Firebird™ Version 1.5

Release Notes v.1.5

5. Februar 2004 – Dokumentversion 1.08

Inhaltsverzeichnis

Generelle Informationen

Die Firebird™ Datenbank Engine wurde durch ein unabhängiges Team von freiwilligen Programmierern weiterentwickelt. Grundlage dafür war der InterBase™ Sourcecode, welcher am 25 Juli 2000 von Borland unter der "InterBase Public License v.1.0" veröffentlicht wurde.

Die Entwicklung des Firebird 2 Sourcecodes begann mit der Portierung des Firebird 1 C-Codes zu C++ und der ersten grösseren Codebereinigung im frühen Stadium der "Firebird 1"-Entwicklung. Firebird 1.5 ist das erste Release des Firebird 2 Sourcecodes. Dieses Release gilt als wichtiger Meilenstein für die Entwickler und das gesamte Firebird Projekt, jedoch wird die Version ständig weiterentwickelt. Während Firebird 1.5 veröffentlicht und freigegeben wird, werden weitere Codeanpassungen bis zum nächsten Release (Firebird 2) nachgeführt.

Firebird 1.0.x wird weiterhin mit wichtigen Bug-Fixes und Erweiterungen, welche von 1.5 übernommen werden, gepflegt.

Die Firebird 1.5 Binaries

Die Firebird Binaries können über die Firebird Website – http://sourceforge.net/project/showfiles.php?group_id=9028 heruntergeladen werden.

Versionsbezeichnungen von Firebird 1.5 Releases

Win32: "WI-V1.5.0.nnnn Firebird 1.5"
Linux: "LI-V1.5.0.nnnn Firebird 1.5"
und so weiter, wobei nnnn die Build-Number darstellt.

Bitte beachten Sie den Abschnitt Dokumentation, in dem auf weitere empfohlene Dokumentationen verwiesen wird.

Neue Features

Neuer Codebase, bessere Optimierung

Dieses Release ist mit dem Code, welcher von C nach C++ portiert wurde, erstellt worden. Mike Nordell begann im Jahr 2000 mit umfangreichen Bereinigungen und Fehlerbehebungen, gefolgt von neuem Speichermanagement und Spracherweiterungen. Nicht zu vergessen sind die Erweiterungen und Anpassungen des SQL Query Optimizers, welche Arno Brinkman und andere während des V1.5 Entwicklungsprozesses vorgenommen haben. Das Resultat sind Geschwindigkeitsverbesserungen von 30 - 60 Prozent und mehr.

Architektur

Zwei wesentliche Ergänzungen für Windows Plattformen sind der Classic Server und der Embedded Server.

- Während fast acht Jahren gab es keinen Classic Server auf Windows. Dieser unterstützt, im Gegensatz zum Windows Superserver, mehrere Prozessoren. Obwohl bereits einsetzbar, sollte der Classic Server als experimentell betrachtet werden.

- Der Embedded Server ist eine dll welche eine einzelne Clientanwendung mit dem Firebird Superserver zu einer sehr schnellen und effizienten Standalone-Anwendung verbindet.

Einige wichtige Spracherweiterungen sind seit Version 1.0.x hinzugefügt worden, einschließlich den SQL-92 Entscheidungsfunktionen CASE, COALESCE und NULLIF. Für die Syntax dieser und anderer neuer Sprachimplementationen beziehen Sie sich auf den Abschnitt Spracherweiterungen, weiter unten in diesem Dokument.

Installierte Module und Sicherheit

Wenn Sie bis jetzt Firebird 1.0.x genutzt haben, werden Sie die erheblichen Änderungen in der Namensbenennung der Module und den Zugriffsregeln bemerken. Die folgenden Anmerkungen beschreiben die wichtigsten Änderungen. Für detailliertere Informationen über die Installation, Verzeichnisstruktur und Konfiguration, beachten Sie die entsprechenden Kapitel in diesem Dokument.

1. Die meisten Module und Konstanten sind umbenannt worden. Die neuen Namen enthalten vorwiegend Bezeichnungen wie "firebird" oder "fb". Zum Beispiel ist die API Bibliothek neu zu "fbclient.dll" auf Windows und "libfbclient.so" auf anderen Plattformen geändert worden. Die Ausnahme von der Regel bildet die Sicherheitsdatenbank, welche von "isc4.gdb" nach "security.fdb" umbenannt worden ist.
2. Externe Dateien (UDF Bibliotheken, BLOB Filter, Zeichensatzbibliotheken, externe Tabellen), welche vom Server verwendet werden, unterliegen nun den Einstellungen der Betriebssystem-

sicherheit, welche sich in einigen Fällen von den Konfigurationen unter Firebird 1.0.x oder InterBase unterscheiden können.

3. Die neue Server-Konfigurationsdatei, firebird.conf, welche ibconfig (Windows) und isc_config (andere Plattformen) ersetzt, enthält neue Konfigurationsmöglichkeiten zusammen mit einer verbesserten Dokumentation und Gliederung.
4. Eine Datenbank-Aliasing-Erweiterung wurde in 1.5 integriert. Sie können jetzt optional das Datenbankverzeichnis in Form eines frei wählbaren Alias, welcher die Pfadangabe ersetzt, in Ihrem Programmcode "soft-coden". Die aktuellen Pfadangaben werden in einer Textdatei aliases.conf gespeichert. Die grundsätzliche Absicht des Aliasing ist jedoch der Schutz des tatsächlichen Datenbankpfades vor bösartigem "Sniffen" im Netzwerk.
5. Die Standardeinstellungen (und vergangene Verfahren) auf Windows Server Plattformen machten es möglich, dass der Firebird Service beim Systemstart von einem lokalen Systembenutzer installiert werden konnte. Dies stellt eine ernstzunehmende Sicherheitsschwachstelle dar, wenn der Firebird-Server gehackt werden sollte, da ein Fenster zur Verfügung gestellt wird, über welches ein Hacker den vollständigen Zugriff auf die Machine erlangen kann. Die 1.5 Version dieses Programms (instsvc.exe) erlaubt nun einen Windows-Benutzernamen für die Service-Installation zu verwenden. Es wird strengstens empfohlen, dass Sie für diesen Zweck einen Firebird-Benutzer einrichten und von dieser Erweiterung Gebrauch machen, falls der Server in irgendeiner Weise mit dem Internet verbunden ist.

Trimmen von Varchar Feldern für Remoteprotokolle

Die Arbeit an dieser Funktion wurde wieder aufgenommen und mit dieser Erweiterung für den 1.5 Client abgeschlossen. Varchar-Felder werden nun über das Netzwerk "right-trimmed" (rechtsbündige Leerzeichen werden abgeschnitten) mit der tatsächlichen Länge plus zwei Bytes übertragen.
Achtung: Da es der Client ist, welcher den Server beauftragt, Varchars zu trimmen, wird der Firebird 1.5 Client (fbclient.dll oder libfbclient.so) die Felder immer trimmen, auch wenn dieser zu einer älteren Version als der 1.5 Server Version verbunden wird. Wenn Sie eine alte Client-Bibliothek verwenden, dann werden die Felder weder in der Server Version 1.5, noch in neueren Versionen getrimmt.

Multi-Aktions-Trigger Semantik

Neu können Sie nun bedingte Insert / Update /Delete Aktionen in einem Before oder After Trigger formulieren. Dieser eine Trigger enthält dann alle DML Aktionen der entsprechenden Phase. Diese Maßnahme reduziert die Pflege und Wartung von Triggern, ohne dass die Möglichkeit von mehreren Triggern pro Phase verhindert wird.

