Firebird™ Versione 1.5

Note sulla release 1.5

5 Febbraio 2004

Contenuti

Generalità

Il motore database Firebird™ è stato sviluppato da una squadra indipendente di sviluppatori volontari a partire dal codice sorgente di InterBase™ , che è stato reso pubblico dalla Borland sotto la InterBase Public License v.1.0 il 25 Luglio 2000.

Lo sviluppo della base del codice di Firebird 2 iniziò nelle prime fasi dello sviluppo di Firebird 1, con il porting del codice C di Firebird 1 al C++ e con la prima rilevante messa a punto del codice. Firebird 1.5 è la prima versione frutto del codice Firebird 2. È un importante risultato per gli sviluppatori e per l'intero progetto Firebird, ma non un punto di arrivo in sé stesso. Mentre Firebird 1.5 viene rilasciato, significative revisioni sono in corso per la release della prossima versione nel cammino verso Firebird 2.

Firebird 1.0.x continua nella manutenzione attiva, con la soluzione dei principali problemi e l'aggiornamento retroattivo delle migliorie introdotte dalla versione 1.5.

I file binari di Firebird 1.5

I file binari di Firebird possono essere scaricati dal sito Web di Firebird - http://sourceforge.net/project/showfiles.php?group_id=9028

Stringhe di versione per i rilasci di Firebird 1.5

Win32: "WI-V1.5.0.nnnn Firebird 1.5"
Linux: "LI-V1.5.0.nnnn Firebird 1.5"
e così via, dove nnnn è il numero di build

Riferitevi alla sezione Documentazione per trovare la documentazione raccomandata.

Nuove caratteristiche

Nuova base di codice, migliore ottimizzazione

Questo rilascio è frutto del codice portato dal C originale al C++, un processo iniziato da Mike Nordell nel 2000. É stata realizzata una massiccia pulizia del codice e correzione di errori, insieme ad una nuova gestione della memoria e a miglioramenti del linguaggio. Non ultimo, durante lo sviluppo della versione 1.5 l'ottimizzatore di query SQL ha subito migliorie e correzioni per opera di Arno Brinkman e altri, con il risultato di incrementi di velocità osservati nell'ordine del 30 - 60 per cento e più.

Architettura

Due significative aggiunte alla piattaforma Windows sono il server Classic e la versione Embedded.

- Non c'è stato un server Classic per Windows per quasi otto anni. Questa versione può utilizzare processori multipli, il che ancora non è possibile con la versione Superserver. Sebbene utilizzabile in produzione, la versione Classic dovrebbe essere considerata sperimentale.

- Il server Embedded è una dll che unisce una singola connessione client con un Firebird Superserver per creare soluzioni portatili e stand-alone molto velocemente ed in modo efficiente.

Molte nuove caratteristiche del linguaggio sono state aggiunte dalla versione 1.0.x, incluso il costrutto e le funzioni SQL-92 CASE, COALESCE e NULLIF. Per la sintassi di queste ed altre nuove implementazioni del linguaggio, si prega di fare riferimento alla sezione Miglioramenti del linguaggio, più avanti in questo documento.

Moduli installati e sicurezza

Se avete usato Firebird 1.0.x finora noterete grossi cambiamenti nei nomi dei moduli e nelle regole per accedervi e per localizzarli. Di seguito alcune osservazioni; per informazioni dettagliate sull'installazione, il layout su disco e la configurazione, consultate le sezioni rilevanti di questo documento.

1. La maggior parte dei moduli e delle costanti sono state rinominate. In molti casi i nuovi nomi implicano varianti di "firebird" o "fb". Per esempio, la libreria API è ora una libreria condivisa chiamata "fbclient.dll" su Windows e "libfbclient.so" su altre piattaforme. L'eccezione alla regola è il database di sicurezza, prima chiamato "isc4.gdb" ed ora chiamato "security.fdb".
2. I file esterni usati dal server (librerie UDF, filtri BLOB, librerie di set di caratteri, tabelle esterne) sono ora soggette a livelli di protezione del filesystem che, in alcuni casi, hanno valori di default diversi da quelli che si avevano sotto Firebird 1.0.x o InterBase.
3. Il nuovo file di configurazione, firebird.conf, che sostituisce ibconfig (Windows)e isc_config (altre piattaforme) contiene diverse nuove caratteristiche configurabili, insieme ad una migliore auto-documentazione e organizzazione.

4. La versione 1.5 offre il database-aliasing. Ora è possibile opzionalmente codificare la posizione del database nel codice dell'applicazione usando la propria scelta di un alias per sostituire la stringa con il percorso. I reali percorsi sono in un file di testo, aliases.conf. Lo scopo principale dell'aliasing è tuttavia proteggere i percorsi fisici dall'essere "catturati" da malintenzionati mentre attraversano la rete.
5. La pratica standard (ora obsoleta) sulle piattaforme server Windows, prevedeva che l'utente locale del sistema lanciasse il programma che installa il servizio Firebird all'avvio del sistema. Questa può essere una seria fonte di vulnerabilità se il servizio Firebird è manomesso, perché fornisce una finestra tramite cui un hacker può accedere all'intera macchina. La versione 1.5 di questo programma (instsvc.exe) ora accetta un log-on di un utente Windows per l'installazione del servizio. É fortemente raccomandato che venga creato un utente Firebird per questo scopo e che si faccia uso della nuova caratteristica di logon se il vostro server è connesso in qualsiasi modo a Internet.

Eliminazione di spazi in eccesso da campi Varchar per protocolli remoti

Il lavoro su questa complicata caratteristica è stato ripreso e finito per il client 1.5 ed i campi varchar ora attraversano la rete con gli spazi in eccesso a destra rimossi, fino alla lunghezza reale più due byte. NOTA Siccome è il client che richiede al server di rimuovere gli spazi in eccesso nei varchar, il client Firebird 1.5 (fbclient.dll o libfbclient.so) lo farà anche se connesso ad un server pre-1.5. Se usate un vecchio client, non avrete viceversa l'eliminazione degli spazi in eccesso nemmeno se connessi ad un server 1.5 o successivo.

Semantica multi-azione dei trigger

Si possono ora scrivere azioni condizionali di insert/update/delete in un trigger Before o After per avere un singolo trigger in grado di fare tutte le azioni DML per quella fase del trigger. Ciò semplifica molto la composizione e manutenzione dei trigger senza diminuire la possibilità di avere multipli triggers per fase.

Miglioramenti ai vincoli con nome

Gli indici che impongono vincoli di integrità con nome possono ora essere denominati con identificatori definiti dall'utente.
Attenzione, se usate questa caratteristica il database non potrà più essere ritrasportato alla v.1.0.x o a Interbase®

Aumento del massimo numero di indici per tabella

Sia nella Release 1.0 che in questa — il massimo numero di indici che si possono definire per una tabella è stato aumentato da 64 a 256.

Blocco dei record pessimistico

Per i rari casi in cui occorre imporre una politica di pessimistic lock, questa release aggiunge una sintassi per eseguire un lock in lettura sulle righe man mano che vengono inviate al client. Usare con cautela.

Caching della connessione al Security database

La connessione al security database è in cache nelle build SuperServer. Ciò implica che security.fdb è caricato allo stabilirsi della prima connessione e resta collegato finché l'ultima connessione di un client è attiva.

Miglioramenti nella messaggistica d'errore

Ove possibile, i messaggi di errore riportano la causa degli errori SQL ad un livello più dettagliato. É IMPORTANTE notare che si incontreranno messaggi di errore bizzarri se si usa un vecchio file interbase.msg o firebird.msg.

Services API con Classic Server per Linux

Un limitato supporto per le Services API è ora disponibile su Classic server su Linux. I servizi disponibili da gbak (backup/restore) e gfix (validazione database, shutdown/online, ecc.) funzionano. Altri (gstat, log del server, ecc.) non sono stati testati e probabilmente non sono funzionali.

Cambiamenti nelle librerie client

Client Windows

La libreria client è ora chiamata "fbclient.dll". Tutte le utility del server (gbak, gfix, ecc.) usano solo la libreria client fbclient.dll; potete connettere nuove applicazioni a fbclient.dll, senza bisogno di gds32.dll (raccomandato).
Per compatibilità con le applicazioni esistenti, è possibile generare un "clone" di fbclient.dll con nome "gds32.dll" usando una nuova utility chiamata instclient.exe. Per i dettagli vedere la sezione di installazione e una versione recente delle note di installazione distribuite con il vostro kit Windows.

Client Linux

La libreria client Superserver è ora chiamata "libfbclient.so". Per compatibilità con le applicazioni esistenti viene installato un symlink (link simbolico) "libgds.so", che punta a libfbclient.so. La libreria client locale per applicazioni embedded che si connettono ad un server Classic sono state rinominate libfbembed.so.

File rinominati e moduli

Piattaforma	Moduli	Firebird 1.0	Firebird 1.5	Note speciali
Tutte	Variabili di ambiente	INTERBASE INTERBASE_LOCK INTERBASE_MSG INTERBASE_TMP	FIREBIRD FIREBIRD_LOCK FIREBIRD_MSG FIREBIRD_TMP	Punta al percorso della radice di installazione Punta alla posizione del file dei lock Punta alla posizione del file dei messaggi Punta alla directory per lo spazio di ordinamento
Tutte	Database di sicurezza	Isc4.gdb	security.fdb	
Tutte	File messaggi	Interbase.msg	firebird.msg	
Tutte	Log del Server	interbase.log	firebird.log	
Tutte	Versione ODS	10	10.1	La nuova ODS (10.1) non causa incompatibilità con precedenti ODS ma la versione non viene aggiornata. Firebird 1.0 e 1.5 possono servire sia DB ODS 10.0 che 10.1. Tuttavia il backup/restore è ancora la procedura raccomandata per la migrazione di DB a differenti versioni del server.
Linux	Eseguibile server Classic	gds_inet_server	fb_inet_server	
Linux	Manager dei lock Classic	ib_lock_mgr	fb_lock_mgr	
Linux	Controllo Superserver	ibmgr.bin	fbmgr.bin	
Linux	Eseguibile Superserver	ibserver	fbserver	
Linux	File di configu-razione	isc_config	firebird.conf	
Linux	Libreria client	libgds.so	libfbclient.so libfbembed.so	Client remoto thread-safe e client loopback TCP/IP per il Superserver Client locale (singolo utente, non thread-safe) per Classic
Linux	Libreria client (symlink per compa-tibilità)	N/A	libgds.so	

Piattaforma	Modulo	Firebird 1.0	Firebird 1.5	Note speciali
Windows	Guardian	ibguard.exe	fbguard.exe	
Windows	Eseguibile Superserver	ibserver.exe	fbserver.exe	Non multi-processore.
Windows	Eseguibile Classic	N/A	fb_inet_server.exe	Connessione locale Windows non disponibile. TCP/IP, NetBEUI sono OK. Versione multi-processore.
Windows	Libreria Client	gds32.dll	fbclient.dll	Le versioni Fb 1.5 delle utility del server e tutte le nuove applicazioni richiedono solo fbclient.dll. Leggere le note sotto relative alla compatibilità con gds32.dll per vecchie applicazioni.
Windows	File di configu-razione	Ibconfig	firebird.conf	
Windows	Porta IPC locale	InterBaseIPI	FirebirdIPI	Con le impostazioni di default del server non si possono fare connessioni locali da applicazioni usando una vecchia libreria client (gds32.dll). Se occorre, si può impostare il server per usare il vecchio nome della mappa IPC tramite firebird.conf.
Windows	Chiave del Registry predefinita	HKLM\SOFTWARE\Borland\InterBase	HKLM\SOFTWARE\Firebird Project\Firebird Server\Instances	Il percorso è memorizzato nel parametro "DefaultInstance". Non c'è più la chiave "CurrentVersion" e "RootDirectory" è sostituita da "DefaultInstance".
I nuovi nomi di servizio sotto Windows sono "Firebird Guardian – DefaultInstance" e "Firebird Server – DefaultInstance".				

Compatibilità

On-disk structure (ODS)

La On-Disk Structure di Firebird 1.5 è denominata ODS 10.1. Questa evoluzione minore dell'ODS è stata necessaria per:
- tre nuovi indici per tabelle di sistema
- modifiche minori nel BLR di due trigger di sistema
- codifica migliorata di RDB$TRIGGER_TYPE.

Certe migliorie che richiedono modifiche all'ODS sono state procrastinate fino al rilascio della versione 2. Nel frattempo dovreste essere in grado di trasportare I vostri database Firebird 1.0.x direttamente. Fate backup (e controllateli) dei vostri database Firebird 1.0.x prima di portarli a 1.5.

Database InterBase™

Se state considerando di utilizzare Firebird con un database InterBase esistente, con l'intenzione di ritornare in seguito a InterBase, vi preghiamo di prendere ogni precauzione nell'eseguire il backup del vostro archivio usando l'appropriata versione InterBase di gbak. Per cominciare a lavorare con il vostro database in Firebird 1.5, usate la versione Firebird 1.5 di gbak per ripristinare il vostro backup.
La Guida Operativa della Documentazione InterBase® 6.0 beta contiene la sintassi del comando per il programma di backup e restore gbak.

I database IB 7.x e probabilmente IB 6.5 possono funzionare in modo non corretto dopo la migrazione a FB 1.5 via backup/restore, se in essi sono state usate nuove caratteristiche specifiche di IB.

Nomi di file e posizioni

In questa release un numero sostanziale di file hanno nuovi nomi, come parte di un graduale processo di sostituzione dei nomi ereditati da InterBase® 6. Leggete la sezione Nomi di file e posizioni per descrizioni e raccomandazioni.

Esecuzione di server concorrenti

Modifiche ad alcuni nomi di oggetti di sistema consentono a FB 1.5 di essere installato e usato su un computer che abbia gia InterBase o Firebird 1.0.x installati. Sotto Windows FB 1.5 usa anche una diversa chiave del Registry. Se impostate I server per usare porte diverse di rete è possibile eseguire alcune istanze dei server in modo concorrente, o eseguire FB 1.5 mentre IB o FB 1.0.x sono in esecuzione.

Riconversione a Firebird 1.0.x

A seguito di un gran numeri di correzioni di errori il comportamento dei database può cambiare se eseguite un downgrade da un DB v.1.5 ad uno v.1.0.x. In particolare, se create chiavi primarie, uniche o foreign come vincoli con nome, i nomi predefiniti degli indici saranno incompatibili con la v.1.0.x. Attendetevi l'uscita di un futuro README contenente dettagli su questi argomenti.

Compatibilità Linux

Per una serie di problemi con il compilatore GNU C++, le versioni Firebird 1.5 Linux richiedono versioni del runtime glibc più recenti che in passato. In questo momento, a causa di questi problemi, l'esito dell'installazione e dell'esecuzione dei binari di Firebird 1.5 è strettamente dipendente dalla distribuzione. La tabella seguente può essere d'aiuto, tuttavia gradiremmo ulteriori informazioni; vi preghiamo di condividere le vostre esperienze con queste ed altre distribuzioni nel forum firebird-devel.

Distribuzione	Livello	Classic	Superserver
Red Hat	7.x	No	No
	8.0	Sì	Sì
	9.0	Sì	Sì
Mandrake	8.x	No	No
	9.0, 9.1, 9.2	Sì	Sì
SuSE	7.3	Sì – installare i packages libgcc-3.2-44.i586.rpm & libstdc++-3.2-44.i586.rpm prima di installare Firebird 1.5.	Ignoto
	8.0, 8.1	Sì	8.0 Sì (8.1 Ignoto)

Miglioramenti del linguaggio

TIPI DI DATI

(1.5) Nuovi tipi di dati SQL nativi

BIGINT
Tipo numerico esatto secondo SQL99, 64-bit con segno, senza decimali. Disponibile solo nel Dialetto 3.

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

METADATI

(1.5) Miglioramenti ai vincoli con nome

Dmitry Yemanov

Gli indici che realizzano fisicamente I vincoli con nome ora possono avere identificatori scelti dall'utente.

Sebbene precedentemente fosse possibile creare vincoli con nome di tipo PRIMARY, FOREIGN KEY e UNIQUE, l'identificatore del corrispondente indice automaticamente generato era calcolato dal sistema, ad es. RDB$FOREIGN13, ed era inalterabile. Questo resta il comportamento normale se non vengono usati i vincoli con nome.

Sono state aggiunte estensioni del linguaggio che consentono:

a) agli indici generati dal sistema di ricevere automaticamente lo stesso identificatore del vincolo con nome a cui corrispondono
b) ad un indice corrispondente ad un vincolo con o senza nome di ricevere un identificatore scelto dall'utente esplicitamente e di essere opzionalmente creato in ordine DISCENDENTE. NOTA Non è correntemente possibile usare un indice pre-esistente.

Sintassi

```
...
[ADD] CONSTRAINT [<constraint-identifier>]
<constraint-type> <constraint-definition>
[USING [ASC[ENDING] | DESC[ENDING]] INDEX <index_name>]
```

Caveat: Assicuratevi che gli indici sulla chiave foreign e quella primaria usino lo **stesso ordine di sort** (DESC | ASC).

Esempi

i) Vincolo con nome e indice con nome esplicito

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

Il seguente comando crea un vincolo su chiave primaria con nome PK_ATEST ed il relativo indice discendente chiamato IDX_PK_ATEST:

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

ii) Alternativo a i) sopra:

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

iii) Questo comando crea la tabella ATEST con chiave primaria PK_ATEST. L'indice corrispondente è chiamato PK_ATEST anch'esso.

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

(1.5) Trigger multi-azione

Dmitry Yemanov

I trigger sono migliorati per consentire la gestione condizionata di operazioni multiple a livello di riga.

Sintassi

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

<multiple_action> ::= <single_action> [OR <single_action> [OR <single_action>]]
<single_action> ::= {INSERT | UPDATE | DELETE}

Esempi

i)

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

ii)

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

Modifiche all'ODS

La codifica del campo RDB$TRIGGER_TYPE (relazione RDB$TRIGGERS) è stata estesa per consentire azioni trigger complesse. Per I dettagli, riferitevi al documento readme.universal_triggers.txt nel ramo /doc/sql.extensions dell'albero CVS di Firebird.

Note:
1. I trigger a singola azione sono del tutto compatibili a livello ODS con FB 1.0.
2. La codifica RDB$TRIGGER_TYPE è dipendente dall'ordine, cioè BEFORE INSERT OR UPDATE e BEFORE UPDATE OR INSERT sono codificati in modo diverso, sebbene abbiano la stessa semantica e siano eseguiti esattamente allo stesso modo.
3. Sia la variabile di contesto OLD che la NEW sono disponibili in trigger ad azione multipla. Se l'invocazione del trigger ne impedisce una (a es. il contesto OLD per un'operazione di INSERT), allora tutti i campi di quel contesto avranno il valore NULL. Se assegnati ad un contesto improprio, si avrà un'eccezione runtime.
4. Le nuove variabili di contesto booleane INSERTING/UPDATING/DELETING possono essere usate per verificare il tipo di operazione runtime. (Vedi sotto.)

(1.5) RECREATE VIEW

É esattamente uguale a CREATE VIEW se la view non esiste già. Se invece esiste, RECREATE VIEW cerca prima di fare un drop e crea quindi un oggetto completamente nuovo. RECREATE VIEW fallisce se l'oggetto è in uso.
Usa la stessa sintassi di CREATE VIEW.

(1.5) CREATE OR ALTER {TRIGGER | PROCEDURE }

Comando che crea un nuovo trigger o procedura (se non esiste già) o la altera (se esiste già) e la ricompila. La sintassi CREATE OR ALTER preserva le dipendenze e permessi esistenti.

La sintassi è la stessa usata per CREATE TRIGGER | CREATE PROCEDURE, rispettivamente, eccetto per le parole chiave aggiuntive "OR ALTER".

(1.5) NULL in vincoli e indici unici

Dmitry Yemanov

É ora possibile applicare un vincolo UNIQUE o un indice unico a una colonna che non ha il vincolo NOT NULL. Ciò è in accordo con SQL-99. Siate cauti nell'uso di questa caratteristica se pensate di riconvertire il vostro DB in Firebird 1.0.x o qualsiasi versione di InterBase.

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

dove <unique column list> può contenere una o più colonne senza l'attributo NOT NULL, se <unique specification> è UNIQUE o UNIQUE INDEX nome-indice. Notate che tutte le colonne in PRIMARY KEY devono ancora essere dichiarate NOT NULL.

Il vincolo consente l'esistenza delle sole righe per cui la condizione di ricerca (i) o (ii) ha valore True, secondo la seguente logica:

i) Se <unique specification> specifica PRIMARY KEY, allora la condizione di ricerca sarà:

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

ii) Altrimenti, la condizione di ricerca <search condition> sarà:

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

In questo caso, la condizione UNIQUE non può essere vera se (SELECT UCL FROM TN) ha potuto fornire due righe dove tutte le coppie di segmenti non non-null sono accoppiate.

