas funções na linguagem. Ainda durante o desenvolvimento da versão 1.5, o "SQL Query Optimizer" recebeu uma série de melhorias e correções, implementadas por Arno Brickman e outros, resultando em ganhos de performance de 30 a 60 % e mesmo mais em alguns casos.

Arquitetura

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

- Não existia uma versão do modelo "Classic" para windows há mais de 8 anos. Este versão pode utilizar múltiplos processadores, algo que ainda não é suportado de forma adequada na versão "SuperServer" para Windows. Embora utilizável, a versão do modo "Classic" deve ser vista ainda como experimental.

- "Embedded Server" (ou servidor embutido) é uma DLL que integra um cliente com uma única conexão e um servidor "SuperServer" para permitir a construção rápida e eficiente de aplicações mono usuário e transportáveis.

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

Módulos Instalados e Segurança

Se você está familiarizado com o Firebird 1.0.x, vai notar várias diferenças nos nomes dos módulos e nas regras para o seu acesso e localização. Em seguida destacaremos algumas destas mudanças, mas para informações detalhadas sobre a instalação, estrutura em disco e configuração refira-se as respectivas seções.

1. A maior parte dos módulos e constantes mudaram de nome. Na maior parte dos casos, os novos nomes possuem alguma variante da palavra "firebird" ou do prefixo "fb". Por exemplo, a biblioteca API faz agora parte da biblioteca compartilhada "fbclient.dll" no Windows e "libfbclient.so" nas outras plataformas. A exceção a esta regra é o banco de dados de segurança, que antigamente chamava-se "isc4.gdb", e agora foi renomeado para "security.fdb".
2. Os arquivos externos utilizados pelo servidor (bibliotecas UDF, filtros BLOBs, bibliotecas de set de caracteres, tabelas externas) estão agora sujeitas a níveis de proteção no sistema de arquivos que, em alguns casos, são por padrão diferentes dos utilizados nas versões 1.0.x e pelo Interbase.
3. O novo arquivo de configuração do servidor firebird.conf substitui os anteriores ibconfig (Windows) e isc_config (outras plataformas), e contém mais opções de configuração, além de uma nova organização e documentação integrada.
4. Possibilidade de criar apelidos para as bases de dados na versão 1.5. Agora é permitido, opcionalmente, ocultar a localização de uma base de dados atribuindo-lhe um apelido, ou "alias". A localização original dos arquivos encontra-se em outro arquivo, aliases.conf. O principal objetivo desta função é proteger a localização física dos arquivos de um "sniffer" que espione a rede.
5. Por padrão (e por prática) nos Servidores Windows utiliza-se o usuário "local system" para executar o programa que instala o Firebird como serviço na inicialização do sistema. Isto pode ser uma séria vulnerabilidade de segurança, pois deixa uma porta aberta para acesso a toda a máquina. A versão 1.5 do instalador (instsvc.exe) aceita usuários específicos do windows para a instalação do serviço. É altamente recomendável que se crie um usuário Firebird para este propósito, e que se utilize esta nova funcionalidade se o seu servidor estiver de alguma forma conectado à internet.

"Trimming" de Campos Varchar em protocolos remotos

Foi terminado o trabalho de implementar 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 quem solicita ao servidor a execução do "trim" dos varchars, o cliente Firebird 1.5 (fbclient.dll ou libfbclient.so) fará o trim, mesmo quando conectado a um servidor de versão anterior à 1.5. Uma versão anterior do cliente não obterá o "trimming", mesmo que conectado a um servidor 1.5 ou posterior.

Semântica de Triggers para Múltiplas ações

Agora você pode escrever em uma única "Before" ou "After" Trigger ações condicionais para as diferentes operações de tabela : "insert", "update" ou "delete". Esta nova semântica facilita a elaboração e a manutenção de Triggers sem eliminar a possibilidade de estabelecer múltiplos triggers para cada fase.

Melhoria na nomenclatura de "constraints"

Os índices que forçam a integridade podem agora possuir um nome definido pelo usuário.
Atenção ! Se utilizada esta funcionalidade, o banco de dados não será mais compatível com a v.1.0.x ou com o InterBase®.

Número máximo de índices por Tabela foi 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 for preciso aplicar um bloqueio pessimista, esta versão adiciona uma nova sintaxe para impor um bloqueio de leitura em linhas enquanto elas são manipuladas pelo cliente. Use este recurso com muito cuidado.

"Cache de Conexão" à base de dados de Segurança

A conexão à base de dados de segurança é mantida em cache na versão SS. Isto é, o security.fdb é carregado em memória quando a primeira conexão é feita, e é mantido em memória enquanto houver conexões clientes ativas.

Melhores mensagens de erro

Sempre que possível, as mensagens que reportam erros de SQL são agora mais detalhadas. É importante ressaltar que podem ocorrer mensagens estranhas se forem utilizados arquivos interbase.msg ou firebird.msg de versões anteriores.

Serviços API na versão Classic do Linux

Está disponível o suporte limitado para os Serviços API para a versão do Classic Server em Linux. Os Serviços disponíveis são os de gbak (backup/restore) e gfix (validar base de dados, shutdown/online, etc). Os demais (gstat, registros do servidor, etc.) não foram testados e provavelmente permanecem não funcionais.

Alterações nas bibliotecas do Cliente

Clientes Windows

A biblioteca cliente foi renomeada para fbclient.dll. Todos os utilitários do servidor (gbak, gfix, etc) utilizam apenas esta biblioteca. Conecte as novas aplicações diretamente pela fbclient.dll, sem necessidade da gds32.dll (Recomendado).
Para manter a compatibilidade com as aplicações existentes, é possível gerar uma cópia da biblioteca fbclient.dll com o nome de gds32.dll usando o novo utilitário 'instclient.exe'. Para instruções mais detalhadas veja a seção de instalação e as notas mais recentes da distribuição Windows do Firebird.

Clientes Linux

A biblioteca cliente do Super Server 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.

Arquivos e Módulos renomeados

Plataforma	Módulo	Firebird 1.0	Firebird 1.5	Notas especiais
Todas	Variáveis de Ambiente	INTERBASE	FIREBIRD	Diretório raiz de instalação
		INTERBASE_LOCK	FIREBIRD_LOCK	Local do arquivo de Lock
		INTERBASE_MSG	FIREBIRD_MSG	Local do arquivo de Mensagem
		INTERBASE_TMP	FIREBIRD_TMP	Diretório utilizado como espaço para Sort

Plataforma	Módulo	Firebird 1.0	Firebird 1.5	Notas especiais
Todas	Base de Dados de Segurança	Isc4.gdb	security.fdb	
Todas	Arquivo de Mensagens	Interbase.msg	firebird.msg	
Todas	Arquivo de Log	interbase.log	firebird.log	
Todas	Versão de ODS	10	10.1	Novo ODS (10.1). Não provoca nenhuma incompatibilidade com versões anteriores de ODS mas a versão mesmo assim não é atualizada automaticamente. Tanto o Firebird 1.0 como o 1.5 suportam 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.
Linux	Binário do Classic server	Gds_inet_server	fb_inet_server	
Linux	Gerenciador de lock Classic	ib_lock_mgr	fb_lock_mgr	
Linux	Controle Superserver	ibmgr.bin	fbmgr.bin	
Linux	Binário Superserver	ibserver	fbserver	
Linux	Arquivo 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 (mono usuário, não "thread-safe") para Classic
Linux	Symlink para Biblioteca cliente para Compatibilida-de	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.

Plataforma	Módulo	Firebird 1.0	Firebird 1.5	Notas especiais
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 a seguir sobre compatibilidade do gds32.dll para aplicações antigas.
Windows	Biblioteca Cliente para compatibilida-de	N/A	gds32.dll	
Windows	Arquivo de Configuração	ibconfig	firebird.conf	
Windows	Local IPC port	InterBaseAPI	FirebirdAPI	Com as configurações padrões não é possível estabelecer conexões locais com a versão anterior da biblioteca cliente (gds32.dll). Se necessário, o servidor pode ser configurado para usar o nome antigo do mapa IPC, via firebird.conf.
Windows	"Registry key" Padrão	HKLM\SOFTWARE\Borland\InterBase	HKLM\SOFTWARE\Firebird Project\Firebird Server\Instances	O diretório é armazenado no parâmetro "DefaultInstance". i.e., não existe mais a chave "CurrentVersion", e a chave "RootDiretory" foi substituída por "DefaultInstance".
No Windows os novos nomes dos serviços são "Firebird Guardian – DefaultInstance" e "Firebird Server – DefaultInstance".				

Compatibilidade

Estrutura Em Disco (On-Disk structure - ODS)

A Estrutura Em Disco do Firebird 1.5 foi designada como 10.1. Esta pequena atualização do ODS foi necessária pelos seguintes motivos :

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

Algumas outras novas funcionalidades que requerem alterações na ODS foram adiadas para a versão 2. Até lá, será possível transportar diretamente bases de dados Firebird 1.0.x. Tenha uma cópia testada das bases Firebird 1.0.x antes de transferi-las para um servidor 1.5.

Base de Dados InterBase™

Caso queira "brincar" com o Firebird usando uma base de dados InterBase, e pretenda reverter a base para o Interbase mais tarde, 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 para backup e restore.

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

Local e Nomes de Arquivo

Nesta versão, uma quantidade substancial de arquivos foi renomeada, fruto do trabalho da substituição de nomes herdados pelo InterBase® 6. Leia a secção de Local e Nomes de Arquivos para obter as descrições e algumas recomendações.

Execução Simultânea de Servidores

As alterações efetuadas nos nomes dos objetos de sistema permitem que o Firebird 1.5 seja instalado e utilizado numa máquina que possua o InterBase ou o Firebird 1.0.x instalado. No Windows, o FB 1.5 ainda usa outra chave de registro. Se o servidor for configurado para usar outras portas de rede, é possível executar várias instâncias do servidor simultaneamente, ou executar a versão 1.5 ao concorrentemente com 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 for feito um "downgrade" de uma base v.1.5 para v.1.0.x. Concretamente, caso sejam criadas chaves primárias, únicas ou estrangeiras como "constraints", os valores padrões dos nomes dos índices serão incompatíveis com a v.1.0.x. Serão publicados arquivos README detalhando estes casos, na medida em que eles ocorrerem.

Compatibilidades em Linux

Devido a um histórico de problemas envolvendo o compilador GNU C++, as versões de Linux do Firebird 1.5 requerem 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 instalar e

executar a versão 1.5. O quadro abaixo pode ser usado como referência. Porém, agradecemos toda a informações que possam ser fornecidas. Por favor, compartilhe 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 inteiro de 64 bits, compatível SQL99, sinalizado e de escala zero. Disponível apenas no dialeto 3.

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

META DADOS

(1.5) Melhorias a "named constraints"

Dmitry Yemanov

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

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 que :
a) um índice gerado automaticamente pelo sistema receber automaticamente como identificador o mesmo nome que a "constraint" que ele força.
b) um índice que força uma "constraint", com ou sem nome, possua ele mesmo um nome atribuído por um identificador e opcionalmente ser construído em ordem descendente.
NOTA: Não é ainda possível utilizar um índice pré-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 Ações

Dmitry Yemanov

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

Sintaxe

```
CREATE TRIGGER nome FOR tabela
  [ATIVE | INATIVE]
  {BEFORE | AFTER} <ação múltipla >
  [POSITION número]
AS Corpo_do_Trigger
```

<ação múltipla> ::= <ação simples> [OR <ação simples> [OR <ação simples>]]
<ação simples> ::= {INSERT | UPDATE | DELETE}

Exemplos

i)
```
CREATE TRIGGER TRIGGER1 FOR TABLE1
[ATIVE] BEFORE INSERT OR UPDATE AS
...;
```

ii)
```
CREATE TRIGGER TRIGGER2 FOR TABLE2
[ATIVE] 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 no diretório /doc/sql.extensions na árvore CVS do Firebird.
Nota (s):

1. Triggers de apenas uma ação são totalmente compatíveis ao nível de 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 compartilhem a mesma semântica e sejam executados exatamente da mesma forma.
3. As variáveis de contexto OLD e NEW estão disponíveis em triggers de múltipla ação. Se a operação que disparar o trigger for incompatível com o seu uso (por exemplo, OLD numa operação de INSERT), todos os campos nesse contexto serão tratados como NULL. A tentativa de atribuir um valor a uma variável num contexto inválido irá provocar uma exceção.
4. Foram criadas novas variáveis de contexto - INSERTING/UPDATING/DELETING – que podem ser utilizadas para verificar a operação em tempo de execução. (Veja mais adiante)

(1.5) RECREATE VIEW

Exatamente o mesmo que CREATE VIEW se a view ainda não existir. Se ela já existir, RECREATE VIEW irá tentar destrui-la antes de criar um novo objeto. RECREATE VIEW irá falhar se o objeto estiver em uso.
Utiliza a mesma sintaxe que CREATE VIEW.

(1.5) CREATE OR ALTER {TRIGGER | PROCEDURE }

Esta declaração irá criar um novo trigger ou procedure (se ainda não existirem) ou alterá-lo (caso eles já existam) 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, exceto 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, em acordo com o SQL-99. Tenha cuidado com a utilização desta nova funcionalidade se planeja reverter a sua base de dados para Firebird 1.0.x ou qualquer versão do Interbase.

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

Onde < lista de colunas únicas > pode conter uma ou mais colunas sem o atributo NOT NULL se a < especificação única > utilizada for UNIQUE OR UNIQUE INDEX. Note que todas as colunas que participem de uma chave primária ainda precisam ser declaradas como NOT NULL**.**

A constraint permite a existência somente daquelas linhas que retornem como verdadeiras as condições de procura (i) or (ii) , seguindo a seguinte lógica :

i) Se a < especificação única > for PRIMARY KEY, então a condição de procura deverá ser :

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

ii) De outra forma, a < condição de procura > deverá ser :

```
UNIQUE ( SELECT LCU FROM TN )
```

Neste caso, a condição única não será verdadeira se (SELECT LCU FROM TN) retornar duas linhas em que os correspondentes segmentos não nulos sejam iguais.

A "constraint" permite a existência somente das linhas onde a já mencionada < condição de procura > seja avaliada como verdadeira. Em um < UNIQUE INDEX > ou sobre uma constraint UNIQUE, dois conjuntos de valores de colunas será considerado distinto, e assim legítimos se :

a) Ambos os conjuntos contém apenas nulos, ou
b) Há pelo menos um par de valores onde um dos valores correspondentes é diferente de nulo e o outro é ou nulo ou um valor não nulo diferente.

Exemplos

UNIQUE constraint:

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

ou UNIQUE index:

```
CREATE TABLE t (a INTEGER, b INTEGER);
COMMIT;
CREATE UNIQUE INDEX uqx ON t(a, b);
COMMIT;
INSERT INTO t VALUES (NULL, NULL); /* ok, nulos permitidos */
INSERT INTO t VALUES (1, 2); /* assim como não nulos */
INSERT INTO t VALUES (1, NULL); /* e combinações */
INSERT INTO t VALUES (NULL, NULL); /* ok, pares nulos são consideredos distintos */
```

mas

```
INSERT INTO t VALUES (1, NULL); /* incorreto porque os segmentos não nulos correspondentes são iguais */
```

Ou seja, uma constraint ***PRIMARY KEY*** não permite nulos enquanto uma constraint UNIQUE ou um UNIQUE INDEX permite quantidades arbitrárias de nulos. Para resultados multi-colunados de (SELECT UCL FROM TN), as regras comuns de nulo se aplicam , 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 permitem como argumentos variáveis locais (em PSQL) e expressões (em DSQL e PSQL).

(1.5) Novas construções para Expressões Condicionais

Arno Brinkman

a) CASE

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

Sintaxe

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

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

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

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

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

<cláusula when simples> ::= WHEN <operador de when> THEN <resultado>
<cláusula when condicional > ::= WHEN <condição de procura> THEN <resultado>
<operador de when> ::= <expressão ou valor>
<cláusula else> ::= ELSE <resultado>
<resultado> ::= <expressão resultado>  | NULL
<expressão resultado> ::= <expressão ou 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) condicional

```
  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 definido por um série de expressões, onde a primeira destas expressões que resulte em um valor não nulo (NOT NULL) será o valor retornado.

Formato

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

Regras de Sintaxe

i) COALESCE (V1, V2) equivale à seguinte expressão <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 NomeDoProjeto,
    COALESCE(e.FULL_NAME,'[> Não Atribuído <]') AS Funcionario
FROM
    PROJET p
    LEFT JOIN EMPLOYEE e ON (e.EMP_NO = p.TEAM_LEADER);

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

a) NULLIF

Retorna NULL para uma expressão se ela resultar no valor especificado, caso contrário retorna o próprio valor da expressão.

Formato
<abreviatura de case> ::=
NULLIF <parent. esquerdo> <expressão ou valor> <vírgula> <expressão ou 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 usuário (também conhecidos como "*nested transactions*") disponibilizam um método conveniente para tratar erros de lógica sem a necessidade de fazer um rollback da transação como um todo. Disponível apenas em DSQL.

Utilize uma declaração SAVEPOINT para identificar um ponto de transação, ao qual será possível retornar através de um "Rollback" parcial.

```
SAVEPOINT <identificador>;
```

<identificador> especifica o nome do SAVEPOINT que será estabelecido. Depois de criado um SAVEPOINT você pode continuar o processamento, executar um commit ou um rollback de toda a transação, ou ainda um rollback até o SAVEPOINT.

Os nomes dos SAVEPOINT devem que ser únicos no contexto de uma transação. Se um segundo SAVEPOINT for estabelecido com o mesmo identificador, o primeiro será sobreposto.

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

Esta declaração executa as seguintes operações:
- Rollback das alterações executadas na transação após o SAVEPOINT
- Destrói todos os SAVEPOINTs criados após este SAVEPOINT. Este SAVEPOINT é mantido, para que você possa fazer um rollback para o mesmo SAVEPOINT múltiplas vezes. Os SAVEPOINTs anteriores a este também serão mantidos.
- Libera todos os locks obtidos implícita e explicitamente após o SAVEPOINT. Outras transações que tenham requerido acesso a linhas que estavam protegidas pelas alterações efetuadas após a criação do SAVEPOINT terão que continuar aguardando que a transação termine, por commit ou rollback. Outras transações, que não tivessem requerido ainda estas linhas, poderão requerer e acessa-las imediatamente.

