# Firebird 2.1 Release Notes

Helen Borrie (Collator/Editor)

4 Settembre 2009 - Documento v.0213_13 - per Firebird 2.1.3

# Firebird 2.1 Release Notes

4 Settembre 2009 - Documento v.0213_13 - per Firebird 2.1.3

Helen Borrie (Collator/Editor)
Traduzione in italiano: Umberto Masotti

# Sommario

# Lista delle Tabelle

Capitolo 1

# Note Generali

Firebird 2.1 è una nuova completa release costruita sulle modifiche architetturali introdotte con la versione 2.0. Grazie a tutti quelli che hanno provato le versioni Alfa e Beta durante tutto il 2007 e gli inizi del 2008, ora abbiamo una release che risplende di nuove caratteristiche e di notevoli miglioramenti, alcuni dei quali attesi da tanto tanto tempo, tra cui le tabelle temporanee globali (global temporary tables), un insieme di meccanismi per monitorare in tempo reale il server, i trigger di database e l'aggiunta di una dozzina di funzione interne nel linguaggio SQL.

## Release intermedia 2.1.3

**Importante**

Firebird 2.1.3 corregge diversi problemi scoperti subito dopo il rilascio della versione 2.1.2. Questo rilascio sostituisce completamente la V.2.1.2, così come la precedente 2.1.1 e l'originale V.2.1. Le precedenti versioni 2.1.x dovrebbero essere sostituite con la V.2.1.3 poiché è stata ufficialmente rilasciata.

### *Questioni notevoli*

- Il metodo di autenticazione degli utenti sulla piattaforme Windows non è più per default *mixed*. Si è cambiato in *native*. Per maggiori informazioni, vedere le note sul parametro di configurazione Authentication.
- In Firebird 2.5 è stato sistemato un comportamento «selvaggio» dell'utilità nBackup ed è stato riportato anche in questo rilascio. Per altre informazioni vedere la nota Miglioramenti della V.2.1.3 nella sezione nBackup del capitolo sulle *Utilità*.
- A grande richiesta, Dmitry Yemanov ha aggiunto delle utili linee guida sulle prestazioni riguardo al funzionamento interno del sistema di monitoraggio dei database (Sarebbero le tabelle «MON$»). Ciò per assistere nell'uso di tali strumenti su sistemi pesantemente caricati in modo da ottenere le informazioni desiderate minimizzando l'impatto sulle prestazioni per l'utente finale.
- **Problemi noti**

  Ad un certo punto dello sviluppo delle V.2.1, si è prodotta una spaccature nel comportamento del parametro di configurazione *TempDirectories*, con cui si può configurare una lista di spazi dove il motore può scrivere risultati intermedi per gli ordinamenti quando non ha più spazio a disposizione per tenerli in memoria RAM. A partire dalla V.2.1 (e tuttora), sebbene il motore usi il primo degli spazi elencati in TempDirectories, non riesce a «mettere le mani» su nessun'altro degli spazi listati se il primo esaurisce tutto lo spazio. (Riferimento sul tracker CORE-2422).

  A causa dell'impatto sul codice relativo a ciò, tale problema non può essere aggiustato in questa V.2.1.3. L'unica alternativa possibile al momento è assicurarsi di configurare abbastanza spazio nella prima ed unica locazione che il motore riesce ad accedere.

## Release intermedia 2.1.2

Con Firebird 2.1.2 si correggono un certo numero di problemi che si sono verificati fin dalle versioni 2.1 e 2.1.1.

> **Modifica importante nei parametri del DPB nelle API**
>
> Per parecchio tempo, attraverso una scappatoia nella gestione dei parametri del DPB, si potevano abilitare gli utenti ordinare ad impostare parametri di connessione che avrebbero potuto danneggiare le basi di dati o attribuire loro l'accesso ad operazioni riservate al SYSDBA. Questo buco è stato chiuso, ma è una modifica che può influenzare le applicazioni esistenti, gli strumenti di gestione, e i gestori di connessione (driver o componenti). I dettagli sono nel capitolo 3, Modifiche alle API di Firebird e all'ODS.

Inoltre include tre aggiustamenti minori provenienti da richieste degli utenti:

- In CORE-2243 si richiede un miglioramento per ridurre la confusione e le duplicazioni derivate dalle regole del Microsoft Visual C 8 runtime assembly per le piattaforme XP, Server2003 e Vista, che hanno finora danneggiato l'installazione del pacchetto per Windows della versione 2.1. Vlad Khorsun ha soddisfatto tali richieste in questa versione. Per maggiori informazioni, fare riferimento alla sezione per l'installazione su Windows del documento *Firebird 2 Migrazione e installazione*.

- Dmitry Yemanov ha migliorato il sistema di monitoraggio in modo da permettere ad un utente non SYSDBA di consultare lo stato di tutte le sue connessioni, non solo la CURRENT_CONNECTION dell'utente. Questo è stato richiesto in CORE-2233. Vedere anche Monitorare connessioni multiple nel capitolo relativo alle *Funzioni di amministrazione*.

- Dmitry ha risolto un problema di lentezza nell'esecuzione di una cross join tra tre o più tabelle, in modo particolare quando una o più di quelle tabelle sono vuote. Vedere queste note e nel tracker CORE-2200.

## La release di correzione 2.1.1

La release 2.1.1 di Firebird 2.1.1 corregge una serie di problemi che sono stati evidnziati dopo il rilascio della versione finale 2.1. Sono inclusi anche correzioni per i problemi che hanno reso il programma di utilità nBackup inutilizzabile in quella release, alcune modifiche per migliorare le caratteristiche delle nuove funzioni di monitoraggio nei sistemi molto carichi ed altre modifiche per risolvere problemi di rallentamento in certe operazioni DML. Per i dettagli fare riferimento alla versione più recente del documento sulle correzioni (Bugfix) che accompagna queste note.

> **Importante**
>
> Si è scoperto che finora il database della sicurezza veniva creato con le Forced Writes ad OFF. Come qualsiasi database con le FW disabilitate, il database della sicurezza risulta vulnerabile nei sistemi dove non è garantita la stabilizzazione dell'alimentazione elettrica. Per tanto la subrelease 2.1.1 e tutte le successive rettificano questa situazione e creano il database *security2.fdb* con le FW ad ON.
>
> Per importare un database della sicurezza esistente da ogni precedente rilascio, si raccomanda di usare il programma di utilità *gfix* per attivare le Forced Writes.
>
> Notare che, impostando le Forced Writes ad ON ad qualsiasi database su un server Firebird 1.5.x o precedenti su Linux non avrebbe nessun effetto: le FW non funzionano su Linux in questi server Firebird.

### Riguardo queste note

Queste note per la versione 2.1 sono completate con quanto già scritto durante lo sviluppo ed il rilascio nel passaggio dalla versione 2.0 a questa 2.1.x. Una documentazione separata per la versione 2.0.x non viene distribuita con il package della 2.1.x.

Le sezioni che riguardano l'installazione, la migrazione e compatibilità ed i problemi risolti, sono state rimosse da questo documento e sono distribuiti come documenti indipendenti nella sottodirectory `$fbroot$/doc/`. Come le note di rilascio, ese coprono sia la versione 2.0 che la 2.1.

Per ammorbidire la transizione da versioni precedenti, sarebbe necessario studiare attentamente per intero sia le note di rilascio che la guida di migrazione e installazione. Inoltre si consiglia di prendersi il tempo necessario a collaudare questa release con l'applicazione finale, possibilmente con dati e carico di lavoro reali. Alcune query potrebbero non funzionare come previsto o non funzionare più per nulla, a causa di una serie di correzioni implementate nella logica. Per alcuni casi sono stati previsti dei sistemi temporanei o aggiramenti tecnici (workarounds) per tornare al funzionamento precedente: ovviamente **si consiglia di cercare tali punti nella documentazione prima fare domande al supporto tecnico**.

## Dal team di supporto

Sebbene questa sia stata definita una release stabile, adatta all'ambiente di lavoro, introduce molte novità. Si incoraggia pertanto di valutare quanto queste nuove caratteristiche migliorino il vostro lavoro ed informarci di ogni carenza identificata al più presto.

Vi si invita caldamente a descrivere alla lista di sviluppo firebird (in inglese esclusivamente!) firebird-devel buone descrizioni di ogni eventuale problema o bestiaccia incontrata, oppure ad inserire descrizioni dei problemi incontrati direttamente al nostro Issue Tracker. Infatti si persegue una linea di rilasciare versioni di aggiornamento la cui qualità e periodicità dipendono fortemente dalle comunicazioni ricevute «dal campo».

## Descrivere un problema

- Convinti di aver scoperto un nuovo problema in questa release, è consigliato leggere attentamente le istruzioni presenti nell'articolo How to Report Bugs Effectively per presentare il problema al sito web del progetto Firebird.

- Se si pensa che una risoluzione non abbia funzionato a dovere o abbia causato una regressione, sarebbe molto utile localizzare la descrizione del problema originale nel Tracker, riaprirla se necessario e seguire le istruzioni seguenti.

Queste sono le linee guida per gestire l'analisi dei problemi:

1. È necessario poter descrivere un rapporto dettagliato del problema, descrivendo la versione esatta del kit Firebird utilizzato (di solito un numero di 5 cifre che segue la versione). Inoltre è bene aggiungere anche i dettagli della piattaforma utilizzata (ad es. sistema operativo, processore). Inoltre è molto importante poter aggiungere un test riproducibile ed i relativi dati nel rapporto ed inviare tutto al nostro Tracker.

2. Nel caso, siete caldamente incoraggiati a farvi conoscere come field-tester sottoscrivendo la relativa lista dei field-testers ed inviando la miglior descrizione possibile dei vostri problemi.

3. Per discutere su un problema od i particolari di una implementazione, ci si può iscrivere alla lista degli sviluppatori firebird-devel. In quella lista di posta elettronica si possono leggere tutti i problemi che arrivano al tracker e le relative discussioni riguardo a questa release.

## Altra Documentazione

Si possono trovare alcuni README per molte delle nuove caratteristiche delle versioni 2.0 e 2.1 nel kit di installazione, per default nel sottodirettorio /doc/*.

Una pagina automatizzata delle "Release Notes" nel Tracker accede alle liste ed ai link per tutte le registrazioni di questa versione ivi presenti e tutte le build relative. Si può usare questo link.

Per vostra comodità, le molte correzioni (riguardo buchi e regressioni) effettuate durante lo sviluppo di Firebird 2.0 e 2.1 sono elencate in ordine cronologico discendente in un documento separato (Bugfixes).

—*Firebird 2 Project Team*

Capitolo 2

# Le novità di Firebird 2

## Nuove caratteristiche implementate

Questo capitolo riassume le nuove caratteristiche implementate in Firebird 2, comprendendo sia la serie 2.1 che la 2.0.x.

### Struttura su disco (On-Disk Structure)

I Database creati o recuperati da un backup sotto Firebird 2 hanno una struttura su disco (ODS) di livello 11 o superiore.

- Firebird 2.1 crea database con ODS 11.1. Può leggere database anche ODS inferiori, ma molte delle nuove caratteristiche non saranno disponibili a tali database.
- Firebird 2.0.x crea database con ODS 11 (alle volte indicata come 11.0). Per avere tutte le caratteristiche della versione 2.1, è necessario operare un aggiornamento dalla ODS 11 e precedenti, facendo una copia di backup e recuperando tale copia *con un server v.2.1*.

### Trigger a livello Database

**(v.2.1)** Questi nuovi «database triggers» sono moduli PSQL definiti dall'utente che possono essere utilizzati per l'esecuzione durante gli eventi a livello di connessione o di transazione. Vedere Trigger a livello di database.

### SQL ed Oggetti SQL

#### Tabelle temporanee globali

**(v.2.1)** Sono state implementate secondo lo standard SQL le tabelle temporanee globali (global temporary tables o GTT). Queste tabelle predefinite sono instanziate su richiesta per un uso specifico a livello di transazione o di connessione con dati non persistenti, che il motore Firebird memorizza su tabelle temporanee. Vedere Tabelle temporanee globali (Global Temporary Tables).

#### Espressioni con tabelle comuni (CTE), e query con DSQL recursive.

**(v.2.1)** Si tratta di una nuova caratteristica definita nello standard, per rendere possibili le query recursive dinamiche. Vedere Common Table Expressions.

### Clausola RETURNING

**(v.2.1)** Questa clausola opzionale, RETURNING, è disponibile per tutte le operazioni singole di UPDATE, INSERT e DELETE. Vedere La clausola RETURNING.

### Frase UPDATE OR INSERT

**(v.2.1)** Adesso si possono scrivere frasi SQL capaci di effettuare un aggiornamento ad un record esistente oppure un inserimento, in funzione dell'esistenza o meno del record selezionato. Vedere La frase UPDATE OR INSERT.

### Frase MERGE

**(v.2.1)** Questa nuova sintassi esegue un aggiornamento ad un record esistente, se soddisfa una certa condizione,oppure esegue un inserimento altrimenti. Vedere La frase MERGE.

### La funzione LIST()

**(v.2.1)** La nuova funzione di aggregazione LIST(<QUALCOSA>) recupera tutti i QUALCOSA in un gruppo e li raggruppa in una lista separata da virgola. Vedere La funzione LIST.

### Tante nuove funzioni integrate

**(v.2.1)** Tutte queste nuove funzioni sostituiscono le vecchie equivalenti funzioni UDF distribuite nelle librerie UDF. Per la lista completa con esempi vedere Funzioni integrate.

### I BLOB «corti» possono trasformarsi in VARCHAR lunghi.

**(v.2.1)** A vari livelli di valutazione, il motore adesso tratta i BLOB di testo di dimensione entro i 32,765 byte come se fossero VARCHAR. Pertanto funzioni come CAST(), LOWER(), UPPER(), TRIM() e SUBSTRING funzionano su questi BLOB, così come la concatenzione e l'assegnazione ai tipi carattere. Vedere Compatibilità fra testo e BLOB.

> **Uso di SUBSTRING con i BLOB di testo**
>
> Se il primo argomento della funzione SUBSTRING() è un BLOB di testo, riporta un BLOB e non più un VARCHAR.</para>

## SQL Procedurale

### Uso dei domini per definire variabili ed argomenti in PSQL

**(v.2.1)** Le variabili locali e gli argomenti di ingresso/uscita PSQL per le stored procedure possono essere ora dichiarate usando i domini invece dei tipi di dato canonici. Vedere Domini in PSQL.

### COLLATE nelle Stored Procedure e nei Parametri

**(v.2.1)** Ordinamenti lessicografici possono essere applicati agli argomenti e alle variabili PSQL. Vedere COLLATE nelle STORED PROCEDURE.

### Miglioramenti nella gestione degli errori PSQL

V. Khorsun

Richiesta CORE-970

**(v.2.1)** Negli errori PSQL adesso sono evidenziati anche linea e colonna in cui è avvenuto l'errore.

## Sicurezza

### Utenti autenticati attraverso la sicurezza di Windows

**(v.2.1)** Il sistema di validazione degli utenti di Windows può essere utilizzato per autenticare gli utenti Firebird su un server Windows. Vedere Sicurezza con Windows Trusted Users.

## Supporto alle lingue internazionali

### Il comando CREATE COLLATION

**(v.2.1)** Il comando DDL `CREATE COLLATION` è stato introdotto per creare un ordinamento lessicografico, ovviando alla necessità di utilizzare uno script. Vedere CREATE COLLATION.

### Ordinamenti Unicode dovunque

**(v.2.1)** Due nuovi ordinamenti Unicode possono essere applicati ad ogni set di caratteri usando un nuovo meccanismo. Vedere UNICODE Collations.

## Nuove piattaforme supportate

### Windows 2003 64-bit

D. Yemanov

Richieste CORE-819 e CORE-682

**(v.2.1)** La piattaforma a 64-bit Windows (AMD64 e Intel EM64T) è supportata per i modelli Classic, Superserver ed Embedded.

# Amministrazione

## Monitoraggio dei database attraverso SQL

**(v.2.1)** È stato implementato un sistema di monitoraggio in tempo reale (transazioni, tabelle, ecc.) in SQL attraverso nuove tabelle di sistema virtuali. Vedere Monitoring Tables.

Nel set di tabelle, ce n'è una chiamata MON$DATABASE che contiene molte delle informazioni contenute nell'header del database che non era possibile ottenere precedentemente in SQL: si tratta di informazioni quali versione della struttura su disco (ODS), il dialetto SQL, l'intervallo di sweep, OIT e OAT ed altro ancora.

Si può utilizzare l'informazione ottenuta dalle tabelle di monitoraggio per cancellare una query interminabile. Vedere Cancel a Running Query.

## Maggiori informazioni di contesto

Nel contesto sono state aggiunte informazioni sulla versione del motore server, in modo da recuperarle attraverso una SELECT alla funzione RDB$GET_CONTEXT. Vedere More Context Information.

## Nuovo programma di utilità a linea di comando: fbsvcmgr

**(V.2.1)** La nuova utility *fbsvcmgr* serve come interfaccia a linea di comando all'API di servizio, abilitando l'accesso ad ogni servizio eventualmente abilitato in Firebird.

Sebbene ci siano molti strumenti di amministrazione che si interfacciano all'API di servizio attraverso un'interfaccia grafica, il nuovo strumento risolve il problema degli amministratori che devono accedere a sistemi Unix remoti attraverso un'interfaccia unicamente testuale. Precedentemente una tale richiesta richiedeva un programmatore. Qui i dettagli.

# Interfaccia remota

**(v.2.1)** Il protocollo remoto è stato ampiamente migliorato per operare meglio nelle reti lente dal momento stesso in cui i driver vengono aggiornati per utilizzare le modifiche. I controlli hanno mostrato una diminuzione di round trip API del 50% circa, con una diminuzione del 40% dei round trip TCP. Vedere Remote Interface Improvement.

# Tabelle derivate

A. Brinkman

Il supporto per le tabelle derivate in DSQL (cioè subqueries nella clausola FROM) come definite nello standard SQL200X. Una tabella derivata è un set derivato da uno statement dinamico di SELECT. Le tabelle derivate possono essere innestate, se necessario, per costruire query ancora più complesse e possono far parte di JOIN come se fossero normali tabelle o viste.

Altri dettagli in Tabelle derivate nel capitolo relativo al DML.

## Supporto PSQL ai cursori nominativi

D. Yemanov

Sono supportati più cursori nominativi (cioè espliciti) in PSQL e nello statement DSQL EXECUTE BLOCK. Altre informazioni nel capitolo sul PSQL Cursori esplicitati.

## Rifatto il protocollo su Windows

D. Yemanov

Sono state fatte due modifiche sostanziali al protocollo in Windows.

### Protocollo Locale--XNET

Firebird 2.0 ha modificato la precedente implementazione del protocollo di trasporto locale (spesso chiamata IPC oppure IPServer) con una completamente nuova, chiamata XNET.

Ha esattamente lo stesso obiettivo, permettere un modo efficiente di collegare un server locato sulla stessa macchina del client senza l'uso di un nodo remoto nella stringa di connessione. La nuova implementazione è differente e risolve tutti i problemi noti del precedente sistema.

Come il vecchio IPServer, anche XNET usa memoria condivisa per la comunicazione fra processi. Tuttavia XNET elimina l'uso di messaggi window per inviare richieste di connessione ed inoltre implementa una diversa logica di sincronizzazione.

#### Benefici di XNET rispetto IPServer

Oltre a dare un protocollo di comunicazione più robusto, XNET porta con sé alcuni evidenti benefici effetti:

- funziona col Classic Server
- funziona con servizi non interattivi e le sessioni terminali
- elimina i blocchi quando vengono tentate una certo numero di connessioni contemporanee

#### Prestazioni

Ci si aspetta che XNET sia un po' più veloce di IPServer, o al peggio uguale.

#### Svantaggi

L'unico svantaggio è che sono fra loro incompatibili. Questo fa sì che la versione di fbclient.dll debba per forza essere identica alla versione del server che si sta usando (fbserver.exe o fb_inet_server.exe). Non è possibile instaurare una connessione locale se questo dettaglio viene ignorato. (Una connessione TCP localhost attraverso tra client e server disaccoppiati risolve il problema, naturalmente).

### Modifiche al protocollo WNET («NetBEUI»)

Il protocollo WNET (noto anche come NetBEUI) non permette più la personificazione del client.

In tutte le precedenti versioni di Firebird, le richieste remote attraverso WNET venivano eseguite nel contesto di un *client security token*. Poiché il server soddisfa ogni connessione in accordo alle sue credenziali di sicurezza, questo significa che, se la macchina client è in un sistema operativo interno ad un dominio NT, quell'utente dovrebbe avere permessi fisici appropriati per accedere al file del database, alle librerie UDF, ecc, nel sistema di file del server. Questa situazione è contraria a quanto generalmente viene visto come un appropriata impostazione per proteggere i database in un sistema client-server.

Tale personificazione è stata rimossa nelle connessioni WNET in Firebird 2.0, che adesso sono veramente client-server e funzionano allo stesso modo di quelle TCP, cioè non fanno nessuna assunzione riguardo ai diritti del di sistema dell'utente.

## Rielaborazione del sistema di Garbage Collection

V. Khorsun

Fin da prima di Firebird 1.0, il motore Superserver eseguiva un sistema di *background garbage collection* (un processo per il recupero della memoria non più utilizzata), gestendo informazioni su ogni nuova versione dei record prodotta da una frase UPDATE o DELETE. Nel momento stesso in cui la vecchia versione non risultava più «interessante», cioè diventava più vecchia della più vecchia transazione snapshot (Oldest Snapshot, si vede nell'output del comando *gstat -header*) il motore segnala al sistema di recupero di rimuoverla o, in altre parole, di liberare la memoria che prima era occupata dalla versione del record.

Il motore Classic e le versioni di Interbase precedenti la 6.0 avevano un altro sistema di Garbage Collection, noto come *garbage collection cooperativo*. Questo sistema, cioè il GC cooperativo, ha la necessità di rileggere le pagine contenenti le versioni dei record non più interessanti attraverso un `SELECT COUNT(*) FROM aTable` oppure altre query di scansione di tabella da parte di un utente. Il GC in background non ha questa necessità; inoltre risolve il problema di recuperare la memoria per quelle pagine che vengono lette raramente.

Uno sweep, naturalmente, troverebbe queste versioni dei record non utilizzate e le cancellerebbe, ma questo non accade necessariamente a breve termine. Un ulteriore beneficio è la riduzione di I/O, a causa dell'alta probabilità che le pagine richieste risiedano già nella memoria cache.

Tra il momento in cui il motore notifica al garbage collector l'esistenza di pagine che contengono versioni non più usate ed il momento che il garbage collector riesca a leggere quella pagina, una nuova transazione potrebbe aggiornare uno dei record contenuti. Il garbage collector non può pulire il record se questo numero di transazione successivo è maggiore dell'Oldest Snapshot oppure è ancora attivo. Il motore poi notifica ancora al garbage collector lo stesso numero di pagina, sovrascrivendo la precedente notifica e pertanto la pulizia dovrà essere fatta in un tempo successivo.

In Firebird 2.0 Superserver, sono possibili contemporaneamente sia il GC cooperativo che quello in background. Per gestirli, è stato introdotto un nuovo parametro di configurazione *GCPolicy* che può essere impostato a:

- cooperative - il garbage collector viene gestito solamente in modo cooperativo (come nel Classic) ed il motore non tiene traccia delle vecchie versioni dei record. Questo riporta il funzionamento del GC a quello di IB 5.6. È l'unica opzione per la versione Classic.

- background - il garbage collector viene gestito solo da threads in background, come nel caso di Firebird 1.5 e precedenti. Le scansioni di tabelle da parte dell'utente non rimuovono le versioni dei record non più usate ma provocano la notifica al thread di GC di ogni pagina dove viene trovato una versione di record non più usata. Il motore inoltre ricorda quei numeri di pagina dove le frasi di UPDATE e DELETE hanno creato delle versioni obsolete.

- combined (l'installazione default per Superserver) - sono attivi sia il GC in background che il cooperativo.

> **Nota**
>
> Il server di tipo Classic ignora questo parametro e lavora sempre in modalità «cooperativa».

## Completamento della API di servizio al motore Classic

N. Samofatov

È stato completata la codifica delle API di servizio per l'architettura Classic. Sono pertanto disponibili tutte le funzioni API sia su Linux che su Windows per i server Classic, senza limitazione alcuna. Sono state eliminate pure tutte le problematiche note riguardo agli errori di *gsec* riportate nelle precedenti versioni di Firebird.

## Scadenza del tempo di blocco nelle WAIT Transactions

A. Karyakin, D. Yemanov

Tutte le versioni di Firebird hanno due modi di attesa per le transazioni: *NO WAIT* e *WAIT*. Il modo NO WAIT significa che i conflitti di blocco e di abbraccio mortale (deadlock) sono riportati immediatamente, mentre con WAIT la transazione attende bloccata finché le transazioni concorrenti finiscono venendo confermate (commit) o rinunciate (rolled back).

Questa nuova caratteristica estende la modalità WAIT permettendo di impostare un tempo finito di attesa delle transazioni concorrenti. Alla scadenza del tempo viene riportato un errore (isc_lock_timeout).

Gli intervalli di attesa sono specificati a livello di transazione, usando la nuova costante TPB *isc_tpb_lock_timeout* nell'API oppure, in DSQL, la clausola *LOCK TIMEOUT <value>* della frase SET TRANSACTION.

## Nuova implementazione degli operatori di ricerca nelle stringhe

N. Samofatov

1. Gli operatori ora operano correttamente con i BLOB di ogni dimensione. Sono stati rimossi tutti i problemi noti al riguardo (ricerca solo nel primo segmento, ricerche mancate perché a cavallo di segmenti).

2. Il riconoscimento delle stringhe ora usa l'algoritmo a passo singolo Knuth-Morris-Pratt, migliorando le prestazioni quando vengono adoperati campioni di ricerca complessi.

3. Il motore non crepa più quando si usa NULL come carattere di ESCAPE nell'operatore LIKE

## Rifacimento delle viste aggiornabili

D. Yemanov

Un rifacimento è stato fatto per risolvere i problemi relativi alle viste che sono implicitamente aggiornabili, ma mantengono ancora dei trigger di UPDATE. Questa è un'importante modifica che influenza i sistemi con procedure scritte per avvantaggiarsi di una (dis)funzione non documentata delle precedenti versioni.

Per i dettagli, vedere nelle note nel capitolo riguardo alla compatibilità nel documento separato delle Note di Installazione.

## Introduzione di modalità addizionali di spegnimento

N. Samofatov

Sono stati implementate le modalità Single-user (utente singolo) e full shutdown (spegnimento completo) usando il nuovo parametro *[state]* per i comandi `gfix -shut` e `gfix -online`.

**Sintassi attuale**

```
gfix <command> [<state>] [<options>]
<command>> ::= {-shut | -online}
<state> ::= {normal | multi | single | full}
<options> ::= {-force <timeout> | -tran | -attach}
```

- *normal* state = il database è in linea
- *multi* state = modalità spegnimento multi utente (quello usuale, sono permesse connessioni multiple di SYSDBA o del proprietario)
- *single* state = spegnimento single-user (solo una connessione viene permessa, usata dal processo di recupero)
- *full* state = spegnimento completo o esclusivo (non viene permessa alcuna connessione)

Per ulteriori dettagli, vedere la sezione su Gfix in New Shutdown Modes, nel capitolo dei programmi di utilità.

Per una lista dei simboli di stato di spegnimento ed esempi di di uso, vedere Shutdown State in the API.

## Migliorata la gestione dei NULL nelle UDF

C. Valderrama

### Passare un NULL SQL

- Possibilità di passare un NULL SQL attraverso un puntatore NULL (vedere Signal SQL NULL in UDFs).
- Le funzioni esterne della libreria ib_udf sono state aggiornate così da permettere alle funzioni stringa ASCII_CHAR, LOWER, LPAD, LTRIM, RPAD, RTIM, SUBSTR e SUBSTRLEN di riporare un NULL e di interpretarlo correttamente nei parametri.

  Lo script `ib_udf_upgrade.sql` può essere applicato ai database precedenti alla versione 2.0 che hanno queste funzioni dichiarate, per aggiornarne la dichiarazione in modo da funzionare con le nuove versioni della libreria. Questo script deve essere usato solo usando la nuova libreria ib_udf con Firebird 2.* e le chiamate dovranno essere modificate per gestire i NULL.

## Controllo in Run-time per il sovradimensionamento in concatenazione

D. Yemanov

Il controllo per il sovradimensionamento nella concatenazione di stringhe è stato spostato dalla fase di compilazione alla fase di run-time.

A partire da Firebird 1.0, le operazioni di concatenazione vengono controllate per evitare che la stringa risultante possa superare il limite di circa 32,000 bytes, cioè un sovradimensionamento (overflow). Questo controllo veniva effettuato durante la fase di preparazione dello statement, usando le dimensioni dichiarate degli operandi ed avrebbe dato errore per una espressione dl tipo:

```
CAST('qwe' AS VARCHAR(30000)) || CAST('rty' AS VARCHAR(30000))
```

A partire da Firebird 2.0, questa espressione lancia solo un'avvertenza (warning) durante la fase di preparazione ed il controllo di sovradimensionamento viene ripetuto all'esecuzione, usando la dimensione degli operandi reali. Il risultato permette all'esempio di essere eseguito senza errori. L'eccezione *isc_concat_overflow* viene ora pertanto lanciata solo per i sovradimensionamenti reali, riportando pertanto il funzionamento della loro rilevazione simile a quella delle operazioni aritmetiche.

## Modifiche alla logica di sincronizzazione

N. Samofatov

(N.d.T. quanto segue è veramente tecnico anche in lingua originale. Di solito rinuncio a tradurre tali termini quando sono così per addetti ai lavori.)

1. È stata ridotta significativamente la lock contention nel lock manager e nel SuperServer thread pool manager.
2. È stata trovata e corretta una rara condizione di deriva che poteva bloccare il Superserver nel processare le richieste fino all'arrivo di una nuova successiva richiesta.
3. I memory dump del lock manager danno maggiori informazioni e OWN_hung viene correttamente recepito.
4. È stato implementato il disaccoppiamento degli oggetti di sincronizzazione del lock manager per le differenti istanze del motore.

## Supporto alle piattaforme a 64-bit

A. Peshkov, N. Samofatov

Firebird 2.0 supporta le piattaforme a 64-bit.

## Aumentati i limiti di conteggio dei record

N. Samofatov

È stato introdotto il conteggio dei record a 40-bit (64-bit internamente) per superare il limite di circa 30GB per tabella imposto dal precedente limite a 32-bit.

## Miglioramenti per il debugging

Contributi Vari

### Migliorati i rapporti da Bugcheck

I messaggi di log BUGCHECK ora includono il nome del file ed il numero di linea sorgente. (A. Brinkman)

### Aggiornato il rapporto della struttura interna

Sono state aggiornate le procedure che stampano le varie strutture interne (DSQL node tree, BLR, DYN, ecc). (N. Samofatov)

### Nuovi strumenti per il log di debug

Sono state implementate procedure di utilità thread-safe e signal-safe per il debug. (N. Samofatov)

### Miglioramenti nella diagnostica

I messaggi di Syslog vengono copiate alla tty dell'utente se c'è attaccato un processo. (A. Peshkov)

## Migliorata la gestione delle connessioni su Superserver POSIX

A. Peshkov

Le versioni per Posix SS ora gestiscono SIGTERM e SIGINT per chiudere tutte le connessioni in modo corretto e non brusco. (A. Peshkov)

## Rifatta la gestione degli invarianti in PSQL

N. Samofatov

Il tracciamento degli invarianti è quel processo eseguito dal compilatore BLR e dall'ottimizzatore per decidere se una certa espressione (come potrebbe essere una subquery interna) è "invariante", cioè indipendente dal contesto principale. Viene usato per eseguire una unica valutazione di tali espressioni e di riutilizzare i risultati senza ricalcolarli.

La gestione del tracciamento degli invarianti in PSQL e la logica di duplicazione delle richieste sono state rifatte per risolvere un certo numero di problemi con le procedure recursive, ad esempio SF bug #627057.

Se alcune invarianti sfuggono alla determinazione, c'è una perdita di prestazioni. Se alcune variabili sono trattate come invarianti, si ottengono risultati errati.

**Esempio**

```
select * from rdb$relations
  where rdb$relation_id <
    ( select rdb$relation_id from rdb$database )
```

Questa query esegue solo una lettura da rdb$database invece di valutare la subquery per ogni riga di rdb$relations.

## Supporto per la sintassi ROLLBACK RETAIN

D. Yemanov

Firebird 2.0 aggiunge un opzionale `clausola RETAIN` allo statement DSQL `ROLLBACK` per renderlo simmetrico al COMMIT [RETAIN].

Vedere ROLLBACK RETAIN nel capitolo sul DML.

## Nei server Win32 non viene più letta la Registry

D. Yemanov

La ricerca del direttorio radice è stata modificata in modo tale che i processi server su Windows non cercano più di leggere la Registry.

> **Importante**
>
> Le utilità a linea di comando leggono e controllano ancora la Registry.

## Altri miglioramenti all'ottimizzatore

A. Brinkman

Nelle procedure dell'ottimizzatore sono stati migliorati i calcoli per la scelta degli indici.

Capitolo 3

# Modifiche alle API di Firebird e all'ODS

## API (Application Programming Interface)

Sono state effettuate alcune necessarie modifiche alle API di Firebird. Queste includono:

### *Restrizioni nel DPB all'utente generico*

A. Peshkov

**Dalla versione 2.1.2**
Alcuni parametri di DPB sono stati resi inaccessibili all'utente generico, in modo da chiudere alcuni buchetti pericolosi. In alcuni casi questi riguardano impostazioni che potrebbero modificare le impostazioni nell'header del database e potenzialmente provocare disastri se non effettuate sotto il controllo dell'amministratore; in altri, iniziano operazioni che sono riservate al solo SYSDBA. Esse sono:

- isc_dpb_shutdown and isc_dpb_online
- isc_dpb_gbak_attach, isc_dpb_gfix_attach and isc_dpb_gstat_attach
- isc_dpb_verify
- isc_dpb_no_db_triggers
- isc_dpb_set_db_sql_dialect
- isc_dpb_sweep_interval
- isc_dpb_force_write
- isc_dpb_no_reserve
- isc_dpb_set_db_readonly
- isc_dpb_set_page_buffers (on Superserver)

Il parametro `isc_dpb_set_page_buffers` può essere ancora usato da utenti generici in architetture Classic per impostare temporaneamente la dimensione del buffer solo per quella sessione di quel particolare utente. Quando è usato dal SYSDBA, in entrambe le architetture (Superserver o Classic) modifica il numero dei buffers direttamente nell'header del database, facendo pertanto una modifica permanente alla dimensione di default del buffer.

**Per gli sviluppatori ed utenti di software ed interfacce per l'accesso ai dati**

Questa modifica coinvolge tutti i parametri di DPB elencati che sono esplicitamente impostati, sia includendoli nel DPB attraverso valori di default delle proprietà o abilitandoli in strumenti o applicazioni che accedono al database come utenti ordinari. Per esempio, un'applicazione Delphi che imposta nelle proprietà dei parametri del database sia 'RESERVE PAGE SPACE=TRUE' che 'FORCED WRITES=TRUE', che non avrebbero dato problemi connettendosi a Firebird 1.x, 2.0.x o 2.1.0/2.1.1, tentando di connettersi a FB 2.1.2 e successivi come utente generico (non SYSDBA) la connessione viene rifiutata con ISC ERROR CODE 335544788, «Unable to perform operation. You must be either SYSDBA or owner of the database.»(Impossibile effettuare l'operazione. Devi essere SYSDBA o proprietario del database)

## Pulizia in ibase.h

D. Yemanov, A. Peshkov

Il file da includere in linguaggio C, ibase.h è stato ripulito, in modo tale che non siano più incluse definizioni private nelle parti di interfaccia pubbliche.

## Scadenza del blocco nelle transazioni WAIT

A. Karyakin, D. Yemanov

Questa nuova caratteristica estende la gestione delle transazioni WAIT dando la possibilità di definire un tempo finito per la transazione corrente. Alla scadenza, viene lanciato un errore (isc_lock_timeout).

Gli intervalli possono essere specificati a livello di transazione, usando nell'API la nuova costante di TPB, *isc_tpb_lock_timeout*.

**Nota**

L'equivalente DSQL viene effettuato attraverso la clausola *LOCK TIMEOUT <value>* dello statement SET TRANSACTION.

## isc_dsql_sql_info() include ora anche gli alias della relazione

D. Yemanov

Per ritrovare anche gli alias delle relazioni, se necessario, è stata estesa la chiamata isc_dsql_sql_info().

## Miglioramenti a isc_blob_lookup_desc()

A. dos Santos Fernandes

isc_blob_lookup_desc() è in grado ora di descrivere i blob che sono output di stored procedure.

## API identifica la versione del client

N. Samofatov

La definizione macro `FB_API_VER` è stata aggiunta in ibase.h per recuperare la versione attuale delle API. Il numero corrisponde alla versione Firebird appropriata.

Il valore attuale di FB_API_VER è 20 (due cifre equivalenti a 2.0). Questa macro può essere usata dalle applicazioni per verificare la versione di ibase.h con cui vengono compilate.

## Aggiunte alla struttura isc_database_info()

V. Khorsun

Alla struttura della chiamata a funzione isc_database_info() sono state aggiunte le seguenti opzioni:

### isc_info_active_tran_count

Riporta il numero delle transazioni attive al momento.

### isc_info_creation_date

Riporta la data e l'ora di quando è stato (ri)creato il database.

Per decodificare il valore riportato, chiamare la funzione `isc_vax_integer` due volte: la prima per estrarre la data e la seconda l'ora della parte ISC_TIMESTAMP. Poi usare la funzione `isc_decode_timestamp()` come al solito.

## Aggiunte alla struttura isc_transaction_info()

V. Khorsun

Alla struttura della chiamata a funzione isc_transaction_info() sono stati aggiunti le seguenti opzioni:

### isc_info_tra_oldest_interesting

Riporta il numero della più vecchia transazione interessante (OIT) all'inizio della transazione attuale. Per le transazioni snapshot, questo coincide col numero della più vecchia transazione nella copia privata della pagina con l'elenco delle transazioni (transaction inventory page, detta TIP).

### isc_info_tra_oldest_active

- Per le transazioni read-committed, riporta il numero della transazione corrente.
- Per tutte le altre transazioni, riporta il numero della più vecchia transazione attiva al momento d'inizio della transazione corrente.

### isc_info_tra_oldest_snapshot

Riporta il numero della minore `tra_oldest_active` fra tutte le transazioni attive all'inizio della transazione corrente.

> **Nota**
>
> Questo valore è usato come soglia ("high-water mark") dal sistema di garbage collection.

### isc_info_tra_isolation

Riporta il livello di isolamento della transazione corrente. Il formato del dato riportato è il seguente:

```
isc_info_tra_isolation,
  1, isc_info_tra_consistency | isc_info_tra_concurrency |
  2, isc_info_tra_read_committed,
     isc_info_tra_no_rec_version | isc_info_tra_rec_version
```

Cioè, per transazioni tipo Read Committed, vengono riportati due elementi (isolation level e record versioning policy) mentre per le altre transazioni si riporta solo un elemento (isolation level).

### isc_info_tra_access

Riporta il modo di accesso (sola lettura oppure lettura-scrittura) della transazione corrente. Il formato degli elementi ritornati è il seguente:

```
isc_info_tra_access, 1, isc_info_tra_readonly | isc_info_tra_readwrite
```

### isc_info_tra_lock_timeout

Riporta il tempo di scadenza del blocco impostato per la transazione corrente (utile per le transazioni WAIT).

## Miglioramenti alle API di servizio

Sono stati eseguiti i seguenti miglioramenti alle API di servizio:

### Parametro isc_spb_trusted_auth

**(V.2.1, ODS 11.1)** `isc_spb_trusted_auth` si applica solo a Windows e viene utilizzato per costringere Firebird ad usare il sistema di autenticazione verificato da Windows per il servizio richiesto.

### Parametro isc_spb_dbname

**(V.2.1, ODS 11.1)** Risulta relativo ad ogni servizio correlato al database della sicurezza. Permette di fornire il nome di tale database quando si invoca un servizio di sicurezza remoto. Risulta equivalente al fornire il parametro `-database` utilizzando il programma di utilità *gsec* da remoto.

### Ottimizzata l'esecuzione delle task

D. Yemanov

I servizi sono eseguiti adesso come thread e non come processi per le architewtture Classic su alcuni sistemi che lo supportano (al momento solo Windows a 32- bit Windows e Solaris).

## Nuova funzione per inviare messaggi d'errore

C. Valderrama

Per estrarre il testo di un messaggio d'errore Firebird dal vettore di stato in un buffer utente, la nuova funzione `fb_interpret()` sostituisce la precedente isc_interprete().

> **Importante**
>
> isc_interprete() è vulnerabile ai sottodimensionamenti ed è sconsigliata e non sicura. Si dovrebbe usare la nuova funzione al suo posto.

## Gestione dei nuovi parametri di stato per il Shutdown

D. Yemanov

L'accesso API alla sconnessione del database è effettuata attraverso i flag aggiunti al parametro isc_dpb_shutdown nell'argomento DBP passato alla isc_attach_database(). I simboli per i flag di stato adesso sono:

```
#define isc_dpb_shut_cache              0x1
#define isc_dpb_shut_attachment         0x2
#define isc_dpb_shut_transaction        0x4
#define isc_dpb_shut_force              0x8
#define isc_dpb_shut_mode_mask         0x70

#define isc_dpb_shut_default            0x0
#define isc_dpb_shut_normal            0x10
#define isc_dpb_shut_multi             0x20
#define isc_dpb_shut_single            0x30
#define isc_dpb_shut_full              0x40
```

**Esempio di uso in C/C++**

```
char dpb_buffer[256], *dpb, *p;
ISC_STATUS status_vector[ISC_STATUS_LENGTH];
isc_db_handle handle = NULL;

dpb = dpb_buffer;

*dpb++ = isc_dpb_version1;

const char* user_name = «SYSDBA»;
const int user_name_length = strlen(user_name);
*dpb++ = isc_dpb_user_name;
*dpb++ = user_name_length;
memcpy(dpb, user_name, user_name_length);
dpb += user_name_length;

const char* user_password = «masterkey»;
const int user_password_length = strlen(user_password);
*dpb++ = isc_dpb_password;
*dpb++ = user_password_length;
```

```
memcpy(dpb, user_password, user_password_length);
dpb += user_password_length;

// Per forzare un completo scollegamento del database
*dpb++ = isc_dpb_shutdown;
*dpb++ = 1;
*dpb++ = isc_dpb_shut_force | isc_dpb_shut_full;

const int dpb_length = dpb - dpb_buffer;

isc_attach_database(status_vector,
                    0, «employee.db»,
                    &handle,
                    dpb_length, dpb_buffer);

if (status_vector[0] == 1 && status_vector[1])
{
  isc_print_status(status_vector);
}
else
{
  isc_detach_database(status_vector, &handle);
}
```

# Modifiche alla struttura su disco (ODS = On-Disk Structure)

Le modifiche alla struttura su disco (ODS) includono:

## *Nuovo numero identificativo*

Firebird 2.0 crea i database con un ODS con versione 11.