Il vincolo consente l'esistenza solo di quelle righe per cui la condizione precedentemente menzionata <search condition> ha valore True. In un indice unico o sotto il vincolo UNIQUE, due insiemi di valori colonna sono considerati distinti e quindi consentiti se:

a) Entrambi gli insiemi contengono solo null, o
b) C'è almeno una coppia di valori corrispondenti di cui uno è non-null e l'altro è null o un diverso valore non-null.

Esempi

Vincolo UNIQUE:

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

O indice UNIQUE:

```
CREATE TABLE t (a INTEGER, b INTEGER);
COMMIT;
CREATE UNIQUE INDEX uqx ON t(a, b);
COMMIT;
INSERT INTO t VALUES (NULL, NULL); /* ok, nulls consentiti */
INSERT INTO t VALUES (1, 2); /* come i non-nulls */
INSERT INTO t VALUES (1, NULL); /* e le combinazioni */
INSERT INTO t VALUES (NULL, NULL); /* ok, tutte le coppie di nulls sono distinte */
```

ma

```
INSERT INTO t VALUES (1, NULL); /* fallisce perché tutti i segmenti non-null corrispondenti sono
accoppiati */
```

Significa che ***il vincolo PRIMARY KEY non consente NULL*** mentre il vincolo UNIQUE e gli indici unici consentono un numero arbitrario di NULL. Per insiemi risultato multi-colonna di (SELECT UCL FROM TN), sono applicate le regole comuni per i NULL, cioè (1, NULL) è distinto da (NULL, 1) ed un (NULL, NULL) è distinto da ogni altro (NULL, NULL).

DSQL

(1.5) Espressioni e variabili come argomenti di procedure

Dmitry Yemanov

Le chiamate a EXECUTE PROCEDURE ProcName(<Argument-list>) e SELECT <Output-list> FROM ProcName(<Argument-list>) accettano ora variabili locali (in PSQL) ed espressioni (in DSQL e PSQL) come argomenti.

(1.5) Nuovi costrutti per le espressioni CASE

Arno Brinkman

a) CASE

Consente che il risultato di una colonna sia determinato dal risultato di un gruppo di condizioni esclusive.

Sintassi

```
<case expression> ::=
    <case abbreviation>  | <case specification>

<case abbreviation> ::=
    NULLIF <left paren> <value expression> <comma> <value expression> <right paren>
  | COALESCE <left paren> <value expression> { <comma> <value expression> }... <right paren>

<case specification> ::=
    <simple case>  | <searched case>

<simple case> ::=
  CASE <value expression>  <simple when clause>...
    [ <else clause> ]
  END

<searched case> ::=
  CASE <searched when clause>...
    [ <else clause> ]
  END

<simple when clause> ::= WHEN <when operand> THEN <result>
<searched when clause> ::= WHEN <search condition> THEN <result>
<when operand> ::= <value expression>
<else clause> ::= ELSE <result>
<result> ::= <result expression>  | NULL
<result expression> ::= <value expression>
```

Esempi

i) semplice

```
SELECT
    o.ID,
    o.Description,
    CASE o.Status
```

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

ii) con ricerca

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

b) COALESCE

Consente di calcolare il valore di una colonna tramite un numero di espressioni, delle quali la prima a rendere un valore non-NULL è resa come risultato.

Formato

<case abbreviation> ::=
| COALESCE <left paren> <value expression> { <comma> <value expression> }... <right paren>

Regole di sintassi

i) COALESCE (V1, V2) equivale alla seguente specifica
<case specification>:
CASE WHEN V1 IS NOT NULL THEN V1 ELSE V2 END

ii) COALESCE (V1, V2,..., Vn), for n >= 3, equivale alla seguente specifica:
<case specification>:
CASE WHEN V1 IS NOT NULL THEN V1 ELSE COALESCE (V2,...,Vn) END

Esempi

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

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

c) NULLIF

Rende NULL per una sub-espressione se questa ha un valore specifico, altrimenti rende il valore della sub-espressione.

Formato

<case abbreviation> ::=
    NULLIF <left paren> <value expression> <comma> <value expression> <right paren>

Regole sintassi

NULLIF (V1, V2) equivale alla seguente specifica
<case specification>:
  CASE WHEN V1 = V2 THEN NULL ELSE V1 END

Esempio

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

(1.5) Punti di salvataggio secondo SQL99

Nickolay Samofatov

I punti di salvataggio utente (detti anche *transazioni nidificate*) forniscono un modo conveniente per gestire errori di business logic senza il bisogno di un roll back della transazione. Disponibili solo in DSQL.

Usate il comando SAVEPOINT per identificare un punto in una transazione a cui potere in seguito fare un roll back.

```
SAVEPOINT <identifier>;
```

<identifier> specifica il nome di un punto di salvataggio da creare. Dopo che questo è stato creato potete continuare l'elaborazione, eseguire un commit del vostro lavoro, fare un roll back dell'intera transazione, o un roll back fino al punto di salvataggio.

I punti di salvataggio devono essere distinti entro una data transazione. Se create un secondo punto di salvataggio con lo stesso nome di uno precedente, quest'ultimo viene cancellato.

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

Questo comando esegue le seguenti operazioni:

- Esegue un roll back delle modifiche apportate nella transazione dopo il punto di salvataggio
- Cancella tutti I punti di salvataggio creati dopo questo. Il punto di salvataggio con nome è mantenuto, in modo che sia possibile un roll back allo stesso punto di salvataggio più volte. Anche i punti di salvataggio precedenti sono mantenuti.
- Rilascia tutti I blocchi di record impliciti ed espliciti acquisiti dal punto di salvataggio in poi. Altre transazioni che abbiano richiesto accesso a righe bloccate dopo il punto di salvataggio devono continuare ad aspettare finché la transazione è salvata o viene effettuato un roll back. Altre transazioni che non hanno ancora richiesto le righe possono richiederle ed accedere alle righe immediatamente.
  Nota: questo comportamento potrebbe cambiare in future versioni del prodotto.

Il registro per l'annullamento dei punti di salvataggio può consumare significative quantità di memoria del server, specialmente se si aggiorna gli stessi record nella stessa transazione più volte. Usate il

comando RELEASE SAVEPOINT per liberare le risorse di sistema consumate dalla gestione dei punti di salvataggio.

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

Il comando RELEASE SAVEPOINT cancella il punto di salvataggio <identifer> dal contesto della transazione. A meno che non si specifichi la parola chiave ONLY, tutti i punti di salvataggio definiti dopo il punto di salvataggio <identifier> sono cancellati anch'essi.

Esempio di uso di punti di salvataggio

```
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; -- rende nessuna riga
rollback to y;
select * from test; -- rende due righe
rollback;
select * from test; -- rende una riga
```

Punti di salvataggio interni
Per default il motore usa automaticamente un punto di salvataggio di sistema a livello di transazione per effettuare il rollback di transazioni. Quando date un comando ROLLBACK, tutte le modifiche effettuate in questa transazione sono salvate in un punto di salvataggio a livello di transazione e la transazione subisce un commit. Questa logica riduce la necessità di garbage collection causata da transazioni di cui viene chiesto il roll back.

Quando il volume di modifiche effettuate sotto un punto di salvataggio a livello di transazione diventa grande (10^4-10^6 record interessati) il motore rilascia il punto di salvataggio a livello di transazione e usa il meccanismo TIP per il roll back della transazione, se necessario. Se vi aspettate che il volume di modifiche nella vostra transazione sia grande, potete usare il flag TPB isc_tpb_no_auto_undo per evitare che venga creato il punto di salvataggio automatico a livello di transazione.

Punti di salvataggio e PSQL
L'implementazione di punti di salvataggio utente nel layer PSQL non rispetterebbe la regola dell'atomicità dei comandi, inclusi i comandi di chiamata di procedure. Firebird fornisce la gestione delle eccezioni in PSQL per annullare le modifiche eseguite in stored procedure e trigger. Ogni comando SQL/PSQL è eseguito sotto un sistema di punti di salvataggio interni automatici, per cui o l'intero comando viene completato con successo oppure TUTTE le sue modifiche subiscono un roll back e viene sollevata un'eccezione. Ogni blocco di gestione delle eccezioni in PSQL è anche confinato tra punti di salvataggio automatici di sistema.

(1.5) Blocco dei record esplicito

Nickolay Samofatov

L'aggiunta della clausola opzionale WITH LOCK fornisce una limitata esplicita capacità di blocco pessimistico dei record per un cauto uso in condizioni in cui il recordset affetto è a) molto piccolo (idealmente un singleton) e b) precisamente controllato dal codice dell'applicazione.

Il bisogno di un blocco pessimistico in Firebird è molto raro e dovrebbe in realtà essere ben compreso prima di considerare l'uso di questa estensione.

Sintassi

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

Se la clausola WITH LOCK ha successo, assicura un blocco sulle righe selezionate ed impedisce ad ogni altra transazione di ottenere accesso in scrittura ad alcuna di quelle righe, o loro dipendenti, fino alla fine della vostra transazione.

Se la clausola FOR UPDATE è inclusa, il blocco viene applicato ad ogni riga, una ad una, mentre è portata nella cache di riga dal lato server. Diventa allora possibile che un blocco che sembrava aver avuto successo quando richiesto, possa fallire successivamente, quando si fa un tentativo di leggere una riga che è stata bloccata da un'altra transazione.

É essenziale capire gli effetti dell'isolamento della transazione e di altri attributi della transazione prima di tentare di implementare il blocco esplicito dei record nelle vostre applicazioni.

Il costrutto SELECT... WITH LOCK è disponibile in DSQL e PSQL. Può avere successo solo in un comando al livello più alto di SELECT su una singola tabella. Non è disponibile nelle specifiche di subquery, né per insiemi joined. Non può essere specificato con l'operatore DISTINCT, una clausola GROUP BY o qualsiasi altra operazione di aggregazione. Non può essere usato in o con una view, né con una tabella esterna, né con l'output di una stored procedure selezionabile.

Comprendere la clausola WITH LOCK

Mentre il motore considera a turno ogni record che entra in un comando di blocco esplicito, rende o la versione del record che è stata oggetto più recentemente di commit, indipendentemente dallo stato del database quando il comando è stato inviato, oppure un'eccezione.

Il comportamento di attesa e di reportistica di conflitti dipende dai parametri della transazione specificati nel blocco TPB.

Modo TPB	***Comportamento***
isc_tpb_consistency	I blocchi espliciti sono superati da blocchi impliciti o espliciti a livello di tabelle e sono ignorati
isc_tpb_concurrency + isc_tpb_nowait	Se un record è modificato da qualsiasi transazione che sia stata soggetta a commit da quando la transazione che cerca di ottenere il blocco esplicito è iniziata, o se una transazione attiva ha fatto modifiche su questo record viene immediatamente sollevata eccezione di conflitto di aggiornamento.

Continua →

Modo TPB	*Comportamento*
isc_tpb_concurrency + isc_tpb_wait	Se un record è modificato da una transazione che ha avuto un commit da quando la transazione che cerca di ottenere il blocco esplicito è cominciata, viene subito sollevata un'eccezione di conflitto di aggiornamento. Se una transazione attiva sta mantenendo il possesso di questo record (tramite un blocco esplicito o un normale blocco ottimistico in scrittura) la transazione che tenta il blocco esplicito attende il risultato della transazione bloccante e, quando questa finisce, tenta di ottenere il blocco del record di nuovo. Ciò implica che se la transazione bloccante fa un commit di modifiche su quel record ne conseguirà un'eccezione di conflitto di aggiornamento.
isc_tpb_read_committed + isc_tpb_nowait	Se c'è una transazione attiva che mantiene il possesso di questo record (tramite un blocco esplicito o un normale aggiornamento), un'eccezione di conflitto di aggiornamento è immediatamente sollevata.
isc_tpb_read_committed + isc_tpb_wait	Se c'è una transazione attiva che mantiene il possesso di questo record (tramite blocco esplicito o normale blocco ottimistico in scrittura), la transazione che tenta il blocco esplicito attende l'esito della transazione bloccante e quando questa termina tenta di nuovo di ottenere il blocco sul record. Le eccezioni di conflitto di aggiornamento non vengono mai sollevate da un comando di blocco esplicito in questo modo TPB.

Quando un comando UPDATE tenta di accedere ad un record bloccato da un'altra transazione, o solleva eccezione di conflitto di aggiornamento oppure attende che la transazione che mantiene il blocco termini, a seconda del modo TPB. Il comportamento del motore qui è lo stesso che se il record fosse già stato modificato dalla transazione bloccante. I conflitti che implicano lock pessimistici non rendono codici gds particolari.

Il motore garantisce che tutti i record resi da un comando di blocco esplicito siano effettivamente bloccati e che RISPETTINO le condizioni di ricerca specificate nella clausola WHERE, fintanto che le condizioni di ricerca non dipendono da altre tabelle tramite join, subquery, ecc. Garantisce anche che le righe che non soddisfano le condizioni di ricerca non siano bloccate dal comando. NON può garantire che non ci siano righe che, sebbene soddisfino le condizioni di ricerca, non siano bloccate. Questa situazione può nascere se altre transazioni parallele eseguono un commit delle loro modifiche nel corso dell'esecuzione del comando di blocco.

Il motore blocca le righe al momento di trasferirle in memoria. Ciò ha importanti conseguenze se bloccate più righe alla volta. Molti metodi di accesso ai database Firebird per default producono l'output in pacchetti di qualche centinaio di righe alla volta ("buffered fetches" o letture bufferizzate). Molti componenti di accesso ai dati non sono in grado di riportarvi le righe contenute nell'ultimo pacchetto, dove è avvenuto un errore.

La clausola FOR UPDATE fornisce una tecnica per prevenire l'uso di letture bufferizzate, opzionalmente con OF <column-names> per abilitare aggiornamenti posizionali. Alternativamente, può essere possibile impostare nei vostri componenti di accesso la misura del buffer di lettura a 1. Ciò vi consente di processare la riga correntemente bloccata prima di leggere e bloccare la successiva, oppure di gestire gli errori senza effettuare il roll back della vostra transazione.

Il roll back di un punto di salvataggio implicito o esplicito rilascia I blocchi di record che erano stati ottenuti sotto quel punto di salvataggio, ma non notifica le transazioni in attesa. Le applicazioni non dovrebbero dipendere da questo comportamento perché potrebbe cambiare nel futuro.

Mentre i blocchi espliciti possono essere usati per prevenire e/o gestire errori di conflitti di aggiornamento anomali, il volume degli errori di deadlock (stallo) crescerà se non progetterete la vostra strategia di blocco con attenzione e la controllerete rigorosamente. La maggior parte delle applicazioni non necessita per nulla di blocchi espliciti. Lo scopo primario dei blocchi espliciti è (1) prevenire una costosa gestione di conflitti di aggiornamento in applicazioni pesanti e (2) mantenere l'integrità degli oggetti mappati in database relazionali in ambienti a cluster (grappolo). Se il vostro uso dei blocchi espliciti non cade in una di queste due categorie, allora è il modo sbagliato di svolgere il compito con Firebird.

Il blocco esplicito è una caratteristica avanzata, non abusatene ! Mentre soluzioni per questi tipi di problemi possono essere molto importanti per siti Web che gestiscono migliaia di scritture concorrenti o per sistemi ERP/CRM che operano in grosse multinazionali, la maggior parte degli applicativi non ha bisogno di lavorare in queste condizioni.

Esempi

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

ii) (multiple righe, elaborate una ad una con un cursore DSQL)
```
  SELECT * FROM DOCUMENT WHERE PARENT_ID=? FOR UPDATE WITH LOCK
```

(1.5) Miglioramento della gestione degli aggregati

Arno Brinkman

Originalmente gli insiemi aggregati potevano essere raggruppati solo su colonne con nome. In Firebird 1.0 è divenuto possibile fare il GROUP BY su un'espressione UDF. In 1.5 molte ulteriori estensioni alla gestione delle funzioni di aggregazione e alla clausola GROUP BY consente ora raggruppamenti in base al *grado delle colonne* nelle specifiche di output (la loro "posizione ordinale da sinistra a destra" con base 1, come nella clausola ORDER BY) oppure in base ad una varietà di espressioni.
NOTA: Non tutte le espressioni sono correntemente consentite nella lista GROUP BY. Per esempio la concatenazione non è consentita.

Sintassi Group By

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

group_by_list : group_by_item [, group_by_list];

group_by_item : column_name
                | degree (ordinal)
                | udf
                | group_by_function;

group_by_function : numeric_value_function
                | string_value_function
                | case_expression;

numeric_value_function  : EXTRACT '(' timestamp_part FROM value ')';

string_value_function   :  SUBSTRING '(' value FROM pos_short_integer ')'
                        | SUBSTRING '(' value FROM pos_short_integer FOR
nonneg_short_integer ')'
                        | KW_UPPER '(' value ')';
```

L'oggetto su cui si esegue il group_by non può essere un riferimento ad una funzione di aggregazione (incluse quelle inserite in un'espressione) dello stesso contesto.

HAVING
La clausola HAVING consente solo funzioni di aggregazione o espressioni valide che sono parte della clausola GROUP BY. Precedentemente era consentito l'uso di colonne che non erano parte della clausola GROUP BY e di usare espressioni non valide.

ORDER BY
Quando il contesto è un comando di aggregazione, la clausola ORDER BY consente solo espressioni valide che siano funzioni di aggregazione o parti di espressione della clausola GROUP BY. Precedentemente era consentito usare espressioni non valide.

Funzioni di aggregazione in subquery
É ora possibile usare funzioni di aggregazione o espressioni contenute nella clausola GROUP BY all'interno di una subquery.

Esempi
```
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
```

Mescolanza di funzioni di aggregazione di contesti diversi
Funzioni di aggregazione di diversi contesti possono essere usate in un'espressione.

Esempio
```
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! Questa query dà risultati in FB1.0, ma sono SBAGLIATI!

Supporto di subquery all'interno di funzioni di aggregazione
L'uso di un'espressione singleton select (un singleton select è un comando select tale per cui ci si aspetta che renda un singolo record) in una funzione di aggregazione è supportato.

Esempio
```
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
```

Funzioni di aggregazione nidificate
L'uso di una funzione di aggregazione all'interno di un'altra funzione di aggregazione è possibile se la più interna è di un contesto di più basso livello (vedi esempio sopra).

Raggruppamento per grado (numero ordinale)
L'uso del numero di grado della colonna di output nella clausola GROUP BY 'copia' l'espressione dalla lista list (come fa la clausola ORDER BY). Ciò implica che quando un numero di grado si riferisce ad una subquery, la subquery sia eseguita almeno due volte.

(1.5) La clausola ORDER BY può specificare espressioni e la posizione dei null

Nickolay Samofatov

La clausola ORDER BY lascia specificare qualsiasi valida espressione per ordinare i risultati di una query. Se l'espressione consiste di un singolo numero è interpretata come numero di colonna (grado), come precedentemente.
L'ordinamento dei null nell'insieme risultato può essere controllato usando la clausola di posizionamento dei null. I risultati possono essere ordinati in modo che i null siano piazzati o sopra (NULLS FIRST) o sotto (NULLS LAST) i non-null ordinati.
Il comportamento quando non si specifica nulls_placement è NULLS LAST.

Sintassi

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

Restrizioni

- Se si specifica NULLS FIRST, non verrà usato alcun indice per l'ordinamento.
- Il risultato di un sort basato su valori resi da una UDF o stored procedure sarà imprevedibile se i valori resi non possono essere usati per determinare una sequenza di ordinamento logica.
- Il numero di invocazioni di procedure dalla specifica di un sort basato su una UDF o stored procedure sarà imprevedibile, indipendentemente dal fatto che l'ordinamento sia specificato dall'espressione stessa o da un numero ordinale che rappresenta un'espressione nella specifica della lista di colonne.
- Una clausola di ordinamento per l'ordinamento dell'output di una query di unione può usare solo numeri ordinali (gradi) per riferirsi alle colonne di ordinamento.

Esempi

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 (Linguaggio per le stored procedure ed i trigger)

(1.5) EXECUTE STATEMENT

Alex Peshkov

Estensione PSQL che riceve una stringa che è un valido statement in dynamic SQL e lo esegue come se fosse stato inviato a DSQL.
Disponibile in trigger e stored procedure.

La sintassi può avere tre forme.

Sintassi 1

Esegue <string> come un'operazione SQL che non rende alcuna riga di dati, come INSERT, UPDATE, DELETE, EXECUTE PROCEDURE o qualsiasi statement DDL tranne CREATE/DROP DATABASE.

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

Esempio

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

Sintassi 2

Esegue <string> come operazione SQL, rendendo una singola riga di dati. Solo operatori singleton SELECT possono essere eseguiti con questa forma di EXECUTE STATEMENT.

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

Esempio

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

Sintassi 3

Esegue <string> come operazione SQL, rendendo multiple righe di dati. Qualsiasi operatore SELECT può essere usato con questa forma di EXECUTE STATEMENT.

```
FOR EXECUTE STATEMENT <string> INTO :var1, …, :varn DO
   <compound-statement>;
```

Esempi

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

Note aggiuntive su EXECUTE STATEMENT

La stringa DSQL 'EXECUTE STATEMENT' non può contenere parametri in nessuna variazione di sintassi. Tutte le sostituzioni di variabili nella parte statica dello statement SQL dovrebbero essere eseguite prima dell'esecuzione di EXECUTE STATEMENT.