Nota: este comportamento pode ser alterado em futuras versões.

O “undo log” do SAVEPOINT pode consumir uma grande quantidade de memória do servidor, principalmente se forem feitas modificações sobre os mesmos registros múltiplas vezes na mesma transação. Utilize a declaração RELEASE SAVEPOINT para liberar os recursos consumidos para a manutenção do SAVEPOINT.

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

A declaração RELEASE SAVEPOINT destrói o SAVEPOINT <identificador> do contexto da transação. A menos que especifique ONLY, todos os SAVEPOINTs criados desde o SAVEPOINT <identificador> serão também 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 padrão, o Servidor estabelece um SAVEPOINT de sistema automático para cada transação para possibilitar o seu ROLLBACK. Quando se executa um ROLLBACK, todas as alterações efetuadas na transação são desfeitas utilizando-se este SAVEPOINT de transação, e posteriormente é efetuado o COMMIT da mesma. Esta lógica permite reduzir a quantidade de “garbage collection” originada por transações “rolled back”.

Quando o volume de dados alterados a partir do SAVEPOINT de uma transação é demasiado grande (10^4-10^6 registros afetados) o Servidor libera o SAVEPOINT da transação, e utiliza o mecanismo TIP para, se necessário, executar um RollBack da transação. Se for prevista uma grande quantidade de alterações dentro de uma transação, pode se utilizar o TPB flag isc_tpb_no_auto para desabilitar o SAVEPOINT automático de transação.

SAVEPOINTs e PSQL

Implementar SAVEPOINTs definidos pelo usuário em PSQL altera as regras de atomicidade para as declarações, incluindo as declarações de chamadas a procedures. O Firebird disponibiliza o tratamento de exceções em PSQL para reverter alterações efetuadas em Stored Procedures e Triggers. Cada declaração SQL/PSQL executada é assegurada por um SAVEPOINT interno automático, onde ou toda a declaração é executada com sucesso, ou TODAS as alterações efetuadas por ela são “rolled back” e uma exceção ocorre. Cada bloco de tratamento de exceções PSQL está também incluído em um sistema automático de SAVEPOINT.

(1.5) Locking Explicito

Nickolay Samofatov

O uso da cláusula adicional WITH LOCK oferece uma capacidade limitada de lock pessimista explícito, para um uso cuidadoso e em condições em que a quantidade de linhas afetadas seja a) extremamente pequena (idealmente apenas uma) e b) controlada com precisão a partir do código da aplicação.

NOTA: É raro que seja necessário o uso de lock pessimista no Firebird, e esta funcionalidade deve ser bem compreendida antes que seu uso seja considerado.

Sintaxe

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

Se a cláusula WITH LOCK for bem sucedida, será estabelecido um lock nas linhas selecionadas que impedirá que outras transações obtenham acessos de escrita a qualquer uma delas, ou de suas dependentes, até o fim da transação.

Se a cláusula FOR UPDATE for utilizada, o lock será aplicado a todas as linhas, uma a uma, tão logo a linha seja transferida para o cache do servidor. Assim, é possível que um lock que tenha sido bem sucedido ao ser requerido, falhe subsequentemente, quando for feita uma tentativa de carregar uma linha sobre a qual já tenha sido obtido um lock por uma outra transação.

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

SELECT... WITH LOCK está disponível em DSQL e PSQL. Pode ser bem sucedido apenas no primeiro nível de um SELECT de uma só tabela. Não está disponível para uma sub-query, nem em joins. Não pode ser utilizado com o operador DISTINCT, ou com uma cláusula GROUP BY, nem com qualquer outro tipo de agregação. Não pode ser utilizado a partir de uma view, nem com uma tabela externa, ou no resultado de uma stored procedure selecionável.

Compreendendo a Clausula WITH LOCK

O Servidor considera cada registro contido numa declaração lock explícita, devolvendo ou a sua mais recente versão "commited", qualquer que seja o estado da base de dados no momento em que a declaração foi submetida, ou uma exceção.

O comportamento do "Wait" e a notificação de conflito de lock dependerão dos parâmetros da transação especificados no bloco TPB.

Modo TPB	***Comportamento***
isc_tpb_consistency	Locks explícitos são eliminados por locks implícitos ou explícitos da tabela e são ignorados.

Modo TPB	*Comportamento*
isc_tpb_concurrency + isc_tpb_nowait	Se um registro é modificado por alguma transação que foi "committed" depois que a transação, que está tentado adquirir o lock explicito, começou, ou alguma transação ativa modificou o registro, uma exceção de conflito ocorre imediatamente.
isc_tpb_concurrency + isc_tpb_wait	Se um registro é modificado por alguma transação que foi "committed" depois que a transação que está tentando adquirir o lock explicito começou, uma exceção de conflito é imediatamente retornada. Se uma transação ativa mantém um lock do registro (através de um lock explicito ou de um lock normal de escrita), a transação, que requereu o lock explicito, aguarda pela sua resolução e, após o seu término, tenta novamente obter o lock do registro. Assim, se a transação que bloqueava o registro fizer o "commit" de uma versão modificada do registro, uma exceção de conflito será recebida.
isc_tpb_read_committed + isc_tpb_nowait	Se uma transação ativa bloqueia o registro (através de um lock explicito ou de um lock normal de escrita), uma exceção de conflito ocorre imediatamente.
isc_tpb_read_committed + isc_tpb_wait	Se uma transação ativa bloqueia o registro (via um lock explicito ou por um lock normal otimista de escrita), a transação, que requereu o lock explicito, aguarda pela sua resolução, e após o seu término, tenta novamente obter o lock do registro. Nunca acontecerá uma exceção de conflito neste modo de TPB.

Quando um comando UPDATE encontra um registro bloqueado por outra transação, ele ou recebe uma exceção de conflito, ou aguarda que a transação que possui o bloqueio termine, dependendo do modo TPB. O Comportamento do Servidor será o mesmo que seria caso o registro já tivesse sido modificado pela transação que estabeleceu o bloqueio. Nenhum código de erro especificado será retornado por conflitos de lock envolvendo locks explícitos ou pessimistas.

O Servidor garante que todos os registros recebidos por comando de bloqueio específico estão efetivamente bloqueados e que efetivamente todos os parâmetros da cláusula "where" foram avaliados, desde que a condição de procura não dependa de outras tabelas, via joins, subqueries, etc. Garante ainda que as linhas que não satisfazem a condição WHERE não serão bloqueadas pela transação. Porém, não pode garantir que não existam linhas que, embora satisfaçam a cláusula "where", já estivessem bloqueadas. Esta situação pode acontecer caso alguma outra transação em paralelo tenha executado um "commit" durante o curso de execução do comando de bloqueio.

O Servidor bloqueia as linhas na altura do "fetch". Este mecanismo tem conseqüências importantes caso sejam bloqueadas várias linhas de uma só vez. Muitos métodos de acesso ao Firebird, por padrão, utilizam processos de obtenção de linhas através de pacotes de algumas centenas a cada acesso ("buffered fetches"). A maior parte dos componentes de acesso a dados não devolverá as linhas contidas no último pacote onde tenha ocorrido um erro.

A declaração FOR UPDATE oferece uma técnica que previne a utilização de "buffered fetches", e ainda a opção OF <nome-das-colunas> para ativar UPDATEs posicionais. Alternativamente, pode ser possível configurar nos componentes de acesso o tamanho do buffer de "fetch" para 1. Isto permitirá que uma linha efetivamente bloqueada seja processada, antes que a próxima linha seja obtida e bloqueada, ou

que um eventual erro seja devidamente tratado, sem a necessidade de se efetuar um "rollback" da transação.

O Rollback de um SAVEPOINT implícito ou explicito libera os bloqueios de registros que tenham sido adquiridos nesse SAVEPOINT, mas não notifica outras transações que possam estar aguardando esta liberação. As aplicações não devem basear-se neste comportamento, pois é provável que ele seja modificado no futuro.

Embora os locks explícitos possam ser utilizados para prevenir ou processar erros de conflitos pouco comuns, o volume de erros de deadlock irá aumentar a menos que se estabeleça uma estratégia cuidadosa e rigorosamente controlada. A maior parte das aplicações não deveriam precisar usar locks explícitos. Os principais objetivos dos locks explicitos são (1) evitar custosos processamentos de um grande volume de erros de conflitos de atualização, em aplicações de grande carga de processamento e (2) manter a integridade de objetos mapeados em uma base de dados relacional, em ambientes de clusters. Se o seu uso de locks explícitos estive fora destas duas categorias, então a forma usada não é correta para executar essa tarefa no Firebird.

O lock explícito é uma "Funcionalidade Avançada", não a use inadequadamente ! Embora soluções obtidas através do seu uso sejam importantes para web sites que lidam com milhares de escritas concorrentes, ou para aplicações ERP/CRM em uso em grandes empresas, a maior parte das aplicações não precisam operar 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 era permitido apenas por colunas nomeadas. No Firebird 1.0, já era possível o agrupamento por uma expressão UDF. Na versão 1.5, várias extensões foram implementadas para permitir a processamento de funções agregadas na cláusula GROUP BY, que permite agora o uso da posição da coluna na cláusula (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 claúsula GROUP BY. Por exemplo, a concatenação não é ainda 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_timestamp FROM valor ')';

função_valor_texto      : SUBSTRING '(' valor FROM posição_inteiro ')'
                        | SUBSTRING '(' valor FROM posição_inteiro FOR inteiro
')'
                        | KW_UPPER '(' valor ')'
                        ;
```

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

HAVING

A cláusula HAVING apenas permite funções agregadas ou expressões válidas que façam 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 um comando agregado, 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 retorna valores no FB1.0, mas eles são incorretos!

Subqueries são suportadas numa Função Agregada
O uso de 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 interna 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' "reproduz" 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 seja executada 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 de um número, ele será interpretado como o grau da coluna, como previamente.
A ordem dos nulos no resultado pode ser definida utilizando-se a cláusula de posicionamento dos nulos. Os resultados podem ser ordenados de forma que os nulos sejam posicionados antes (NULLS FIRST) ou depois (NULLS LAST) dos valores não nulos. Quando nada é especificado, a posição dos nulos é NULLS LAST.

Sintaxe

```
SELECT ... FROM .... ORDER BY order_list ....;
order_list : order_item [, order_list];
order_item : <expressão> [order_diretion] [nulls_placement]
order_diretion : 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 retornados por uma UDF ou Stored Procedure serão imprevisíveis se os valores retornados não puderem ser utilizados para determinar a sequência lógica de ordenação.
- O número de vezes que uma UDF ou Stored Procedure será acessado é indeterminado, não importando se a ordenação é especificada pela expressão propriamente ou por um número representando a expressão na lista de colunas do SQL.
- Uma cláusula de ordenação aplicada a uma query com union pode utilizar somente o número de grau (posicionamento) para se referir às colunas de ordenação.

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 SQL para uso em Stored procedures e triggers)

(1.5) EXECUTE STATEMENT

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.

A Sintaxe pode ter três variantes.

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

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

Exemplo

```
CREATE PROCEDURE ExemploDinamico1 (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 um comando SQL, que devolve apenas uma linha. Somente SELECTs que devolvam apenas uma única linha ("singleton") podem ser utilizados com esta sintaxe.

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

Exemplo

```
CREATE PROCEDURE ExemploDinamico2 (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 um comando SQL, devolvendo várias linhas. Qualquer operador SELECT pode ser utilizado nesta variante.

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

Exemplo

```
CREATE PROCEDURE DynamicSampleThree (
   TextField VARCHAR(100),
   TableName VARCHAR(100))
RETURNING_VALUES (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 para o seu formato fixo devem ser efetuadas antes de se processar a declaração EXECUTE STATEMENT.

Esta funcionalidade deve ser utilizada com cautela, e após um exame cuidadoso de todos os fatores. Deve se observar como regra a utilização de um EXECUTE STATEMENT apenas quando outros métodos não são possíveis, ou quando a performance do EXECUTE STATMENT for a mais aceitável.

A declaração EXECUTE STATEMENT é potencialmente insegura nas seguintes condições:

1. Não existe forma de verificar a sintaxe correta da declaração contida na string.
2. Não são feitas verificações de dependência que possam garantir que tabela ou colunas requeridas não tenham sido destruídas.
3. As operações serão mais lentas porque o comando contido terá que ser preparado a cada execução.
4. Os valores devolvidos são verificados e restritos para cada tipo de dado, para evitar resultados imprevisíveis de conversão. Por exemplo, a string '1234' seria convertida para um inteiro, 1234, mas 'abc' geraria um erro de conversão.
5. Se a Stored Procedure tiver privilégios especiais sobre alguns objetos, a declaração dinâmica submetida na string do EXECUTE STATEMENT não os herdará. Os privilégios serão restritos aos atribuídos ao usuário que 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 ativa e da transação corrente, respectivamente. O tipo devolvido é INTEGER. Disponível em PSQL e DSQL. Como estes valores são armazenados no cabeçalho do banco de dados, os seus valores serão zerados 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 afetadas pela última declaração DML. Disponível em PSQL, no contexto de uma trigger ou stored procedure. Na implementação atual, retorna 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 funcionalidade não pode ser utilizada para verificar a quantidade de linhas afetadas por uma declaração EXECUTE STATEMENT.

SQLCODE e GDSCODE

Cada variável de contexto devolve um número inteiro que corresponde ao código de erro da exceção ativa. Disponível em PSQL, dentro do contexto de um bloco de tratamento de exceção. Fora do bloco, ambas retornam zero.

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

Um bloco de exceção 'WHEN SQLCODE' ou 'WHEN ANY' irá receber um valor diferente de zero para a variável SQLCODE e devolver zero para GDSCODE. Apenas um bloco 'WHEN GDSCODE' irá receber um valor diferente de zero para a variável GDSCODE (e devolverá zero em SQLCODE). Se uma exceção criada pelo usuário ocorrer, ambas as variáveis SQLCODE e GDSCODE terão valor zero, independentemente do tipo de bloco de exceção.

Sintaxe

```
SQLCODE
GDSCODE
```

Exemplo

```
BEGIN
   ...
   WHEN SQLCODE -802 THEN
     EXCEPTION E_EXCEPTION_1;
   WHEN SQLCODE -803 THEN
```

```
    EXCEPTION E_EXCEPTION_2;
  WHEN ANY DO
    EXECUTE PROCEDURE P_ANY_EXCEPTION(SQLCODE);
END
```

Veja ainda Melhorias no tratamento de exceções, mais a frente, e o documento README.exception_handling no firebird2/doc/sql.extensions da árvore CVS do Firebird.

INSERTING

UPDATING

DELETING

Estas três expressões pseudo booleanas podem ser utilizadas para testar ou determinar qual a operação de DML que está ocorrendo. Disponível em PSQL, apenas em triggers. Projetadas para uso no contexto de triggers universais (ver METADATA mais adiante).

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 para uma exceção em tempo de execução.
2) Provocar novamente ("re-raise") uma exceção dentro do contexto do bloco da exceção.
3) Obter um erro numérico da exceção tratada.

1) Mensagens de Exceção em "Run-time"

Sintaxe

```
EXCEPTION <nome_exceção> <valor_mensagem>;
```

Exemplos

i)

```
EXCEPTION E_EXCEPTION_1 'Erro!';
```

ii)

```
EXCEPTION E_EXCEPTION_2 'Tipo errado para o registro com o ID=' || new.ID;
```

2) Disparando novamente uma exceção
Nota – não tem qualquer efeito fora de um bloco de exceção.

Sintaxe
```
EXCEPTION;
```

Exemplos
i)
```
BEGIN
  ...
  WHEN SQLCODE -802 THEN
    EXCEPTION E_ARITH_EXCEPT;
  WHEN SQLCODE -802 THEN
    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 Tempo de Execução
Ver SQLCODE / GDSCODE (acima).

(1.5) Declarações LEAVE | BREAK

Interrompe o processamento de um loop, transferindo o processamento para a declaração após o END que delimita o loop. Disponível apenas para estruturas WHILE, FOR SELECT e FOR EXECUTE; qualquer outro uso gerará um erro de compilação. O uso da cláusula SQL-99 LEAVE deverá ter preferência sobre o BREAK existente. Disponível tanto em triggers como em stored procedures.

Sintaxe
```
BEGIN
   <declarações>;
   IF (<condições>) THEN
     LEAVE;
   <declarações>;
END
```

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

Declarações válidas de PLAN podem ser incluídas em Triggers

Ignacio J. Ortega

Até o momento, um trigger que contivesse uma declaração PLAN seria 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 aceitos em módulos PSQL. Por exemplo, pode-se agora escrever módulos "stub" como:

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

(1.5) Declarar e definir variáveis locais numa única declaração

Claudio Valderrama

Simplificação da sintaxe para permitir que variáveis locais possam ser declaradas e inicializadas em uma única declaração.

Sintaxe

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

Exemplo

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

(1.0) SELECT [FIRST (<expr. inteira M>)] [SKIP (<expr.inteira N>)]

(1.5) SELECT FIRST pode agora ter zero como argumento

O FB 1.5 aceita zero como argumento de FIRST. O resultado será uma tabela vazia.

Retorna as primeiras M linhas do resultado da seleção. A cláusula opcional SKIP faz com que as primeiras N linhas sejam ignoradas, retornando um conjunto de M linhas, a partir da linha N + 1. Na forma mais simples, m e n são inteiros mas qualquer expressão Firebird cujo resultado seja um inteiro pode ser utilizada. Um identificador que represente um valor inteiro pode ser usado no em GDML, mas não em SQL ou DSQL.

O parêntesis é obrigatório para os argumentos das expressões e opcional nos outros casos.

Podem também ter sua definição variável, por exemplo, SKIP ? * FROM ATABLE retornará todos os registros após descartar as n linhas iniciais, onde n será definido através da variável "?". SELECT FIRST ? COLUMNA, COLUMNB FROM ATABLE retornará primeiras n linhas e descartará as demais.