## *Aumentata la dimensione massima per i messaggi d'eccezione*

V. Khorsun

La dimensione massima per i messaggi di eccezione è aumentata da 78 a 1021 bytes.

## *Nuovo campo «descrizione» per i generatori*

C. Valderrama

Il campo RDB$DESCRIPTION è stato aggiunto alla tabella RDB$GENERATORS, così da poter aggiungere testi descrittivi ai generatori creati.

## *Nuovi campi descrittivi per gli SQL Roles*

C. Valderrama

I campi RDB$DESCRIPTION e RDB$SYSTEM_FLAG sono stati aggiunti alla tabella RDB$ROLES per permettere rispettivamente un testo descrittivo e determinare i ruoli definiti dall'utente.

## Riconoscimento del tipo di «ODS»

N. Samofatov

Per distinguere i database Firebird da quelli Interbase, è stato introdotto il concetto di tipo di ODS.

## Rapporto errori migliorato per DSQL

C. Valderrama

Il parser del DSQL cerca ora di descrivere anche la linea e la colonna in cui determina una frase incompleta.

## Nuova colonna nella tabella RDB$Index_Segments

D. Yemanov, A. Brinkman

Per memorizzare la selettività di ciascun segmento di un indice multi-chiave, è stata aggiunta la colonna RDB $STATISTICS alla tabella di sistema RDB$INDEX_SEGMENTS.

> **Nota**
>
> C'è una colonna con lo stesso nome nella tabella di sistema RDB$INDICES che viene mantenuta per motivi di compatibilità e tuttora rappresenta la selettività totale dell'indice, ma non è più usata dall'ottimizzatore per un confronto sull'indice completo a partire dai database con ODS11 e successivi.

Capitolo 4

# Miglioramenti di Firebird 2.1

Alcuni miglioramenti e modifiche sono stati effettuati in Firebird 2.1, per avvicinare lo sviluppo del motore alle modifiche architetturali pianificate per Firebird 3.

> **Nota**
>
> A meno di indicazione contraria, si tratta di modifiche e miglioramenti a partire dalla versione v.2.1 e successive.

## Finalmente funzionano le Forced Writes su Linux!

A. Peshkov

Per la massima sicurezza dei database, si usa configurare il server in modalità a scritture sincrone (mettendo a *ON* le c.d. *Forced Writes*). Questo funzionamento, fortemente raccomandato nei sistemi installati presso i clienti, fa in modo che le chiamate di sistema atte a scrivere, cioè **write()**, proseguano solo dopo aver effettivamente e completamente effettuato la scrittura fisica sul disco. Di conseguenza, questo garantisce che dopo un COMMIT, tutti i dati modificati dalla transazione siano fisicamente sul disco, e non in attesa in qualche memoria tampone del sistema operativo.

La sua implementazione su Linux è molto semplice - invocare prima la procedura **fcntl(dbFile, F_SETFL, O_SYNC)**.

*Nonostante ciò, succedeva che i database su Linux fossero rovinati ugualmente.*

### *Un po' di dettagli tecnici*

Alcuni test su Linux mostrarono che impostando o meno O_SYNC scrivendo file non cambiavano per nulla le prestazioni! Bene, si poteva pensare di avere un sistema operativo veloce. Invece no: è un problema documentato del kernel di Linux!

Stando al manuale di Linux, «Su Linux, questo comando (cioè fcntl(fd, F_SETFL, flags)) può solo cambiare i flag O_APPEND, O_ASYNC, O_DIRECT, O_NOATIME, e O_NONBLOCK». Sebbene non sia documentato in nessun posto a me noto, qualsiasi tentativo di impostare ogni altro flag oltre a quelli citati nel manuale (come appunto O_SYNC, ad esempio) oltre a non funzionare, fcntl() non riporta nemmeno nessun messaggio o stato di errore.

Per le versioni di Firebird e Interbase fino a questo punto, significa che le Forced Writes su Linux non hanno *mai* funzionato. Su Windows invece funzionano (e hanno funzionato) correttamente. Sembra inoltre che tal problema non abbia influenzato altri sistemi operativi, sebbene non si possa garantirlo. Per essere sicuri bisogna controllare se l'implementazione di **fcntl()** nel proprio sistema operativo è in grado di impostare il flag O_SYNC.

La tecnica usata adesso, introdotta nella versione Beta 2 di Firebird 2.1, è quella di riaprire il file. Questo dovrebbe garantire la corretta operatività in ogni sistema operativo, premesso che la chiamata di sistema **open()** funzioni correttamente sotto questo aspetto. Apparentemente finora nessuno ha mostrato questo tipo di problemi.

Il team di sviluppo di Firebird non si riesce a spiegare perché un tale buco possa rimanere irrisolto per due anni dopo essere stato immesso nel Linux kernel's bug-tracker. Apparentemente, in Linux un problema documentato diventa una caratteristica...

### Soluzione rapida per le versioni di Firebird precedenti

Questo è un suggerimento per una soluzione rapida al problema per le versioni precedenti di Firebird: usare l'opzione «sync» su ogni partizione montata con un database Firebird installato. Un esempio di una linea in /etc/fstab:

```
/dev/sda9    /usr/database    ext3    noatime,sync    1   2
```

## Database su periferiche fisiche

A. Peshkov

Quando un database in modo Forced Writes cresce rapidamente, l'I/O del sistema di file può degradare molto le prestazioni. Su Linux, che è privo delle chiamate a sistema per far crescere in modo efficiente la dimensione del database, le prestazioni con le Forced Writes possono essere fino a tre volte più lente che con la scrittura asincrona.

In presenza di tali condizioni, le prestazioni possono essere molto migliorate scavalcando completamente il sistema operativo e recuperando il database direttamente su una periferica disco fisica. Infatti, un database Firebird può essere ricreato su un qualsiasi tipo di periferica a blocchi.

### Spostare un database su una periferica fisica

Spostare un database su una periferica fisica è semplice come recuperare una copia direttamente su una partizione non formattata nel sistema di dischi locale. Ad esempio,

```
gbak -c miacopia.fbk /dev/sda7
```

recupera la copia di nome miacopia.fbk nel terzo disco logico della partizione estesa del primo disco rigido SCSI o SATA (disk0).

> **Nota**
>
> Il database non ha altro nome che il nome della periferica stessa; nell'esempio dato il «nome del database» è proprio '/dev/sda7'.

### Gestione particolare di nbak/nbackup

Per evitare che il file differenziale del programma di utilità fisico **nbackup** venga scritto nel direttorio /dev/, bisogna attribuirgli un nome file completo in modo esplicito. Questo si ottiene usando ad esempio ISQL, attraverso il seguente comando:

```
# isql /dev/sda7
SQL> alter database add difference file '/tmp/dev_sda7';
```

Al fine di mantenere la copia fatta con nbak entro dimensioni ragionevoli, è utile sapere quanto spazio del disco è stato al momento occupato. Lo switch '-s' di nbackup riporta la dimensione del database in *pagine di database*.

```
# nbackup -s -l /dev/sda7
77173
```

Non va confuso il risultato qui dato con il numero di blocchi occupati nella periferica, e quindi la dimensione di pagina con la dimensione del blocco fisico della periferica. Il valore riportato, 77173, è il numero di pagine di database occupate. La dimensione fisica (in bytes) si calcola con (numero delle pagine * dimensione della pagina). Se non si conosce la dimensione di pagine, si può interrogare l'header del database con gstat -h:

```
 # gstat -h /dev/sda7
 Database "/dev/sda7"
 Database header page information:
        Flags               0
        Checksum            12345
        Generation          43
        Page size           4096  <———
        ODS version         11.1
  .......
```

### Esempi dell'uso di nbackup su una periferica fisica

1. Si può utilizzare uno script per fare la copia, usando direttamente lo switch '-s' per ottenere l'output. Ad esempio,

   ```
   # DbFile=/dev/sda7
   # DbSize=`nbackup -L $DbFile -S` || exit 1
   # dd if=$DbFile ibs=4k count=$DbSize | # compress and record DVD
   # nbackup -N $DbFile
   ```

2. Usando nbackup, si può fare una copia fisica direttamente col comando:

   ```
   # nbackup -B 0 /dev/sda7 /tmp/lvl.0
   ```

## Ulteriori informazioni sulle periferiche fisiche

Sebbene non siano noti problemi specifici ad oggi sull'uso delle periferiche fisiche per i database, bisogna tenere presente che

- la crescita reale e potenziale di un database fisico non è molto ovvia per un utente finale rispetto a quello che invece viene gestito dal file system di un sistema operativo. Se l'ambiente di installazione dal cliente è fuori

dalla propria diretta gestione, è necessario assicurarsi di lasciare documentazione adeguata per ogni sistema di monitoraggio che possa risultare utile

- gli esperti in Windows potrebbero provare i database su periferica fisica su Windows . Non è una delle priorità del progetto valutare come si possa ottenere su quel tipo di piattaforma, tuttavia se si crede di conoscere un modo per farlo, si prega di riferire ogni tentativo (riuscito o meno) nel proprio laboratorio windows alla lista degli sviluppatori firebird-devel.

- **Suggerimento**

  Gestite le periferiche fisiche con aliases.conf. In quel modo, avendo la necessità di riconfigurare l'hardware, non è necessario riconfigurare tutte le stringhe di connessione in tutto il codice.

## Miglioramenti nel protocollo di connessione remota

V. Khorsun, D. Yemanov

Richiesta di miglioramento CORE-971

Il protocollo remoto è stato modificato per non rallentare in presenza di reti a basse prestazioni, ed, in generale, migliorato. Per ottenere ciò, si effettua una maggior raccolta dei pacchetti, insieme alla ottimizzazione della trasmissione usando un po' di memoria tampone. Queste modifiche hanno dimostrato con test reali circa il 50% in meno di API roundtrips, che pertanto sono collegate a circa il 40% in meno di TCP roundtrips.

In Firebird 2.1 l'interfaccia remota limita la dimensione del pacchetto usato in risposta alle varie chiamate isc_XXX_info alla dimensione realmente usata dai dati contenuti, mentre prima reinviava tutto il pacchetto intero al client, anche se solo 10 bytes erano effettivamente riempiti. In quest'ultimo caso l'interfaccia remota di Firebird 2.1 invia un pacchetto di soli 10 bytes.

Alcuni utenti potrenno vedere benefici da queste modifiche, specialmente con client connessi attraverso Internet con applicazioni two-tier.

I cambiamenti possono essere riassunti con

a. Spedizione accorpata dei pacchetti. Richiede che sia il server che il client siano di versione 2.1, e si abilita a fronte di accordo positivo sul protocollo. Alcuni pacchetti di certi tipi sono ritardati accorpandoli col pacchetto successivo. Ad esempio, in questa categoria di operazioni ricadono gli statement di allocate e deallocate.)

b. recupero anticipato di parti di informazione di statement o altre richieste e mantenedole nel client per una (probabile) successiva chiamata API. Questo è implementato solo dalla parte client, ma beneficia dei ridotti round trips descritti in (a) su cui fa affidamento.

   Funziona con ogni versione di server, dando un discreto beneficio anche ad applicazioni scritte male, sebbene il meglio non è previsto venga dato con server precedenti alla versione 2.1.

c. Dimensione ridotta delle risposte dal server (eliminati gli zero finali). Poiché questa parte è solo sul server, richiede che sia una versione 2.1 ed un qualsiasi client. Anche i client vecchi pertanto funzionano con i server Firebird 2.1 possono godere di alcuni benefici dalla riduzione dei round trips, sebbene queste vecchie interfaccie, a differenza delle nuove, rispondono sempre con grossi pacchetti per la isc_dsql_prepare().

d. Un altro risparmio nel round-trip, chiamato «defer execute» (esecuzione ritardata), in cui le richieste SELECT vengono trattenute dal momento immediatamente precedente alla `isc_dsql_execute` fino alla

chiamata API dello stesso statement. Il beneficio diventa visibile sopratutto quando c'è una gran quantità di richieste SELECT il cui risultato riempie uno o due pacchetti di rete.

Questo funziona solo se il client ed il server sono v.2.1 o superiore.

> **Nota**
>
> Un possibile effetto collaterale potrebbe avvenire se fallisse l'`isc_dsql_execute` con una eccezione, tale eccezione viene allora riportata al client in risposta alla chiamata API che viene *in quel momento* eseguita; in altre parole, invece di essere riportato l'errore dalla `isc_dsql_execute` verrebbe riportato da `isc_dsql_fetch`, `isc_dsql_info`, o qualsiasi altra chiamata API attualmente richiesta dalla chiamata `op_execute`.
>
> Nella maggior parte dei casi, l'effetto collaterale sarebbe trasparente: potrebbe mostrarsi nel caso in cui un qualche errore avvenga con valori default per parametri o variabili PSQL e verrebbe allora evidenziato come un vettore di eccezioni dove le eccezioni sono inviate in un'insolita sequenza.

Le modifiche funzionano sia con TCP/IP che con NetBEUI. Sono compatibili con le versioni precedenti, pertanto le applicazioni esistenti restano funzionali. Tuttavia, usando un strato software che implementa una propria versione del protocollo remoto, come Jaybird JDBC driver ed il FirebirdClient .NET driver, il codice esistente non è in grado di abilitare questi miglioramenti a meno di non aggiornare anche gli strati software intermedi.

# Modifiche alle API

## *XSQLVAR*

A. dos Santos Fernandes

Per passare l'identificatore del set di caratteri della connessione oppure il set di caratteri di un BLOB, quando il set di caratteri della connessione è NONE, si usa l'elemento `XSQLVAR::sqlscale` dei BLOB di tipo testo.

# Ottimizzazione

## *Ottimizzazione per scansioni multiple degli indici*

V. Khorsun

Caratteristica richiesta CORE-1069

Nel caso in cui più di un indice deve essere letto a causa di congiunzioni AND, è stata operata una ottimizzazione.

## *Ottimizzate varie operazioni bitmap*

V. Khorsun

Caratteristica richiesta CORE-1070

Nel caso in cui i valori sono per lo più consecutivi, sono state ottimizzate le operazioni bitmap (set, test e clear).

# Configurazione e regolazione

## Aumentati i limiti ed i default del Lock Manager

D. Yemanov

Caratteristiche richieste CORE-958 e CORE-937

- è stato aumentato il **massimo numero di hash slot** da 2048 a 65,536. Poiché il valore da usare deve essere un numero primo, il massimo supportato è 65,521 (il più grande numero primo inferiore a 65,536). Il valore minimo è 101.
- il nuovo **numero di default per gli hash slots** è stato impostato a 1009
- la **dimensione della tabella di lock** di default è stata aumentata a 1 Mb su tutte le piattaforme

## Sconsigliate le dimensioni di pagina di 1K e 2K

D. Yemanov

Caratteristica richiesta CORE-969

Sono sconsigliate le dimensioni di pagina da 1K e da 2K perché sono inefficienti.

> **Nota**
>
> Questa restrizione alle pagine piccole si applica ai database nuovi (ODS 11 e oltre). Ci si può collegare alle vecchie basi di dati indipendentemente dalla loro dimensione di pagina.

## Problemi in allocazione su disco

V. Khorsun

Caratteristica richiesta CORE-1229

Fino a Firebird 2.1, non essendoci regole particolari nell'allocare lo spazio per le pagine dei file di database, a causa delle dipendenze delle pagine che gestisce autonomamente, per servire la strategia di «careful write», ha sempre scritto le nuove pagine appena allocate in ordine qualsiasi.

Per i database con ODS 11.1 e successive, i server Firebird a partire dalla versione 2.1 usano un algoritmo per allocare lo spazio su disco, per risolvere due problemi riconosciuti associati al sistema precedente:

1. **Disastri provocati da condizioni di disco pieno**

   Nel momento in cui la memoria tampone ha tante pagine «sporche» (dirty pages) e Firebird ha necessità di chiedere lo spazio per una nuova pagina per scriverle, se non c'è abbastanza spazio su disco per soddisfare la richiesta, ci sono problemi in presenza di un ordine non determinato delle scritture. Infatti, in tale situazione,

succede spesso che l'amministratore del sistema decida di scollegare il database per dargli in qualche modo più spazio su disco, ma in tal modo le pagine sporche ancora nella memoria tampone vanno perse. Questo comporta seri disastri alla struttura del database.

2. **Frammentazione del file**

   Allocando lo spazio su disco a pezzettini relativamente piccoli, si ha una significativa frammentazione del file di database al livello del file system, peggiorando le prestazioni delle scansioni grandi, come per esempio quelle di un backup.

## La soluzione

Per risolvere il problema sono state introdotte alcune regole e razionalizzazioni per gestire l'allocazione delle nuove pagine in funzione dello spazio disponibile.

a. Si scrive su disco ogni nuova pagina allocata immediatamente prima di tornare al motore. Se la pagina non può essere scritta, l'allocazione non avviene: il bit PIP non viene cancellato e si segnala l'appropriato errore di I/O. Il dataase non si rovina poiché è garantito che tutte le pagine sporche nella memoria hanno spazio su disco allocato e possono essere pertanto scritte senza errori.

   Poiché questa modifica aggiunge una scrittura in più per ogni pagina allocata, ci sarebbe un peggioramento nelle prestazioni. Invece le pagine vengono allocate in blocchi fino a 128 Kb e Firebird tiene traccia del numero di queste pagine così «inizializzate» nel PIP header.

   > **Nota**
   >
   > Una pagina che è stata allocata, rilasciata e sucessivamente riallocata è spazio «riservato», nel senso che non è necessaria nessuna verifica ulteriore per inizializzarla. Pertanto una pagina da allocare è soggetta alla doppia scrittura se e solo se non è mai stata allocata prima.

b. Per risolvere la frammentazione del file, Firebird ora usa le chiamate alle API del file system per *preallocare* lo spazio su disco a «pezzettoni».

   La preallocazione permette di evitare disastri in caso di «disco pieno». Ci sono buone possibilità che il database abbia abbastanza spazio preallocato per terminare le operazioni per dare all'amministratore il tempo di creare più spazio su disco.

   > **Importante**
   >
   > **Per Windows solamente (per il momento)**
   >
   > Attualmente, solo Windows ha pubbliche tali chiamate API, pertanto per il momento questoaspetto è supportato pienamente solo nella versione Windows di Firebird. Tuttavia, nelle versioni più recenti, sono state aggiunte simili utilità alla Linux API, facendo ritenere che in futuro tali funzioni saranno disponibili per un file system popolare come *ext3*.

## Il parametro di configurazione DatabaseGrowthIncrement

Per un miglior controllo della preallocazione dello spazio su disco, è stato aggiunto a `firebird.conf` il nuovo parametro *DatabaseGrowthIncrement*. Rappresenta il limite superiore della dimensione del blocco preallocato in bytes.

> **Importante**
>
> Si prega di leggere attentamente i dettagli riguardo tale configurazione, in DatabaseGrowthIncrement nel capitolo relativo ai «Nuovi parametri e modifiche alla configurazione».

### *Neutralizzare la cache del filesystem su Superserver*

V. Khorsun

Caratteristica richiesta CORE-1381 e CORE-1480

Per la memoria tampone delle pagine, Firebird usa e gestisce un suo proprio sistema di cache in memoria. Il sistema operativo, a sua volta, potrebbe avere parte degli stessi dati gestiti nella memoria tampone propria del filesystem. Configurando Firebird ad usare una memoria cache relativamente grande rispetto alla memoria disponibile e ad usare le Forced Writes, questa duplicazione della memoria cache può consumare così tante risorse disponibili da non dare nessun beneficio.

Inoltre, se il sistema operativo cerca di gestire un file grandissimo, può spostare la cache delle pagine di Firebird addirittura in swap, provocando un'intensa (e inutile) paginazione su disco. In pratica, se la dimensione della cache delle pagine di Firebird Superserver è oltre l'80% della RAM disponibile, la gestione delle risorse potrebbe risultare insostenibile.

> **Nota**
>
> La cache del filesystem è utile nelle scritture su disco, ma solo con Forced Writes disabilitate (OFF), che non è raccomandato nella maggior parte delle situazioni.

Adesso Superserver, sia su Windows che POSIX, ha un nuovo parametro di configurazione, **MaxFileSystemCache**, per (dis)abilitare la cache del filesystem. Può permettere di avere più memoria libera per altre operazioni (es. l'ordinamento) e, dove ci sono molti database, può ridurre le richieste di risorse all'host.

> **Nota**
>
> Per la versione Classic non c'è scappatoia.

Per dettagli sul parametro **MaxFileSystemCache**, vedere MaxFileSystemCache.

## Altri miglioramenti globali

### *Razionalizzazione della Garbage Collection*

V. Khorsun

Caratteristica richiesta CORE-1071

Il processo di raccolta in background leggeva tutte le versioni precedenti dei record su una pagina, comprese quelle create da transazioni attive. Poiché queste ultime non vanno considrate per la garbage collection, era una perdita di tempo leggerle.

## Immediato rilascio dei file esterni

V. Khorsun

Caratteristica richiesta CORE- 961

Il motore ora rilascia i file esterni non appena sono rilasciate da qualsiasi richiesta degli utenti.

## Sincronizzazione degli oggetti nella cache metadata DSQL per il Classic server

A. dos Santos Fernandes

Richiesta CORE-976

Dettagli assenti.

## Miglioramenti sui BLOB

A. dos Santos Fernandes

Richiesta CORE-1169

La conversione tra il blob temporanea ed il tipo di blob destinazione avviene solo alla materializzazione e non prima.

## Flag di tipo per le Stored Procedures

D. Yemanov

Richiesta CORE-779

È stato introdotto un flag di tipo per le stored procedure, aggiungendo la colonna RDB$PROCEDURE_TYPE alla tabella RDB$PROCEDURES. I possibili valori sono:

*- 0 o NULL -*
procedura compatibile (non viene effettuato nessun controllo)

*- 1 -*
SELECT procedure (una procedura che contiene almeno uno statement SUSPEND)

*- 2 -*
EXECUTE procedure (non ci sono statement SUSPEND, e pertanto non è possibile usarla in una SELECT)

## Informazioni utili per il Core Dump su Linux

A. Peshkov

Richiesta CORE-1558

Il server, normalmente, dopo un problema grave (bugcheck) cerca di proseguire le operazioni. Il parametro di configurazione BugcheckAbort fa in modo invece di chiamare immediatamente la funzione **abort()** e creare un file di core dump. Poiché il bugcheck è il risultato di un problema non riconosciuto dal server, continuare le operazioni con un problema non risolto non dovrebbe essere comunque possibile, id il file di core dump può dare informazioni utili per il debug.

Nelle distribuzioni più recenti di Linux, quando un'applicazione abortisce per default non viene più fatto il core dump core. Gli utenti hanno spesso problemi nell'ottenerlo. Le regole diverse tra Classic e Superserver, oltre all'inconsistenza tra gli strumenti di gestione del sistema operativo tra ditribuzione e distribuzione, rendono pressoché impossibile aiutare l'utente generico con una qualsiasi «regola generale».

Pertanto è stato aggiunto, al Classic e al Superserver su Linux, codice per neutralizzare questi problemi ed automatizzare la generazione di un file di core dump all'abort() causato da BUGCHECK. Il server Firebird esegue il necessario 'cwd' (change working directory) ad una locazione scrivibile (/tmp) e imposta la dimensione limite del file di core in modo tale che il limite 'soft' eguagli il limite 'hard'.

> **Nota**
>
> Nelle versioni normali, cioè «release», il sistema automatico di core dump viene attivato solo se in `firebird.conf` viene impostato a true il parametro BugcheckAbort. Nelle versioni «debug», è sempre attivo.
>
> Per (dis)abilitare questo sistema, non va dimenticato che il server Firebird va fatto ripartire, così come ad ogni modifica dei parametri.

Capitolo 5

# Novità sul Data Definition Language (DDL)

In questo capitolo sono elencati le aggiunte ed i miglioramenti aggiunti alla parte relativa al linguaggio di definizione dei dati SQL (DDL) durante lo sviluppo di Firebird 2. Quelli evidenziati come introdotti nella versione 2.1 sono disponibili solo nei database con ODS 11.1 e successive.

## Ricerca rapida



### *Trigger a livello database*

Adriano dos Santos Fernandes

**(v.2.1)** Un *trigger di database* è un modulo scritto in PSQL che viene eseguito in associazione ad un evento di connessione o di transazione. Gli eventi e relativo ordine sono come segue.-

*`CONNECT`*

- Viene stabilita una connessione ad un database
- parte una transazione
- vengono eseguiti i trigger, le eccezioni non gestite fanno rollback della transazione e disconnettono il client e sono riportate al client
- la transazione è confermata

*DISCONNECT*

- Parte una transazione
- Vengono eseguiti i trigger, le eccezioni non gestite fanno rollback della transazione e disconnettono la connessione e sono ignorate
- La transazione è confermata
- Si viene scollegati

*TRANSACTION START*
: I trigger sono eseguiti alla creazione della nuova transazione; le eccezioni non gestite sono riportate al client e fa rollback della transazione.

*TRANSACTION COMMIT*
: I trigger sono eseguiti alla conferma della transazione; le eccezioni non gestite fanno rollback del punto di salvataggio del trigger, la conferma è abortita e l'eccezione riportata al client.

> **Nota**
>
> Per le transazioni «two-phase», i trigger vengono lanciati in fase di «preparazione», non in fase di conferma.

*TRANSACTION ROLLBACK*
: I trigger sono eseguiti durante il rollback della transazione. Tutte le modifiche sono ovviamente cancellate insieme al resto della transazione. Le eccezioni non gestite sono ignorate

**Sintassi**

```
<database-trigger> ::=
  {CREATE | RECREATE | CREATE OR ALTER}
    TRIGGER <name>
    [ACTIVE | INACTIVE]
    ON <event>
    [POSITION <n>]
  AS
    BEGIN
      ...
    END

<event> ::=
  CONNECT
    | DISCONNECT
    | TRANSACTION START
    | TRANSACTION COMMIT
    | TRANSACTION ROLLBACK
```

**Regole e limitazioni**

1. Il tipo di un trigger di database non può essere cambiato.
2. Il permesso per i comandi CREATE, RECREATE, CREATE OR ALTER, o DROP su un trigger di database è limitato al proprietario del database e a SYSDBA.

### Supporto ai trigger di database nei programmi di utilità

Sono stati aggiunti nuovi parametri a gbak, nbackup e isql per inibire l'esecuzione dei trigger di database. Sono disponibili solo al proprietario del database e a SYSDBA:

```
gbak -nodbtriggers
isql -nodbtriggers
nbackup -T
```

## Tabelle temporanee globali (Global Temporary Tables)

Vlad Khorsun

**(v.2.1)** Le tabelle temporanee globali (GTT) sono tabelle memorizzate permanentemente nel metadata del sistema, ma con dati temporanei. I dati provenienti da differenti connessioni (o transazioni in funzione della definizione) sono isolati gli uni dagli altri, ma il metadata della GTT è condiviso fra tutte le connessioni e transazioni.

Pertanto ci sono due tipi di GTT:

- con dati persistenti per tutta la durata della connessione un cui è riferita la specifica GTT
- con dati persistenti per tutta la durata della transazione in cui si è utilizzata la GTT

### Sintassi e regole per le GTT

```
CREATE GLOBAL TEMPORARY TABLE
  ...
  [ON COMMIT <DELETE | PRESERVE> ROWS]
```

Questo crea il metadata della tabella temporanea nel sistema.

La clausola ON COMMIT imposta il tipo della tabella temporanea:

*ON COMMIT PRESERVE ROWS*
: tutti i dati lasciati nella tabella alla fine della transazione restano a disposizione nel database fino al termine della connessione.

*ON COMMIT DELETE ROWS*
: tutti i dati nella tabella data sono eliminati dal database immediatamente alla fine della transazione. ON COMMIT DELETE ROWS è il default se non si usa la clausola opzionale ON COMMIT.

*CREATE GLOBAL TEMPORARY TABLE*
: è un comando DDL che è processato dal motore come una normale CREATE TABLE. Allo stesso modo, non è possibile la CREATE, o la DROP di una GTT all'interno di un trigger o di una procedura.

**Tipo della relazione**

Le definizione delle GTT sono distinte nel catalogo di sistema l'una dall'altra e dalle tabelle permanenti, tramite il valore del campo `RDB$RELATIONS.RDB$RELATION_TYPE`:

- Una GTT con l'opzione ON COMMIT PRESERVE ROWS ha RDB$RELATION_TYPE = 4

- Una GTT con l'opzione ON COMMIT DELETE ROWS ha RDB$RELATION_TYPE = 5.

> **Nota**
>
> Per la completa lista dei valori, vedere RDB$TYPES.

**Supporto alle caratteristiche del linguaggio**

Le stesse funzionalità che si possono utilizzare con le tabelle standard (indici, trigger, vincoli a livello di campo e di tabella) sono disponibili anche nelle GTT, con certe limitazioni sulle correlazioni tra GTT e tabelle normali.-

a. sono proibiti riferimenti tra GTT e tabelle regolari

b. Una GTT con ON COMMIT PRESERVE ROWS non può avere riferimenti con una GTT con ON COMMIT DELETE ROWS

c. Un vincolo di dominio non può riferirsi a nessuna GTT.

### Note di implementazione

Un'istanza di una GTT, come insieme di record di dati creati e visibili all'interno una data transazione o connessione, è creata quando la GTT è utilizzata per la prima volta, di solito al momento di una «prepare». Ogni istanza ha il suo insieme privato di pagine nel quale sono memorizzate i dati e gli indici. I record e gli indici hanno la stessa struttura fisica delle tabelle permanenti.

Al termine della connessione o transazione, tutte le pagine dell'istanza di una GTT sono rilasciate immediatamente. Assomiglia a quanto accade quando si esegue una DROP TABLE, eccetto che la definizione del metadata non si perde, naturalmente. Questa però è molto più veloce della cancellazione tradizionale fatta record per record e successiva garbage collection delle versioni dei record deletati.

> **Nota**
>
> Questa cancellazione non fa scattare i trigger di DELETE, pertanto non siate tentati di definire trigger di Before o After Delete ritenendo di poter ottenere una qualche esecuzione «post mortem» dei vostri dati!

I dati e le pagine indice di tutte le istanze delle GTT sono messe in file temporanei separati. Ogni connessione ha il suo file temporaneo creato nel momento in cui la connessione fa riferimento a una qualche GTT.

> **Nota**
>
> Questi files temporanei sono sempre aperti con Forced Writes = OFF, indipendentemente dall'impostazione delle Forced Writes.

Non c'è nessun limite al numero di istanze di GTT che possono coesistere contemporaneamente. Avendo N transazioni attive simultaneamente e se ogni transazione ha un riferimento ad una certa GTT allora ci saranno N istanze della GTT.

## Miglioramenti alle viste

D. Yemanov

In Firebird 2.1 sono stati fatti un paio di miglioramenti alle definizioni delle viste.-

### *Uso degli alias di colonna in CREATE VIEW*

Caratteristica richiesta CORE-831

**(v.2.1)** Gli alias di colonna possono essere utilizzati come nomi di colonna nelle definizioni di una vista.

**Esempio**

```
CREATE VIEW V_TEST AS
  SELECT ID,
         COL1 AS CODICE,
         COL2 AS NOME
  FROM TABELLA;
```

## *CREATE TRIGGER in accordo a SQL2003*

A. dos Santos Fernandes

Richiesta CORE-711

**(v.2.1)** È disponibile una sintassi alternativa per il CREATE TRIGGER che è in accordo a SQL2003.

**Schemi sintattici**

Forma esistente

```
create trigger t1
  FOR atable
  [active] before insert or update
as
  begin
    ...
  end
```

Forma SQL2003

```
create trigger t2
  [active] before insert or update
  ON atable
as
  begin
    ...
  end
```

Notare la posizione differente della clausola identificante la tabella e le differenti parole chiave che precedono il nome della tabella (esistente FOR; SQL2003: ON).

Entrambe le sintassi sono valide e disponibili per tutti i CREATE TRIGGER, RECREATE TRIGGER e CREATE OR ALTER TRIGGER.

## Alternativa conforme a SQL2003 per i campi calcolati

D. Yemanov

Richiesta CORE-1386

**(v.2.1)** Per definire un campo calcolato in una CREATE TABLE o ALTER TABLE, è stata resa disponibile la sintassi alternativa **GENERATED ALWAYS AS** conforme a SQL.

**Sintassi**

```
<nome colonna> [<tipo>] GENERATED ALWAYS AS ( <espressione> )
```

è completamente equivalente alla forma compatibile con le versioni precedenti:

```
<nome colonna> [<tipo>] COMPUTED [BY] ( <espressione> )
```

**Esempio**

```
CREATE TABLE T (PK INT, EXPR GENERATED ALWAYS AS (PK + 1))
```

## CREATE SEQUENCE

D. Yemanov

È stata introdotta la parola chiave SEQUENCE come sinonimo di GENERATOR, in accordo con lo standard SQL-99. SEQUENCE è un termine descritto nelle specifiche SQL, mentre GENERATOR è un termine compatibile InterBase. L'uso della sintassi standard SEQUENCE è raccomandata nelle applicazioni.

Un generatore di sequenza è un meccanismo per generare valori numerici interi incrementali, uno alla volta. Un generatore di sequenza è un oggetto dello schema con nome, che in dialetto 3 è BIGINT, mentre in dialetto 1 è INTEGER.

**Sintassi**

```
CREATE { SEQUENCE | GENERATOR } <nome>
DROP { SEQUENCE | GENERATOR } <nome>
SET GENERATOR <nome> TO <valore_partenza>
ALTER SEQUENCE <nome> RESTART WITH <valore_partenza>
GEN_ID (<nome>, <valore_di_incremento>)
NEXT VALUE FOR <nome>
```

**Esempi**

1.

```
CREATE SEQUENCE S_EMPLOYEE;
```

2.

```
ALTER SEQUENCE S_EMPLOYEE RESTART WITH 0;
```

Vedere anche le note su NEXT VALUE FOR.

> **Avvertimento**
>
> ALTER SEQUENCE, come SET GENERATOR, è un bel modo per rovinare la generazione dei valori delle chiavi!

## REVOKE ADMIN OPTION

D. Yemanov

SYSDBA, il creatore del database ed il proprietario di un oggetto possono rilasciare i diritti su quell'oggetto ad altri utenti. Inoltre, tali diritti possono essere resi ereditabili con la clausola **WITH GRANT OPTION**. In tal caso il garante dà al garantito i diritti di diventare a sua volta garante per lo stesso oggetto. Il garante originale può rimuovere questa concessione al garantito con il comando **REVOKE GRANT OPTION FROM user**.

Tuttavia, c'è una forma secondaria che coinvolge i ruoli. Invece di specificare gli stessi diritti per diversi utenti (che presto diventa un incubo nella manutenzione), si può creare un ruolo, assegnare a questo nuovo ruolo un insieme di diritti ed infine garantire quel ruolo ad uno o più utenti. Ogni modifica ai diritti del ruolo influenzerà i diritti di tutti gli utenti a cui è stato garantito quel ruolo. Si dice anche che tutti gli utenti «appartenenti» a quel ruolo hanno i medesimi diritti.

Simmetricamente, con la clausola **WITH ADMIN OPTION**, il garante (di solito il creatore del ruolo) attribuisce ai garantiti la possibilità di diventare a loro volta garanti per lo stesso ruolo. Fino a FB v2, questa abilità non poteva essere rimossa senza mettere mano direttamente alle tabelle di sistema. Ora, il garante originale può rimuovere la concessione ad attribuire le garanzie del ruolo ad altri utenti con il comando **REVOKE ADMIN OPTION FROM user**.

## Clausole di SET/DROP DEFAULT per il comando ALTER TABLE

C. Valderrama

I domini permettono di modificare o rimuovere i loro default. Sembra naturale pertanto che anche i campi di tabella possano essere maneggiati allo stesso modo senza manipolazione delle tabelle di sistema.

**Sintassi**

```
ALTER TABLE t ALTER [COLUMN] c SET DEFAULT default_value;
ALTER TABLE t ALTER [COLUMN] c DROP DEFAULT;
```

> **Nota**
>
> - I campi di tipo array non possono avere un valore di default.
> - Cambiando il tipo del dato, si rischia di lasciare il suo valore di default precedente. Questo perché ad un campo può essere assegnato il tipo di un dominio che ha un default ma il campo ha la possibilità di sovrascrivere tale dominio. D'altra parte ad un campo si può dare direttamente un nuovo tipo di dato nel qual caso il default appartiene logicamente al campo ed è mantenuto in un dominio creato e gestito in automatico dietro le quinte.

## Sintassi per modificare le eccezioni

D. Yemanov

I comandi di DDL **RECREATE EXCEPTION** e **CREATE OR ALTER EXCEPTION** (richiesta SF #1167973) sono stati implementati permettendo di ricreare, oppure di creare o modificare una eccezione definita dall'utente, in funzione della sua attuale esistenza.

### RECREATE EXCEPTION

**RECREATE EXCEPTION** si comporta esattamente come **CREATE EXCEPTION** se l'eccezione non esiste ancora. Se esiste già la sua definizione viene completamente sostituita, a meno che non vi siano dipendenze che blocchino la modifica.

### CREATE OR ALTER EXCEPTION

**CREATE OR ALTER EXCEPTION** crea un'eccezione se non esiste già, oppure la modifica, senza alcun effetto sulle sue dipendenze.

## ALTER EXTERNAL FUNCTION

C. Valderrama

È stata implementato il comando **ALTER EXTERNAL FUNCTION** in modo da permettere la modifica dell'`entry_point` o del `module_name` della dichiarazione di una UDF, quando questa non può essere rimossa a causa di dipendenze attualmente insistenti sulla sua definizione.

## COMMENT ON

C. Valderrama

Il comando **COMMENT ON** è stato implementato per permettere l'inserimento di descrizioni e commenti nel metadata.

**Syntax Pattern**

```
COMMENT ON DATABASE IS {'txt'|NULL};
COMMENT ON <basic_type> name IS {'txt'|NULL};
COMMENT ON COLUMN tblviewname.fieldname IS {'txt'|NULL};
COMMENT ON PARAMETER procname.parname IS {'txt'|NULL};
```

Una stringa vuota di commento, cioè '', equivale a NULL in quanto il codice interno (nel caso DYN) funziona in quel modo per i blob.

```
<basic_type>:
   DOMAIN
   TABLE
   VIEW
   PROCEDURE
```

```
TRIGGER
EXTERNAL FUNCTION
FILTER
EXCEPTION
GENERATOR
SEQUENCE
INDEX
ROLE
CHARACTER SET
COLLATION
SECURITY CLASS[1]
```

[1]non implementato in quanto il tipo è nascosto.

## Estensioni alle specifiche di CREATE VIEW

D. Yemanov

Nelle specifiche di una vista si possono usare adesso anche le sintassi relative a FIRST/SKIP e ROWS e le clausole PLAN e ORDER BY.

A partire da Firebird 2.0 in poi, le viste sono trattate come espressioni di **SELECT** complete di tutte le caratteristiche. Di conseguenza nelle viste adesso le clausole relative a FIRST/SKIP, ROWS, UNION, ORDER BY e PLAN sono permesse e funzionano come previsto.

**Sintassi**

Per i dettagli della sintassi, fare riferimento a SELECT e sintassi delle espressioni nel capitolo relativo al DML.

## Implementata la frase RECREATE TRIGGER

D. Yemanov

La frase di DDL per ricreare un trigger è disponibile nella forma RECREATE TRIGGER. La sintassi è simile a quella di altre frasi di RECREATE.

## Migliorie nell'uso

Le modifiche descritte in seguito influenzano in vario modo l'utilizzo o l'esistenza di alcuni aggiramenti tecnici nelle applicazioni o nelle basi di dati delle versioni precedenti a Firebird 2.

### L'accesso esclusivo non è più richiesto per creare vincoli di chiave esterna

V. Horsun

Ora si possono creare i vincoli di chiave esterna (foreign key constraints) senza la necessità di ottenere un accesso esclusivo all'intero database.

### Logica modificata per gli aggiornamenti delle viste

Si consiglia di applicare i vincoli di NOT NULL soltanto alle tabelle di base, e di ignorare quelli che le colonne della vista ereditano dalle definizioni del dominio.

### *Identificatori descrittivi per i sottotipi dei BLOB*

A. Peshkov, C. Valderrama

Fino alla versione di Firebird 2.0 per dichiarare un filtro per i blob, l'unica sintassi possibile era:

```
declare filter <name> input_type <number> output_type <number>
  entry_point <function_in_library> module_name <library_name>;
```

In alternativa adesso c'è questa nuova sintassi:

```
declare filter <name> input_type <mnemonic> output_type <mnemonic>
  entry_point <function_in_library> module_name <library_name>;
```

dove <mnemonic> si riferisce ad un identificatore del sottotipo riconosciuto dal motore.

Attualmente essi sono binary, text ed altri sopratutto di esclusivo uso interno, ma un utente con particolari necessità potrebbe aggiungere un nuovo codice mnemonico in rdb$types ed usarlo, poiché è controllato solo al momento della dichiarazione. Il motore prende e memorizza poi il solo valore numerico associato. Va ricordato che i valori definibili dall'utente per i codici mnemonici dei sottotipi blob possono essere soltanto negativi.

Per ottenere la lista dei tipi predefiniti, lanciare:

```
select RDB$TYPE, RDB$TYPE_NAME, RDB$SYSTEM_FLAG
     from rdb$types
     where rdb$field_name = 'RDB$FIELD_SUB_TYPE';

   RDB$TYPE   RDB$TYPE_NAME                RDB$SYSTEM_FLAG
   ========= ============================ =================

           0 BINARY                                       1
           1 TEXT                                         1
           2 BLR                                          1
           3 ACL                                          1
           4 RANGES                                       1
           5 SUMMARY                                      1
           6 FORMAT                                       1
           7 TRANSACTION_DESCRIPTION                      1
           8 EXTERNAL_FILE_DESCRIPTION                    1
```

**Esempi**

Dichiarazione originale:

```
declare filter pesh input_type 0 output_type 3
  entry_point 'f' module_name 'p';
```

Dichiarazione alternativa:

```
declare filter pesh input_type binary output_type acl
  entry_point 'f' module_name 'p';
```

Per dichiarare un nome per un sottotipo di blob definito dall'utente è necessario ricordarsi di effettuare il **COMMIT** dopo l'inserzione che viene effettuata come nell'esempio seguente da comandi ISQL:

```
SQL> insert into rdb$types
CON> values('RDB$FIELD_SUB_TYPE', -100, 'XDR', 'test type', 0);
SQL> commit;
SQL> declare filter pesh2 input_type xdr output_type text
CON> entry_point 'p2' module_name 'p';
SQL> show filter pesh2;
BLOB Filter: PESH2
        Input subtype: -100 Output subtype: 1
        Filter library is p
        Entry point is p2
```

Capitolo 6

# Novità nel Data Manipulation Language (DML)

In questa parte si tratta delle novità e dei miglioramenti aggiunti a quella parte del linguaggio SQL dedicato alla manipolazione dei dati durante lo sviluppo di Firebird 2. Quelli identificati come introdotti nella versione 2.1 sono disponibili solo con database in ODS 11.1 e superiori.