Verbesserung von Namensbeschränkungen

Indexe, welche eine Namensintegrität erfordern, können nun mit selbst definierten Bezeichnungen versehen werden.
Achtung: Sollten Sie diese Neuerung verwenden, dann kann die Datenbank mit Firebird v1.0.x oder InterBase® nicht mehr verwendet werden.

Maximum der Indexe pro Tabelle erhöht

Das Maximum an Indexen, welche pro Tabelle definiert werden können, ist in Version 1.0 und in diesem Release von 64 auf 256 erhöht worden.

Pessimistisches Sperren (pessimistic locking)

Wenn Sie ab und zu eine pessimistische Sperre benötigen, dann steht Ihnen in diesem Release eine Syntax zur Verfügung, mit welcher sie eine "Lesersperre" für die an den Client zurückgegebenen Datensätzen aktivieren können. Benutzen Sie diese Funktion allerdings mit Vorsicht.

Cachen von Verbindungen zur Sicherheitsdatenbank

Verbindungen zur Sicherheitsdatenbank werden nun in SS Builds gecached. Dies bedeutet, dass die security.fdb bei der ersten Verbindung geladen wird und solange verbunden bleibt, bis alle Client-Verbindungen getrennt werden.

Verbesserte Fehlerauswertung

Wenn möglich, werden Fehlermeldungen, welche aufgrund von SQL-Fehlern auftreten, in detaillierterer Form ausgegeben. Es ist WICHTIG zu wissen, dass bizarre Meldungen auftreten, wenn Sie eine alte interbase.msg oder firebird.msg Datei benutzen.

Services API für Classic unter Linux

Die Services API ist nun eingeschränkt auch für Classic unter Linux verfügbar. Die Services verfügbar von gbak (Backup/Restore) und gfix (Datenbankvalidierung, Shutdown/Online, usw.) sind funktionsfähig. Andere (gstat, Server-Logs, usw.) wurden nicht getestet, und sind somit wahrscheinlich nicht funktionsfähig.

Änderungen in den Client-Bibliotheken

Windows-Clients

Die Client-Bibliothek heisst nun "fbclient.dll". Alle Serverprogramme (gbak, gfix, etc) verwenden nur die Client-Bibliothek fbclient.dll. Wir empfehlen Ihnen, die Verbindungen von neuen Anwendungen über fbclient.dll, ohne dass die Existenz von gds32.dll notwendig ist, herzustellen.
Aus Kompatbilitätsgründen zu bereits existierenden Applikationen ist es nun möglich, dass ein "Klon" von fbclient.dll mit dem Namen "gds32.dll", mit einem neuen Tool instclient.exe, erstellt werden kann. Nähere Details dazu finden Sie im Installationsabschnitt dieses Dokuments und in den letzten Installationshinweisen der Firebird-Distribution für Windows.

Linux-Clients

Die Superserver Client-Bibliothek heisst nun "libfbclient.so". Aus Kompatibilitätsgründen zu existierenden Applikationen, wird ein Symlink "libgds.so" installiert, welcher auf die libfbclient.so verweist. Die lokale Client-Bibliothek für integrierte Applikationsverbindungen zum Classic Server ist in libfbembed.so umbenannt worden.

Umbenannte Dateien und Module

Plattform	Modul	Firebird 1.0	Firebird 1.5	Bemerkungen
Alle	Environment variables	INTERBASE	FIREBIRD	Zeigt auf das Stammverzeichnis der Installation
		INTERBASE_LOCK	FIREBIRD_LOCK	Zeigt auf den Speicherort der Sperrdatei
		INTERBASE_MSG	FIREBIRD_MSG	Zeigt auf den Speicherort der Meldungsdatei
		INTERBASE_TMP	FIREBIRD_TMP	Zeigt auf das Verzeichnis für den Sortierspeicher
Alle	Security database	Isc4.gdb	security.fdb	
Alle	Message file	Interbase.msg	firebird.msg	
Alle	Server log file	interbase.log	firebird.log	
Alle	ODS version	10	10.1	Neue ODS (10.1). Verursacht keine Inkompatiblitäten mit vorhergehenden ODS, aber die ODS wird nicht automatisch upgegraded. Firebird 1.0 und 1.5 können beide ODS 10.0 und ODS 10.1 Datenbanken verarbeiten.

Plattform	Modul	Firebird 1.0	Firebird 1.5	Bemerkungen
				Nichtsdestotrotz ist ein Backup / Restore die empfohlene Vorgehensweise für die Migration von Datenbanken auf eine andere Serverversion.
Linux	Classic server binary	gds_inet_server	fb_inet_server	
Linux	Classic lock manager	ib_lock_mgr	fb_lock_mgr	
Linux	Superserver control	Ibmgr.bin	fbmgr.bin	
Linux	Superserver binary	ibserver	fbserver	
Linux	Configuration file	Isc_config	firebird.conf	
Linux	Client library	Libgds.so	libfbclient.so libfbembed.so	Thread-Sicherer Remote Client und lokaler TCP/IP Client für Superserver Lokaler Client (single-user, nicht-thread-sicher) für Classic
Linux	Client library symlink for compatibility	N/A	libgds.so	
Windows	Guardian	ibguard.exe	fbguard.exe	
Windows	Superserver binary	ibserver.exe	fbserver.exe	Nicht Multiprozessorfähig
Windows	Classic binary	N/A	fb_inet_server.exe	Windows lokale Verbindung nicht verfügbar. TCP/IP, NetBEUI OK. Multiprozessorfähig.
Windows	Client library	gds32.dll	fbclient.dll	Fb 1.5 Versionen von Serverprogrammen, und alle neuen Applikationen benötigen nur die fbclient.dll. Beachten Sie auch die untenstehenden Bemerkungen bzgl. der gds32.dll Kompatibilität für ältere Applikationen.
Windows	Configuration file	ibconfig	firebird.conf	
Windows	Local IPC port	InterBaseIPI	FirebirdIPI	Mit den Standard-Servereinstellungen können Sie keine lokale Verbindung herstellen, wenn Sie mit Applikationen arbeiten, welche eine alte Clientbibliothek (gds32.dll) verwenden. Wenn nötig, können Sie dem Server via firebird.conf angeben, für IPC den alten Namen zu verwenden.

Plattform	Modul	Firebird 1.0	Firebird 1.5	Bemerkungen
Windows	Default Registry key	HKLM\SOFTWARE\Borland\InterBase	HKLM\SOFTWARE\Firebird Project\Firebird Server\Instances	Der Pfad wird im "DefaultInstance" Parameter gespeichert. z.B. gibt es keinen "CurrentVersion" Schlüssel mehr, und "RootDirectory" ist durch "DefaultInstance" ersetzt worden.
Die neuen Servicenamen unter Windows lauten "Firebird Guardian – DefaultInstance" und "Firebird Server – DefaultInstance".				

Kompatibilität zu älteren Versionen

On-Disk Struktur (ODS)

Die On-Disk Struktur von Firebird 1.5 ist ODS 10.1. Dieser kleine ODS Upgrade war aufgrund folgender Punkte notwendig:
- drei neue Indexe für Systemtabellen
- kleinere Änderungen in den BLR von zwei Systemtriggern
- erweiterte Codierung des RDB$TRIGGER_TYPE.

Einige Erweiterungen, welche Änderungen der ODS voraussetzten, wurden auf die Version 2 verschoben. Währenddessen sollte es möglich sein, Firebird 1.0.x Datenbanken direkt zu migrieren. Benutzen Sie getestete Backups von Firebird 1.0.x Datenbanken bevor Sie diese nach 1.5 portieren.