Questa caratteristica è intesa solo per un uso molto cauto e dovrebbe essere usata tenendo conto di tutti i fattori. Come regola di massima usate EXECUTE STATEMENT solo quando altri metodi sono impossibili, o hanno prestazioni anche peggiori di EXECUTE STATEMENT.

EXECUTE STATEMENT è potenzialmente insicuro in molti modi:

1. Non c'è modo di controllare la correttezza della sintassi dello statement accluso.
2. I controlli di dipendenza non sono in grado di rilevare tabelle o colonne che sono state cancellate da un DROP.
3. Le operazioni saranno lente perché lo statement fornito deve essere preparato ad ogni esecuzione.
4. I valori resi sono controllati strettamente per il loro tipo di dati, per evitare eccezioni imprevedibili di type-casting. Per esempio, la stringa '1234' verrebbe convertita in un intero, 1234, ma 'abc' darebbe un errore di conversione.
5. Se la stored procedure ha speciali privilegi su qualche oggetto, lo statement dinamico inviato come stringa a EXECUTE STATEMENT non li eredita. I privilegi sono ristretti a quelli garantiti all'utente che esegue la procedura.

(1.5) Nuove variabili di contesto

Dmitry Yemanov

CURRENT_CONNECTION

e

CURRENT_TRANSACTION

Ognuna di queste variabili di contesto rende l'identificatore di sistema della connessione attiva o del contesto della transazione corrente, rispettivamente. Il tipo del dato reso è INTEGER. Disponibile in DSQL e PSQL. Poiché questi valori sono memorizzati nella pagina di intestazione del database, saranno ripristinati dopo un restore del database.

Sintassi

```
CURRENT_CONNECTION
CURRENT_TRANSACTION
```

Esempi

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

ROW_COUNT

Rende un integer, il numero di righe affette dall'ultimo statement DML. Disponibile in PSQL, nel contesto del modulo della procedure o trigger. Rende correntemente zero da uno statement SELECT.

Sintassi

ROW_COUNT

Esempio

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

Nota: questa variabile non può essere usata per il controllo delle righe affette da un comando EXECUTE STATEMENT.

SQLCODE e GDSCODE

Ogni variabile di contesto rende un intero che è il codice numerico di errore per l'eccezione attiva. Disponibile in PSQL, entro lo scope del particolare blocco di gestione delle eccezioni. Entrambi valgono zero fuori dal blocco.

La variabile GDSCODE rende una rappresentazione numerica del codice di errore GDS (ISC), cioè ad es. '335544349L' rende 335544349.

Un blocco di eccezione 'WHEN SQLCODE' o 'WHEN ANY' avrà un valore non nullo per la variabile SQLCODE e renderà zero per GDSCODE. Solo un blocco 'WHEN GDSCODE' avrà una variabile GDSCODE non a zero (e renderà zero in SQLCODE). Se è sollevata un'eccezione definita dall'utente, entrambe le variabili SQLCODE e GDSCODE conterranno zero, indipendentemente dal tipo del blocco di gestione delle eccezioni.

Sintassi

```
SQLCODE
GDSCODE
```

Esempio

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

Leggere anche MIGLIORAMENTI DELLA GESTIONE DELLE ECCEZIONI, sotto, ed il documento README.exception_handling nel ramo firebird2/doc/sql.extensions dell'albero CVS di Firebird.

INSERTING

UPDATING

DELETING

Tre espressioni pseudo-Booleane che possono essere valutate per determinare il tipo di operazione DML in esecuzione. Disponibile in PSQL, solo in un trigger. Intese per l'uso con trigger universali (vedere METADATA, sopra).

Sintassi

```
INSERTING
UPDATING
DELETING
```

Esempio

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

(1.5) Miglioramenti alla gestione delle eccezioni in PSQL

Dmitry Yemanov

La sintassi comune di uno statement EXCEPTION in PSQL è:

```
  EXCEPTION [name [value]];
```

I miglioramenti in 1.5 vi consentono di
1) definire un messaggio run-time per un'eccezione con nome.
2) re-iniziare (ri-sollevare) un'eccezione già catturata entro lo scope (visibilità) del blocco di eccezione
3) Ottenere un codice di errore numerico per l'eccezione catturata

1) Messaggistica di eccezione run-time

Sintassi

```
EXCEPTION <exception_name> <message_value>;
```

Esempi

i)

```
EXCEPTION E_EXCEPTION_1 'Errore!';
```

ii)

```
EXCEPTION E_EXCEPTION_2 'Tipo sbagliato per il record con ID=' || new.ID;
```

2) Ri-sollevare un eccezione

Nota – questo non ha effetti fuori da un blocco di gestione di eccezione.

Sintassi

```
EXCEPTION;
```

Esempi

i)

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

ii)

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

3) Codici di errore run-time

Veder SQLCODE / GDSCODE (sopra).

(1.5) Statement LEAVE | BREAK

Termina il flusso di esecuzione in un ciclo, facendo spostare il flusso di controllo allo statement che segue l'END che completa quel ciclo. Disponibile per statement WHILE, FOR SELECT e per costrutti di linguaggio FOR EXECUTE soltanto, altrimenti viene generato un errore del parser. La parola chiave LEAVE dello standard SQL-99 rende obsoleto e sconsigliato l'esistente BREAK. Disponibile in trigger e in stored procedure.

Sintassi

```
LEAVE;
```

Esempi

```
(i)
BEGIN
   <statements>;
   IF (<conditions>) THEN
     LEAVE;
   <statements>;
END

(ii)
WHILE (<condition>) DO
  BEGIN
    <statements>;
    WHEN ... DO
      LEAVE;
  END
```

NOTA Gli statement LEAVE | BREAK e EXIT possono ora essere usati nei trigger

(1.5) Ora è possibile includere nei trigger statement PLAN validi

Ignacio J. Ortega

Finora un trigger contenente uno statement PLAN sarebbe stato rifiutato dal compilatore. Ora un PLAN valido può essere usato.

(1.5) Blocchi BEGIN..END vuoti

Dmitry Yemanov

Sono ora legali blocchi BEGIN..END in moduli PSQL. Per esempio, si possono ora scrivere moduli "stub" (o segnaposto) come:

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

(1.5) Dichiarazione e definizione di variabili locali in singoli statement

Claudio Valderrama

Semplifica la sintassi e consente di dichiarare e definire (inizializzare) variabili locali in uno statement.

Sintassi

```
DECLARE [VARIABLE] name <variable_type> [{'=' | DEFAULT} value];
```

Esempio

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

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

(1.5) SELECT FIRST accetta ora zero come argomento

FB 1.5 ora consente zero come argomento di FIRST. Viene reso un insieme vuoto.

Rende le prime *m* righe dell'insieme risultato selezionato. La clausola opzionale SKIP fa scartare le prime *n* righe e rende un insieme risultato di *m* righe che iniziano a *n + 1*. Nella sua più semplice forma, *m* ed *n* sono interi ma ogni espressione Firebird che rende un intero è valida. Un identificatore che rende un intero può essere usato anche in GDML, sebbene non in SQL o DSQL.

Le parentesi sono richieste per argomenti di espressioni e opzionali altrimenti.

Possono anche impiegare variabili, ad es. SKIP ? * FROM ATABLE rende il restante dataset dopo aver scartato le *n* righe in testa, se n è passato nella variabile "?". SELECT FIRST ? COLUMNA, COLUMNB FROM ATABLE rende le prime *m* righe (se m è passato in "?") e scarta il resto.

La prima clausola FIRST è anche opzionale, cioè potete includere SKIP in uno statement senza FIRST per avere un insieme di output che semplicemente esclude le righe indicate da SKIP.

Disponibile in SQL e DSQL, eccetto dove diversamente indicato.

Esempi:

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

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

SELECT FIRST 5 DISTINCT FIELD FROM MYTABLE;
```

Due problemi con SELECT FIRST

1. Questo

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

cancellerà tutte le righe della tabella. Ahi! il sub-select valuta ogni 10 righe candidate alla cancellazione, le cancella, va avanti di altre 10…ad infinitum, finché non restano più righe. Attenzione!

2. Query come:

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

Non funzionano come ci si aspetta, perché l'ottimizzazione eseguita dal motore trasforma predicati correlati WHERE…IN (SELECT…) in predicati correlati EXISTS. É ovvio che in tale caso FIRST N non ha alcun senso:

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

Miglioramenti ai set di caratteri

Aggiunti in 1.5

- Aggiunta collazione WIN1251_UA (per il Russo e l'Ucraino) per i set di caratteri WIN1251.
- Corretto il maiuscolo predefinito per WIN1251
- Aggiunto ISO_HUN (per l'Ungherese) per i set di caratteri ISO8859_2

Nuovi set di caratteri (senza collazioni non-binarie) aggiunti

Blas Rodriguez Somoza

- DOS737 PC Greco
- DOS775 PC Baltico
- DOS858 Variante di Cp850 con la lettera Euro (€)
- DOS862 PC Ebreo
- DOS864 PC Arabo
- DOS866 MS-DOS Russo
- DOS869 IBM Greco Moderno
- WIN1255 Windows Ebreo
- WIN1256 Windows Arabo
- WIN1257 Windows Baltico
- ISO8859_3 Latin 3 (Esperanto, Maltese, Pinyi, Sami, Croato e altri)
- ISO8859_4 Latin 4 (Baltico, Groenlandico, Lappone)
- ISO8859_5 Cirillico
- ISO8859_6 Arabo
- ISO8859_7 Grecoi
- ISO8859_8 Ebreo
- ISO8859_9 Turco
- ISO8859_13 Baltico

Aggiunti in 1.0

- Aggiunto set di collazione Ungherese indipendente da maiuscolo/minuscolo, sviluppato e testato da Sandor Szollosi (ssani@freemail.hu).

- Firebird ora supporta il set di caratteri ISO8859-2 (per la lingua Ceca).

ESTENSIONI DEL LINGUAGGIO TRASPORTATE DA FIREBIRD 1.0.x

Le seguenti estensioni del linguaggio, introdotte in Firebird 1.0.x, sono qui riprodotte per vostra convenienza.

(1.0) CURRENT_USER e CURRENT_ROLE

Queste due nuove variabili di contesto sono state aggiunte per poter fare riferimento all'USER e (se implementato[1]) al ROLE del corrente contesto di connessione.

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

COMMIT;

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

CURRENT_USER è un sinonimo DSQL per USER che appare nello standard SQL. Sono identici, non c'è alcun vantaggio di CURRENT_USER rispetto a USER.

[1] Se si vuole usare un database InterBase v.4.x o 5.1 con Firebird, ROLE non è supportato, così current_role sarà NONE (come raccomandato dallo standard SQL in assenza di uno specifico ruolo) anche se l'utente passa un nome di ruolo. Se usate IB 5.5, IB 6 o Firebird, il ROLE passato è verificato. Se il ruolo non esiste, è reimpostato a NONE senza rendere un errore.

Ciò implica che in FB non potete ricevere da CURRENT_ROLE un ROLE non valido, perché sarebbe reimpostato a NONE. Ciò contrasta con IB, dove il valore di comodo è gestito internamente, sebbene non sia visibile a SQL.

(1.0) DROP GENERATOR

Consente di rimuovere generatori non usati dal database. Si libera spazio per il riutilizzo dopo il successivo RESTORE. Disponibile in SQL e DSQL.

```
DROP GENERATOR <generator name>;
```

(1.0) GROUP BY UDF

É ora possibile aggregare una SELECT facendo un grouping by sull'output di una UDF, ad es.:

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

Un effetto collaterale delle modifiche che consentono il group by sul risultato di una UDF è che, mentre prima non si potevano chiamare funzioni built-in di Firebird nei GROUP BY, ora, creando un wrapper di comodo, si può scrivere:

```
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

Questo nuovo comando DDL consente di creare una nuova stored procedure con lo stesso nome di una esistente, rimpiazzando la vecchia procedura senza bisogno di eseguirne prima il drop. La sintassi è identica a CREATE PROCEDURE.
Disponibile in SQL e DSQL.

(1.0) RECREATE TABLE

Questo nuovo comando DDL consente di creare una nuova struttura con lo stesso nome di una esistente, rimpiazzando la vecchia tabella senza bisogno di eseguirne prima il drop. La sintassi è identica a CREATE TABLE.

Osservate che RECREATE TABLE non preserva i dati nella vecchia tabella.

Disponibile in SQL e DSQL.

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

Funzione interna che implementa la funzione ANSI SQL SUBSTRING(). Rende uno stream di byte dalla posizione *<pos>* a tutti i successivi byte fino a fine stringa. Se si usa l'opzione FOR *<length>*, rende il minimo tra *<length>* byte o il numero di byte fino ala fine dell'input stream.

Il primo argomento può essere qualsiasi espressione, costante o identificatore che renda una stringa.
<pos> deve rendere un intero.
<pos> comincia a 1, come altri comandi SQL.
Né *<pos>* né *<length>* possono essere parametri di una query.

Poiché <pos> e <length> sono posizioni di byte, l'identificatore può essere un blob binario, o un blob di testo di sub_type 1, con un sottostante set di caratteri di un byte per carattere. La funzione correntemente non gestisce blob di testo con set di caratteri Cinese (2 byte/car massimo) o Unicode (3

byte/car massimo). Come argomento stringa (in alternativa a un blob), la funzione gestisce qualsiasi set di caratteri.

Disponibile in SQL e DSQL.

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

Riferitevi alla sezione sulle Funzioni Esterne (UDF) seguente per dettagli sulle modifiche e aggiunte alle funzioni sottostringa esterne nella libreria di UDF standard.

(1.5) Miglioramento al marcatore di commento su singola riga

Dmitry Yemanov

Commenti su singola riga possono essere in qualunque posizione sulla riga, non solo all'inizio.
Così in 1.5 il marcatore "--" può essere usato per un commento a fine riga di uno script, una stored procedure, un trigger o uno statement DSQL. Può così essere usato per "commentare via" parti indesiderate di uno statement. Tutti I caratteri dal marcatore "--" fino al prossimo carriage return or line feed saranno ignorati.

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

(1.0) Nuovo marcatore di commento

Claudio Valderrama

Da usare in script, DSQL, stored procedure e trigger.

Esempio

```
-- Questo è un commento
```

Questo nuovo marcatore può essere usato per "commentare via" una singola riga di codice in uno script, statement DDL/DML, stored procedure o trigger.

La logica è di ignorare i caratteri come segue:

1. Salta '--' se si trova come prima coppia di caratteri dopo un marcatore di fine-riga (LF sotto Linux/Unix, CRLF sotto Windows)
2. Continua a ignorare caratteri fino al prossimo marcatore fine-riga

Questa logica NON è intesa per essere usata contemporaneamente alla logica dei commenti a blocchi (/* un commento */). In altri termini, non usate lo stile di commento '--' entro un commento a blocco e non usate lo stile di commento a blocco entro una riga '--'.

SESSIONI INTERACTIVE ISQL: Ricordatevi di questo quando lavorate in sessioni interattive **isql**. **isql** accetta pezzi di statement in separati segmenti di continuazione, mostrando il prompt 'CON>' finché riceve un simbolo terminatore (normalmente '**;**'). Se battete una coppia '--' all'inizio di una riga di continuazione, la logica in base a cui vengono ignorati i caratteri del commento terminerà al marcatore di fine-riga che viene stampato sullo schermo del vostro file di OUTPUT quando premete Invio. C'è il potenziale per errori se successivamente aggiungete una continuazione, aspettandovi che sia ignorata.

Il problema con isql nasce perché ha i suoi speciali comandi, che devono essere riconosciuti solo da isql. Se non sono riconosciuti a causa della posizione di "--", allora vengono passati al motore. Ovviamente, il motore non capisce i comandi isql SET e SHOW e li rifiuta.

(1.0) Alter Trigger non incrementa più il contatore delle modifiche in una tabella

Quando il contatore delle variazioni ai metadati su una qualsiasi singola tabella arriva al massimo di 255, il database diventa non disponibile. Sono necessari un backup e un restore per reimpostare il contatore delle modifiche e rendere di nuovo il database disponibile. L'intenzione di questa caratteristica è costringere ad una pulizia del database quando una strutture di tabella hanno subito molte variazioni, non inibire utili capacità nel motore.

Precedentemente ogni volta che un trigger era impostato a ACTIVE|INACTIVE da uno statement ALTER TRIGGER, il contatore delle modifiche della tabella associata era incrementato. Ciò incideva sull'utilità di disabilitare e riabilitare il codice dei trigger per l'uso normale, poiché avrebbe causato una rapida crescita del contatore.

Nuove parole riservate

Le seguenti **nuove parole chiave Firebird** dovrebbero essere aggiunte alla lista delle parole riservate pubblicata per 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		

Le seguenti parole chiave sono riservate per usi futuri pianificati:

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

Le seguenti parole chiave erano riservate in Firebird 1.0 e non lo sono più in Firebird 1.5:

BREAK	DESCRIPTOR	FIRST
IIF	SKIP	SUBSTRING

Le seguenti parole non riservate sono riconosciute in 1.5 come parole chiave se usate nei loro rispettivi contesti strutturali:

COALESCE	DELETING	INSERTING
LAST	LEAVE	LOCK
NULLIF	NULLS	STATEMENT
UPDATING	USING	

Le seguenti **nuove parole chiave InterBase 6.5 e 7** (non riservate in Firebird) dovrebbero anche essere trattate come se lo fossero, per compatibilità:

BOOLEAN	FALSE	GLOBAL
PERCENT	PRESERVE	ROWS
TEMPORARY	TIES	TRUE

Caratteristiche ISQL

Capacità “readline” nella shell isql

Mark O'Donohue

Il supporto della storia dei comandi (come il readline di Unix) è stato aggiunto alla shell isql. Ora potete usare i tasti freccia Su e Giù per muovervi avanti e indietro sui comandi inviati nella sessione isql.

Funzioni definite dall’utente

In ib_udf

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

Juan Guerrero

Riempie a destra la stringa fornita *instring* aggiungendo caratteri *padcharacter*s finché la stringa risultante ha la data lunghezza *length*. La stringa di input può avere qualsiasi lunghezza inferiore a 32766 byte. La lunghezza non deve eccedere i 32765 byte.

Dichiarazione

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

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

Riempie a sinistra la stringa fornita *instring* anteponendo caratteri *padcharacter*s finché la stringa risultante ha la data lunghezza *length*. La stringa di input può avere qualsiasi lunghezza inferiore a 32766 byte. La lunghezza non deve eccedere i 32765 byte.