> **Importante**
>
> Un nuovo parametro di configurazione, *RelaxedAliasChecking*, è stato aggiunto a **firebird.conf** in Firebird 2.1 al fine di permettere una certa compatibilità con le versioni precedenti, rilassando le restrizioni di Firebird 2.0.x quando ci sono nomi di tabelle ed alias multipli nelle query (vedere In DSQL la scansione dei nomi di tabella è più rigida, più oltre).
>
> Questo parametro non sarà una caratteristica permanente di Firebird ma ha lo scopo di permettere la migrazione per coloro che hanno necessità di sistemare il codice esistente. Altre informazioni sono in RelaxedAliasChecking nel capitolo relativo ai «Nuovi parametri di configurazione».

## Ricerca rapida

- UNION DISTINCT
- Coercizione del tipo di dato nelle UNION migliorata
- UNION permesse nelle subquery ANY/ALL/IN
- Nuovo predicato [NOT] DISTINCT
- Ammorbidita la regola per comparare i NULL
- Modifiche all'ordinamento dei NULL
- Insiemi UNION nelle subquery
- Nuove estensioni ad UPDATE e DELETE
- Estensioni alle variabili di contesto
- Miglioramenti ai PLAN
- GROUP BY e ORDER BY con alias
- GROUP BY su espressione
- Ordinamento di SELECT * per numero

  ORDER BY numerici e parametri: attenzione!
- NEXT VALUE FOR
- **Articoli**
  1. SELECT e sintassi delle espressioni
  2. Tipo del dato risultante da un'aggregazione
  3. la sezione chiamata «La sintassi abbreviata: trucchetto per i letterali di data»

## Common Table Expressions

Vlad Khorsun
Basato sul lavoro di Paul Ruizendaal per il progetto Fyracle

**(v.2.1)** Una *espressione di tabella comune*, in inglese *common table expression* o più brevemente `CTE` è come una vista definita localmente all'interno di una query principale. Il motore tratta una CTE come una tabella derivata e non viene effettuata nessuna materializzazione intermedia dei dati.

### Benefici delle CTE

L'uso delle CTEs permette di specificare query dinamioche che possono essere anche recursive:

- Il motore inizia l'esecuzione a partire da un elemento non recursivo.
- Per ogni riga valutata, esegue ciascun membro recursivo uno alla volta, usando i valori attuali della riga esterna come parametri.
- Se l'istanza attualmente in esecuzione di un membro recursivo non produce righe, l'esecuzione risale di un livello e recupera la riga successiva dalle righe dell'elemento esterno.

Nel caso di una CTE recursiva il carico di lavoro per la memoria e la CPU è molto inferiore di quello di una equivalente STORED PROCEDURE recursiva.

### Limiti nella recursione

Attualmente la profondità massima della recursione è fissato nel codice a 1024 livelli.

### Sintassi e regole per le CTE

```
select :
  select_expr for_update_clause lock_clause
select_expr :
  with_clause select_expr_body order_clause rows_clause
            | select_expr_body order_clause rows_clause
with_clause :
  WITH RECURSIVE with_list | WITH with_list
with_list :
  with_item | with_item ',' with_list
with_item :
  symbol_table_alias_name derived_column_list
    AS '(' select_expr ')'
select_expr_body :
  query_term
  | select_expr_body UNION distinct_noise query_term
              | select_expr_body UNION ALL query_term
```

E con una rappresentazione meno formale:

```
WITH [RECURSIVE]
 CTE_A [(a1, a2, …)]
 AS ( SELECT … ),

 CTE_B [(b1, b2, …)]
 AS ( SELECT … ),
...
SELECT ...
  FROM CTE_A, CTE_B, TAB1, TAB2 ...
 WHERE ...
```

### Regole per le CTE non recursive

- Si possono definire in una unica query più di una espressione
- Qualsiasi clausola legale in una SELECT è legale in una CTE
- Una CTE può fare riferimento ad altre CTE
- I riferimenti fra varie CTE non possono contenere circolarità
- Le CTE possono essere usate in ogni parte della query principale o di un'altra CTE
- Una stessa CTE può essere usata più di una volta nella query principale
- Le CTE possono essere usate nelle frasi tipo INSERT, UPDATE o DELETE come subquery
- Le CTE sono legali nelle STORED PROCEDURE e nei TRIGGER (in generale nel PSQL)
- Una clausola WITH non può essere interna ad un'altra clausola WITH (cioè non possono essere innestate)

**Esempio di una CTE non recursiva**

```
WITH
  DEPT_YEAR_BUDGET AS (
    SELECT FISCAL_YEAR, DEPT_NO,
        SUM(PROJECTED_BUDGET) AS BUDGET
      FROM PROJ_DEPT_BUDGET
    GROUP BY FISCAL_YEAR, DEPT_NO
  )
SELECT D.DEPT_NO, D.DEPARTMENT,
  B_1993.BUDGET AS B_1993, B_1994.BUDGET AS B_1994,
       B_1995.BUDGET AS B_1995, B_1996.BUDGET AS B_1996
  FROM DEPARTMENT D
    LEFT JOIN DEPT_YEAR_BUDGET B_1993
      ON D.DEPT_NO = B_1993.DEPT_NO
      AND B_1993.FISCAL_YEAR = 1993
    LEFT JOIN DEPT_YEAR_BUDGET B_1994
      ON D.DEPT_NO = B_1994.DEPT_NO
      AND B_1994.FISCAL_YEAR = 1994
    LEFT JOIN DEPT_YEAR_BUDGET B_1995
      ON D.DEPT_NO = B_1995.DEPT_NO
      AND B_1995.FISCAL_YEAR = 1995
    LEFT JOIN DEPT_YEAR_BUDGET B_1996
      ON D.DEPT_NO = B_1996.DEPT_NO
      AND B_1996.FISCAL_YEAR = 1996

  WHERE EXISTS (
    SELECT * FROM PROJ_DEPT_BUDGET B
    WHERE D.DEPT_NO = B.DEPT_NO)
```

### *Regole per le CTE recursive*

- Una CTE recursiva è autoreferenziante (ha un riferimento a sé stessa)
- Una CTE recursiva è una UNION di membri recursivi e non recursivi.
  - Deve essere presente almeno un elemento non recursivo (ancoraggio o elemento fisso)
  - I membri non recursivi devono essere messi prima nella UNION
  - I membri recursivi sono separati dagli elementi di ancoraggio fissi e gli uni dagli altri con una clausola UNION ALL, cioè,

    membro non recursivo (àncora)
    UNION [ALL | DISTINCT]
    membro non recursivo (àncora)
    UNION [ALL | DISTINCT]
    membro non recursivo (àncora)
    UNION ALL
    membro recursivo
    UNION ALL
    membro recursivo

- I riferimenti fra le varie CTE non devono avere circolarità
-

Le aggregazioni (DISTINCT, GROUP BY, HAVING) e le funzioni di aggregazione (SUM, COUNT, MAX ecc) non sono permesse nei membri recursivi

- Un membro recursivo può avere uno ed un solo un riferimento a sé stessa e solo nella sua clausola FROM
- Un riferimento recursivo non può partecipare ad una OUTER JOIN

**Esempio di una CTE recursiva**

```
WITH RECURSIVE
  DEPT_YEAR_BUDGET AS
  (
    SELECT FISCAL_YEAR, DEPT_NO,
        SUM(PROJECTED_BUDGET) AS BUDGET
      FROM PROJ_DEPT_BUDGET
    GROUP BY FISCAL_YEAR, DEPT_NO
  ),

  DEPT_TREE AS
  (
    SELECT DEPT_NO, HEAD_DEPT, DEPARTMENT,
        CAST('' AS VARCHAR(255)) AS INDENT
      FROM DEPARTMENT
     WHERE HEAD_DEPT IS NULL

    UNION ALL

    SELECT D.DEPT_NO, D.HEAD_DEPT, D.DEPARTMENT,
    H.INDENT || '  '
      FROM DEPARTMENT D
      JOIN DEPT_TREE H
        ON D.HEAD_DEPT = H.DEPT_NO
  )

  SELECT D.DEPT_NO,
 D.INDENT || D.DEPARTMENT AS DEPARTMENT,
 B_1993.BUDGET AS B_1993,
 B_1994.BUDGET AS B_1994,
 B_1995.BUDGET AS B_1995,
 B_1996.BUDGET AS B_1996

  FROM DEPT_TREE D
    LEFT JOIN DEPT_YEAR_BUDGET B_1993
      ON D.DEPT_NO = B_1993.DEPT_NO
      AND B_1993.FISCAL_YEAR = 1993

    LEFT JOIN DEPT_YEAR_BUDGET B_1994
      ON D.DEPT_NO = B_1994.DEPT_NO
      AND B_1994.FISCAL_YEAR = 1994

    LEFT JOIN DEPT_YEAR_BUDGET B_1995
      ON D.DEPT_NO = B_1995.DEPT_NO
      AND B_1995.FISCAL_YEAR = 1995

    LEFT JOIN DEPT_YEAR_BUDGET B_1996
      ON D.DEPT_NO = B_1996.DEPT_NO
      AND B_1996.FISCAL_YEAR = 1996
```

## La funzione LIST

Oleg Loa
Dmitry Yemanov

**(v.2.1)** Questa funzione riporta una stringa che è il concatenamento di tutti i valori non NULL di un raggruppamento. Riporta NULL se non ci sono valori non-NULL.

**Formato**

```
<list function> ::=
  LIST '(' [ {ALL | DISTINCT} ] <espressione> [',' <delimitatore>
   ] ')'

<delimitatore> ::=
      { <stringa> | <parametro> | <variabile> }
```

**Regole sintattiche**

1. Se non si specifica né ALL e neppure DISTINCT, si intende implicitamente ALL.

2. Se è omesso il <delimitatore>, si adopera una virgola per separare i valori concatenati.

**Altre note**

1. I valori numerici e data/ora sono implicitamente convertiti in stringhe nella valutazione delle espressioni.

2. Il risultato è di tipo BLOB con SUB_TYPE TEXT in ogni caso, eccetto una lista esplicita di BLOB con sottotipi diversi.

3. L'ordine dei valori nel gruppo è dipendente dall'implementazione (quindi non fateci affidamento, perché può cambiare!).

**Esempi**

```
/* A */
  SELECT LIST(ID, ':')
  FROM MY_TABLE

/* B */
  SELECT TAG_TYPE, LIST(TAG_VALUE)
  FROM TAGS
  GROUP BY TAG_TYPE
```

## La clausola RETURNING

Dmitry Yemanov
Adriano dos Santos Fernandes

**(v.2.1)** Lo scopo di questa clausola è quella di permettere al client di avere di ritorno dalle frasi di `INSERT`, `UPDATE`, `UPDATE OR INSERT` e `DELETE` i valori inseriti (o eliminati) in una tabella.

L'uso tipico è recuperare il valore generato per una chiave primaria dall'esecuzione di un trigger `BEFORE`. La clausola `RETURNING` è opzionale ed è disponibile sia in DSQL che in PSQL, sebbene ci siano leggere differenze nella sintassi.

In DSQL, l'esecuzione dell'operazione stessa ed il riporto dei valori avvengono una unico ciclo di esecuzione.

Poiché la clausola `RETURNING` è stata progettata per riportare un solo record di dati (singleton) in risposta all'esecuzione di una operazione su una singola registrazione, non è consentito specificare tale clausola nelle frasi che inseriscono, aggiornano o cancellano più di un record.

> **Nota**
>
> In DSQL, si riporta comunque una riga, anche se nessun record ha soddisfatto le condizioni per l'operazione, e quindi nessun record è stato toccato. Pertanto, a questo stadio dell'implementazione, c'è la possibilità di riportare un record «vuoto», cioè contenente tutti NULL. Questo potrebbe essere modificato in una futura release.
>
> In PSQL, invece, se il comando non influisce su nessuna registrazione, non riporta valori e pertanto nessuna delle variabili di ritorno, specificate nella clausola INTO, viene modificata.

**Regole sintattiche generali**

```
INSERT INTO ... VALUES (...)
    [RETURNING <column_list> [INTO <variable_list>]]

INSERT INTO ... SELECT ...
    [RETURNING <column_list> [INTO <variable_list>]]

UPDATE OR INSERT INTO ... VALUES (...) ...
    [RETURNING <column_list> [INTO <variable_list>]]

UPDATE ... [RETURNING <column_list> [INTO <variable_list>]]

DELETE FROM ...
    [RETURNING <column_list> [INTO <variable_list>]]
```

### *Regole nell'uso della clausola `RETURNING`*

1. La parte della clausola INTO (cioè la lista delle variabili) è permessa solo in PSQL, per assegnare il record di risultati in uscita direttamente alle variabili locali. Ovviamente non è accettata in DSQL.

2. La della clausola RETURNING fa sì che un INSERT venga descritto dall'API come una `isc_info_sql_stmt_exec_procedure` e non come una `isc_info_sql_stmt_insert`. Gli attuali driver di interfaccia dovrebbero essere in grado di gestire questa caratteristica senza particolari modifiche.

3. La clausola RETURNING ignora ogni esplicita modifica o cancellazione che possa risultare dall'esecuzione di un AFTER trigger.

4. Le variabili di contesto OLD e NEW possono essere utilizzate nella clausola RETURNING di una frase UPDATE o INSERT OR UPDATE.

5. Nelle frasi UPDATE o INSERT OR UPDATE, ai riferimenti di campi non qualificati o qualificati dal nome di una tabella o di un alias vengono attribuiti i valori delle corrispondenti variabili di contesto NEW.

**Esempi**

1.
```
INSERT INTO T1 (F1, F2)
    VALUES (:F1, :F2)
      RETURNING F1, F2 INTO :V1, :V2;
```

2.
```
INSERT INTO T2 (F1, F2)
    VALUES (1, 2)
      RETURNING ID INTO :PK;
```

3.
```
DELETE FROM T1
  WHERE F1 = 1
    RETURNING F2;
```

4.
```
UPDATE T1
    SET F2 = F2 * 10
      RETURNING OLD.F2, NEW.F2;
```

## *La frase `UPDATE OR INSERT`*

Adriano dos Santos Fernandes

**(v.2.1)** Questa sintassi è stata introdotto per poter inserire o aggiornare un record in funzione della sua effettiva esistenza (verificata con `IS NOT DISTINCT`). La frase è disponibili sia in DSQL che in PSQL.

**Sintassi**

```
UPDATE OR INSERT INTO <table or view> [(<column_list>)]
    VALUES (<value_list>)
    [MATCHING (<column_list>)]
    [RETURNING <column_list> [INTO <variable_list>]]
```

**Esempi**

1.
```
UPDATE OR INSERT INTO T1 (F1, F2)
    VALUES (:F1, :F2);
```

2.
```
UPDATE OR INSERT INTO EMPLOYEE (ID, NAME)
    VALUES (:ID, :NAME)
    RETURNING ID;
```

3.
```
UPDATE OR INSERT INTO T1 (F1, F2)
```

```
        VALUES (:F1, :F2)
        MATCHING (F1);
```

4.

```
    UPDATE OR INSERT INTO EMPLOYEE (ID, NAME)
        VALUES (:ID, :NAME)
        RETURNING OLD.NAME;
```

### Note sull'uso

1. Quando si omette la clausola `MATCHING` è richiesta l'esistenza di una chiave primaria.
2. Sono necessari i relativi permessi di `INSERT` e `UPDATE` sulla <table or view>.
3. Se è presente la clausola `RETURNING`, allora la frase è descritta dall'API con una `isc_info_sql_stmt_exec_procedure`; altrimenti è descritta come una `isc_info_sql_stmt_insert`.

> **Nota**
>
> Un errore di «multiple rows in singleton select» (righe multiple in una select che richiede un risultato a riga singola) viene lanciato se la clausola `RETURNING` è presente e più di un record soddisfa le condizioni di ricerca.

## La frase `MERGE`

Adriano dos Santos Fernandes

**(v.2.1)** Questa sintassi è stata aggiunta per permettere di modificare o inserire un record se viene soddisfatta una certa condizione data. La frase è disponibile sia in DSQL che in PSQL.

**Sintassi**

```
<merge> ::=
MERGE
  INTO <tabella o vista> [ [AS] <nome di correlazione> ]
  USING <tabella o vista o tabella derivata> [ [AS] <nome di correlazione> ]
    ON <condizione>
    [ <merge quando soddisfatto> ]
    [ <merge quando non soddisfatto> ]

<merge quando soddisfatto> ::=
  WHEN MATCHED THEN
    UPDATE SET <lista assegnazioni>

<merge quando non soddisfatto> ::=
  WHEN NOT MATCHED THEN
    INSERT [ <parentesi aperta> <lista di colonne> <parentesi chiusa> ]
    VALUES <parentesi aperta> <lista di valori> <parentesi chiusa>
```

### Regole per il `MERGE`

1.

Deve essere sempre specificata almeno una delle due fra le condizione <merge quando soddisfatto> e <merge quando non soddisfatto>.

2. Nessuna delle due condizioni può essere specificata più di una volta.

> **Nota**
>
> Per l'esecuzione viene effettuata una `RIGHT JOIN` tra le tabelle specificate nelle parti `INTO` e `USING`, facendo uso della condizione specificata. L'`UPDATE` viene eseguita quando esiste un record che soddisfa la condizione, nella tabella a sinistra (INTO), altrimenti si effettua l'operazione di `INSERT`.
>
> Se la JOIN non riporta nessun record, non viene effettuata nessuna operazione e pertanto neanche la INSERT.

**Esempio**

```
MERGE INTO customers c
  USING (SELECT * FROM customers_delta WHERE id > 10) cd
  ON (c.id = cd.id)
    WHEN MATCHED THEN
      UPDATE SET
        name = cd.name
    WHEN NOT MATCHED THEN
      INSERT (id, name)
      VALUES (cd.id, cd.name)
```

## Nuovi tipi di JOIN

Adriano dos Santos Fernandes

**(v.2.1)** Sono stati introdotti due nuovi tipi di `JOIN`: la join *NAMED COLUMNS* (colonne in comune) e la join `NATURAL`.

### Sintassi e regole

```
<named columns join> ::=
  <table reference> <join type> JOIN <table reference>
    USING ( <column list> )

<natural join> ::=
 <table reference> NATURAL <join type> JOIN <table primary>
```

#### Named columns join (colonne in comune)

1. Tutte le colonne specificate nella lista di colonne <column list> devono esistere da entrambe le parti, cioè nelle tabelle destra e sinistra.

2. Viene automaticamente creata una equi-join (<tabella sinistra>.<colonna> = <tabella desta>.<colonna>) per ogni colonna elencata e, se ce n'è più di una, vengono fra di loro tutte collegate da `AND`.

3. Nella lista di campi specificati per l'output della `SELECT`, le colonne specificate in `USING` possono essere riportate senza qualificatori. In tal caso, il risultato è equivalente a COALESCE(<tabella sinistra>.<colonna>, <tabella destra>.<colonna>).

4. In caso di «SELECT *», le colonne di `USING` sono espanse una sola volta, usando la regola sopra specificata.

### *Natural join*

1. Viene creata automaticamente una «named columns join» con tutte le colonne comuni tra le due tabelle coinvolte.

2. Se non ci sono colonne comuni si effettua una `CROSS JOIN`.

**Esempi**

```
/* 1 */
select * from employee
  join department
  using (dept_no);

/* 2 */
select * from employee_project
  natural join employee
  natural join project;
```

## *CROSS JOIN*

D. Yemanov

**(V.2.0.x)** La sintassi CROSS JOIN è adesso supportata. Dal punto di vista logico, la sintassi:

```
A CROSS JOIN B
```

è equivalente alla seguente:

```
A INNER JOIN B ON 1 = 1
```

oppure, più semplicemente a:

```
FROM A, B
```

### *Prestazioni migliorate nella V.2.1.2*

D. Yemanov

Nel raro caso in cui in una CROSS JOIN con tre o più tabelle coinvolte, una o più di queste fossero vuote si è registrato un brusco calo nelle prestazioni *(CORE-2200)* . Un miglioramento delle prestazioni è stato ottenuto

istruendo l'ottimizzatore a non sprecare tempo ed energie nello scorrere le tabelle piene nel tentativo di trovare impossibili riscontri con tabelle vuote.

## I default per l'INSERT

D. Yemanov

Richiesta di nuova specifica

**(v.2.1)** Adesso è possibile effettuare inserimenti senza specificare valori, se ci sono trigger BEFORE INSERT e/o defaults disponibili per ogni colonna necessaria e nessuno di questi dipende da un valore di campo NEW fornito dal comando standard. In soldoni, niente deve dipendere dalla parte VALUES(<lista di valori>).

**Esempio**

```
INSERT INTO <table>
  DEFAULT VALUES
  [RETURNING <values>]
```

## Compatibilità fra testo VARCHAR e BLOB

A. dos Santos Fernandes

**(v.2.1)** A vari livelli di gestione, il motore adesso tratta i BLOB di testo che contengono stringhe più corte di 32766 byte come se fossero di tipo VARCHAR. La lunghezza massima per questa conversione implicita è pertanto 32765 bytes (non caratteri!).

Le operazioni che permettono ai BLOB di testo di comportarsi come stringhe VARCHAR sono le assegnazioni, conversioni e concatenazioni (fino al limite sopra specificato per il risultato della concatenazione), così come le funzioni CAST, LOWER, UPPER, TRIM e SUBSTRING.

> **Attenzione!**
>
> La funzione SUBSTRING(), se applicata ad un BLOB di testo, adesso ha per risultato un BLOB di testo, e non più un VARCHAR come nelle versioni precedenti. Questa modifica potrebbe influenzare il risultato nel codice esistente sia client che PSQL.

## Comparazione sul contenuto completo dei BLOB di testo

**(v.2.0.x)** Si possono effettuare comparazioni sull'intero contenuto di un BLOB di testo.

## RDB$DB_KEY riporta NULL nelle OUTER JOIN

A. dos Santos Fernandes

Richiesta CORE-979

**(v.2.1)** Per qualche anomalia, la RDB$DB_KEY fisica ha sempre riportato un valore in ogni riga quando si specifica una OUTER JOIN, pertanto effettuando un test su quei valori dove ci si aspetterebbe un NULL per

mancanza di corrispondenza, nelle versioni precedenti il test avrebbe riportato un risultato errato (rovesciato: falso al posto di vero). Ora, RDB$DB_KEY riporta correttamente NULL quando dovrebbe.

## Riammesso l'ordinamento nelle colonne BLOB e ARRAY

Dmitry Yemanov

**(v.2.1)** Nelle prime pre-release di Firebird 2.1, sono stati introdotti controlli per rifiutare gli ordinamenti (nelle operazioni di ORDER BY, GROUP BY e SELECT DISTINCT) al momento della preparazione se la clausola di ordinamento implicitamente o esplicitamente implicava il sort su una colonna di tipo BLOB o ARRAY.

Questa modifica è stata eliminata nella pre-release RC2, non perché sbagliata ma perché molti utenti si sono lamentati che le loro applicazioni non funzionavano più.

> **Importante**
>
> Questo ritorno al «passato» non implica in nessun modo che tali query possano magicamente riportare risultati corretti. Un BLOB generico non può essere automaticamente convertito ad un tipo di dato ordinabile, e pertanto, come un tempo, gli ordinamenti DISTINCT e gli argomenti di ORDER BY che includono BLOB, usano il BLOB_ID. Come succedeva prima, le query dove gli argomenti di GROUP BY sono tipi BLOB, si possono prepare, ma provocano eccezioni durante l'esecuzione.

## Funzioni integrate

**(v.2.1)** Sono state migliorate alcune delle funzioni integrate esistenti, e ne sono state aggiunte un bel po'.

### Nuove funzioni integrate

Adriano dos Santos Fernandes
Oleg Loa
Alexey Karyakin

Un certo numero di funzioni sono state integrate direttamente nel motore di Firebird 2.1 per sostituire analoghe funzioni presenti nelle UDF con lo stesso nome. Le funzioni integrate non vengono usate se nel database sono dichiarate con lo stesso nome le funzioni delle UDF.

> **Nota**
>
> La scelta tra UDF e funzione integrata viene decisa al momento della compilazione dello statement. Questo implica che tutto il codice PSQL compilato mentre sono definite le UDF nel database continuerà a richiedere che le dichiarazioni delle UDF siano presenti fino a che non viene ricompilato.
>
> La nuova funzione integrata *DECODE()* non ha un'equivalente nelle librerie UDF distribuite con Firebird.

L'elenco di tali funzioni è esposto in dettagli in Appendice A.

> **Nota**
>
> Molte di queste funzioni integrate erano già disponibili in Firebird 2/ODS 11, ad esempio LOWER(), TRIM(), BIT_LENGTH(), CHAR_LENGTH() e OCTET_LENGTH().

### Miglioramenti alle funzioni integrate

A. dos Santos Fernandes

*EXTRACT(WEEK FROM DATE)*

Richiesta in CORE-663

La funzione EXTRACT() è stata estesa per supportare i numeri di settimana dell'anno secondo lo standard ISO-8601. Per esempio:

```
EXTRACT (WEEK FROM date '30.09.2007')
```

riporta 39. Altro esempio:

```
ALTER TABLE XYZ
ADD WeekOfTheYear
COMPUTED BY (
  CASE
    WHEN (EXTRACT(MONTH FROM DataEsempio) = 12)
    AND (EXTRACT(WEEK FROM DataEsempio) = 1)
    THEN
      'Settimana '|| EXTRACT (WEEK FROM DataEsempio)||' dell''anno '
      || (1 + (EXTRACT( YEAR FROM DataEsempio)))
    else 'Settimana '||EXTRACT (WEEK FROM DataEsempio)||' dell''anno '
      ||EXTRACT( YEAR FROM DataEsempio)
  end )
```

*Specificare la scala in TRUNC()*

Richiesta in CORE-1340

Fino alla versione Beta 1 la funzione integrata TRUNC() aveva un solo argomento, il valore da troncare. A partire dalla Beta 2 può essere specificato un secondo argomento opzionale per indicare la scala del troncamento. Ad esempio:

```
select
  trunc(987.65, 1),
  trunc(987.65, -1)
  from rdb$database;
```

riporta i due valori 987.60 e 980.00

Per altri esempi d'uso della TRUNC() con o senza l'argomento di scala opzionale, vedere all'elenco alfabetico delle funzioni nell'Appendice A.

*Gestione dei millisecondi in EXTRACT(), DATEADD() e DATEDIFF()*

Richiesta in CORE-1387

A partire da Firebird 2.1 Beta 2, le funzioni integrate EXTRACT(), DATEADD() e DATEDIFF() possono gestire i millisecondi rappresentati da un numero intero di 4 cifre. Ad esempio:

```
EXTRACT ( MILLISECOND FROM timestamp '01.01.2000 01:00:00.1234' )
```

riporta 123

```
DATEADD ( MILLISECOND, 100, timestamp '01.01.2000 01:00:00.0000' )
DATEDIFF ( MILLISECOND, timestamp '01.01.2000 02:00:00.0000', timestamp '01.01.2000 01:00:00.
```

Per esempi più particolareggiati sull'uso di DATEADD() e DATEDIFF(), fare riferimento all'elenco delle funzioni integrate in Appendice A.

## Funzioni migliorate già dalla versione 2.0.x

Alcune migliorie alle funzioni integrate erano già disponibili nelle versioni 2.0.x di Firebird:

### IIF()

O. Loa

**(V.2.0.x)** Per testare una condizione ed avere esattamente due soli valori in alternativa, si può usare la funzione IIF() al posto della più prolissa espressione CASE. IIF() riporta il valore della prima di due espressioni se la condizione di test è vera, altrimenti riporta il valore della seconda espressione. Il test è per la precisione una condizione di ricerca, ed ammette la stessa sintassi della parte WHERE di una SELECT.

```
IIF (<condizione_di_ricerca>, <valore_se_vero>, <valore_altrimenti>)
```

viene implementato come un'abbreviazione di:

```
CASE
  WHEN <condizione_di_ricerca> THEN <valore_se_vero>
  ELSE <valore_altrimenti>
END
```

**Esempio**

```
 SELECT IIF(VAL > 0, VAL, -VAL) FROM OPERATION
```

### Miglioramenti nel comportamento di CAST()

D. Yemanov

**(V.2.0.x)** L'errore «Datatype unknown» (SF Bug #1371274) che compariva tentando certi CAST è stato eliminato. Ora è possibile usare CAST() per avvisare il motore del tipo di dato di certi parametri.

*Esempio*

```
SELECT CAST(? AS INT) FROM RDB$DATABASE
```

### Argomenti espressione per SUBSTRING()

O. Loa, D. Yemanov

**(V.2.0.x)** La funzione integrata SUBSTRING() può ricevere ora nei parametri espressioni valutabili come interi.

Precedentemente la funzione integrata SUBSTRING() poteva accettare solo valori interi come secondo e terzo parametro(rispettivamente posizione iniziale e lunghezza). Ora tali argomenti possono essere costituiti da qualsiasi espressione che possa avere un valore numerico, tra cui parametri host, risultati di funzione, espressioni, subquery ecc. ecc.

> **Suggerimento**
>
> Se nel tentativo di usare questa caratteristica ci sono errori di «invalid token», ricordarsi che le espressioni usate negli argomenti spesso devono essere racchiuse tra parentesi tonde!

### Modifiche ai risultati ottenuti da SUBSTRING()

**(V.2.1.x)** Per aderire allo standard, la lunghezza in caratteri applicando una SUBSTRING() ad un campo VARCHAR o CHAR è adesso un VARCHAR della stessa lunghezza dichiarata o dedotta per il valore del primo argomento.

In Firebird 2.0 e 1.5, il valore era un CHAR con la stessa lunghezza in caratteri del valore dichiarato o implicito ancora del primo argomento. Tale regola avrebbe creato problemi in in Firebird 2.0 nel caso in cui la stringa in ingresso era CHAR e l'argomento del FOR era un'espressione la cui dimensione sarebbe stata ignota la momento della preparazione della zona di memoria che avrebbe dovuto contenere la stringa risultante. La modifica della V.2.1 ha corretto questo fatto.

Non è necessario ridefinire ogni variabile PSQL che è stata dichiarata a ricevere il risultato da una SUBSTRING(). Rimane corretto dichiarare la sua dimensione grande abbastanza a contenere il dato effettivamente riportato. È sufficiente essere certi che qualsiasi espressione usata nell'argomento del FOR non possa avere un valore intero superiore al numero dei caratteri dichiarati di quella variabile.

> **Problemi dei campi BLOB**
>
> Chiaramente, essendo un testo BLOB di lunghezza indeterminata, non può rientrare in un paradigma in cui può riempire una stringa di dimensione massima nota. Pertanto, il risultato riportato da SUBSTRING() applicato ad un BLOB di testo non è un VARCHAR() come specificato sopra, ma ancora un BLOB di testo.
>
> Questa modifica può pertanto bloccare codice PSQL ed altre espressioni funzionanti.
>
> - Attenzione ai sottodimensionamenti! Fare particolare attenzione ai CAST e alle concatenazioni.
> - Fare attenzione all'uso della memoria quando si assegnano BLOB temporanei nei cicli! Il motore alloca un minimo di una pagina di memoria per ogni BLOB temporaneo, indipendentemente dalla sua reale dimensione.

## In DSQL la scansione dei nomi di tabella è più rigida

A. Brinkman

La gestione degli alias e la ricerca dei nomi di campo ambigui è stata migliorata. In sostanza:

1. Quando viene specificato l'alias per una tabella, o si usa l'alias o non si usa niente. Non è più permesso specificare il nome della tabella.

2. Il controllo di ambiguità per prima cosa ora verifica nell'ambito del livello corrente le ambiguità presenti nei nomi di campo, validandole in certi casi come colonne utilizzabili senza qualificatori in un ambito di livello più alto.

**Esempi**

1. Quando è specificato un alias per una tabella, i suoi campi vanno specificati con l'alias oppure non va usato nulla.

   a. Questa query era permessa fino a FB1.5:

```
SELECT
  RDB$RELATIONS.RDB$RELATION_NAME
FROM
  RDB$RELATIONS R
```

   ma ora riporterà correttamente un errore specificando che il campo "RDB$RELATIONS.RDB $RELATION_NAME" non può essere trovato.

   Invece va usato così (preferibilmente):

```
SELECT
  R.RDB$RELATION_NAME
FROM
  RDB$RELATIONS R
```

   oppure così:

```
SELECT
  RDB$RELATION_NAME
FROM
  RDB$RELATIONS R
```

   b. La frase seguente correttamente ora recupera il campo FieldID dalla subquery e dalla tabella da :

```
UPDATE
  TableA
SET
  FieldA = (SELECT SUM(A.FieldB) FROM TableA A
    WHERE A.FieldID = TableA.FieldID)
```

> **Nota**
>
> In Firebird è possibile specificare un alias nelle frasi di aggiornamento. Sebbene molti altri database non lo supportino, questa possibilità dovrebbe aiutare quegli sviluppatori che hanno richiesto di rendere il linguaggio SQL di Firebird più compatibile con quei prodotti che la supportano.

c. Questo esempio non funziona correttamente in Firebird 1.5 e precedenti:

```
SELECT
  RDB$RELATIONS.RDB$RELATION_NAME,
  R2.RDB$RELATION_NAME
FROM
  RDB$RELATIONS
  JOIN RDB$RELATIONS R2 ON
    (R2.RDB$RELATION_NAME = RDB$RELATIONS.RDB$RELATION_NAME)
```

Se RDB$RELATIONS contiene 90 record, riporta 90 * 90 = 8100 record, ma in Firebird 2 e successivi riporta correttamente solo 90 records.

2. a. Questo dava un errore di sintassi in Firebird 1.5, ma è accettato in Firebird 2:

```
SELECT
  (SELECT RDB$RELATION_NAME FROM RDB$DATABASE)
FROM
  RDB$RELATIONS
```

b. Il controllo di ambiguità nelle subqueries: la query seguente funzionava in Firebird 1.5 senza segnalare alcuna ambiguità, cosa che invece viene effettuata da Firebird 2:

```
SELECT
  (SELECT
     FIRST 1 RDB$RELATION_NAME
   FROM
     RDB$RELATIONS R1
     JOIN RDB$RELATIONS R2 ON
       (R2.RDB$RELATION_NAME = R1.RDB$RELATION_NAME))
FROM
  RDB$DATABASE
```

## *La frase EXECUTE BLOCK*

V. Khorsun

EXECUTE BLOCK è un'estensione al linguaggio SQL che permette l'uso del "PSQL dinamico" all'interno di una SELECT. Ha l'effetto di ottenere un blocco di codice PSQL autocontenuto in modo da essere eseguito in una SQL dinamico come se fosse una STORED PROCEDURE.

**Sintassi**

```
EXECUTE BLOCK [ (param datatype = ?, param datatype = ?, ...) ]
  [ RETURNS (param datatype, param datatype, ...) ]
```

```
AS
[DECLARE VARIABLE var datatype; ...]
BEGIN
  ...
END
```

Per il client, la chiamata `isc_dsql_sql_info` col parametro `isc_info_sql_stmt_type` riporta

- `isc_info_sql_stmt_select` se il blocco ha parametri in uscita. La semantica della chiamata è simile a quella di una SELECT: il client ha un cursore aperto, può ricevere dati attraverso di esso, e lo deve chiudere dopo l'uso.
- `isc_info_sql_stmt_exec_procedure` se il blocco non ha parametri in uscita. La semantica della chiamata è simile a quella di una EXECUTE: il client non ha nessun cursore e l'esecuzione continua fino alla fine del blocco o termina per aver incontrato una SUSPEND.

Il client deve preprocessare solo la testata dello statement SQL oppure usare '? invece di ':' come indicatore di parametro, perché, nel corpo del blocco, potrebbero esserci riferimenti a variabili locali o ad argomenti con il prefisso ':' .

**Esempio**

Se il comando SQL da usare è

```
EXECUTE BLOCK (X INTEGER = :X)
  RETURNS (Y VARCHAR)
AS
DECLARE V INTEGER;
BEGIN
  INSERT INTO T(...) VALUES (... :X ...);
  SELECT ... FROM T INTO :Y;
  SUSPEND;
END
```

Il comando SQL preprocessato deve essere

```
EXECUTE BLOCK (X INTEGER = ?)
  RETURNS (Y VARCHAR)
AS
DECLARE V INTEGER;
BEGIN
  INSERT INTO T(...) VALUES (... :X ...);
  SELECT ... FROM T INTO :Y;
  SUSPEND;
END
```

## Tabelle derivate (Derived Tables oppure DT)

A. Brinkman

Il supporto per le tabelle derivate in DSQL (che sono sottoquery poste nella clausola FROM) è stato aggiunto come definito nello standard SQL200X. Una tabella derivata è un insieme di righe ottenute da una frase SELECT dinamica. Le tabelle derivate possono essere innestate, se necessario, per costruire complesse query e possono far parte di JOIN come se fossero normali tabelle o viste.

**Sintassi**

```
SELECT
  <elenco di selezioni>
FROM
  <elenco riferimenti tabellari>

<elenco riferimenti tabellari> ::= <riferimento tabellare> [{<virgola> <riferimento tabellar

<riferimento tabellare> ::=
    <tabella primaria>
  | <tabella di join>

<tabella primaria> ::=
    <tabella> [[AS] <pseudonimo>]
  | <tabella derivata>

<tabella derivata> ::=
    <espressione query> [[AS] <pseudonimo>]
      [<parentesi aperta> <elenco di colonne derivate> <parentesi chiusa>]

<elenco di colonne derivate> ::= <nome colonna> [{<virgola> <nome colonna>}...]
```

**Esempi**

a) Tabella derivata semplice:

```
SELECT
  *
FROM
  (SELECT
     RDB$RELATION_NAME, RDB$RELATION_ID
   FROM
     RDB$RELATIONS) AS R (RELATION_NAME, RELATION_ID)
```

b) Aggregazione su una tabella derivata che contiene a sua volta un aggregato

```
SELECT
  DT.FIELDS,
  Count(*)
FROM
  (SELECT
     R.RDB$RELATION_NAME,
     Count(*)
   FROM
     RDB$RELATIONS R
     JOIN RDB$RELATION_FIELDS RF ON (RF.RDB$RELATION_NAME = R.RDB$RELATION_NAME)
   GROUP BY
     R.RDB$RELATION_NAME) AS DT (RELATION_NAME, FIELDS)
GROUP BY
  DT.FIELDS
```

c) Esempio con UNION e ORDER BY:

```
SELECT
```

```
  DT.*
FROM
  (SELECT
     R.RDB$RELATION_NAME,
     R.RDB$RELATION_ID
   FROM
     RDB$RELATIONS R
   UNION ALL
   SELECT
     R.RDB$OWNER_NAME,
     R.RDB$RELATION_ID
   FROM
     RDB$RELATIONS R
   ORDER BY
     2) AS DT
WHERE
  DT.RDB$RELATION_ID <= 4
```

**Note essenziali**

- Ogni colonna nelle tabelle derivate deve avere un nome. Le espressioni innominate come le costanti dovrebbero essere aggiunte con una alias oppure specificando l'elenco delle colonne risultato.
- Il numero di campi nella lista di colonne risultante dovrebbe essere lo stesso del numero di colonne nella espressione della query.
- L'ottimizzatore è in grado di gestire in modo molto efficiente le tabelle derivate. Tuttavia, se la tabella derivata è coinvolta in una INNER JOIN e contiene subquery, non si può ottenere un ordinamento sulla join e le prestazioni ne soffrono.

## Sintassi per ROLLBACK RETAIN

D. Yemanov

In DSQL adesso è supportata anche la frase ROLLBACK RETAIN.

Il «rollback retaining» era stato introdotto in InterBase 6.0, ma questa modalità di rollback poteva essere usata solo attraverso una chiamata API: la *isc_rollback_retaining()*. Invece, il «commit retaining» poteva essere invocato sia con la chiamata API *isc_commit_retaining()*, sia usando il comando DSQL `COMMIT RETAIN`.

Firebird 2.0 aggiunge una clausola opzionale `RETAIN` alla comando DSQL `ROLLBACK` per renderlo consistente al COMMIT [RETAIN].

La *sintassi*: è simile a quella del COMMIT RETAIN.

## Sintassi ROWS

D. Yemanov

La sintassi ROWS si usa per limitare il numero di righe riportate da un'espressione di selezione. Viene applicata alla fine di una frase di SELECT di livello principale, e specifica il numero di righe da riportare al programma principale. Di fatto costituisce un'alternativa meglio comprensibile alle clausole FIRST/SKIP, e, oltre ad essere in accordo con lo standard SQL più recente, la sintassi ROWS da' alcuni benefici extra. Può infatti essere usata nelle UNION, in ogni tipo di subquery e nelle frasi UPDATE o DELETE.

Disponibile sia in DSQL che in PSQL.

**Sintassi**

```
SELECT ...
  [ORDER BY <expr_list>]
  ROWS <expr1> [TO <expr2>]
```

**Esempi**

1.

```
SELECT * FROM T1
  UNION ALL
SELECT * FROM T2
  ORDER BY COL
  ROWS 10 TO 100
```

2.

```
SELECT COL1, COL2,
  ( SELECT COL3 FROM T3 ORDER BY COL4 DESC ROWS 1 )
FROM T4
```

3.

```
DELETE FROM T5
  ORDER BY COL5
  ROWS 1
```

**Note essenziali**

1. Quando si omette <expr2>, allora ROWS <expr1> ha lo stesso significato di FIRST <expr1>. Quando si usano entrambe le espressioni <expr1> e <expr2>, allora `ROWS <expr1> TO <expr2>` ha lo stesso significato di `FIRST (<expr2> - <expr1> + 1) SKIP (<expr1> - 1)`

2. Non c'è nulla di equivalente alla clausola SKIP usata senza la clausola FIRST.

## Miglioramenti alla gestione delle UNION

Le regole per le query UNION sono state migliorate come segue:

### Implementazione di UNION DISTINCT

D. Yemanov

UNION DISTINCT è attualmente permesso come sinonimo di una semplice UNION, in accordo con le specifiche SQL-99. C'è una piccola differenza: DISTINCT è il funzionamento di default, in accordo allo standard. Precedentemente, Firebird non supportava l'esplicita inclusione della parola chiave opzionale DISTINCT.

*Sintassi*

```
UNION [{DISTINCT | ALL}]
```

### *Coercizione del tipo di dato nelle UNION migliorata*

A. Brinkman

Il sistema che risolve quale tipo di dato attribuire in automatico ad una aggregazione di valori fra loro compatibili, come tra le varie espressioni di un CASE o le colonne nella stessa posizione in una query unione, adesso utilizza un insieme di regole più raffinato.

**Regole sintattiche**

Sia DTS l'insieme dei tipi di dato da cui si deve determinare il tipo di dato risultante

1. Tutti i dati in DTS devono essere fra loro compatibili.
2. Si possono verificare i seguenti casi:
   a. se uno qualsiasi dei tipi di dato in DTS è una stringa di caratteri, allora:
      i. se uno qualsiasi dei tipi di dato in DTS è una stringa di caratteri a lunghezza variabile, allora il tipo risultante è una stringa di caratteri a lunghezza variabile con lunghezza massima (in caratteri) uguale al più grande dei tipi di dato nel DTS.
      ii. Altrimenti, il tipo del dato risultante è una stringa di caratteri a lunghezza fissa con lunghezza (in caratteri) uguale alla massima lunghezza dei tipi di dato nel DTS.
      iii. Il set di caratteri e l'ordinamento vengono presi dalla prima stringa del DTS.
   b. se tutto il DTS è in precisione numerica esatta, allora il risultato è numerico esatto con scala uguale al massimo delle scale dei tipi di dato nel DTS e la precisione massima dei dati del DTS.