InterBase™ Datenbanken

Wenn Sie planen, Firebird mit einer existierenden InterBase Datenbank zu testen und die Absicht haben, diese später wieder mit InterBase zu verwenden, dann treffen Sie bitte alle Vorkehrungen um die aktuelle Version mit der dazugehörigen gbak-Version von InterBase zu sichern. Um mit der Arbeit in Firebird 1.5 mit Ihrer Datenbank zu beginnen, verwenden Sie gbak-Version von Firebird 1.5, um Ihr Backup wiederherzustellen.
Der Operations Guide des InterBase® 6.0 Beta Dokumentationssatzes enthält die Befehlssyntax des gbak Backup und Restore Programms.

IB 7.x und vermutlich auch IB 6.5 Datenbanken können, wenn neue IB spezifische Erweiterungen genutzt werden, nach der Migration zu FB 1.5 via Backup/Restore inkorrekt arbeiten.

Dateinamen und Speicherorte

In diesem Release wurden die Namen mehrerer Dateien geändert. Der Grund dafür ist die schrittweise Ersetzung der geerbten Namen von InterBase® 6. Lesen Sie bitte den Abschnitt "Umbenannte Dateien und Module" für Erklärungen und Empfehlungen.

Gleichzeitige Nutzung von mehreren Servern

Da einige Systemobjektbezeichnungen in diesem Release geändert wurden, ist es möglich, FB 1.5 auf einem System zu nutzen, welches bereits InterBase oder Firebird 1.0.x installiert hat. Unter Windows nutzt FB 1.5 einen anderen Registry-Key. Wenn Sie die Server daingehend konfigurieren, dass diese unterschiedliche Ports verwenden, so ist es möglich, mehrere Instanzen gleichzeitig auszuführen oder FB 1.5 zu verwenden, während IB oder FB 1.0.x läuft.

Zurückgreifen (Abwärtskompatibilität) zu Firebird 1.0.x

Aufgrund einer grossen Anzahl von Bug-Fixes kann sich die Verhaltensweise von Datenbanken ändern, wenn Sie eine v.1.5 DB auf eine v.1.0.x downgraden. Vor allem, wenn Sie einen Primary, Unique und

Foreign Key als benannten Constraint erstellen, so ist der Default-Indexname mit v1.0.x nicht mehr kompatibel. Beachten Sie die zukünfitgen README-Dateien, in denen solche Themen abgehandelt werden.

Linux Kompatibilitäten

Aufgrund mehrerer Probleme mit dem GNU C++ Compiler, benötigen Firebird 1.5 Linux Versionen neuere Versionen der glibc runtimes als bisher. Bedauerlicherweise bedeutet dies, dass es sehr schwierig ist vorauszusagen, ob die 1.5 Binaries auf einer bestimmten Distribution installiert und genutzt werden können. Die folgende Tabelle soll die bekannten Kompatibilitäten aufzeigen. Sollten Sie Erfahrungen mit Firebird unter anderen Distributionen haben, dann würden wir Sie bitten, uns dies mitzuteilen.

Distribution	Level	Classic	Superserver
Red Hat	7.x	Nein	Nein
	8.0	Ja	Ja
	9.0	Ja	Ja
Mandrake	8.x	Nein	Nein
	9.0, 9.1, 9.2	Ja	Ja
SuSE	7.3	Ja - Die Packages libgcc-3.2-44.i586.rpm und libstdc++-3.2-44.i586.rpm müssen vor der Installation von Firebird 1.5 installiert werden.	Unbekannt
	8.0, 8.1	Ja	8.0 Ja (8.1 Unbekannt)

Sprach-Erweiterungen

DATENTYPEN

(1.5) Neue Native SQL Datentypen

BIGINT
SQL99-kompatibler, exakter numerischer Typ, 64-bit, vorzeichenbehaftet, mit einer Skala von 0. Nur in Dialekt 3 verfügbar.

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

METADATEN

(1.5) Erweiterungen von benannten Constraints

Dmitry Yemanov

Indexe, die automatisch von Constraints angelegt werden, können jetzt mit benutzer-definierten Namen versehen werden.

Bisher wurde, obwohl es möglich war, PRIMARY, FOREIGN KEY und UNIQUE Constraints zu benennen, zusätzlich automatisch ein vom System benannter Index angelegt, z.B. RDB$FOREIGN13, dessen Bezeichnung nicht geändert werden konnte. Dies bleibt das Standardverhalten, wenn benannte Constraints nicht verwendet werden.

Es wurden jedoch Spracherweiterungen hinzugefügt, die erlauben dass
a) ein systemerzeugter Index automatisch den gleichen Identifizierer wie der benannte Constraint, von dem er automatisch erzeugt wurde, bekommt.
b) ein Index, der automatisch von einem benannten oder unbenannten Constraint erzeugt wird und von diesem explizt einen Standard-Identifizierer zugewiesen bekommt und optional auch in absteigender Sortierung ("DESCENDING Order") angelegt werden kann.
HINWEIS: Momentan ist es nicht möglich, einen bereits zuvor existierenden Index zu verwenden.

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

Warnung: Stellen Sie sicher, dass der Fremdschlüssel und der Primärschlüssel Indizes mit der **selben Sortierreihenfolge** verwenden (DESC | ASC).

Beispiele
i) Benannter Constraint und explizit benannter Index

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

```
COMMIT;
```

Die folgende Anweisung erzeugt einen Primary Key Constraint namens PK_ATEST und einen automatisch erzeugten, absteigenden Index namens IDX_PK_ATEST:

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

ii) Alternative zu i) obenstehend:

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

iii) Dieses Statement erzeugt die Tabelle ATEST mit dem Primary Key PK_ATEST. Der automatisch erzeugte Index wird ebenso PK_ATEST benannt.

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

(1.5) Multi-Aktionen Trigger

Dmitry Yemanov

Trigger wurden erweitert, um es ihnen zu ermöglichen, multiple Operationen, bedingt, auf Datensatz-ebene durchzuführen.

Syntax

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

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

Beispiele

i)

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

ii)

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

ODS Änderung

Die Kodierung des Feldes RDB$TRIGGER_TYPE (Systemtabelle RDB$TRIGGERS) wurde erweitert, um komplexe Trigger-Aktionen zu ermöglichen. Für weitere Informationen, lesen Sie die Datei readme.universal_triggers.txt im /doc/sql.extensions Zweig des Firebird CVS Baums.
Hinweis(e):

1. Ein-Aktion-Trigger sind auf der ODS-Ebene vollkommen kompatibel mit FB 1.0.
2. Die (interne) Kodierung von RDB$TRIGGER_TYPE ist abhängig von der Reihenfolge, d.h., BEFORE INSERT OR UPDATE und BEFORE UPDATE OR INSERT werden unterschiedlich kodiert, obwohl beide die selbe Semantik haben und sich in der Ausführung exakt gleich verhalten.
3. Sowohl die OLD als auch die NEW Kontext-Variablen stehen in Multi-Aktions-Triggern zur Verfügung. Sollte die Art des Triggers den Gebrauch einer dieser Variablen unterbinden (z.B. die OLD Kontext-Variablen bei "for INSERT" Operationen), dann werden alle Felder, die diesem Kontext zugeordnet sind, mit NULL belegt. Wenn Sie einem ungültigen Kontext zugewiesen werden, wird eine Laufzeit-Exception ausgelöst.
4. Die neuen Booleschen Kontext-Variablen INSERTING/UPDATING/DELETING können benutzt werden, um die ausgeführte Operation zur Laufzeit abzufragen (siehe unten).