A cláusula FIRST também é opcional, ou seja, é possível incluir SKIP numa declaração sem FIRST para obter um conjunto de dados que exclua as linhas indicadas pelo SKIP.

Disponível em SQL e DSQL exceto quando 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 problemas com SELECT FIRST

1. O comando abaixo:

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

apagará todos os registros da tabela. Nossa! O "sub-select" marca as primeiras 10 linhas para eliminação, apaga-as, marca as próximas 10, apaga-as, e assim sucessivamente até que não existam mais linhas disponíveis na tabela.

Cuidado!

2. Queries como esta:

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

não funcionam como esperado, por causa da otimização processada atualmente pelo servidor que trata os predicados correlacionados WHERE...IN (SELECT...) em predicados correlacionados EXISTS. É óbvio que neste contexto 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 no Set de Caracteres

Adicionadas na versão 1.5

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

Novos sets de caracteres ("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, sem diferenciação de maiúsculas, desenvolvido e testado por Sandor Szollosi (ssani@freemail.hu).

- Nesta versão o Firebird suporta o set de caracteres ISO8859-2 (para a linguagem tcheca)

EXTENSÔES DE LINGUAGEM TRANSPORTADAS DO FIREBIRD 1.0.x

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

(1.0) CURRENT_USER e CURRENT_ROLE

Estas duas novas variáveis de contexto foram adicionadas para permitir referenciar o USER e (se implementado[1]) o ROLE do contexto da presente 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
ATIVE AFTER INSERT POSITION O 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 sinônimo DSQL de USER que aparece no SQL standard. Eles são idênticos. Não existe qualquer vantagem de CURRENT_USER sobre USER.

[1] Se você ainda estiver usando um banco de dados InterBase v.4.x ou 5.1 com o Firebird, ROLE não será suportado, assim o CURRENT_ROLE será NONE (como definido pelo standard SQL na ausência de um ROLE declarado) mesmo que o usuário informe ROLE válido. Bancos com versão IB 5.5, IB 6 ou Firebird, tem o ROLE informado e verificado. Caso o ROLE não exista, ele passará a ser tratado como NONE sem que qualquer erro ocorra.

Isto significa que no FB jamais devolverá um ROLE inválido como CURRENT_ROLE, porque o valor retornado será NONE. Este comportamento contrasta com o IB, onde o valor inválido é aceito internamente, embora não de forma visível para o SQL.

(1.0) DROP GENERATOR

Permite que GENERATORs não utilizados sejam removidos da base de dados. O armazenamento será liberado para reutilização no próximo RESTORE. Acessível em SQL e DSQL.

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

(1.0) GROUP BY UDF

Agora é possível agregar um SELECT por grupo pelo resultado de uma UDF.
Por exemplo.

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

Um efeito colateral das alterações para permitir o agrupamento por UDFs : Previamente não era permitida a utilização de funções internas do Firebird em uma cláusula GROUP BY. Agora isto é possível através da criação de uma estrutura similar a uma 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, sem que seja necessário fazer antes um DROP da mesma. 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 já existente, sem necessidade de que se faça antes um DROP da tabela antiga. A sintaxe é idêntica a CREATE TABLE.

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

Disponível em SQL e DSQL.

(1.0) SUBSTRING (<*express alfanumérica*> FROM <*pos*> [FOR <*tam*>])

Função interna que implementa a função ANSI SQL SUBSTRING(). Irá devolver uma variável contendo o byte da posição <*pos*> e todos os bytes subsequentes até ao final da expressão. Se a opção FOR <*tam*> estiver especificada, a extensão será a menor destes dois valores: o tamanho <tam> especificado, ou o número de bytes até o final da expressão.

O primeiro argumento poderá ser uma expressão qualquer, constante ou identificador, que possa ser tratado como alfanumérico.

<*pos*> deve resultar em um inteiro.
<*pos*> tem índice inicial 1, como outras declarações SQL.
Nem <*pos*> nem <*tam*> podem ser parâmetros de Querys.

Como <pos> e <tam> são posições de byte, o identificador pode ser um blob binário, ou um blob sub_type 1 tipo texto, desde que o set de caracteres definido utilize um byte por caractere. A função não trata, no momento, blobs de texto com sets de caracteres Chineses (2 byte/caractere) ou Unicode (3 bytes/caractere no máximo). Para um argumento do tipo alfanumérico (ao contrário de um blob), a função suporta qualquer set de caracteres.

Disponível em SQL e DSQL.

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

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

(1.5) Aperfeiçoamento do Marcador de Comentário de uma Linha

Dmitry Yemanov

Os comentários de uma linha podem ser utilizados em qualquer posição, não apenas no início da linha. 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. Assim, pode ser utilizado para comentar partes não desejadas de uma declaração. Todos os caracteres a partir do marcador '—' até o 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 uso 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 para ignorar caracteres é a seguinte:

1. Ignorar os caracteres "--" se este par de caracteres for o primeiro depois de um indicador de Fim de Linha (LF em Linux/Unix, CRLF em Windows)
2. Continuar a ignorar os caracteres até o próximo indicador de fim-de-linha (EOL)

Esta lógica não prevista para ser aplicada juntamente com blocos de comentário lógico (/* um comentário */). Em outras palavras, não use os marcadores "--" dentro de um bloco de comentário, e não utilize marcadores de bloco em uma linha que contenha marcadores "--".

SESSÕES ISQL INTERATIVAS: Tenha em mente o seguinte quando trabalhar numa sessão ISQL interativa : O ISQL aceita particionar uma declaração em segmentos contínuos, mostrando a mensagem CON> até que seja recebido o caractere de terminação (normalmente ;). Se for digitado "--" no início de uma linha de continuação, a lógica de ignorar será mantida até encontrar a marcação de fim-de-linha (EOL) na tela ou arquivo de saída se pressionado ENTER. Uma fonte potencial de erros é contar que a linha seguinte também será ignorada.

O problema com o ISQL acontece porque ele tem seus comandos específicos que precisam ser tratados apenas pelo próprio ISQL. Se eles não forem identificados devido a uma colocação incorreta de "--", então serão encaminhados ao servidor. Obviamente, o servidor não entenderá os comandos do ISQL SET e SHOW e portanto eles serão rejeitados.

(1.0) Alter Trigger não mais incrementa o "Contador de Alterações" na Tabela

Quando o contador das alterações de meta dados de uma tabela qualquer atinge o valor máximo de 255, a base de dados fica indisponível. Uma operação de "Backup/Restore" é necessária para zerar o contador e para disponibilizar novamente a base de dados. A intenção desta funcionalidade é forçar limpezas da base de dados sempre que a sua estrutura tiver sofrido muitas alterações, e não para inibir o uso de recursos do servidor.

Anteriormente, cada vez que um trigger fosse definido como ATIVE/INATIVE através de uma operação ALTER TRIGGER, o contador de alterações da tabela associada era incrementado. Assim, a o uso da ativação e desativação de triggers como recurso normal de operação era afetado, porque este uso faria o contador crescer 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 ao histórico de comandos (como o readline do Unix) foi adicionado a interface do ISQL. Agora, as teclas Acima e Abaixo podem ser usadas para navegar pelos comandos já submetidos na sessão ISQL.

Funções Definidas pelo Usuário (UDF's)

No ib_udf

rpad (*instring, tam, car***)**

Juan Guerrero

Preenche a string fornecida, através da adição do caractere padchar até a string resultante ter o tamanho *tam*. A string de entrada poderá ter no máximo 32766 bytes. Tam não poderá 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, tam, car***)**

Juan Guerrero

Preenche a string fornecida, através da aposição do caractere car até a string resultante ter o tamanho *tam*. A string de entrada poderá ter no máximo 32766 bytes. Tam não poderá 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 possuía um erro muito antigo, em que os argumentos x e y estavam invertidos. Deveria devolver o logaritmo base x de y, mas de fato devolvia o log base y de x. Foi corrigido.
Se a função era utilizada previamente nas suas aplicações, VERIFIQUE O SEU CÓDIGO! Ou os resultados obtidos eram incorretos, ou os valores estavam deliberadamente trocados para que fossem obtidos os resultados corretos.

Em fb_udf

1. As funções *NVL e *NULLIF foram mantidas para manter compatibilidade com versões anteriores, mas estão obsoletas em razão da introdução das novas funções internas CASE, COALESCE e NULLIF.
2. Note que a fbudf não consegue processar campos string com um tamanho maior do que 32Kb - 1 bytes. Esta limitação poderá causar efeitos colaterais quando strings são concatenados antes de serem passadas como argumentos as UDFs. Se a soma dos campos ultrapassar o limite, o comportamento será indefinido. A função poderá devolver resultados sem sentido ou o código do fbudf poderá executar uma operação ilegal.
3. Em um Banco de Dados portado que tenha sido criado com o Firebird 1.0.x, e em que tenham sido declaradas as funções fbudf **truncate** e **round**, estas declarações não serão funcionais para o Firebird 1.5 porque as posições de entrada foram alteradas. Será preciso fazer um DROP das funções e uma nova declaração das mesmas a partir do fbudf.sql script do diretório de UDF da versão 1.5.

Novo Arquivo de Configuração – firebird.conf

O Diretório Raiz do Firebird

O diretório raiz de instalação do Firebird é utilizado de diversas formas, durante a instalação, e também como atributo para rotinas do servidor, parâmetros de configuração e de clientes. Como existem várias formas de indicar ao servidor onde encontrar o valor para este atributo, programadores e administradores de sistema devem conhecer a lógica de precedência que o servidor observa na partida, para a determiná-lo corretamente.

Win32 Superserver e Classic (tanto o servidor como o cliente):
1) Variável de ambiente FIREBIRD
2) Parâmetro RootDiretory no firebird.conf
3) Registro:
   HKLM\SOFTWARE\ SOFTWARE\Firebird Projet\Firebird Server\Instances\DefaultInstance
   verificando a chave RootDiretory.
4) O diretório um nível acima ao do executável do servidor.

Win32 Embedded (Servidor Embutido):
1) Variável de ambiente FIREBIRD
2) Parâmetro RootDiretory no firebird.conf
3) O diretório onde fbembed.dll (renomeado fbclient.dll) está localizado.

Linux Classic:
1) Variável de ambiente FIREBIRD
2) Parâmetro RootDiretory no firebird.conf
3) Caminho padrão de instalação (/opt/firebird)

Linux Superserver:
1) Variável de ambiente FIREBIRD
2) Parâmetro RootDiretory no firebird.conf
3) O diretório localizado um nível acima ao do executável do servidor. (obtido via symlink "/proc/self/exe", quando suportado)
4) Caminho padrão da instalação /opt/firebird)

Parâmetros

Quase todos os Parâmetros possuem valores padrões. Nomes de parâmetros e seus valores diferenciam maiúsculas no Linux, mas não em Windows. Para alterar o valor padrão de um parâmetro para elimine o marcador de comentário (#) e altere o valor. O arquivo de configuração pode ser alterado com o servidor em execução, mas para efetivar as alterações é necessário parar e reiniciar o serviço.

As configurações seguem o formato:

Nome_parâmetro *valor*

- Nome_parâmetro é um alfanumérico que não contém espaços em branco e que nomeia uma propriedade de configuração do servidor.

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

Parâmetros do Sistema de Arquivos

RootDiretory

Uma string, caminho absoluto do diretório raiz no sistema de arquivos local. Deve permanecer comentada, a menos que se deseje forçar o procedimento de partida a utilizar um diretório de instalação do servidor Firebird, diferente daquele que seria automaticamente detectado.

DatabaseAccess

Suporta a nova funcionalidade de Alias. Em versões anteriores, o servidor podia conectar-se a qualquer base de dados no seu sistema de arquivos local, e era acionado pelos clientes que informavam o caminho absoluto para o arquivo da base de dados. Este parâmetro permite restringir o acesso aos bancos apenas através do seu Alias, ou a bases de dados localizadas em determinados pontos da árvore de diretórios do sistema de arquivos.
DatabaseAccess pode ser None, Restrict ou Full.
Full (o valor padrão) permite o acesso a arquivos de bases de dados qualquer parte do sistema de arquivos local.
None restringe o acesso a bases de dados que se encontrem definidas em **aliases.conf**.
Restrict permite configurar os grupos de diretórios que conterão arquivos de bancos de dados acessíveis pelo servidor. Pode-se definir um ou mais diretórios, separados por 'ponto e vírgula', para definir um ou mais locais permitidos.
Por exemplo,
Unix: `/db/databases;/userdir/data`
Windows: `D:\data`
Caminhos relativos são tratados como pertencentes ao caminho que o servidor estabeleceu como sendo o seu diretório raiz. Por exemplo, no Windows, se o raiz do Firebird é o diretório C:\Programas\Firebird, então o valor abaixo irá restringir acesso do servidor somente aos arquivos que estejam localizados em C:\Programas\Firebird\userdata:

```
DatabaseAccess = Restrict userdata
```

NOTA Shadowing de Banco de Dados – a implementação atual de DatabaseAccess tem uma falha que torna obrigatório o uso da configuração Restrict caso o shadowing esteja ativado para qualquer banco no Servidor.

ExternalFileAccess

Era *external_file_diretory* em isc_config/ibconfig mas a sintaxe foi alterada.

Provê três níveis de segurança concernentes a EXTERNAL FILES (arquivos texto com formato fixo que são acessados como tabelas da base de dados). O valor é uma string a qual se pode atribuir None, Full ou Restrict.
None (o valor padrão) desativa o uso de arquivos externos no servidor;
Restrict permite restringir a localização dos arquivos externos a diretórios específicos. Indique um ou mais diretórios separados por ´ponto e vírgula', onde os arquivos externos deverão estar localizados.
Por exemplo:
Unix: `/db/extern;/mnt/extern`
Windows: `C:\ExternalTables`

Caminhos relativos são tratados como relativos ao caminho que o servidor reconhece como sendo o seu diretório raiz de instalação.
Por exemplo, no Windows, se o caminho reconhecido for o diretório C:\Programas\Firebird, então o valor abaixo irá restringir o acesso do servidor somente a arquivos externos que estejam localizados em C:\Programas\Firebird\userdata\ExternalTables:

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

Full permite que os arquivos externos sejam acessados em qualquer diretório do sistema.

Veja a recomendação de **Cuidado** após o próximo parâmetro de configuração, UdfAccess.

UdfAccess

Era *external_function_diretory* em isc_config/ibconfig mas a sintaxe foi alterada.

Foi mudado não só o nome do parâmetro, mas também a forma como os valores são definidos. O propósito das alterações foi ativar opcionalmente níveis de proteção para módulos de biblioteca externa definidas pelo usuário, um conhecido alvo de ataques maliciosos. UdfAccess pode ser None, Restrict ou Full.

Restrict (o valor padrão) mantêm a funcionalidade que era disponibilizada pelo parâmetro **external_function_diretory** no Firebird 1.0, para restringir a localização das bibliotecas externas a locais específicos do sistema de arquivos. Forneça um ou mais diretórios, separados por 'ponto-e-vírgula', para armazenagem das bibliotecas de UDF, Filtros BLOB e definições de set de caracteres.
Por exemplo,:
Unix: `/db/extern;/mnt/extern`
Windows: `C:\ExternalModules`
Caminhos relativos são tratados como relativos ao caminho que o servidor reconheceu como sendo o diretório de raiz de instalação do Firebird. Assim, no Windows, se o diretório raiz do Firebird for C:\Programas\Firebird\userdata, o valor abaixo restringirá o acesso do servidor a arquivos externos localizados em C:\Programas\Firebird\ExternalModules:

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

None desabilita o uso de bibliotecas externas.

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

Cuidado :: Evite definição de diretórios particularizados para UdfAccess e ExternalFileAccess que provoque o compartilhamento de uma mesma árvore de diretórios. Os valores padrões são seguros. Caso esteja configurando caminhos próprios, e não tome o cuidado de estabelecer locais separados para os dois tipos de arquivos, você deixará o servidor vulnerável à execução de código não autorizado. Um exemplo do que deve ser evitado:

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

UdfAccess e ExternalFileAccess terão uma parte da árvore em comum, `/bad_dir/files`, onde um usuário poderia criar um arquivo externo `/bad_dir/files/hackudf.so` com código próprio, e executá-lo neste sistema comprometido.

Parâmetros dos Recursos

CpuAffinityMask

Era *cpu_affinity* em isc_config/ibconfig

Na versão SuperServer do Firebird em Windows, e em máquinas SMP, existe uma falha que faz com que o sistema operacional transfira constante todo o processamento entre os processadores existentes. Isto acaba com a performance. Este parâmetro pode ser utilizado em sistemas SMP em Windows para estabelecer uma afinidade do Firebird com um único processador.

NOTA O Firebird SuperServer, até e incluindo a versão 1.5, podem não suportar o recurso de HyperThreading no Windows em algumas placas mães mais recentes. Para evitar problemas de balanceamento, pode ser necessário desabilitar o HyperThreading no Bios do Sistema.

CpuAffinityMask é um inteiro, a máscara da CPU.

Exemplo

`CpuAffinityMask = 1`
Usa apenas a primeira CPU (CPU 0).

`CpuAffinityMask = 2`
usa apenas a segunda CPU (CPU 1).

`CpuAffinityMask = 3`
Usa a primeira e segunda CPU.

Calculando o valor da máscara da afinidade
Você pode usar esta sinalização para definir a afinidade do Firebird com qualquer processador ou combinação de processadores (no servidor Classic) instalado no sistema.
Considere as CPUs como uma matriz numerada de 0 a *n*-1, onde *n* é o número total de processadores instalados e *i* é o número da matriz de uma CPU. M é outra matriz contendo a o valor da máscara de cada CPU selecionada. O valor de A é a soma dos valores de M.
Use a fórmula seguinte para estabelecer M e obter o valor da máscara A:

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

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

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

DeadlockTimeout

Era *deadlock_timeout* em isc_config/ibconfig

Intervalo em segundos (inteiro) que o gerenciador de bloqueio aguardará se um conflito ocorrer, antes de eliminar os bloqueios de processos mortos e efetuar um novo ciclo de verificação de "deadlock". Normalmente, deadlocks são detectados instantaneamente. Este intervalo só ocorrerá quando algum erro o provocar.