      > **Nota**
      >
      > ATTENZIONE: il controllo per l'overflow di precisione viene fatto solamente durante l'esecuzione del comando. Chi sviluppa deve prendere le necessarie precauzioni per evitare che l'aggregazione risolva ad un overflow di precisione.

   c. Se un qualche dato del DTS è un numero approssimato, allora ogni dato nel DTS deve essere numerico altrimenti viene generato un errore.
   d. Se un qualche dato del DTS è di un tipo orario (DATE, TIME, TIMESTAMP), allora ogni dato nel DTS deve essere esattamente dello stesso tipo del primo.
   e. se un qualche dato del DTS è un BLOB, allora tutti i dati del DTS devono essere BLOB e tutti dello stesso sottotipo.

### *UNION permesse nelle subquery ANY/ALL/IN*

D. Yemanov

L'elemento subquery in una ricerca ANY, ALL o IN può essere una query UNION.

## Miglioramenti nella gestione del NULL

Sono state aggiunte le seguenti caratteristiche che riguardano la gestione del NULL in DSQL:

### Nuovo predicato [NOT] DISTINCT per valutare l'uguaglianza di due operandi NULL

O. Loa, D. Yemanov

Si tratta di un nuovo predicato di equivalenza che si comporta esattamente come i predicati di uguaglianza/disuguaglianza, ma, invece di controllare l'uguaglianza, verifica se i due operandi sono distinti fra loro.

Pertanto, IS NOT DISTINCT tratta (NULL uguale NULL) come se fosse vero, poiché NULL (o un'espressione che vale NULL) non è distinguibile da un'altra. Questo è disponibile sia in DSQL che in PSQL.

**Sintassi**

```
<valore> IS [NOT] DISTINCT FROM <valore>
```

**Esempi**

1.

```
SELECT * FROM T1
  JOIN T2
    ON T1.NAME IS NOT DISTINCT FROM T2.NAME;
```

2.

```
SELECT * FROM T
  WHERE T.MARK IS DISTINCT FROM 'test';
```

**Precisazioni essenziali**

1. Il predicato DISTINCT non considera distinti due valori NULL, pertanto non riporta un risultato sconosciuto come i predicati di uguaglianza/disuguaglianza. Come il predicato IS [NOT] NULL, può essere solamente o vero o falso.

   **Per maggiori informazioni sui NULL**

   Per maggiori informazioni su come sono valutati i confronti che coinvolgono campi a NULL, esaminare la *Guida sull'uso di NULL nel linguaggio SQL di Firebird* (Firebird Null Guide), disponibile in italiano sul sito di Firebird nell'Indice dei documenti.

2. Il predicato NOT DISTINCT può essere ottimizzato usando un indice, se è disponibile.

### *Ammorbidita la regola per i NULL espliciti*

D. Yemanov

Un esplicito NULL può essere ora usato come un valore in tutte le espressioni, senza dare un errore di sintassi. Pertanto espressioni come le seguenti sono adesso valide:

```
A = NULL
B > NULL
A + NULL
B || NULL
```

> **Nota**
>
> Tutte queste espressioni sono valutate a NULL. La modifica non altera il significato dell'espressione per il motore, semplicemente la fa accettare come espressione valida.

### *Modifiche all'ordinamento dei NULL per aderire allo standard*

N. Samofatov

La posizione dei NULL in un insieme ordinato è stata modificata in accordo con le specifiche dello standard SQL in modo tale che sia consistente con l'ordinamento, cioè se un ordinamento ASC[ENDING] li mette alla fine, allora un ordinamento DESC[ENDING] li mette all'inizio o viceversa. Questo si applica alle basi di dati create con la ODS 11 o successiva, poiché per funzionare questa modifica ha bisogno delle modifiche apportate agli indici.

> **Importante**
>
> Se si forza il posizionamento di default, non può essere usato nessun indice per l'ordinamento. Pertanto, non potrà essere usato nessun indice né per un ordinamento ASCENDING se viene specificato NULLS LAST, né per un ordinamento DESCENDING se si specifica NULLS FIRST.

**Esempi**

```
Database:  proc.fdb
SQL> create table gnull(a int);
SQL> insert into gnull values(null);
SQL> insert into gnull values(1);
SQL> select a from gnull order by a;
           A
============
      <null>
           1

SQL> select a from gnull order by a asc;

           A
============
      <null>
           1
```

```
SQL> select a from gnull order by a desc;

           A
============
           1
      <null>

SQL> select a from gnull order by a asc nulls first;

           A
============
      <null>
           1

SQL> select a from gnull order by a asc nulls last;

           A
============
           1
      <null>


SQL> select a from gnull order by a desc nulls last;

           A
============
           1
      <null>

SQL> select a from gnull order by a desc nulls first;

           A
============
      <null>
           1
```

## Le subquery e l'INSERT possono accettare insiemi UNION

D. Yemanov

Le specifiche di una SELECT nelle subquery e nelle frasi INSERT INTO <specifica-di-insert> SELECT... possono ora specificare anche insiemi UNION.

## Nuove estensioni alle sintassi di UPDATE e DELETE

O. Loa

Le specifiche di ROWS e le clausole PLAN e ORDER BY possono ora essere usate nelle frasi di UPDATE e di DELETE.

Gli utenti possono ora definire PLAN specifici per le frasi di UPDATE e di DELETE in modo da ottimizzarle a mano. Inoltre è possibile limitare il numero dei record influenzati dalla direttiva con una clausola ROWS, opzionalmente usata in combinazione con una clausola ORDER BY per ottenere un insieme di registrazioni ordinato.

*Sintassi*

```
UPDATE ... SET ... WHERE ...
[PLAN <elementi di plan>]
[ORDER BY <lista elementi ordinamento>]
[ROWS <valnum-1> [TO <valnum-2>]]
```

or

```
DELETE ... FROM ...
[PLAN <elementi di plan>]
[ORDER BY <lista elementi ordinamento>]
[ROWS <valnum-1> [TO <valnum-2>]]
```

## Estensioni alle variabili di contesto

Un certo numero di caratteristiche sono state aggiunte per estendere le informazioni di contesto che possono essere gestite:

### Precisione e millesimi di secondo nelle variabili Time e DateTime

D. Yemanov

#### CURRENT_TIMESTAMP e 'NOW' ora riportano i millisecondi

La variabile di contesto CURRENT_TIMESTAMP ed il letterale 'NOW' ora riportano le frazioni di secondo in millisecondi.

#### Specificabile la precisione in secondi in CURRENT_TIME e CURRENT_TIMESTAMP

CURRENT_TIME e CURRENT_TIMESTAMP ora possono opzionalmente specificare una precisione nei secondi.

La caratteristica è disponibile sia in DSQL che in PSQL.

**Sintassi**

```
CURRENT_TIME [(<precisione>)]
CURRENT_TIMESTAMP [(<precisione>)]
```

**Esempi**

```
1. SELECT CURRENT_TIME FROM RDB$DATABASE;
2. SELECT CURRENT_TIME(3) FROM RDB$DATABASE;
3. SELECT CURRENT_TIMESTAMP(3) FROM RDB$DATABASE;
```

**Nota**

1. La massima precisione specificabile è 3, che significa l'accuratezza ad un millesimo di secondo. Questa precisione potrebbe essere aumentata in futuro.
2. Se non si specifica la precisione, i valori di default impliciti sono:
   - 0 per CURRENT_TIME
   - 3 per CURRENT_TIMESTAMP

## *Nuove funzioni di sistema per gestire le variabili di contesto*

N. Samofatov

I valori delle variabili di contesto possono ora essere ottenuti usando le funzioni di sistema RDB $GET_CONTEXT() e RDB$SET_CONTEXT(). Queste nuove funzioni integrate accedono in SQL ad informazioni relative alla connessione e transazione attuali. Attraverso di esse è possibile ora gestire dati di contesto definiti dall'utente associandoli alla transazione o alla connessione.

**Sintassi**

```
RDB$SET_CONTEXT( <namespace>, <variabile>, <valore> )
RDB$GET_CONTEXT( <namespace>, <variabile> )
```

Queste funzioni sono in realtà una specie di funzioni esterne che però fanno parte integrante di tutti i database invece di dover essere richiamate da una libreria esterna caricata dinamicamente. Le seguenti dichiarazioni sono fatte automaticamente dal motore alla creazione del database:

**Dichiarazioni (implicite)**

```
DECLARE EXTERNAL FUNCTION RDB$GET_CONTEXT
    VARCHAR(80),
    VARCHAR(80)
RETURNS VARCHAR(255) FREE_IT;

DECLARE EXTERNAL FUNCTION RDB$SET_CONTEXT
    VARCHAR(80),
    VARCHAR(80),
    VARCHAR(255)
RETURNS INTEGER BY VALUE;
```

**Uso**

RDB$SET_CONTEXT e RDB$GET_CONTEXT impostano e recuperano il valore attuale per una variabile di contesto. Raggruppamenti di variabili di contesto con proprietà simili sono reperiti con identificatori di «spazi di nomi» (namespace). Ogni «spazio di nomi» determina le regole per l'uso, quali ad esempio l'accessibilità in lettura o scrittura da parte di un utente.

**Nota**

Gli spazi di nomi e i nomi delle variabili sono sensibili alle maiuscole/minuscole.

- RDB$GET_CONTEXT recupera il valore attuale di una variabile. Se la variabile non esiste nello spazio di nomi specificato, la funzione riporta NULL.
- RDB$SET_CONTEXT imposta un valore per la specificata variabile, se è possibile l'accesso in scrittura. La funzione riporta il valore 1 se la variabile esisteva già valorizzata al momento della chiamata, altrimenti riporta 0.
- Per eliminare una variabile da un contesto, impostarla a NULL.

### *Nomi di spazi predefiniti*

Sono disponibili un certo numero di spazi di nomi prefissati:

### *USER_SESSION*

Permette l'accesso a variabili definite dall'utente specifiche per la sessione corrente. Si possono definire ed impostare i valori a variabili con qualsiasi nome in questo contesto.

### *USER_TRANSACTION*

Offre le stesse possibilità del precedente, però a livello di singola transazione.

### *SYSTEM*

Permette un accesso in sola lettura alle seguenti variabili predefinite:

- NETWORK_PROTOCOL :: Il protocollo di rete utilizzato dal client per connettersi. I valori attualmente definiti sono : «TCPv4», «WNET», «XNET» e NULL.
- CLIENT_ADDRESS :: L'indirizzo di rete del client remoto, come stringa. Il valore è, per i protocolli TCPv4. un indirizzo IP nella forma "xxx.xxx.xxx.xxx" oppure, per il protocollo XNET è l'ID del processo locale oppure, per ogni altro protocollo, NULL.
- DB_NAME :: Nome canonico del database corrente. Può essere il nome dell'alias (se la connessione attraverso i nomi di file è disabilitata con DatabaseAccess = NONE in firebird.conf) oppure, il nome del file completo per esteso.
- ISOLATION_LEVEL :: Il livello di isolamento della transazione corrente. Il valore può essere uno fra "READ COMMITTED", "SNAPSHOT", "CONSISTENCY".
- TRANSACTION_ID :: L'ID numerico della transazione corrente. Il valore è lo stesso della pseudo-variabile CURRENT_TRANSACTION.
- SESSION_ID :: L'ID numerico della sessione attuale. Il valore riportato è lo stesso della pseudo variabile CURRENT_CONNECTION.
- CURRENT_USER :: L'utente attuale. Il valore riportato è lo stesso riportato dalla pseudo-variabile CURRENT_USER o dalla variabile predefinita USER.
- CURRENT_ROLE :: Il ruolo della connessione. Riporta lo stesso valore della pseudo-variabile CURRENT_ROLE.

### *Note*

Per impedire attacchi del tipo DoS (Denial of Service) al Server Firebird, il numero di variabili memorizzate in ogni contesto di sessione o transazione è limitato a 1000.

**Esempi d'uso**

```
   set term ^;
   create procedure set_context(User_ID varchar(40), Trn_ID integer) as
   begin
     RDB$SET_CONTEXT('USER_TRANSACTION', 'Trn_ID', Trn_ID);
     RDB$SET_CONTEXT('USER_TRANSACTION', 'User_ID', User_ID);
   end ^

  create table journal (
     jrn_id integer not null primary key,
     jrn_lastuser varchar(40),
     jrn_lastaddr varchar(255),
     jrn_lasttransaction integer
  )^

CREATE TRIGGER UI_JOURNAL FOR JOURNAL BEFORE INSERT OR UPDATE
  as
  begin
    new.jrn_lastuser = rdb$get_context('USER_TRANSACTION', 'User_ID');
    new.jrn_lastaddr = rdb$get_context('SYSTEM', 'CLIENT_ADDRESS');
    new.jrn_lasttransaction = rdb$get_context('USER_TRANSACTION', 'Trn_ID');
  end ^
  commit ^
  execute procedure set_context('skidder', 1) ^

  insert into journal(jrn_id) values(0) ^
  set term ;^
```

Poiché rdb$set_context riporta 1 o zero, può funzionare anche in una semplice frase SELECT:

**Esempio**

```
  SQL> select rdb$set_context('USER_SESSION', 'Nickolay', 'ru')
  CNT> from rdb$database;

  RDB$SET_CONTEXT
  ===============
                0
```

Il risultato uguale a 0 significa che precedentemente non era definita e che è stata impostata, in questo caso a 'ru'.

```
  SQL> select rdb$set_context('USER_SESSION', 'Nickolay', 'ca')
  CNT> from rdb$database;

  RDB$SET_CONTEXT
  ===============
                1
```

Il risultato 1 significa che era già definita e che l'abbiamo modificata in questo caso in 'ca'.

```
SQL> select rdb$set_context('USER_SESSION', 'Nickolay', NULL)
CNT> from rdb$database;

RDB$SET_CONTEXT
===============
              1
```

1 dice che era definita; cambiandola in NULL l'abbiamo cancellata.

```
SQL> select rdb$set_context('USER_SESSION', 'Nickolay', NULL)
CNT> from rdb$database;

RDB$SET_CONTEXT
===============
              0
```

A questo punto 0 significa che non è successo nulla: non esisteva prima e non esiste adesso.

## *Miglioramenti nella gestione dei PLAN specificati dall'utente*

D. Yemanov

1. Frammenti di PLAN sono propagati ai livelli di JOIN interni, permettendo l'ottimizzazione manuale di OUTER JOIN complesse
2. La correttezza di un PLAN specificato dall'utente viene verificata nelle OUTER JOIN
3. È stata aggiunta l'ottimizzazione di corto circuito per i PLAN specificati dall'utente
4. Un percorso d'accesso specificato dall'utente può essere assegnato ad ogni frase o clausola basata sulla SELECT

**Regole sintattiche**

Lo schema seguente descrive le regole sintattiche per la clausola PLAN per essere d'aiuto nella loro stesura.

```
PLAN ( { <stream_retrieval> | <sorted_streams> | <joined_streams> } )

<stream_retrieval> ::= { <natural_scan> | <indexed_retrieval> |
    <navigational_scan> }

<natural_scan> ::= <stream_alias> NATURAL

<indexed_retrieval> ::= <stream_alias> INDEX ( <index_name>
    [, <index_name> ...] )

<navigational_scan> ::= <stream_alias> ORDER <index_name>
    [ INDEX ( <index_name> [, <index_name> ...] ) ]

<sorted_streams> ::= SORT ( <stream_retrieval> )

<joined_streams> ::= JOIN ( <stream_retrieval>, <stream_retrieval>
    [, <stream_retrieval> ...] )
```

```
| [SORT] MERGE ( <sorted_streams>, <sorted_streams> )
```

**Dettagli**

*Natural scan* (scansione naturale) significa che tutte le righe vengono recuperate nell'ordine naturale di memorizzazione. Pertanto bisogna leggere tutte le pagine prima di poter validare un criterio di ricerca.

*Indexed retrieval* (accesso indicizzato) usa un particolare metodo indicizzato (range index scan) per trovare le righe che concordano con il criterio di ricerca dato. Le corrispondenze trovate sono combinate in una «mappa di bit sparsi» (sparse bitmap) che viene ordinata per numeri di pagina, in modo da dover leggere una pagina solo una volta. Dopo di ché le pagine dei dati vengono lette e da esse si recuperano le righe richieste.

*Navigational scan* (scansione in ordine) utilizza un indice per riportare le righe in certo ordine, se si può effettuare una tale operazione.

- L'indice ad albero binario viene percorso dal nodo più a sinistra a quello più a destra
- Se un qualsiasi criterio di ricerca è definito su una colonna specificata anche in una clausola ORDER BY, la navigazione è limitata da un qualche sottoalbero ristretto, che dipende dal predicato.
- Se un qualche criterio di ricerca è definito su colonne indicizzate, allora si effettua per prima cosa un range index scan, e ogni chiave recuperata ha il suo ID validato nel bitmap risultante. A quel punto si legge la pagina dati e si recupera la riga richiesta.

> **Nota**
>
> Una scansione in ordine (navigational scan) provoca una lettura in ordine sparso delle pagine, in quanto le letture non sono ottimizzate.

Una operazione di SORT (*sort operation*) effettua un ordinamento esterno del flusso di dati recuperato.

Una operazione di *JOIN* può essere effettuata sia attraverso un algoritmo ciclico innestato (JOIN plan) oppure con un algoritmo di sort merge (MERGE plan).

- Una JOIN con ciclo innestato interno (*inner nested loop join*) può contenere tutti quanti flussi si richiede di unire. Sono tutti fra loro equivalenti.
- Una JOIN con ciclo innestato esterno (*outer nested loop join*) opera sempre e solo con due flussi, pertanto nel caso di dover unire 3 o più flussi esterni, si vedranno clausole di JOIN innestate.

Un *sort merge* (ordina e riunisce) opera su due flussi in ingresso che sono dapprima ordinati e poi riuniti in una unica esecuzione.

**Esempi**

```
SELECT RDB$RELATION_NAME
FROM RDB$RELATIONS
WHERE RDB$RELATION_NAME LIKE 'RDB$%'
PLAN (RDB$RELATIONS NATURAL)
ORDER BY RDB$RELATION_NAME

SELECT R.RDB$RELATION_NAME, RF.RDB$FIELD_NAME
FROM RDB$RELATIONS R
  JOIN RDB$RELATION_FIELDS RF
  ON R.RDB$RELATION_NAME = RF.RDB$RELATION_NAME
PLAN MERGE (SORT (R NATURAL), SORT (RF NATURAL))
```

**Note**

1. Una clausola PLAN può essere messa in tutte le espressioni di selezione, incluse le subquery, tabelle derivate, e definizioni di viste. Può inoltre essere usata nelle frasi UPDATE e DELETE, perché sono implicitamente basate su espressioni di selezione.

2. Se una clausola PLAN contiene descrizioni di accesso non valide, allora o viene riportato un errore oppure la clausola errata viene silenziosamente ignorata: ciò dipende dalla gravità dell'errore.

3. Il tipo di PLAN specificato con ORDER <navigational_index> INDEX ( <filter_indices> ) viene riportato dal motore e può essere usato nei PLAN fatti dall'utente a partire da FB 2.0.

## Migliorie nelle operazioni di ordinamento

A. Brinkman

Nelle operazioni di ordinamento sono stati fatti alcuni utili sviluppi:

### GROUP BY e ORDER BY su un <alias>

I nomi di alias sono ora permessi in entrambe le clausole.

**Esempi:**

1. ORDER BY

```
SELECT RDB$RELATION_ID AS ID
FROM RDB$RELATIONS
ORDER BY ID
```

2. GROUP BY

```
SELECT RDB$RELATION_NAME AS ID, COUNT(*)
FROM RDB$RELATION_FIELDS
GROUP BY ID
```

### GROUP BY su espressione arbitraria

Una condizione di GROUP BY può ora contenere una qualsiasi espressione valida.

**Esempio**

```
...
  GROUP BY
  SUBSTRING(CAST((A * B) / 2 AS VARCHAR(15)) FROM 1 FOR 2)
```

### Ordinamento di SELECT * attraverso un numero implicito

L'ordinamento per grado (cioè per posizione del numero di colonna) adesso può funzionare anche con SELECT *

**Esempio**

```
SELECT *
  FROM RDB$RELATIONS
  ORDER BY 9
```

### *ORDER BY numerici e parametri: attenzione!*

Stando alle regole, a partire da Firebird 1.5, si può fare una ORDER BY <espressione>, e tale <espressione> può essere sia una variabile che un parametro. Si sarebbe tentati di pensare che fare un ORDER BY <numero> possa permettere sostituire <numero> con un parametro di input, o una espressione contenente un parametro.

Tuttavia, mentre il parser DSQL non respinge la clausola ORDER BY parametrizzata se risolve ad un intero, l'ottimizzatore richiede un valore costante assoluto per poter identificare la *posizione nella lista* generata della colonna o campo derivato su cui effettuare l'ordinamento. Se un parametro è accettato dal parser, il risultato è sottoposto ad un ordinamento fittizio, e il risultato non sarà ordinato.

## *L'espressione NEXT VALUE FOR*

D. Yemanov

L'espressione *NEXT VALUE FOR <sequence_name>* conforme a SQL-99 è stata aggiunta come sinonimo di GEN_ID(<generator-name>, 1), integrando l'inserimento della sintassi SQL standard *CREATE SEQUENCE* come equivalente di CREATE GENERATOR.

**Esempi**

1.

```
SELECT GEN_ID(S_EMPLOYEE, 1) FROM RDB$DATABASE;
```

2.

```
INSERT INTO EMPLOYEE (ID, NAME)
  VALUES (NEXT VALUE FOR S_EMPLOYEE, 'John Smith');
```

**Nota**

1. Attualmente, i valori di incremento diversi da 1 (uno) possono essere gestiti solo con la funzione GEN_ID. Si prevede nelle future versioni di supportare la gestione di generatori completamente secondo lo standard SQL-99, che permettono di specificare al livello DDL i richiesti intervalli di incremento. A meno che non ci sia la impellente necessità di usare un passo che non sia 1, si raccomanda di usare l'espressione NEXT VALUE FOR anziché la funzione GEN_ID.
2. GEN_ID(<name>, 0) permette di sapere qual'è il valore corrente di un generatore, ma non dovrebbe mai essere usato con frasi di INSERT o UPDATE, perché si corre il grosso rischio di violare il vincolo di unicità se ci sono processi concorrenti.

# Articoli

## *SELECT e sintassi delle espressioni*

Dmitry Yemanov

*La semantica*

- Una frase SELECT si usa per riportare dati al chiamante (un modulo PSQL o un programma client)
- Una espressione di selezione SELECT recupera le parti dai dati per costruire colonne che possono far parte o dell'insieme finale risultante, o di un qualsiasi insieme intermedio. Le espressioni di selezione sono note anche come subquery.

*Regole sintattiche*

```
<frase di select> ::=
  <espressione di select> [FOR UPDATE] [WITH LOCK]

<espressione di select> ::=
  <specifiche di query> [UNION [{ALL | DISTINCT}] <specifiche di query>]

<specifiche di query> ::=
  SELECT [FIRST <valore>] [SKIP <valore>] <lista di selezione>
  FROM <lista di espressioni tabellari>
  WHERE <condizioni di ricerca>
  GROUP BY <lista valori di raggruppamento>
  HAVING <condizioni di raggruppamento>
  PLAN <lista di elementi PLAN>
  ORDER BY <lista di valori per l'ordinamento>
  ROWS <value> [TO <value>]

<espressione tabellare> ::=
  <nome tabella> | <tabella in join> | <tabella derivata>

<tabella in join> ::=
  {<cross join> | <qualified join>}

<cross join> ::=
  <espressione tabellare> CROSS JOIN <espressione tabellare>

<qualified join> ::=
  <espressione tabellare> [{INNER | {LEFT | RIGHT | FULL} [OUTER]}] JOIN <espressione tabellar
  ON <join condition>

<tabella derivata> ::=
  '(' <espressione di select> ')'
```

*Conclusioni*

- La modalità FOR UPDATE ed il blocco di riga WITH LOCK possono essere effettuati solo sull'insieme di dati risultante, non possono essere applicati ad una subquery

- Le unioni con UNION sono possibili in qualsiasi subquery
- Le clausole FIRST, SKIP, PLAN, ORDER BY, ROWS sono permesse in ogni subquery

*Note*

- Può essere specificata nella stessa SELECT solo una fra FIRST/SKIP o ROWS, altrimenti mischiando le sintassi avviene un errore di sintassi
- Una frase INSERT accetta una espressione di selezione SELECT per definire un insieme di righe da inserire in una tabella. La parte di SELECT relativa supporta tutte le caratteristiche definite per le frasi ed espressioni di selezione.
- Le frasi di UPDATE e DELETE sono sempre basate su un cursore implicito applicato alla tabella specificata e limitato dalla clausola WHERE. Si possono inoltre specificare le parti finali della sintassi di selezione per limitare il numero di registrazioni modificate od ottimizzare il comando.

  Le clausole permesse alla fine di una frase di UPDATE o DELETE sono PLAN, ORDER BY e ROWS.

## *Tipo del dato risultante da un'aggregazione*

Arno Brinkman

Quando si effettua un'aggregazione per generare colonne con CASE o UNION con una miscela di tipi di dato compatibili, il motore deve scegliere un singolo tipo di dato per il risultato di ogni colonna. Chi sviluppa deve spesso preparare una variabile per depositarvi il risultato e può essere disorientato se avviene un'eccezione sul tipo di dato. Le seguenti regole sono quelle seguite per determinare il tipo di dato scelto dal motore per i valori di una colonna in tali condizioni.

1. Sia DTS l'insieme dei tipi di dato presenti in una colonna risultato di cui dobbiamo determinare il tipo di dato risultante (DTR).
2. Tutti i tipi di dato del DTS devono essere fra loro compatibili.
3. Nel caso in cui
   a. un qualsiasi valore del DTS sia una stringa di caratteri
      i. se tutti i dati del DTS sono di lunghezza fissa, allora anche il risultato è di lunghezza fissa; altrimenti il risultato è a lunghezza variabile.

         La lunghezza della stringa risultante, in caratteri, è uguale alla maggiore delle lunghezze, in caratteri, dei vari tipi di dato del DTS.

      ii. Il set di caratteri e l'ordinamento usati sono presi dal tipo di dato della prima stringa nel DTS.

   b. tutto il DTS contiene precisione numerica esatta

      il tipo risultante è in precisione numerica esatta con scala uguale alla massima delle scale presenti nel DTS e con precisione uguale alla massima precisione presente fra tutti i dati del DTS.

   c. un qualunque dato del DTS è in precisione numerica approssimata

      ogni dato del DTS deve essere numerico, altrimenti viene segnalato un errore.

d. un qualunque dato del DTS è del tipo data/ora

   tutti i dati del DTS devono essere *esattamente dello stesso tipo* di data/ora del primo, altrimenti viene segnalato un errore.

e. un qualunque dato del DTS è un BLOB

   tutti i dati del DTS devono essere BLOB e tutti dello stesso sotto-tipo.

## La sintassi abbreviata: trucchetto per i letterali di data

H. Borrie

Nel passato, prima dell'avvento delle variabili di contesto come CURRENT_DATE, CURRENT_TIMESTAMP, ecc. , si usavano delle convenzionali *stringhe letterali predefinite* (*predefined date literals*), come 'NOW', 'TODAY', 'YESTERDAY' e così via. Queste letterali predefinite sono sopravvissute nel linguaggio SQL attuale di Firebird e sono ancora molto utili.

In Interbase 5.x e precedenti, la seguente frase era «legale» e riportava un valore di tipo DATE ( ricordare che allora il tipo DATE era ciò che adesso è il tipo TIMESTAMP):

```
select 'NOW' from rdb$database /* riporta data ed ora di sistema */
```

In un database con ODS 10 o superiore, quella frase riporta solo la stringa 'NOW'. Bisogna pertanto forzare il tipo del dato con CAST per ottenere adesso il risultato voluto:

```
select cast('NOW' as TIMESTAMP) from rdb$database
```

Per molto tempo, probabilmente già da IB 6, c'è stata una non documentata «sintassi abbreviata» per forzare *qualsiasi* letterale, non solo quelli di data ed ora. Attualmente, è definito nello standard. Molti di noi semplicemente non sapevano di questa possibilità. Questa sintassi ha la forma `<tipo-di-dato> <letterale>`. Con l'esempio di CAST precedente in mente, l'espressione sintattica abbreviata è:

```
select TIMESTAMP 'NOW' from rdb$database
```

Questo tipo di sintassi abbreviata può essere utilizzata anche in altri modi. L'esempio seguente mostra un'operazione aritmetica sulle date con un letterale predefinito:

```
update mytable
  set OVERDUE = 'T'
  where DATE 'YESTERDAY' - DATE_DUE > 10
```

Capitolo 7

# SQL Procedurale (PSQL)

Una manciata di miglioramenti è stata aggiunta alle estensioni del PSQL di Firebird 2. In evidenza ci sono le nuove capacità ad usare i domini e gli ordinamenti alfabetici (collazioni) nella dichiarazione delle variabili e degli argomenti nelle procedure e nei trigger.

## Ricerca Rapida



## *Domini in PSQL*

Adriano dos Santos Fernandes

**(V.2.1)** Ora è possibile usare un dominio per dichiarare il tipo di dato degli argomenti e delle variabili nei moduli PSQL. In funzione dell'esigenza, si può dichiarare un argomento o una variabile usando in alternativa:

- il solo identificatore di dominio, al posto del tipo di identificatore nativo, in modo che la variabile erediti tutti gli attributi del dominio.
- il solo tipo di dato del dominio, senza ereditare pertanto i vincoli di CHECK ed il valore di DEFAULT, se dichiarati nel dominio, preponendo le parole `TYPE OF` nella dichiarazione (vedi sintassi qui sotto)

**Sintassi**

```
data_type ::=
    <builtin_data_type>
    | <domain_name>
    | TYPE OF <domain_name>
```

**Esempi**

```
CREATE DOMAIN DOM AS INTEGER;
```

```
CREATE PROCEDURE SP (
  I1 TYPE OF DOM,
  I2 DOM)
RETURNS (
  O1 TYPE OF DOM,
  O2 DOM)
AS
  DECLARE VARIABLE V1 TYPE OF DOM;
  DECLARE VARIABLE V2 DOM;

BEGIN
  ...
END
```

> **Nota**
>
> Il nuovo campo RDB$VALID_BLR è stato aggiunto alle tabelle RDB$PROCEDURES e RDB$TRIGGERS per indicare se la procedura o il trigger restano validi sopo una operazione di ALTER DOMAIN. Il valore di RDB$VALID_BLR viene mostrato in ISQL con i comandi SHOW PROCEDURE oppure SHOW TRIGGER.

## COLLATE nelle STORED PROCEDURE e nei parametri

A. dos Santos Fernandes

**(V.2.1)** Alle variabili PSQL e agli argomenti delle procedure, possono essere applicati gli ordinamenti (collazioni).

## Nelle viste può essere utilizzato WHERE CURRENT OF

Richiesta CORE-1213

**(V.2.1)** L'operatore sui cursori **WHERE CURRENT OF** può percorrere un cursore su un record set di una vista, come se fosse il cursore sul risultato di una SELECT su una tabella. Ad esempio:

```
...
FOR SELECT ...
   FROM MY_VIEW INTO ... AS CURSOR VIEW_CURSOR DO
BEGIN
  ...
  DELETE FROM MY_VIEW
    WHERE CURRENT OF VIEW_CURSOR;
  ...
END
```

## La variabile di contesto ROW_COUNT

D. Yemanov

**ROW_COUNT** è stata migliorata in modo tale che adesso riporta il numero di record recuperati da una frase SELECT.

Per esempio, può essere usata per verificare se una SELECT INTO ha effettivamente assegnato qualcosa:

```
..
BEGIN
  SELECT COL FROM TAB INTO :VAR;

  IF (ROW_COUNT = 0) THEN
    EXCEPTION NO_DATA_FOUND;
END
..
```

Vedere anche gli esempi più oltre sull'uso dei cursori espliciti in PSQL.

## Cursori espliciti

D. Yemanov

In PSQL si possono ora dichiarare ed usare cursori multipli. I cursori espliciti sono disponibili anche nelle strutture DSQL **EXECUTE BLOCK** oltre che nelle **STORED PROCEDURE** e nei **TRIGGER**.

**Sintassi**

```
  DECLARE [VARIABLE] <cursor_name> CURSOR FOR ( <select_statement> );
  OPEN <cursor_name>;
  FETCH <cursor_name> INTO <var_name> [, <var_name> ...];
  CLOSE <cursor_name>;
```

**Esempi**

1.

```
DECLARE RNAME CHAR(31);
DECLARE C CURSOR FOR (
  SELECT RDB$RELATION_NAME
  FROM RDB$RELATIONS
);
BEGIN
  OPEN C;
  WHILE (1 = 1) DO
  BEGIN
    FETCH C INTO :RNAME;
    IF (ROW_COUNT = 0) THEN
      LEAVE;
    SUSPEND;
  END
  CLOSE C;
END
```

2.

```
DECLARE RNAME CHAR(31);
DECLARE FNAME CHAR(31);
DECLARE C CURSOR FOR (
```

```
  SELECT RDB$FIELD_NAME
  FROM RDB$RELATION_FIELDS
  WHERE RDB$RELATION_NAME = :RNAME
  ORDER BY RDB$FIELD_POSITION
);
BEGIN
  FOR
    SELECT RDB$RELATION_NAME
    FROM RDB$RELATIONS
    INTO :RNAME
  DO
  BEGIN
    OPEN C;
    FETCH C INTO :FNAME;
    CLOSE C;
    SUSPEND;
  END
END
```

**Nota**

- La dichiarazione di un cursore è ammessa solo nella sezione delle dichiarazioni del PSQL (blocco, procedura o trigger che sia), come qualsiasi altra regolare dichiarazione di variabile locale.
- I nomi dei cursori devono essere univoci all'interno dello stesso contesto. Non devono entrare in conflitto con il nome di un altro cursore «annunciato» dalla clausola **AS CURSOR**, da un cursore **FOR SELECT**. Tuttavia un cursore può condividere il nome con ogni altro tipo di variabile nello stesso contesto, poiché le operazioni disponibili per ciascuno sono diverse.
- Si possono effettuare aggiornamenti e cancellazioni posizionali con i cursori attraverso la clausola **WHERE CURRENT OF**.
- Sono proibiti i tentativi di **FETCH** e **CLOSE** da un cursore **FOR SELECT**.
- Falliscono i tentativi di aprire un cursore già aperto e di recuperare i dati o chiudere un cursore già chiuso.
- Tutti i cursori non esplicitamente chiusi, verranno chiusi automaticamente all'uscita dall'attuale blocco/procedura/trigger PSQL.
- La variabile di sistema **ROW_COUNT** può essere usata dopo ogni frase FETCH per verificare se effettivamente è stata riportata una registrazione.

## *Argomenti di default nelle Stored Procedure*

V. Khorsun

Si possono dichiarare valori di default per gli argomenti delle STORED PROCEDURE.

La sintassi è identica alla definizione di un valore default per una colonna o per un dominio, eccetto che si può usare '=' al posto della parola chiave 'DEFAULT'.

Gli argomenti con valori di default devono venire per ultimi nella lista degli argomenti; cioè non si può dichiarare un argomento che non ha un default dopo un argomento che è stato dichiarato con un valore di default. Il chiamante deve comunque fornire tutti gli argomenti precedenti a quelli di cui vuole adoperare il proprio default.

Pertanto non si possono fare cose come fornire il gli primo e secondo argomento, non fornire il terzo argomento e fornire il quarto . . .

La sostituzione dei valori di default avviene durante l'esecuzione. Se è definita una procedura con default P1 e la si chiama da un'altra procedura P2 saltando alcuni dei parametri finali che hanno il default, allora i valori di default per P1 verranno sostituiti dal motore solo al momento in cui inizia l'esecuzione di P1. Questo significa che, cambiando i valori di default non è necessario ricompilare P2.

Tuttavia è sempre necessario scollegare tutte le connessioni client, come è descritto nel Borland InterBase 6 beta "Data Definition Guide" (DataDef.pdf), nella sezione "Altering and dropping procedures in use".

**Esempi**

```
CONNECT ... ;
SET TERM ^;
CREATE PROCEDURE P1 (X INTEGER = 123)
RETURNS (Y INTEGER)
AS
BEGIN
  Y = X;
  SUSPEND;
END ^
COMMIT ^
SET TERM ;^

SELECT * FROM P1;

           Y
============

         123

EXECUTE PROCEDURE P1;

           Y
============
         123

SET TERM ^;
CREATE PROCEDURE P2
RETURNS (Y INTEGER)
AS
BEGIN
  FOR SELECT Y FROM P1 INTO :Y
  DO SUSPEND;
END ^
COMMIT ^
SET TERM ;^

SELECT * FROM P2;

           Y
============

         123

SET TERM ^;
ALTER PROCEDURE P1 (X INTEGER = CURRENT_TRANSACTION)
         RETURNS (Y INTEGER)
AS
BEGIN
  Y = X;
 SUSPEND;
```

```
END; ^
COMMIT ^
SET TERM ;^

SELECT * FROM P1;

           Y
============

        5875

SELECT * FROM P2;

           Y
============

         123

COMMIT;

CONNECT  ... ;

SELECT * FROM P2;

           Y
============

        5880
```

**Nota**

1. Il sorgente ed il BLR per gli argomenti default sono memorizzati in RDB$FIELDS.
2. Come è stato evidenziato in una Tracker entry, gli esempi riportati sopra non devono essere presi come una raccomandazione per usare **SUSPEND** per avere i valori di ritorno da una STORED PROCEDURE di sola esecuzione (*executable* SP). L'autore ha usato **SUSPEND** qui al solo scopo esemplificativo per illustrare gli aspetti essenziali della nuova caratteristica.

## *Nuova sintassi LEAVE <label>*

D. Yemanov

La nuova sintassi `LEAVE <label>` permette di marcare i cicli PSQL con etichette e di uscirne in modo simile al linguaggio Java. Lo scopo è quello di terminare l'esecuzione del blocco corrente e riprendere dall'etichetta specificata. Dopodiché l'esecuzione prosegue dalla frase immediatamente successiva al ciclo terminato.

**Sintassi**

```
<nome_etichetta>: <ciclo>
...
LEAVE [<nome_etichetta>]
```

dove <ciclo> è uno fra: **WHILE**, **FOR SELECT**, **FOR EXECUTE STATEMENT**.

**Esempi**

1.

```
FOR
  SELECT COALESCE(RDB$SYSTEM_FLAG, 0), RDB$RELATION_NAME
    FROM RDB$RELATIONS
    ORDER BY 1
  INTO :RTYPE, :RNAME
  DO
  BEGIN
    IF (RTYPE = 0) THEN
      SUSPEND;
    ELSE
      LEAVE; -- esce dal ciclo corrente
  END
```

2.

```
CNT = 100;
L1:
WHILE (CNT >= 0) DO
BEGIN
  IF (CNT < 50) THEN
    LEAVE L1; -- esce dal ciclo WHILE
  CNT = CNT - l;
END
```

3.

```
STMT1 = 'SELECT RDB$RELATION_NAME FROM RDB$RELATIONS';
L1:
FOR
  EXECUTE STATEMENT :STMT1 INTO :RNAME
DO
BEGIN
  STMT2 = 'SELECT RDB$FIELD_NAME FROM RDB$RELATION_FIELDS
    WHERE RDB$RELATION_NAME = ';
  L2:
  FOR
    EXECUTE STATEMENT :STMT2 || :RNAME INTO :FNAME
  DO
  BEGIN
    IF (RNAME = 'RDB$DATABASE') THEN
      LEAVE L1; -- esce dal ciclo esterno
    ELSE IF (RNAME = 'RDB$RELATIONS') THEN
      LEAVE L2; -- esce dal ciclo interno
    ELSE
      SUSPEND;
  END
END
```

> **Nota**
>
> Notare che **LEAVE** senza una esplicita etichetta interrompe il ciclo attuale (il più interno).

## Ora le variabili di contesto OLD sono in sola lettura

D. Yemanov

L'insieme delle variabili di contesto OLD disponibili nei trigger sono ora in sola lettura. Un tentativo di assegnare un valore a OLD.QUALUNQUECAMPO viene rifiutato.

> **Nota**
>
> Le variabili di contesto NEW sono anch'esse adesso in sola lettura nei trigger AFTER.

## PSQL Stack Trace

V. Khorsun

Le API client possono ora estrarre un semplice *stack trace* dallo Status Vector di errore quando avviene un'eccezione nell'esecuzione di un PSQL (stored procedure o trigger). Uno stack trace è rappresentato da ina stringa (massimo 2048 byte) e consiste di tutti i nomi di STORED PROCEDURE e TRIGGER, a partire dal punto in cui è avvenuta l'eccezione, risalendo fino alla chiamata più esterna. Se la traccia è più lunga di quanto specificato al momento viene troncata.

Elementi aggiuntivi sono in coda allo Status Vector come segue:

```
isc_stack_trace, isc_arg_string, <string length>, <string>
```

`isc_stack_trace` è un nuovo codice di errore con valore 335544842L.

**Esempi**

Creazione di metdata:

```
CREATE TABLE ERR (
  ID INT NOT NULL PRIMARY KEY,
  NAME VARCHAR(16));

CREATE EXCEPTION EX '!';
SET TERM ^;

CREATE OR ALTER PROCEDURE ERR_1 AS
BEGIN
  EXCEPTION EX 'ID = 3';
END ^

CREATE OR ALTER TRIGGER ERR_BI FOR ERR
  BEFORE INSERT AS
BEGIN
  IF (NEW.ID = 2)
  THEN EXCEPTION EX 'ID = 2';

  IF (NEW.ID = 3)
  THEN EXECUTE PROCEDURE ERR_1;
```

```
  IF (NEW.ID = 4)
  THEN NEW.ID = 1 / 0;
END ^

CREATE OR ALTER PROCEDURE ERR_2 AS
BEGIN
  INSERT INTO ERR VALUES (3, '333');
END ^
```

1. Eccezione utente da un trigger:

```
SQL" INSERT INTO ERR VALUES (2, '2');
Statement failed, SQLCODE = -836
exception 3
-ID = 2
-At trigger 'ERR_BI'
```

2. Eccezione utente da una procedura chiamata da un trigger:

```
SQL" INSERT INTO ERR VALUES (3, '3');
Statement failed, SQLCODE = -836
exception 3
-ID = 3
-At procedure 'ERR_1'
At trigger 'ERR_BI'
```

3. Eccezione in esecuzione in un trigger (divisione per zero):

```
SQL" INSERT INTO ERR VALUES (4, '4');
Statement failed, SQLCODE = -802
arithmetic exception, numeric overflow, or string truncation
-At trigger 'ERR_BI'
```

4. Eccezione utente da una procedura:

```
SQL" EXECUTE PROCEDURE ERR_1;
Statement failed, SQLCODE = -836
exception 3
-ID = 3
-At procedure 'ERR_1'
```

5. Eccezione utente da una procedura ad un livello più profondo dello stack:

```
SQL" EXECUTE PROCEDURE ERR_2;
Statement failed, SQLCODE = -836
exception 3
-ID = 3
-At procedure 'ERR_1'
At trigger 'ERR_BI'
At procedure 'ERR_2'
```

## *Chiamare una UDF come una procedura*

N. Samofatov

In PSQL, le UDF supportate, ad esempio RDB$SET_CONTEXT, possono essere chiamate come se fossero funzioni che non riportano nulla, come le «procedure» dell'Object Pascal. Ad esempio:

```
BEGIN
   ...
   RDB$SET_CONTEXT('USER_TRANSACTION', 'MY_VAR', '123');
   ...
END
```

Capitolo 8

# Nuove parole chiave e modifiche

Le seguenti parole chiave sono state aggiunte o hanno cambiato il loro stato a partire da Firebird 1.5. Quelle marcate con un asterisco (*) non sono presenti nello standard SQL.