(1.5) RECREATE VIEW

Entspricht CREATE VIEW, wenn der View noch nicht existiert. Sollte der View bereits existieren, wird RECREATE VIEW versuchen ihn zu löschen (drop view) und ein komplett neues Objekt zu erzeugen. RECREATE VIEW wird fehlschlagen, wenn das Objekt momentan in Benutzung ist (object is in use). Verwendet die gleiche Syntax wie CREATE VIEW.

(1.5) CREATE OR ALTER {TRIGGER | PROCEDURE }

Diese Anweisung wird entweder einen neuen Trigger oder eine neue Prozedur erstellen (so diese nicht bereits existieren) oder sie ändern (so sie bereits existieren). Die CREATE OR ALTER Syntax lässt vorhandene Abhängigkeiten und Berechtigungen bestehen.

Die Syntax ist wie bei CREATE TRIGGER bzw. CREATE PROCEDURE, außer den zusätzlichen Schlüsselwörtern "OR ALTER".

(1.5) NULLs in Unique Constraints und Indizes

Dmitry Yemanov

Es ist jetzt möglich, einem UNIQUE Constraint oder einem UNIQUE Index eine Spalte zuzuweisen, die keinen NOT NULL Constraint besitzt. Dies ist konform zu SQL-99. Seien Sie vorsichtig, wenn Sie planen ihre Datenbank nach Firebird 1.0.x oder in eine InterBase Version zurückzuwandeln.

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

wobei <unique column list> eine oder mehrere Spalten ohne das NOT NULL Attribut enthalten kann, wenn <unique specification> UNIQUE ist oder ein UNIQUE INDEX Index-Name verwendet wird. Bitte beachten Sie, dass alle Spalten des Primary Keys nach wie vor mit NOT NULL definiert werden müssen.

Dieser Constraint erlaubt nur die Existenz von Datensätzen für die die Suchbedingungen (i) oder (ii), anhand der folgenden Logik, als Wahr ausgewertet werden:

i) Wenn die <unique specification> einen PRIMARY KEY spezifiziert, dann ist die Suchbedingung:

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

ii) Andernfalls, ist die <Suchbedingung>:

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

In diesem Falle kann die UNIQUE Bedingung nicht Wahr sein, wenn (SELECT UCL FROM TN) zwei Datensätze zurückgeben könnte, bei denen alle NOT NULL Segemente übereinstimmen.

Dieser Constraint erlaubt lediglich die Existenz von Datensätzen für die o.a. <Suchbedingung> als Wahr ausgewertet wird. In einem UNIQUE Index oder einem UNIQUE Constraint sind zwei Spalten erlaubt und dadurch zulässing, wenn:

a) Beide Mengen lediglich NULLS enthalten, oder
b) Es zumindest ein Wertepaar gibt, von denen eines nicht NULL ist und das andere NULL ist oder einen anderen Wert ungleich NULL hat.

Beispiele

UNIQUE constraint:

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

oder UNIQUE index:

```
CREATE TABLE t (a INTEGER, b INTEGER);
COMMIT;
CREATE UNIQUE INDEX uqx ON t(a, b);
COMMIT;
INSERT INTO t VALUES (NULL, NULL); /* ok, nulls erlaubt */
INSERT INTO t VALUES (1, 2); /* Not Null Werte */
INSERT INTO t VALUES (1, NULL); /* Kombinationen */
INSERT INTO t VALUES (NULL, NULL); /* ok, alle Paare mit NULLS sind unterschiedlich */
```

aber

```
INSERT INTO t VALUES (1, NULL); /* schlägt fehl, weil alle Nicht-NULL-Segmente übereinstimmen */
```

Dies bedeutet, ***dass der Primray Key Constraint NULLs nicht zuläßt***, während der UNIQUE Constraint und Unique Indizes eine beliebige Anzahl von NULLs zulassen.
Für mehrspaltige Ergebnismengen von (SELECT UCL FROM TN) werden die allgemeinen Regeln von NULLs angewendet, d.h. (1, NULL) ist unterschiedlich zu (NULL, 1) und jedes (NULL, NULL) wiederum ist unterschiedlich zu jedem anderen (NULL, NULL).

DSQL

(1.5) Ausdrücke und Variablen als Prozedur-Argumente

Dmitry Yemanov

Aufrufe von EXECUTE PROCEDURE ProcName(<Argument-list>) und
SELECT <Output-list> FROM ProcName(<Argument-list>)
akzeptieren jetzt lokale Variablen (in PSQL) und Ausdrücke (in DSQL und PSQL) als Argumente.

(1.5) Neues Konstrukt für CASE Ausdrücke

Arno Brinkman

a) CASE

Emöglicht, dass das Ergebnis einer Spalte durch das Rückgabeergebnis einer Gruppe von exklusiven Bedingungen bestimmt wird.

Syntax

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

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

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

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

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

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

Beispiele

i) simple

```
SELECT
    o.ID,
    o.Description,
    CASE o.Status
      WHEN 1 THEN 'bestätigt'
```

```
      WHEN 2 THEN 'in Produktion'
      WHEN 3 THEN 'fertig'
      WHEN 4 THEN 'verschickt'
      ELSE 'Unbekannter Status ''' || o.Status || ''''
    END
FROM Orders o;
```

ii) searched

```
  SELECT
    o.ID,
    o.Description,
    CASE
      WHEN (o.Status IS NULL) THEN 'neu'
      WHEN (o.Status = 1) THEN 'bestätigt'
      WHEN (o.Status = 3) THEN 'in Produktion'
      WHEN (o.Status = 4) THEN 'fertig'
      WHEN (o.Status = 5) THEN 'verschickt'
      ELSE 'Unbekannter Status ''' || o.Status || ''''
    END
  FROM Orders o;
```

b) COALESCE

Ermöglicht, dass der Wert einer Spalte durch eine Anzahl von Ausdrücken ermittelt wird, wobei der erste Ausdruck, der einen Nicht-Null-Wert zurückliefert den Rückgabewert bestimmt.

Format

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

Syntaxregeln

i) COALESCE (V1, V2) ist gleichbedeutend mit Folgendem <case specification>:
CASE WHEN V1 IS NOT NULL THEN V1 ELSE V2 END

ii) COALESCE (V1, V2,..., Vn), für n >= 3, ist gleichbedeutend mit Folgendem:
<case specification>:
CASE WHEN V1 IS NOT NULL THEN V1 ELSE COALESCE (V2,...,Vn) END

Beispiele

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

SELECT
    COALESCE(Telefon,Handy,'Unbekannt') AS "Telefonnummer"
FROM
    Relations
```

c) NULLIF

Gibt NULL für einen untergeordneten Ausdruck zurück, wenn dieser einen spezifizierten Wert hat, ansonsten wird der Wert des untergeordneten Ausdrucks zurückgegeben.

Format

<case abbreviation> ::=

NULLIF <left paren> <value expression> <comma> <value expression> <right paren>

Syntaxregeln
NULLIF (V1, V2) ist gleichbedeutend mit folgender <case specification>:
CASE WHEN V1 = V2 THEN NULL ELSE V1 END

Beispiel

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

(1.5) SQL99-konforme Savepoints

Nickolay Samofatov

Anwenderspezifische Savepoints (auch *nested transactions* genannt) stellen eine praktische Methode zum Behandeln von Fehlern in der Ablauflogik dar, ohne ein komplettes Rollback der Transaktion durchführen zu müssen. Nur in DSQL verfügbar.

Benutzen Sie die SAVEPOINT Anweisung um einen Punkt in der Transaktion zu markieren, bis zu dem später ein Rollback durchgeführt werden kann.