Dichiarazione

```
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

Questa funzione aveva un vecchio bug, per cui gli argomenti x e y erano erroneamente scambiati. Dovrebbe rendere il logaritmo in base x di y ma in effetti rendeva il logaritmo in base y di x. É stata corretta.
Se era usata nelle vostre applicazioni SI PREGA DI CONTROLLARE IL CODICE! Questo rendeva il risultato sbagliato oppure qualcuno in qualche momento per correzione dovette invertire gli argomenti deliberatamente per avere il calcolo corretto.

In fbudf

1. Le funzioni *NVL e *NULLIF rimangono per compatibilità all'indietro, ma sono sconsigliate dall'introduzione delle nuove funzioni interne CASE, COALESCE e NULLIF.
2. Si noti che fbudf non può gestire campi stringa più lunghi di 32Kb - 1 byte. Questo limite può avere effetti negativi se le stringhe sono concatenate prima di essere passate ad UDF che accettano argomenti stringa. Se la somma dei campi è oltre quel limite, il comportamento sarà non definito. La funzione può rendere un risultato senza senso o il codice fbudf può eseguire un'operazione illegale.
3. Se state portando un database creato in Firebird 1.0.x e avete dichiarato le funzioni di fbudf **truncate** e **round**, quelle dichiarazioni non funzioneranno più con Firebird 1.5, perché i nomi degli entry_point sono stati cambiati. Dovrete fare un drop delle funzioni e ridichiararle usando le dichiarazioni dello script fbudf.sql nella directory /UDF di 1.5.

Nuovo file di configurazione – firebird.conf

La directory radice (root) di Firebird

La directory radice (root) della vostra installazione Firebird è usata in molti modi, sia durante l'installazione che come attributo da cui le routine del server, i parametri di configurazione ed i client dipendono. Siccome vi sono parecchi modi per dire al server dove trovare il valore per questo attributo, gli sviluppatori e gli amministratori di sistema dovrebbero essere consci delle precedenze che il server segue all'avvio per determinarlo correttamente.

Win32 Superserver e Classic build (sia server che client):
1) Variabile di ambiente FIREBIRD
2) Parametro RootDirectory in firebird.conf
3) Registry:
   HKLM\SOFTWARE\Firebird Project\Firebird Server\Instances\DefaultInstance
   e si cerca il campo DefaultInstance.
4) La directory un livello sopra quello dove si trova l'eseguibile del server

Win32 Embedded:
1) Variabile di ambiente FIREBIRD
2) Parametro RootDirectory in firebird.conf
3) La directory dove si trova fbembed.dll (rinominato fbclient.dll)

Linux Classic:
1) Variabile di ambiente FIREBIRD
2) Parametro RootDirectory in firebird.conf
3) Percorso predefinito di installazione (/opt/firebird)

Linux Superserver:
1) Variabile di ambiente FIREBIRD
2) Parametro RootDirectory in firebird.conf
3) La directory un livello sopra quella dove si trova il file binario del server (recuperato tramite il symlink "/proc/self/exe", se supportato)
4) Percorso di installazione predefinito (/opt/firebird)

Parametri

I valori predefiniti sono applicabili alla maggior parte dei parametri. I nomi e valori dei parametri sono sensibili a maiuscole/minuscole sotto Linux, ma non sotto Windows. Per impostare qualsiasi parametro ad un valore diverso dal predefinito, cancellate il marcatore di commento (#) e modificate il valore. Potete modificare il file di configurazione mentre il server è in esecuzione. Per attivare le modifiche di configurazione, è però necessario fermare e riavviare il servizio.
Le voci sono nella forma:

parameter_name *value*

- parameter_name è una stringa NON contenente spazi e cita il nome della proprietà del server che viene configurata.

- *value* è un numero, valore logico (1=True, 0=False) o stringa che specifica il valore del parametro

Parametri connessi al filesystem

RootDirectory

Stringa, il percorso assoluto alla directory root nel filesystem locale. Dovrebbe restare commentato, a meno che non vogliate forzare la procedura di avvio a scegliere un percorso diverso dal predefinito per la root dell'installazione del server Firebird, che altrimenti viene riconosciuta da sola.

DatabaseAccess

Supporta la caratteristica del database-aliasing. In precedenti versioni, il server poteva connettersi a qualsiasi database nel suo filesystem locale e forniva l'accesso alle applicazioni passando il percorso assoluto del file nel filesystem. Questo parametro offre l'opzione di restringere l'accesso del server solo ai database con alias, o solo ai database posti in specifici punti dell'albero del filesystem.
DatabaseAccess può essere None, Restrict o Full.
Full (valore predefinito) permette l'accesso ai file database ovunque nel filesystem locale.
None permette al server di connettersi solo ai database elencati in **aliases.conf**.
Restrict consente di limitare la posizione dei file database a liste specificate di rami del filesystem. Fornite una lista di uno o più origini, separate da punto e virgola, per definire una o più posizioni permesse.
Per esempio,
Unix: `/db/databases;/userdir/data`
Windows: `D:\data`
I percorsi relativi sono trattati come relativi al percorso che il server in esecuzione riconosce come root directory. Ad esempio, sotto Windows, se la root directory è C:\Programmi\Firebird, allora il seguente valore limita il server ad accedere ai file solo se sono posti sotto C:\Programmi\Firebird\userdata:

```
DatabaseAccess = Restrict userdata
```

ExternalFileAccess

Era *external_file_directory* in isc_config/ibconfig ma la sintassi è cambiata.

Fornisce tre livelli di sicurezza per EXTERNAL FILES (file di testo a formato fisso che devono essere usati come tabelle di database). Il valore è una stringa, che può essere None, Full o Restrict.
None (il valore predefinito) disabilita qualsiasi uso di file esterni sul vostro server.
Restrict dà la possibilità di limitare la posizione dei file esterni per l'accesso database a specifici percorsi. Fornite una lista di uno o più punti di partenza di alberi del filesystem, separati da punto e virgola (;), entro e sotto cui i file esterni possono essere memorizzati.
Per esempio,
Unix: `/db/extern;/mnt/extern`
Windows: `C:\ExternalTables`
I percorsi relativi sono trattati come relativi al percorso che il server in esecuzione riconosce come la root directory dell'installazione Firebird.
Per esempio, sotto Windows, se la root che il server in esecuzione riconosce come root directory dell'installazione Firebird è C:\Programmi\Firebird, allora il seguente valore limita il server ad accedere ai soli file esterni che si trovano in C:\Programmi\Firebird\userdata\ExternalTables:

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

Full permette l'accesso a file esterni ovunque nel sistema.

Vedere **ATTENZIONE** sotto il prossimo argomento, UdfAccess.

UdfAccess

era *external_function_directory* in isc_config/ibconfig ma la sintassi è cambiata.

Non solo sostituisce il nome del vecchio parametro ma anche la forma in cui i valori sono presentati. Lo scopo delle modifiche è consentire livelli opzionali di protezione per librerie esterni di moduli definiti dall'utente, un noto bersaglio di attacchi malintenzionati di intrusi. UdfAccess può essere None, Restrict o Full.

Restrict (il valore predefinito) mantiene la funzionalità fornita dal parametro **external_function_directory** in Firebird 1.0, di limitare la posizione delle librerie esterne invocabili a specifiche posizioni del filesystem. Fornite una lista di uno o più punti iniziali di alberi del filesystem, separati da punto e virgola (;), entro e sotto le quali le definizioni di UDF, di filtri BLOB e di set di caratteri possono essere memorizzate.
Per esempio,
Unix: `/db/extern;/mnt/extern`
Windows: `C:\ExternalModules`
I percorsi relativi sono trattati come relativi al percorso che il server in esecuzione riconosce come root directory. Ad esempio, sotto Windows, se la root directory è C:\Programmi\Firebird, allora il seguente valore limita il server ad accedere ai file solo se sono posti sotto C:\Programmi\Firebird\userdata\ExternalModules:

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

None impedisce ogni uso di librerie esterne definite dall'utente.

Full permette di accedere a librerie esterne ovunque poste nel sistema.

ATTENZIONE: evitate di impostare alberi di directory personalizzati per UdfAccess e ExternalFileAccess tali per cui condividano una directory root padre. Le impostazioni predefinite sono sicure. Se state impostando le vostre e non create directory separate, allora il server potrà essere manomesso facilmente per fargli eseguire codice non autorizzato. Un esempio di ciò che è da evitare:

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

UdfAccess ed ExternalFileAccess qui hanno un sotto-albero in comune, `/bad_dir/files`, dove qualcuno può piazzare il suo file esterno `/bad_dir/files/hackudf.so` ed eseguire il suo codice sul sistema compromesso.

Parametri relativi alle risorse

CpuAffinityMask

era *cpu_affinity* in isc_config/ibconfig

Con Firebird SuperServer sotto Windows, c'è un problema con il sistema operativo che continuamente scambia l'intero processo del SuperServer avanti ed indietro tra i processori su macchine SMP. Questo danneggia le prestazioni. Questo parametro può essere usato su sistemi SMP sotto Windows per impostare il l'affinità di processore del SuperServer Firebird ad una singola CPU.

CpuAffinityMask accetta un integer, la CPU mask.

Esempio

```
CpuAffinityMask = 1
```

Esegue solo sulla prima CPU (CPU 0).

`CpuAffinityMask = 2`
Esegue solo sulla seconda CPU (CPU 1).

`CpuAffinityMask = 3`
Esegue sia sulla prima che sulla seconda CPU.

Il calcolo del valore della maschera di affinità
Potete usare questo flag per impostare l'affinità di Firebird a qualsiasi singolo processore o (su un server Classic) qualsiasi combinazione delle CPU installate nel sistema.
Considerate le CPU come una matrice numerata da 0 a *n*-1, dove *n* è il numero dei processori installati e *i* è l'indice di array di una CPU. M è un altro array, contenente i MaskValue di ogni CPU scelta. Il valore A è la somma dei valori in M.
Usate la seguente formula per arrivare a M e calcolare il MaskValue A:

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

Per esempio, per selezionare il primo e quarto processore (processore 0 e processore 3) calcolate come segue:

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

DeadlockTimeout

era *deadlock_timeout* in isc_config/ibconfig

Numero di secondi (integer) che il manager di blocco attende dopo che si verifica un conflitto, prima di eliminare i blocchi dei processi morti ed eseguire un ulteriore ciclo di ricerca dei deadlock. Normalmente, il motore riconosce istantaneamente i deadlock. Il timeout dei deadlock scatta se qualcosa non funziona.

Il default di 10 secondi è normalmente adatto alla maggior parte delle condizioni. Valori più bassi non migliorano necessariamente la velocità con cui i deadlock problematici rendono un'eccezione di conflitto. Se il valore è troppo basso, il risultato possono essere ulteriori, non necessarie ricerche di deadlock, che degradano le prestazioni del sistema.

DefaultDbCachePages

era *database_cache_pages* in isc_config/ibconfig

Valore predefinito a livello di server (integer) del numero delle pagine di database da allocare in memoria per database. Il valore configurato può essere superato da impostazioni a livello di database.
Il valore predefinito per il SuperServer è 2048 pagine. Per il Classic, è 75.
SuperServer e Classic usano la cache differentemente: SS mette in condivisione la sua cache per l'uso da parte di ogni connessione; Classic alloca una cache statica per ogni connessione.

EventMemSize

Intero, rappresenta il numero di byte di memoria riservati per il manager degli eventi. Il default è 65536 (64 Kb).

LockAcquireSpins

era *lock_acquire_spins*

É rilevante solo su macchine SMP con Classic server in esecuzione. Con Classic server, solo un processo client può accedere alla tabella dei blocchi in un certo istante. Un mutex governa l'accesso alla tabella dei blocchi. I processi client possono richiedere il mutex condizionatamente o incondizionatamente. Se è condizionale, la richiesta fallisce e deve essere ritentata. Se è incondizionata, la richiesta attende finché è soddisfatta. LockAcquireSpins stabilisce il numero di tentativi che saranno effettuati se la richiesta del mutex è condizionata.
É un integer, con valore predefinito 0 (incondizionata). Non c'è un minimo o massimo raccomandato.

LockHashSlots

Era *lock_hash_slots* in isc_config/ibconfig

Usate questo parametro per mettere a punto la lista di hash dei blocchi. Sotto carico pesante, si può migliorare il throughput aumentando il numero di slot hash, per disperdere la lista in catene di hash più corte. Si raccomandano valori interi che siano numeri primi. Il default è 101.

LockGrantOrder

era *lock_grant_order* in isc_config/ibconfig

Quando una connessione vuole un blocco su un oggetto, si procura una struttura di richiesta del blocco che specifica l'oggetto ed il livello di blocco richiesto. Ogni oggetto bloccato ha una struttura di richiesta di blocco. I blocchi richiesti sono connessi a queste strutture o come blocchi richiesti e concessi o come richieste in corso.
Il parametro LockGrantOrder è un Boolean. Il valore predefinito (1=True) indica che i blocchi sono concessi con una politica di primo-arrivato-primo-servito. L'impostazione False (0), che emula il comportamento di InterBase v3.3, concede il blocco appena diventa disponibile. Può dare il risultato che le richieste di blocco appaiano "congelate".

LockMemSize

Questo parametro integer rappresenta il numero di byte di memoria condivisa allocati per il gestore dei blocchi. Per un Classic server, LockMemSize dà l'allocazione iniziale, che poi cresce dinamicamente fino all'esaurimento della memoria (*"Il gestore dei blocchi non ha più spazio - Lock manager is out of room").* se ci sono molte connessioni o pagine cache molto ampie, aumentate questo parametro per evitare questi errori.

Se si usa SuperServer, la memoria allocata per il gestore dei blocchi non cresce.
Il valore predefinito su Linux e Solaris è 98304 byte (96 Kb). Sotto Windows, è 262144 (256 Kb).

LockSemCount

Parametro integer, specifica il numero di semafori disponibili per la comunicazione interprocesso (IPC). Il valore predefinito è 32. Impostate questo parametro in ambienti non-threading per aumentare o diminuire il numero di semafori disponibili.

SortMemBlockSize

Questo parametro vi consente di configurare, in byte, la dimensione di ogni blocco di memoria usato nel modulo di ordinamento in-memoria. Il valore predefinito di installazione è 1 Mb; potete riconfigurarlo a qualsiasi dimensione fino alla massimo valore consentito correntemente configurato, impostato con il parametro SortMemUpperLimit (vedi sotto).

SortMemUpperLimit

La massima quantità di memoria, in byte, da allocare al modulo di ordinamento in-memoria. Il valore predefinito di installazione è 67108864 bytes (64 Mb) per il SuperServer e 8388608 (8 Mb) per il Classic server.

> CAUTION Se usate Classic, tenete a mente che aumentare o la dimensione del blocco o il limite massimo ha effetto su ogni connessione di client/istanza del server e farà crescere di conseguenza il consumo di memoria del server.

Parametri relativi alle comunicazioni

ConnectionTimeout

era *connection_timeout* in isc_config/ibconfig

Numero di secondi da attendere prima di abbandonare un tentativo di connessione. Il valore predefinito è 180.

DummyPacketInterval

era *dummy_packet_interval* in isc_config/ibconfig

Questo è il numero di secondi (un integer) che il server deve attendere su una connessione inattiva prima di inviare pacchetti fittizi al fine di causare una risposta.
NON USATE QUESTA OPZIONE su un server Win32 che abbia client TCP/IP. Causa un persistente aumento nell'uso di memoria non-paginata del kernel che può mandare in blocco o addirittura in crash Windows dal lato client, come spiegato qui:
http://support.microsoft.com/default.aspx?kbid=296265

Escludendo Win32-con-TCP/IP, è l'unico modo per riconoscere e disconnetter client inattivi quando si usano i protocolli NamedPipes (NetBEUI), XNET o IPC. Non ci sono problemi noti su sistemi POSIX.

Normalmente, Firebird usa l'opzione socket SO_KEEPALIVE per tener traccia delle connessioni attive. Se non vi piace il valore predefinito di timeout keepalive di due ore, modificate le impostazioni del S.O. del vostro server in modo appropriato:

- su S.O. tipo UNIX, modificate i contenuti di /proc/sys/net/ipv4/tcp_keepalive_*.
- Sotto Windows, seguite le istruzioni di questo articolo:

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

Il valore predefinito dovrebbe essere 0 – non 60 che era il vecchio valore predefinito in Firebird 1.0 e la maggior parte dei candidati al rilascio della versione 1.5. Un'impostazione di 60 dovrebbe essere trattata come il valore predefinito sui sistemi dove dovete usare questo polling con pacchetti fittizi.

RemoteServiceName

Default = gds_db

RemoteServicePort

Questi due parametri forniscono la possibilità di forzare o il nome del servizio TCP/IP o il numero della porta TCP/IP usato per ascoltare le richieste di connessione a database dei client, se una di queste differisce dal valore predefinito dell'installazione (gds_db/tcp 3050).

Cambiate uno o l'altro dei valori, non entrambi. RemoteServiceName è controllato prima per vedere se esiste un corrispondente elemento nel file dei servizi. Se c'è, viene usato il numero di porta configurato in RemoteServicePort. Se non c'è, viene usata la porta 3050, il valore predefinito di installazione.

NOTA Se un numero di porta viene fornito nella stringa di connessione TCP/IP, ha sempre la precedenza su RemoteServicePort.

RemoteAuxPort

Il comportamento ereditato da InterBase di passare indietro allo strato di rete i messaggi di notifica di eventi attraverso porte TCP/IP selezionate a caso è stato una causa persistente di errori di rete e conflitti con firewall, a volte fino al punto di causare un crash del server in determinate condizioni. Questo parametro consente di configurare una singola porta TCP per tutto il traffico di notifica degli eventi.
Il valore predefinito di installazione (0) mantiene il tradizionale comportamento con porte casuali. Per dedicare una porta specifica agli eventi di notifica, usate un integer corrispondente ad una porta disponibile.

RemoteBindAddress

Per default, i client possono connettersi da qualsiasi interfaccia di rete attraverso cui l'host accetti il traffico. Questo parametro consente di accoppiare il servizio Firebird alle richieste in arrivo attraverso una scheda NIC e di rifiutare le richieste di connessione da qualsiasi altra interfaccia di rete. Questo dovrebbe aiutare a risolvere i problemi in alcune sottoreti in cui il server gestisce traffico attraverso NIC multiple.
Si tratta di una stringa, in formato IP valido (XXX.XXX.XXX.XXX). L'impostazione predefinita (nessun accoppiamento specifico) si rappresenta non fornendo alcun valore.

TcpRemoteBufferSize

Il motore legge in anticipo rispetto al client e può inviare parecchie righe di dati in un singolo pacchetto. Maggiore è la dimensione del pacchetto, più righe di dati sono inviate ad ogni trasferimento. Usate questo parametro—con cautela e completa comprensione dei suoi effetti sulle prestazioni della rete!—se vi serve aumentare o ridurre la dimensione del pacchetto TCP/IP per i buffer di invio e ricezione. Ciò ha conseguenze sia sul server che sul client.
Il valore è un integer (dimensione del pacchetto in byte) nell'intervallo da 1448 a 32768. Il valore predefinito di installazione è 8192.

Parametri specifici POSIX

LockSignal

Parametro intero, segnale UNIX da usare per le comunicazioni interprocesso. Default: 16

RemoteFileOpenAbility

USATE SOLO CON ESTREMA CAUTELA

Parametro boolean che, se impostato a True, consente al motore di aprire file database che risiedono su una partizione montata di un filesystem di rete (NFS). Siccome il filesystem è fuori del controllo del sistema locale, questa è una *caratteristica molto rischiosa* che non dovrebbe essere attivata allo scopo di aprire in lettura e scrittura alcun database la cui sopravvivenza vi interessi.
Il valore predefinito è 0 (False, disabilitato) e dovreste lasciarlo così a meno che non vi siano estremamente chiari i suoi effetti.

TcpNoNagle

era *tcp_no_nagle* in isc_config/ibconfig

Sotto Linux, per default, la libreria socket minimizza le scritture fisiche bufferizzandole prima di inviare effettivamente i dati, usando un algoritmo interno (implementato come opzione TCP_NODELAY della connessione socket) noto come Algoritmo di Nagle. É stato progettato per evitare problemi con piccoli pacchetti, chiamati tinygrams, su reti lente.
Per default, TCP_NODELAY è abilitato (valore 0) se Firebird Superserver è installato sotto Linux. Su reti lente, si può in realtà migliorare la velocità disabilitandolo. Attenti alla doppia negazione—impostate il parametro a True per disabilitare TCP_NODELAY e a False per abilitarla.
In release fino a ed inclusa la v.1.5, questa caratteristica è attiva solo per il Superserver.

Parametri specifici di Windows

CreateInternalWindow

Il protocollo "Windows local" usa una finestra nascosta per le comunicazioni interprocesso tra client locale e server. Questa finestra IPC è creata all'avvio del server quando CreateInternalWindow è True (1, il default). Impostatela a 0 (spento) per eseguire il server senza una finestra e quindi disabilitare il protocollo locale. Con il protocollo locale disabilitato è possibile eseguire istanze multiple del server simultaneamente.

DeadThreadsCollection

Un'impostazione per lo scheduler dei thread di Windows, questo parametro intero stabilisce il numero di cicli di cambio di priorità (vedere PrioritySwitchDelay, sotto) che lo scheduler deve eseguire prima che un thread sia distrutto (o chiuso).
La distruzione immediate (o chiusura) di thread di lavoro richiederebbe un semaforo e chiamate di blocco, che generano un significativo lavoro in più. Invece, lo scheduler di thread mantiene i thread in un pool. Quando un thread ha completato il suo lavoro, è marcato come in ozio. Il thread ozioso è distrutto (o chiuso) dopo *n* iterazioni del loop dello scheduler, dove *n* è il valore del parametro DeadThreadsCollection.
Per un server che gestisce un alto numero di connessioni—parecchie centinaia o più—il valore del parametro andrà aumentato Dal suo valore predefinito di 50.

GuardianOption

Parametro boolean usato sotto server Windows per determinare se il Guardian deve riavviare il server ogni volta che questo termina in modo anormale. Il valore predefinito di installazione prevede di fare così (1=True). Per disabilitare il comportamento di riavvio impostate questo parametro a spento (0=False).

IpcMapSize

era *server_client_mapping* in ibconfig

Misura in byte della porzione di memoria riservata ad un client del file mappato in memoria usato per le comunicazioni interprocesso (IPC) nel modello di connessione usato per la connessione "Windows locale". Non ha equivalente su altre piattaforme. Integer, da 1024 a 8192. Il default è 4096.
Aumentando la dimensione della mappa si può migliorare le prestazioni se si usano dataset con record molto grandi o numerosi, ad es. Quelli che rendono BLOB grafici.

> NOTA Questo non può più essere alterato nel dialog invocato dall'icona di Guardian nel system tray.

IpcName

Valore predefinito: FirebirdIPI
Il nome dell'area di memoria condivisa usata come canale di trasporto nel protocollo locale.
Il valore predefinito nella Release 1.5—FirebirdIPI—non è compatibile con le release precedenti di Firebird né con InterBase®. Usate il valore InterBaseIPI per ripristinare la compatibilità, se occorre.

MaxUnflushedWrites

Parametro introdotto nella Versione 1.5 per gestire un errore del server nel sistema operativo Windows, per cui le scritture asincrone non erano mai fisicamente scritte su disco eccetto quando il server Firebird subiva una chiusura controllata. (Le scritture asincrone non sono supportate in Windows 9x o ME.) Quindi su sistemi 24/7 (accesi 24 ore al giorno per sette giorni su sette), le scritture asincrone non erano mai inviate fisicamente su disco.
Questo parametro determina con che frequenza le pagine trattenute sono scritte sul disco quando le scritture forzate (Forced Writes) sono disabilitate (ed è abilitata la scrittura asincrona). Il suo valore è un integer che imposta il numero di pagine trattenute in memoria prima che una scrittura su disco venga marcata come da farsi la volta successiva che una transazione subisce un commit. Il valore predefinito è 100 in installazioni Windows e -1 (disabilitato) nelle installazioni per ogni altra piattaforma.
Se la fine del ciclo MaxUnflushedWriteTime (vedi sotto) è raggiunta prima che il conteggio delle pagine trattenute in memoria raggiunga il valore MaxUnflushedWrites, allora la scrittura fisica è avviata immediatamente ed il conteggio delle pagine trattenute è reimpostato a zero.

MaxUnflushedWriteTime

Questo parametro determina la massima lunghezza del tempo per cui le pagine trattenute in memoria per la scrittura asincrona restano in tale stato quando Forced Writes è disabilitato (la scrittura asincrona è abilitata). Il suo valore è un integer che imposta l'intervallo, in secondi, tra l'ultima scrittura fisica su disco e l'impostazione del flag che indica di effettuare una scrittura fisica su disco la successiva volta che una transazione subisce un commit. Il valore predefinito è 5 secondi in installazioni Windows e -1 (disabilitato) in installazioni su ogni altra piattaforma.

PrioritySwitchDelay

Un'impostazione dello scheduler dei thread sotto Windows, questo integer stabilisce il tempo, in millisecondi, che deve trascorrere prima che la priorità di un thread inattivo sia ridotta a LOW (bassa) o la priorità di un thread attivo sia avanzata a HIGH (alta). Una iterazione di questa variazione di priorità rappresenta un ciclo dello scheduler dei thread.
Il valore predefinito è 100 ms, scelto sulla base di esperimenti su processori Intel PIII/P4. Per processori con velocità di clock inferiori sarà richiesto un ritardo maggiore.

PriorityBoost

Integer, imposta il numero di cicli extra dati ad un thread quando la sua priorità diventa HIGH (alta). Il valore predefinito di installazione è 5.

ProcessPriorityLevel

era *server_priority_class* in ibconfig
Livello/classe di priorità del processo del server. Questo parametro sostituisce il parametro server_priority_class delle release pre-1.5—vedi sotto—con una nuova implementazione.
I valori sono interi, come segue:

- 0 - normale priorità,
- valore positivo – alta priorità (lo stesso dello switch -B[oostPriority] delle opzioni *configure* r *start* di instsvc.exe)
- valore negativo – bassa priorità.

Nota: Tutte le variazioni a questo valore dovrebbero essere verificate attentamente per assicurare che effettivamente migliorino la prontezza a rispondere alle richieste del motore.

RemotePipeName

Applicabile solo per connessioni NetBEUI
Parametro string, nome del pipe usato come canale di trasporto nel protocollo NetBEUI. Il named pipe è equivalente ad un numero di porta per TCP/IP. Il valore predefinito—interbas— è compatibile con release precedenti di Firebird e con InterBase®.

Parametri per la configurazione dello spazio di ordinamento temporaneo

Quando la dimensione di un buffer di ordinamento interno è troppo piccola per accettare le righe interessate da una operazione di ordinamento, Firebird deve creare file di ordinamento temporanei sul filesystem del server. Per default, cerca il percorso specificato nella variabile di ambiente **FIREBIRD_TMP**. Se questa variabile non è presente, cerca di usare la directory radice del filesystem **/tmp** sotto Linux/UNIX, o **C:\temp** sotto Windows NT/2000/XP. Nessuna di queste locazioni può essere configurata in dimensione.
Firebird fornisce un parametro per configurare lo spazio disco che sarà usato per memorizzare questi file temporanei. É prudente usarlo, per assicurarsi che sufficiente spazio di ordinamento sia disponibile in ogni condizione.
Tutte le richieste CONNECT o CREATE DATABASE condividono la stessa lista di directory per i file temporanei ed ognuno crea i suoi file temporanei. I file di ordinamento sono rilasciati quando il sort è finito o la richiesta è rilasciata.
Nella Release 1.5, il nome del parametro è cambiato da **tmp_directory** a **TempDirectories** e la sintassi del valore del parametro è cambiata anch'essa.

TempDirectories

Sostituisce le voci *tmp_directory* in isc_config/ibconfig

Fornisce una lista di una o più directory, separate da punto e virgola (;), sotto cui i file di ordinamento possono essere memorizzati. Ogni voce può includere un parametro opzionale che specifichi la dimensione, in byte, per limitarne l'ingombro. Se l'argomento è omesso o è non valido, Firebird userà lo spazio in quella directory finché non lo esaurisce, prima di spostarsi alla successiva directory in lista.
Per esempio,
Unix: `/db/sortfiles1 100000000;/firebird/sortfiles2`
Windows: `E:\sortfiles 500000000`
I percorsi relativi sono trattati relativi al percorso che il server in esecuzione riconosce come la root directory dell'installazione Firebird. Per esempio, sotto Windows, se la root directory è C:\Programmi\Firebird, allora il valore seguente dirà al server di memorizzare i file temporanei di ordinamento in C:\Programmi\Firebird\userdata\sortfiles, fino al limite di 500 Mb:

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

NB I percorsi non sono più tra virgolette come era necessario in Firebird 1.0

Parametri di compatibilità

CompleteBooleanEvaluation

Stabilisce il metodo di valutazione dei valori logici (completo o cortocircuitato). Il default (0=False) è di "cortocircuitare" un'espressione boolean da valutare, se implica i predicati AND o OR, ritornando appena si verifica l'ottenimento di un risultato True o False che non può più essere alterato dalla valutazione delle restanti parti dell'espressione.
Sotto condizioni molto rare (di solito evitabili), può avvenire che un'operazione entro una condizione OR o AND che rimane non valutata per il comportamento di cortocircuitazione, abbia il potenziale di cambiare il risultato. Se avete la sfortuna di ereditare un'applicazione che ha tali caratteristiche nella sua logica SQL, potreste voler usare questo parametro per forzare la valutazione completa finché non avrete l'opportunità di "operarla". Il tipo di parametro è Boolean.

> Non sottovalutate il fatto che le modifiche a questo flag hanno effetto su tutte le valutazioni di Boolean e su ogni database del server.

OldParameterOrdering

La versione 1.ha indirizzato e risolto un vecchio problema di InterBase che causava la restituzione di parametri di output al client con un ordinamento incompatibile delle strutture XSQLDA. L'errore era talmente di lunga data che molte applicazioni esistenti, driver e componenti di interfaccia hanno soluzioni interne per correggere il problema dal lato client.
La release 1.5 e successive riflettono la corretta condizione nelle API e sono installate con OldParameterOrdering=0 (False). Impostate questo parametro boolean a True se vi serve tornare alla vecchia condizione per compatibilità con il codice esistente.

Aliasing dei file DB

Firebird release 1.5 introduce l'aliasing dei file database per migliorare la portabilità di applicazioni e per restringere il controllo sia dell'accesso interno che esterno ai file database.

Aliases.conf

Configurate gli alias dei file database nel file di testo aliases.conf, posto nella root directory della vostra installazione del server Firebird. Il file aliases.conf installato è simile a questo:

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

Come in tutti i file di configurazione di Firebird, i simboli '#' sono marcatori di commenti. Per configurare un alias, semplicemente cancellate il '#' e cambiate la riga fittizia con l'appropriato percorso al database:

```
# fbdb1 è su un server Windows:
```

```
fbdb1 = c:\Firebird\sample\Employee.fdb
# fbdb2 è su un server Linux
fbdb2 = /opt/databases/killergames.fdb
#
```

Potete modificare aliases.conf mentre il server è in esecuzione. Non c'è bisogno di fermarlo e riavviarlo per fargli riconoscere le nuove voci di aliases.conf.

Connessione usando un percorso database con alias

La stringa di connessione modificata nella vostra applicazione client ha questo aspetto:

```
Server_name:aliasname
```

Con l'esempio sopra, la seguente stringa di connessione chiederà al server Firebird in esecuzione su un Linux box chiamato "myserver" di trovare e connettere il client al database che si trova al percorso identificato in aliases.conf come "fbdb2":

```
myserver:fbdb2
```

Nota: poiché lo **strumento gstat** non usa una connessione a database per leggere il file database, è necessario specificare il percorso completo per usare gstat. (Questo potrà cambiare).

Assegnazione dei nomi ai database sotto Windows

Notate che ora l'estensione raccomandata per i file database sotto Windows ME e XP è ".fdb" per evitare possibili conflitti con le caratteristiche della funzione di "ripristino di sistema" di Windows. La mancata attenzione a questo punto su queste piattaforme farà nascere i noti problemi di ritardo alla prima connessione ad un database i cui file primari e/o secondari abbiano ancora nomi che seguono la convenzionale estensione".gdb".

Squadre di Sviluppo di Firebird

Sviluppatore	Paese	Principali compiti
Dmitry Yemanov	Russian Federation	Release coordinator; DSQL and PSQL enhancements; implementor of Embedded Server, numerous metadata enhancements, database aliasing, multi-action triggers, BigInt data type, new context variables, Windows Classic server; correzione numerosi errori
Nickolay Samofatov	Russian Federation	SQL feature designer/implementor (punti di salvataggio, blocco pessimistico); migliori metadati; major engine re-implementations; rilevamento e correzione errori; architectural troubleshooter; enabled Services API on Linux Classic; performance enhancements; Linux Classic builds
Arno Brinkman	The Netherlands	Optimizer enhancements; many new DSQL features
Claudio Valderrama	Chile	Code scrutineer; rilevamento e correzione errori; PSQL enhancements; UDF fixer, designer and implementor
Alex Peshkoff	Russian Federation	New PSQL and DSQL features; security features coordinator and author; correzione errori; Linux Superserver builds
Mike Nordell	Sweden	Translated Firebird codebase to C++; performance enhancements; feature porting; rilevamento e correzione errori
Blas Rodriguez Somoza	Spain	Developer of new character sets; major code cleaner and tree surgeon; MinGW builds
Roman Rokytskyy	Germany	Jaybird implementor and co-coordinator
David Jencks	U.S.A.	JayBird designer and co-coordinator; designer of Firebird documentation tools
Carlos Guzman Alvarez	Spain	Developer and coordinator of .NET provider for Firebird
John Bellardo	U.S.A.	Implemented plug-in interface for character sets; coordinator, Darwin builds; initial implementor of new memory model
Erik Kunze	Germany	Rilevamento e correzione errori; code-cleaner; SINIX-Z builds
Dmitry Sibiryakov	Russian Federation	Code cleanup; MinGW builds
Pavel Cisar	Czech Republic	Linux builds (Release 1.0); QA tools designer/coordinator
Ann Harrison	U.S.A.	Correzione errori; consigliere tecnico; aumento numero massimo indici consentiti
Mark O'Donohue	Australia	readline feature in isql; bug-fixing; boot builds (Release 1.0); code-fixing (Release 1.0)
Paul Reeves	France	QA; Win32 installers; standard Win32 control panel applet
Ignacio J. Ortega	Spain	Added PLAN feature to triggers; code cleanup

Developer	Country	Major tasks
Konstantin Kuznetsov	Russian Federation	Solaris Intel builds
Olivier Mascia	Belgium	Re-implementor of Win32 installation services
Peter Jacobi	Germany	Improvements, updating of character sets
Tilo Muetze	Germany	Firebird documentation project coordinator
Paul Vinkenoog	The Netherlands	Coordinator, Firebird documentation project; UDF fixes
Artur Anjos	Portugal	Enhanced Win32 control panel applets; Firebird Configuration Manager developer; internationalization of same
Achim Kalwa	Germany	Enhanced Win32 control panel applets
Sean Leyne	Canada	Bugtracker organizer; code cleaner
Ryan Baldwin	U.K.	Jaybird Type 2 driver developer
Sandor Szollosi	Hungary	Implementor of character set collations
Dmitry Kuzmenko	Russian Federation	Correzione errori GSTAT
Artem Petkevych	Ukraine	Correzione errori tipo ARRAY
Vlad Horsun	Ukraine	Speed-boosted sweep; fixed 2-phase commit bug
Tomas Skoda	Slovakia	Correzione errori
Evgeny Kilin	Russian Federation	Correzione errori
Oleg Loa	Russian Federation	Correzione errori
Erik S. La Bianca	U.S.A.	Correzione errori
Tony Caduto	U.S.A.	Unofficial Win32 installers
Juan Guerrero	Spain	New UDFs
Chris Knight	Australia	FreeBSD builds
Neil McCalden	U.K.	Solaris builds
Grzegorz Prokopsi	Hungary	Debian builds
Paul Beach	U.K.	HP-UX builds
Geoffrey Speicher	U.S.A.	FreeBSD builds
Helen Borrie	Australia	Release notes author; field-tester and Thought Police

"GLI EROI DEI TEST SUL 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

NOTE DI INSTALLAZIONE

Installare Firebird 1.5 sotto Windows 32

LEGGETE PRIMA QUESTO!

Con l'introduzione di due nuovi modelli di server sotto Win32 le scelte per l'installazione di Firebird sono proliferate.

- Assicuratevi di avere eseguito il login come Amministratore di sistema (ciò non si applica a Win9x o ME)

- Tutti i modelli—Superserver, Classic e Embedded Server così come i soli strumenti del server ed il solo client—possono essere installati usando l'applicazione Windows di installazione. Per un'installazione completa della release è fortemente raccomandato di usare l'installatore se ce n'è uno a disposizione.

- Usate **gbak** per effettuare il salvataggio del vostro vecchio database di sicurezza **isc4.gdb**. Potrete ripristinarlo in seguito come **security.fdb**

- Se avete speciali impostazioni in **ibconfig** possono esserci alcuni valori che volete trasferire agli equivalenti parametri in **firebird.conf**. Studiate le note su firebird.conf per determinare quel che può essere copiato direttamente e quali parametri richiede la nuova sintassi.

- Se determinati file di configurazione esistono nella directory di installazione saranno preservati se eseguite l'installatore e SOVRASCRITTI se decomprimete uno zip kit nella posizione predefinita. I file sono
  security.fdb
  firebird.log
  firebird.conf
  aliases.conf

- Ogni modello può essere installato da un file zip. Questo metodo è più veloce dell'installatore se avete un bel po' di esperienza nell'installazione di Firebird 1.5 da file zip. Sarà del tutto esasperante se siete un novellino con Firebird.

- Si assume che
  1 capiate come funziona la vostra rete.
  2 capiate perché un sistema client/server necessiti sia di un server che di client
  3 abbiate letto il resto di queste note alla release—o almeno realizziate che dovete leggerle se qualcosa va storto
  4 sappiate di cercare aiuto nella lista firebird-support se rimanete incagliati. Potete unirvi alla lista a http://www.yahoogroups.com/groups/firebird-support

Se già avete una versione precedente di Firebird o InterBase® sul vostro server e pensate di poter voler fare marcia indietro ad essa, predisponete le vostre misure di rientro prima di cominciare.

- Usate la versione esistente di GBAK per fare un backup dei vostri file database in formato trasportabile

- Andate nella vostra directory di sistema e salvate una copia di sicurezza di gds32.dll. Potreste rinominarla "gds32.dll.ib5" o "gds32.dll.fb103" o qualcosa di similmente informativo; o nasconderlo in un'altra directory

- Potrebbe essere una buona idea fare anche una copia del runtime Microsoft C++, msvcp60.dll. L'installatore non dovrebbe sovrascrivere la vostra versione di questo file, ma si sa che sono avvenute strane cose.

- **FERMATE OGNI SERVER FIREBIRD O INTERBASE CHE SIA IN ESECUZIONE** L'installatore tenterà di riconoscere se una versione esistente di Firebird o InterBase è installata e/o in esecuzione. Se non fate uso dell'installatore siete abbandonati a voi stessi!

- La posizione predefinita della radice di Firebird 1.5 sarà C:\Programmi\Firebird_1_5. Se la vostra vecchia versione è già lì e volete usare i valori predefiniti di 1.5, rinominate quella directory

- Per installare Firebird come servizio: se volete fare uso della nuova caratteristica di **login sicuro**, create un "utente di servizio firebird" sul sistema—qualsiasi nome e password vogliate—come un utente ordinario e con gli appropriati privilegi. Dovreste leggere il documento di nome README.instsvc.txt prima. Se avete un kit zip, lo troverete nella directory /doc della radice del file zip. Se non ce l'avete, il file sarà disponibile solo dopo l'installazione. Potete leggere lo stesso documento all'URL:

  http://cvs.sourceforge.net/viewcvs.py/firebird/firebird2/doc/README.instsvc

LEGGETE QUANTO SEGUE!

Uno degli scopi nella progettazione di Firebird 1.5 è di preparare la via per installazioni multiple del server. Ciò consentirà agli utenti di eseguire versioni differenti contemporaneamente. Firebird 1.5 supporta queste situazioni, sebbene non siano ben documentate e richiedano l'intervento di un utente smaliziato. Future versioni di Firebird renderanno il processo molto meno complicato. Nel frattempo Firebird 1.5 deve preparare il terreno. Questo ci costringe ad affrontare il problema dell'installazione di librerie. Allo stesso tempo, la Microsoft ha fatto i propri passi per gestire l'installazione di librerie di versioni differenti. Considerate insieme, queste due questioni implicano un nuovo approccio all'installazione delle librerie per Firebird 1.5 e successive versioni.

Installazione delle librerie di sistema Microsoft

Il problema di installare versioni diverse di librerie di sistema Microsoft è così noto da aver preso il nome di 'Inferno delle DLL'.

Dalla release di Windows 2000 in poi la Microsoft ha reso quasi impossibile aggiornare dll di sistema. Per risolvere questo, la Microsoft ora raccomanda che ogni applicazione installi copie locali delle librerie di sistema richieste.

Firebird 1.5 segue questa prassi e posiziona le librerie richieste nella directory \bin insieme al server.

Installazione di fbclient.dll

Firebird 1.5 e successivi non usano più gds32.dll come libreria client; è ora chiamata fbclient.dll. Dati i problemi della Microsoft con l'Inferno delle DLL non avrebbe avuto molto senso per noi continuare a salvare la libreria client di Firebird nella directory <system>. E poiché vogliamo consentire l'installazione di motori database multipli simultaneamente, ci saremmo creati il nostro Inferno delle DLL se avessimo continuato la pratica di usare la directory <system> per la libreria client. Così, da Firebird 1.5 in poi, la libreria client risiede nella directory \bin insieme agli altri file binari.

Una nuova chiave del registry è stata aggiunta e tutte le applicazioni compatibili Firebird devono ora usare questa chiave per localizzare la corretta versione di Firebird che vogliono usare. La nuova chiave è:

HKEY_LOCAL_MACHINE\SOFTWARE\Firebird Project\Firebird Server\Instances

Firebird garantirà che un valore sotto questa chiave sia comunque disponibile. Sarà nota come

"DefaultInstance"

e memorizzerà il percorso alla directory radice (sì, avete indovinato) dell'installazione predefinita. Chi non ha interesse a specifiche installazioni potrà sempre usare l'istanza predefinita per localizzare fbclient.dll.

Le versioni future di Firebird avranno altre voci sotto Instances. Le applicazioni potranno enumerare le voci del registry per determinare quale istanza di libreria vogliono caricare.

Supporto di applicazioni ereditate e driver

Tradizionalmente, le applicazioni che usano InterBase o Firebird si aspettavano di caricare la libreria client gds32.dll dalla directory <system>. Firebird 1.5 è distribuito con uno strumento chiamato 'instclient.exe' che può installare un clone di fbclient.dll nella directory di sistema di Windows. Questo clone viene modificato al volo in modo che le sue informazioni di versione comincino con "6.3", per fornire la compatibilità con le vecchie applicazioni che controllano le versioni di GDS32.DLL e non sanno cosa fare trovando una stringa come "1.5".

Durante il processo di installazione l'installatore controlla se un'installazione di InterBase o Firebird esiste. Se non c'è niente di installato scriverà gds32.dll nella directory <system>. Se scopre che una qualsiasi versione di Firebird o InterBase può essere già installata, non installerà gds32.dll nella directory <system>. Lo strumento 'instclient.exe' può essere usato in seguito per farlo.

Si intende che le versioni future di Firebird non tenteranno di installare gds32.dll nella directory <system> e che in definitiva sarà completamente rimosso dalla distribuzione.

Questo strumento 'instclient.exe' può anche installare la stessa FBCLIENT.DLL nella directory di sistema di Windows, se necessario. Questo accontenta le applicazioni che devono caricarlo da lì.

L'utilità instclient.exe dovrebbe trovarsi nella directory 'bin' della vostra installazione Firebird e deve essere eseguita da lì.

Uso di instclient.exe:

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

dove *library* è: `fbclient | gds32`

'-z' può essere usato con tutte le altre opzioni, stampa la versione.

Le informazioni di versione e il contatore delle librerie condivise sono gestiti automaticamente. Potete fornire l'opzione -f[orce] per forzare i controlli di versione.

NOTA Se forzate l'installazione con -f[orce], ciò potrebbe far malfunzionare un'altra versione di Firebird o InterBase® già installata. Potreste dover riavviare la macchina per finalizzare la copia.
Per ulteriori dettagli, leggete il documento *README.Win32LibraryInstallation.txt* posto nella directory radice del vostro percorso di installazione o in ..\doc.

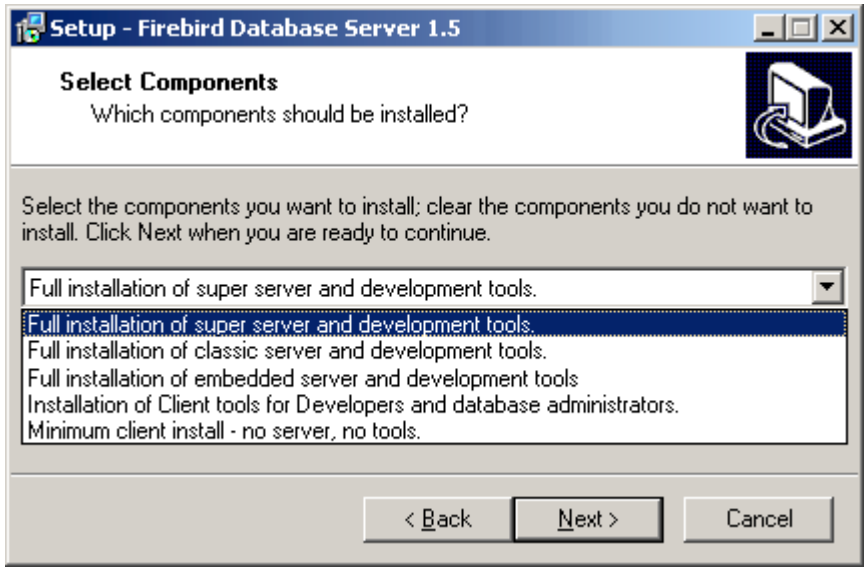
Pulizia di installazioni di versioni candidate al rilascio

Si noti che l'installatore rimuove fbclient.dll dalla directory <system> se il file è trovato in quella posizione. L'installatore rimuove anche la chiave del registry HKLM\Software\FirebirdSQL.

Uso del programma di installazione Win32 di Firebird

Questa è la parte facile.

Semplicemente lanciate l'eseguibile e rispondete ai dialog. Dopo aver risposto a circa quattro dialog, dovreste vederne uno con una lista a discesa che - una volta fatta scendere - ha questo aspetto. Questa è effettivamente l'ultima chance che avete di scegliere l'installazione desiderata.



Scegliete l'installazione desiderata e premete "Next" per continuare a rispondere ai dialog.

Servizio o applicazione?

Se scegliete di installare Superserver o Classic, e la vostra versione di S.O. supporta i servizi, vi sarà chiesto se eseguire Firebird come servizio o applicazione. A meno che non abbiate un vero *bisogno* di eseguire il server come applicazione, scegliete servizio.

Manuale o automatico?

Con l'opzione avvio automatico, Firebird si avvierà ogni qualvolta avviate la macchina ospite. Con l'opzione avvio manuale dovrete avviare il server quando occorre.

L'opzione Guardian

Guardian è un'utility che può essere eseguita "sopra" Superserver e lo riavvia se si blocca per qualsiasi ragione. Per lo sviluppo, potreste scegliere di non usarlo. Per la distribuzione, può evitarvi situazioni in cui il server si ferma e nessuno sa dove trovare l'amministratore di sistema per riavviarlo.

La directory radice (root) di installazione

Se decidete di non usare la directory predefinita come root, sfogliate fino ad una posizione che avete creato precedentemente oppure semplicemente digitate un percorso completo. Questo percorso non deve necessariamente esistere: l'installatore vi chiederà conferma e lo creerà se non esiste.

Alla fine i dialog finiranno ed il server si avvierà in silenzio oppure vi sarà richiesto il permesso di riavviare la macchina—il riavvio sarebbe necessario se l'installatore avesse sovrascritto il vostro file msvcp60.dll, oppure se un vecchio gds32.dll era già in memoria quando il programma di installazione è stato avviato.

Installazione del Superserver da uno zip kit

L'installazione di FB 1.5 è simile in principio alle precedenti versioni.
Se non avete uno speciale programma di installazione (è distribuito separatamente) i passi sono i seguenti:

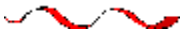
- Decomprimete l'archivio zip in una directory separata (poiché alcuni nomi di file sono stati cambiati, non ha senso decomprimere i file della v1.5 nella stessa directory di IB/FB1)
- cambiate la directory corrente in <root>\bin (<root> è la directory in e sotto cui si trovano i file di v1.5)
- eseguite instreg.exe:

  instreg.exe install

  ciò causa la scrittura del percorso di installazione della predetta directory nel registry (HKLM\Software\Firebird Project\Firebird Server\Instances\DefaultInstance)
- se volete anche registrare un servizio, eseguite pure instsvc.exe:

  instsvc.exe install

- come opzione, dovreste anche copiare sia fbclient.dll che gds32.dll nella directory del S.O.



Installazione del server Classic da un kit zip

Per installare il motore Classic Server, l'unica differenza è uno switch aggiuntivo di instsvc.exe:
    instsvc.exe install -classic

Si noti che ciò significa che potete avere una sola architettura del motore— fbserver.exe (Superserver) o fb_inet_server.exe (il processo padre di Classic)—installata come servizio.
L'applet del Pannello di controllo non viene installata deliberatamente nella versione Classic. Non tentate di installarla ed usarla: il concetto di terminazione di un servizio non si applica al modello Classic.

Setup semplificato

Se non vi serve un servizio registrato, potete evitare di eseguire entrambi instreg.exe e instsvc.exe. In questo caso dovreste solo decomprimere l'archivio zip in una directory separata ed eseguire il server:
    fbserver.exe -a
Questi dovrebbe in tal caso trattare la sua propria directory padre come root directory.

Disinstallazione

Per rimuovere FB 1.5 senza programma di disinstallazione Windows dovete:

- fermare il server
- eseguire "instreg.exe remove"
- eseguire "instsvc.exe remove"
- cancellare la directory di installazione
- cancellare fbclient.dll e gds32.dll dalla directory di sistema del S.O.

Installazione di embedded server da uno zip kit

Il server embedded è un client con un server del tutto funzionale linkato come dll (fbembed.dll). Ha esattamente le stesse caratteristiche del normale Superserver ed esporta gli entry point delle API Firebird standard.

Registry Le voci del registry per Firebird (dove normalmente il server cerca la posizione della root directory) sono ignorate. La root directory del server embedded è la directory sopra quella in cui il suo file eseguibile è posto.

Accesso ai database Solo l'"accesso locale" puro è consentito. Il server embedded non ha supporto per i protocolli remoti, così nemmeno gli accessi tramite "localhost" funzioneranno.

Autenticazione e sicurezza Il database di sicurezza (security.fdb) non è usato nel server embedded e quindi non è richiesto. Ogni utente può connettersi a qualsiasi database. Poiché sia il server che il client eseguono nello stesso spazio (locale) di indirizzi, la sicurezza diventa una questione di accessi fisici.
I privilegi SQL sono controllati, come in altri modelli di server.

Compatibilità Potete eseguire qualsiasi numero di applicazioni con l' embedded server senza alcun conflitto. Nemmeno l'esecuzione simultanea di server IB/FB è un problema.

Dovreste tuttavia sapere che non potete accedere allo stesso database da multipli server embedded simultaneamente, perché hanno architettura SuperServer e quindi bloccano in modo esclusivo i database connessi.

Struttura dei file di Embedded Server
Copiate fbembed.dll nella directory in cui la vostra applicazione risiede. Rinominatelo quindi in fbclient.dll o gds32.dll, a seconda del vostro software di connettività database. Fate copie con *entrambi i nomi* se vi serve usare gli strumenti server (isql, gbak, ecc.)

Dovreste copiare nella stessa directory anche firebird.msg, firebird.conf (se necessario) e ib_util.dll.

Se la vostra applicazione richiede librerie esterne, ad es. supporto INTL (fbintl.dll) o librerie UDF, dovrebbero trovarsi altrove rispetto alla directory dell'applicazione. Per poterle usare, piazzatele in un albero di directory che emuli quello del server Firebird, cioè in sotto-directory con nome /intl e /udf subito sotto la directory dove sono posti i file root di Firebird.

Esempio