## Nuove parole riservate

```
BIT_LENGTH (v.2.0)
BOTH (v.2.0)
CHAR_LENGTH (v.2.0)
CHARACTER_LENGTH (v.2.0)
CLOSE (v.2.0)
CONNECT (v.2.1) <<—
CROSS (v.2.0)
DISCONNECT (v.2.1) <<—
FETCH (v.2.0)
GLOBAL  (v.2.1) <<—
INSENSITIVE  (v.2.1) <<—
LEADING (v.2.0)
LOWER (v.2.0)
OCTET_LENGTH (v.2.0)
OPEN (v.2.0)
RECURSIVE  (v.2.1) <<—
ROWS (v.2.0)
SENSITIVE  (v.2.1) <<—
START (v.2.1) <<—
TRAILING (v.2.0)
TRIM (v.2.0)
```

## Spostate da non riservate a riservate

```
USING (v.2.0)
```

## Parole chiave aggiunte come non riservate

```
ABS  (v.2.1) <<—
ACCENT * (v.2.1) <<—
ACOS * (v.2.1) <<—
```

```
ALWAYS * (v.2.1) <<—
ASCII_CHAR * (v.2.1) <<—
ASCII_VAL * (v.2.1) <<—
ASIN * (v.2.1) <<—
ATAN * (v.2.1) <<—
ATAN2 * (v.2.1) <<—
BACKUP * (v.2.0)
BIN_AND * (v.2.1) <<—
BIN_OR * (v.2.1) <<—
BIN_SHL * (v.2.1) <<—
BIN_SHR * (v.2.1) <<—
BIN_XOR * (v.2.1) <<—
BLOCK * (v.2.0)
CEIL (v.2.1) <<—
CEILING (v.2.1) <<—
COLLATION  (v.2.0)
COMMENT * (v.2.0)
COS * (v.2.1) <<—
COSH * (v.2.1) <<—
COT * (v.2.1) <<—
DATEADD * (v.2.1) <<—
DATEDIFF * (v.2.1) <<—
DECODE * (v.2.1) <<—
DIFFERENCE *  (v.2.0)
EXP (v.2.1) <<—
FLOOR (v.2.1) <<—
GEN_UUID (v.2.1) <<—
GENERATED (v.2.1) <<—
HASH * (v.2.1) <<—
IIF * (v.2.0)
LIST * (v.2.1) <<—
LN (v.2.1) <<—
LOG * (v.2.1) <<—
LOG10 * (v.2.1) <<—
LPAD * (v.2.1) <<—
MATCHED (v.2.1) <<—
MATCHING * (v.2.1) <<—
MAXVALUE * (v.2.1) <<—
MILLISECOND * (v.2.1) <<—
MINVALUE * (v.2.1) <<—
MOD (v.2.1) <<—
NEXT (v.2.0)
OVERLAY (v.2.1) <<—
PAD (v.2.1) <<—
PI * (v.2.1) <<—
PLACING (v.2.1) <<—
POWER (v.2.1) <<—
PRESERVE (v.2.1) <<—
RAND * (v.2.1) <<—
REPLACE * (v.2.1) <<—
RESTART (v.2.0)
RETURNING * (v.2.0)
REVERSE * (v.2.1) <<—
ROUND * (v.2.1) <<—
RPAD * (v.2.1) <<—
SCALAR_ARRAY * (v.2.0)
SEQUENCE (v.2.0)
SIGN * (v.2.1) <<—
SIN * (v.2.1) <<—
SINH * (v.2.1) <<—
SPACE (v.2.1) <<—
```

```
SQRT (v.2.1) <<–
TAN * (v.2.1) <<–
TANH * (v.2.1) <<–
TEMPORARY (v.2.1) <<–
TRUNC * (v.2.1) <<–
WEEK * (v.2.1) <<–
```

## Parole chiave non più riservate

```
ACTION (v.2.0)
CASCADE (v.2.0)
FREE_IT * (v.2.0)
RESTRICT (v.2.0)
ROLE (v.2.0)
TYPE (v.2.0)
WEEKDAY * (v.2.0)
YEARDAY * (v.2.0)
```

## Non più riservate come parole chiave

```
BASENAME * (v.2.0)
GROUP_COMMIT_WAIT * (v.2.0)
NUM_LOG_BUFS * (v.2.0)
CACHE * (v.2.0)
LOGFILE * (v.2.0)
RAW_PARTITIONS * (v.2.0)
CHECK_POINT_LEN * (v.2.0)
LOG_BUF_SIZE * (v.2.0)
```

Capitolo 9

# Indici e ottimizzazioni

## Ottimizzazioni in V.2.1

I miglioramenti nell'ottimizzatore in v.2.1 includono:

***(v.2.1)*** *Riduzione nelle letture dagli indici per MIN() e MAX()*

Le aggregazioni MIN/MAX indicizzate costringevano a tre letture degli indici mentre invece basta una sola. Così, con un indice ascendente sulla colonna non annullabile COL, la query

```
SELECT MIN(COL) FROM TAB
```

avrebbe dovuto essere completamente equivalente a

```
SELECT FIRST 1 COL FROM TAB
ORDER BY 1 ASC
```

effettuando entrambi una sola lettura di record. Tuttavia, la prima query richiedeva tre letture indicizzate mentre la seconda effettuava solo l'unica lettura prevista. Ora entrambe effettuano una sola lettura.

La stessa ottimizzazione si applica alla funzione MAX() se collegata ad un indice DESC.

## Clausola PLAN migliorata

D. Yemanov

**(V.2.0.x)** Una clausola PLAN permette opzionalmente di specificare le istruzioni di accesso ai dati al il motore e di fargli ignorare quelle calcolate dall'ottimizzatore. I miglioramenti in Firebird 2 permettono di specificare altri percorsi possibili al motore. Ad esempio:

```
PLAN (A ORDER IDX1 INDEX (IDX2, IDX3))
```

Per altri dettagli, fare riferimento all'argomento Miglioramenti ai PLAN nel capitolo del DML.

## Miglioramenti all'ottimizzatore

Questa sezione presenta l'insieme delle modifiche effettuate in Firebird 2 per migliorarne le prestazioni.

## Per tutti i database

Il primo gruppo di modifiche influenza versione ODS del database, inclusi pertanto quelli non ancora aggiornati alla ODS 11.x.

### Miglioramenti vari

O. Loa, D. Yemanov

- Algoritmi più rapidi nel gestire l'albero delle pagine «sporche»

  Firebird 2 offre una gestione più efficiente delle pagine modificate, il cosiddetto «albero delle pagine sporche». Ciò influenza tutti i tipi di modifiche dei dati effettuate in una singola transazione ed elimina alcuni noti problemi di rallentamento delle prestazione quando si ha una buffer di cache maggiore di 10K pagine.

  Questo inoltre migliora le prestazioni in generale nella modifica dei dati.

- La cache delle pagine è stata aumentata a 128K pagine (2GB nel caso di pagine da 16Kbyte)

### Calcoli più veloci per IN() ed OR

O. Loa

Il predicato IN con costanti o più valori booleani in OR fra loro sono calcolati più rapidamente.

Le operazioni con mappe di bit sparsi sono state ottimizzate per gestire i predicati OR e IN (<lista di costanti>) più efficacemente, migliorando le prestazioni su queste operazioni.

### Ricerca su UNIQUE migliorato

A. Brinkman

L'ottimizzatore ora usa un costo più realistico per accedere attraverso un indice UNIQUE.

### Altre ottimizzazioni sulle condizioni NOT

D. Yemanov

Le condizioni con NOT sono state semplificate ed ottimizzate attraverso l'uso di un indice quando possibile.

**Esempio**

```
(NOT NOT A = 0) -> (A = 0)
(NOT A > 0) -> (A <= 0)
```

### Distribuzione di HAVING alla clausola WHERE

Se la clausola HAVING o una SELECT di livello più esterno riferiscono ad un campo da raggruppare, questa congiunzione viene distribuita più in profondità nel percorso esecutivo rispetto al raggruppamento, permettendo così la scansione di un indice. In altre parole, permette alla clausola HAVING non solo di essere utilizzata in questo caso come una WHERE, ma anche di essere ottimizzata allo stesso modo.

**Esempi**

```
select rdb$relation_id, count(*)
from rdb$relations
group by rdb$relation_id
having rdb$relation_id > 10

select * from (
  select rdb$relation_id, count(*)
  from rdb$relations
  group by rdb$relation_id
  ) as grp (id, cnt)
where grp.id > 10
```

In entrambi questi casi, si effettua una scansione attraverso l'indice invece che una lettura completa.

### Distribuzione di UNION ai percorsi interni

Questo tipo di distribuzione viene fatto sempre che ne esista la possibilità.

### CROSS JOIN e MERGE/SORT

Gestione migliorata per cross join e merge/sort,

### Ordine delle join con INNER e OUTER insieme

Quando ci sono più JOIN sia INNER che OUTER mescolate fra loro si cerca di raggrupparle nel modo migliore possibile

### Espressioni con uguaglianze

Ora possono essere generati MERGE PLAN nelle join che confrontano per uguaglianza le espressioni.

## Per i soli database in ODS 11

Questo gruppo di ottimizzazioni influenza solo i database creati o recuperati in sotto Firebird 2 o successivi.

### Uso della selettività a livello di segmento

Vedere Selectivity Maintenance per Segment.

### Supporto migliore per IS NULL e STARTING WITH

Prima, i predicati IS NULL e STARTING WITH venivano ottimizzati separatamente dagli altri, provocando PLAN non ottimali in certe espressioni booleane complesse con molti AND e OR. A partire dalla 2.0 con ODS11, questi predicati sono gestiti in modo normale e pertanto possono beneficiare di tutte le possibili strategie di ottimizzazione.

### Utilizzo degli indici per OR ed AND

Espressioni booleane complesse con molti predicati AND ed OR ora possono sfruttare a pieno l'utilizzo di sottostanti indici se possibile. Precedentemente, tali espressioni complesse potevano essere ottimizzate malamente.

### Miglior ordinamento per le JOIN

Si è lavorato per migliorare la stima del costo per migliorare l'ordine delle JOIN.

### Nelle OUTER JOIN abilitato l'accesso indicizzato

Si può utilizzare un accesso indicizzato nelle OUTER JOIN, cioè un attraversamento guidato.

## Miglioramenti agli indici

### Superato il limite a 252-byte per indice

A. Brinkman

Il codice, in parte nuovo ed in parte rifatto, che gestisce gli indici è molto più veloce e tollera un gran numero di duplicazioni. Il vecchio limite per le chiavi aggregate di 252 bytes è stato rimosso. Ora il limite dipende dalla dimensione della pagina: la dimensione massima della chiave in byte è 1/4 della dimensione della pagina (512 su 2048, 1024 su 4096, ecc.)

Un numero di record a 40-bit viene incluso nei rami («non leaf-level pages») ed i duplicati nelle chiavi sono ordinati secondo questo numero.

### Espressioni negli indici

O. Loa, D. Yemanov, A. Karyakin

Si possono indicizzare le righe attraverso espressioni arbitrarie in DDL dinamico applicate ai valori della riga, permettendo ai predicati di ricerca percorsi di accesso basati su espressioni.

**Sintassi**

```
CREATE [UNIQUE] [ASC[ENDING] | DESC[ENDING]] INDEX <nome indice>
  ON <nome tabella>
  COMPUTED BY ( <espressione> )
```

**Esempi**

1.

```
CREATE INDEX IDX1 ON T1
  COMPUTED BY ( UPPER(COL1 COLLATE PXW_CYRL) );
```

```
COMMIT;
/*  */
SELECT * FROM T1
  WHERE UPPER(COL1 COLLATE PXW_CYRL) = 'ÔÛÂÀ'
-- PLAN (T1 INDEX (IDX1))
```

2.

```
CREATE INDEX IDX2 ON T2
  COMPUTED BY ( EXTRACT(YEAR FROM COL2) || EXTRACT(MONTH FROM COL2) );
COMMIT;
/*  */
SELECT * FROM T2
  ORDER BY EXTRACT(YEAR FROM COL2) || EXTRACT(MONTH FROM COL2)
-- PLAN (T2 ORDER IDX2)
```

> **Nota**
>
> 1. L'espressione usata nel predicato della condizione deve essere *identico* all'espressione usata nella dichiarazione dell'indice, affinché il motore possa scegliere il percorso con accesso indicizzato. L'indice definito non sarebbe utilizzabile per l'accesso o l'ordinamento se le espressioni sono diverse.
> 2. Le espressioni per gli indici hanno le stesse caratteristiche e limitazioni degli altri indici, eccetto il fatto che, per definizione, non possono essere composite o multi segmento.

## Modifiche alla gestione dei NULL

V. Khorsun, A. Brinkman

- Le chiavi a NULL sono ignorate per il controllo di unicità. (V. Khorsun)

  Se un nuovo valore è introdotto in un indice univoco, il motore salta tutte le chiavi a NULL prima di verificare se la chiave è duplicata. Questo da' un miglioramento nelle prestazioni, in quanto a partire dalla v.1.5, i NULL non vengono più considerati duplicati.

- I NULL sono ignorati nello scorrimento dell'indice, quando ha senso ignorarli. (A. Brinkman)

  Precedentemente, le chiavi a NULL erano sempre lette per tutti i predicati. A partire dalla v.2.0, le chiavi a NULL sono generalmente ignorate all'inizio della scansione, permettendo una scansione dell'indice più veloce.

> **Nota**
>
> I predicati IS NULL e IS NOT DISTINCT FROM richiedono ancora comunque la l'accesso alle chiavi a NULL, e pertanto disabilitano la summenzionata ottimizzazione.

## Miglioramenti nella compressione degli indici

A. Brinkman

L'algoritmo che comprime gli indici è stato completamente rifatto migliorando grandemente le prestazioni di moltissime query.

## *Manutenzione della selettività per segmento*

D. Yemanov, A. Brinkman

Le selettività degli indici sono ora memorizzate per ogni segmento. Questo significa che per un indice composito sulle colonne (A, B, C), vengono calcolati tre indici di selettività, che riflettono sia un riscontro totale sull'indice che tutti i parziali. Sarebbe a dire che la selettività dell'indice multi segmento coinvolge sia quella del solo segmento A (come sarebbe se ci fosse un indice a segmento unico), dei segmenti A e B insieme (come sarebbe se ci fosse in indice con segmento doppio) ed il riscontro con tutti e tre i segmenti (A, B, C); praticamente in tutti i modi si possa usare un indice composto.

Questo lascia più opportunità all'ottimizzatore per migliori decisioni sui percorsi da utilizzare nei casi in cui si cerchino riscontri su indici parziali.

I valori di selettività per segmento sono memorizzati nella colonna RDB$STATISTICS della tabella RDB $INDEX_SEGMENTS. La colonna con lo stesso nome in RDB$INDICES è mantenuta per sola compatibilità e tuttora rappresenta l'indice di selettività totale, che viene usato per un riscontro su tutto l'indice.

Capitolo 10

# Supporto alle lingue internazionali (INTL)

Adriano dos Santos Fernandes

Questo capitolo descrive la nuova interfaccia al supporto per le lingue internazionali che è stato introdotto con Firebird 2. A partire da allora, sono state aggiunte funzionalità e migliorie, inclusa la possibilità di implementare generici ordinamenti UNICODE da librerie esterne. Inoltre è stata aggiunta una sintassi DDL per applicare tutto ciò, nella forma della frase CREATE COLLATION.

## Interfaccia INTL per i set di caratteri non ASCII

A. dos Santos Fernandes

Definita inizialmente da N. Samofatov, la nuova interfaccia di Firebird 2 per i caratteri internazionali ha molte nuove funzionalità sviluppate dall'autore.

### *Architettura*

Firebird permette di definire il set di caratteri e l'ordinamento in qualsiasi dichiarazione di campo o variabile a caratteri. Il set di caratteri di default può essere inoltre specificato al momento della creazione del database, in modo che ogni dichiarazione CHAR o VARCHAR usi quel default se non include una specifica clausola CHARACTER SET.

Alla connessione normalmente si specifica il set di caratteri che il *client* intende usare per leggere le stringhe. Se non viene specificato un set di caratteri "client" (o di "connessione"), si impone il set di caratteri NONE.

Due speciali set di caratteri, NONE e OCTETS, possono essere utilizzati nelle dichiarazioni. Tuttavia, OCTETS non può essere specificato per le connessioni. I due set sono simili, tranne che per NONE il carattere spazio è ASCII 0x20, mentre per OCTETS è 0x00.

NONE e OCTETS sono «speciali» nel senso che usano nelle conversioni regole differenti da quelle degli altri set di caratteri.

- Negli altri set di caratteri le conversioni sono effettuate in questo modo: CHARSET1->UNICODE->CHARSET2.
- Con NONE/OCTETS i bytes che compongono le stringhe sono copiati direttamente così come sono : NONE/OCTETS->CHARSET2 e CHARSET1->NONE/OCTETS.

### *Miglioramenti*

I miglioramenti apportati da questo nuovo sistema includono:

### Controllo sulla corretta formazione

Alcuni set di caratteri (specialmente a byte multipli) non accettano proprio qualsiasi stringa. Ora, il sistema verifica che le stringhe siano corrette quando le assegna da NONE/OCTETS e quando le stringhe sono inviate al client (sia come statement che come parametri).

### Maiuscole

In Firebird 1.5.x, solo i caratteri ASCII sono convertiti in maiuscole nell'ordinamento default (binario) di qualsiasi set di caratteri, che è quello usato quando non viene specificata la COLLATION.

Per esempio,

```
isql -q -ch dos850
SQL> create database 'test.fdb';
SQL> create table t (c char(1) character set dos850);
SQL> insert into t values ('a');
SQL> insert into t values ('e');
SQL> insert into t values ('á');
SQL> insert into t values ('é');
SQL>
SQL> select c, upper(c) from t;

C      UPPER
====== ======
     a      A
     e      E
     á      á
     é      é
```

In Firebird 2 il risultato è invece questo:

```
C      UPPER
====== ======
     a      A
     e      E
     á      Á
     é      É
```

### Lunghezza massima di una stringa

In Firebird 1.5.x non viene verificata la lunghezza logica di una stringa di un set di caratteri multi byte (MBCS). Pertanto, un campo UNICODE_FSS prende tre volte il numero di caratteri della dimensione dichiarata del campo, essendo tre byte la lunghezza massima di un carattere UNICODE_FSS.

Questo è stato mantenuto per compatibilità solo per i set di caratteri preesistenti. Tuttavia i nuovi set di caratteri, ad esempio UTF8, non mantengono questo limitazione.

### sqlsubtype ed il set di caratteri per la connessione

Quando il set di caratteri di un campo CHAR o VARCHAR non è né NONE né OCTETS, ed il set di caratteri della connessione non è NONE, il membro *sqlsubtype* di una XSQLVAR pertinente a quella colonna ora contiene il numero del set di caratteri della connessione e non quello del set di caratteri della colonna dichiarato nel DDL.

### Miglioramenti per i BLOB

Sono stati fatti diversi miglioramenti per la gestione dei BLOB di testo.

#### Clausola COLLATE per i BLOB

Si può specificare una nuova clausola DML COLLATE ai BLOB.

**Esempio**

```
select blob_column from table
  where blob_column collate unicode = 'foo';
```

#### Confronti sul contenuto completo dei BLOB

Si possono effettuare operazioni di confronto sull'intero contenuto di un BLOB di testo.

#### Conversioni sul set di caratteri dei BLOB

Si possono effettuare conversioni fra i set di caratteri durante l'assegnazione di un BLOB ad una stringa o ad un altro BLOB

## INTL Plug-in

I set di caratteri e gli ordinamenti sono installati usando un *manifest file* (documento di configurazione in formato particolare).

Il manifest file deve essere messo nel sottodirettorio `intl` dell'installazione (cioè `$rootdir/intl` in Linux o `C:\Programmi\Firebird\Firebird_2_1\intl` in Windows) con l'estensione `.conf`. Viene usato per trovare i set di caratteri e gli ordinamenti nelle librerie. Se un set di caratteri o un ordinamento sono elencati più di una volta, non viene caricato e viene inserito un errore nel log.

Il file `/intl/fbintl.conf` è un esempio di manifest file da cui sono state estratte le righe seguenti:

```
<intl_module fbintl>
    filename    $(this)/fbintl
</intl_module>

<charset ISO8859_1>
    intl_module    fbintl
    collation    ISO8859_1
    collation    DA_DA
    collation    DE_DE
```

```
    collation    EN_UK
    collation    EN_US
    collation    ES_ES
    collation    PT_BR
    collation    PT_PT
</charset>

<charset WIN1250>
    intl_module    fbintl
    collation    WIN1250
    collation    PXW_CSY
    collation    PXW_HUN
    collation    PXW_HUNDC
</charset>
```

> **Nota**
>
> Il simbolo $(this) viene usato per indicare la directory in cui è il manifest file stesso. Inoltre l'estensione della libreria va omessa.

## Nuovi set di caratteri ed ordinamenti

Due nuovi set di caratteri aggiunti in Firebird 2 sono di particolare interesse per chi ha litigato in passato con i pasticci generati da UNICODE_FSS nelle versioni precedenti.

### Set di caratteri UTF8

Il set di caratteri UNICODE_FSS ha diversi problemi: è una vecchia versione di UTF8 che accetta stringhe malformate e non determina la lunghezza massima corretta di una stringa. In FB 1.5.X UTF8 è di fatto un alias di UNICODE_FSS.

Ora, UTF8 è un nuovo set di caratteri, senza i problemi correlati a UNICODE_FSS.

### Ordinamenti UNICODE (per UTF8)

UCS_BASIC ordina come UTF8 senza nessuna specifica (sarebbe UNICODE in ordine binario). L'ordinamento UNICODE ordina invece usando l'UCA (Unicode Collation Algorithm).

**Esempio di ordinamenti:**

```
isql -q -ch dos850
SQL> create database 'test.fdb';
SQL> create table t (c char(1) character set utf8);
SQL> insert into t values ('a');
SQL> insert into t values ('A');
SQL> insert into t values ('á');
SQL> insert into t values ('b');
SQL> insert into t values ('B');
SQL> select * from t order by c collate ucs_basic;

C
======
A
```

```
B
a
b
á

SQL> select * from t order by c collate unicode;

C
======
a
A
á
b
B
```

## Sviluppi nella V.2.1

La versione 2.1 aggiunge delle funzionalità relative a

1. usare i set di caratteri ICU definendoli in `fbintl.conf`
2. L'ordinamento UNICODE (charset_UNICODE) è disponibile per tutti set di caratteri in *fbintl*
3. attributi per usare gli ordinamenti
4. Frasi SQL per CREATE/DROP COLLATION
5. Comando SHOW COLLATION ed estrazione degli ordinamenti in ISQL
6. Verifica della buona formattazione dei BLOB di testo
7. Translitterazione dei BLOB di testo in automatico

## Set di caratteri ICU

Tutti i set di caratteri non grandi e basati su ASCII presenti in ICU possono essere usati da Firebird 2.1. Per ridurre la dimensione del kit in distribuzione, si è personalizzato ICU per includere solo i set di caratteri essenziali e quelli per i quali c'è stata una specifica richiesta.

Se il set di caratteri di cui si ha bisogno non è incluso, si possono sostituire le librerie ICU con altri moduli, reperibili al nostro sito o già installati nel sistema operativo.

### Registrare un modulo per set di caratteri ICU

Per usare un modulo per set di caratteri alternativo, bisogna registrarlo in due parti:

1. nel file di configurazione dei linguaggi del server, `intl/fbintl.conf`
2. in ogni database che richiede di usarlo

#### Registrare un set di caratteri nel server

Usando un editor di testi, registrare il nuovo set di caratteri nel manifest file `intl/fbintl.conf`, come segue:

```
<charset         NAME>
    intl_module     fbintl
    collation       NAME [REAL-NAME]
</charset>
```

### Registrare un set di caratteri in un database

Per registrare il modulo in un database, lanciare la procedura `sp_register_character_set`, il cui sorgente si trova nel sottodirettorio `misc/intl.sql` della radice di Firebird 2.1

### Uso della Stored Procedure

**Un esempio**

Questa è ad esempio una dichiarazione in `fbintl.conf`:

```
<charset         GB>
    intl_module     fbintl
    collation       GB GB18030
</charset>
```

La STORED PROCEDURE ha due parametri: una stringa che è l'identificatore del set di caratteri come dichiarato nel file di configurazione ed un intero che è il massimo numero di byte che occupa un singolo carattere in quella codifica. Nel nostro esempio:

```
execute procedure sp_register_character_set ('GB', 4);
```

### La frase CREATE COLLATION

**Sintassi per CREATE COLLATION**

```
CREATE COLLATION <name>
    FOR <charset>
    [ FROM <base> | FROM EXTERNAL ('<name>') ]
    [ NO PAD | PAD SPACE ]
    [ CASE SENSITIVE | CASE INSENSITIVE ]
    [ ACCENT SENSITIVE | ACCENT INSENSITIVE ]
    [ '<specific-attributes>' ]
```

> **Nota**
>
> Attributi specifici devono essere separati da punto e virgola (cioè ';' ) e sono sensibili alle maiuscole.

**Esempi**

```
/* 1 */
CREATE COLLATION UNICODE_ENUS_CI
    FOR UTF8
    FROM UNICODE
```

```
    CASE INSENSITIVE
    'LOCALE=en_US';
/* 2 */
CREATE COLLATION NEW_COLLATION
    FOR WIN1252
    PAD SPACE;

/* NEW_COLLATION deve essere dichiarata nel file .conf
   bel direttorio $root/intl  */
```

## Gli ordinamenti UNICODE

Gli ordinamenti UNICODE (sia sensibili che insensibili alle maiuscole) possono essere applicati a tutti i set di caratteri presenti in fbintl. Essi sono già registrati in `fbintl.conf`, ma bisogna registrarli nel database con le associazioni ed attributi prescelti.

**Convenzioni nel nome**

Per convenzione il nome da usare è nella forma `<nome set caratteri>_<ordinamento>`. Ad esempio,

```
create collation win1252_unicode
  for win1252;

create collation win1252_unicode_ci
  for win1252
   from win1252_unicode
   case insensitive;
```

> **Nota**
>
> Il nome del set di caratteri deve essere identico a come specificato in `fbintl.conf` (ad esempio ISO8859_1 e non ISO88591).

## Attributi specifici per gli ordinamenti

> **Nota**
>
> Alcuni attributi possono non funzionare con certi ordinamenti, anche se non riportano errori.

*DISABLE-COMPRESSIONS*
: Disabilita le contrazioni, cambiando di fatto l'ordine di un gruppo di caratteri.

  Valido per i set di caratteri ad un byte.

  Formato: DISABLE-COMPRESSIONS={0 | 1}

  **Esempio**

  DISABLE-COMPRESSIONS=1

*DISABLE-EXPANSIONS*

Disabilita le espansioni, cambiando l'ordine di un carattere in un gruppo di caratteri.

Valido per ordinamenti su set di caratteri ad un byte.

Formato: DISABLE-EXPANSIONS={0 | 1}

**Esempio**

DISABLE-EXPANSIONS=1

*ICU-VERSION*
Specifica quale versione della libreria ICU deve essere usata. I valori validi, sono quelli definiti in nel file di configurazione (`intl/fbintl.conf`) nella dichiarazione intl_module/icu_versions.

Valido per UNICODE e UNICODE_CI.

Formato: ICU-VERSION={default | major.minor}

**Esempio**

ICU-VERSION=3.0

*LOCALE*
Specifica un ordinamento locale.

Valido per UNICODE e UNICODE_CI. Richiede la versione completa delle librerie ICU.

Formato: LOCALE=xx_XX

**Esempio**

LOCALE=en_US

*MULTI-LEVEL*
Usa più di un livello per effettuare gli ordinamenti.

Valido per ordinamenti su set di caratteri ad un solo byte.

Formato: MULTI-LEVEL={0 | 1}

**Esempio**

MULTI-LEVEL=1

*SPECIALS-FIRST*
Ordina i caratteri speciali (spazi, simboli, ecc.) prima dei caratteri alfanumerici.

Valido per ordinamenti su set di caratteri a singolo byte.

Formato: SPECIALS-FIRST={0 | 1}

**Esempio**

SPECIALS-FIRST=1

### Modifiche agli ordinamenti in V.2.1

*Spagnolo*
L'ordinamento ES_ES (come il nuovo ES_ES_CI_AI) automaticamente usa gli attributi DISABLE-COMPRESSIONS=1; SPECIALS-FIRST=1.

> **Nota**
>
> Gli attributi sono memorizzati al momento della creazione del database, pertanto tale modifica non si applica a database con ODS precedente alla 11.1.

L'ordinamento ES_ES_CI_AI è stato standardizzato all'uso corrente.

*UTF-8*
Ordinamento insensibile alle maiuscole per UTF-8. Vedere la richiesta CORE-972

## Conversione del testo dei meta-dati

La versione 2.0.x di Firebird aveva due problemi relativi al set di caratteri ed all'estrazione dei meta-dati:

1. Creando o modificando gli oggetti, il testo associato ai meta-dati non viene traslitterato dal set di caratteri del Client a quello del sistema (UNICODE_FSS) per tali colonne BLOB. Invece venivano memorizzati i bytes così come stavano, nudi e crudi.

   I tipi di testo influenzati da questa cosa erano i sorgente PSQL, le descrizioni, testi associati a vincoli ed indici, e così via.

   > **Nota**
   >
   > Anche nella versione attuale (2.1.x) il problema può ancora accadere se un'operazione CREATE oppure ALTER viene effettuata col set di caratteri della connessione a NONE o UNICODE_FSS e si usano dati non UNICODE_FSS.

2. Leggendo i BLOB di testo, la translitterazione dal set di caratteri del BLOB al set di caratteri client non veniva effettuata.

### Riparare il testo dei meta-dati

Se i meta-dati del tuo database sono stati creati con una codifica non ASCII, bisogna riparare il database per poter leggere i meta-dati correttamente dopo un aggiornamento alla versione 2.1.

**Importante**

La procedura e composta da più passaggi attraverso il database, con l'utilizzo di alcuni script. Si raccomanda fortemente di disconnettersi e riconnettersi al database dopo ciascun passo.

Il database deve essere già stato portato in ODS11.1 attraverso un ciclo di backup e di restore utilizzando gbak.

Ovviamente, prima di iniziare a fare qualsiasi cosa, farsi una copia di sicurezza del database.

Nell'esempio seguente, *$fbroot$* rappresenta il percorso del direttorio radice dell'installazione di Firebird, ad esempio `/opt/firebird` su Linux oppure `C:\Programmi\Firebird\Firebird_2_1` su Windows.

Gli esempi indicano i percorsi per Linux. Su Windows vanno indicati secondo il suo standard.

### Creare le procedure nel database

```
[1] isql /percorso/al/tuo/database.fdb
[2] SQL> input '$fbroot$/misc/upgrade/metadata/metadata_charset_create.sql';
```

### Verificare il database

```
[1] isql /percorso/al/tuo/database.fdb
[2] SQL> select * from rdb$check_metadata;
```

La procedura *rdb$check_metadata* elenca tutti gli oggetti che verifica.

- Se non viene elevata nessuna eccezione, i meta-dati del database sono a posto e si può proseguire alla sezione « Rimuovere le procedure d'aggiornamento ».
- Altrimenti, l'ultimo oggetto listato prima dell'eccezione è quello che ha problemi.

### Aggiustare i meta-dati

Per sistemare i meta-dati, bisogna sapere in che set di caratteri è stato creato l'oggetto. Lo script d'aggiornamento funziona correttamente solo se tutti i meta-dati sono stati creati usando lo stesso set di caratteri.

```
[1] isql /percorso/al/tuo/database.fdb
[2] SQL> input '$fbroot$/misc/upgrade/metatdata/metadata_charset_create.sql';
[3] SQL> select * from rdb$fix_metadata('WIN1251');  -- sostituire WIN1251 col set di caratteri
[4] SQL> commit;
```

La procedura *rdb$fix_metadata* riporta le stesse informazioni di *rdb$check_metadata*, ma cambia i testi dei meta-dati.

**Importante**

Deve essere lanciato una ed una sola volta!

Dopo ciò si possono rimuovere le procedure di aggiornamento.

### *Rimuovere le procedure di aggiornamento*

```
[1] isql /percorso/al/tuo/database.fdb
[2] SQL> input '$fbroot$/misc/upgrade/metadata/metadata_charset_drop.sql';
```

## Set di caratteri supportati

Vedere Appendice B alla fine di queste note di rilascio, per la lista completa di tutti i set di caratteri supportati.

Capitolo 11

# Caratteristiche amministrative

Firebird gradualmente aggiunge sempre nuove caratteristiche per snellire l'amministrazione delle basi di dati. Firebird 2.1 vede l'introduzione di una nuova famiglia di tabelle di sistema attraverso le quali gli amministratori possono monitorare all'istante le transazioni e i comandi attivi in una base di dati. Questi strumenti adottano una nuova caratteristica del DDL della versione 2.1, Tabelle temporanee globali (Global Temporary Tables) per dare queste istantanee.

## Tabelle di monitoraggio

Dmitry Yemanov

Firebird 2.1 presenta la possibilità di monitorare dalla parte del server l'attività esistente relativa ad un certo database. Il motore offre un insieme di cosiddette «tabelle virtuali» che danno all'utente una fotografia delle attività in corso per una certa base di dati.

La parola «virtuale» significa che il la tabella in questione non è materializzata se non quando esplicitamente richiesto. Tuttavia, il relativo metadata è stabilmente accessibile e consultabile nello schema.

> **Nota**
>
> Le tabelle virtuali di monitoraggio esistono solo per i database con la ODS 11.1 (e successive), pertanto la migrazione con un ciclo di backup e restore è necessaria prima di poter utilizzare questa caratteristica.

### *L'idea*

Il punto chiave del sistema di monitoraggio è un'istantanea delle attività (*activity snapshot*). Essa rappresenta lo stato corrente del database, comprese una serie di informazioni sul DB stesso, sui collegamenti e gli utenti attivi, le transazioni, i comandi preparati ed in esecuzione, ed altro ancora.

Questa istantanea viene creata nel momento stesso in cui una qualsiasi delle tabelle di monitoraggio viene richiesta in una data transazione e viene conservata fino al termine della transazione, in modo tale che le interrogazioni multitabellari (esempio quelle master-slave) a tali tabelle possano riportare sempre una vista consistente dei dati.

In altre parole, le tabelle di monitoraggio si comportano sempre come in una transazione snapshot table stability («consistency»), anche se la transazione è partita con un livello di isolamento più basso.

Per rinfrescare l'istantanea (o snapshot) la transazione corrente dovrebbe essere terminata e le tabelle di monitoraggio vanno reinterrogate nel contesto di una nuova transazione.

### Monitoraggio di connessioni multiple

D. Yemanov

Nelle versioni 2.1.0 e 2.1.1, un utente generico (non SYSDBA) con più di una connessione avrebbe potuto monitorare solo quanto appartenente alla CURRENT_CONNECTION. A partire dalla 2.1.2 *(miglioria CORE-2233)* , un utente generico (non SYSDBA) può controllare tutte le sue connessioni attive contemporaneamente.

## Scopo e sicurezza

- L'accesso alle tabelle di monitoraggio è permesso in DSQL e PSQL.
- Il monitoraggio completo è permesso a SYSDBA e al proprietario del database.
- Gli utenti normali sono vincolati alle informazioni relative alla propria connessione, cioè le altre connessioni sono a loro invisibili.

## Metadata

*MON$DATABASE (database collegato)*

- MON$DATABASE_NAME (database pathname o alias)
- MON$PAGE_SIZE (page size)
- MON$ODS_MAJOR (parte intera del numero di versione ODS)
- MON$ODS_MINOR (parte frazionaria del numero di versione ODS)
- MON$OLDEST_TRANSACTION (numero della Oldest Interesting Transaction)
- MON$OLDEST_ACTIVE (numero della Oldest Active Transaction)
- MON$OLDEST_SNAPSHOT (numero dell'Oldest Snapshot Transaction)
- MON$NEXT_TRANSACTION (successivo numero di transazione)
- MON$PAGE_BUFFERS (numero di pagine allocate in cache)
- MON$SQL_DIALECT (dialetto SQL del database)
- MON$SHUTDOWN_MODE (tipo di shutdown attuale)
  0: online
  1: multi-user shutdown
  2: single-user shutdown
  3: full shutdown
- MON$SWEEP_INTERVAL (intervallo sweep)
- MON$READ_ONLY (flag di sola lettura)
- MON$FORCED_WRITES (flag forced writes attivo)
- MON$RESERVE_SPACE (flag per spazio riservato)
- MON$CREATION_DATE (data ed ora di creazione)
- MON$PAGES (numero di pagine allocate su disco)
- MON$BACKUP_STATE (stato attuale di backup fisico)
  0: normal
  1: stalled
  2: merge
- MON$STAT_ID (ID della statistica)

*MON$ATTACHMENTS (connessioni correnti)*

- MON$ATTACHMENT_ID (ID della connessione)
- MON$SERVER_PID (server process ID)
- MON$STATE (stato della connessione)
  0: idle
  1: active
- MON$ATTACHMENT_NAME (nome della connessione)
- MON$USER (nome utente)
- MON$ROLE (ruolo della connessione)
- MON$REMOTE_PROTOCOL (nome del protocollo remoto)
- MON$REMOTE_ADDRESS (indirizzo remoto)
- MON$REMOTE_PID (ID del processo client remoto)
- MON$REMOTE_PROCESS (pathname del processo client remoto)
- MON$CHARACTER_SET_ID (character set della connessione)
- MON$TIMESTAMP (data ed ora della connessione)
- MON$GARBAGE_COLLECTION (flag per la garbage collection)
- MON$STAT_ID (ID della statistica)

- Le colonne MON$REMOTE_PID e MON$REMOTE_PROCESS sono valorizzate solo se il client è di versione successiva alla 2.1.

- La colonna MON$REMOTE_PROCESS può contenere un valore che non è un path se un'applicazione ha specificato un nome di processo personalizzato attraverso un DPB.

- La colonna MON$GARBAGE_COLLECTION indica se è permesso il GC da questa connessione, come specificato dal DPB in *isc_attach_database*.

*MON$TRANSACTIONS (transazioni in essere)*

- MON$TRANSACTION_ID (ID della transazione)
- MON$ATTACHMENT_ID (ID della connessione)
- MON$STATE (stato della transazione)
  0: idle (stato dopo la prepare, prima dell'inizio dell'esecuzione e dopo la chiusura del cursore)
  1: active (stato durante l'esecuzione e lettura del cursore)
- MON$TIMESTAMP (data ed ora di start della transazione)
- MON$TOP_TRANSACTION (top transaction: vedi sotto)
- MON$OLDEST_TRANSACTION (numero della Oldest Interesting Transaction locale)
- MON$OLDEST_ACTIVE (numero della Oldest Active Transaction locale)
- MON$ISOLATION_MODE (isolation mode)
  0: consistency
  1: concurrency
  2: read committed record version
  3: read committed no record version
- MON$LOCK_TIMEOUT (timeout per il lock)
  -1: attesa infinita
  0: nessuna attesa
  N: timeout impostato ad N secondi
- MON$READ_ONLY (flag di sola lettura)
- MON$AUTO_COMMIT (flag di auto-commit)
- MON$AUTO_UNDO (flag di auto-undo: vedi sotto)
- MON$STAT_ID (ID della statistica)

- MON$TOP_TRANSACTION è il limite superiore usato dalla transazione di sweep quando avanza l'OIT globale. Tutte le transazioni oltre la soglia dell'OIT sono considerate attive. Normalmente è equivalente al MON$TRANSACTION_ID ma i COMMIT RETAINING oppure i ROLLBACK RETAINING possono bloccare il MON$TOP_TRANSACTION in stallo quando si incrementa l'ID della transazione.

- MON$AUTO_UNDO indica lo stato di auto-undo impostato per la transazione, cioè, se è stato impostato un SAVEPOINT a livello di transazione. L'esistenza di un tale punto di salvataggio a livello di transazione permette di cancellare le modifiche effettuate con ROLLBACK e la transazione viene semplicemente committata. Se il punto di salvataggio non esiste oppure esiste ma il numero di modifiche è esorbitante, allora viene eseguito un ROLLBACK e la transazione viene marchiata nel TIP come «dead».

*MON$STATEMENTS (statement preparati)*

- MON$STATEMENT_ID (ID dello statement)
- MON$ATTACHMENT_ID (ID della connessione)
- MON$TRANSACTION_ID (ID della transazione)
- MON$STATE (stato dello statement)
  0: idle
  1: active
- MON$TIMESTAMP (data ed ora di inizio dello statement)
- MON$SQL_TEXT (testo dello statement, quando appropriato)
- MON$STAT_ID (ID della statistica)

- La colonna MON$SQL_TEXT contiene NULL per gli statement di GDML

- Le colonne MON$TRANSACTION_ID e MON$TIMESTAMP contengono valori validi solo per gli statement attivi

- Il PLAN ed i valori di eventuali parametri non sono al momento disponibili

*MON$CALL_STACK (call stack delle richieste PSQL attive)*

- MON$CALL_ID (ID della chiamata)
- MON$STATEMENT_ID (ID dello statement DSQL di primo livello)
- MON$CALLER_ID (ID del chiamante)
- MON$OBJECT_NAME (nome dell'oggetto PSQL)
- MON$OBJECT_TYPE (tipo dell'oggetto PSQL)
- MON$TIMESTAMP (data ed ora di inizio della richiesta)
- MON$SOURCE_LINE (numero di linea del sorgente SQL)
- MON$SOURCE_COLUMN (numero di colonna del sorgente SQL)
- MON$STAT_ID (ID della statistica)

- La colonna MON$STATEMENT_ID raggruppa gli stack di chiamate per statement DSQL di primo livello che hanno cominciato la catena di chiamate. Tale ID rappresenta una registrazione di statement attivo nella tabella MON$STATEMENTS.