```
SAVEPOINT <identifier>;
```

<identifier> gibt den Namen des Savepoints, der erstellt werden soll, an. Nachdem ein Savepoint erzeugt wurde, können Sie entweder mit dem Prozess fortfahren, die Transaktion mit Commit abschliessen, oder ein Rollback auf der kompletten Transaktion oder bis zu dem Savepoint durchführen.

Savepoint-Namen müssen innerhalb einer Transaktion eindeutig sein. Wenn ein zweiter Savepoint mit gleichem Namen wie ein früherer Savepoint angelegt wird, wird der frühere Savepoint gelöscht.

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

Diese Anweisung führt folgende Operationen aus:
- Alle Änderungen, die in der Transaktion nach dem Savepoint durchgeführt werden, werden rückgängig gemacht
- Alle Savepoints, die nach diesem Savepoint angelegt wurden, werden gelöscht. Der benannte Savepoint bleibt bestehen, d.h. es kann mehrfach ein Rollback bis zu diesem Savepoint durchgeführt werden. Ebenso bleiben frühere Savepoints bestehen.
- Alle impliziten und expliziten Satzsperren (record locks), die seit diesem Savepoint gesetzt wurden, werden aufgelöst. Andere Transaktionen, die Datensätze angefordert haben, die nach dem Savepoint gesperrt wurden, müssen weiterhin warten, bis die Transaktion per Commit oder Rollback abgeschlossen wird. Andere Transaktionen, die diese Datensätze bisher noch nicht angefordert hatten, können auf diese Datensätze sofort zugreifen.
  Hinweis: dieses Verhalten kann in künftigen Versionen geändert werden.

Die Verwaltung zum Rückgängigmachen von Savepoints kann große Mengen an Speicher auf dem Server beanspruchen, insbesondere dann, wenn ein Update auf die gleichen Datensätze innerhalb der gleichen Transaktion mehrfach durchgeführt wird. Benutzen Sie die RELEASE SAVEPOINT Anweisung, um die Systemressourcen, die von der Savepoint-Verwaltung reserviert wurden, freizugeben.
```
RELEASE SAVEPOINT <identifier> [ONLY];
```

Die RELEASE SAVEPOINT Anweisung löscht den Savepoint <identifer> aus dem Transaktionskontext. Solange nicht das Schlüsselwort ONLY verwendet wird, werden alle Savepoints, die seit dem Savepoint <Bezeichner> angelegt wurden, ebenfalls gelöscht.

Beispiel: Verwendung von Savepoints

```
a) create table test (id integer);
b) commit;
c) insert into test values (1);
d) commit;                          → 1 Datensatz
e) insert into test values (2);     → 2 Datensätze
f) savepoint y;
g) delete from test;
h) select * from test;       → kein Datensatz
i) rollback to y;            → Rollback bis (f)
j) select * from test;       → 2 Datensätze
k) rollback;                 → Rollback bis (d)
l) select * from test;       → 1 Datensatz
```

Interne Savepoints
Standardmäßig benutzt der Datenbankserver einen automatischen, systemgenerierten Savepoint auf Transaktionsebene um ein Rollback durchzuführen. Wenn Sie eine ROLLBACK Anweisung absetzen, werden alle Aktionen innerhalb dieser Transaktion bis zu dem Savepoint auf Transaktionsebene rückgängig gemacht und auf der Transaktion wird anschließend ein Commit durchgeführt. Diese Art der Implementierung reduziert das Aufkommen von "Garbage Collection", die von Transaktionen, die per Rollback abgeschlossen wurden, verursacht werden.

Wenn die Menge der Änderungen, die seit einem Savepoint auf Transaktionsebene durchgeführt werden sehr groß wird (10^4-10^6 betroffene Datensätze), gibt der Datenbankserver den Savepoint auf Transaktionsebene frei und benutzt den TIP (Transaction Inventory Page)-Mechanismus um ein etwaiges Rollback durchzuführen. Wenn Sie die Menge der zu erwartenden Änderungen als derart groß einschätzen, können Sie das TPB Flag isc_tpb_no_auto_undo verwenden, um das Anlegen eines Savepoints auf Transaktionsebene zu unterbinden.

Savepoints und PSQL
Das Implementieren von Anwender-Savepoints in die PSQL Umgebung würde der Regel der Atomizität von Anweisungen, einschließlich der Prozedur-Aufruf-Anweisungen, widersprechen. Firebird stellt daher ein Exception Handling in PSQL zur Verfügung, um Änderungen, die in einer Stored Procedures und Triggern vorgenommen wurden, rückgängig zu machen. Jede SQL/PSQL Anweisung wird unter einem System von automatischen, internen Savepoints ausgeführt, wobei entweder die komplette Anweisung komplett erfolgreich abgeschlossen oder ALLE Änderungen rückgängig gemacht werden und eine Exception ausgelöst wird. Jeder PSQL Exception Handling Block ist ebenso an automatische System-Savepoints gebunden.

(1.5) Explizites Sperren (Explicit locking)

Nickolay Samofatov

Das Hinzufügen der optionalen WITH LOCK Anweisung ermöglicht ein begrenztes explizites Pessimistisches Sperren (pessimistic locking), das allerdings lediglich mit Vorsicht angewendet werden soll, wenn die betroffenen Datensätze a) extrem wenige (im Idealfall ein einziger) sind, und b) von der Anwendung präzise kontrolliert werden.

HINWEIS: Der Gebrauch des pessimistischen Sperrverhaltens in Firebird ist selten notwendig und sollte genau verstanden sein, bevor davon Gebrauch gemacht wird.

Syntax

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