O valor padrão de 10 segundos é adequado para quase todas as condições. Diminuir este valor não vai necessariamente reduzir o tempo em que um deadlock devolve uma exceção de conflito. Se for baixo demais, o efeito pode ser verificações extras desnecessárias e que podem degradar o desempenho do sistema.

DefaultDbCachePages

Era *database_cache_pages* em isc_config/ibconfig

Número de página de dados alocadas em memória pelo servidor, por banco de dados. O valor configurado pode ser estabelecido para cada base de dados.
O valor padrão para o SuperServer é de 2048 páginas. Para o Classic é 75.
As versões SuperServer e Classic utilizam a memória de formas diferentes. No SS a memória é compartilhada por todas as conexões, enquanto a Classic aloca a memória definida estaticamente para cada conexão.

EventMemSize

Inteiro, representa o número de bytes de memória reservado para o gerenciador de Eventos. O valor padrão é 65536 (64 Kb).

LockAcquireSpins

Era *lock_acquire_spins*

Relevante apenas para ambientes SMP executando servidor Classic. Num servidor Classic, apenas um processo cliente pode acessar a tabela de locks num determinado momento. Um mutex controla este acesso. Os processos clientes podem requerer o mutex condicionalmente ou incondicionalmente. Se for condicional, a requisição poderá falhar e deverá ser repetida. Caso seja incondicional, a requisição aguardará até ser atendida. O parâmetro LockAcquireSpins estabelece o número de tentativas que serão efetuadas se o mutex for condicional.
Inteiro. O padrão 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". Sob grande carga, a taxa de transferência pode ser aumentada ampliando-se o número de "hash slots" para distribuir a lista em cadeias menores de "hash". Um número primo é recomendado. O valor padrão é 101.

LockGrantOrder

Era *lock_grant_order* em isc_config/ibconfig

Quando uma conexão deseja obter o lock ou bloqueio de um objeto, obtém um bloco de requisição de "lock" que especifica o objeto e o nível de lock requerido. Os blocos de requisição são conectados a aos blocos de locks como requisições já atendidas, ou como requisições ainda pendentes.
O parâmetro LockGrantOrder é um Booleano. O padrão (1=True) indica que as requisições de lock devem ser atendidas no modelo primeiro a chegar primeiro a ser servido. A configuração falsa (0), emula o comportamento do InterBase v3.3, que concedia o lock tão logo ele ficasse disponível. Este modo pode resultar em requisições não atendidas por um longo tempo.

LockMemSize

Este inteiro representa a quantidade bytes na memória alocados para o gerenciador de locks. Para um servidor Classic, LockMemSize indica o valor inicial alocado que poderá crescer dinamicamente até esgotar toda a memória (*"Lock manager is out of room").* Se houver muitas conexões ou caches de páginas muito grandes, aumente este valor, para evitar estes erros.

No SuperServer, a memória alocada pelo gerenciador não aumenta dinamicamente.
O valor padrão no Linux e Solaris é de 98.304 bytes (96 Kb). No Windows, é de 262144 (256 Kb).

LockSemCount

Parâmetro inteiro que especifica o número de semáforos disponíveis para comunicação entre processos (IPC). O valor padrão é 32. Use este parâmetro em ambientes que não utilizem processos paralelos para aumentar ou reduzir o número de semáforos disponíveis.

SortMemBlockSize

Este parâmetro permite configurar, em bytes, o tamanho de cada bloco de memória utilizado pelas rotinas de ordenação em memória. O valor padrão é de 1Mb, mas pode ser definido para qualquer valor até o máximo estabelecido pelo parâmetro SortMemUpperLimit (a seguir).

SortMemUpperLimit

Quantidade máxima de memória, em bytes, que pode ser alocada para rotinas de ordenação. O valor padrão é de 67108864 bytes (64 Mb) para o SuperServer e 8388608 (8 Mb) para o Classic.

> CUIDADO: No Classic, considere que aumentar o o tamanho do bloco de memória ou o limite de memória utilizada afeta cada conexão/instância do servidor e aumentará o uso total de memória consideravelmente.

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 padrão é 180.

DummyPacketInterval

Era *dummy_packet_interval* em isc_config/ibconfig

Intervalo em segundos (inteiro) que o servidor irá aguardar antes de enviar um "dummy packet" para um cliente inativo para confirmar uma conexão.
NÃO USE ESTA OPÇÃO em servidores Win32 com clientes TCP/IP. Provoca um aumento constante de utilização da memória não paginada que poderá travar ou derrubar o Windows do lado do cliente, como explicado aqui:
http://support.microsoft.com/default.aspx?kbid=296265

Excluindo-se o Win32-com-TCP/IP, este é o único meio de detectar e desconectar clientes inativos quando NamedPipes (NetBEUI), XNET ou IPC são utilizados. Não há qualquer restrição conhecida para sistemas POSIX.

Normalmente, o Firebird usa a opção do socket SO_KEEPALIVE para controlar as conexões ativas. Se o timeout de 2 horas (valor padrão) não for ideal, ajuste os parâmetros do sistema operacional apropriadamente:

- 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

O padrão deve ser 0 – e não 60 como era no Firebird 1.0 e na maior parte das RC do 1.5. O valor de 60 deverá ser utilizado como padrão em sistemas que façam uso do envio de pacote "dummy".

RemoteServiceName

Valor padrão = gds_db

RemoteServicePort

Estes dois parâmetros permitem substituir o nome do serviço TCP/IP ou a porta TCP/IP utilizada para a conexão dos clientes, quando um deles for diferente dos utilizados como padrão (gds_db/tcp 3050).

Mude um dos parâmetros, não os dois. O RemoteServiceName é verificado primeiramente e caso uma entrada seja encontrada no arquivo de serviços a porta especificada por RemoteServicePort é utilizada. Se o serviço não for localizado, será usada a porta padrão 3050.

NOTA: Se uma porta de conexão for indicada na string de conexão TCP/IP, ela terá precedência sobre RemoteServicePort.

RemoteAuxPort

O comportamento herdado do InterBase de transmitir mensagens de notificação de eventos para a camada de rede através de portas TCP/IP selecionadas de forma aleatória, tem sido uma constante fonte de erros de rede e conflitos com firewalls, a ponto de até mesmo provocar a queda do servidor em certas condições. Este parâmetro permite configurar uma única porta TCP para todo o tráfego de notificações de eventos.
O valor de instalação padrão (0) conserva o comportamento tradicional de portas randômicas. Para definir uma porta específica para a notificação de eventos, use um número inteiro correspondente a uma porta disponível.

RemoteBindAddress

Por padrão, os clientes podem conectar-se por meio de qualquer interface de rede através da qual o Host do servidor aceite ser acessado. Este parâmetro permite relacionar o serviço Firebird a requisições recebidas somente através de uma única interface e rejeitar conexões de quaisquer outras interfaces de rede. Isto deve ajudar a contornar problemas em algumas sub-redes onde o servidor controle o tráfego de múltiplas interfaces.
String, com endereço IP válido. Valor padrão (sem ligação) é em branco.

TcpRemoteBufferSize

O servidor faz leituras para antecipar-se ao cliente e pode enviar várias linhas de dados num único pacote. Quanto maior o tamanho do pacote, mais linhas serão enviadas em cada transferência. Use este parâmetro – com cuidado e compreensão profunda dos seus efeitos no desempenho da rede! – caso

precise aumentar ou reduzir o tamanho do pacote TCP/IP usado para os buffers de envio e recebimento. Afeta tanto o cliente quanto o servidor.
O valor é um inteiro (tamanho do pacote em bytes) entre 1448 e 32768. O valor padrão da instalação é de 8192.

Parâmetros específicos POSIX

LockSignal

Parâmetro inteiro, sinalizador do UNIX para comunicações entre processos. Valor padrão: 16

RemoteFileOpenAbility

USAR APENAS COM EXTREMA CAUTELA

Parâmetro Booleano que, se configurado como True, permite que o Servidor abra arquivos de bases de dados que residam em partições montadas a partir de sistema de arquivos em rede (NFS). Como o sistema de arquivos se encontra fora do controle do sistema local, esta é uma funcionalidade muito arriscada e que não deve ser habilitada com o propósito de ler/escrever bases de dados de importância vital para você.
O valor padrão é 0 (Falso, desativado) e assim deve ser mantido, a menos que se esteja bem ciente dos seus efeitos.

TcpNoNagle

Era *tcp_no_nagle* em isc_config/ibconfig

No Linux, por padrão, a biblioteca de socket minimizará as escritas, acumulando-as antes do real envio dos dados, usando um algoritmo interno (implementado como a opção TCP_NODELAY do socket de conexão) conhecido como o Algoritmo de Nagle. Ele foi proejtado para evitar problemas com pacotes pequenos, conhecidos como "tinygrams", em redes lentas.
Por padrão, TCP_NODELAY é ativado (valor 0) quando o Firebird Superserver é instalado no Linux. A sua desativação poderá aumentar a velocidade em redes lentas. Preste atenção na dupla negação : colocar o parâmetro como True desativa o TCP_NODELAY, e colocá-lo como False ativa-o.
Em versões até e inclusive a 1.5, esta funcionalidade só pode ser ativada para o Superserver.

Parâmetros específicos do Windows

CreateInternalWindow

O protocolo local Windows usa uma janela escondida para a comunicação entre processos entre o cliente local e o servidor. Esta janela IPC é criada na partida do servidor quando o CreateInternalWindow for true (1, valor padrão). Configure-o como 0 (desativado) para executar o servidor sem a janela e desativar o protocolo local. Com o protocolo local desativado, é possível executar múltiplas instâncias do servidor simultaneamente.

DeadThreadsCollection

Uma parametrização para o gerenciador de processos no Windows, este valor inteiro define o número de ciclos de mudança de prioridade (veja PrioritySwitchDelay, a seguir) que o gerenciador executará antes de destruir um processo (ou fechá-lo).
A imediata destruição (ou fechamento) de processos ativos exigiria um semáforo e uma chamada de bloqueio, gerando uma sobrecarga considerável. Em vez disso, o gerenciador de processos os mantêm

em espera. Assim que o processo completar a sua tarefa, ele será marcado como inativo. O processo inativo é destruído (ou fechado) após *n* iterações do ciclo do gerenciador, onde *n* é o valor do parâmetro DeadThreadsCollection.
Para que um servidor possa gerenciar um grande número de conexões — várias centenas ou mais — o valor do parâmetro deverá ser ajustado para acima do valor padrão, que é 50.

GuardianOption

Parâmetro booleano utilizado em servidores de Windows para determinar se o Guardian deve reiniciar o servidor sempre que ele termine de forma anormal. O valor padrão de instalação (1=True) o fará. Para desabilitar este recurso, defina o parâmetro como falso (0).

IpcMapSize

Era *server_client_mapping* em ibconfig

Tamanho em bytes para o segmento de memória mapeado pelo cliente para a comunicação entre processos (IPC) no modelo de conexão usado pela cliente local no Windows. Não possui qualquer equivalência em outras plataformas. Inteiro, de 1024 até 8192. O valor padrão é 4096.
Aumentar o tamanho da memória mapeada poderá melhorar o desempenho na leitura de grandes quantidades de dados, ou de dados de grande dimensão, tais como BLOBs gráficos.

> NOTA: Este parâmetro não poderá mais ser alterado a partir do ícone do Guardian localizado na Área de Notificação.

IpcName

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

MaxUnflushedWrites

Este parâmetro foi introduzido na Versão 1.5 para contornar uma falha nos sistemas operacionais Windows Server, em que escritas assíncronas nunca eram gravadas no disco, exceto quando o servidor Firebird era fechado de forma controlada. (Escritas assíncronas não são suportadas no Windows 9x ou ME.) Assim, em sistemas 24/7, as escritas assíncronas nunca eram efetivadas de fato.
Este parâmetro determina a freqüência com que páginas retidas serão gravadas no disco quando o parâmetro "Forced Writes" estiver desativado (e a escrita assíncrona ativada). O valor é um inteiro, que representa a quantidade de páginas retidas antes de ser solicitada uma gravação em disco, que será executada da próxima vez que for efetuado um commit da transação. O valor padrão é 100 nas instalações do Windows e -1 (desativado) nas instalações das outras plataformas.
Se o fim de um ciclo MaxUnflushedWriteTime (ver abaixo) ocorrer antes que o número limite de páginas retidas o seja, a gravação é requisitada imediatamente e o número de páginas retidas é zerado.

MaxUnflushedWriteTime

Este parâmetro define o tempo máximo de retenção de páginas para a escrita assíncrona, antes que elas sejam gravadas em disco, quando o Forced Writes estiver desativado (escrita assíncrona ativada). O seu valor é um inteiro que definirá o intervalo, em segundos, entre a última gravação em disco e a

ativação de um sinal para que seja executada a gravação, na próxima vez que for efetuado um commit da transação. O valor padrão é de 5 segundos nas instalações Windows e -1 (desativado) nas instalações das outras plataformas.

PrioritySwitchDelay

Configuração do Gerenciador de Processos no Windows, este valor inteiro estabelece o tempo, em milissegundos, que deverá ser aguardado antes que a prioridade de um processo inativo seja colocada como baixa e prioridade de um processo ativo seja definida como alta. Uma iteração desta série de alterações de prioridade representa um ciclo do gerenciador de processos.
O valor padrão é 100 ms; estabelecido através de medições com processadores Intel PIII/P4. Para processadores com de menor freqüência de velocidade, será preciso um intervalo maior.

PriorityBoost

Valor Inteiro que define o número de ciclos extra dado a um processo quando sua prioridade é elevada para alta. O valor padrão 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 pré 1.5, — ver abaixo — com uma nova implementação.
Os valores inteiros são :

- 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 eles efetivamente melhoraram o tempo de resposta do Servidor.

RemotePipeName

Aplicável apenas para conexões NetBEUI
Parâmetro String, o nome do pipe usado como canal de transporte no protocolo NetBEUI. O nome da pipe equivale ao número da porta para o TCP/IP. O valor padrão, interbas, é compatível com versões anteriores do Firebird e com o InterBase®.

Parâmetros para configurar o espaço temporário usado para ordenações

Quando o tamanho do buffer interno é pequeno demais para acomodar as linhas envolvidas em uma operação de ordenação, o Firebird precisa criar arquivos temporários de ordenação no sistema de arquivos do servidor. Por padrão, será verificado o caminho especificado pela variável de ambiente **FIREBIRD_TMP**. Se a variável não estiver disponível, o servidor tentará usar a pasta **/tmp** do sistema de arquivos do Linux/UNIX, ou **C:\temp** no Windows NT/2000/XP. Nenhum destes locais permite configuração do espaço a ser utilizado.
O Firebird dispõe de um parâmetro que permite configurar o espaço em disco que será usado para armazenar estes arquivos temporários. Convém utilizá-lo, para assegurar que sempre haverá espaço suficiente para ordenação em todas as condições.
Todas as requisições CONNECT ou CREATE DATABASE compartilham a mesma relação de diretórios temporários e cada uma cria os seus próprios arquivos temporários. Os arquivos de ordenação serão liberados após o fim da ordenação ou após a liberação da requisição.
Na Versão 1.5, o nome do parâmetro mudou de **tmp_diretory** para **TempDiretories** e a sintaxe do parâmetro também foi alterada.

TempDiretories

Substitui as entradas *tmp_diretory* em isc_config/ibconfig

Informa uma lista de um ou mais diretórios, separados por ponto-e-vírgula (;), onde arquivos temporários de ordenação podem ser armazenados. Cada item poderá incluir um argumento opcional de tamanho máximo, em bytes, de armazenamento. Se este argumento for omitido ou inválido, o Firebird esgotará todo o espaço no diretório antes de passar para a próximo diretório da lista.
Por exemplo :
Unix: `/db/sortfiles1 100000000;/firebird/sortfiles2`
Windows: `E:\sortfiles 500000000`
Caminhos relativos são tratados como relativos ao caminho que o servidor em execução estabeleceu como o do diretório raiz de instalação do Firebird. Por exemplo, no Windows, se o diretório raiz for C:\Programas\Firebird, o valor abaixo indicará ao servidor para que armazene os arquivos temporários em C:\Programas\Firebird\userdata\sortfiles, até ao limite de 500 Mb:

```
TempDiretories = userdata\sortfiles 500000000
```

Nota: Sem aspas, ao contrário do que era requerido no Firebird 1.0

Parâmetros de Compatibilidade

CompleteBooleanEvaluation

Implementa o método de avaliação de expressões Booleanas (completa ou parcial). O valor padrão (0=False) faz com que a avaliação de expressões booleanas com predicados AND e OR seja interrompida tão logo seja possível obter um resultado VERDADEIRO OU FALSO que não poderá mais ser afetado por quaisquer avaliações posteriores.
Em certas e raras condições (normalmente evitáveis), é possível que uma expressão dentro de uma condição OR ou AND e que não tenha sido avaliada em razão desta configuração parcial, tenha o potencial de afetar o resultado final. Caso você tenha a infelicidade de herdar uma aplicação que tenha tais características na sua lógica de SQL, use este parâmetro para forçar a avaliação completa até ter oportunidade de corrigi-la. O parâmetro é um booleano.

> Não subestime o fato de que esta opção afeta todas as avaliações de Booleanas em todos os bancos de dados do servidor.

OldParameterOrdering

A versão 1.5 encaminhou e resolveu um conhecido erro do InterBase que provocava uma ordenação idiossincrática na estrutura XSQLDA dos parâmetros de saída devolvidos aos clientes. O erro vinha de tão longe que muitas aplicações, drivers e interfaces existentes possuíam soluções embutidas para contornar o problema no lado do cliente.
As versões 1.5 e posteriores corrigiram esta condição na API e são instaladas com o OldParameterOrdering=0 (Falso). Mude o parâmetro Booleano para True caso precise do comportamento anterior, para manter compatibilidade com códigos já existentes.