- Le colonne MON$SOURCE_LINE e MON$SOURCE_COLUMN contengono informazioni relative alla linea ed alla colonna del sorgente PSQL attualmente in esecuzione

*MON$IO_STATS (statistiche di I/O)*

- MON$STAT_ID (ID della statistica)
- MON$STAT_GROUP (gruppo della statistica)
  0: database
  1: attachment
  2: transaction
  3: statement
  4: call
- MON$PAGE_READS (numero di pagine lette)
- MON$PAGE_WRITES (numero di pagine scritte)
- MON$PAGE_FETCHES (numero di pagine cercate)
- MON$PAGE_MARKS (numero di pagine con modifiche pendenti, cioè non ancora scritte)

*MON$RECORD_STATS (statistiche a livello di record)*

- MON$STAT_ID (ID della statistica)
- MON$STAT_GROUP (gruppo della statistica)
  0: database
  1: attachment
  2: transaction
  3: statement
  4: call
- MON$RECORD_SEQ_READS (numero di record letti sequenzialmente)
- MON$RECORD_IDX_READS (numero di record letti da indice)
- MON$RECORD_INSERTS (numero di record inseriti)
- MON$RECORD_UPDATES (numero di record aggiornati)
- MON$RECORD_DELETES (numero di record cancellati)
- MON$RECORD_BACKOUTS (numero di record dove una nuova versione
  di record primaria esistente o di una sua modifica è cancellata da un
  rollback o dal richiamo di un punto di salvataggio)
- MON$RECORD_PURGES (numero di record in cui la catena di record version
  sta per essere pulita delle versioni non più necessarie all'OAT
  o da transazioni più recenti)
- MON$RECORD_EXPUNGES (numero di record in cui la catena di record version
  sta per essere eliminata a causa di cancellazioni in transazioni più vecchie
  della OAT)

> **Nota**
>
> Le descrizioni testuali dei valori di tutti gli stati e modi esistenti si trova nella tabella di sistema RDB$TYPES.

## Uso

La creazione di uno snapshot (istantanea) è normalmente un'operazione veloce, ma ci potrebbe essere un certo ritardo in caso di notevole carico di lavoro (specialmente nel Classic Server). Le linee guida per minimizzare l'impatto del monitoraggio in tali condizioni, vedere oltre nelle linee guida sulle prestazioni.

Per ricevere i dati di monitoraggio occorre essere connessi in modo valido ad un database. Le tabelle di monitoraggio danno informazioni solo sul database connesso. Se ci si connette a più database sullo stesso server, ognuno deve essere connesso e monitorato separatamente.

Le variabili di sistema `CURRENT_CONNECTION` e `CURRENT_TRANSACTION` possono essere utilizzate per filtrare i dati relativi rispettivamente alla attuale connessione o transazione del chiamante. Queste variabili corrispondono infatti alle colonne ID delle tabelle di monitoraggio appropriate.

**Esempi**

Per quanto spiegato finora, al fine di ottenere prove significative, è bene sperimentare su un database con più connessioni attive e collegandosi come SYSDBA o come proprietario del database.

1. Ottenere gli ID di tutti i processi di un Classic Server che al momento caricano la CPU

```
SELECT MON$SERVER_PID
  FROM MON$ATTACHMENTS
  WHERE MON$ATTACHMENT_ID <> CURRENT_CONNECTION
  AND MON$STATE = 1
```

2. Ottenere informazioni sulle applicazioni client

```
SELECT MON$USER, MON$REMOTE_ADDRESS,
    MON$REMOTE_PID,
    MON$TIMESTAMP
  FROM MON$ATTACHMENTS
  WHERE MON$ATTACHMENT_ID <> CURRENT_CONNECTION
```

3. Recuperare il livello di isolamento della transazione attuale

```
SELECT MON$ISOLATION_MODE
  FROM MON$TRANSACTIONS
  WHERE MON$TRANSACTION_ID = CURRENT_TRANSACTION
```

4. Recuperare gli statement che sono attivi al momento

```
SELECT ATT.MON$USER,
 ATT.MON$REMOTE_ADDRESS,
 STMT.MON$SQL_TEXT,
 STMT.MON$TIMESTAMP
  FROM MON$ATTACHMENTS ATT
    JOIN MON$STATEMENTS STMT
ON ATT.MON$ATTACHMENT_ID = STMT.MON$ATTACHMENT_ID
  WHERE ATT.MON$ATTACHMENT_ID <> CURRENT_CONNECTION
    AND STMT.MON$STATE = 1
```

5. Listare gli stack delle chiamate per tutte le connessioni

```
WITH RECURSIVE HEAD AS
  (
     SELECT CALL.MON$STATEMENT_ID,
CALL.MON$CALL_ID,
CALL.MON$OBJECT_NAME,
CALL.MON$OBJECT_TYPE
     FROM MON$CALL_STACK CALL
 WHERE CALL.MON$CALLER_ID IS NULL
     UNION ALL
 SELECT CALL.MON$STATEMENT_ID,
  CALL.MON$CALL_ID,
  CALL.MON$OBJECT_NAME,
```

```
    CALL.MON$OBJECT_TYPE
  FROM MON$CALL_STACK CALL
    JOIN HEAD
      ON CALL.MON$CALLER_ID = HEAD.MON$CALL_ID
  )
  SELECT MON$ATTACHMENT_ID,
    MON$OBJECT_NAME,
    MON$OBJECT_TYPE
  FROM HEAD
    JOIN MON$STATEMENTS STMT
ON STMT.MON$STATEMENT_ID = HEAD.MON$STATEMENT_ID
    WHERE STMT.MON$ATTACHMENT_ID <> CURRENT_CONNECTION
```

## Cancellare una query già lanciata

Le query interminabili e lunghe possono essere bloccate e cancellate *da una connessione separata*.

Non c'è nessuna primitiva API dedicata a questa funzione. Sarà compito dell'amministratore del database (SYSDBA o il proprietario) utilizzare le informazioni provenienti dalle tabelle di monitoraggio e determinare un appropriato meccanismo per mettere in riga gli statement fuori controllo.

**Esempio**

Come esempio molto grezzo, col seguente comando si terminano tutti gli statement attualmente in esecuzione sul database tranne quelli che girano sulla connessione *separata* che viene usata dall'amministratore stesso che ha lanciato il comando:

```
delete from mon$statements
  where mon$attachment_id <> current_connection
```

## Guida alle prestazioni nell'uso delle tabelle MON$ sotto stress

Il monitoraggio è basato su due fondamenti: la memoria condivisa e le notifiche.

### Memoria condivisa

Tutti i processi del server condividono alcune regioni di memoria dove sono memorizzate le informazioni delle attività in corso. Tali informazioni sono costituite da elementi multipli a lunghezza variabile che descrivono i dettagli delle varie attività. Tutti gli oggetti appartenenti ad un particolare processo sono raggruppati insieme in un unico blocco per essere processati come un tutt'uno.

La raccolta delle informazioni di monitoraggio ed il popolamento delle tabelle non avviene in tempo reale, in quanto i processi server scrivono i loro dati nella memoria condivisa solo quando gli viene esplicitamente richiesto di farlo. Alla scrittura, i nuovi blocchi sostituiscono quelli vecchi. Quindi, quando viene letta la regione di memoria condivisa, il processo di lettura ispeziona tutti i blocchi rimuovendo la «spazzatura», cioè eliminando i blocchi che appartengono a processi terminati e ricompattando lo spazio della memoria condivisa.

### Notifiche

Ogni processo del server ha un indicatore della sua possibilità di rispondere ad una richiesta di monitoraggio nel momento stesso in cui viene fatta. Quando una connessione utente lancia una interrogazione su una qualunque

tabella di monitoraggio, il processo attivo su quella connessione notifica a tutti gli altri processi una richiesta di informazioni aggiornate. Tali processi rispondono aggiornando i loro blocchi nella regione di memoria condivisa, spegnendo poi l'indicatore di «pronto» alle richieste di monitoraggio.

Una volta che tutti i processi notificati hanno terminato l'aggiornamento dei blocchi, il processo richiedente legge la memoria condivisa, filtra i vari elementi in funzione dei permessi dell'utente, trasforma la rappresentazione interna dei dati in campi di registrazioni e riempie in memoria le tabelle di monitoraggio.

Tutti i processi che sono rimasti inattivi dopo l'ultimo monitoraggio rimangono coll'indicatore di «pronto» spento, indicando che non hanno nulla da aggiornare nella memoria condivisa. L'indicatore *spento* significa che verranno ignorati dalle notifiche e pertanto che sono esentati dall'effettuare il prossimo giro di aggiornamenti.

L'indicatore di «pronto» viene acceso non appena avviene qualcosa di significativo nell'ambito del processo e pertanto ritornerà a rispondere alle richieste di monitoraggio.

### Blocchi

Per coordinare tutte le operazioni di lettura/scrittura, il richiedente pone un blocco esclusivo che influenza sia tutte le connessioni utente attive al momento sia quelle in corso per essere stabilite.

Richieste multiple contemporanee di monitoraggio sono accodate in serie.

### Limitazioni e problemi noti

1. In un server Classic estremamente caricato, le richieste di monitoraggio possono richiedere parecchio tempo per l'esecuzione. Nel frattempo, altre attività (sia nuove interrogazioni che tentativi di connessione) potrebbero essere bloccati fino al termine della richiesta di monitoraggio.

   > **Nota**
   >
   > Migliorato nellav2.1.3.

2. Talvolta, particolari condizioni di scarsa memoria possono far fallire alcune richieste di monitoraggio, oppure provocano lo swap su disco di altri processi attivi. La causa principale di questo problema è il fatto che ogni record di MON$STATEMENTS ha un blob MON$SQL_TEXT che resta per tutto il periodo della transazione di monitoraggio.

   Nella V.2.1.x, ogni blob occupa una intera pagina di memoria, anche se il suo effettivo contenuto è più piccolo Quando c'è un elevato numero di frasi preparate nel sistema, diventa possibile esaurire tutta la memoria ed ottenere dei fallimenti.

3. Un'altra possibile causa di esaurimento della memoria potrebbe essere la temporanea (per un tempo molto breve, in realtà) crescita della memoria in cui i dati sono messi mentre si provvede ad unificarli in un solo blocco.

   > **Nota**
   >
   > Al momento non c'è soluzione per la v.2.1.3. Tuttavia è stato migliorato per la v.2.5.0.

## Altre informazioni sul contesto

Ci sono altre informazioni disponibili sul contesto relativamente al server, attraverso statement di SELECT della funzione RDB$GET_CONTEXT col parametro 'SYSTEM', inclusa la versione del motore.

**Esempio**

```
SELECT RDB$GET_CONTEXT('SYSTEM', 'ENGINE_VERSION')
  FROM RDB$DATABASE
```

Per informazioni dettagliate sull'uso di queste chiamate, far riferimento alle più recenti versioni delle note di rilascio di Firebird v.2.0.

Capitolo 12

# Sicurezza

In questo capitolo ci sono i dettagli delle modifiche alla sicurezza in Firebird effettuate col rilascio di Firebird 2 e delle versioni successive. Sono inoltre evidenziati le ulteriori modifiche ed i miglioramenti introdotti nella V.2.1.

## Introduzione alle modifiche

Il miglioramento dei sistemi di sicurezza ha avuto un notevole impulso in Firebird 2.0. Quanto segue è un riepilogo delle principali modifiche.

### Nuovo database della sicurezza

Il nuovo database per la sicurezza è stato rinominato `security2.fdb`. Internamente, la tabella di autenticazione degli utenti, dove sono memorizzati i nomi e le password degli utenti, è stata rinominata `RDB$USERS`. Non esiste più la tabella «users» ma una nuova *vista* su RDB$USERS che è nominata «USERS». Attraverso tale vista gli utenti possono cambiarsi le loro password.

Per dettagli sul nuovo database, vedere Nuovo database della sicurezza nella sezione sull'autenticazione più oltre in questo capitolo.

Per istruzioni sul come aggiornare dalla precedente versione del database della sicurezza, fare riferimento alla fine di questo capitolo alla sezione Gestione del nuovo database di sicurezza.

### Usare il sistema di autenticazione degli utenti di Windows

**(V.2.1)** A partire da Firebird 2.1, si può utilizzare il sistema di autenticazione degli utenti di Windows («Trusted User» security) per verificare le credenziali degli utenti Firebird su una piattaforma Windows. Il contesto degli utenti convalidati viene passato al server Firebird invece del nome utente e password e, se positivo, viene usato per determinare il nome utente Firebird.

Per dettagli vedere più avanti la sezione, Sicurezza con Windows Trusted Users.

### Migliorata la crittografia delle password

A. Peshkov

Il sistema di criptazione/decriptazione delle password adesso usa un migliore algoritmo di calcolo.

### Gli utenti possono modificarsi la password

A. Peshkov

Il SYSDBA rimane il proprietario del database della sicurezza. Tuttavia gli utenti possono modificarsi la loro propria password.

### Solo il server può connettersi al database di sicurezza

A. Peshkov

Anche il programma di utilità *gsec* ora usa le API di servizio. Il server rifiuta qualsiasi tipo di accesso al security2.fdb se non attraverso le API del gestore dei servizi (Services Manager).

### Protezione attiva contro gli attacchi brutali

A. Peshkov

I tentativi di accedere al server usando tecniche brutali sugli nominativi e le password sono rilevati e bloccati.

- Da ogni client remoto è obbligatorio connettersi con la password
- I client che cercano di effettuare troppi tentativi di connessioni sono bloccati per un certo periodo di tempo

Il supporto di protezione contro gli attacchi brutali è stato incluso nelle funzioni di connessione sia delle API di Firebird sia delle API di servizio. Per altri dettagli vedere Protezione dagli attacchi brutali

### Vulnerabilità varie chiuse

A. Peshkov, C. Valderrama

Alcune note vulnerabilità nell'API sono state chiuse.

> **Attenzione**
>
> Va notato che rimettere la **funzionalità del server redirection ("multi-hop")** in Firebird 2 può potenzialmente reintrodurre una nuova vulnerabilità. Per tale ragione ciò è controllato da un parametro in `firebird.conf` (Redirection), che non va abilitato a meno che non sappia esattamente tutte le reali implicazioni di ciò che si sta' facendo.
>
> In questi giorni, la possibilità di re-direzionare ad altri server le richieste è considerata pericolosa. Supponendo di avere un server firebird accuratamente protetto, a cui sia possibile accedere da Internet. In una situazione in cui esso ha accesso non ristretto alla LAN interna, può lavorare come gateway per tutte le richieste entranti come `firebird.your.domain.com:internal_server:/private/database.fdb`.
>
> Sapere il nome o l'indirizzo IP di un qualsiasi server interno nella LAN è sufficiente per un intruso: non ha bisogno di login su un server esterno. Tale gateway facilmente soprassiede qualsiasi firewall che sta' proteggendo la LAN dagli attacchi esterni.

## Dettagli sulle modifiche di sicurezza in Firebird 2

Sono state messe a fuoco alcune debolezze riconosciute nella sicurezza di Firebird contro gli attacchi malevoli:

- la mancanza di un sistema di criptazione delle password resistente agli attacchi brutali nel database della sicurezza

- la possibilità di ogni utente remoto di aprire il database della sicurezza e leggervi le password criptate (specialmente se combinato col punto precedente)
- la impossibilità degli utenti a cambiare le loro password
- la mancanza di protezione contro gli attacchi brutali alle password nel server stesso

## Autenticazione

L'autenticazione in Firebird controlla su un database di sicurezza a livello di server per decidere se autorizzare una connessione ad un database o al server. Il database della sicurezza memorizza i nomi utente e le password di tutte le identità autorizzate.

### Autenticazione in Firebird 1.5

In Firebird 1.5 si usa l'algoritmo DES due volte per criptare la password: prima dal client e poi dal server, prima di confrontarle col dato memorizzato nel database della sicurezza. Tuttavia, questa sequenza viene completamente rotta quando SYSDBA cambia una password. Il client effettua due volte il calcolo e memorizza il risultato nel database della sicurezza. Pertanto la gestione della password è completamente dipendente dal client (o, meglio, definita dal client).

### Firebird 2: calcolo lato server

Per utilizzare sistemi di criptazione più robusti, è stato necessario un altro approccio. La password criptata da memorizzare sul server dovrebbe essere sempre calcolata dal server stesso. Tale schema esiste già in Firebird all'interno delle API di servizio. Questo portò alla decisione di utilizzarle per ogni attività client relativa alla gestione degli utenti. Adesso, *gsec* e le funzioni API isc_user_* (con * uguale add, modify o delete) usano tutte le API di servizio per accedere al database della sicurezza. (L'eccezione è l'accesso embedded al server Classic su sistemi POSIX, vedere sotto).

Effettuare modifiche alla gestione delle password è diventato abbastanza semplice, è sempre effettuata dal server. Non è più un compito della *gsec* calcolare la password criptata per il database della sicurezza: chiede semplicemente al server di fare il lavoro!

Vale la pena notare che la nuova *gsec* funziona anche con Firebird più vecchi, purché le architetture del server supportino i servizi.

### L'algoritmo di criptazione SHA-1

Questo metodo porta ad una situazione in cui

1. una parola criptata per l'utente A non è valida per l'utente B (ovvero a partire da password uguali vengono generate codici criptati diversi)
2. quando un utente cambia la sua password, anche se la stringa è identica alla precedente, il dato memorizzato in RDB$USERS.RDB$PASSWD è diverso.

Sebbene questa situazione da sola non aumenti la resistenza ad un tentativo di attacco brutale di ottenere la password, rende l'analisi "visuale" di una password rubata molto più difficile.

### Il nuovo database della sicurezza

La struttura del database della sicurezza è stata cambiata. In generale, adesso contiene una patch di Ivan Prenosil, con qualche piccola modifica, che permette ad ogni utente di modificare la propria password.

- In Firebird 1.5 la tabella USERS poteva essere letta da PUBLIC, una necessità del motore senza la quale il sistema di validazione delle password non avrebbe funzionato. La soluzione di Ivan usa una vista, con la condizione "WHERE USER = "". Questo funzionava a causa di un altro problema nel motore che in SQL lasciava la variabile USER vuota, non 'authenticator', come sembra dal codice del motore.

  Una volta corretto tale problema, diventa possibile aggiungere la condizione "USER = 'authenticator'". A breve termine questo andava bene in quanto il nome utente viene sempre convertito in maiuscolo.

- Successivamente è stata trovata una soluzione migliore che evita il ricorso ad un trucchetto SQL per fare l'autenticazione dell'utente. Il risultato è che un utente non-SYSDBA può vedere solo il suo proprio nome di login in tutti gli strumenti di amministrazione (*gsec*, o tutte le interfacce grafiche che usano le API di servizio). SYSDBA continua ad avere accesso completo alle credenziali utente.

#### Struttura del nuovo database della sicurezza

Il database della sicurezza di Firebird 2 è `security2.fdb`. Per l'autenticazione degli utenti ha la nuova tabella `RDB$USERS` che memorizza le password criptate col nuovo metodo. Una vista su questa tabella sostituisce la vecchia tabella USERS e permette agli utenti di modificare le proprie password.

Il DDL delle nuove strutture è descritto nell'Appendice Appendice C.

## gsec in Firebird 2

Misure speciali sono state adottate per rendere completamente impossibili le connessioni remote al database della sicurezza. Non ci si deve sorprendere se qualche vecchio programma non riesce ad accedere direttamente: questo è voluto. Adesso si può accedere alle informazioni degli utenti solo attraverso le API di servizio e l'accesso interno ai servizi equivalente implementato ora nelle funzioni API `isc_user_*`.

## Protezione dagli attacchi brutali

Le attuali velocissime CPU e le connessioni WAN veloci rendono possibile tentare di attaccare brutalmente le password degli utenti di un server Firebird. Questo è pericolo specialmente per il Superserver che effettua, fin dalla versione 1.5, un'autenticazione dell'utente molto veloce. Il server Classic è più lento, poiché deve creare un nuovo processo per ogni connessione, collegarsi al database della sicurezza attraverso quella connessione e preparare una richiesta alla tabella RDB$USERS prima di poter validare login e password. Il Superserver bufferizza le connessioni e le richieste, permettendo una validazione dell'utente molto più veloce.

Data la lunghezza massima di 8 byte delle tradizionali password di Firebird, l'attacco brutale ha una ragionevole possibilità di successo per irrompere nell'installazione Firebird.

La versione 2.0 del Superserver ha una protezione attiva per rendere più difficili gli attacchi. Dopo alcuni tentativi di accesso falliti, l'utente e l'indirizzo IP vengono bloccati per alcuni secondi, impedendo per un breve periodo ogni tentativo di accesso successivo con quel nome utente OPPURE da quel particolare indirizzo IP.

Non sono richieste impostazioni o configurazioni per questa caratteristica. Viene automaticamente attivata appena parte il SuperServer Firebird 2.0.

# Usare Windows Security per autenticare gli utenti

Alex Peshkov

**(V.2.1)** A partire da Firebird 2.1 in poi, si può utilizzare la sicurezza di Windows «Trusted User» per autenticare gli utenti su una piattaforma Windows. Il contesto di sicurezza della Trusted User viene passato al server Firebird, e se ha successo, viene usato per determinare la sicurezza in Firebird a partire dal nome utente.

Omettendo semplicemente i parametri utente e password dal DPB/SPB si applica automaticamente l'autenticazione attraverso Windows Trusted User, in quasi tutti i casi. Vedere la sezione sull'Environment, sotto, per le eccezioni.

**Esempio dimostrativo**

Supponiamo di entrare nel server Windows SRV con l'utente 'Mario', e di connettersi al server Firebird SRV con *isql*, senza specificare un utente ed una password Firebird, in questo modo:

```
isql srv:employee
```

A questo punto si fa:

```
SQL> select CURRENT_USER from rdb$database;
```

si ottiene una cosa come:

```
USER
==================================================
SRV\Mario
```

## Privilegi SQL

Gli utenti Windows possono ricevere garanzie di accesso agli oggetti e ai ruoli del database allo stesso modo degli utenti Firebird standard, similmente a quanto è sempre stato disponibile per gli utenti dei database Firebird ospitati su Unix e Linux.

## Amministratori

Se un amministratore locale o un membro del gruppo predefinito di Amministratori di Dominio si connette a Firebird utilizzando l'autenticazione trusted, viene connesso come SYSDBA.

## Configurazione del parametro «Authentication»

Il nuovo parametro `Authentication` è stato aggiunto in firebird.conf per configurare il metodo si autenticazione in Windows. I possibili valori sono:

*Authentication = Native*
Permette la piena compatibilità con le versioni di Firebird precedenti, impedendo l'autenticazione trusted.

*Authentication = Trusted*
Il database della sicurezza viene ignorato e viene usata la sola autenticazione di Windows. Sotto certi aspetti, l'autenticazione Windows è più sicura di quella nativa, nel senso che non è né più né meno sicura del sistema operativo in cui gira.

*Authentication = Mixed*
Questa è l'impostazione di default.

Per mantenere il comportamento compatibile, se sono impostate nell'ambiente le variabili `ISC_USER` e `ISC_PASSWORD`, queste vengono prese e usate al posto dell'autenticazione trusted.

> **Nota**
>
> L'autenticazione trusted può essere forzata a sovrastare le variabili d'ambiente quando sono state impostate — vedere le note qui sotto.

## Forzare l'autenticazione trusted

Nel caso in cui sia necessaria l'autenticazione trusted e c'è la possibilità di trovare impostate `ISC_USER` e `ISC_PASSWORD`, c'è un nuovo parametro in DPB che può essere aggiunto al DPB, `isc_dpb_trusted_auth`.

Molte delle utilità a linea di comando di Firebird supportano parametri attraverso lo switch `-tru[sted]` (la forma abbreviata è disponibile, secondo le regole usuali per abbreviare gli switch).

> **Nota**
>
> Le utilità *qli* e *nbackup* non seguono questa linea: esse usano switch a singola lettera che è uno stile un po' vecchiotto. Lo switch per *qli* è nel caso -K. Per *nbackup*, invece è spiegato qui sotto. La possibilità di forzargli l'autenticazione trusted deve essere ancora implementata.

**Esempio**

```
C:\Pr~\bin>isql srv:db            -- entra usando l'autenticazione trusted
C:\Pr~\bin>set ISC_USER=utente1
C:\Pr~\bin>set ISC_PASSWORD=12345
C:\Pr~\bin>isql srv:db            -- entra usando 'utente1' specificato nell'ambiente
C:\Pr~\bin>isql -trust srv:db     -- entra usando l'autenticazione trusted
```

> **Importante**
>
> Le regole in Windows per i nomi utente nei domini permettono nomi più lunghi del massimo di 31 caratteri permesso da Firebird per i nomi utente. **Il limite di 31 caratteri è assicurato** e, a partire dalla V.2.1, viene disabilitata la possibilità di entrare con nomi più lunghi. Questa rimane la situazione finché non verrà implementata in una futura versione di Firebird la mappatura degli oggetti del sistema operativo con gli oggetti del database.

## Il Server Classic in POSIX

Per ragioni sia tecniche che storiche, il server Classic in POSIX con client embedded è particolarmente vulnerabile riguardo alla sicurezza. Gli utenti che accedono ai database attraverso l'embedded DEVONO almeno avere i diritti di accesso in lettura al database della sicurezza.

Questa è la principale ragione che ha determinato la necessità di implementare una migliore crittazione delle password. Un utente malevolo con accesso utente a Firebird avrebbe potuto facilmente fare una copia del database della sicurezza, portarsela a casa e con calma forzare brutalmente la vecchia crittazione DES! Dopodiché avrebbe potuto modificare i dati sui database importanti memorizzati sul server. Firebird 2 è molto meno vulnerabile a questo tipo di attacchi.

Comunque il server embedded in POSIX ha un altro problema di sicurezza: la sua implementazione delle API di servizio chiama l'utilità a linea di comando *gsec*, come gli utenti normali. Pertanto un'utilità di manutenzione degli utenti su embedded server deve avere accesso pieno al database della sicurezza.

La principale ragione per restringere l'accesso diretto al database della sicurezza era di proteggerlo dalle vecchie versioni del software client. Contemporaneamente, per caso fortuito, minimizza anche l'esposizione del Classic embedded in POSIX, poiché è improbabile la presenza contemporanea in un ambiente operativo di un vecchio client e di un nuovo server.

## In tutte le piattaforme

> **Attenzione**
>
> Il livello di sicurezza non è ancora soddisfacente sotto un aspetto molto serio, pertanto leggete attentamente questa sezione prima di aprire la porta 3050 ad internet.

Un problema di sicurezza importante rimane irrisolto in Firebird: la trasmissione «in chiaro» attraverso la rete di password malamente criptate. Non è possibile risolvere questo problema senza bloccare i client più vecchi.

Mettendola in altro modo, un utente che ha impostato la password usando un nuovo metodo sicuro, sarebbe incapace di usare un client vecchio per collegarsi al server. Avendo questo presente nei piani di aggiornare alcuni aspetti delleAPI nelle prossime versioni, è stata presa la decisione di non modificare il metodo di trasmissione delle password in Firebird 2.0.

Nell'immediato il problema può essere risolto facilmente usando un software di IP-tunneling (come ZeBeDee) per trasferire i dati da e per un server Firebird, per entrambe le versioni 1.5 e 2.0. Al momento rimane il modo raccomandato di accedere ad un server Firebird attraverso Internet.

## Altri miglioramenti

### *isc_service_query() erroneamente rivelava il percorso completo di un db*

Richiesta CORE-1091

**(V.2.1)** Configurando il server con "DatabaseAccess = None", `isc_service_query()` riportava il percorso completo di nome file del database. Adesso è stato corretto per riportare il solo alias, un argomento in più a favore di rendere standard l'uso degli alias!

### *Chiunque poteva leggere il log del server con le API di servizio*

Richiesta CORE-1148

Questa era una vulnerabilità della sicurezza minore. Gli utenti normali adesso sono bloccati se cercano di accedere al log del server attraverso le API di servizio. Le richieste sono verificate esplicitamente al fine di assicurare che l'utente autenticato sia SYSDBA.

## Gestione del nuovo database della sicurezza

A. Peshkov

Se si cerca di mettere un database di sicurezza di una versione precedente alla 2, sia esso un `security.fdb` oppure un `isc4.gdb` rinominati, nella directory di installazione di Firebird, cercando di connettervisi si riceve il messaggio d'errore "Cannot attach to password database" («Impossibile collegarsi al database delle password»). Le cosa stanno così per scelta progettuale. Un database della sicurezza di una versione precedente di Firebird non può essere utilizzato direttamente in Firebird 2.0 o successivi.

Il database della sicurezza rinnovato nella struttura si chiama security2.fdb.

Per riutilizzare un vecchio database della sicurezza, è necessario lanciare lo script di aggiornamento `security_database.sql`, che è nella sotto-cartella `../upgrade` dell'installazione del server Firebird.

> **Nota**
>
> Una copia dello script è anche in Appendice C.

### *Effettuare l'aggiornamento del DB della sicurezza*

Per effettuare tale aggiornamento, seguire questa procedura:

1. mettere il vecchio database della sicurezza in un posto sicuro, ma non nella cartella della nuova installazione di Firebird. Farne una copia per tenerla comunque a disposizione!
2. Far partire Firebird 2, usando il suo originale nuovo security2.fdb.
3. Convertire il vecchio database della sicurezza a ODS11 (cioè farne il backup ed il restore usando Firebird 2.0). Senza questo passo, lo script security_database.sql non funzionerà!
4. Connettersi a quest'ultimo dopo il restore come SYSDBA e lanciare lo script.
5. Fermare il servizio Firebird.
6. Copiare il database aggiornato nella cartella di installazione di Firebird 2 come `security2.fdb`.
7. Far ripartire Firebird.

Ora dovrebbe essere possibile collegarsi al server Firebird 2 utilizzando i vecchi login e password.

### NULL nel campo RDB$PASSWD

Nelle versioni di Firebird precedenti alla 2.0 era possibile avere utenti con password a NULL. A partire dalla versione 2.0 in poi nel database della sicurezza il campo RDB$PASSWD è vincolato ad essere NOT NULL.

Tuttavia per evitare eccezioni durante lo script di aggiornamento, il campo è creato dallo script come annullabile. Se si è sicuri di non avere password vuote nel database della sicurezza, potete modificare lo script a mano prima di lanciarlo. Ad esempio si può modificare la linea

```
RDB$PASSWD RDB$PASSWD,
```

in:

```
RDB$PASSWD RDB$PASSWD NOT NULL,
```

### Attenzione sul parametro LegacyHash

Se si configura `LegacyHash = 1` in `firebird.conf`, la sicurezza di Firebird non funziona più per bene. Per raddrizzare le cose, è necessario fare quanto segue:

1. Cambiare la password di SYSDBA
2. Fare in modo che gli utenti cambino la loro password (in 2.0 ogni utente può cambiare la sua password).
3. Rimettere LegacyHash a 0, cioè al valore di default, oppure commentarla.
4. Fermare e far ripartire Firebird per attivare la modifica alla configurazione.

Capitolo 13

# Utilità a linea di comando

## Miglioramenti

### Supporto ai TRIGGER di Database

**(V. 2.1)** Un nuovo parametro è stato aggiunto a *gbak*, *nbackup* e *isql* per non eseguire Trigger a livello di database. Ciò è disponibile solo al proprietario del database e a SYSDBA:

```
gbak -nodbtriggers
isql -nodbtriggers
nbackup -T
```

### Occultamento password

Alex Peshkov

Le utilità a linea di comando che accettano il parametro **-password** sono vulnerabili alla cattura della password, specialmente quando sono lanciate da uno script. Un ulteriore passo nell'ostacolare tutto ciò nelle piattaforme POSIX, consiste nel mostrare nella lista dei processi l'argomento [PASSWORD] con un asterisco ( * ), mentre prima veniva visualizzato in chiaro.

## Servizi di Firebird

### Nuova utilità a linea di comando fbsvcmgr

Alex Peshkov

**(V.2.1)** La nuova utilità *fbsvcmgr* permette un'interfaccia a linea di comando verso le API di servizio, permettendo l'accesso a tutti i servizi implementati in Firebird.

Sebbene ci siano in giro molti strumenti con interfaccia grafica per l'amministrazione dei database attraverso le API di servizio,questo nuovo strumento risolve agli amministratori il problema di accedere a server Unix remoti nelle grandi reti attraverso una connessione a linea di comando. Precedentemente per ottenere una cosa del genere era necessario ricorrere ad un programmatore.

## Uso di fbsvcmgr

*fbsvcmgr* non simula i parametri implementati nelle tradizionali «g*» utilità. Invece, semplicemente è un'interfaccia attraverso cui possono transitare le funzioni ed i parametri delle API di servizio. Gli utenti pertanto devono essere pratici della versione attuale delle API di servizio. Il file di definizioni delle API `ibase.h`, nella cartella di installazione di Firebird `../include`, va preso come primaria fonte di informazioni su quanto è disponibile, supportato dalla lettura (In inglese) del manuale «InterBase 6.0 beta API Guide».

### Parametri

*Specificare il gestore del servizio*
: Il primo parametro richiesto nella linea di comando è il «gestore dei servizi» (Services Manager) a cui ci si vuole collegare:

- Per una connessione locale usare il semplice simbolo `service_mgr`
- Per collegarsi ad un server remoto, bisogna usare il formato con il nome del server `nomeserver:service_mgr`

*Specificare i successivi blocchi dei parametri di servizio (SPB)*
: Successivamente vanno specificati gli SPB (Service Parameter Block), con eventuali valori richiesti. Ogni SPB può essere opzionalmente prefisso da un singolo simbolo '-'. Per le lunghe linee di comando che usualmente servono con *fbsvcmgr*, l'uso di uno '-' migliora la leggibilità della linea di comando. Confrontare, ad esempio, le seguenti (sono linee di comando su singola linea, ignorare gli a capo messi per necessità di stampa):

```
# fbsvcmgr service_mgr user sysdba password masterke
        action_db_stats dbname employee sts_hdr_pages
```

e

```
# fbsvcmgr service_mgr -user sysdba -password masterke
        -action_db_stats -dbname employee -sts_hdr_pages
```

### Sintassi degli SPB

La sintassi degli SPB che *fbsvcmgr* è in grado di capire coincide con quanto si trova nel file `ibase.h` o nella documentazione delle API di InterBase 6.0, eccetto che una certa forma d'abbreviazione può essere usata per ridurre un po' la lunghezza delle linee di comando. Ecco come funziona.

Tutti i parametri SPB hanno una di queste due forme: (1) `isc_spb_VALORE` oppure (2) `isc_VALORE1_svc_VALORE2`. In *fbsvcmgr* c'è bisogno solamente delle parti `VALORE, VALORE1 oppure VALORE2` quando si deve scrivere il parametro.

Di conseguenza, nel caso (1) si scrive semplicemente `VALORE`, mentre nel caso (2) si deve scrivere VALORE1_VALORE2. Ad esempio:

```
isc_spb_dbname => dbname
isc_action_svc_backup => action_backup
isc_spb_sec_username => sec_username
isc_info_svc_get_env_lock => info_get_env_lock
```

e così via.

> **Nota**
>
> L'eccezione è `isc_spb_user_name`: può essere specificata sia come `user_name` o più semplicemente con `user`.

Non è realistico descrivere tutti i parametri di SPB in queste note di rilascio. Nella documentazione di InterBase 6.0 beta il tutto prende circa 40 pagine! La successiva sezione evidenzia solo alcune note differenze tra il modo di operare di *fbsvcmgr* e quello che si potrebbe arguire dalla vecchia documentazione di Interbase.

### *fbsvcmgr: sintassi specifica*

*«Cosa fare e cosa non fare»*

Con *fbsvcmgr* si effettua una singola azione e quando ne è il caso se ne ottiene il suo risultato, oppure si possono recuperare più elementi informativi dal gestore dei servizi. Non si possono fare entrambe le cose in un unico comando.

Ad esempio,

```
# fbsvcmgr service_mgr -user sysdba -password masterke
    -action_display_user
```

elenca tutti gli utenti sul server firebird locale:

```
SYSDBA                          Sql Server Administrator      0    0
QA_USER1                                                      0    0
QA_USER2                                                      0    0
QA_USER3                                                      0    0
QA_USER4                                                      0    0
QA_USER5                                                      0    0
GUEST                                                         0    0
SHUT1                                                         0    0
SHUT2                                                         0    0
QATEST                                                        0    0
```

...e...

```
# fbsvcmgr service_mgr -user sysdba -password masterke
    -info_server_version -info_implementation
```

elenca sia la versione che l'implementazione del server:

```
Server version: LI-T2.1.0.15740 Firebird 2.1 Alpha 1
Server implementation: Firebird/linux AMD64
```

Ma un tentativo di ottenere tutto ciò in una unica linea di comando:

```
# fbsvcmgr service_mgr -user sysdba -password masterke
    -action_display_user -info_server_version -info_implementation
```

da' un errore:

```
Unknown switch «-info_server_version»
```

*Elementi non documentati*
La funzione `isc_spb_rpr_list_limbo_trans` è omessa dalla documentazione beta di IB6, comunque *fbsvcmgr* la supporta.

*Supporto per i nuovi elementi nelle API v.2.1*
Due nuovi elementi sono stati aggiunti nelle API di servizio in Firebird 2.1 e sono supportati da *fbsvcmgr*:

- `isc_spb_trusted_auth` (da scrivere `trusted_auth`) si applica solo a Windows. Forza Firebird ad usare l'autenticazione trusted di Windows.
- `isc_spb_dbname` (da scrivere `dbname`) permette di impostare nei parametri il nome di un database in tutte le azioni di servizio per accedere al database della sicurezza da un client remoto, in modo equivalente allo specificare in *gsec* lo switch `-database`.

> **Nota**
>
> Lo switch `-database` in *gsec* è usato soprattutto per specificare un server remoto da amministrare. In *fbsvcmgr*, il nome del server è già specificato col primo parametro (attraverso il simbolo `service_mgr`) cosicché il parametro `[isc_spb_]dbname` non è particolarmente necessario.

*Errori nella documentazione*
Il formato descritto per alcuni parametri nella documentazione di InterBase 6 beta contiene diversi errori. Nel caso ci fossero problemi, la fonte adatta per le informazioni sulla forma corretta è `ibase.h` .

*Funzioni non supportate*
- Tutto ciò che si riferisce alla licenza d'uso è stato rimosso dal codice sorgente aperto originale di InterBase 6 e pertanto non è supportato né in Firebird né da *fbsvcmgr*.
- Le funzioni per vedere o modificare il vecchio file di configurazione sono state eliminate sin da Firebird 1.5 e quindi non implementate in *fbsvcmgr*.

## Risolti alcuni roblemi del servizio di backup

A. Peshkov

Richiesta CORE-1232

**(V.2.1)** Mentre il gestore dei servizi effettuava operazioni di backup o restore, potevano accadere cose strane se alcuni parametri mancavano oppure se erano nell'ordine sbagliato. Il problema c'è ancora nelle versioni più vecchie, inclusa la v.2.0.x, pertanto bisogna fare attenzione nello specificare tutti gli swirtch richiesi fornendo il nome del database e del file di backup nell'ordine corretto con lo switch **-se[vice_mgr]**.

### *L'accesso ai servizi privilegiati consentito solo a SYSDBA*

A. Peshkov

Richiesta CORE-787

L'accesso alle informazioni riportate dalle API di servizio riguardanti gli utenti ed i percorsi alle cartelle dei database è stato inibito agli utenti non amministratori. Tuttavia, un utente non privilegiato può ottenere informazioni su sé stesso.

## Strumenti di Backup

Ci sono molte novità in Firebird 2 per fare copie di sicurezza dei database: un nuovo programma di utilità per ottenere copie incrementali in linea e altri miglioramenti alla *gbak* per evitare alcuni trappoloni in cui avrebbero potuto incorrere gli utenti finali.

### *Il nuovo backup incrementale*

N. Samofatov

Un sistema per effettuare copie di sicurezza veloci, on-line, a livello di pagina ed incrementale è stato implementato ex-novo.

Il sistema di backup è composto di due parti:

- NBak, il modulo di supporto al motore
- NBackup, lo strumento che realmente effettua il backup

#### *Nbak*

Le incombenze funzionali di NBAK sono:

1. redirezionare le scritture sui file di differenza quando richiesto (frase `ALTER DATABASE BEGIN BACKUP`)
2. produrre un GUID che fotografa il database e scriverlo nella testata dello stesso prima che ritorni dall'esecuzione la frase `ALTER DATABASE BEGIN BACKUP`
3. integrare le differenze nel database quando richiesto (frase `ALTER DATABASE END BACKUP`)
4. segnare le pagine scritte dal motore con il valore corrente del contatore SCN [scansione di pagina] per il database
5. incrementare l'SCN ad ogni cambio dello stato di backup

Il ciclo degli stati di backup è:

**nbak_state_normal -> nbak_state_stalled -> nbak_state_merge -> nbak_state_normal**

- Nello stato *normal* (normale) le scritture vanno direttamente ai file principali del database.
- Nello stato *stalled* (in stallo) le scritture vanno solo ai file delle differenze ed i file principali diventano in sola lettura.
- Nello stato di *merge* (integrazione) le nuove pagine non vengono allocate per i file delle differenze. Si scrive direttamente sui file principali del database. Le letture delle pagine mappate confrontano entrambe le versioni delle pagine e riportano la versione più recente, perché è impossibile sapere se è già stata integrata oppure no.

> **Nota**
>
> Questa logica dello stato di merge ha una parte un po' particolare. Sia Microsoft che Linux definiscono il contenuto del file che cresce come «indefinito» cioè sporco, ed entrambi lo inizializzano a zero.
>
> Questo è il motivo per cui non vengono lette le pagine mappate oltre la fine originale del file di database principale e vengono invece tenute aggiornate nel file delle differenze fino alla fine dell'integrazione. Qui c'è almeno la metà di tutta la logica di lettura e scrittura di NBak, provata usando PIO modificato su file esistenti che contenevano dati sporchi.

### NBackup

Le responsabilità funzionali di NBackup sono

1. rendere possibile un modo conveniente di lanciare ALTER DATABASE BEGIN/END BACKUP
2. sistemare il database dopo la copia del file (cambiando fisicamente lo stato da `nbak_state_diff` a `nbak_state_normal` nella testata del database)
3. creare e recuperare backup (copie di sicurezza) incrementali.

   Le copie di sicurezza incrementali sono multi livello. Ciò significa che, ad esempio, facendo un backup di livello 2 ogni giorno ed un backup di livello 3 ogni ora, ogni backup di livello 3 contiene tutte le pagine modificate dall'inizio del giorno (momento in cui è stato fatto l'ultimo backup di livello 2) fino al momento in cui viene fatto il backup di livello 3.

### Fare il backup

Le copie incrementali seguono il seguente algoritmo:

1. lanciano `ALTER DATABASE BEGIN BACKUP` per redirezionare le scritture al file delle differenze
2. Controllano SCN e GUID del backup più recente del livello precedente
3. Accodano le pagine di database che hanno un SCN maggiore di quello trovato al passo 2 nel file di backup
4. Scrivono il GUID della copia di livello precedente nell'header, rendendo possibile la verifica della consistenza della catena di copie durante la fase di recupero (restore).
5. lanciano `ALTER DATABASE END BACKUP`
6. aggiungono in `RDB$BACKUP_HISTORY` la registrazione della operazione appena eseguita, registrando il livello corrente, SCN, la GUID immagine del database e altre informazioni ad uso e consumo dell'utente.

### Recupero

Il recupero o restore dei dati è semplice: si ricostruisce l'immagine fisica del database attraverso la catena dei file di copia, verificando che il backup_guid di ogni file corrisponda al prev_guid del successivo, e poi si aggiorna il tutto cambiando lo stato in header di nuovo in nbak_state_normal.