```
  [WITH LOCK]
...;
```

Wenn die WITH LOCK Anweisung erfolgreich durchgeführt werden konnte, wird eine sicherer Sperre (Lock) auf den selektierten Datensätzen gesetzt, wodurch verhindert wird, dass andere Transaktionen einen Schreibzugriff auf einen dieser oder davon abhängigen Datensätzen bekommt, bis Ihre Transaktion beendet wird.

Wenn die FOR UPDATE Anweisung mitangegeben wird, dann wird diese Sperre jedem Datensatz, einem nach dem anderen zugewiesen. Die Sperre erfolgt in der Reihenfolge, in der die Datensätze in den Server-Cache eingelesen werden. Dadurch ist es möglich, dass eine Sperre, die zum Zeitpunkt der Anfrage scheinbar erfolgreich war, dennoch in der Folge, beim Versuch einen Datensatz einzulesen, der mittlerweile von einer anderen Transaktion gesperrt wurde, fehlschlagen kann.

Es ist grundlegend, dass Sie die Auswirkungen der Transaktions-Isolationen und anderer Transaktionseigenschaften kennen, bevor Sie versuchen, ein explizites Sperren in Ihrer Anwendung zu verwenden.

Die SELECT... WITH LOCK Anweisung ist in DSQL und PSQL verfügbar. Es kann nur in einer Anweisung auf oberster Ebene (Top-Level) und einer SELECT-Abfrage auf einer einzelnen Tabelle erfolgreich sein. In einer Subquery oder in Join-Abfragen ist diese nicht verfügbar. Der DISTINCT-Operator darf nicht verwendet werden, ebenso wie eine GROUP BY Anweisung oder eine Aggregatsfunktion die Verwendung unterbindet. Des weiteren ist die Verwendung innerhalb eines VIEWs, einer externen Tabelle oder der Ergebnismenge einer selektierbaren Stored Procedure nicht möglich.

Verstehen der "WITH LOCK" Anweisung

Die Datenbank wird für jeden Datensatz, der unter eine expliziten Sperr-Anweisung fällt, entweder die Datensatzversion der zuletzt gültigen Version (unabhängig des Status der Datenbank zum Zeitpunkt der Abfrage) oder eine Exception zurückgeben.

Das Verhalten beim Warten und die Benachrichtigung über Konflikte hängen von den Transaktionsparametern, die im TPB Block definiert werden, ab.

TPB Modus	***Verhalten***
isc_tpb_consistency	Explizite Sperren (Locks) werden von impliziten oder expliziten Sperren (Locks) auf Tabellenebene überschrieben und damit ignoriert
isc_tpb_concurrency + isc_tpb_nowait	Wird ein Datensatz geändert, sei es durch eine Transaktion, die in dem Zeitraum, in dem die sperrende Transaktion versuchte die explizite Sperre zu setzen, abgeschlossen wurde, oder durch eine aktive Transaktion, die eine Änderung dieses Datensatzes durchgeführt hat, so wird sofort eine Fehlermeldung bzgl. eines vorliegenden "Update" Konflikts ausgegeben.
isc_tpb_concurrency + isc_tpb_wait	Wenn ein Datensatz durch irgendeine Transaktion geändert wurde, die abgeschlossen wurde, während die Transaktion versuchte eine Sperre zu setzen, dann wird sofort eine Fehlermeldung bzgl. eines vorliegenden "Update" Konflikts ausgegeben. Wenn eine aktive Transaktion den "Besitz" an einem Datensatz innehat (sei es durch eine explizite Sperre oder durch eine normale "Optimistische Schreibsperre"), dann wird die Transaktion, die versucht die Sperre zu setzen darauf warten, dass die sperrende Transaktion beendet wird. Tritt dies ein, wird die Transaktion erneut versuchen die Sperre zu setzen. Dies bedeutet, dass, wenn die blockierende Transaktion eine geänderte Version des Datensatzes mit Commit abschließt, eine Fehlermeldung bzgl. eines vorliegenden "Update" Konflikts auslösen wird.
isc_tpb_read_committed	Wenn es eine aktive Transaktion gibt, die den "Besitz" an diesem Datensatz

+ isc_tpb_nowait	innehat (sei es durch eine explizite Sperre oder ein normales Update), so wird sofort eine Fehlermeldung bzgl. eines vorliegenden "Update" Konflikts ausgegeben.
isc_tpb_read_committed + isc_tpb_wait	Wenn es eine aktive Transaktion gibt, die den "Besitz" an diesem Datensatz innehat (sei es durch eine explizite Sperre oder durch eine normale "Optimistische Schreib-Sperre"), dann wird die Transaktion, die versucht die Sperre zu setzen darauf warten, dass die sperrende Transaktion beendet wird. Tritt dies ein, wird die Transaktion erneut versuchen die Sperre zu setzen. Fehlermeldungen bzgl. eines vorliegenden "Update Konflikts" können in diesem TPB Modus bei Anweisungen zum expliziten Sperren nicht auftreten.

Wenn eine UPDATE Anweisung versucht auf einen Datensatz zuzugreifen, der von einer anderen Transaktion gesperrt ist, wird, abhängig von dem TPB Modus, entweder eine Fehlermeldung bzgl. des "Update Konflikts" ausgegeben, oder es wird auf das Ende der sperrenden Transaktion gewartet. Das Verhalten des Datenbanksystems entspricht damit dem Verhalten, als ob der Datensatz bereits durch die sperrende Transaktion verändert worden wäre. Es werden keine speziellen GDS-Codes zurückgegeben, falls es sich um einen Konflikt im Zusammenhang mit pessimistischen Sperren handelt.

Das Datenbanksystem garantiert, dass alle Datensätze, die von einer Anweisung mit explizitem Sperren zurückgegeben werden, auch wirklich gesperrt sind und den Einschränkungen in der WHERE Anweisung, solange sich diese Suchbedingungen nicht auf anderen Tabellen, etwa über Joins oder Subqueries beziehen, entsprechen. Ebenso ist sichergestellt, dass Datensätze, die nicht der Suchbedingung entsprechen, nicht von dieser Anweisung gesperrt werden. Es kann jedoch nicht garantiert werden, dass es Datensätze gibt, die der Suchbedingung entsprechen, jedoch nicht gesperrt wurden (und damit auch nicht zurückgegeben werden). Diese Situation kann auftreten, wenn parallele Transaktionen ihre Änderungen zurückschreiben, während die Anweisung die Sperren am Setzen ist.

Das Datenbanksystem sperrt die Sätze zu dem Zeitpunkt, zu dem sie in den Servercache gelesen werden. Dies hat wichtige Konsequenzen, wenn mehrere Sätze zugleich gesperrt werden. Bei vielen Zugriffsmethoden für Firebird, werden die Daten standardmäßig in Rückgabepaketen zu mehreren hundert Datensätzen ("buffered fetches") abgeholt. Die meisten Datenzugriffskomponenten können jedoch die Datensätze, aus dem zuletzt abgeholten Datenpaket nicht verarbeiten, wenn dort dabei ein Fehler auftrat.

Die FOR UPDATE Anweisung stellt eine Technik zur Verfügung, um die Verwendung von sog. "Buffered Fetches" zu unterbinden. Geben sie dazu optional OF <Feldnamen> an, um positionsbezogene Updates durchführen zu können. Alternativ hierzu kann evtl. die Eigenschaft FetchBuffer ihrer Datenzugriffskomponente auf 1 gesetzt werden. Dies würde sie in die Lage versetzen, den aktuellen, gesperrten Datensatz, vor dem "Abholen" des nächsten Datensatzes zu verarbeiten oder eine Methode zur Fehlerbehandlung einzuführen, die es Ihnen ermöglicht weiterzufahren, ohne die Transaktion per Rollback abschließen zu müssen.

Ein Rollback auf impliziten oder expliziten Savepoints gibt alle Datensatzsperren, die unter diesem Savepoint vorgenommen wurden, frei; es gibt dabei jedoch keinen Mechanismus, der die wartenden Transaktionen benachrichtigt! Anwendungen sollten sich jedoch nicht auf dieses Verhalten verlassen, da dies in Zukunft geändert werden könnte.

Da explizite Sperren dazu verwendet werden können um Konflikte bei Updates außerhalb des üblichen Rahmens zu verhindern oder zu verarbeiten, wird die Anzahl der Deadlock Fehler zunehmen, es sei denn, sie konzipieren ihre Strategie des Sperrens sorgfältig und kontrollieren sie rigoros. Die meisten Anwendungen werden überhaupt keine expliziten Sperren verwenden müssen. Die Hauptanwendungsgebiete für explizites Sperren liegen in (1) dem Verhindern von aufwändigen Fehlerbehandlungsroutinen bei Update-Konflikten in Anwendungen mit massiven Datenzugriffen und (2) der Bewahrung der Integrität von Objekten, die in einer relationalen Datenbank in Cluster-Umgebungen abgelegt

werden. Wenn explizites Sperren nicht aufgrund einer dieser beiden Kategorien eingesetzt wurde, dann haben Sie den falschen Weg gewählt um eine Aufgabe in Firebird zu erledigen.

Das explizite Sperren ist ein fortgeschrittenes Feature, mißbrauchen Sie es nicht! Während Lösungen dieser Art in Websites, die tausende von gleichzeitigen Schreibzugriffen behandeln müssen, oder in ERP/CRM-Systemen die in großen Firmen eingesetzt werden, sehr wichtig sind, werden die meisten Anwendungsprogramme nicht unter solchen Bedingungen arbeiten müssen.

Beispiele

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

ii) (mehrere Datensätze, Abholen der Datensätze, "einer-nach-dem-anderen" mit einem DSQL Cursor)
```
SELECT * FROM DOCUMENT WHERE PARENT_ID=? FOR UPDATE WITH LOCK
```

(1.5) Verbesserte Behandlung von Aggregaten

Arno Brinkman

Bisher konnten gruppierte Datenmengen nur anhand eines benannten Feldes gruppiert werden. Mit Firebird 1.0 ist es möglich auch mittels eines UDF-Ausdruckes zu gruppieren. In 1.5 wurden einige weitere Erweiterungen bzgl. der Behandlung von Aggregatsfunktionen und der GROUP BY Anweisung eingeführt, die es nun ermöglichen, Gruppierung anhand des *Grades des Feldes* in der Rückgabespezifikation (ihrer 1-basierten "ordinalen Position von links-nach-rechts", wie auch in der ORDER BY Anweisung) oder durch eine Vielzahl von Ausdrücken durchzuführen.
HINWEIS: Momentan sind nicht alle Ausdrücke innerhalb der GROUP BY Liste zulässig, z.B. ist die Verkettung (concatenation) nicht erlaubt.

Group By Syntax

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

group_by_list : group_by_item [, group_by_list];

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

group_by_function : numeric_value_function
                | string_value_function
                | case_expression
                ;

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

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

Das group_by_item darf keine Referenz zu einer Aggregatsfunktion (einschließlich aller, die innerhalb eines Ausdrucks "versteckt" sind) innerhalb des gleichen Kontext enthalten.

HAVING

Die HAVING-Anweisung erlaubt lediglich Aggregatsfunktionen oder gültige Ausdrücke, die Bestandteil der GROUP BY Anweisung sind. Bisher war es erlaubt, Felder, die nicht Teil der GROUP BY Anweisung wa-ren und ungültige Ausdrücke zu verwenden.

ORDER BY
Wenn es sich um eine Aggregatsanweisung handelt, dann darf die ORDER BY Anweisung nur gültige Ausdrücke verwenden, die Aggregatsfunktionen oder Teile der Ausdrücke in der GROUP BY Anweisung sind. Bisher war es erlaubt, ungültige Ausdrücke zu verwenden.

Aggregatsfunktionen innerhalb von Unterabfragen
Es ist jetzt möglich eine Aggregatsfunktion oder –Ausdruck, die innerhalb einer Unter-abfrage der GROUP BY Anweisung liegt, zu verwenden.

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

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

Vermischen von Aggregatsfunktionen aus verschiedenen Kontexten
Aggregatsfunktionen aus verschiedenen Kontexten können jetzt innerhalb eines Ausdrucks verwendet werden.

Beispiel
```
SELECT
    r.RDB$RELATION_NAME,
    MAX(i.RDB$STATISTICS) AS "Max1",
    (SELECT
       COUNT(*) || ' - ' || MAX(i.RDB$STATISTICS)
     FROM
       RDB$RELATION_FIELDS rf
     WHERE
       rf.RDB$RELATION_NAME = r.RDB$RELATION_NAME) AS "Max2"
  FROM
```