```
D:\my_app\app.exe
D:\my_app\gds32.dll (rinominato fbembed.dll)
D:\my_app\fbclient.dll (rinominato fbembed.dll)
D:\my_app\firebird.conf
D:\my_app\aliases.conf
D:\my_app\isql.exe
D:\my_app\ib_util.dll
D:\my_app\gbak.exe
D:\my_app\firebird.msg
```

```
D:\my_app\intl\fbintl.dll
D:\my_app\udf\fbudf.dll
```

Avviate quindi la vostra applicazione. Userà il server embedded come libreria client e sarà in grado di accedere ai database locali.

NOTA Ammesso che impostiate la struttura delle directory in base a queste regole, non sarà necessario configurare esplicitamente RootDirectory in firebird.conf. Se invece decidete di fornire il server embedded e la vostra applicazione in una struttura di directory in qualche modo diversa, assicuratevi di leggere prima il documento README_embedded.txt della distribuzione del Server Embedded, per istruzioni su configurazioni aggiuntive.

Disinstallazione

La routine di disinstallazione di Firebird preserva e rinomina i seguenti file chiave:

preserva security.gdb o lo rinomina in security.fbnnnn
preserva firebird.log
preserva firebird.conf o lo rinomina in firebird.confnnnn
preserva aliases.conf o lo rinomina in aliases.confnnnn

"nnnn" è il numero di build della vecchia installazione.
Nessun tentativo viene fatto di disinstallare file che non erano parte dell'installazione originale.
File condivisi come fbclient.dll e gds32.dll saranno cancellati se il contatore di condivisione indicherà che nessun'altra applicazione li usa più.
Le chiavi del registry create saranno rimosse.

Altre note

Winsock2

Firebird richiede WinSock2. Tutte le piattaforme Win32 dovrebbero averlo, tranne Win95. Durante l'installazione viene eseguito un test sulla presenza della libreria Winsock2. Se non la si trova l'installazione fallisce. Per sapere come aggiornarvi visitate:
http://support.microsoft.com/default.aspx?scid=kb;EN-US;q177719

Windows ME e XP

In Windows ME e XP (Home e Professional) c'è una caratteristica detta System Restore (ripristino di sistema), che causa l'auto-aggiornamento (o memorizzazione di sicurezza?) di tutti i file del sistema con estensione ".gdb". L'effetto è rallentare l'accesso ai database InterBase/Firebird fino ad un virtuale blocco, mentre i file vengono salvati ogni volta che un operazione di I/O viene eseguita. (Sotto XP non c'è System Restore sui server .NET).

Un file nella directory Windows di ME, c:\windows\system\filelist.xml, contiene i "tipi di file protetti". ".gdb" è citato qui. Charlie Caro originariamente raccomandava di cancellare l'estensione GDB dalla sezione "includes" di questo file. Da allora, tuttavia, è stato dimostrato che WinME può ricostruire questa lista. In XP, non è possibile modificare per niente filelist.xml.

Su ME, si suggerisce una delle seguenti soluzioni definitive:

- usare FDB (Firebird DB) come estensione dei vostri file database primari
- spostare i database in C:\My Documents, che è ignorato da System Restore
- disabilitare del tutto System Restore (per le istruzioni consultare la documentazione di Windows).

Su Windows XP Home e Professional potete spostare i vostri database in una partizione separata e impostare System Restore per escludere quel volume.

Windows XP usa smart copy, così il lavoro inutile visto in Windows ME può costituire un problema minore in su XP, almeno per i file più piccoli. Per i più grossi (ad es. i file database di Firebird, accidenti!) non sembrano esserci risposte migliori finché vi sono file ".gdb" nel filesystem generale.

Questo lascia il database di sicurezza isc4.gdb, che è considerato scrivibile dal codice che dovrebbe solo validare il login di un utente, in modo che l'intestazione di isc4 sia aggiornata per quella transazione. Quindi WinME probabilmente farà una copia di backup ogniqualvolta un utente farà un login.
Stiamo cercando di ottenere un'accurata descrizione del problema ed una soluzione collaudata da pubblicare qui. Se potete aiutare con la descrizione del problema e/o con la soluzione, inviate un messaggio alla lista firebird-support o all'interfaccia del newsgroup firebird-devel a news://news.atkin.com

Il comportamento in chiusura di Windows XP è notoriamente una "area buia". C'è una pausa significativamente lunga al momento della chiusura del servizio server. In quel periodo vi sono indicazioni che Firebird è in esecuzione come applicazione.
Il problema sembra solo riguardare Windows XP e appare solo se Guardian non viene usato per fermare il servizio server. Questa è la soluzione temporanea per ora.

Installazione su UNIX / Linux

(Originariamente di Mark O'Donohue, rivisto per 1.5)

Il server Firebird ha due forme, Classic che viene eseguito come un servizio, e SuperServer, che viene eseguito come un demone sullo sfondo. Classic è il più tradizionale servizio UNIX, mentre Superserver usa i threads invece che processi. Per gli utenti alle prime armi con Firebird ambedue vanno bene, sebbene il server Classic sia probabilmente più adatto come piattaforma per gli esperimenti iniziali con Firebird.

NOTE – LEGGERE PRIMA QUESTO

1) Dovete essere il root user per installare Firebird.
2) L'installazione su Linux richiede un package glibc installato che sia uguale o successivo a glibc-2.2.5 e una libstdc++.so uguale o successiva a libstdc++-5.0.
3) Per una rozza valutazione delle distribuzioni Linux compatibili, riferitevi a questa tabella, ma non consideratela l'"ultima parola". Le distribuzione binarie di Linux come escono dalla scatola possono variare in funzione di dove e quando sono state create.
4) Assicuratevi che i package degli editor 'ed' e 'vim' siano installati sul vostro sistema. Se l'editor richiesto non è presente, il package si installerà ma gli script di installazione falliranno. (NOTA questa dipendenza potrà in futuro essere rimpiazzata da 'sed').

INSTALLAZIONI LINUX

Le seguenti istruzioni descrivono l'installazione di Classic. Per installazioni di Superserver il "CS" nel nome del package è sostituito da "SS". Per esempio, il package FirebirdCS-1.5.0-nnnn.i686.rpm è sostituito da FirebirdSS-1.5.0-nnnn.i686.rpm.

Per installazioni rpm linux

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

Per installazioni linux .tar.gz

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

* o `FirebirdSS-1.5.0-nnnn`

Cosa farà il programma di installazione per Linux
Le installazioni LINUX:

1. Tenteranno di fermare qualsiasi server in esecuzione
2. Aggiungerà l'utente 'firebird' ed il gruppo 'firebird' se non esistono già
3. Installeranno il software nella directory /opt/firebird e creeranno collegamenti alle librerie in /usr/lib ed ai file header in /usr/include
4. Automaticamente aggiungeranno gds_db per la porta 3050 a /etc/services se la voce non esiste già
5. Automaticamente aggiungeranno localhost.localdomain e HOSTNAME a /etc/host.equiv
6. SuperServer installa anche uno script di avvio del server in /etc/rc.d/init.d/firebird.
7. Il server Classic installa uno script di avvio /etc/xinetd.d/firebird oppure, per i sistemi inetd più vecchi, aggiunge una voce al file /etc/inetd
8. Specificamente per SuSE, un nuovo link rcfirebird è creato in /usr/bin per lo script init.d ed è creata una voce Firebird in /etc/rc.config.
9. Avvia il Server/servizio. Firebird dovrebbe avviarsi automaticamente in runlevel 2, 3 o 5
10. Genera e imposta una nuova password SYSDBA casuale e la memorizza nel file /opt/firebird/SYSDBA.password.
11. Aggiunge una voce ad aliases.conf per il database di esempio, employee.fdb.

L'installazione Classic imposta automaticamente la voce xinetd se la directory /etc/xinetd.d è presente, altrimenti imposta una voce inetd. Siccome molte distribuzioni posizionano xinetd diversamente da /etc/xinetd.d, il setup manuale può essere necessario in queste condizioni.

Verificate la vostra installazione Linux

Passo 1 – Accesso ad un database

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

SQL>connect localhost:employee.fdb /* percorso con alias */


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

SQL>quit;
```

* Una password è stata generata per voi durante l'installazione. Può essere ricavata dal file /opt/firebird/SYSDBA.password.

Passo 2 – Creazione di un database

Dalla versione 1.5 in poi, il server Firebird esegue in modo predefinito come utente 'firebird'. Sebbene questa sia sempre stata la configurazione raccomandata, i valori predefiniti precedenti prevedevano che il server eseguisse come utente 'root'. In tale modo il server aveva ampie facoltà di leggere, creare e cancellare file database ovunque su un filesystem POSIX. Per ragioni di sicurezza, il servizio dovrebbe avere più limitate capacità di lettura, cancellazione e creazione di file.

Sebbene la nuova configurazione sia migliore dal punto di vista della sicurezza, richiede alcune particolari considerazioni per la creazione di nuovi database:

a) l'utente 'firebird' deve avere permessi di scrittura nella directory in cui volete creare il database.
b) il valore raccomandato dell'attributo DatabaseAccess nel file /opt/firebird/firebird.conf dovrebbe essere impostato a None, per permettere l'accesso solo attraverso le voci del file aliases.conf.
c) usate voci in aliases.conf per astrarre gli utenti dalle locazioni fisiche dei database. Più informazioni sugli alias sono disponibili qui.

Le procedure per la creazione di un nuovo database variano se scegliete configurazioni diverse ma i seguenti sono i passi che raccomando con la configurazione suggerita:

1) Se una directory di proprietà dell'utente 'firebird' non esiste, cambiate utente e come utente root create la directory:

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

2) Create un nuovo database fisico e impostate una voce di alias che punti ad esso. Come root o utente firebird, eseguite questo script:

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

(L'uso è: createDBAlias.sh <dbname> <pathtodb>)

3) Come alternativa (al passo 2) i passi nello script createDBAlias.sh possono essere eseguiti a mano:

```
$vi /opt/firebird/aliases.conf
e aggiungete la riga alla fine del file:
test.fdb /var/firebird/test.fdb
```

4) Create quindi il database:

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

5) Se il valore di DatabaseAccess in /opt/firebird/firebird.conf è impostato a Full o ad un valore di percorso ristretto (ad esempio: DatabaseAccess=/var/firebird) un'altra alternativa al passo 2 è creare un file database fisico direttamente, usando il percorso assoluto con il nome di file:

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

Se usate questa configurazione, al file database si può accedere anche direttamente, senza voci nel file degli alias:

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

* Una password è stata generata per voi durante l'installazione. Può essere ricavata dal file /opt/firebird/SYSDBA.password.

Suggerimenti e script utili

In aggiunta ai file standard di installazione i seguenti script sono forniti nella directory bin di questa release:-

- **changeDBAPassword.sh**—Cambia la password dell'utente SYSDBA di Firebird. Per il SuperServer, questo script cambia anche lo script iniziale /etc/rc.d/init.d/firebird inserendovi la nuova password.

- **createAliasDB.sh**—Uso: createAliasDB.sh <dbname> <dbpath>.
  Questo script crea un nuovo database fisico e aggiunge una voce nel file aliases.conf.

- **fb_config**—Uno script che può essere usato nei makefile per generare i percorsi di include necessari e le direttive lib per la versione installata di Firebird. fb_config -help darà una lista completa di opzioni.

- **changeGdsLibraryCompatibleLink.sh**—Solo Classic—Cambia i link alla libreria client in libgds.so tra la versione multithreaded libfbclient.so e la versione a singolo thread libfbembed.so che consente l'apertura diretta da embedded del file db. Per compatibilità con installazioni precedenti, libgds.so punta inizialmente a libfbembed.so.

- **Accesso embedded o diretto ai file database**
  L'installazione di Classic offre un modo di accesso embedded che consente ai programmi di aprire i file database direttamente. Per operare in tal modo, un utente deve avere accesso privilegiato ad alcuni dei file di configurazione e di stato di Firebird.

- **Accesso ai database:** Ora che è 'firebird' (e non root) l'utente predefinito che esegue il software, dovete sapere come **inserire un utente nel gruppo firebird** per consentire l'accesso ai database. É documentato nelle note *readme*, ma i seguenti passi dovrebbero mettervi in grado di farlo: Per aggiungere un utente (ad es. skywalker) al gruppo firebird, l'utente root deve eseguire:

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

  La volta successiva che 'skywalker' fa il login, può cominciare a lavorare con i database firebird.