### Uso

```
nbackup <opzioni>
```

### Opzioni valide

```
-L <database>   Blocca il database per fare la copia
-N <database>   Sblocca un database precedentemente bloccato
-F <database>   Sistema il database dopo la copia
-B <livello> <database> [<nomefile>]  Crea un backup incrementale
-R <database> [<file0> [<file1>...]]   Restore di un backup incrementale
-U <utente>     Nome dell'utente
-P <password>   Password dell'utente
```

> **Nota**
>
> 1. <database> può anche specificare un alias del database
> 2. non è supportato il backup incrementale dei database multi file
> 3. "stdout" può essere usato come valore per <nomefile> nella opzione -B

## Miglioramenti in V.2.1.3

A. Peshkov

Nella versione di Firebird 2.5 beta, nelle versioni POSIX si è risolto un problema in cui lo strumento di copia completa di *nBackup* si impadroniva di tutte le risorse di I/O nel fare la copia di grandi database, bloccando completamente gli altri lavori sul server. Questo miglioramento è stato riportato anche in V.2.1.3. Ora, nBackup cerca di leggere prima dalla cache del sistema prima di cercare di leggere da disco, riducendo sostanzialmente il quantitativo di I/O necessario.

> **Nota**
>
> Il costo potrebbe essere un aumento variabile dal 10 al 15 per cento in più del tempo impiegato per effettuare una copia completa in condizioni di grande carico di lavoro.

Riferimenti CORE-2316.

### Manuale utente

P. Vinkenoog

Un manuale utente (ancora solo in inglese) per NBak/NBackup è stato già preparato. Può essere scaricato dal sito Firebird nella zona relativa alla documentazione: www.firebirdsql.org/pdfmanual/. Il nome del file è `Firebird-nbackup.pdf`.

## gbak, programma di utilità per copia, trasporto e recupero

Un certo numero di miglioramenti sono stati apportati al programma di utilità *gbak*.

### Comportamento e nuove funzioni

V. Khorsun

Il nuovo switch separato per la gbak

```
-RECREATE_DATABASE [OVERWRITE]
```

è stato aggiunto per impedire a chi non se lo aspetta di sovrascrivere accidentalmente il proprio database, come poteva facilmente accedere con la forma abbreviata del vecchio switch:

```
-R[EPLACE_DATABASE]
```

**In sintesi:**

- gbak -R (oppure gbak -r) adesso applica il nuovo switch -R[ECREATE_DATABASE] e non sovrascrive più un database esistente se manca l'argomento O[VERWRITE].
- La precedente vecchia forma abbreviata -R[EPLACE_DATABASE] è stata cambiata in -REP[LACE_DATABASE]. Questo switch non accetta l'argomento O[VERWRITE].
- Lo switch -REP[LACE_DATABASE] deve essere ritenuto «sconsigliato», cioè potrebbe diventare non più disponibile in una prossima futura release di Firebird.

Questa modifica comporta che ogni procedura script o periodica che utilizza «gbak -r» o «gbak -R» senza modifiche darà un'eccezione se il database esiste.

Per continuare a far funzionare gli script permettendo loro di sovrascrivere incondizionatamente i database, bisogna modificare i comandi in modo da usare il nuovo switch con l'argomento OVERWRITE oppure la nuova forma abbreviata del vecchio switch -REPLACE_DATABASE.

### gbak compatibile con tutte le versioni

C. Valderrama

**(V.2.1)** nella sua ultima versione, *gbak* può essere utilizzato per fare il restore di una qualsiasi precedente versione di Firebird.

### Nascondere il nome utente e la password

A. Peshkov

Richiesta CORE-867

**(V.2.1)** *gbak* adesso cambia il primo parametro per impedire che si possano leggere il nome utente e la password con il comando `ps axf` il Linux.

### gbak -V ed il parametro «contatore»

Durante lo sviluppo di Firebird 1, era stato aggiunto allo switch -V[erbose] della gbak un opzionale argomento numerico <*contatore*> sia per il backup che per il restore. Era stato pensato per specificare un numero ed ottenere un conteggio delle righe processate aggiornato ogni intervallo di righe specificato. Poiché procurava effetti collaterali indesiderati, è stato rimosso ancora prima che uscisse Firebird 1. Purtroppo, sebbene non lo sia mai stato, era stato documentato come esistente nelle note di rilascio ed in altri posto.

## ISQL, utilità di interrogazione

Il lavoro sull'ISQL ha comportato un gran quantitativo di correzioni e l'introduzione di un certo numero di utili nuove caratteristiche.

Una particolarità da notare è che ora i tipi CHAR e VARCHAR definiti col set di caratteri OCTETS (cioè di fatto BINARY) ora mostrano il contenuto in formato esadecimale. Attualmente questa caratteristica non può essere cambiata.

### Nuovi switch

Sono stati aggiunti i seguenti switch alla linea di comando:

### -b[ail], cioè "termina"

D. Ivanov, C. Valderrama

Lo switch -b nella linea di comando impone a isql di terminare riportando al sistema operativo un errore quando viene utilizzata in modalità non interattiva.

Usando degli script in ingresso nella linea di comando, è completamente inappropriato permettere alla isql di continuare l'esecuzione di una serie di comandi dopo che è avvenuto un errore. Pertanto l'opzione "-b[ail]" fa terminare l'esecuzione al primo errore incontrato. Non verranno più eseguiti i successivi comandi dello script in ingresso e isql riporta un codice di errore al sistema operativo.

- Sono stati coperti sicuramente la maggior parte dei casi, ma siccome questo è un lavoro in itinere, se si dovessero trovare eventualmente degli errori non riconosciuti dalla isql bisognerebbe informarne il progetto.
-

Al momento non si differenzia per tipo o codice di errore, qualsiasi valore riportato diverso da zero deve essere considerato un errore. In funzione di altre opzioni (come -o, -m e -m2), isql può mostrare il messaggio d'errore su video oppure inviarlo ad un file.

### Alcune caratteristiche

- Anche se si eseguono script annidati, isql termina qualsiasi esecuzione e ritorna al sistema operativo non appena incontra un errore. Si possono ottenere script annidati quando uno script A richiamato nella linea di comando contiene a sua volta il comando INPUT per caricare lo script B e così via. Isql non effettua controlli culla recursione diretta o indiretta, pertanto se il programmatore sbaglia e lo script A chiama sé stesso o carica uno script B che a sua volta richiama A, isql continua finché esaurisce la memoria o il database riporta un errore, al che, se è stato attivato -bail, ferma ogni attività ed esce.
- Gli errori di DML sono colti durante la preparazione o l'esecuzione, ciò dipende dal tipo di errore.
- In molti casi, isql riporta il numero di linea in cui avviene un errore in un comando DML che fallisce nell'esecuzione di uno script. (C'è di più in error line numbers)
- Gli errori di DDL per default sono colti durante la preparazione o l'esecuzione, poiché isql usa AUTODDL ON per default. Tuttavia se si pone AUTO DLL a OFF, il server si lamenta solo quando lo script richiede una esplicita COMMIT e questo può comprendere vari comandi.
- La caratteristica può essere attivata o disattivata interattivamente o da script attraverso il comando

```
SET BAIL [ON | OFF]
```

  Come altri comandi SET, usando solo SET BAIL semplicemente rovescia lo stato precedente scambiando attivato con disattivato. Usando solo SET mostrerà lo stato di tutti gli switch.
- Anche se BAIL è attivo, non significa che cambia il funzionamento di isql. L'altro requisito che deve essere soddisfatto è che la sessione non sia interattiva. Una sessione non interattiva è quella in cui l'utente chiama isql in modalità processo dandogli in ingresso uno script.

  **Esempio**

```
isql -b -i my_fb.sql -o results.log -m -m2
```

> **Suggerimento**
>
> Tuttavia, se l'utente carica interattivamente isql e successivamente lancia uno script con il comando INPUT, questa resta ancora una sessione interattiva anche se isql sa di eseguire uno script.

  **Esempio**

```
isql
Use CONNECT or CREATE DATABASE to specify a database
SQL> set bail;
SQL> input my_fb.sql;
SQL> ^Z
```

Qualsiasi sia il contenuto dello script, verrà eseguito completamente, errori e tutto, anche se l'opzione BAIL è abilitata.

### -m2 per visualizzare statistiche e PLAN

C. Valderrama

Questa è un'opzione da linea di comando per inviare le statistiche ed i PLAN allo stesso output degli altri comandi (eventualmente con lo switch -o[utput]).

Se l'utente specifica di voler inviare i messaggi su un file, per anni ci sono solo state due sole possibilità:

- o dalla linea di comando, con lo switch -o seguito da il nome di un file
- oppure col comando OUTput seguito dal nome di un file, sia come batch che da sessione interattiva. In qualsiasi caso, specificando solo OUTput è sufficiente per riavere i messaggi in uscita sula console. I messaggi di errore, tuttavia, sono mostrati alla console ma non sono re-direzionati su file.

Sulla linea di comando è stato aggiunto lo switch -m, per inserire i messaggi di errore nel normale flusso di uscita ovunque esso venga redirezionato.

Questo lascia aperto ancora un caso: come far riportare dal server le statistiche sulle operazioni (il comando SET STAT) ed i PLAN SQL. I comandi SET PLAN e SET PLANONLY sono stati sempre trattati come messaggi diagnostici e, come tali, sono stati sempre inviati solo alla console.

Il comando -m2 inserisce le statistiche e le informazioni dei PLAN ovunque venga redirezionato il flusso di uscita.

> **Nota**
>
> Né -m né -m2 hanno una controparte interattiva attraverso un comando SET. Essi possono essere usati solo come opzioni della linea di comando isql.

### -r2 per passare un nome di ruolo con maiuscole e minuscole

C. Valderrama

Il solo scopo di questo parametro è permettere di specificare il nome di un ruolo che contiene caratteri sia maiuscoli che minuscoli.

- Lo switch di default per questo parametro è -r. Ovviamente in tal caso i ruoli passati nella linea di comando sono convertiti tutti in maiuscolo
- Con lo switch -r2, il ruolo viene passato al motore esattamente come è scritto nella linea di comando.

## Nuovi comandi e miglioramenti

I seguenti comandi sono stati aggiunti o migliorati.

### Ctrl-C per interromepre il flusso di un'interrogazione

M. Kubecek
A. dos Santos Fernandes

Richiesta CORE-704

**(V. 2.1)** In una sessione interattiva di isql ora si può interrompere una SELECT usando Ctrl-C. Notare, questo semplicemente blocca il recupero delle righe dalla memoria di ingresso, non cancella l'esecuzione della query.

### Estensione del comando SHOW SYSTEM di isql

A. dos Santos Fernandes

Richiesta CORE-978

**(V. 2.1)** Il comando SHOW <tipo_di_oggetto> serve per mostrare all'utente tutti gli oggetti di quel tipo. Il comando SHOW SYSTEM era stato pensato per mostrare gli oggetti di sistema, ma fin'ora visualizzava solo le tabelle di sistema. Ora elenca le UDF predefinite di sistema incorporate in FB 2.

Può essere migliorato per elencare anche le viste di sistema nel caso ne vengano create nel futuro.

### Il comando SHOW COLLATIONS

A. dos Santos Fernandes

**(V. 2.1)** Elenca tutte le coppie set di caratteri/ordinamenti dichiarati nel database.

### SET HEAD[ing]

C. Valderrama

Alcuni ritengono utile eseguire in isql una SELECT ed ottenerne il risultato in un file, per poterlo elaborare successivamente in un secondo momento, specie se il numero delle colonne rende la visualizzazione di isql poco pratica. Tuttavia, isql per default stampa continuamente le testate di colonna e in tal caso sono una seccatura.

Pertanto, la stampa delle testate che prima era una caratteristica fissa, ora è un'opzione impostabile interattivamente o da script con il comando isql

```
SET HEADing [ON | OFF]
```

Come per altri comensi SET, usare semplicemente SET HEAD scambia lo stato fra attivo e non attivo.

> **Nota**
>
> Non esiste un'opzione nella linea di comando per impostare le testate.

Usando il solo comando SET mostra lo stato di SET HEAD, insieme a tutti gli altri switch che possono essere impostati in isql.

### SET SQLDA_DISPLAY ON/OFF

A. dos Santos Fernandes

Il comando SQLDA_DISPLAY mostra i parametri di SQLDA in ingresso per le INSERT, UPDATE e DELETE. Era precedentemente disponibile solo nei pacchetti di DEBUG e adesso è stato inserito anche nei pacchetti pubblici. Visualizza le informazioni delle SQLVAR. Ogni SQLVAR rappresenta un campo nel XSQLDA, che è la struttura principale usata nelle API di FB per comunicare con i client trasferendo i dati da e per il server.

> **Nota**
>
> Lo stato di questa opzione non è visualizzato nella lista generata con il comando `SET;` usato in isql per visualizzare la maggior parte delle opzioni.

### SET TRANSACTION migliorato

C. Valderrama

Il comando SET TRANSACTION è stato migliorato in modo tale che adesso sono supportate tutte le opzioni di TPB:

- NO AUTO UNDO
- IGNORE LIMBO
- LOCK TIMEOUT <numero>

*Esempio*

```
SET TRANSACTION WAIT SNAPSHOT NO AUTO UNDO LOCK TIMEOUT 10
```

Vedere anche la documentazione in *doc/sql.extensions/README.set_transaction.txt*.

### SHOW DATABASE riporta anche la versione di ODS

C. Valderrama

La versione ODS (On-Disk Structure) viene visualizzata con il comando SHOW DATABASE (C. Valderrama)

### Possibilità di mostrare il numero di linea in cui vi è un errore in uno script

C. Valderrama

Nelle versioni precedenti, l'unico modo possibile per sapere dove uno script cadeva in errore era adoperare il comando -e per vedere i comandi, -o per inviare l'uscita su un file e -m per inviare anche i messaggi di errore su quel file. In tal modo si potevano vedere i comandi *isql* che venivano eseguiti e i possibili errori esistenti. Lo script continuava fino alla fine, ovviamente. Per alcuni errori DSQL, il server ora comunica il numero di linea relativo al singolo comando (o frase) che sta' eseguendo. Per altri errori, si può solo sapere che un certo comando ha problemi.

Con l'opzione -b già descritta sopra in (1), l'utente ha la possibilità di fermare lo script isql non appena rileva un errore, ma si ha ancora bisogno dell'opzione eco ( -e -o ) su un file di uscita per sapere quale comando ha causato l'errore.

Ora, la possibilità di segnalare la linea nello script che provoca un errore permette all'utente di trovare nello script il punto incriminato. Se il server indica la linea e la colonna si può identificare esattamente dove nello script DML è avvenuto il problema. Quando il server invece indica semplicemente un errore, si ha la linea iniziale, relativa all'intero script, del comando che ha causato un problema.

Questo funziona anche in script annidati, in particolare, se lo script SA include lo script SB e SB crea un problema, il numero di linea è relativo a SB. Alla fine di SB, isql continua eseguendo il resto di SA e pertanto isql ritorna a contare le linee di SA, poiché ogni file ha un contatore di linee distinto. Si dice che uno script SA include lo script SB quando in SA si usa il comando INPUT per caricare SB.

Le linee sono contate nel modo in cui riesce a farlo il substrato di I/O, in funzione cioè di come riesce a separare le linee. Nei sistemi in cui esiste EDITLINE, una linea è quanto la readline() recupera in una singola chiamata. Il limite per la lunghezza massima di una linea non cambia e rimane fissato a 32767 byte.

### *Migliorato l'aiuto da comando*

M. Kubecek

Se si adoperano dei parametri sconosciuti, isql mostra tutti i suoi parametri da linea di comando e la relativa spiegazione invece sella semplice lista di switch permessi.

```
opt/firebird/bin] isql -?
Unknown switch: ?
usage:    isql [options] [<database>]
        -a(all)                 extract metadata incl. legacy non-SQL tables
        -b(ail)                 bail on errors (set bail on)
        -c(ache) <num>          number of cache buffers
        -ch(arset) <charset>    connection charset (set names)
        -d(atabase) <database>  database name to put in script creation
        -e(cho)                 echo commands (set echo on)
        -ex(tract)              extract metadata
        -i(nput) <file>         input file (set input)
        -m(erge)                merge standard error
        -m2                     merge diagnostic
        -n(oautocommit)         no autocommit DDL (set autoddl off)
        -now(arnings)           do not show warnings
        -o(utput) <file>        output file (set output)
        -pag(elength) <size>    page length
        -p(assword) <password>  connection password
        -q(uiet)                do not show the message "Use CONNECT..."
        -r(ole) <role>          role name
        -r2 <role>              role (uses quoted identifier)
        -sqldialect <dialect>   SQL dialect (set sql dialect)
        -t(erminator) <term>    command terminator (set term)
        -u(ser) <user>          user name
        -x                      extract metadata
        -z                      show program and server version
```

## gsec per l'autenticazione

Le modifiche all'utilità *gsec* includono:

### *valore di ritorno per gsec*

C. Valderrama

*gsec* adesso riporta un codice di errore quando non viene usato interattivamente. Zero indica successo; ogni altro valore indica problemi.

## gfix utilità del server

Le modifiche apportate alla utilità *gfix* includono:

### *Nuovi stati (o modi) di shutdown*

N. Samofatov, D. Yemanov

Le opzioni di `gfix -shut[down]` sono state estese per includere i due stati o modi extra per gestire lo shutdown.

**Nuova sintassi**

```
    gfix <comando> [<stato>] [<opzioni>]

<comando> ::= {-shut | -online}
<stato> ::= {normal | multi | single | full}
<opzioni> ::= {-force <timeout> | -tran | -attach}
```

- stato «normal» = database online

- stato «multi» = modo shutdown multi-utenza (quello compatibile, permesse connessioni illimitate da SYSDBA/proprietario)

- stato «single» = shutdown in singola-utenza (solo una connessione permessa, usato dal processo di restore)

- stato «full» = shutdown completo o esclusivo (non sono permesse connessioni)

> **Nota**
>
> «Multi» è lo stato di default per il comando -shut, «normal» invece quello del comando -online.

I modi possono essere modificati sequenzialmente secondo questo ordine:

normal <-> multi <-> single <-> full

**Esempi**

```
gfix -shut single -force 0
gfix -shut full -force 0
gfix -online single
gfix -online
```

Non si può usare `-shut` per far salire un database di un livello «più in linea» e non si può usare `-online` per rendere un database più protetto (viene dato un errore).

Queste sequenze sono pertanto proibite:

```
gfix -shut single -force 0
gfix -shut multi -force 0

gfix -online
gfix -online full

gfix -shut -force 0
gfix -online single
```

# Pacchetti e installazioni

## *Aggiunto a instsvc.exe un parametro per il nome dell'istanza*

D. Yemanov

Richiesta CORE-673

**(V.2.1)** `instsvc.exe` adesso permette installazioni multi istanza.

## *Revisionata la documentazione di instalzzazione su Win32*

P. Reeves

**(V.2.1)** La documentazione per l'installazione a linea di comando su Windows è stata rivista. Vedere `doc/install_windows_manually.txt`.

## *Aiuto sugli switch a linea di comando*

Richiesta CORE-548

**(V.2.1)** L'aiuto in linea è disponibile sugli switch dell'installazione a linea di comando su Windows.

## *Riconoscimento di Gentoo/FreeBSD durante l'installazione*

A. Peshkov

Richiesta CORE-1047

Durante la configurazione sono ora riconosciuti sia Gentoo che FreeBSD, facendo in modo che l'installazione possa funzionare con successo su queste piattaforme.

Capitolo 14

# Funzioni esterne (UDF)

## Segnalare un valore SQL NULL attraverso un puntatore Null

C. Valderrama

Prima di Firebird 2, gli autori delle UDF potevano ipotizzare che le loro UDF riportassero un NULL, ma non avevano possibilità di verificarlo. Questo provocava una serie di problemi con le UDF. Spesso si è ritenuto di poter interpretare come se fossero NULL nel passaggio dei parametri rispettivamente una stringa vuota, il valore numerico zero e la data di origine delle date.

L'autore non può sempre ritenere un particolare valore numerico NULL se l'UDF viene compilata per un contesto in cui sarebbe stato noto che normalmente il NULL non viene riconosciuto.

Molte UDF, inclusa la libreria `ib_udf` distribuita con Firebird, hanno sempre ritenuto che una stringa vuota servisse a segnalare più probabilmente un parametro a NULL invece che una stringa di lunghezza zero. Il trucchetto potrebbe funzionare con parametri di tipo CHAR, poiché la lunghezza minima dichiarabile di un CHAR è uno, e dovrebbe contenere un carattere di spazio: pertanto uno zero binario nella prima posizione avrebbe l'effetto desiderato di segnalare il NULL.

Tuttavia non è applicabile ai VARCHAR o CSTRING, dove invece la lunghezza zero è valida.

Un'altra soluzione sarebbe stata fare affidamento sui descrittori, ma questo avrebbe imposto molte più cose da controllare di quante se ne potevano affrontare. Il problema più spinoso è che il motore non rispetta il tipo dichiarato per il parametro: semplicemente invia ciò che ha per quel parametro, così che è l'UDF a dover decidere se rifiutare il dato o cercare di convertirlo nel tipo di dato previsto.

Poiché le UDF non hanno un meccanismo formale per segnalare un errore, il valore riportato deve essere utilizzato anche come indicatore.

Il problema di fondo è mantenere la semplicità tipica delle dichiarazioni (senza descrittori) ed al tempo stesso essere in grado di segnalare il NULL.

Il motore normalmente passava i parametri alle UDF per riferimento. In pratica, ciò significa passare un puntatore al dato per indicare che abbiamo un NULL. Tuttavia non si può correre il rischio di rompere un imprecisato numero di UDF esistenti pubbliche e private che al momento non si aspettano un NULL. Era necessario migliorare la sintassi per far richiedere esplicitamente la gestione del NULL.

La soluzione, pertanto, è restringere la richiesta per segnalare il NULL alle sole UDF che sono in grado di interagire col nuovo scenario. Per evitare di aggiungere nuove parole chiave, la parola NULL viene aggiunta al tipo del parametro dell'UDF senza effettuare altre modifiche.

**Esempio**

```
declare external function sample
  int null
```

```
returns int by value...;
```

Usando già le funzioni della ib_udf e per sfruttare segnalazione e riconoscimento del NULL in alcune funzioni, ci si può connettere al database e lanciare lo script `../misc/upgrade/ib_udf_upgrade.sql` che è nella cartella di Firebird, ed effettuare il commit.

> **Attenzione**
>
> Si raccomanda di fare questo quando non ci sono altri utenti connessi al database.

Il codice nelle funzioni elencate in quello script è stato modificato per riconoscere has il NULL solo quando viene segnalato dal motore. Pertanto, a partire da FB v2,le funzioni `rtrim()`, `ltrim()` ed altre funzioni sulle stringhe non ritengono che una stringa vuota sia una stringa NULL.

Le funzioni continuano a funzionare anche se non si effettua lo script di aggiornamento: semplicemente non sono in grado di decifrare l'arrivo di un NULL.

Se non si è mai usata ib_udf in un database ed avendone la necessità, ci si deve collegare al database, lanciare lo script `../udf/ib_udf2.sql`, preferibilmente sempre quando non vi sono altri utenti connessi al database e subito dopo fare il commit.

> **Nota**
>
> - Notare il "2" alla fine del nome.
> - Lo script originale per FB v1.5 è ancora disponibile nella stessa cartella.

## Miglioramenti dei messaggi diagnostici delle librerie UDF

A. Peshkov

La diagnostica per una UDF mancante o inutilizzabile era poco chiara, per cui non si capiva se mancava il modulo o era impossibile accedervi a causa della configurazione di UDFAccess in firebird.conf. Adesso ci sono messaggi distinti e comprensibili per entrambi i casi.

## UDF aggiunte e modificate

Le UDF aggiunte o migliorate nelle librerie di Firebird 2.0 sono:

### *IB_UDF_rand() e IB_UDF_srand()*

F. Schlottmann-Goedde

Nelle precedenti versioni, la funzione esterna `rand()` imposta il punto di partenza per il generatore di numeri casuali basandolo sull'ora corrente e poi generava un valore pseudo-casuale.

```
srand((unsigned) time(NULL));
return ((float) rand() / (float) RAND_MAX);
```

Questo algoritmo, se si effettuano due chiamate al'interno dello stesso secondo, ha il problema che entrambe riportano lo stesso valore.

Per risolvere il problema, la funzione `rand()` è stata cambiata in Firebird 2.0 in modo tale che il punto di partenza non sia impostato esplicitamente. Questo fa in modo che vengano ottenuti sempre valori diversi.

Per ottenere il comportamento compatibile a disposizione in caso fosse necessario, è stata aggiunta la funzione *srand()*. Fa esattamente quello che faceva la versione precedente della `rand()`.

### IB_UDF_lower

La funzione `IB_UDF_lower()` nella libreria IB_UDF era in conflitto con la nuova funzione interna `lower()`, cercandola di definire in un database con lo script `ib_udf.sql` di una versione precedente di Firebird.

```
/* ib_udf.sql declaration that now causes conflict */
DECLARE EXTERNAL FUNCTION lower
    CSTRING(255)
    RETURNS CSTRING(255) FREE_IT
    ENTRY_POINT 'IB_UDF_lower' MODULE_NAME 'ib_udf';
```

Il problema è stato risolto nell'ultima versione del nuovo script `ib_udf2.sql`, dove la vecchia UDF è dichiarata usando un identificatore quotato fra doppi apici.

```
/* New declaration in ib_udf2.sql */
DECLARE EXTERNAL FUNCTION "LOWER"
  CSTRING(255) NULL
  RETURNS CSTRING(255) FREE_IT
  ENTRY_POINT 'IB_UDF_lower' MODULE_NAME 'ib_udf';
```

> **Suggerimento**
>
> È preferibile utilizzare la funzione interna LOWER() piuttosto che chiamare la UDF.

## Altre modifiche alle UDF

### Modifiche nella compilazione

C. Valderrama Contributor

La libreria FBUDF non dipende più dalla libreria `[lib]fbclient` per essere compilata.

Capitolo 15

# Novità e modifiche nei parametri di configurazione

## Autenticazioni

A. Peshkov

**(V.2.1)** Sulla piattaforma Windows, a partire dalla versione 2.1, si può usare *authentication* per configurare la modalità di accesso al server nel caso in cui se ne preferisca una diversa dal default *native*.

> **Importante**
>
> In Firebird 2.1 precedentemente alla v.2.1.3, il default era *mixed*. Facendo un aggiornamento alla v.2.1.3 dalla v.2.1, v.2.1.1 o v.2.1.2 senza impostare esplicitamente il parametro, l'autenticazione *trusted* non funziona. Per tornare all'impostazione precedente, togliere il commento al parametro e impostarlo a *mixed*. (Tracker reference CORE-2376).

- *trusted* fa uso del sistema di autenticazione nativo di Windows, la c.d. «trusted authentication» che, nelle giuste condizioni, dovrebbe essere il modo più sicuro per autenticare gli utenti in Windows.
- *native* imposta il funzionamento dell'autenticazione nel tradizionale modo di Firebird, richiedendo agli utenti di identificarsi con un nome utente ed una password definiti nel database di sicurezza.
- *mixed* permette entrambi i funzionamenti.

## RelaxedAliasChecking

V. Khorsun

**(V.2.1)** *RelaxedAliasChecking* è un nuovo parametro di configurazione aggiunto al fine di eliminare la restrizione imposta in Firebird 2.0.x sull'uso degli alias nelle relazioni ed i nomi di tabella nelle query. Per esempio, con RelaxedAliasChecking posto a vero (=1) in firebird.conf, la seguente query non riporta errore in Firebird 2.1, mentre fallirebbe in v.2.0.x, o in v.2.1 col parametro impostato al suo default cioè 0:

```
SELECT ATABLE.FIELD1, B.FIELD2
  FROM ATABLE A JOIN BTABLE B
  ON A.ID = BTABLE.ID
```

**Attenzione**

Questa è una *utilità temporanea* il cui scopo è quello di concedere ai sistemi che hanno del codice per Interbase o per versioni vecchie di Firebird, il tempo necessario a migrare ad una sintassi orientata allo standard SQL.

- Pertanto, questo parametro va abilitato solamente se si ha nelle applicazioni o nei moduli PSQL del codice che ha questo tipo di problema. Non deve essere considerato un invito a scrivere del codice non professionale!
- Consideratela come una bomba a tempo. Verrà completamente rimosso in una versione successiva.

## MaxFileSystemCache

V. Khorsun

**(V.2.1)** Imposta una soglia che determina quando la cache delle pagine di Firebird può o meno essere duplicata nella cache del sistema. Se questo parametro è impostato ad un qualsiasi valore intero positivo, il suo effetto dipende dalla dimensione effettiva di default della cache delle pagine: se questa è (in pagine) inferiore al valore di MaxFileSystemCache (in pagine) allora viene abilitata la cache del sistema, altrimenti resta disabilitata.

**Nota**

Questo è valido sia che la dimensione della memoria della cache delle pagine sia impostata implicitamente dall'impostazione DefaultDBCachePages o esplicitamente dall'attributo nell'header del database.

Pertanto,

- Per disabilitare del tutto il sistema di cache del sistema, impostare MaxFileSystemCache a zero.
- Per tenerlo sempre abilitato, impostare MaxFileSystemCache ad un valore abbastanza grande da superare la dimensione della cache delle pagine del database. Ricordare che l'effetto di questo valore è influenzato da modifiche successive alla dimensione della cache delle pagine.

**Importante**

Il valore di default per MaxFileSystemCache è 65536 pagine, cioè la cache del sistema è abilitata.

## DatabaseGrowthIncrement

V. Khorsun

**(V.2.1)** Al fine di controllare meglio la preallocazione dello spazio su disco, è stato aggiunto in `firebird.conf` il nuovo parametro *DatabaseGrowthIncrement*. Esso rappresenta *in byte* il limite superiore della dimensione dello spazio su disco di cui viene richiesta la preallocazione quando le pagine devono essere scritte dalla cache. Il default è di 134,217,728 byte (128 MB).

Informazioni propedeutiche si trovano in Enlarge Disk Allocation Chunks nel capitolo relativo ai «Miglioramenti generali per Firebird 2.1».

Quando il motore ha bisogno di più spazio su disco, alloca un blocco che è 1/16 dello spazio già occupato dal database, ma non meno di 128 KB e non maggiore del valore di DatabaseGrowthIncrement. Il valore di

DatabaseGrowthIncrement può essere innalzato per aumentare la dimensione massima di allocazione dei nuovi blocchi dal default di 128 MB. Impostandolo a zero si disabilita la preallocazione.

> **Nota**
>
> - Il limite minimo della dimensione del blocco è bloccato appositamente a 128 KB e non può essere modificato.
> - Non c'è preallocazione per i file ombra del database (database shadow files).
> - La preallocazione è disabilitata per i database che hanno attiva l'impostazione «No reserve».

## ExternalFileAccess

A. Peshkov

Come modificata in Firebird 2, un nuovo file esterno viene creato per default nella directory specificata per prima in ExternalFilesAccess.

## LegacyHash

A. Peshkov

Questo parametro permette di rigettare in Firebird 2 un vecchio hash DES nel database di sicurezza aggiornato. Se non si usa la procedura di aggiornamento del database di sicurezza, questo parametro non influenza l'operatività di Firebird. Un hash DES non può arrivare nel nuovo security2.fdb.

Vedere anche nella sezione relativa alla sicurezza in Security DB Upgrade per istruzioni sulle metodologie di aggiornamento da Firebird 1.5 del security.fdb (o di un vecchio isc4.gdb rinominato) al nuovo database di sicurezza.

Il valore di default è 1 (true).

## Redirection

A. Peshkov

Questo parametro permette di controllare la redirezione di richieste remote. Controlla le prestazioni del multi-hop che erano bucate già dai tempi di InterBase 6 e sono state riattivate in Firebird 2.

### *Riguardo il Multi-hop*

Connettendosi ad un database usando più host nella stringa di connessione, solo l'ultimo nella lista di questi è quello che realmente apre il database. Gli altri host agiscono da intermediari sempre sulla porta gds_db. Precedentemente questa caratteristica era disponibile sempre. Ora può invece essere configurata.

Infatti la redirezione remota è disabilitata per defrault.

**Attenzione**

Nel caso in cui si considerasse di abilitare il multi-hop, si prega di considerare con attenzione quanto descritto in Warning nel capitolo sulla Sicurezza e nella documentazione di questo parametro nel file di configurazione `firebird.conf`.

## GCPolicy

V. Khorsun

Riguarda la «Garbage collection policy». Adesso è possibile scegliere quale gestione della «raccolta della spazzatura» adottare nel SuperServer. I possibili valori sono *cooperative*, *background* e *combined*, come spiegato nelle note della `GCPolicy` in `firebird.conf`.

Non è applicabile al server tipo Classic, perchè supporta solo la GC cooperative.

## OldColumnNaming

P. Reeves

Il parametro OldColumnNaming è stato ripreso da Firebird 1.5.3. Questo parametro permette agli utenti di attribuire alle colonne, calcolate nelle espressioni di SELECT, i nomi che avevano prima della versione 1.5 di Firebird. Il default in installazione è 0 (disabilitato, cioè nuova convenzione sui nomi). Se viene abilitato, il motore non attribuirà più identificatori quali ad esempio CONCATENATION per campi derivati dove lo sviluppatore esplicitamente non ha attribuito un identificatore.

**Importante**

Questa impostazione influenza tutti i database acceduti dal server e potenzialmente può produrre eccezioni o risultati inaspettati quando sono attive applicazioni diverse.

## UsePriorityScheduler

A. Peshkov

Impostando questo parametro a zero, ora si disabilita completamente lo scambio delle priorità dei thread. Influenza solo il Superserver per Win32.

## È stato cambiato TCPNoNagle

K. Kuznetzov

Il valore di default per TcpNoNagle è adesso TCP_NODELAY.

## Modificato il comportamento di IPCName

N. Samofatov

**(V.2.1, non documentato precedentemente la V.2.1.3)** *IPCName*, che ha il default a FIREBIRD a partire dalla versione 2.0, è lo spazio dei nomi del kernel in cui l'istanza XNET viene creata per le connessioni locali dirette su Windows. Su Vista ed altre piattaforme Windows, è spesso necessario modificare questo parametro aggiungendo il prefisso «Global\» per assicurare ad un client locale non privilegiato (che cioè ha un accredito ristretto) possa avere l'autorità necessaria a creare quello spazio dei nomi.

Una modifica in Firebird 2.1 fa in modo che la routine di connessione possa applicare il prefisso al default IpcName incondizionatamente se il primo tentativo dell'utente dovesse fallire a causa di permessi troppo restrittivi.

## Parametri rinominati

### *SortMemBlockSize cambiato in TempCacheBlockSize*

D. Yemanov

Considerato un termine più appropriato.

### *SortMemUpperLimit cambiato in TempCacheUpperLimit*

D. Yemanov

Considerato un termine più appropriato.

## Parametri rimossi o sconsigliati

### *CreateInternalWindow*

D. Yemanov

Questa opzione non è più richiesta per lanciare istanze multiple del server ed è stata rimossa.

### *DeadThreadsCollection non è più usato*

A. Peshkov

Il parametro DeadThreadsCollection non è più usato. I thread defunti sono adesso rilasciati «al volo» in modo efficiente, rendendo la configurazione inutile. Firebird 2.0 ignora silenziosamente questo parametro.

Capitolo 16

# Le squadre del progetto Firebird 2

**Tabella 16.1. Team di sviluppo di Firebird**

| Autore | Paese | Attività principale |
|---|---|---|
| Dmitry Yemanov | Federazione Russa | Progettazione e sviluppo a tempo pieno, responsabile della squadra di sviluppo |
| Alex Peshkov | Federazione Russa | Coordinatore del sistema di sicurezza, esperto nella pacchettizzazione e del porting |
| Claudio Valderrama | Cile | Esamina il codice sorgente, scova e risolve i buchi, progetta ed implementa su vari fronti fra cui miglioramenti all'ISQL e alle UDF |
| Vladislav Khorsun | Ucraina | Ingegnere DB, progetta ed implementa le funzionalità SQL |
| Arno Brinkman | Olanda | Miglioramenti al sistema di indici e di ottimizzazione; nuove funzionalità DSQL |
| Adriano dos Santos Fernandes | Brasile | Sviluppo del nuovo set di caratteri internazionali; gestione dei campi testo e BLOB; nuove funzionalità DSQL; esaminatore del codice sorgente |
| Nickolay Samofatov | Federazione Russa/Canada | Ha progettato ed implementato NBackup; ha corretto il codice ed ingegnerizzato il DB durante parte dello sviluppo della versione 2.0 |
| Paul Beach | Francia | Responsabile dei rilasci; pacchettizzazioni HP-UX ; MacOS e Solaris |
| Pavel Cisar | Repubblica Ceca | Progettazione e coordinamento dei sistemi di test; sviluppo e manutenzione dei driver Python per Firebird |
| Philippe Makowski | Francia | Test della qualità del prodotto |
| Paul Reeves | Francia | Pacchettizzazioni ed installazioni Win32 |
| Sean Leyne | Canada | Organizzazione del bugtracker |
| Dimitrios Ioannides | Grecia | Implementazione del nuovo bugtracker Jira |
| Ann Harrison | U.S.A. | Specialista tecnico molto attivo |
| Jim Starkey | U.S.A. | Specialista tecnico attivo; occasionali correzioni al codice |
| Roman Rokytskyy | Germania | Implementazione e coordinamento di Jaybird |

| Autore | Paese | Attività principale |
|---|---|---|
| Ryan Baldwin | Inghilterra | Sviluppo del Jaybird Type 2 driver |
| Evgeny Putilin | Federazione Russa | Implementazione delle STORED PROCEDURES in Java |
| Carlos Guzman Alvarez | Spagna | Sviluppo e coordinamento del provider .NET per Firebird fino al 2007 |
| Jiri Cincura | Repubblica Ceca | Sviluppo e coordinamento del provider .NET per Firebird a partire dal gennaio 2008 |
| Alexander Potapchenko | Russia | Coordinatore del driver ODBC/JDBC per Firebird |
| David Rushby (d.) | U.S.A. | Coordinamento e sviluppo dell'interfaccia Python KInterbase per Firebird fino alla sua morte per incidente nel luglio 2007 |
| Konstantin Kuznetsov | Federazione Russa | Pacchettizzazioni Solaris Intel |
| Paul Vinkenoog | Olanda | Coordinatore del progetto di documentazione Firebird; scrittore di documentazione e sviluppo di utilità |
| Norman Dunbar | Inghilterra | Documentazione |
| Pavel Menshchikov | Federazione Russa | Traduttore della documentazione |
| Tomneko Hayashi | Giappone | Traduttore della documentazione |
| Umberto (Mimmo) Masotti | Italia | Traduttore della documentazione |
| Olivier Mascia | Belgio | Sviluppatore dell'interfaccia C++ IBPP; reimplementazione dei servizi di installazione per Win32 |
| Oleg Loa | Federazione Russa | Contribuzioni varie |
| Grzegorz Prokopski | Ungheria | Pacchettizzazioni Debian |
| Erik Kunze | Germania | Abilitazione dei database su periferica fisica (raw device); porting su SINIX-Z |
| Helen Borrie | Australia | Redattrice delle note di rilascio; capo dell'ortodossia del verbo Firebird. |

# Appendice A: Le nuove funzioni integrate

**(Firebird 2.1)**

| Funzione | Formato | Descrizione |
|---|---|---|
| *ABS* | ABS( <number> ) | Riporta il valore assoluto di un numero. |
| `select abs(amount) from transactions;` | | |
| *ACOS* | ACOS( <number> ) | Riporta l'arco coseno di un numero . L'argomento deve essere compreso tra -1 e 1 e riporta un valore tra 0 e # (cioè 3.1416...). |
| `select acos(x) from y;` | | |
| *ASCII_CHAR* | ASCII_CHAR( <number> ) | Riporta carattere ASCII specificato dal codice. L'argomento deve essere tra 0 e 255, il risultato è nel set di caratteri NONE. |
| `select ascii_char(x) from y;` | | |
| *ASCII_VAL* | ASCII_VAL( <string> ) | Riporta il codice ASCII del primo carattere della stringa.<br>1. Vale 0 se la stringa è vuota<br>2. Dà un errore se il primo carattere è multi-byte |
| `select ascii_val(x) from y;` | | |
| *ASIN* | ASIN( <number> ) | Riporta l'arco seno di un numero reale compreso tra -1 e 1. Riporta un risultato compreso tra -#/2 e #/2. |
| `select asin(x) from y;` | | |
| *ATAN* | ATAN( <number> ) | Riporta l'arco tangente di un numero reale che è un valore compreso tra -#/2 e #/2. |
| `select atan(x) from y;` | | |
| *ATAN2* | | |

| Funzione | Formato | Descrizione |
|---|---|---|
| | ATAN2( <number1>, <number2> ) | Riporta l'arco tangente del rapporto number1/number2. Ripora un valore tra -# e #. |
| `select atan2(x, y) from z;` | | |
| *BIN_AND* | BIN_AND( <number> [, <number> ...] ) | Riporta l'AND binario effettuato su tutti gli argomenti presenti. |
| `select bin_and(flags, 1) from x;` | | |
| *BIN_OR* | BIN_OR( <number> [, <number> ...] ) | Riporta l'OR binario effettuato su tutti gli argomenti presenti. |
| `select bin_or(flags1, flags2) from x;` | | |
| *BIN_SHL* | BIN_SHL( <n1>, <n2> ) | Riporta lo shift left binario effettuato sugli argomenti interi (n1 << n2). |
| `select bin_shl(flags1, 1) from x;` | | |
| *BIN_SHR* | BIN_SHR( <n1>, <n2> ) | Riporta lo shift right binario effettuato sugli argomenti interi (n1 >> n2). |
| `select bin_shr(flags1, 1) from x;` | | |
| *BIN_XOR* | BIN_XOR( <number> [, <number> ...] ) | Riporta lo XOR binario effettuato su tutti gli argomenti presenti. |
| `select bin_xor(flags1, flags2) from x;` | | |
| *BIT_LENGTH* | BIT_LENGTH( <string> \| <string_expr> ) | Riporta la lunghezza di una stringa in bit |
| `select`<br>`  rdb$relation_name,`<br>`  bit_length(rdb$relation_name),`<br>`  bit_length(trim(rdb$relation_name))`<br>`from rdb$relations;` | | |
| *CEIL \| CEILING* | { CEIL \| CEILING }( <number> ) | Riporta il più piccolo intero maggiore o uguale all'argomento dato. |
| `1) select ceil(val) from x;`<br>`2) select ceil(2.1), ceil(-2.1) from rdb$database;  -- riporta 3, -2` | | |

| Funzione | Formato | Descrizione |
|---|---|---|
| *CHAR_LENGTH* \| *CHARACTER_LENGTH* | CHAR_LENGTH( <string> \| <string_expr> ) | Riporta il numero di caratteri in una stringa o un'espressione. |

```
select
  rdb$relation_name,
  char_length(rdb$relation_name),
  char_length(trim(rdb$relation_name))
from rdb$relations;
```

| | | |
|---|---|---|
| *COS* | COS( <number> ) | Valuta il coseno di un angolo espresso in radianti riportando un valore compreso tra -1 ed 1. |

```
select cos(x) from y;
```

| | | |
|---|---|---|
| *COSH* | COSH( <number> ) | Riporta il coseno iperbolico di un numero. |

```
select cosh(x) from y;
```

| | | |
|---|---|---|
| *COT* | COT( <number> ) | Riporta la cotangente, cioè 1 / tan(argument). |

```
select cot(x) from y;
```

| | | |
|---|---|---|
| *DATEADD* | Vedi sotto | Riporta un valore date/time/timestamp aumentato o diminuito secondo quanto specificato. |

Formato:

```
DATEADD( <number> <timestamp_part> TO <date_time> )
DATEADD( <timestamp_part>, <number>, <date_time> )
timestamp_part ::= { YEAR | MONTH | DAY | HOUR |
                     MINUTE | SECOND | MILLISECOND }
```

1. YEAR, MONTH e DAY non possono essere usati con espression di tipo TIME.
2. HOUR, MINUTE, SECOND e MILLISECOND non possono essere usati con espressioni di tipo DATE.
3. Tutti i timestamp_part possono essere usati con espressioni TIMESTAMP.