```
    RDB$RELATIONS r
    JOIN RDB$INDICES i on (i.RDB$RELATION_NAME = r.RDB$RELATION_NAME)
  GROUP BY
    r.RDB$RELATION_NAME
  HAVING
    MIN(i.RDB$STATISTICS) <> MAX(i.RDB$STATISTICS)
```

Hinweis! Diese Abfrage führt auch in FB1.0 zu einem Ergebnis, dieses ist jedoch FALSCH!

Unterabfragen werden innerhalb einer Aggregatsfunktion unterstützt
Die Verwendung eines "singleton select" Ausdrucks innerhalb einer Aggregatsfunktion wird unterstützt.

Beispiel

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

Verschachtelte Aggregatsfunktionen
Die Verwendung einer Aggregatsfunktion innerhalb einer anderen Aggregatsfunktion ist möglich, wenn die innere Aggregatsfunktion aus einem niedrigeren Kontext stammt (siehe Beispiel oben).

Gruppieren nach Grad (Ordinale Zahl)
Bei der Verwendung des Grades des Ausgabefeldes in der GROUP BY Anweisung, wird der Ausdruck aus der SELECT-Liste "kopiert" (wie dies auch die ORDER BY Anweisung durchführt). Wenn eine Gradzahl auf eine Unterabfrage verweist, dann bedeutet dies, dass diese Unterabfrage auch mindestens zweimal ausgeführt wird.

(1.5) ORDER BY Anweisung kann Ausdrücke und die Einordnung von NULLs spezifizieren

Nickolay Samofatov

Die ORDER BY Anweisung ermöglicht es Ihnen, jegliche gültigen Ausdrücke zur Sortierung der Rückgabemenge zu spezifizieren. Wenn der Ausdruck aus einer einzelnen Zahl besteht, wird er, wie zuvor, als Spalte (Gradzahl) interpretiert.
Die Sortierung von Nulls innerhalb der Rückgabemenge kann durch die Verwendung der "Nulls Placement" Anweisung kontrolliert werden. Ergebnismengen können dergestalt sortiert werden, dass Nulls entweder vor (NULLS FIRST) oder nach (NULLS LAST) den sortierten Nicht-Null-Werten stehen.
Wird die nulls_placement Anweisung nicht benutzt, wird NULLS LAST verwendet.

Syntax

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

Einschränkungen

- Wird NULLS FIRST verwendet, wird bei der Sortierung kein Index verwendet.
- Die Ergebnisse einer Sortierung, die auf Werten basieren, die von einer UDF oder einer Stored Procedure zurückgegeben werden, sind unvorhersehbar, da diese Werte nicht zur Bestimmung einer logischen Sortiersequenz verwendet werden können.
- Die Anzahl der Aufrufe einer Prozedur zur Bestimmung einer Sortierung die auf einer UDF oder einer Stored Procedure basieren sind nicht vorhersehbar. Dabei spielt es keine Rolle, ob die Sortierung durch einen Ausdruck oder eine ordinale Zahl, die einen Ausdruck innerhalb der Feldliste repräsentiert, bestimmt wird.
- Eine Sortieranweisung für das Sortieren von Datenmengen, die auf einer Union Abfrage basieren, darf lediglich Ordinalzahlen (Grade) verwenden, um auf Sortierfelder zu verweisen.

Beispiele

i)