Aliasing dos Arquivos BD

A versão 1.5 do Firebird introduz o apelido ou alias de arquivos de base de dados para melhorar a portabilidade das aplicações e para aumentar o controle interno e externo do acesso às bases de dados.

Aliases.conf

Configure os alias de arquivos de base de dados no arquivo texto aliases.conf, localizado no diretório raiz da Instalação do Firebird. O aliases.conf é similar a :

```
#
# List of known database aliases
# ------------------------------
#
# Exemplos:

#
#   dummy = c:\data\dummy.fdb
#
```

Como em todos os arquivos de configuração do Firebird, o símbolo '#' funciona como indicador de comentários. Para configurar um alias, simplesmente apague o '#' e altere a linha com o dummy para o valor correto do caminho do banco 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 com o servidor em execução. Não existe qualquer necessidade de parar ou reiniciar o servidor para que as novas entrada no aliases.conf sejam aplicadas.

Conexão usando um caminho com alias

A string de conexão alterada na sua aplicação cliente deverá ser :

```
Nome_servidor:alias
```

Seguindo esta estrutura, no exemplo abaixo, a seguinte string de conexão solicitaria ao servidor Firebird sendo executado em um computador Linux de nome "Servidor" para localizar e conectar ao cliente o banco de dados armazenado no caminho identificado como "fbd2" em aliases.conf :

```
Servidor:fbdb2
```

Nomear bases de dados no Windows

Note que agora a extensão recomendada para bases de dados no Windows ME e XP é ".FDB", para evitar possíveis conflitos com o recurso de Recuperação de Sistema do Windows. Caso este procedimento não seja observado nestas plataformas, ocorrerão os conhecidos problemas de atraso na primeira conexão à base de dados em que os arquivos primários e/ou secundários tenham a extensão convencional ".gdb".

Equipes de Desenvolvimento Firebird

Programador	País	Tarefas principais
Dmitry Yemanov	Federação Russa	Coordenador de Versões; melhorias no DSQL e PSQL; implementação do Embedded Server; diversos aperfeiçoamentos do meta dados; Alias para base de dados; trigger múltipla ação; tipo de dado BigInt; novas variáveis de contexto; servidor Classic para Windows; inúmeras resoluções de erros.
Nickolay Samofatov	Federação Russa	Especificação e implementação de funcionalidades SQL (SAVEPOINTs, Bloqueio pessimista); melhorias no meta dados; re-implementações do Servidor; pesquisa e correção de erros; problemas de arquitetura; habilitação de Serviços API no Classic em Linux; melhoria do desempenho; versões Classic para Linux.
Arno Brinkman	Países Baixos	Melhorias do Otimizador; diversas novas funcionalidades DSQL.
Claudio Valderrama	Chile	Avaliação do Código; pesquisa e correção de erros; melhorias do PSQL; Correção, especificação e implementação de UDFs.
Alex Peshkoff	Federação Russa	Novas funcionalidades PSQL e DSQL; autor e coordenador de funcionalidades de segurança; correção do código; compilações Superserver para Linux
Mike Nordell	Suécia	Conversão do código do Firebird para C++; melhoria do desempenho; transporte de funcionalidades; pesquisa e correção de erros.
Blas Rodriguez Somoza	Espanha	Programador de novos sets de caracteres; profunda limpeza do código e remodelação da árvore; compilações MinGW.
Roman Rokytskyy	Alemanha	Programador e co-coordenador do Jaybird.
David Jencks	U.S.A.	Especificador e co-coordenador do JayBird; Especificação das ferramentas de documentação do Firebird.
Carlos Guzman Alvarez	Espanha	Desenvolvedor e coordenador do .NET provider para o Firebird.
John Bellardo	U.S.A.	Implementou interface PLUG-IN para o set de caracteres; coordenador de compilações Darwin; implementação inicial do novo modelo de memória.
Erik Kunze	Alemanha	Pesquisa e correção de erros; limpeza de código; versões SINIX-Z.
Dmitry Sibiryakov	Federação Russa	Limpeza de código; compilações MinGW.
Pavel Cisar	República Checa	Compilações Linux (Versão 1.0); especificador / coordenador de ferramentas QA.
Ann Harrison	U.S.A.	Correção de erros; consultoria técnica; aumento do número máximo de índices.

Programador	País	Tarefas principais
Mark O'Donohue	Austrália	Funcionalidades de linha do isql; correção de erros; compilações boot (Versão 1.0); correção do código (Versão 1.0).
Paul Reeves	França	QA; instaladores Win32; painel de controle padrão Win32.
Ignacio J. Ortega	Espanha	Adicionadas funcionalidades de PLAN para triggers; limpeza de código.
Konstantin Kuznetsov	Federação Russa	Compilações Intel Solaris.
Olivier Mascia	Bélgica	Re-implementação de serviços de instalação no Win32.
Peter Jacobi	Alemanha	Aperfeiçoamentos, atualizações de sets de caracteres.
Tilo Muetze	Alemanha	Coordenador do projeto de documentação Firebird.
Paul Vinkenoog	Países Baixos	Coordenador do projeto de documentação do Firebird; correções do UDF.
Artur Anjos	Portugal	Melhorias no painel de controle Win32; desenvolvedor do Gerenciador de Configuração do Firebird; Internacionalização do mesmo.
Achim Kalwa	Alemanha	Melhorias no programa do painel de controle do Win32
Sean Leyne	Canada	Organizador do Bugtracker; limpeza de código.
Ryan Baldwin	U.K.	Programador do driver Jaybird Tipo 2.
Sandor Szollosi	Hungria	Implementador de collations dos sets de caracteres.
Dmitry Kuzmenko	Federação Russa	Resolução de erros de GSTAT.
Artem Petkevych	Ucrânia	Resolução de erros para 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	Compilações FreeBSD.

Programador	País	Tarefas principais
Neil McCalden	U.K.	Compilações Solaris.
Grzegorz Prokopsi	Hungria	Compilações Debian.
Paul Beach	U.K.	Compilações HP-UX.
Geoffrey Speicher	U.S.A.	Compilações 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 ANTES!

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

- Tenha certeza de estar logado como Administrador (não se aplica ao Win9x ou ME)

- Todos os modelos de Servidor, Super, Classic e Embedded, bem como as ferramentas de administração e o cliente podem ser instalados a partir do programa de instalação para Windows. Para uma instalação completa de versão, é altamente recomendável usar o instalador se houver um disponível.

- Use o **gbak** para fazer uma cópia reserva da sua base de dados de segurança **isc4.gdb**. Você poderá restaurá-la mais tarde com o nome de **security.fdb.**

- Caso utilize configurações específicas no **ibconfig** poderá haver algumas configurações que você queira converter para os parâmetros equivalentes no **firebird.conf**. Estude as notas sobre o firebird.conf para poder definir o que pode ser simplesmente copiado e o quais os parâmetros precisarão ser adaptados para a nova Sintaxe.

- Caso determinados arquivos de configuração existam no diretório de instalação eles serão mantidos pelo programa de instalação e SOBREPOSTOS se você descompactá-los de um kit zipado para o local padrão. Estes arquivos são:
  security.fdb
  firebird.log
  firebird.conf
  aliases.conf

- Todas as versões podem ser instaladas a partir de um arquivo zip. Este método será mais rápido do que através do instalador caso você esteja familiarizado com instalações do Firebird 1.5 a partir de arquivos zip. Todavia poderá ser uma tarefa áspera para um iniciante em Firebird.

- Pressupõe-se que:
  1 Você compreende como funciona a sua rede.
  2 Você compreende porque um sistema cliente/servidor precisa ter servidor e clientes.
  3 Você leu todo o resto destas notas da versão — ou pelo menos tem a noção de que precisará lê-las se algo parecer estar incorreto.
  4 Você sabe como acessar o grupo de discussão do firebird-support quando 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 acha que poderá desejar retornar a ela, estabeleça um ponto de retorno antes de começar.

- Use a versão existente do GBAK para efetuar uma cópia dos seus arquivos de banco de dados em formato transportável.

- Vá até o diretório de Sistema e faça uma cópia de reserva do gds32.dll. Recomendamos que renomeie a cópia como "gds32.dll.ib5" ou "gds32.dll.fb103", ou algo igualmente informativo; ou esconda-o em um outro diretório.

- Poderá também ser uma boa idéia fazer uma cópia de segurança da biblioteca de execução do Microsoft C++, msvcp60.dll. O programa de instalação não deverá sobrepor a sua versão deste arquivo, mas algo de anormal pode ocorrer.

- **PARE QUALQUER SERVIDOR FIREBIRD OU INTERBASE QUE ESTEJA EM EXECUÇÃO**
  O programa de instalação tentará detectar versões prévias do Firebird ou InterBase instaladas e/ou em execução. Em uma instalação que não o utilize, você estará por contra própria!

- O local padrão do Firebird 1.5 será C:\Arquivos de Programas\Firebird_1.5. Se a sua versão anterior já estiver instalada neste diretório e você tiver a intenção de usar os locais padrões, renomeie o diretório existente.

- Para instalar o Firebird como serviço: caso pretenda usar a nova funcionalidade de **login seguro**, crie um "usuário de serviço firebird" no sistema — com nome e senha à sua escolha — como um usuário normal com os privilégios apropriados. Você deverá ler primeiro o documento nomeado README.instsvc.txt. Se você possuir um kit compactado, poderá encontrá-lo no diretório /doc a partir da raiz do arquivo zip. Caso não possua um kit compactado, o arquivo só estará acessível ao final da instalação. Você pode ler o mesmo documento no seguinte endereço:
http://cvs.sourceforge.net/viewcvs.py/firebird/firebird2/doc/README.instsvc

LEIA EM SEGUIDA !

Uma das metas de especificação do Firebird 1.5 era preparar o caminho para múltiplas instalações do Servidor. Isto permitirá aos usuários executar diferentes versões lado a lado. O Firebird 1.5 suporta isto, embora de forma não muito bem documentada, e ainda requerendo muitas intervenções de um usuário experiente. Futuras versões do Firebird tornarão este processo menos complicado. Mas neste meio tempo, o Firebird 1.5 precisará preparar o terreno. O que nos força a confrontar o problema da instalação da biblioteca. Ao mesmo tempo, a Microsoft tem tomado as suas próprias medidas para controlar a instalação de bibliotecas de diferentes versões. Analisadas em conjunto, estas duas políticas implicam em um novo procedimento para a instalação de bibliotecas para o Firebird 1.5 e futuras versões.

Instalação das bibliotecas de sistema Microsoft

Os problemas associados com a instalação de diferentes versões das bibliotecas de sistema Microsoft são tão conhecidos, que receberam o codinome de “Inferno das Dlls”.

A partir do lançamento do Windows 2000, a Microsoft tornou quase impossível a atualização das DLLs de sistema. Para contornar este problema, a Microsoft recomenda agora que cada aplicação instale cópias locais das bibliotecas que são requeridas.

O Firebird 1.5 segue esta prática e coloca as bibliotecas requeridas no diretório \bin junto com o Servidor.

Instalação do fbclient.dll

A partir da versão 1.5 gds32.dll não é mais utilizada como biblioteca cliente. Agora ela chama-se fbclient.dll. Dados os problemas enfrentados pela própria Microsoft com o inferno das DLLs, não fazia muito sentido continuar instalando a biblioteca cliente do Firebird no diretório de sistemas. Como

desejamos permitir a instalação de múltiplas instâncias de Servidor estaríamos criando nosso próprio inferno se prosseguíssemos com a prática de instalar o cliente no diretório de sistemas. Assim, a partir da versão 1.5 a biblioteca cliente residirá na pata \bin juntamente com todos os outros binários.

Uma nova chave de registro foi adicionada e todas as aplicações compatíveis com o Firebird deverão utilizá-la para localizar a versão correta que elas desejam usar. A nova chave é :

HKEY_LOCAL_MACHINE\SOFTWARE\Firebird Project\Firebird Server\Instances

Firebird garantirá a existência de pelo menos uma entrada neste registro. Ela será conhecida como :

"DefaultInstance"

e conterá o caminho para o diretório da (sim, você adivinhou) instalação padrão. Aplicações que não tenham interesse por uma instalação em particular poderão usar a instância padrão para localizar a fbclient.dll.

Versões futuras do Firebird irão encontrar novas entradas na chave Instances. Aplicações serão capazes de enumerar as entradas do registro e determinar a biblioteca da instância que elas desejam carregar.

Suportando drivers e aplicações legadas

Tradicionalmente, aplicações que usam Interbase ou Firebird têm esperado carregar o cliente GDS32.dll encontrado na pasta de sistemas. O Firebird 1.5 fornece uma ferramenta chamada "instclient.exe" que pode instalar um clone do fbclient.dll na pasta de sistemas do Windows. Este clone é modificado instantaneamente para que sua informação de versão informe '6.3', para prover compatibilidade com aplicações prévias que verificam a versão do arquivo GDS32.dll e não tem como interpretar valores como '1.5'.

Durante o processo de instalação o instalador procura por instalações anteriores do Interbase e do Firebird. Caso nada seja encontrado será copiada a nova GDS32.dll. Caso seja detectada qualquer versão possível do Interbase ou do Firebird instalado a GDS32.dll não será copiada. De qualquer forma, a ferramenta "instclient.exe" poderá ser utilizada mais tarde.

Pretende-se que futuras versões do Firebird não tentem mais instalar gds32.dll na pasta de sistemas e que em um determinado momento ela seja removida da distribuição.

A ferramenta "instclient.exe" pode também instalar a própria FBCLIENT.DLL na pasta de sistemas do Windows, para satisfazer aplicações e ferramentas que precisem carrega a partir desta pasta.

Uso do instclient.exe:

instclient	i[install] [-f[orce]] *biblioteca* q[uery] *biblioteca* r[emove] *biblioteca*

onde biblioteca é: fbclient | gds32.

'-z', que pode ser usada com qualquer opção, mostra a versão.

A informação de versão e a contagem de compartilhamento da biblioteca são tratadas automaticamente. O uso da opção –f[orce] desconsidera a verificação de versão.

NOTA A opção –f[orce] pode danificar outra versão de Firebird ou Interbase já instalada. Você poderá ter que reiniciar o sistema para finalizar a instalação da cópia.

Para mais detalhes, veja o documento README.Win32LibraryInstallation.txt que está localizado no diretório raiz ou na pasta \doc da sua instalação.

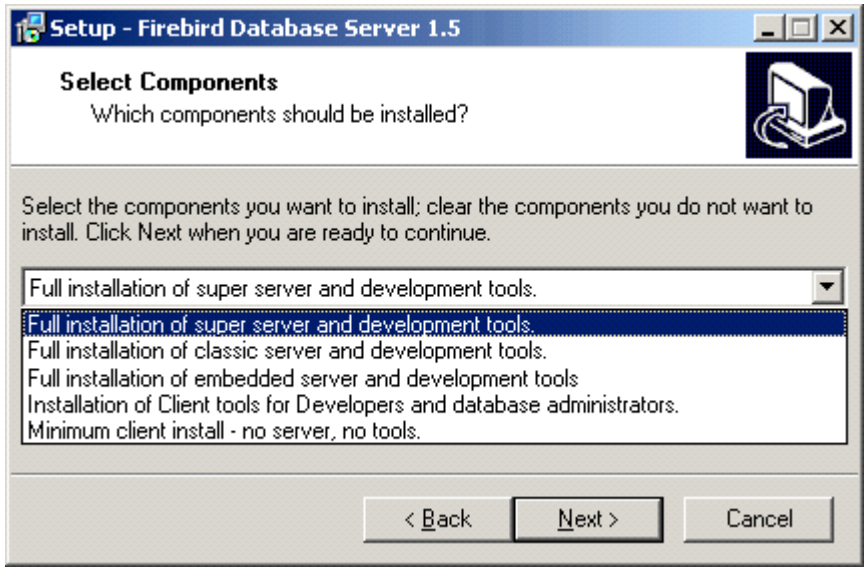
Desinstalação de versões de teste

Deve ser destacado que o instalador remove qualquer fbclient.dll que seja encontrada na pasta de sistemas. O instalador também remove a chave obsoleta de registro HKLM\Software\FirebirdSQL.

Usando o programa de Instalação Firebird para o Win32

Esta é a instalação mais simples.

Basta executar o instalador e responder os diálogos de forma apropriada. Depois ter respondido por volta de diálogos, será visualizada uma caixa com uma lista "drop-down" que, quando aberta, será similar a esta. Esta realmente é a última oportunidade para selecionar o tipo de instalação que você quer.



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

Serviço ou aplicação?

Se você selecionar a versão de instalação Superserver ou Classic, e a versão do seu SO suportar serviços, você deverá indicar se pretende executar o Firebird como serviço ou como aplicação. A menos que tenha uma necessidade específica para executar o servidor como aplicação, opte por serviço.

Manual ou automático?

Com a opção automática, o Firebird irá partir toda vez que o computador onde foi instalado seja ligado. Com a opção manual você iniciará o servidor apenas quando requisitar.

Opção do Guardian

O Guardian é um utilitário que pode ser executado "em cima" do Superserver e reiniciá-lo caso ele caia por qualquer razão. No desenvolvimento, você pode optar por não usá-lo. Em ambiente de trabalho ele poderá remediar uma situação em que o aplicativo pare de funcionar e o DBA (administrador) não se encontre disponível para o reiniciar.

Diretório de Instalação (Root)

Se decidir não usar o diretório padrão de instalação, navegue até o local previamente criado; ou simplesmente digite o seu caminho completo. A pasta destino não precisa existir previamente: o programa de instalação pedirá a sua confirmação e irá criá-la, se necessário.

Eventualmente, os diálogos irão parar e o servidor irá, ou partir silenciosamente ou pedir a sua permissão para reiniciar a máquina — a tarefa de reinicialização será necessária caso o programa de instalação tenha sobreposto a msvcp60.dll, ou se uma gds32.dll mais antiga estava carregada quando o programa de instalação se iniciou.