Esempi

```
select dateadd(day, -1, current_date) as yesterday
  from rdb$database;

  /* oppure con la sintassi alternativa */
select dateadd(-1 day to current_date) as yesterday
  from rdb$database;
```

| Funzione | Formato | Descrizione |
|---|---|---|
| *DATEDIFF* | Vedi sotto | Riporta un valore numerico che rappresenta l'intervallo di tempo esistente fra i due valori date/time/timestamp. |

Formato:

```
DATEDIFF( <timestamp_part> FROM <date_time> TO <date_time> )
DATEDIFF( <timestamp_part>, <date_time>, <date_time> )
timestamp_part ::= { YEAR | MONTH | DAY |
                     HOUR | MINUTE | SECOND | MILLISECOND }
```

1. Il valore riportato è positivo quando il secondo termine è maggiore del primo, zero se sono uguali, negativo viceversa.
2. Il confronto di espressioni DATE con espressioni TIME non è valido.
3. YEAR, MONTH, e DAY non possono essere usati con tipi TIME.
4. HOUR, MINUTE, SECOND e MILLISECOND non possono essere usati con espressioni DATE.
5. Tutti i valori timestamp_part possono essere usati con espressioni timestamp.

Esempio

```
select datediff(
  DAY, (cast('TOMORROW' as date) -10), current_date)
    as datediffresult
from rdb$database;
```

| | | |
|---|---|---|
| *DECODE* | Vedi sotto | DECODE è un'abbreviazione per l'espressione CASE ... WHEN ... ELSE. |

Formato:

```
DECODE( <expression>, <search>, <result>
  [ , <search>, <result> ... ] [, <default> ])
```

Esempio

```
  select decode(state, 0, 'deleted', 1, 'active', 'unknown') from things;
```

| | | |
|---|---|---|
| *EXP* | EXP( <number> ) | Riporta l'esponenziale dell'argomento. |

```
  select exp(x) from y;
```

| | | |
|---|---|---|
| *FLOOR* | FLOOR( <number> ) | Riporta il maggiore intero che è minore o uguale all'argomento dato. |

```
    1) select floor(val) from x;
```

| Funzione | Formato | Descrizione |
|---|---|---|
| 2) select floor(2.1), floor(-2.1)<br>from rdb$database; -- riporta 2, -3 | | |
| *GEN_UUID* | GEN_UUID() | Riporta un numero univoco del tipo UUID. Non ha argomenti. |
| insert into records (id) value (gen_uuid()); | | |
| *HASH* | HASH( <string> ) | Riporta il calcolo dell'HASH di una stringa. |
| select hash(x) from y; | | |
| *LEFT* | LEFT( <string>, <number> ) | Riporta il numero specificato di caratteri della parte sinistra di una stringa data. |
| select left(name, char_length(name) - 10)<br>from people<br>where name like '% FERNANDES';<br><br>1. La prima posizione di una stringa è 1, non 0.<br>2. Se l'argomento <number> non è intero, si applica un arrotondamento di tipo bancario. | | |
| *LN* | LN( <number> ) | Riporta il logaritmo del numero dato. |
| select ln(x) from y; | | |
| *LOG* | LOG( <number>, <number> ) | LOG(x, y) riporta il logaritmo in base x di y. |
| select log(x, 10) from y; | | |
| *LOG10* | LOG10( <number> ) | Riporta il logaritmo in base decimale dell'argomento. |
| select log10(x) from y; | | |
| *LOWER* | LOWER( <string> ) | **(v.2.0.x)** Riporta la stringa data convertita tutta in caratteri minuscoli. |
| isql -q -ch dos850<br><br>SQL> create database 'test.fdb';<br>SQL> create table t (c char(1) character set dos850);<br>SQL> insert into t values ('A');<br>SQL> insert into t values ('E');<br>SQL> insert into t values ('Á'); | | |

| Funzione | Formato | Descrizione |
|---|---|---|
| SQL> insert into t values ('É');<br>SQL> select c, lower(c) from t;<br>C LOWER<br>====== ======<br>A a<br>E e<br>Á á<br>É é | | |
| *LPAD* | LPAD( <string>, <number> [, <string> ] ) | LPAD(string1, length, string2) aggiunge ripetutamente string2 all'inizio di string1 finchè la lunghezza totale non raggiunge length. |
| 1. Se la seconda stringa manca, si considera uno spazio.<br>2. Se la stringa risultante supera la lunghezza data, viene troncata la seconda stringa.<br>Esempio<br>`select lpad(x, 10) from y;` | | |
| *MAXVALUE* | MAXVALUE( <value> [, <value> ...] ) | Riporta il maggiore fra una lista di valori. |
| `select maxvalue(v1, v2, 10) from x;` | | |
| *MINVALUE* | MINVALUE( <value> [, <value> ... ) | Riporta il minore di una lista di valori. |
| `select minvalue(v1, v2, 10) from x;` | | |
| *MOD* | MOD( <number>, <number> ) | Modulo: MOD(X, Y) è il resto della divisione di X per Y. |
| `select mod(x, 10) from y;` | | |
| *OCTET_LENGTH* | OCTET_LENGTH( <string> \| <string_expr> ) | Riporta la lunghezza in bytes di una stringa o di un'espressione. |
| `select`<br>`  rdb$relation_name,`<br>`  octet_length(rdb$relation_name),`<br>`  octet_length(trim(rdb$relation_name))`<br>`from rdb$relations;` | | |
| *OVERLAY* | Vedi sotto | Sostituisce in string1 i caratteri a partire da <start> per <length> con <string2> e la riporta. |
| Formato: | | |

| Funzione | Formato | Descrizione |
|---|---|---|

```
OVERLAY( <string1> PLACING <string2> FROM <start> [ FOR <length> ] )
```

La funzione OVERLAY è equivalente a:

```
  SUBSTRING(<string1>, 1 FOR <start> - 1) || <string2> ||
  SUBSTRING(<string1>, <start> + <length>)
```

1. La prima posizione in una stringa è 1, e non 0.
2. Se gli argomenti <start> e/o <length> non sono interi, si applica l'arrotondamento bancario.
3. Se non si specifica <length>, si intende CHAR_LENGTH( <string2> ).

| | | |
|---|---|---|
| *PI* | PI() | Non ha argomenti e riporta la costante # (3.1459...). |

```
  val = PI();
```

| | | |
|---|---|---|
| *POSITION* | Vedi sotto | Da' la posizione iniziale della prima stringa all'interno della seconda a partire dall'inizio. Nella seconda forma si può dare la posizione dalla quale far iniziare la ricerca, per cui i caratteri precedenti vengono ignorati. |

```
  POSITION(  <string> IN <string>  )
  POSITION( <string>, <string> [, <posizione-iniziale>] )
```

```
  select rdb$relation_name
    from rdb$relations
    where position('RDB$' IN rdb$relation_name) = 1;
/* */
SELECT position ( 'ser', 'Essere o non essere', 10 )
from rdb$database; -- riporta 16
```

Il secondo esempio riporta 16 perchè la prima occasione in cui la sottostringa 'ser' compare è precedente al decimo carattere e viene quindi ignorata.

| | | |
|---|---|---|
| *POWER* | POWER( <number>, <number> ) | POWER(X, Y) riporta X elevato alla potenza Y. |

```
  select power(x, 10) from y;
```

| | | |
|---|---|---|
| *RAND* | RAND() | Non ha argomenti e riporta un numero casuale tra 0 and 1. |

| Funzione | Formato | Descrizione |
|---|---|---|
| `select * from x order by rand();` | | |
| *REPLACE* | REPLACE( <stringtosearch>, <findstring>, <replstring> ) | Sostituisce <findstring> con <replstring> tutte le volte in cui compare in <stringtosearch>. |
| `select replace(x, ' ', ',') from y;` | | |
| *REVERSE* | REVERSE( <value> ) | Rovescia i caratteri della stringa. Utile per creare espressioni indice per stringhe da destra a sinistra. |
| `create index people_email on people`<br>`  computed by (reverse(email));`<br>`select * from people`<br>`  where reverse(email) starting with reverse('.br');` | | |
| *RIGHT* | RIGHT( <string>, <number> ) | Riporta il numero specificato di caratteri della parte destra di una stringa data. |
| `select right(rdb$relation_name, char_length(rdb$relation_name) - 4)`<br>`  from rdb$relations`<br>`  where rdb$relation_name like 'RDB$%';` | | |
| *ROUND* | ROUND( <number>, [<number>] ) | Riporta il numero arrotondato alla scala specificata. |
| Esempio<br>`select round(salary * 1.1, 0) from people;`<br>Se la scala, cioè il secondo parametro, è negativo o omesso, si arrotonda la parte intera.<br>Ad esempio, ROUND(123.456, -1) riporta 120.000. | | |
| *RPAD* | RPAD( <string1>, <length> [, <string2> ] ) | Aggiunge <string2> alla fine di <string1> finchè la lunghezza della stringa risultante raggiunge <length>. |
| Esempio<br>`select rpad(x, 10) from y;`<br>1. Se manca la seconda stringa si prende uno spazio.<br>2. Se la stringa risultante supera la lunghezza data, si tronca l'ultima ripetizione di <string2>. | | |

| Funzione | Formato | Descrizione |
|---|---|---|
| *SIGN* | SIGN( <number> ) | Riporta 1, 0, o -1 in funzione del segno del parametro, cioè a seconda che sia rispettivamente positivo, zero o negativo. |
| `select sign(x) from y;` | | |
| *SIN* | SIN( <number> ) | Riporta il seno di un angolo espresso in radianti. |
| `select sin(x) from y;` | | |
| *SINH* | SINH( <number> ) | Riporta il seno iperbolico di un numero. |
| `select sinh(x) from y;` | | |
| *SQRT* | SQRT( <number> ) | Riporta la radice quadrata di un numero. |
| `select sqrt(x) from y;` | | |
| *TAN* | TAN( <number> ) | Riporta la tangente di un angolo espresso in radianti. |
| `select tan(x) from y;` | | |
| *TANH* | TANH( <number> ) | Riporta la tangente iperbolica di un numero. |
| `select tanh(x) from y;` | | |
| *TRIM* | Vedi sotto | **(V.2.0.x)** Elimina i caratteri terminali (per default gli spazi) dalla fine sinistra e/o destra di una stringa secondo quanto specificato. |

```
TRIM <left paren> [ [ <trim specification> ] [ <trim character> ]
  FROM ] <value expression> <right paren>

   <trim specification> ::=  LEADING  | TRAILING  | BOTH

   <trim character> ::= <value expression>
```

**Regole**

1. Se non si specifica <trim specification>, si assume BOTH.
2. Se manca il <trim character>, si assume ' '.
3. Se sono specificati <trim specification> e/o <trim character>, deve esserci anche FROM.

| Funzione | Formato | Descrizione |
|---|---|---|
| 4. Se mancano sia <trim specification> che <trim character>, FROM non va aggiunto.<br><br>Esempio A)<br><br>`select`<br>`  rdb$relation_name,`<br>`  trim(leading 'RDB$' from rdb$relation_name)`<br>`from rdb$relations`<br>`  where rdb$relation_name starting with 'RDB$';`<br><br>Esempio B)<br><br>`select`<br>`  trim(rdb$relation_name) || ' is a system table'`<br>`from rdb$relations`<br>`  where rdb$system_flag = 1;` | | |
| *TRUNC* | TRUNC( <number> [, <number> ] ) | Riporta la parte intera di un numero fino ad una scala specificata. |
| `1) select trunc(x) from y;`<br>`2) select trunc(-2.8), trunc(2.8)`<br>`  from rdb$database;  -- riporta -2, 2`<br>`3) select trunc(987.65, 1), trunc(987.65, -1)`<br>`  from rdb$database;  -- riporta 987.60, 980.00` | | |

# Appendice B: Set di caratteri internazionali

A. dos Santos Fernandes & Others

## Nuovi set di caratteri ed ordinamenti implementati

Nelle versioni di Firebird 2 sono stati implementati i seguenti set di caratteri e/o ordinamenti:

| Set di caratteri | Ordinamento | Descrizione | Implementato da |
|---|---|---|---|
| ISO8859_1 | ES_ES_CI_AI | Ordinamento per lo spagnolo: ignora maiuscole ed accenti. Per il set di caratteri ISO8859_1. | A. dos Santos Fernandes |
| " | PT_BR | Ordinamento per Portoghese Brasiliano per il set di caratteri ISO8859_1. | A. dos Santos Fernandes, P. H. Albanez |
| ISO8859_2 | ISO_PLK | Ordinamento polacco per il set di caratteri ISO8859_2. | J. Glowacki, A. dos Santos Fernandes |
| KOI8R | KOI8_RU | Set ed ordinamento standard per il Russo. | O. Loa, A. Karyakin |
| KOI8U | KOI8U_UA | Set ed ordinamento standard per l'Ucraino. | O. Loa, A. Karyakin |
| WIN1250 | BS_BA | Ordinamento per il Bosniaco per il set di caratteri WIN1250. | F. Hasovic |
| " | WIN_CZ_AI | Ordinamento per la lingua Ceca: ignora gli accenti. Per il set di caratteri WIN1250. | I. Prenosil, A. dos Santos Fernandes |
| " | WIN_CZ_CI_AI | Ordinamento per la lingua Ceca: ignora gli accenti e le maiuscole. Per il set di caratteri WIN1250. | I. Prenosil, A. dos Santos Fernandes |
| WIN1252 | WIN_PTBR | Ordinamento Portoghese Brasiliano per il set di caratteri WIN1252. | A. dos Santos Fernandes, P. H. Albanez |
| WIN1257 | WIN1257_LV | Ordinamento Lettone. | O. Loa, A. Karyakin |
| " | WIN1257_LT | Ordinamento Lituano. | O. Loa, A. Karyakin |

| Set di caratteri | Ordinamento | Descrizione | Implementato da |
|---|---|---|---|
| " | WIN1257_EE | Ordinamento Estone. | O. Loa, A. Karyakin |
| WIN1258 | (n/a) | Sottoinsieme del set di caratteri Vietnamita WIN1258. | Nguyen The Phuong, A. dos Santos Fernandes |
| UTF8 | UCS_BASIC | Supporto di Unicode 4.0 col set di caratteri UTF8 ed ordinamento UCS_BASIC. | A. dos Santos Fernandes |
| " | UNICODE | Supporto di Unicode 4.0 col set di caratteri UTF8 ed ordinamento UNICODE. | A. dos Santos Fernandes |
| " | UNICODE_CI | Supporto per Unicode 4.0 con il set di caratteri UTF8 ed ordinamento indifferente alle maiuscole/minuscole. | A. dos Santos Fernandes |
| ISO8859_1 | FR_FR_CI_AI | (V.2.1) Ordinamento per lalingua francese. Ignora gli accenti e le maiuscole. | A. dos Santos Fernandes |
| CP943C | CP943C_UNICODE | (V.2.1) Set di caratteri giapponese. | A. dos Santos Fernandes |
| TIS620 | TIS620_UNICODE | (V.2.1) Set di caratteri tailandese, single byte. | (ignoto) |

## Insiemi di caratteri a singolo byte

```
CYRL,
DOS437, DOS737, DOS775, DOS850, DOS852, DOS857, DOS858, DOS860,
DOS861, DOS862, DOS863, DOS864, DOS865, DOS866, DOS869,
ISO8859_1, ISO8859_13, ISO8859_2, ISO8859_3, ISO8859_4,
ISO8859_5, ISO8859_6, ISO8859_7, ISO8859_8, ISO8859_9,
KOI8R, KOI8U,
NEXT,
TIS620,
WIN1250, WIN1251, WIN1252, WIN1253, WIN1254, WIN1255, WIN1256,
WIN1257 and WIN1258.
```

## Insiemi di caratteri ICU

```
UTF-8 ibm-1208 ibm-1209 ibm-5304 ibm-5305 windows-65001 cp1208
UTF-16 ISO-10646-UCS-2 unicode csUnicode ucs-2
UTF-16BE x-utf-16be ibm-1200 ibm-1201 ibm-5297 ibm-13488
  ibm-17584 windows-1201 cp1200 cp1201 UTF16_BigEndian
UTF-16LE x-utf-16le ibm-1202 ibm-13490 ibm-17586
  UTF16_LittleEndian windows-1200
UTF-32 ISO-10646-UCS-4 csUCS4 ucs-4
UTF-32BE UTF32_BigEndian ibm-1232 ibm-1233
```

```
UTF-32LE UTF32_LittleEndian ibm-1234
UTF16_PlatformEndian
UTF16_OppositeEndian
UTF32_PlatformEndian
UTF32_OppositeEndian
UTF-7 windows-65000
IMAP-mailbox-name
SCSU
BOCU-1 csBOCU-1
CESU-8
ISO-8859-1 ibm-819 IBM819 cp819 latin1 8859_1 csISOLatin1
  iso-ir-100 ISO_8859-1:1987 l1 819
US-ASCII ASCII ANSI_X3.4-1968 ANSI_X3.4-1986 ISO_646.irv:1991
  iso_646.irv:1983 ISO646-US us csASCII iso-ir-6 cp367 ascii7
  646 windows-20127
ISO_2022,locale=ja,version=0 ISO-2022-JP csISO2022JP
ISO_2022,locale=ja,version=1 ISO-2022-JP-1 JIS JIS_Encoding
ISO_2022,locale=ja,version=2 ISO-2022-JP-2 csISO2022JP2
ISO_2022,locale=ja,version=3 JIS7 csJISEncoding
ISO_2022,locale=ja,version=4 JIS8
ISO_2022,locale=ko,version=0 ISO-2022-KR csISO2022KR
ISO_2022,locale=ko,version=1 ibm-25546
ISO_2022,locale=zh,version=0 ISO-2022-CN
ISO_2022,locale=zh,version=1 ISO-2022-CN-EXT
HZ HZ-GB-2312
ISCII,version=0 x-iscii-de windows-57002 iscii-dev
ISCII,version=1 x-iscii-be windows-57003 iscii-bng windows-57006
  x-iscii-as
ISCII,version=2 x-iscii-pa windows-57011 iscii-gur
ISCII,version=3 x-iscii-gu windows-57010 iscii-guj
ISCII,version=4 x-iscii-or windows-57007 iscii-ori
ISCII,version=5 x-iscii-ta windows-57004 iscii-tml
ISCII,version=6 x-iscii-te windows-57005 iscii-tlg
ISCII,version=7 x-iscii-ka windows-57008 iscii-knd
ISCII,version=8 x-iscii-ma windows-57009 iscii-mlm
gb18030 ibm-1392 windows-54936
LMBCS-1 lmbcs
LMBCS-2
LMBCS-3
LMBCS-4
LMBCS-5
LMBCS-6
LMBCS-8
LMBCS-11
LMBCS-16
LMBCS-17
LMBCS-18
LMBCS-19
ibm-367_P100-1995 ibm-367 IBM367
ibm-912_P100-1995 ibm-912 iso-8859-2 ISO_8859-2:1987 latin2
  csISOLatin2 iso-ir-101 l2 8859_2 cp912 912 windows-28592
ibm-913_P100-2000 ibm-913 iso-8859-3 ISO_8859-3:1988 latin3
  csISOLatin3 iso-ir-109 l3 8859_3 cp913 913 windows-28593
ibm-914_P100-1995 ibm-914 iso-8859-4 latin4 csISOLatin4
  iso-ir-110 ISO_8859-4:1988 l4 8859_4 cp914 914 windows-28594
ibm-915_P100-1995 ibm-915 iso-8859-5 cyrillic csISOLatinCyrillic
  iso-ir-144 ISO_8859-5:1988 8859_5 cp915 915 windows-28595
ibm-1089_P100-1995 ibm-1089 iso-8859-6 arabic csISOLatinArabic
  iso-ir-127 ISO_8859-6:1987 ECMA-114 ASMO-708 8859_6 cp1089
  1089 windows-28596 ISO-8859-6-I ISO-8859-6-E
ibm-813_P100-1995 ibm-813 iso-8859-7 greek greek8 ELOT_928
```

```
  ECMA-118 csISOLatinGreek iso-ir-126 ISO_8859-7:1987 8859_7
  cp813 813 windows-28597
ibm-916_P100-1995 ibm-916 iso-8859-8 hebrew csISOLatinHebrew
  iso-ir-138 ISO_8859-8:1988 ISO-8859-8-I ISO-8859-8-E 8859_8
  cp916 916 windows-28598
ibm-920_P100-1995 ibm-920 iso-8859-9 latin5 csISOLatin5
  iso-ir-148 ISO_8859-9:1989 l5 8859_9 cp920 920 windows-28599
  ECMA-128
ibm-921_P100-1995 ibm-921 iso-8859-13 8859_13 cp921 921
ibm-923_P100-1998 ibm-923 iso-8859-15 Latin-9 l9 8859_15 latin0
  csisolatin0 csisolatin9 iso8859_15_fdis cp923 923 windows-28605
ibm-942_P12A-1999 ibm-942 ibm-932 cp932 shift_jis78 sjis78
  ibm-942_VSUB_VPUA ibm-932_VSUB_VPUA
ibm-943_P15A-2003 ibm-943 Shift_JIS MS_Kanji csShiftJIS
  windows-31j csWindows31J x-sjis x-ms-cp932 cp932 windows-932
  cp943c IBM-943C ms932 pck sjis ibm-943_VSUB_VPUA
ibm-943_P130-1999 ibm-943 Shift_JIS cp943 943 ibm-943_VASCII_VSUB_VPUA
ibm-33722_P12A-1999 ibm-33722 ibm-5050 EUC-JP
  Extended_UNIX_Code_Packed_Format_for_Japanese
  csEUCPkdFmtJapanese X-EUC-JP eucjis windows-51932
  ibm-33722_VPUA IBM-eucJP
ibm-33722_P120-1999 ibm-33722 ibm-5050 cp33722 33722
  ibm-33722_VASCII_VPUA
ibm-954_P101-2000 ibm-954 EUC-JP
ibm-1373_P100-2002 ibm-1373 windows-950
windows-950-2000 Big5 csBig5 windows-950 x-big5
ibm-950_P110-1999 ibm-950 cp950 950
macos-2566-10.2 Big5-HKSCS big5hk HKSCS-BIG5
ibm-1375_P100-2003 ibm-1375 Big5-HKSCS
ibm-1386_P100-2002 ibm-1386 cp1386 windows-936 ibm-1386_VSUB_VPUA
windows-936-2000 GBK CP936 MS936 windows-936
ibm-1383_P110-1999 ibm-1383 GB2312 csGB2312 EUC-CN ibm-eucCN
  hp15CN cp1383 1383 ibm-1383_VPUA
ibm-5478_P100-1995 ibm-5478 GB_2312-80 chinese iso-ir-58
  csISO58GB231280 gb2312-1980 GB2312.1980-0
ibm-964_P110-1999 ibm-964 EUC-TW ibm-eucTW cns11643 cp964 964
  ibm-964_VPUA
ibm-949_P110-1999 ibm-949 cp949 949 ibm-949_VASCII_VSUB_VPUA
ibm-949_P11A-1999 ibm-949 cp949c ibm-949_VSUB_VPUA
ibm-970_P110-1995 ibm-970 EUC-KR KS_C_5601-1987 windows-51949
  csEUCKR ibm-eucKR KSC_5601 5601 ibm-970_VPUA
ibm-971_P100-1995 ibm-971 ibm-971_VPUA
ibm-1363_P11B-1998 ibm-1363 KS_C_5601-1987 KS_C_5601-1989 KSC_5601
  csKSC56011987 korean iso-ir-149 5601 cp1363 ksc windows-949
  ibm-1363_VSUB_VPUA
ibm-1363_P110-1997 ibm-1363 ibm-1363_VASCII_VSUB_VPUA
windows-949-2000 windows-949 KS_C_5601-1987 KS_C_5601-1989
  KSC_5601 csKSC56011987 korean iso-ir-149 ms949
ibm-1162_P100-1999 ibm-1162
ibm-874_P100-1995 ibm-874 ibm-9066 cp874 TIS-620 tis620.2533
  eucTH cp9066
windows-874-2000 TIS-620 windows-874 MS874
ibm-437_P100-1995 ibm-437 IBM437 cp437 437 csPC8CodePage437
  windows-437
ibm-850_P100-1995 ibm-850 IBM850 cp850 850 csPC850Multilingual
  windows-850
ibm-851_P100-1995 ibm-851 IBM851 cp851 851 csPC851
ibm-852_P100-1995 ibm-852 IBM852 cp852 852 csPCp852 windows-852
ibm-855_P100-1995 ibm-855 IBM855 cp855 855 csIBM855 csPCp855
ibm-856_P100-1995 ibm-856 cp856 856
ibm-857_P100-1995 ibm-857 IBM857 cp857 857 csIBM857 windows-857
```

```
ibm-858_P100-1997 ibm-858 IBM00858 CCSID00858 CP00858
  PC-Multilingual-850+euro cp858
ibm-860_P100-1995 ibm-860 IBM860 cp860 860 csIBM860
ibm-861_P100-1995 ibm-861 IBM861 cp861 861 cp-is csIBM861
  windows-861
ibm-862_P100-1995 ibm-862 IBM862 cp862 862 csPC862LatinHebrew
  DOS-862 windows-862
ibm-863_P100-1995 ibm-863 IBM863 cp863 863 csIBM863
ibm-864_X110-1999 ibm-864 IBM864 cp864 csIBM864
ibm-865_P100-1995 ibm-865 IBM865 cp865 865 csIBM865
ibm-866_P100-1995 ibm-866 IBM866 cp866 866 csIBM866 windows-866
ibm-867_P100-1998 ibm-867 cp867
ibm-868_P100-1995 ibm-868 IBM868 CP868 868 csIBM868 cp-ar
ibm-869_P100-1995 ibm-869 IBM869 cp869 869 cp-gr csIBM869
  windows-869
ibm-878_P100-1996 ibm-878 KOI8-R koi8 csKOI8R cp878
ibm-901_P100-1999 ibm-901
ibm-902_P100-1999 ibm-902
ibm-922_P100-1999 ibm-922 cp922 922
ibm-4909_P100-1999 ibm-4909
ibm-5346_P100-1998 ibm-5346 windows-1250 cp1250
ibm-5347_P100-1998 ibm-5347 windows-1251 cp1251
ibm-5348_P100-1997 ibm-5348 windows-1252 cp1252
ibm-5349_P100-1998 ibm-5349 windows-1253 cp1253
ibm-5350_P100-1998 ibm-5350 windows-1254 cp1254
ibm-9447_P100-2002 ibm-9447 windows-1255 cp1255
windows-1256-2000 windows-1256 cp1256
ibm-9449_P100-2002 ibm-9449 windows-1257 cp1257
ibm-5354_P100-1998 ibm-5354 windows-1258 cp1258
ibm-1250_P100-1995 ibm-1250 windows-1250
ibm-1251_P100-1995 ibm-1251 windows-1251
ibm-1252_P100-2000 ibm-1252 windows-1252
ibm-1253_P100-1995 ibm-1253 windows-1253
ibm-1254_P100-1995 ibm-1254 windows-1254
ibm-1255_P100-1995 ibm-1255
ibm-5351_P100-1998 ibm-5351 windows-1255
ibm-1256_P110-1997 ibm-1256
ibm-5352_P100-1998 ibm-5352 windows-1256
ibm-1257_P100-1995 ibm-1257
ibm-5353_P100-1998 ibm-5353 windows-1257
ibm-1258_P100-1997 ibm-1258 windows-1258
macos-0_2-10.2 macintosh mac csMacintosh windows-10000
macos-6-10.2 x-mac-greek windows-10006 macgr
macos-7_3-10.2 x-mac-cyrillic windows-10007 maccy
macos-29-10.2 x-mac-centraleurroman windows-10029 x-mac-ce macce
macos-35-10.2 x-mac-turkish windows-10081 mactr
ibm-1051_P100-1995 ibm-1051 hp-roman8 roman8 r8 csHPRoman8
ibm-1276_P100-1995 ibm-1276 Adobe-Standard-Encoding
  csAdobeStandardEncoding
ibm-1277_P100-1995 ibm-1277 Adobe-Latin1-Encoding
ibm-1006_P100-1995 ibm-1006 cp1006 1006
ibm-1098_P100-1995 ibm-1098 cp1098 1098
ibm-1124_P100-1996 ibm-1124 cp1124 1124
ibm-1125_P100-1997 ibm-1125 cp1125
ibm-1129_P100-1997 ibm-1129
ibm-1131_P100-1997 ibm-1131 cp1131
ibm-1133_P100-1997 ibm-1133
ibm-1381_P110-1999 ibm-1381 cp1381 1381
ibm-37_P100-1995 ibm-37 IBM037 ibm-037 ebcdic-cp-us ebcdic-cp-ca
  ebcdic-cp-wt ebcdic-cp-nl csIBM037 cp037 037 cpibm37 cp37
ibm-273_P100-1995 ibm-273 IBM273 CP273 csIBM273 ebcdic-de cpibm273
```

```
  273
ibm-277_P100-1995 ibm-277 IBM277 cp277 EBCDIC-CP-DK EBCDIC-CP-NO
  csIBM277 ebcdic-dk cpibm277 277
ibm-278_P100-1995 ibm-278 IBM278 cp278 ebcdic-cp-fi ebcdic-cp-se
  csIBM278 ebcdic-sv cpibm278 278
ibm-280_P100-1995 ibm-280 IBM280 CP280 ebcdic-cp-it csIBM280
  cpibm280 280
ibm-284_P100-1995 ibm-284 IBM284 CP284 ebcdic-cp-es csIBM284
  cpibm284 284
ibm-285_P100-1995 ibm-285 IBM285 CP285 ebcdic-cp-gb csIBM285
  ebcdic-gb cpibm285 285
ibm-290_P100-1995 ibm-290 IBM290 cp290 EBCDIC-JP-kana csIBM290
ibm-297_P100-1995 ibm-297 IBM297 cp297 ebcdic-cp-fr csIBM297
  cpibm297 297
ibm-420_X120-1999 ibm-420 IBM420 cp420 ebcdic-cp-ar1 csIBM420 420
ibm-424_P100-1995 ibm-424 IBM424 cp424 ebcdic-cp-he csIBM424 424
ibm-500_P100-1995 ibm-500 IBM500 CP500 ebcdic-cp-be csIBM500
  ebcdic-cp-ch cpibm500 500
ibm-803_P100-1999 ibm-803 cp803
ibm-838_P100-1995 ibm-838 IBM-Thai csIBMThai cp838 838 ibm-9030
ibm-870_P100-1995 ibm-870 IBM870 CP870 ebcdic-cp-roece
  ebcdic-cp-yu csIBM870
ibm-871_P100-1995 ibm-871 IBM871 ebcdic-cp-is csIBM871 CP871
  ebcdic-is cpibm871 871
ibm-875_P100-1995 ibm-875 IBM875 cp875 875
ibm-918_P100-1995 ibm-918 IBM918 CP918 ebcdic-cp-ar2 csIBM918
ibm-930_P120-1999 ibm-930 ibm-5026 cp930 cpibm930 930
ibm-933_P110-1995 ibm-933 cp933 cpibm933 933
ibm-935_P110-1999 ibm-935 cp935 cpibm935 935
ibm-937_P110-1999 ibm-937 cp937 cpibm937 937
ibm-939_P120-1999 ibm-939 ibm-931 ibm-5035 cp939 939
ibm-1025_P100-1995 ibm-1025 cp1025 1025
ibm-1026_P100-1995 ibm-1026 IBM1026 CP1026 csIBM1026 1026
ibm-1047_P100-1995 ibm-1047 IBM1047 cpibm1047
ibm-1097_P100-1995 ibm-1097 cp1097 1097
ibm-1112_P100-1995 ibm-1112 cp1112 1112
ibm-1122_P100-1999 ibm-1122 cp1122 1122
ibm-1123_P100-1995 ibm-1123 cp1123 1123 cpibm1123
ibm-1130_P100-1997 ibm-1130
ibm-1132_P100-1998 ibm-1132
ibm-1140_P100-1997 ibm-1140 IBM01140 CCSID01140 CP01140 cp1140
  cpibm1140 ebcdic-us-37+euro
ibm-1141_P100-1997 ibm-1141 IBM01141 CCSID01141 CP01141 cp1141
  cpibm1141 ebcdic-de-273+euro
ibm-1142_P100-1997 ibm-1142 IBM01142 CCSID01142 CP01142 cp1142
  cpibm1142 ebcdic-dk-277+euro ebcdic-no-277+euro
ibm-1143_P100-1997 ibm-1143 IBM01143 CCSID01143 CP01143 cp1143
  cpibm1143 ebcdic-fi-278+euro ebcdic-se-278+euro
ibm-1144_P100-1997 ibm-1144 IBM01144 CCSID01144 CP01144 cp1144
  cpibm1144 ebcdic-it-280+euro
ibm-1145_P100-1997 ibm-1145 IBM01145 CCSID01145 CP01145 cp1145
  cpibm1145 ebcdic-es-284+euro
ibm-1146_P100-1997 ibm-1146 IBM01146 CCSID01146 CP01146 cp1146
  cpibm1146 ebcdic-gb-285+euro
ibm-1147_P100-1997 ibm-1147 IBM01147 CCSID01147 CP01147 cp1147
  cpibm1147 ebcdic-fr-297+euro
ibm-1148_P100-1997 ibm-1148 IBM01148 CCSID01148 CP01148 cp1148
  cpibm1148 ebcdic-international-500+euro
ibm-1149_P100-1997 ibm-1149 IBM01149 CCSID01149 CP01149 cp1149
  cpibm1149 ebcdic-is-871+euro
ibm-1153_P100-1999 ibm-1153 cpibm1153
```

```
ibm-1154_P100-1999 ibm-1154 cpibm1154
ibm-1155_P100-1999 ibm-1155 cpibm1155
ibm-1156_P100-1999 ibm-1156 cpibm1156
ibm-1157_P100-1999 ibm-1157 cpibm1157
ibm-1158_P100-1999 ibm-1158 cpibm1158
ibm-1160_P100-1999 ibm-1160 cpibm1160
ibm-1164_P100-1999 ibm-1164 cpibm1164
ibm-1364_P110-1997 ibm-1364 cp1364
ibm-1371_P100-1999 ibm-1371 cpibm1371
ibm-1388_P103-2001 ibm-1388 ibm-9580
ibm-1390_P110-2003 ibm-1390 cpibm1390
ibm-1399_P110-2003 ibm-1399
ibm-16684_P110-2003 ibm-16684
ibm-4899_P100-1998 ibm-4899 cpibm4899
ibm-4971_P100-1999 ibm-4971 cpibm4971
ibm-12712_P100-1998 ibm-12712 cpibm12712 ebcdic-he
ibm-16804_X110-1999 ibm-16804 cpibm16804 ebcdic-ar
ibm-1137_P100-1999 ibm-1137
ibm-5123_P100-1999 ibm-5123
ibm-8482_P100-1999 ibm-8482
ibm-37_P100-1995,swaplfnl ibm-37-s390 ibm037-s390
ibm-1047_P100-1995,swaplfnl ibm-1047-s390
ibm-1140_P100-1997,swaplfnl ibm-1140-s390
ibm-1142_P100-1997,swaplfnl ibm-1142-s390
ibm-1143_P100-1997,swaplfnl ibm-1143-s390
ibm-1144_P100-1997,swaplfnl ibm-1144-s390
ibm-1145_P100-1997,swaplfnl ibm-1145-s390
ibm-1146_P100-1997,swaplfnl ibm-1146-s390
ibm-1147_P100-1997,swaplfnl ibm-1147-s390
ibm-1148_P100-1997,swaplfnl ibm-1148-s390
ibm-1149_P100-1997,swaplfnl ibm-1149-s390
ibm-1153_P100-1999,swaplfnl ibm-1153-s390
ibm-12712_P100-1998,swaplfnl ibm-12712-s390
ibm-16804_X110-1999,swaplfnl ibm-16804-s390
ebcdic-xml-us
```

# Appendice C: Aggiornamento del Database della Sicurezza per Firebird 2

A. Peshkov

## Script di aggiornamento del DB di sicurezza

```
/* Script security_database.sql
 *
 *  The contents of this file are subject to the Initial
 *  Developer's Public License Version 1.0 (the "License");
 *  you may not use this file except in compliance with the
 *  License. You may obtain a copy of the License at
 *  http://www.ibphoenix.com/main.nfs?a=ibphoenix&page=ibp_idpl.
 *
 *  Software distributed under the License is distributed AS IS,
 *  WITHOUT WARRANTY OF ANY KIND, either express or implied.
 *  See the License for the specific language governing rights
 *  and limitations under the License.
 *
 *  The Original Code was created by Alex Peshkov on 16-Nov-2004
 *  for the Firebird Open Source RDBMS project.
 *
 *  Copyright (c) 2004 Alex Peshkov
 *  and all contributors signed below.
 *
 *  All Rights Reserved.
 *  Contributor(s): ______________________________________.
 *
 */

-- 1. temporary table to alter domains correctly.
CREATE TABLE UTMP (
  USER_NAME VARCHAR(128) CHARACTER SET ASCII,
  SYS_USER_NAME VARCHAR(128) CHARACTER SET ASCII,
  GROUP_NAME VARCHAR(128) CHARACTER SET ASCII,
  UID   INTEGER,
  GID   INTEGER,
  PASSWD   VARCHAR(64) CHARACTER SET BINARY,
  PRIVILEGE  INTEGER,
  COMMENT  BLOB SUB_TYPE TEXT SEGMENT SIZE 80
    CHARACTER SET UNICODE_FSS,
  FIRST_NAME  VARCHAR(32) CHARACTER SET UNICODE_FSS
    DEFAULT _UNICODE_FSS '',
  MIDDLE_NAME VARCHAR(32) CHARACTER SET UNICODE_FSS
    DEFAULT _UNICODE_FSS '',
  LAST_NAME VARCHAR(32) CHARACTER SET UNICODE_FSS
    DEFAULT _UNICODE_FSS ''
  );
```

```
COMMIT;

-- 2. save users data
INSERT INTO UTMP(USER_NAME, SYS_USER_NAME, GROUP_NAME,
  UID, GID, PRIVILEGE, COMMENT, FIRST_NAME, MIDDLE_NAME,
  LAST_NAME, PASSWD)
SELECT USER_NAME, SYS_USER_NAME, GROUP_NAME,
  UID, GID, PRIVILEGE, COMMENT, FIRST_NAME, MIDDLE_NAME,
  LAST_NAME, PASSWD
 FROM USERS;
COMMIT;

-- 3. drop old tables and domains
DROP TABLE USERS;
DROP TABLE HOST_INFO;
COMMIT;

DROP DOMAIN COMMENT;
DROP DOMAIN NAME_PART;
DROP DOMAIN GID;
DROP DOMAIN HOST_KEY;
DROP DOMAIN HOST_NAME;
DROP DOMAIN PASSWD;
DROP DOMAIN UID;
DROP DOMAIN USER_NAME;
DROP DOMAIN PRIVILEGE;
COMMIT;

-- 4. create new objects in database
CREATE DOMAIN RDB$COMMENT AS BLOB SUB_TYPE TEXT SEGMENT SIZE 80
  CHARACTER SET UNICODE_FSS;
CREATE DOMAIN RDB$NAME_PART AS VARCHAR(32)
  CHARACTER SET UNICODE_FSS DEFAULT _UNICODE_FSS '';
CREATE DOMAIN RDB$GID AS INTEGER;
CREATE DOMAIN RDB$PASSWD AS VARCHAR(64) CHARACTER SET BINARY;
CREATE DOMAIN RDB$UID AS INTEGER;
CREATE DOMAIN RDB$USER_NAME AS VARCHAR(128)
CHARACTER SET UNICODE_FSS;
CREATE DOMAIN RDB$USER_PRIVILEGE AS INTEGER;
COMMIT;


CREATE TABLE RDB$USERS (
  RDB$USER_NAME       RDB$USER_NAME NOT NULL PRIMARY KEY,
  /* local system user name
     for setuid for file permissions */
  RDB$SYS_USER_NAME   RDB$USER_NAME,
  RDB$GROUP_NAME      RDB$USER_NAME,
  RDB$UID             RDB$UID,
  RDB$GID             RDB$GID,
  RDB$PASSWD          RDB$PASSWD,  /* SEE NOTE BELOW */

  /* Privilege level of user -
     mark a user as having DBA privilege */
  RDB$PRIVILEGE       RDB$USER_PRIVILEGE,

  RDB$COMMENT         RDB$COMMENT,
  RDB$FIRST_NAME      RDB$NAME_PART,
  RDB$MIDDLE_NAME     RDB$NAME_PART,
  RDB$LAST_NAME       RDB$NAME_PART);
COMMIT;
```

```
CREATE VIEW USERS (USER_NAME, SYS_USER_NAME, GROUP_NAME,
  UID, GID, PASSWD, PRIVILEGE, COMMENT, FIRST_NAME,
  MIDDLE_NAME, LAST_NAME, FULL_NAME) AS

  SELECT RDB$USER_NAME, RDB$SYS_USER_NAME, RDB$GROUP_NAME,
    RDB$UID, RDB$GID, RDB$PASSWD, RDB$PRIVILEGE, RDB$COMMENT,
    RDB$FIRST_NAME, RDB$MIDDLE_NAME, RDB$LAST_NAME,
    RDB$first_name || _UNICODE_FSS ' ' || RDB$middle_name
      || _UNICODE_FSS ' ' || RDB$last_name
    FROM RDB$USERS
    WHERE CURRENT_USER = 'SYSDBA'
       OR CURRENT_USER = RDB$USERS.RDB$USER_NAME;
COMMIT;

GRANT ALL ON RDB$USERS to VIEW USERS;
GRANT SELECT ON USERS to PUBLIC;
GRANT UPDATE(PASSWD, GROUP_NAME, UID, GID, FIRST_NAME,
  MIDDLE_NAME, LAST_NAME)
 ON USERS TO PUBLIC;
COMMIT;

-- 5. move data from temporary table and drop it
INSERT INTO RDB$USERS(RDB$USER_NAME, RDB$SYS_USER_NAME,
  RDB$GROUP_NAME, RDB$UID, RDB$GID, RDB$PRIVILEGE, RDB$COMMENT,
  RDB$FIRST_NAME, RDB$MIDDLE_NAME, RDB$LAST_NAME, RDB$PASSWD)
SELECT USER_NAME, SYS_USER_NAME, GROUP_NAME, UID, GID,
  PRIVILEGE, COMMENT, FIRST_NAME, MIDDLE_NAME, LAST_NAME,
  PASSWD
    FROM UTMP;
COMMIT;

DROP TABLE UTMP;
COMMIT;
```

**Nota**

Questo campo andrebbe vincolato a NOT NULL. Per informazioni al riguardo si veda Nullability of RDB $PASSWD nel capitolo relativo alla sicurezza.