  Per elencare i gruppi cui un utente appartiene scrivete quanto segue sulla riga di comando:

  ```
  $ groups
  ```

- **Problemi NTPL su versioni recenti di Linux:**

La nuova NPTL (Native POSIX Thread Library) in Red Hat 9 (finora) causerà problemi in SuperServer ed in programmi compilati localmente, incluse le utilità. Gbak, in particolare, causerà un errore Broken Pipe. Per risolverlo:

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

Questo si occupa dell'istanza del server. Dovete impostare la variabile di ambiente nell'ambiente locale, così:

2. Aggiungete quanto di seguito a /etc/profile, per assicurarvi che ogni utente lo recepisca per le utilità a linea di comando

dopo

```
  HISTSIZE=1000
```
aggiungete
```
  LD_ASSUME_KERNEL=2.25
```
e sulla riga seguente esportatela:
```
 export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUT_RC LD_ASSUME_KERNEL
```

Disinstallazione su Linux

Se dovete eseguire la disinstallazione, fatelo come utente root. Il seguente esempio usa CS o ClassicServer ma lo stesso vale per SuperServer sostituendo il CS con SS.

Per i package rpm:

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

o per l'installazione .tar.gz:

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

Installazione di Firebird Classic & SuperServer su Solaris 2.7 Sparc

Non disponibile correntemente. Riferitevi alle note della release v.1 come riferimento per l'installazione di 1.5.

Installazione di Firebird Classic su MacOS X / Darwin

Non disponibile correntemente. Riferitevi alle note della release v.1 come riferimento per l'installazione di 1.5.

Creazione o installazione di Firebird su FreeBSD

Non disponibile correntemente. Riferitevi alle note della release v.1 come riferimento per l'installazione di 1.5.

Configurazione del servizio di porta sul client e sul server

Un server Firebird attende per predefinizione le richieste di connessione dei client sulla porta 3050. Il nome registrato del servizio di porta è **gds_db**. Le buone nuove sono che non dovete fare nulla per configurare questo servizio sul server o sui client se i valori predefiniti sono per voi accettabili.

Potete usare una diversa porta, un diverso nome del servizio, o entrambi. Potrebbe essere necessario se la porta 3050 fosse già usata da un altro servizio, per esempio un gds_db concorrente configurato per una diversa versione di Firebird o per un server InterBase®. Ci sono vari modi di cambiare i valori predefiniti. Sia il server che i client devono essere configurati per cambiare il nome del servizio di porta o il numero della porta, o entrambi, in almeno uno di questi modi:

- nella stringa di connessione del client
- nel comando usato per avviare l'eseguibile del server
- attivando i parametri RemoteServicePort o RemoteServiceName in firebird.conf (da V.1.5 in poi)
- nella configurazione del daemon (per Classic su POSIX)
- con una voce nel file Services.

Prima di esaminare ognuna di queste tecniche sarà utile considerare la logica usata dal server per impostare la porta d'ascolto e dal client per impostare la porta da usare per le richieste.

Come il server imposta la porta d'ascolto

L'eseguibile del server ha uno switch opzionale della riga di comando (-p) con cui può indicarsi o **il numero di porta su cui ascolterà** o il **nome del servizio di porta** che sarà ascoltato. A questo punto, se lo switch è presente, o il numero di porta 3050 o il nome del servizio di porta (gds_db) è sostituito dall'argomento fornito con lo switch–p.

In seguito — o precedentemente se lo switch –p non è specificato — un server v.1.5 controlla firebird.conf per trovare i parametri RemoteServiceName e RemoteServicePort:

- Se entrambi sono commentati con "#" allora si assumono i valori predefiniti e non avvengono ulteriori cambiamenti. Resta valido l'argomento di -p e un argomento "mancante" resta al valore predefinito.

- Se RemoteServiceName è stato reso attivo (non è commentato), ma non altrettanto RemoteServicePort, allora il nome del servizio di porta è sostituito *solo* se non è già superato dall'uso dello switch -p.

- Se RemoteServicePort è stato reso attivo (non è commentato), ma non altrettanto RemoteServiceName, allora il numero della porta è sostituito *solo* se non è già superato dall'uso dello switch –p.

- Se sia RemoteServicePort che RemoteServiceName sono stati resi attivi (non sono commentati) allora RemoteServiceName ha la precedenza, se non è già superato dall'argomento di –p. Se c'è già un valore prevalente del nome del servizio di porta, il valore RemoteServiceName è ignorato ed il valore di RemoteServicePort ha la precedenza sul valore predefinito 3050.

- A questo punto, se è stato segnalato un valore con precedenza del numero di porta o del nome del servizio, sia i server v.1.0 che i v.1.5 procedono al controllo del file Services per vedere se vi è una voce con la corretta combinazione dei valori del nome del servizio e del numero della porta. Se si trova un accoppiamento tutto va bene. Se così non è, ed il nome del servizio di porta non è gds_db, il server causerà un'eccezione e si rifiuterà di avviarsi. Se gds_db è il nome del servizio di porta e non potrà essere risolto ad una porta diversa, sarà mappato automaticamente sulla porta 3050.

Se si deve modificare il nome predefinito del servizio, occorrerà anche scrivere una voce nel file Services — vedere l'argomento seguente, *Configurazione del file Services*.

Uso dello switch –p.

Si noti che questo switch era disponibile anche in Firebird 1.0.x, ma era precedentemente non documentato.

L'avvio del server con lo switch opzionale–p vi consente di modificare sia il numero predefinito della porta (3050) che il nome predefinito del servizio di porta (gds_db) che il server usa per ascoltare le richieste di connessione. Lo switch può modificare uno di questi valori, ma non entrambi. Da Firebird 1.5 in poi, potete usare lo switch–p switch in combinazione con il file di configurazione firebird.conf per modificare sia il numero della porta che il nome del servizio di porta.

Sintassi per TCP/IP

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

Per esempio, per avviare il Superserver come applicazione modificando il nome del servizio da gds_db a fb_db:

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

Oppure, per cambiare la porta 3050 in 3051:

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

Sintassi per la redirezione WNet

Su una rete WNet, sostituite la sintassi dell'argomento sopraesposto dello switch –p con la seguente sintassi "barra contraria-barra contraria-punto-chiocciola":

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

oppure

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

Classic su POSIX: il daemon *inetd* o *xinetd*

Con Firebird Classic server su Linux o UNIX, il daemon **inetd** o **xinetd** è configurato per ascoltare sulla porta predefinita e trasmette il nome di servizio predefinito. Lo script di installazione scriverà la voce appropriata nel file di configurazione /etc/inetd.conf o /etc/xinetd.conf.

Potete modificare il file di configurazione corrente se necessario. Il seguente è un esempio di quel che dovreste vedere in xinetd.conf o in inetd.conf dopo aver installato Firebird Classic su Linux:

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

Se avete configurato il servizio di porta diversamente dal valore predefinito dovrete alterare xinetd.conf o inetd.conf di conseguenza. Riavviate xinetd (o inetd) con kill -HUP per assicurarvi che il daemon usi la nuova configurazione.

NOTA Le richieste di connessione falliranno se sia xinetd (o inetd) che fbserver (o ibserver) tenteranno di ascoltare sulla stessa porta. Se la vostra macchina host ha questa doppia configurazione, sarà necessario impostare le cose in modo che ogni versione del server abbia il suo proprio servizio di porta.

Uso di un parametro del file di configurazione

Potete configurare RemoteServiceName o RemoteServicePort in firebird.conf per modificare o il numero predefinito della porta (3050) o il nome predefinito del servizio di porta (gds_db) il server usa per ascoltare le richieste di connessione.

Il motore userà un parametro RemoteService*, ma non entrambi. Se li configurate entrambi ignorerà RemoteServicePort in tutti i casi, tranne quando il comando di avvio del server sia stato invocato con lo switch –p fornendo un valore sostitutivo al nome del servizio di porta. Così potete usare lo switch –p ed un parametro RemoteService* in combinazione, per sostituire sia il numero di porta che il nome del servizio di porta.

Se il nome predefinito del servizio di porta va sostituito dovete inserire una voce nel file Services.

ATTENZIONE! Se rimuovete il commento da RemoteServiceName o RemoteServicePort, ma lasciate i valori predefiniti intatti, questi saranno trattati come variazioni. Sarà quindi necessario inserire una voce esplicita nel file services per le impostazioni predefinite del servizio di porta.

Impostazione di un client per trovare la porta del servizio

Se impostate il server con i valori predefiniti di installazione (servizio gds_db in ascolto sulla porta 3050) non serve alcuna configurazione. Se il server ascolta su un'altra porta o usa un diverso nome del servizio di porta, l'applicazione client e/o la sua macchina ospite richiede una configurazione per aiutare la libreria Firebird del client a trovare la porta di ascolto.
La stringa di connessione usata dal client può includere informazioni per interrogare la porta di ascolto del server in vari modi. I client Firebird 1.5 possono opzionalmente usare una copia locale di firebird.conf. Possono anche servire modifiche nel file services del client.

Uso della stringa di connessione

Se è stato riconfigurato solo il numero della porta o il nome del servizio, basta includere il numero di porta o il nome di sevizio alternativo nella stringa di connessione. Questo funziona per ogni versione di Firebird.

Sintassi per le connessioni TCP/IP

Per connettersi ad un database server chiamato alice che trasmette sulla porta 3050 con nome di servizio fb_db, la stringa di connessione sarebbe:

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

Oppure, se il nome del servizio è gds_db ed il numero di porta è 3051:

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

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

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

Si noti che il separatore tra il nome del server e la porta è una barra e non un due punti. Il carattere due punti prima della stringa con il percorso fisico è ancora richiesto.

Sintassi per le connessioni WNet

Su una rete WNet, usate una notazione in stile UNC:

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

o

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

Se il numero di porta configurato del server o il nome del servizio sono variazioni dei valori predefiniti, dovete inserire una voce nel file services.

Uso della sintassi di porta con alias di database

Per connettersi tramite una porta diversa dalla predefinita con un alias di database, aggiungete come suffisso il numero di porta o il nome di servizio al nome del server, non all'alias. Per esempio, supponete che l'alias del database sia memorizzato in aliases.conf come

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

La stringa di connessione della vostra applicazione per il server 'alice' sarebbe:

```
alice/fb_db:rabbit
```

o

```
alice/3051:rabbit
```

Uso di una copia di firebird.conf

Da Firebird 1.5 in poi potete opzionalmente includere una copia dal lato client di firebird.conf nella directory radice di Firebird e configurare RemoteServiceName o RemoteServicePort per aiutare il client a trovare la porta del server.

- Potete configurare *uno* di questi due parametri per estendere la variazione fornita per l'altro attraverso la stringa di connessione (sopra); o per modificare solo RemoteServiceName o RemoteServicePort senza usare la stringa di connessione per farlo.

- Se volete evitare di passare il nome del servizio di porta o il numero della porta nella stringa di connessione ed il server usa valori diversi dai predefiniti per entrambi, potete configurare sia RemoteServiceName che RemoteServicePort. Usate questa tecnica se la vostra applicazione client richiede di conservare la capacità di connettersi a server InterBase o Firebird 1.0.

Posizione di componenti di Firebird sui client

Quando sfruttate firebird.conf sulle macchine client, è importante che la libreria client sappia dove trovarlo. Dovrete impostare una directory radice Firebird e informare il sistema della sua posizione. Usate la variabile di ambiente FIREBIRD per fare questo. I client Windows possono usare alternativamente la chiave del Registry Firebird. Con un'impostazione corretta del client, potete usare anche una versione locale del file dei messaggi.

Configurazione del file services

Non dovete configurare una voce per la porta di servizio del vostro Firebird server o dei client se il server usa i valori predefiniti di installazione, gds_db sulla porta 3050. Se gds_db è il nome del servizio di porta e non può essere risolto ad una diversa porta, sarà mappato sulla porta 3050 automaticamente.

Se state configurando la porta di servizio per una diversa porta o nome di servizio, sia il **server** che il **client** devono essere esplicitamente aggiornati per riflettere la riconfigurazione. Sia su Linux che su Windows, questa informazione è memorizzata nel file **services**.

Posizione del file services

- su Windows NT/2000/XP/S2003, questo file è C:\windows\system32\drivers\etc\services.

- su Windows 95/98/ME, questo file è C:\windows\services.

- su Linux/UNIX questo file è /etc/services.

Una voce del file services ha questo aspetto:

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

Aprite il file usando un editor di testo e aggiungete una riga o modificate l'esistente voce gds_db come segue:

- per un server o client Firebird 1.0.x, alterate o il nome del servizio o il numero di porta per rispecchiare il modo in cui il vostro server si avvierà.

- Per un server Firebird 1.5 o superiore, modificate o aggiungete una riga, come necessario. Se avete un server Firebird 1.0 o InterBase installato sullo stesso ospite, mantenete le voci che questi richiedono e aggiungete una nuova voce per rispecchiare il modo in cui il vostro server Firebird 1.5 si avvierà.

Ulteriori informazioni

Più informazioni sul motore database Firebird si possono trovare su:

http://firebird.sourceforge.net

o sui siti affiliati:

http://firebirdsql.org
http://www.ibphoenix.com
http://www.cvalde.net

Se siete interessati ad essere coinvolti nello sviluppo di Firebird, o vorreste avvisare di un possibile problema per discuterne, sentitevi liberi di aggiungervi alla nostra lista firebird-devel. Per sottoscriverla, mandate semplicemente un messaggio email vuoto a:

firebird-devel-request@lists.sourceforge.net

con la parola 'subscribe' nel campo Oggetto.

Si prega di non usare la lista firebird-devel per domande di assistenza.

Per il supporto tecnico, unitevi alla lista del gruppo firebird-support andando in
http://www.yahoogroups.com/groups/firebird-support

Per lo sviluppo e supporto di InterClient e Java, c'è una lista specializzata:
http://www.yahoogroups.com/groups/Firebird-Java

La lista firebird-support gestisce i problemi tecnici con Firebird e InterBase(R). Vi preghiamo di condurre le vostre domande su Delphi e altri ambienti di sviluppo client al forum appropriato.

La comunità open source gestisce diverse altre liste di discussione su vari aspetti dello sviluppo di Firebird. Per i dettagli riferitevi alle liste di posta ed ai newsgroup del sito della comunità Firebird.

La lista degli sviluppatori di Firebird e le liste generiche della comunità, insieme ad altre liste di interesse per sviluppatori Firebird e InterBase, hanno un mirror come newsgroup a
news://news.atkin.com

Assistenza a pagamento in tutto il mondo si può ottenere tramite IBPhoenix (indirizzi e numeri di contatto a http://www.ibphoenix.com). Diversi membri della squadra Firebird sono anche disponibili per assistenza e consulenza. Contattateli direttamente.

Servizi di recupero di database per database Firebird o InterBase possono essere gestiti da IBPhoenix. Per analizzare e tentare il recupero da sé di un database rovinato, tentate con IBSurgeon a IBase.ru (www.ibase.ru)
Anche IBase.ru fornisce servizi di recupero in Russia ed Europa.

Richieste/offerte di miglioramenti sponsorizzati a Firebird possono essere inviate direttamente alla FirebirdSQL Foundation Inc. Inviate un'email a foundation@firebirdsql.org. Potreste preferire inizialmente di contattare gli amministratori del progetto Firebird – inviate un'email a firebirds@users.sourceforge.net.

Discussioni generiche sui miglioramenti a FB possono essere gestite nella lista delle priorità di Firebird http://www.yahoogroups.com/community/Firebird-priorities.

IB-Architect (http://www.yahoogroups.com/community/ib-architect)è SOLO per **discussioni tecniche di progetto**. Le domande di assistenza/conversione sono del tutto fuori luogo qui.

Strumenti e driver

Programmi client di amministrazione dei database

Parecchie eccellenti scelte di programmi di amministrazione GUI per Firebird sono elencati nella pagine Contributed Downloads a http://www.ibphoenix.com. Alcuni sono open source, alcuni sono freeware, altri prodotti commerciali consolidati.

Il programma IBConsole della Borland non è raccomandato come client di amministrazione dei database per Firebird 1.5.

Driver e components

JAVA: Il progetto Jaybird JDBC driver sviluppa attivamente come parte del progetto Firebird. Ha rilasciato un driver Type 4 JDBC/JCA ed un driver Type 2 in beta. Sorgenti e binari sono scaricabili dalle pagine delle release Firebird a -

http://sourceforge.net/project/showfiles.php?group_id=9028

Per consigli e per partecipare nello sviluppo e test, unitevi al forum Firebird-java a http://www.yahoogroups.com/community/firebird-java.

.NET: Firebird ha un progetto di driver .NET attivo. Sorgenti ed eseguibili si possono scaricare dalle pagine delle release Firebird a -

http://sourceforge.net/project/showfiles.php?group_id=9028

Per consigli e per partecipare nello sviluppo e test, unitevi al forum sul Firebird .NET provider: http://lists.sourceforge.net/lists/listinfo/firebird-net-provider

Delphi e C++Builder: Gli utenti hanno due potenti alternative per la connettività piena e diretta con le API di Firebird 1.5, entrambe ben supportate:

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

C++: la libreria di accesso freeware C++ 'IBPP' (http://www.ibpp.org), licenza MPL, ospitata su sourceforge.net, del tutto portabile tra Win32 e Linux e probabilmente altre piattaforme POSIX. É utile quando volete un tipo di accesso a basso livello C alle API, ma con un leggero livello di astrazione C++ non legato ad un particolare ambiente di sviluppo.

ODBC: Una lista di driver ODBC può essere reperita nella pagina Contributed Downloads a http://www.ibphoenix.com.

PHP: Un gruppo sta lavorando per portare la vecchia estensione PHP per InterBase allo standard di Firebird. Per saperne di più, unitevi al forum Firebird-PHP a http://www.yahoogroups.com/community/firebird-php

PYTHON: KInterbasDB è un package di estensione Python che implementa le Python Database API 2.0-di supporto compatibile con Firebird. Supporto API nativo pieno per il client Firebird; in condizione di release stabile e sotto sviluppo attivo. Licenza open source BSD. Downloads e notizie a http://kinterbasdb.sourceforge.net/

Documentazione

La documentazione per InterBase v 6.0 si applica anche alla release corrente di FireBird. Una versione beta dei manuali di InterBase(tm) 6 è disponibile in formato Adobe Acrobat da

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

Un indice strutturato della documentazione è mantenuto sul sito della comunità Firebird a

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

Questo è un lavoro in progresso ed ogni aggiunta è benvenuta – inviate un messaggio a

firebird-docs@lists.sourceforge.net

Alcune guide di installazione e altri documenti esplicativi si possono trovare nell'area documentazione, che può essere raggiunta da

http://www.firebirdsql.org

o più direttamente da

http://sourceforge.net/projects/firebird

L'archivio principale per le questioni utente e tecniche è il sito IBPhoenix –

http://www.ibphoenix.com

IB Phoenix pubblica anche regolarmente un CD in abbonamento (sono disponibili anche singoli numeri), contenente i manuali da loro pubblicati – *Using Firebird* e *The Firebird Reference Guide*.