Instalando o Superserver a partir de um kit compactado

A instalação do FB 1.5 é similar em princípio a das versões anteriores.
Caso não possua o programa específico de instalação (distribuído em separado) os passos serão os seguintes:

- Use o unzip para extrair os arquivos para uma pasta separada (como alguns nomes foram alterados, não faz muito sentido extrair os arquivos da v1.5 para a mesma pasta do IB/FB1)
- Vá até o diretório para o qual os arquivos foram extraídos e dentro dela localize a pasta bin (daqui por diante, chamaremos de <raiz> o diretório onde os arquivos da v1.5 foram instalados).
- execute “instreg.exe”
  instreg.exe install

faz com que caminho do diretório de instalação acima seja escrito na chave de registro (HKLM\Software\Firebird Projet\Firebird Server\Instances\DefaultInstance)

- se quiser registrar um serviço, execute também “instsvc.exe“:
  instsvc.exe install.
- Opcionalmente, faça uma cópia de fbclient.dll e gds32.dll para a pasta de sistema do OS.

Instalando o Classic Server a partir de um kit compactado

Para instalar a modalidade CS, a única diferença será indicar um parâmetro adicional para o instsvc.exe:

instsvc.exe install -classic

Repare que isto implica que só é possível ter uma arquitetura do Servidor, ou fbserver.exe (Superserver) ou fb_inet_server.exe (o processo principal o Classic), instalada como serviço.

Instalação simplificada

Caso não precise registrar um serviço, então você não deverá executar o “instreg.exe” ou o “instsvc.exe”. Basta extrair o conteúdo do arquivo zip em uma pasta própria executar o servidor:

fbserver.exe -a

Ele considerará o diretório pai como diretório raiz da instalação.

Desinstalação

Para remover o FB 1.5 sem um desinstalador Windows você deverá:

- parar o servidor.
- executar "instreg.exe remove".
- executar "instsvc.exe remove".
- apagar o diretório de instalação.
- apagar os clientes fbclient.dll e gds32.dll da pasta de sistema do SO

Instalando o servidor Embedded a partir de um kit compactado

O servidor embedded interliga um cliente a um servidor com todas as funcionalidades numa biblioteca dinâmica (fbembed.dll). Ele tem exatamente as mesmas funcionalidades que o Superserver e exporta os mesmo pontos de entrada da interface API padrão do Firebird.

Registro As entradas do Registro para o Firebird (onde o servidor normalmente procura pela localização da pasta de instalação) são ignoradas. O diretório raiz servidor embedded é a pasta um nível abaixo daquela onde o arquivo binário (biblioteca) se encontra.

Acesso à Base de Dados Apenas o acesso “verdadeiramente local” é permitido. O servidor “embedded” não possui suporte para protocolos remotos, então nem mesmo o acesso via "localhost" funcionará.

Autenticação e segurança A base de dados de segurança (security.fdb) não é utilizada pelo servidor “embedded” e portanto não é necessária. Qualquer usuário pode conectar-se a qualquer banco de dados. Assim, como tanto o servidor quanto o cliente são executados no mesmo espaço de endereço (local), a segurança é uma questão de acesso físico.

Os privilégios SQL são validados, tal como nas outras modalidades.

Compatibilidade Qualquer quantidade de aplicações pode ser executada em um servidor “embedded” sem quaisquer conflitos. Ter um servidor IB/FB sendo executado simultaneamente também não causará qualquer problema.

Mas é importante ressaltar que não é possível acessar simultaneamente o mesmo banco de dados a partir de múltiplos servidores “embedded”, porque eles compartilham a arquitetura SuperServer e precisam de acesso exclusivo ao banco de dados.

Estrutura de arquivos para o Servidor Embedded

Basta copiar fbembed.dll para a pasta onde se encontra a aplicação desejada. E então renomeá-lo para fbclient.dll ou gds32.dll, dependendo de como o programa se conecta ao banco de dados. Efetue cópias usando os dois nomes caso precise usar as ferramentas de Administração (isql, gbak, etc.)

Também deverão ser copiados os arquivos firebird.msg, firebird.conf (se necessário) e ib_util.dll na mesma pasta.

Se bibliotecas externas forem requeridas pela sua aplicação, p.ex. suporte INTL (fbintl.dll) ou bibliotecas UDF, elas deverão ser colocadas em uma pasta separada do aplicativo. Para poder utilizá-las, coloque-as em uma pasta que reproduz a estrutura de diretórios padrões do Firebird, i.e., em subdiretórios nomeados /intl e /udf criados diretamente abaixo do diretório onde foram copiados os arquivos básicos do Firebird.

Exemplo

```
D:\my_app\app.exe
D:\my_app\gds32.dll (fbembed.dll renomeado)
D:\my_app\fbclient.dll (fbembed.dll renomeado)
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. Ela usará o servidor "embedded" como uma biblioteca cliente e terá acesso aos bancos de dados locais.

NOTA Desde que você mantenha a estrutura de diretórios de acordo com estas regras, não é necessário alterar o parâmetro RootDirectory no firebird.conf. Porém, se você decidir distribuir sua aplicação com o servidor "embedded" com uma estrutura diferente de diretórios, tome o cuidado de ler antes o documento README_embedded.txt da distribuição do Servidor "embedded", para instruções adicionais de configuração

Desinstalação

A rotina de desinstalação do Firebird mantém e renomeia os seguintes arquivos chaves:

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

"nnnn" é o número da versão da instalação anterior.
Nenhuma tentativa é feita de remover arquivos que não tenham feito parte da instalação original. Arquivos compartilhados como o fbcliente.dll e o gds32.dll serão removidos se o contador de compartilhamento ("share count") indicar que nenhuma outra aplicação está usando-os.
As chaves do Registro criadas serão removidas.

Outras Notas

Winsock2

O Firebird requer o WinSock2. Todas as plataformas Win32 deverão ter o WinSock2, exceto o Win95. É feita uma verificação da presença da biblioteca Winsock2 durante a instalação. Caso ela não seja encontrada a instalação falhará. Para fazer uma atualização 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 Profissional) existe uma funcionalidade denominada Recuperação de Sistema, que provoca uma auto-atualização ("backup caching"?) de todos os arquivos no sistema com extensão ".gdb". O efeito é uma lentidão no acesso aos bancos de dados InterBase/Firebird, a ponto de uma quase paralisia, porque é efetuada uma cópia de segurança destes arquivos cada vez que ocorre uma operação de E/L. (No XP não há opção de Recuperação de Sistema nos Servidores .NET).

Um arquivo no diretório Windows do ME, c:\windows\system\filelist.xml, contém os "tipos protegidos de arquivo", e a extensão ".gdb" está relacionada. Charlie Caro, inicialmente sugeriu que se removesse a extensão GDB da seção "includes" deste arquivo. Porém, desde então, ficou comprovado que o WinME consegue restaurar a lista. No XP, não é sequer possível editar o arquivo filelist.xml.

Em ME, as soluções permanentes que podemos sugerir são:

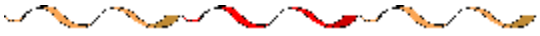
- Utilizar a extensão FDB (Firebird DB) para os arquivos de Banco de Dados
- Mover a o Banco de Dados para a pasta C:\Meus Documentos, que é ignorada pela Recuperação de Sistema.
- Desativar completamente o recurso de Recuperação de Sistema (verifique as instruções na documentação do Windows).

No Windows XP Edições Home e Profissional você poderá mover as bases de dados para uma partição separada, e configurar o "System Restore" para excluir esse volume.

O Windows XP utiliza uma cópia inteligente, assim a lentidão verificada no Windows ME pode não ser tão significativa no XP, pelo menos para arquivos menores. Para arquivos maiores (por exemplo: Bancos de Dados Firebird!) não parece existir uma solução melhor, se os seus arquivos ".gdb" estiverem localizados no sistema de arquivos geral.

Estamos tentando obter uma descrição precisa do problema e uma solução comprovada para comunicarmos aqui.Se você acha que pode ajudar com uma descrição do problema ou com uma solução alternativa, por favor envie uma mensagem para o grupo de discussão ib-support ou firebird-devel em news://news.atkin.com/

O comportamento do desligamento do Windows XP é um conhecido buraco negro. Há uma pausa significativamente longa enquanto o servidor tenta parar o serviço. Durante este tempo a tela indica que o Firebird está sendo executado como uma aplicação. Este problema parece afetar só ao Windows XP e ele só acontece se o Guardian não estiver sendo utilizado para parar o serviço do Firebird. Portanto, usá-lo é a forma de contornar o problema até que uma melhor solução seja encontrada.



Instalação no UNIX / Linux

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

O servidor Firebird é disponibilizado em duas modalidades, Classic que é executada como um serviço, e SuperServer que é executado como um daemon em segundo plano. O usuário que estiver tendo seu primeiro contato com o Firebird pode usar qualquer versão, embora seja provável que a versão clássica ofereça uma melhor plataforma para experiências iniciais com o Firebird.

NOTAS – LEIA ANTES

1) É preciso ter direitos de usuário root para instalar o Firebird.

2) A instalação no Linux requer que o pacote glic instalado tenha versão igual ou superior a gblibc-2.25.
3) Para uma verificação superficial de compatibilidade Linux, consulte esta tabela, mas não a considere palavra final sobre o assunto. Distribuições de binários Linux podem variar dependendo de quando e onde elas foram montadas.
4) Garanta que os editores de pacote, 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. (NOTA Esta dependência pode no futuro ser substituída por sed.)

INSTALAÇÃO EM LINUX

As instruções a seguir referem-se a instalação da Classic. Para instalação da edição Superserver o CS será substituído pelo SS. Por exemplo, o pacote FirebirdCS-1.5.0-nnnn.i86.rpm será substituído por FirebirdSS-1.5.0-nnnn.i86.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.i86
$./install.sh
```

* ou `FirebirdSS-1.5.0-nnnn`

O que uma instalação Linux fará

A instalação Linux irá:

1. Tentar parar algum servidor que esteja em execução.
2. Adicionar o usuário 'firebird' e o grupo 'firebird' se eles ainda não existirem.
3. Instalar a aplicação na pasta /opt/firebird e criar direcionamentos para as bibliotecas em /usr/lib e para os cabeçalhos em /usr/include
4. Automaticamente adicionar gds_db na porta 3050 em /etc/services se a entrada ainda não existir
5. Automaticamente adicionar localhost.localdomain e HOSTNAME em /etc/host.equiv
6. A versão SuperServer instala também um script de partida em /etc/rc.d/init.d/firebird.
7. A versão Classic instala um script de inicialização do servidor firebird em /etc/xinet.d/firebird, ou para sistema inetd antigos, uma entrada é adicionada no arquivo /etc/inetd.
8. Especificamente na distribuição SuSE um novo direcionamento rcfirebird é criado em /usr/bin para o script init.d, e uma entrada Firebird é criada em /etc/rc.config.
9. Inicia o serviço / servidor. O Firebird deve partir automaticamente em runlevel 2, 3 e 5.
10. É gerada randomicamente e configurada uma senha para o usuário SYSDBA que será armazenada em /opt/firebird/SYSDBA.password.
11. É adicionada uma entrada em aliases.conf para o banco de dados exemplo, employee.fdb.

Testes da instalação Linux

Passo 1 – Acessando o banco de dados

```
$cd /opt/firebird/bin

$isql -user sysdba -password <senha*>
```

```
>connect localhost:employee.fdb /*este é um acesso através do alias
>select * from sales;
>select rdb$relation_name from rdb$relations;
>help;

>quit;
```

*A senha foi gerada para você na instalação. Ela pode ser obtida no arquivo /opt/firebird/SYSDBA.password.

Passo 2 – Criando um banco de dados

A partir da versão 1.5, o servidor firebird será executado por padrão pelo usuário 'firebird'. Embora esta sempre tenha sido a recomendação, anteriormente o padrão era executar o servidor pelo usuário 'root'. Quando em execução pelo usuário 'root', o servidor possuía direitos plenos para ler, criar e apagar bancos de dados em qualquer local do sistema de arquivos POSIX. Por questões de segurança, o serviço deveria ter sua capacidade de ler, criar e apagar arquivos limitada.

Enquanto a nova configuração é melhor da perspectiva da segurança, ela requer que cuidados especificos sejam tomados na criação de novos bancos de dados:

1) O usuário 'firebird' deve possuir diretos de escrita na pasta onde será criado o banco de dados.
2) O valor recomendado para o atributo DatabaseAccess no arquivo /opt/firebird/firebird.conf deverá ser None, para permitir o acesso apenas através das entradas do arquivo aliases.conf.
3) Usar entradas no aliases.conf para abstrair dos usuários a localização física dos bancos de dados. Mais informações sobre alias aqui.

Os procedimentos para criação de novos bancos variarão conforme a configuração escolhida. No entanto, recomendamos os seguintes passos com esta sugestão de configuração:

1) Se a pasta do usuário 'Firebird' ainda não foi criada, mude para o usuário root e crie a pasta:

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

2) Crie fisicamente o novo banco e configure uma entrada de alias para ele. Como usuário root ou firebird, execute o seguinte script:

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

(O uso é: createDBAlias.sh <nome do banco> <caminho do banco>)

3) Como uma alternativa ao item 2, os procedimentos do script createDBAlias.sh podem ser executados um a um por:

```
$vi /opt/firebird/aliases.conf
e acrescente uma linha no fim do arquivo:
test.fdb /var/firebird/test.fdb
```

4) Então crie o Banco de Dados:

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

5) Se o valor de DatabaseAccess em /opt/firebird/firebird.conf estiver definido com Full ou um caminho restrito (por exemplo: DatabaseAccess=/var/firebird) outra alternativa para o passo 2 é criar fisicamente o banco de dados diretamente, utilizando o caminho absoluto junto com o nome do arquivo:

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

Caso use esta opção, o arquivo do banco de dados também poderá ser acessado diretamente sem que seja feita uma entrada no arquivo de alias.

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

*A senha foi gerada para você na instalação. Ela pode ser obtida no arquivo /opt/firebird/SYSDBA.password.

Dicas e alguns Scripts úteis

Complementando os arquivos padrões de instalação, estão disponíveis na pasta bin desta versão os seguintes scripts:

- **changeDBAPassword.sh** — Altera a senha do usuário Firebird SYSDBA e, se necessário, altera o script de inicialização /etc/rc.d/init.d/firebird para que seja utilizada também a nova senha.

- **createAliasDB.sh** — Uso: createAliasDB.sh <dbname> <dbpath>. Este script cria fisicamente um novo banco de dados e adiciona uma entrada no arquivo aliases.conf.

- **fb_config** — Um script que pode ser usado com makefiles para gerar os caminhos de include requeridos e incluir as diretivas de lib para a versão instalada do Firebird. fb_config -help mostrará a lista completa de opções.

- **changeGdsLibraryCompatibleLink.sh** — somente Classic — Alterna o direcionamento da biblioteca cliente para libgds.so entre a multi processada libfbclient.so e a uni processada libfbembed.so que permite o acesso embedded direto a um banco de dados. Para manter a compatibilidade com instalações anteriores, libgds.so por padrão aponta para libfbembed.so.

- **Acesso direto ou embedded aos arquivos de banco de dados -** A instalação clássica oferece um modo embedded de acesso que permite que programas acessem diretamente o arquivo de banco de dados. Para operar neste modo, um usuário com permissões de banco de dados precisará de acesso privilegiado a alguns arquivos de configuração e de estado do firebird.

- **Obtendo acesso ao banco de dados:** Agora que 'firebird' (e não root) é o usuário padrão para a execução do aplicativo, você precisará saber como **incluir um usuário no grupo firebird** e como habilitar o seu acesso aos bancos de dados. Isto está documentado nas notas leiame, mas os seguintes passos devem levá-lo até onde você precisa chegar:

Para adicionar um usuário (por exemplo, skywalker) ao grupo firebird, o usuário root precisará fazer:

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

Na próxima vez que o usuário 'skywalker' entrar no sistema ele poderá começar a trabalhar com os bancos de dados firebird.

Para listas dos grupos a que um usuário pertence, digite a seguinte linha de comando:

```
$ groups
```

- **Problemas NTPL com distruibuições Linux recentes:** A nova NPTL (Native POSIX Thread Library) do Red Hat 9 (até o momento) causará problemas para o Superserver e os programas compilados localmente, incluindo as ferramentas de Administração. Gbak, em partircular,indicará um erro de "Broken Pipe". Para contorná-lo:

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

Isto resolve a instância do Servidor. Você precisará também ter uma variável de ambiente definida no seu ambiente local. Então:

2. Adicione o seguinte em /etc/profile, para garantir que todo usuário o utilizará como ferramenta de linha de comando.

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

Desinstalando em Linux

Caso precise fazer uma desinstalação, faça-a como usuário root. Os exemplos a seguir base utilizam a extensão CS do Servidor Classic, mas são válidos também para o Superserver, substituindo-se o CS pelo SS.

Para pacotes rpm:

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

or para instalações .tar.gz:

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

Instalação do Firebird Classic & SuperServer no Solaris 2.7 Sparc

Não está atualmente disponível. Por favor, consulte as Notas da Versão 1 como uma referência para instalações 1.5.

Instalação do Firebird Classic no MacOS X / Darwin

Não está atualmente disponível. Por favor, consulte as Notas da Versão 1 como uma referência para instalações 1.5.

Build ou Instalação do Firebird no FreeBSD

Não está atualmente disponível. Por favor, consulte as Notas da Versão 1 como uma referência para instalações 1.5.

Configurando a porta de serviço no cliente e no servidor

Por padrão um servidor Firebird observa a porta 3050 por requisições de conexão de clientes. Ela é registrada com o nome de gds_db. A boa nova é que, se você quiser manter estes padrões você não precisará fazer nada, quer no servidor ou nos clientes para configurar a porta de serviço.

Você pode optar por uma porta diferente, por um nome de serviço diferente ou alterar ambos. Você pode precisar fazer isto se 3050 for requerido por um outro serviço, por exemplo, um serviço concorrente gds_db configurado para outra versão de servidor Firebird ou InterBase®. Há diversos meios para se modificar os valores padrões. Tanto o servidor como clientes precisam ser configurados para se alterar o nome ou a porta de serviço, ou ambos, através de um dos seguintes métodos:

0 Na string de conexão do cliente.
1 No comando para iniciar o executável do servidor.
2 Ativando os parâmetros RemoteServicePort ou RemoteServiceName no firebird.conf (V.1.5 em diante)
3 No daemon de configuração (para Classic em sistemas POSIX)
4 Através de uma entrada no arquivo de serviços.

Antes de examinar cada uma destas técnicas, será útil examinarmos a lógica utilizada pelo servidor para determinar a porta que será observada e pelo cliente para definir a porta que será utilizada para o envio das suas requisições.

Como o servidor determina a porta observada

O executável do servidor possui um parâmetro de linha opcional (-p) que pode indicar ou o número da porta que será observada ou o nome do serviço da porta que será observada. Neste ponto, se o parâmetro estiver presente, quer a porta 3050, quer o nome do serviço (gds_db) será substituído pelo argumento fornecido através da opção –p.

Depois, ou inicialmente, se não for usada a opção –p, o servidor verifica no firebird.conf os parâmetros RemoteServiceName e RemoteServicePort:

- Se ambos estiverem comentados pelo marcador "#" então é assumido o valor padrão e nenhuma alteração ocorrerá. Qualquer argumento recebido através da opção -p permanece e valores não fornecidos permanecem com o valor padrão.

- Se RemoteServiceName tiver sido ativado, mas não RemoteServicePort, então o nome da porta de serviço será substituído, somente se ele já não tiver sido alterado anteriormente pela opção –p.

- Se RemoteServicePort foi ativado, mas não RemoteServiceName, então o número da porta será substituído, somente se ele já não tiver sido alterado previamente pela opção –p.

- Se ambos RemoteServicePort e RemoteServiceName foram ativados, então RemoteServiceName tem precedência, se já não houve alteração prévia através do argumento –p. Se já houve modificação prévia, o valor de RemoteServiceName é ignorado e o valor de RemoteServicePort substitui 3050.

- Neste ponto, se tiver havido substituição, quer do nome do serviço, quer do número da porta, ambas as versões do servidor, v. 1.0 e v.1.5, executarão uma verificação do arquivo services para procurar uma entrada com a combinação correta de nome de serviço e número de porta. Caso a combinação seja encontrada tudo estará perfeito. Se não, se o nome do serviço não for gds_db uma exceção ocorrerá e o servidor interromperá sua partida. Se gds_db for o nome da porta de serviço e nenhuma outra porta puder ser relacionada ao serviço, ele será automaticamente mapeado para 3050.

Se o nome padrão do serviço for modificado, você terá que adicionar uma entrada no arquivo de Serviços – veja o tópico mais a frente, *Configurando o arquivo de serviços*.

Usando a opção de substituição –p

Note que esta opção já estava disponível no Firebird 1.0.x também, mas não tinha sido ainda documentada.

Iniciar o servidor com o parâmetro opcional –p permite substituir ou o número padrão da porta (3050) ou o nome padrão do serviço (gds_db) que o servidor utiliza para observar as requisições de conexão. O comando permite substituir um deles, mas não ambos. A partir da versão Firebird 1.5 você poderá combinar a opção –p com configurações do firebird.conf para possibilitar a alteração tanto do número da porta quanto do nome do serviço.

Syntax for TCP/IP

```
server-command <other switches> -p port-number | service-name
```

Por exemplo, para iniciar o Superserver como uma aplicação e modificar o nome do serviço de gds_db para fb_db:

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

Ou, para alterar a porta de 3050 para 3051:

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

Syntax for WNet redirection

Em uma rede WNet, substitua a sintaxe do argumento –p acima pela seguinte sintaxe "barra-barra-ponto-arroba":

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

ou

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

Classic no POSIX: o daemon *inetd* ou *xinetd*

Assim como o servidor Firebird Classic em Linux ou UNIX, o daemon **inetd** ou **xinetd** é configurado para observar na porta padrão e para transmitir através do nome de serviço padrão. O script de instalação registrará a notação apropriada no arquivo de configuração /etc/inetd.conf ou /etc/xinetd.conf.

Você poderá alterar a configuração atual, se necessário. A seguir, um exemplo do que você encontrará no arquivo xinetd.conf ou inetd.conf após instalar um servidor 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
}
```

Caso você configure a porta de serviço para valores diferentes dos padrões, você precisará alterar xinetd.conf ou inetd.conf de acordo. Reinicie xinetd (ou inetd) com kill -HUP para assegurar que o daemon use a nova configuração.

NOTA Requisições de conexão falharão se ambos xinetd (ou inetd) e fbserver (ou ibserver) tentarem observar a mesma porta. Se a máquina host possuir esta configuração ambígua, será necessário configurar os parâmetros para que cada servidor tenha a sua própria porta de serviço.

Usando um parâmetro do arquivo de configuração

Você pode configurar tanto RemoteServiceName como RemoteServicePort no firebird.conf para modificar o número da porta padrão (3050) ou o nome padrão do serviço da porta (gds_db) que o servidor usa para aguardar requisições de conexão.

O servidor usará um parâmetro RemoteService* , mas não ambos. Caso você configure ambos, RemoteServicePort será ignorado em todos os casos, exceto quando o servidor for iniciado através de um comando que inclua a opção –p indicando a modificação do nome do serviço. Assim, você pode utilizar a opção –p e o parâmetro RemoteService* , em combinação, para alterar tanto o número da porta quanto o nome do serviço.

Se o valor padrão do nome do serviço for alterado, então será preciso fazer uma indicação no arquivo de Serviços.

CUIDADO! Se você ativar RemoteServiceName ou RemoteServicePort, mas mantiver os valores padrões inalterados, mesmo assim eles serão tratados como modificados. Então, será necessário fazer uma notificação explícita no arquivo de serviços com as configurações padrões de porta.

Configurando o cliente para que ele encontre a porta de serviço

Se você configurar o seu servidor com os valores padrões (serviço gds_db observando na porta 3050) nenhuma configuração adicional será requerida. Se o servidor estiver aguardando em uma porta diferente ou usando um nome diferente de serviço, a aplicação cliente e / ou a máquina onde ele é executado precisarão de alguma ativação de parâmetros para ajudar a biblioteca cliente do firebird a encontrar a porta que está sendo utilizada.
A string de conexão utilizada pelo cliente pode incluir informação sobre a porta do servidor observada. O cliente Firebird 1.5 pode opcionalmente usar uma cópia local do firebird.conf. Pode ainda ser preciso fazer algumas alterações no arquivo de serviços do cliente.

Usando a string de conexão

Se apenas o número da porta tiver sido alterado, inclua então o novo número da porta ou o nome do serviço na string de conexão. Isto funcionará com todas as versões do Firebird.

Sintaxe para conexões TCP/IP

Para se conectar a um banco de dados em um servidor chamado Alice, que está transmitindo na porta 3050, e com o nome serviço fb_db, a string de conexão deveria 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
```

Note que o separador entre o nome do servidor e a porta é uma barra invertida e não dois pontos. Os dois pontos antes do caminho físico ainda são requeridos.

Sintaxe para conexões WNet

Em uma rede WNet, use a notação estilo UNC:

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

Se o número da porta do serviço ou o nome do serviço tiver sido modificado, você precisará registrar a alteração no arquivo de serviços.

Usando a sintaxe de portas com alias de banco de dados.

Para conectar-se através de uma porta não padrão, utilizando o alias de um banco de dados, afixe o número da porta ou serviço ao nome do servidor, e não ao alias. Por exemplo, suponha que o alias do banco esteja definido no aliases.conf como:

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

A string de conexão da sua aplicação com o servidor 'alice' deveria ser:

```
alice/fb_db:rabbit
```
or
```
alice/3051:rabbit
```

Usando cópias do firebird.conf

A partir da versão do Firebird 1.5, você poderá manter opcionalmente uma cópia do firebird.conf do lado do cliente, também no diretório raiz da instalação, e configurar o RemoteServiceName ou o RemoteServicePort para ajudar o cliente a localizar a porta do servidor.

- Você pode configurar um dos dois parâmetros para complementar a outra configuração já alterada pela string de conexão; ou para substituir apenas o RemoteServiceName ou o RemoteServicePort sem utilizar a string de conexão para isto.

- Se você preferir evitar informar o nome do serviço ou o número da porta na string de conexão e o servidor estiver usando valores não padrões para ambas as configurações, você poderá configurar tanto RemoteServiceName como RemoteServicePort. Você deveria usar esta técnica caso a sua aplicação cliente precise manter a capacidade de se conectar com servidores Interbase ou Firebird.

Localização de componentes Firebird nos clientes

Quando você baseia uma aplicação no uso do firebird.conf em máquinas clientes, é importante que a biblioteca cliente saiba onde encontrar este arquivo. Você precisará definir uma pasta Raiz de instalação do Firebird e informar o sistema a sua localização. Use a variável de ambiente FIREBIRD para fazer isto. Alternativamente, os clientes do windows podem usar a chave de Registro do Firebird. Com uma configuração correta de cliente, você poderá também utilizar uma instância local do arquivo de mensagens.

Configurando o arquivo de serviços

Você não precisa modificar a configuração da porta de serviço se o seu Firebird ou os clientes usarem os valores padrões de instalação, gds_db na porta 3050. Se gds_db for o nome do serviço e ele não puder ser relaciona a qualquer outra porta, a 3050 será mapeada automaticamente.

Se você estiver configurando a porta de serviço para usar um número ou nome diferente, tanto o servidor quando os clientes deverão ser explicitamente atualizados para refletir a alteração da configuração. Tanto em Linux quanto em Windows, esta informação é armazenada no arquivo de serviços.

Localizando o arquivo de serviços

- No Windows NT/2000/XP/S2003, o arquivo é C:\windows\system32\drivers\etc\services.
- No Windows 95/98/ME, ele é C:\windows\services.
- No Linux/UNIX é /etc/services.

A notação para um serviço deve parecer com esta:

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

Abra o arquivo usando um editor de textos e adicione uma linha ou edite a notação gds_db existente como segue:

- Para um servidor ou cliente Firebird 1.0.x, altere ou o nome do serviço ou o número da porta para configurar como o servidor irá ser iniciado.

Para um servidor Firebird 1.5 ou mais recente, edite ou adicione uma linha, conforme requerido. Se houver um servidor Firebird 1.0 ou InterBase instalado no mesmo computador, mantenha as entradas requeridas por eles e adicione uma nova entrada para definir como o servidor Firebird 1.5 iniciará.

Informações Adicionais

Informação adicional sobre o Firebird pode ser encontrada em:
http://firebird.sourceforge.net/

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

Se estiver interessado em participar do desenvolvimento do Firebird, ou se quiser iniciar uma discussão sobre alguma possível falha, participe do grupo de desenvolvimento firebird-devel. Para inscrever-se, envie uma mensagem em branco para:

mailto:firebird-devel-request@lists.sourceforge.net

Com a palavra 'subscribe' no campo do assunto.

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

Para suporte técnico, participe do grupo de discussão firebird-support em :
http://www.yahoogroups.com/groups/firebird-support

Para suporte específico do InterClient e desenvolvimento e suporte em Java, use o grupo:
http://www.yahoogroups.com/groups/Firebird-Java

O grupo firebird-support trata questões técnicas sobre o Firebird e o InterBase(R). Por favor, leve os problemas e dúvidas envolvendo Delphi ou outras linguagens de programação para os fóruns apropriados.

A comunidade "open-source" mantém outros grupos de discussão sobre diferentes aspectos do desenvolvimento do Firebird. Por favor, verifique na seção "Mail Lists e Newsgroups" no Site da Comunidade Firebird.

Os grupos de discussão de desenvolvimento, a lista geral da comunidade, assim como outras listas de interesse para programadores Firebird e InterBase, estão espelhadas no servidor de Notícias em :
news://news.atkin.com/

Suporte Pago ao Firebird a nível mundial pode ser obtido através do IBPhoenix (os endereços de contato e os valores estão disponíveis em http://www.ibphoenix.com/). Alguns membros da equipe Firebird também estão disponíveis para suporte ou consultoria. Por favor, contate-os diretamente.

Serviços de Recuperação de Bases de Dados em Firebird ou InterBase também podem ser prestados pelo IBPhoenix. Para fazer uma análise e tentar uma possível recuperação de base de dados corrompida, você pode utilizar o IBSurgeon, que pode ser encontrado em IBase.ru (http://www.ibase.ru/)
IBase.ru também prestas serviços de recuperação na Rússia e Europa.

Ofertas / Pedidos de patrocínio do desenvolvimento do Firebird devem ser enviados diretamente para a Fundação FirebirdSQL Inc. Envie um email para foundation@firebirdsql.org. Se preferir o contato com os administradores do projeto Firebird – envie um email para firebirds@users.sourceforge.net.

Discussão Geral sobre melhorias do FB podem ser acessadas no grupo Firebird-priorities:
http://www.yahoogroups.com/community/Firebird-priorities

IB-Architect (http://www.yahoogroups.com/community/ib-architect)deverá ser utilizado APENAS para **discussões técnicas de design**. Questões de Suporte/conversão não deverão ser tratadas aqui.

Ferramentas e Drivers

Programas Desktop Clientes de Bases de Dados

Diversas opções excelentes de ferramentas de Administração Gráficas para o Firebird estão relacionadas na página "Contributed Downloads" em http://www.ibphoenix.com/. Algumas são Open Source, algumas são gratuitas e outras são produtos comerciais reconhecidos.

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

Drivers e Componentes

JAVA: O projeto de driver Jaybird JDBC é desenvolvido ativamente no âmbito do projeto Firebird. Foram lançados inicialmente um driver Tipo 4 JDBC/JCA e um driver beta Tipo 2. O Código fonte e os executáveis podem ser obtidos na página de versões do Firebird :
http://sourceforge.net/projet/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 projeto driver .NET em andamento. Código fonte e binários poderão ser encontrados na página de versões do Firebird em :
http://sourceforge.net/projet/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 usuários tem a seu dispor duas poderosas alternativas de acesso direto a API do Firebird, ambas com bom suporte próprio e da comunidade de programadores. São elas :

- IB Objets de Jason Wharton em http://www.ibobjets.com/
- FIBPLus em http://www.devrace.com/

C++ : a biblioteca de acesso C++ gratuita 'IBPP' (http://www.ibpp.org/), licença MPL, hospedada no sourceforge.net, é completamente portável para Win32 e Linux e provavelmente outras plataformas POSIX. Ela será útil quando se desejr ter um acesso de baixo nível a API, mas com um nível de abstração C++ independente de qualquer ambiente particular de desenvolvimento.

ODBC: A lista de drivers ODBC pode ser encontrada na página "Contributed Downloads" em http://www.ibphoenix.com/. O espaço para discussão do desenvolvimento do driver ODBC/JDBC Open Source é o fórum :
http://lists.sourceforge.net/lists/listinfo/firebird-odbc-devel

PHP: Um grupo está trabalhando para converter a antiga extensão existente do Interbase para o padrão atual do Firebird. Para informações sobre este projeto, inscreva-se no fórum Firebird-PHP em :
http://www.yahoogroups.com/community/firebird-php

PYTHON: KInterbasDB é uma extensão Python que implementa suporte ao Firebird com compatibilidade padrão Python Database API 2.0. Suporte completo e nativo da API do cliente Firebird; com versões estáveis e em desenvolvimento ativo. Licença Open Source BSD. Versões e informações podem ser encontradas 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 encontrada em :

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

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

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

É um trabalho ainda em curso 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 diretamente em :
http://sourceforge.net/projets/firebird

O principal repositório para questões técnicas e de utilização é o site da IBPhoenix :
http://www.ibphoenix.com/
IB Phoenix também disponibiliza assinaturas de CDs atualizados freqüentemente (versões avulsas também estão disponíveis) contendo os manuais publicados por eles – Using Firebird e The Firebird Reference Guide.