Si può scoprire documentazione addizionale visitando sul sito Borland l'area delle pubblicazioni tecniche:

http://www.borland.com/techpubs/interbase/

Correzioni e aggiunte dalla Release 1.0

Tracker #	Descrizione	Autore del contributo
(no #)	Fixed minor inconsistencies in charsets naming.	P. Jacobi
(no #)	GSTAT crashed in some switch combinations.	D. Yemanov
(no #)	Fixed broken savepoint handling in BREAK|LEAVE/EXIT.	D. Yemanov
(no #)	Fixed optimizer to prefer single indices to composite ones and to prefer full-match unique indices.	A. Brinkman
(no #)	Enhanced Win32 install tools instsvc.exe and instreg.exe	O. Mascia
Improvement	Maximum number of indices per table increased from 64 to (DB_PAGE_SIZE/16)-2	A. Harrison, ported to 1.5 by N.Samofatov
721792	Long runnning connection caused mem leak in OS kernel device	N. Samofatov
775003	UDF log(x, y) in fact returned log(y, x)	P. Vinkenoog, N. Samofatov
774987	UDFs ltrim(") and rtrim(") returned NULL; rtrim forgot 1st char	P. Vinkenoog, N. Samofatov
(no #)	Fixed server crash caused by lost transaction context.	A. Peshkoff
(no #)	Fixed server crash for any combination of sub-select & between.	A. Peshkoff
736318	"<value> STARTING WITH <field>" fails when using indices.	D. Yemanov
(no #)	Non-existent deadlock is raised after execution of pre-(update/delete) triggers.	A. Peshkoff
Improvement	Make INSERTING/UPDATING/DELETING keywords non-reserved.	N. Samofatov
Improvement	Added new (more specific) error messages for some of v1.5 changes.	D. Yemanov, A. Brinkman, A. Peshkoff
Security fix	Added -login switch to instsvc allowing to install FB service as non-localsystem account.	A. Peshkoff
Improvement	Re-introduced trimming of VARCHAR fields in the remote protocols.	D. Yemanov
(no #)	Random server crash in the case of big queries being prepared.	D. Yemanov
Improvement	Configuration improvement – make path management in firebird.conf	A. Peshkoff

	conform to the OS requirements.	
Tracker #	**Description**	**Contributor**
(no #)	Wrong UDF arguments of types DATE/TIME (dialect 3).	Oleg Loa
(no #)	Possible referential integrity violation.	Vlad Horsun, D. Yemanov
745090 and other RC 2 installation issues	Permissions problem for firebird.conf (SF #745090). Also generate aliases.conf on install; use rpmbuild to create Linux packages	Erik S. LaBianca, N. Samofatov
(no #)	Allow easy adjustment of LockSemCount on POSIX platforms. No need to use gds_drop or reboot machine to make new setting take effect	N. Samofatov
Improvement	Make FIRST/SKIP keywords non-reserved.	N. Samofatov
(no #)	Wrong attachment reference after exception in PSQL.	A. Peshkoff
(no #)	BREAK/LEAVE and EXIT statements are now available for usage in triggers.	D. Yemanov
(no #)	Possible index corruptions during garbage collection.	Vlad Horsun, D. Yemanov
(no #)	Solved problems with temporary files management: 1) Security hole on all POSIX platforms except FREEBSD/OPENBSD related to mktemp usage (possible DoS attacks or privileges elevation) Only 27 unique filenames generated on win32 (which could cause unpredictable behavior in SS builds)	N. Samofatov
(no #)	Event manager change: disabled usage of definite aux port in CS builds due to known issues.	D. Yemanov
(no #)	Enabled aggregate functions from different parent context to be used inside another aggregate function. Example: SELECT MAX((SELECT COUNT(*) FROM RDB$RELATIONS)) FROM RDB$RELATIONS	A. Brinkman
(no #)	Possible crashes on disconnect when event notification is used.	D. Yemanov
(no #)	Service manager changes: features of GSTAT/GSEC are not available via Services API in win32 CS (until v1.6 release).	D. Yemanov
(no #)	Wrong record statistics are reported when operation fails for some reason.	D. Yemanov
(no #)	stdin/stdout cannot be used to redirect console I/O in win32 build of	A. Peshkoff

	GBAK.	
Tracker #	**Description**	**Contributor**
(no #)	Broken lock table resizing in CS. No more "lock manager out of room" (Win32 CS 1.5 RC1) or crashes (possible in all other CS builds of Interbase/Firebird).	N. Samofatov
Improvement	INTL improvement: make UPPER function work for WIN1251 charset without explicit collations.	N. Samofatov, D. Yemanov
BUGCHECK(291)	Possible database corruption when you modify/delete the same record in pre-trigger for which this trigger was called.	A. Peshkoff
(no #)	Buffer overrun in isc_database_info() call.	Oleg Loa
(no #)	Configuration manager change: now the server exits on missing / wrong firebird.conf with error report in system log.	A. Peshkoff
(no #)	Fixed Services API: enabled statistics Services API for POSIX CS builds.	N. Samofatov
(no #)	Changed parser. 1) ROWS_AFFECTED is renamed to ROW_COUNT 2) CONNECTION_ID/TRANSACTION_ID are renamed to CURRENT_CONNECTION/CURRENT_TRANSACTION 3) Some of newly introduced tokens are made non-reserved	D. Yemanov
(no #)	Fixed Services API: partly enabled Services API for win32 CS builds.	D. Yemanov
(no #)	Wrong type of event delivery (unnecessary usage of OOB packets).	Jim Starkey, Paul Reeves
(no #)	Improved lock manager: deadlocks are now detected and reported as soon all blocking processes received notifications, i.e. instantly in all normal cases	N. Samofatov
(no #)	Server crashes in some Services API operations.	A. Brinkman
(no #)	Advanced security capabilities: implemented configurable access for databases, external tables and UDF libraries.	A. Peshkoff
(no #)	Fixed resource/memory leaks.	Mike Nordell, A. Peshkoff, N. Samofatov, D. Yemanov
(no #)	Buffer overrun with multidimensional arrays.	D. Yemanov
213460, 678718	Various issues with events used on multihomed hosts. NOTE Now it's also possible to setup a definite port for event processing.	D. Yemanov

Tracker #	Description	Contributor
(no #)	Fixed some resource leaks.	Mike Nordell, A. Peshkoff
(no #)	Fixed Services API: enabled Services API for posix CS builds. Notes: 1. Appropriate changes in Win32 CS are not ready yet 2. Backup/restore service was fixed, tested and should work 3. Database validation was partially fixed and may work 4. Other services are probably non-functional in CS builds yet	N. Samofatov
(no #)	SQL enhancement: allow NULLs in unique constraints and indices (SQL-99 spec).	D. Yemanov, N. Samofatov
(no #)	Performance improvement: VIO undo log now uses B+ tree to store savepoint record data. It improves performance when doing multiple updates of record in a single transaction just a little (usually 2-3 orders of magnitude for 100000 records).	N. Samofatov
(no #)	Database corruption when backing out the savepoint after large number of DML operations so transaction-level savepoint is dropped) and record was updated _not_ under the savepoint and deleted under savepoint.	N. Samofatov
(no #)	Improved EXECUTE STATEMENT. Now it's possible to return values from the dynamic SQL. Syntax: EXECUTE STATEMENT <value> INTO <var_list>; (singleton form) or FOR EXECUTE STATEMENT <value> INTO <var_list> DO <stmt_list>;	A. Peshkoff
(no #)	Server hangs during disconnect after mass updates.	D. Yemanov
(no #)	Improved optimizer: Subselects in SET clause of UPDATE now can use indices.	A. Brinkman
(no #)	"Context already in use" error in the case of DISTINCT with subqueries.	A. Brinkman
(no #)	Enhanced isc_database_info capability: list of currently active transactions is now available via isc_database_info call.	N. Samofatov
(no #)	Performance improvement: shortcut boolean evaluation. NOTE behaviour is controlled by "CompleteBooleanEvaluation" option of firebird.conf. Default is 0 (shortcut evaluation).	Mike Nordell
(Beta 2 bug)	Stack overflow during statement preparation.	D. Yemanov, Mike Nordell

Tracker #	Description	Contributor
(no #)	Performance improvement for IA32 CPU architecture: speed-up for index operations	Mike Nordell
(no #)	Change in universal triggers: allowed access to both (OLD and NEW) contexts in universal triggers.	D. Yemanov
(no #)	Improved optimizer: when an equal-node and other nodes (geq, leq, between...) are available for an index retrieval, then use the equal node always instead of the others.	A. Brinkman
(no #)	Long delays during connecting/disconnecting on WinXP.	A. Brinkman
(no #)	Generic cleanup: removed a lot of unused code.	Blas Rodriguez Somoza, Erik Kunze
523589	View is affecting the result of a query. Comment: Problem was that RSE's (inside a view) were not flagged as variant.	A. Brinkman
(no #)	Changed behaviour of the forced writes mode: now, if FW=off (disabled), you can control how often dirty pages are flushed on disk (allows better reliability when FW is disabled on Win32 platforms).	Blas Rodriguez Somoza
(no #)	The security database has been renamed to security.fdb.	D. Yemanov
(no #)	New configuration file: firebird.conf is finally published.	D. Yemanov
(no #)	New user-defined functions LPAD and RPAD added to IB_UDF library.	Juan Guerrero
(no #)	Sometimes GFIX didn't allow to specify "-user" and "-password" switches ("incompatible swiches" error).	D. Yemanov
(no #)	Security connection cache: connection to the security database is now cached, thus allowing to decrease time of subsequent database attachments.	D. Yemanov
Improvements	1. Reduce memory usage by the server. 2. Direct external I/O when the memory is not available for the sorting. Increased number of streams and predicates supported by the optimizer.	D. Yemanov
508594	LEFT JOIN with VIEWs: simple LEFT JOIN on a VIEW with only an ON clause didn't use an index even if it was possible.	A. Brinkman
(no #)	New character sets: DOS737, DOS775, DOS858, DOS862, DOS864, DOS866, DOS869, WIN1255, WIN1256, WIN1257, ISO8859_3, ISO8859_4, ISO8859_5, ISO8859_6, ISO8859_7, ISO8859_8, ISO8859_9, ISO8859_13 NOTE Collations for the above charsets are not available yet.	Blas Rodriguez Somoza

Tracker #	Description	Contributor
(no #)	CREATE VIEW changes: disallowed PLAN subclause.	D. Yemanov
(no #)	Changed aggregate tracking behavior -- introduced backwards compatibility within aggregates. Deepest field inside aggregate determines where an aggregate-context should belong.	A. Brinkman
(no #)	Improved optimizer: better optimizations of "complex" JOIN queries (LEFT JOIN, views, SPs, etc).	A. Brinkman
(alpha 5 bug)	Major memory leaks are fixed.	D. Yemanov
(no #)	New API functions: IB7-compliant functions to return version of the client library -- isc_get_client_version(), isc_get_client_major_version(), isc_get_client_minor_version()	D. Yemanov
(no #)	Sort/merge improvement: merging (SORT MERGE plans) is now done via in-memory sorting module.	D. Yemanov
(no #)	New memory manager's internal has been changed to give us better performance.	N. Samofatov
(no #)	Win32 build changes: 1. Changed names of USER32 objects to allow the server to run simultaneously with IB/FB1. 2. Map name for local (IPC) protocol is changed, so v1.5 client library is no longer compatible with the previous versions via IPC. 3. All transport protocol names (INET port and service, WNET pipe, IPC map) are now configurable via firebird.conf.	D. Yemanov
(no #)	Trashed RDB$FIELD_LENGTH for views that contain concatenation of long CHAR/VARCHAR fields.	D. Yemanov

Improvement	Triggers improvement: added runtime action checks (INSERTING/UPDATING/DELETING predicates).
	Example:
	if (INSERTING) then
	new.OPER_TYPE = 'I';
	else
	new.OPER_TYPE = 'U';
		D. Yemanov

(no #)	Cursors (WHERE CURRENT OF clause) could not be used in triggers.	D. Yemanov
221921	ORDER BY has no effect.	A. Brinkman

Tracker #	Description	Contributor
213859	Subquery connected with 'IN' clause.	A. Brinkman
Improvement	Allowed arbitrary expressions in the ORDER BY clause.	N. Samofatov
(no #)	Engine crashed when UNIONS were used in a VIEW and that VIEW was used in the WHERE clause inside an subquery.	A. Brinkman
(no #)	Generic code cleanup: structures within Y-valve.	A. Peshkoff, N. Samofatov
Improvement	Single-line comments (--) are now allowed in any position of the SQL statement.	D. Yemanov
(no #)	"Request sycnhronization error" with BREAK statement.	D. Yemanov
625899	Bugcheck 291.	A. Peshkoff
(no #)	PSQL change: EXECUTE VARCHAR is renamed to EXECUTE STATEMENT.	A. Peshkoff
521952	No current record for fetch operation.	A. Brinkman
(no #)	QLI doesn't understand BIGINT datatype.	D. Yemanov
(no #)	Length of text variables inside procs/triggers wasn't copied to descriptor structure.	A. Brinkman
(no #)	FIRST/SKIP and ORDER BY changes -- 1. Implemented ORDER BY clause in subqueries. 2. Disallowed FIRST/SKIP for views. 3. Allowed zero as valid argument for FIRST.	D. Yemanov
(no #)	Buffer overflow (MAXPATHLEN) and rewritten local function dirname.	Erik Kunze
(no #)	Make SQLDA parameter mapping consistent with order and number of parameters in source SQL string. NOTE You can enable older mapping behavior (for backward compatibility) using "OldParameterOrdering" configuration manager parameter.	N. Samofatov
Improvement	Improved optimizer: let subqueries also use indices when their parent is a stored procedure.	A. Brinkman
(no #)	Removed request size limitation.	D. Yemanov
(no #)	Nulls first/last and collation handling in "order by" clause of unions	N. Samofatov

Tracker #	Description	Contributor
(no #)	Generic code cleanup; renamings, new safe macros, support for mingw.	Erik Kunze, Ignacio J. Ortega, D. Sibiryakov
(no #)	Explicit record locking implementation finalized. Should be stable and consistent now.	N. Samofatov
Improvement	Improved optimizer: better handling of AND nodes inside an OR node.	A. Brinkman
(no #)	Exceptions inside for/while loop in triggers are not handled correctly.	A. Peshkoff
623992	Double forward slash in connection string.	Paul Reeves, Mark O'Donohue
(no #)	Deadlock during some database operations.	A. Peshkoff
Improvement	Improved optimizer: if a few indices with much different selectivity could be used for index retrieval, only better of them are used while others are ignored.	D. Yemanov
(no #)	Quoted identifiers problem in plan expressions.	N. Samofatov
Improvement	CS architecture is now supported on Win32, but it still cannot be considered stable, so any feedback is welcome.	D. Yemanov
(no #)	Stored procedures are no longer recompiled before deletion.	N. Samofatov
(no #)	New collation for WIN1251 charset: WIN1251_UA for both Ukrainian and Russian languages.	D. Yemanov
(no #)	Client library change: API routines are no longer exported by ordinals.	D. Yemanov
Improvement	New configuration manager: enable the same plain file based configuration for all supported platforms.	D. Yemanov
Improvement	Improved optimizer: added better support for using indices with "OR". Pick the best available compound index from all "AND" nodes.	A. Brinkman
Improvement	Added support for explicit savepoint management in DSQL.	N. Samofatov
(no #)	Protocol cleanup: IPX/SPX network protocol is no longer supported.	Sean Leyne
(no #)	Obsolete platforms cleanup: some platform are no longer supported by the current source code. DELTA, IMP, DG_X86, M88K, UNIXWARE, Ultrix, NeXT, ALPHA_NT, DGUX, MPE/XL, DecOSF, SGI, HP700, Netware, MSDOS, SUN3_3	Sean Leyne
Improvement	Improved optimizer: added support for detecting use of index with sub-selects in aggregate select.	A. Brinkman
Improvement	Improved thread scheduler for Win32 SS: now the server should be more responsive under heavy load.	A. Peshkoff

Tracker #	Description	Contributor
Improvement	Added support for explicit locking. Wait behavior in isc_tpb_wait transaction modes is not stable yet. Syntax: SELECT <...> [FOR UPDATE [OF col [, col ...]] [WITH LOCK]]	N. Samofatov
558364	Triggers fail to compile if PLAN used.	Ignacio J. Ortega
(no #)	Distributed (2PC) transaction cannot be properly rolled back due to network errors.	Vlad Horsun, Erik Kunze
(no #)	Generic cleanup: ISC_STATUS_LENGTH and MAXPATHLEN macros.	Erik Kunze
496784	When optimizer finds indexes for LEFT JOIN, work like INNER JOIN. Fixed problem which caused complex outer joins to produce wrong results.	N. Samofatov
(no #)	BLOB subtype is ignored in system domains generated for expression fields in views.	D. Yemanov
(no #)	Fixed installation bug: instreg.exe doesn't create "GuardianOptions" registry value.	D. Yemanov
(no #)	Resource leaks in DDL recursive procedure handling which caused some DDL to fail.	N. Samofatov
(no #)	Check constraint which uses only one table field is now dropped automatically when this field is dropped.	N. Samofatov
451927	New ROWS_AFFECTED system variable in PSQL: return number of rows affected by the last INSERT/UPDATE/DELETE statement. For any other statement than INSERT/UPDATE/DELETE, result is always zero.	D. Yemanov
446240	Dynamic exception messages: allow to throw an exception with a message different to the one the exception was created with. Syntax: EXCEPTION name [value];	D. Yemanov
547383	New SQLCODE and GDSCODE system variables providing access to the code of the caught error within the WHEN-block in PSQL. Outside WHEN-block, returns 0 (success).	D. Yemanov
(no #)	Exception re-initiate semantics: allows an already caught exception in PSQL to be re-thrown from the WHEN-block. Syntax: EXCEPTION; No effect outside WHEN-block.	“Digitman”
(no #)	The server crashes during the garbage collection under heavy load.	N. Samofatov

Tracker #	Description	Contributor
Improvement	Deferred metadata compilation: solved a lot of causes of the well-known "object in use" error.	N. Samofatov
Improvement	New NULL order handling: allow user-defined ordering of NULLs.	N. Samofatov
(no #)	gstat showed wrong value for maxdup element.	D. Kuzmenko
(no #)	New registry key is used on win32: SOFTWARE\FirebirdSQL\Firebird.	--
451925	User-defined constraint index names: allows name of an index enforcing a constraint to be either constraint name or user-defined name.	D. Yemanov
Improvement	New RECREATE VIEW statement: shorthand for DROP VIEW / CREATE VIEW coupling of statements. Syntax: RECREATE VIEW name <view_definition>;	D. Yemanov
(no #)	Trigger which name starts with 'RDB$' cannot be altered or dropped at all.	D. Yemanov
(no #)	Renamed distribution files to make sure we're Firebird. Now they're fbserver, fbclient, firebird.msg etc. The client library is fbclient now and it should be used in all new FB-based projects. gds32 contains nothing but redirected exports and is provided for compatibility only.	Various
(Minor ODS upgrade)	Added new system indices (RDB$INDEX_41, RDB$INDEX_42, RDB$INDEX_43), now ODS version is 10.1.	D. Yemanov, N. Samofatov
451935	New CREATE OR ALTER statement for triggers and stored procedures, allows creating or altering a database object according to whether it exists or not. Syntax: CREATE OR ALTER name <object_definition>;	D. Yemanov
(no #)	Broken dependencies (like DB$34) appear in the database after metadata changes.	D. Yemanov
(no #)	Enhanced declaration of local variables: simplify syntax and allow declaring and defining variable at the same time. Syntax: DECLARE [VARIABLE] name <variable_type> [{'=' | DEFAULT} value]; Example: DECLARE my_var INTEGER = 123;	Claudio Valderrama
(no #)	Disabled BREAK statement for triggers (like EXIT) due to known internal limitations.	D. Yemanov

Tracker #	Description	Contributor
555839, 546274	Enhanced grouping: allow to GROUP BY internal functions and subqueries. Also allow to GROUP BY ordinal (i.e. column position, a.k.a degree of column in output set).	A. Brinkman
451917	New COALESCE internal function allowing a column value to be calculated by a number of expressions, the first expression returning a non NULL value is returned as the column value.	A. Brinkman
451917	New NULLIF internal function returns NULL for a sub-expression if it has a specific value, otherwise returns the value of the sub-expression.	A. Brinkman
451917	New CASE internal function allows the result of a column to be determined by a the results of a case expression.	A. Brinkman
545725	Automatic/background sweep hangs.	A. Peshkoff
(no #)	The server crashes when XSQLDA structures are not prepared for all statement parameters.	D. Yemanov
(no #)	PSQL: enabled support for empty BEGIN...END blocks.	D. Yemanov
567931	Partly fixed metadata security hole.	D. Yemanov
(no #)	BigInt arrays didn't work.	Artem Petkevych
437859	Implemented execute procedure and string concat, allowing any expression to be used as a SP parameter.	D. Yemanov
562417	Aggregate concatenated empty char.	D. Yemanov
Improvement	Readline (cmd history) support added to ISQL.	M. O'Donohue
446206	New BIGINT datatype allowing native SQL usage of 64-bit exact numerics (Dialect 3 only).	D. Yemanov
451922	Universal triggers allowing one trigger to be fired for a number of action types.	D. Yemanov
446238, 446243	New CONNECTION_ID and TRANSACTION_ID system available in PSQL. Return appropriate internal identifier stored on the database header page.	D. Yemanov
446180	Server-side database aliases: attach to any database using an "alias" name instead of its physical pathname. The list of known database aliases is stored in aliases.conf file under the server installation root. Example: alias entry in the configuration file: my_database = c:\dbs\my\database.gdb connection string in application: localhost:my_database	D. Yemanov
(no #)	New plugin manager and INTL interface.	John Bellardo

Tracker #	Description	Contributor
Improvement	In-memory sorting: if SORT plan is used for a SQL statement, the sorting is done in memory. If there's not enough memory for this operation, reverts to old method using temporary file.	D. Yemanov
538201	Crash with extract from null as date.	Claudio Valderrama
446256	New EXECUTE VARCHAR PSQL extension statement allows execution of dynamic SQL statements in SPs/triggers. (Subsequently renamed to EXECUTE STATEMENT).	A. Peshkoff
(no #)	Major code cleanup.	Sean Leyne, Erik Kunze
(no #)	New memory manager.	John Bellardo
(no #)	New exception handling logic.	Mike Nordell, John Bellardo
(no #)	New autoconf-based build configuration.	John Bellardo, M. O'Donohue, Erik Kunze
(no #)	The code port from C to C++.	Mike Nordell, John Bellardo, M. O'Donohue

Riconoscimenti

La traduzione italiana delle Note sulla release 1.5 di Firebird è stata curata da

Alessandro Zanello – (alexza@mail.nauta.it),

con validi suggerimenti iniziali di

Marco Kregar (marco.kregar@firebirdsql.it).