Alguma documentação adicional pode ser obtida diretamente na área técnica do site da Borland:
http://www.borland.com/techpubs/interbase/

Correções e Adições desde a versão 1.0

Tracker #	Descrição	Contribuidor
(sem #)	Corrigidas algumas inconsistências nos nomes dos Sets de caracteres.	P. Jacobi
(sem #)	GSTAT caía com algumas combinações de opções.	D. Yemanov
(sem #)	Corrigido o tratamento de SAVEPOINT em BREAK|LEAVE/EXIT.	D. Yemanov
(sem #)	Otimizador alterado para optar por índices simples no lugar de compostos e para utilizar índices únicos com especificação idênticas.	A. Brinkman
(sem #)	Melhoria nas ferramentas de instalação do Win32, 'instsvc.exe' e 'instreg.exe'.	O. Mascia
Melhoria	Número máximo de índices por tabela passou de 64 para (DB_PAGE_SIZE/16)-2,	A. Harrison, transportado para 1.5 por N.Samofatov
721792	Uma conexão aberta for muito tempo provocava "leak" de memória no dispositivo do kernel do Sistema Operacional	N. Samofatov
775003	UDF log(x, y) devolvia log(y, x).	P. Vinkenoog, N. Samofatov
774987	UDFs ltrim(") e rtrim(") devolviam NULL; rtrim omitia o 1º caractere.	P. Vinkenoog, N. Samofatov
(sem #)	Resolvida a queda do servidor causado pela perda do contexto da transação.	A. Peshkoff
(sem #)	Resolvida a queda do servidor para qualquer combinação de sub-select e between.	A. Peshkoff
736318	"<valor> STARTING WITH <campo>" falhava quando usado com índices.	D. Yemanov
(sem #)	Conflitos de Lock não existentes ocorriam após a execução de triggers pré (update/delete).	A. Peshkoff
Melhoria	Definição de INSERTING/UPDATING/DELETING como palavras não reservadas.	N. Samofatov
Melhoria	Adicionadas novas (mais específicas) mensagens de erro para algumas das alterações da versão 1.5.	D. Yemanov, A. Brinkman, A. Peshkoff
Correção de Segurança	Adicionada opção –login a 'instsvc.exe' para permitir a instalação do serviço FB com uma conta diferente de localsystem.	A. Peshkoff

Tracker #	Descrição	Contribuidor
Melhoria	Reintroduzido o trimming de campos VARCHAR nos protocolos remotos.	D. Yemanov
(sem #)	Queda aleatória do servidor quando eram preparadas queries de grande dimensão.	D. Yemanov
Melhoria	Aperfeiçoamento da Configuração – fazre com que o tratamento dos caminhos no firebird.conf esteja em conformidade com os do SO.	A. Peshkoff
(sem #)	Argumentos de UDF do tipo DATE/TIME (dialeto 3) incorretos.	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 ajuste de LockSemCount em plataformas POSIX. Sem necessidade de usar gds_drop ou reiniciar a máquina para que as modificações sejam utilizadas.	N. Samofatov
Melhoria	Definição de FIRST/SKIP como palavras não reservadas.	N. Samofatov
(sem #)	Referência de conexão errada após exceção 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 tabelas temporárias: 1) Falha de segurança em todas as plataformas POSIX exceto FREEBSD/OPENBSD relacionada com o uso de mktemp (possíveis ataques DoS ou aumento de privilégios) Só eram gerados 27 nomes de tabelas únicos em win32 (o que poderia provocar comportamentos inesperados nas versões SS).	N. Samofatov
(sem #)	Alterações no gestor de eventos: desativado o uso de portas definitivas nas versões CS devido a problemas conhecidos.	D. Yemanov
(sem #)	Permitir 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

Tracker #	Descrição	Contribuidor
(sem #)	Possíveis quedas na desconexão quando era usada a notificação de eventos.	D. Yemanov
(sem #)	Modificações no Gerenciador de Serviços: funcionalidades de GSTAT / GSEC não estão disponíveis via Serviços API na win32 CS (até à v1.6).	D. Yemanov
(sem #)	Estatísticas erradas de registro são reportadas quando a operação falha por algum motivo.	D. Yemanov
(sem #)	stdin/stdout não pode ser usado para redirecionar a E/S para o console, 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 quedas (possível em todas as outras versões CS do Interbase/Firebird).	N. Samofatov
Melhoria	Aperfeiçoamento do INTL: permitir que a função UPPER funcione para WIN1251 set de caracteres sem collations explícitos.	N. Samofatov, D. Yemanov
BUGCHECK(291)	Possível corrupção da base de dados quando era modificado/apagado em um pré trigger o próprio registro que o disparou.	A. Peshkoff
(sem #)	Tamanho do Buffer excedido na chamada a isc_database_info().	Oleg Loa
(sem #)	Modificação no Gerenciador de Configuração: O servidor fecha na ausência /erro do firebird.conf registrando erro no log de sistema.	A. Peshkoff
(sem #)	Correção nos Serviços API: ativadas 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 #)	Correção nos Serviços API: serviços API parcialmente disponibilizados para as versões CS win32.	D. Yemanov
(sem #)	Envio incorreto de eventos (uso desnecessário de pacotes OOB).	Jim Starkey, Paul Reeves
(sem #)	Melhoria no tratamento de bloqueios : os conflitos são agora detectados e reportados tão logo sejam recebidas todas as notificações dos processos em bloqueio, i.é., quase instantaneamente na maior parte dos casos.	N. Samofatov
(sem #)	O Servidor caía em algumas operações de Serviços API.	A. Brinkman

Tracker #	Descrição	Contribuidor
(sem #)	Recursos avançados de segurança : 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
(sem #)	Tamanho de Buffer excedido com arrays multidimensionais.	D. Yemanov
213460, 678718	Diversos problemas com eventos 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: ativados os serviços API para plataformas POSIX nas versões CS. Notas: 1. Modificações necessárias no Win32 CS ainda não estão prontas. 2. Serviço de Backup/Restore foi corrigido, testado e deverá funcionar. 3. Validação de Bancos foi parcialmente corrigida e poderá ser usada. 4. É provável que os outros serviços ainda não funcionarão com o CS.	N. Samofatov
(sem #)	Melhorias SQL: permitir Nulos em unique constraints e índices (especificação SQL-99).	D. Yemanov, N. Samofatov
(sem #)	Melhoria de Desempenho: o log de alterações VIO agora usa uma árvore B+ para armazenar os dados de registros dos SAVEPOINT. Melhora do desempenho ao se efetuar múltiplas modificações de registros numa única transação (na ordem de 2-3X para 100.000 registros).	N. Samofatov
(sem #)	Corrupção da Base de Dados após o recuo até um SAVEPOINT depois de grande quantidade de operações DML (que provocasse a liberação do SAVEPOINT da transação) e com registros que tivessem sido modificados antes do SAVEPOINT utilizado e eliminados 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 registro) ou FOR EXECUTE STATEMENT <valor> INTO <lista_var> DO <lista_declarações>;	A. Peshkoff

Tracker #	Descrição	Contribuidor
(sem #)	O servidor ficava preso na desconexão após modificações massivas.	D. Yemanov
(sem #)	Melhorias no otimizador : Subselects na cláusula SET do UPDATE agora podem usar índices.	A. Brinkman
(sem #)	Erro "Context already in use" no caso de DISTINCT com subqueries.	A. Brinkman
(sem #)	Melhorias nas capacidades de isc_database_info: lista das transações atualmente ativas é agora acessível via chamada a isc_database_info.	N. Samofatov
(sem #)	Melhoria de Desempenho: avaliação de expressão booleana parcial. NOTA: o comportamento é controlado pela opção "CompleteBooleanEvaluation" do firebird.conf. Por padrão é 0 (avaliação parcial).	Mike Nordell
(Beta 2 erro)	Estouro da Pilha Interna durante a preparação de comando.	D. Yemanov, Mike Nordell
(sem #)	Melhoria de Desempenho para a arquitetura 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 Otimizador: 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 #)	Longas demoras durante a conexão/desconexão no WinXP.	A. Brinkman
(sem #)	Limpeza genérica: remoção de código não utilizado.	Blas Rodriguez Somoza, Erik Kunze
523589	A View afetava o resultado de um query. Comentário: O problema era que RSE's (dentro de um view) não eram sinalizados como variantes.	A. Brinkman
(sem #)	Modificado o comportamento do modo “forced writes”: agora, se FW=off (desativado), a freqüência com que as “dirty pages” são escritas no disco pode ser configurada (permite maior confiabilidade quando o FW está desativado nas plataformas Win32).	Blas Rodriguez Somoza
(sem #)	A base de dados de segurança foi renomeada para security.fdb.	D. Yemanov
(sem #)	Novo arquivo de configuração : firebird.conf foi finalmente publicado.	D. Yemanov
(sem #)	Novas funções definidas pelo usuário LPAD e RPAD adicionadas na biblioteca IB_UDF.	Juan Guerrero

Tracker #	Descrição	Contribuidor
(sem #)	Algumas vezes GFIX não permitia usar as opções "-user" e "-password" (erro "incompatible swiches").	D. Yemanov
(sem #)	Cache da segurança de conexão : banco de dados de segurança é agora carregado na memória diminuindo o tempo das conexões subseqüentes ao servidor de dados.	D. Yemanov
Melhorias	1. A Memória usada pelo servidor foi reduzida. 2. E/S externa direta quando não existe memória disponível para ordenação. Maior quantidade de streams e predicados é suportado pelo otimizador.	D. Yemanov
508594	LEFT JOIN com VIEWs: simples LEFT JOIN num VIEW com apenas uma cláusula ON não usava um índice mesmo quando isso era possível.	A. Brinkman
(sem #)	Novos sets de caracteres : DOS737, DOS775, DOS858, DOS862, DOS864, DOS866, DOS869, WIN1255, WIN1256, WIN1257, ISO8859_3, ISO8859_4, ISO8859_5, ISO8859_6, ISO8859_7, ISO8859_8, ISO8859_9, ISO8859_13. NOTA: Collations para os sets acima não estão ainda disponíveis.	Blas Rodriguez Somoza
(sem #)	Modificações em CREATE VIEW: proibida a sub cláusula PLAN.	D. Yemanov
(sem #)	Alterado o tratamento de agregados - introduzida a compatibilidade com agregados anteriores. O campo mais interno dentro de uma função agregada determina o contexto ao qual o agregado pertence.	A. Brinkman
(sem #)	Melhorias no otimizador: melhores otimizações de queries JOIN "complexas" (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 retornam a 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 em memória pelo módulo de ordenação.	D. Yemanov
(sem #)	Alterações no build Win32: 1. Alterados os nomes dos objetos USER32 para permitir executar o servidor simultaneamente ao IB/FB1. 2. Nome do Mapa para o protocolo local (IPC) foi alterado. Assim, a biblioteca cliente da v1.5 não é mais compatível com as versões anteriores via IPC. 3. Todos os nomes de protocolos de transporte (porta e serviço INET,	D. Yemanov

	pipe WNET, mapa IPC) são agora configuráveis via firebird.conf.	
Tracker #	**Descrição**	**Contribuidor**
(sem #)	Novo gerenciador de memória interno foi modificado para permitir um melhor desempenho.	N. Samofatov
(sem #)	RDB$FIELD_LENGTH para views que continham concatenações de campos long CHAR/VARCHAR, assumia valores incorretos.	D. Yemanov

Melhoria	Melhoria nos Triggers: adicionadas variáveis de ação em tempo de execução (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 estabelecida por um 'IN'.	A. Brinkman
Melhoria	Permitido uso de expressões arbitrárias na cláusula ORDER BY.	N. Samofatov
(sem #)	O servidor caía 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 registro encontrado para leitura.	A. Brinkman
(sem #)	QLI não compreendia o tipo de dados BIGINT.	D. Yemanov
(sem #)	Tamanho de variáveis de texto dentro de procedures/triggers não era copiado para a estrutura do descritor.	A. Brinkman

Tracker #	Descrição	Contribuidor
(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
(sem #)	Buffer excedido (MAXPATHLEN) e nome local da função diretório 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 ativar o comportamento anterior de mapeamento (por razões de compatibilidade) usando o parâmetro do gerenciador de configuração "OldParameterOrdering".	N. Samofatov
Melhoria	Melhorias no Otimizador: 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 bloqueio explícito de registro finalizada. Deverá estar estável e consistente.	N. Samofatov
Melhoria	Melhoria no Otimizador: melhor tratamento dos nós AND dentro dos nós OR.	A. Brinkman
(sem #)	Exceções dentro de um ciclo for/while em triggers não eram tratados corretamente.	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 Otimizador: se alguns índices com grande diferença de seletividade puderem ser usados para obtenção de dados, apenas os melhores serão usados enquanto os outros serão ignorados.	D. Yemanov
(sem #)	Problema com identificadores com aspas em expressões de “plan”.	N. Samofatov

Tracker #	Descrição	Contribuidor
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 não são mais 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
Melhoria	Novo gerenciador de configuração: ativada a mesma configuração baseada em arquivo texto para todas as plataformas suportadas.	D. Yemanov
Melhoria	Melhoria no Otimizador: foi acrescentado melhor suporte para usar índices com "OR". E obter o melhor índice composto paras nós "AND".	A. Brinkman
Melhoria	Adicionado o suporte para tratamento de SAVEPOINT explícito em DSQL.	N. Samofatov
(sem #)	Limpeza de Protocolo: O protocolo de rede IPX/SPX não é mais 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 Otimizador: adicionado o suporte para a detecção do uso de índice com sub-selects em select agregado.	A. Brinkman
Melhoria	Melhoria no gerenciador de processos para Win32 SS: agora o servidor deverá ser mais rápido na resposta sob “carga intensa”.	A. Peshkoff
Melhoria	Adicionado suporte para bloqueio explícito de registro. O comportamento do Wait nos modos de transação 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 Transação 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

Tracker #	Descrição	Contribuidor
496784	Quando o otimizador 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 registro "GuardianOptions".	D. Yemanov
(sem #)	Perda de Recursos no tratamento recursivo de procedimentos DDL, o que fazia com que alguns DDL falhassem.	N. Samofatov
(sem #)	Check constraint que use apenas um campo de tabela agora é eliminada automaticamente quando este campo é eliminado.	N. Samofatov
451927	Novas variáveis de sistema ROWS_AFFECTED em PSQL: devolve o número de linhas afetadas pela último comando INSERT/UPDATE/DELETE. Para quaisquer outras declarações além de INSERT/UPDATE/DELETE, o resultado será sempre zero.	D. Yemanov
446240	Mensagens de exceção dinâmicas: é permitido definir a mensagem que será exibida na ocorrência de uma exceção diferente daquela com que a exceção foi criada. Sintaxe: EXCEPTION nome [valor];	D. Yemanov
547383	Novas variáveis de sistema SQLCODE e GDSCODE fornecem acesso ao erro ocorrido dentro de um bloco-WHEN em PSQL. Fora do bloco-WHEN, devolve 0 (sucesso).	D. Yemanov
(no #)	Semântica de reinicialização de Exceção : permite que uma exceção já tratada em PSQL seja relançada num bloco WHEN. Sintaxe: EXCEPTION; Nenhum efeito fora do bloco-WHEN.	"Digitman"
(no #)	O servidor caía durante o garbage collection sob carga intensa.	N. Samofatov
Melhoria	Compilação de meta dados deferida: resolvido grande número de causas do conhecido erro; "objeto em uso".	N. Samofatov
Melhoria	Novo tratamento da ordenação de Nulos: permitir ordenação de Nulos definida pelo usuário.	N. Samofatov
(no #)	Gstat mostrava valor errado para elemento maxdup.	D. Kuzmenko

Tracker #	Descrição	Contribuidor
(no #)	É usada uma nova chave de registro no win32: SOFTWARE\FirebirdSQL\Firebird.	--
451925	Nomes de índices de constraints definidos pelo usuário: é permitido usar nomes definidos pelo usuário ou o nome da restrição, para índices que imponham restrições.	D. Yemanov
Melhoria	Nova declaração RECREATE VIEW: substitui o par de declarações DROP VIEW / CREATE VIEW. Sintaxe: RECREATE VIEW nome <definição_view>;	D. Yemanov
(no #)	Trigger cujo nome começa por 'RDB$' não pode ser alterado ou eliminado.	D. Yemanov
(no #)	Arquivos da distribuição foram renomeados de acordo com o nome Firebird. Agora são fbserver, fbclient, firebird.msg, etc. A nova biblioteca cliente é a fbclient, e deverá ser usada pelos novos projetos baseados no FB. GDS32 apenas aponta para a fbclient e é fornecida somente para manter a compatibilidade.	Vários
(Pequeno incremento no ODS)	Adicionados novos índices de sistema (RDB$INDEX_41, RDB$INDEX_42, RDB$INDEX_43). Agora a versão do ODS é a 10.1.	D. Yemanov, N. Samofatov
451935	Novo comando CREATE OR ALTER para triggers e stored procedures, permitindo criar ou alterar um objeto da base de dados dependendo da sua existência prévia. Sintaxe: CREATE OR ALTER nome <definição_objeto>;	D. Yemanov
(no #)	Apareciam dependências inválidas na base de dados (tal como DB$34) após alterações de meta dados.	D. Yemanov
(no #)	Declaração de variáveis locais aperfeiçoada : simplificada a sintaxe e possibilidade de declaração e definição das variáveis em um mesmo comando. Sintaxe: DECLARE [VARIABLE] nome <tipo_de_variável> [{'=' | DEFAULT} valor]; Exemplo: DECLARE my_var INTEGER = 123;	Claudio Valderrama
(no #)	Desativada a declaração BREAK para triggers (como EXIT) devido a limitações internas conhecidas.	D. Yemanov

Tracker #	Descrição	Contribuidor
555839, 546274	Agrupamento aperfeiçoado : permite usar GROUP BY para funções internas e subqueries, bem como para valores posicionais (isto é, a posição relativa da coluna, ou seja, o grau da coluna no conjunto de dados retornados).	A. Brinkman
451917	Nova função interna COALESCE permite que uma coluna possa ser calculada por uma série de expressões, em que a primeira expressão que resulte em um valor não nulo será o valor da coluna.	A. Brinkman
451917	Nova função interna NULLIF para sub-expressões. Retorna NULL se o resultado for igual a um valor específico, ou o próprio 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 condicional.	A. Brinkman
545725	Sweep automático em segundo plano bloqueava.	A. Peshkoff
(no #)	O servidor caía quando estruturas XSQLDA não eram preparadas para todos os parâmetros de um comando.	D. Yemanov
(no #)	PSQL: ativado suporte para blocos BEGIN...END vazios.	D. Yemanov
567931	Resolvida parcialmente uma falha de segurança de meta dados.	D. Yemanov
(no #)	BigInt arrays não funcionavam.	Artem Petkevych
437859	Implementada execute procedure e concatenação de string, visando permitir que qualquer expressão possa ser utilizada 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 inteiros de 64-bit (Apenas no Dialeto 3).	D. Yemanov
451922	Permitir que um Trigger universal seja disparado para diferentes tipos de ações.	D. Yemanov
446238, 446243	Novo CONNECTION_ID e TRANSACTION_ID de sistema acessível em PSQL. Retorna identificador interno apropriado, armazenado na página de cabeçalho da base de dados.	D. Yemanov

Tracker #	Descrição	Contribuidor
446180	Alias de Base de Dados no servidor: conexão a qualquer base de dados usando um "alias" em vez da localização física da mesma. A lista de apelidos de banco de dados conhecidos é armazenada no arquivo aliases.conf localizado na raiz da instalação do servidor. Exemplo: Entrada do alias no arquivo de configuração: minha_base_dados = c:\dbs\my\database.gdb Texto para conexão ao BD na aplicação: localhost:minha_base_dados	D. Yemanov
(no #)	Novo gerenciador 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 arquivo temporário.	D. Yemanov
538201	Queda da aplicação na tentativa de extração de nulo como DATE.	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 gerenciador de memória.	John Bellardo
(no #)	Nova lógica de tratamento de exceções.	Mike Nordell, John Bellardo
(no #)	Nova configuração de compilação baseada no autoconf.	John Bellardo, M. O'Donohue, Erik Kunze
(no #)	O transporte do código de C para C++.	Mike Nordell, John Bellardo, M. O'Donohue

Equipe de Tradução: Hélcio Sugiyama