```
SELECT * FROM MSG
ORDER BY PROCESS_TIME DESC NULLS FIRST
```

ii)

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

iii)

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

PSQL (Stored Procedure und Trigger Sprache)

(1.5) EXECUTE STATEMENT

Alex Peshkov

PSQL-Erweiterung, die einen String, der eine gültige dynamische SQL-Anweisung enthält nimmt, und ihn ausführt, als wäre diese Anweisung direkt an DSQL übergeben worden.
Verfügbar in Triggern and Stored Procedures.

Die Syntax kann drei Formen annehmen.

Syntax 1
Führt <string> als SQL-Operation aus, die keinerlei Daten zurückliefert, also INSERT, UPDATE, DELETE, EXECUTE PROCEDURE oder jegliche DDL Anweisung, außer CREATE/DROP DATABASE.

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

Beispiel

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

Syntax 2
Führt <string> als SQL-Operation aus, die eine einzelne Datenzeile zurückliefert. Nur singleton SELECT Operatoren dürfen mit dieser Form des EXECUTE STATEMENT ausgeführt werden.

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

Beispiel

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

Syntax 3
Führt <string> als SQL-Operation aus, die mehrere Datenzeilen zurückgibt. Jeglicher SELECT Operator kann mit dieser Form des EXECUTE STATEMENT ausgeführt werden.

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

Beispiel

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

Zusätzliche Bemerkungen zu EXECUTE STATEMENT

Der "EXECUTE STATEMENT" DSQL-String kann keine Parameter in jeglicher Syntax-Variation beinhalten. Alle Variablen-Ersetzungen in dem statischen Teil der SQL Anweisung müssen vor der Ausführung des EXECUTE STATEMENT gesetzt werden.

Dieses Feature ist lediglich für die behutsame Verwendung gedacht und es sollten vorher immer alle Faktoren in Betracht gezogen werden. Eine Grundregel sollte sein, dass von EXECUTE STATEMENT lediglich Gebrauch gemacht wird, wenn andere Methoden unmöglich sind, oder eine schlechtere Performanz als EXECUTE STATEMENT haben.

EXECUTE STATEMENT ist auf verschiedene Arten potentiell unsicher:

1. Es gibt keine Möglichkeit die eingeschlossene Anweisung auf syntaktische Korrektheit zu überprüfen.
2. Es gibt keine Abhängigkeitsüberprüfungen, um zu prüfen, ob Tabellen oder Spalten entfernt worden sind.
3. Operationen sind langsam, da die eingebettete Anweisung vor jeder Ausführung vorbereitet (Prepare) werden muss.
4. Die Datentypen der Rückgabewerte werden strikt überprüft, um unvorhersehbare Typecasting Exceptions zu vermeiden. Zum Beispiel würde der String "1234" sich in den Integer 1234 konvertieren lassen, "abc" jedoch würde einen Konvertierungsfehler hervorrufen.
5. Wenn die Stored Procedure spezielle Privilegien auf einigen Objekten hat, wird die dynamische Anweisung, die an EXECUTE STATEMENT übergeben wird, diese nicht erben. Privilegien sind auf diejenigen beschränkt, die dem Benutzer (USER), der diese Prozedur ausführt, zugewiesen sind.

(1.5) Neue Kontext-Variablen

Dmitry Yemanov

CURRENT_CONNECTION

und

CURRENT_TRANSACTION

Jeder dieser Kontext-Variablen gibt den System-Identifizierer der aktiven Datenbankverbindung oder des aktuellen Transaktionskontextes zurück. Der Rückgabewert ist vom Typ INTEGER. Verfügbar in DSQL und PSQL. Weil diese Werte in der Header-Page der Datenbank gespeichert werden, werden sie nach einem Datenbank-Restore zurückgesetzt.

Syntax

```
CURRENT_CONNECTION
CURRENT_TRANSACTION
```

Beispiele

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

ROW_COUNT

Gibt in einem Integer die Anzahl der von der letzten DML-Anweisung betroffenen Zeilen zurück.
Verfügbar in PSQL, im Kontext der Procedure- oder Trigger-Module. Zur Zeit wird für eine SELECT Anweisung 0 zurückgegeben.

Syntax

```
ROW_COUNT
```

Beispiel

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

Hinweis: diese Variable kann nicht dazu benutzt werden, die Anzahl der betroffenen Datensätze, die von einer EXECUTE STATEMENT Anweisung betroffen waren, zurückzugeben.

SQLCODE und GDSCODE

Jede Kontext-Variable gibt einen Integer, der den numerischen Fehlercode für die aktive Exception repräsentiert, zurück. Verfügbar in PSQL, innerhalb des Geltungsbereiches des speziellen Exception-Handling-Blocks. Beide sind außerhalb dieser Blocks mit Null (0) belegt.

Die GDSCODE Variable ist eine numerische Repräsentation des GDS (ISC) Fehler-Codes, z.B. "335544349L" gibt 335544349 zurück.

Ein "WHEN SQLCODE" oder "WHEN ANY" Exception-Block wird einen Nicht-Null-Wert für die SQLCODE Variable abfangen und für den GDSCODE Null (0) zurückgeben. Lediglich ein "WHEN GDSCODE" Block wird Nicht-Null GDSCODE Variable abfangen (und damit Null (0) im SQLCODE zurückgeben). Wenn eine

benutzerdefinierte Exception ausgelöst wird, sind sowohl die SQLCODE- als auch die GDSCODE-Variablen mit Null (0) belegt, ungeachtet des Typs des Exception Handling Blocks.

Syntax

```
SQLCODE
GDSCODE
```

Beispiel

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

Siehe auch EXCEPTION HANDLING ERWEITERUNG, weiter unten, und das Dokument README.exception_handling in dem firebird2/doc/sql.extensions Zweig des Firebird CVS Baums.

INSERTING

UPDATING

DELETING

Drei pseudo-Boolesche Ausdrücke die abgefragt werden können, um den Typ der ausgeführten DML-Operation zu bestimmen. Verfügbar in PSQL, nur in Triggern. Für den Gebrauch in Multi-Aktions-Triggern gedacht (siehe oben METADATEN).

Syntax

```
INSERTING
UPDATING
DELETING
```

Beispiel

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

(1.5) Erweiterungen des Exception-Handlings in PSQL

Dmitry Yemanov

Die allgemeine Syntax für eine EXCEPTION Anweisung in PSQL ist:

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

Die Erweiterungen in 1.5 erlauben es
1) Eine Laufzeit Fehlermeldung für eine benannte Exception zu definieren.
2) Eine abgefangene Exception innerhalb des Geltungsbereiches des Exception-Blocks erneut auszulösen.

3) Einen numerischen Fehlerwert für eine abgefangene Exception auszulesen.

1) Exception-Nachrichten zur Laufzeit

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

Beispiele
i)
```
EXCEPTION E_EXCEPTION_1 'Fehler!';
```
ii)
```
EXCEPTION E_EXCEPTION_2 'Falscher Datentyp für das Feld ID=' || new.ID;
```

2) Erneutes Auslösen einer Exception
Hinweis – dies hat außerhalb des Exception-Blocks keinerlei Auswirkung.

Syntax
```
EXCEPTION;
```

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

3) Fehler-Codes zur Laufzeit
Siehe oben: SQLCODE / GDSCODE.

(1.5) LEAVE | BREAK Anweisung

Beendet den Durchlauf einer Schleife und führt den Code ab der Stelle nach der END Anweisung, die die Schleife beendet, weiter aus. Nur in WHILE, FOR SELECT und FOR EXECUTE Konstrukten verfügbar, ansonsten wird eine Parser Fehlermeldung ausgegeben. Das SQL-99 Standard Schlüsselwort LEAVE macht das existierende BREAK obsolet. Verfügbar sowohl in Triggern als auch in Stored Procedures.

Syntax

```
LEAVE;
```

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

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

HINWEIS LEAVE | BREAK und EXIT Anweisungen können jetzt in Triggern benutzt werden

(1.5) Gültige PLAN Anweisungen können jetzt in Triggern eingebunden werden

Ignacio J. Ortega

Bisher wurde ein Trigger, der eine PLAN Anweisung enthielt, vom Compiler zurückgewiesen. Jetzt kann ein gültiger Plan angegeben und benutzt werden.

(1.5) Leere BEGIN..END Blöcke

Dmitry Yemanov

Leere BEGIN..END Blöcke in PSQL Modulen sind jetzt erlaubt. Es können jetzt "verstümmelte" Module geschrieben werden, wie z.B.:

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

(1.5) Deklaration und Definition von lokalen Variablen in einer einzigen Anweisung

Claudio Valderrama

Vereinfacht die Syntax und erlaubt es, dass lokale Variablen in einer einzigen Anweisung deklariert und definiert (oder initialisiert) werden.

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

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

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

(1.5) SELECT FIRST kann jetzt auch Null (0) als Argument verarbeiten

FB 1.5 erlaubt Null (0) als ein Argument für den FIRST Befehl. Eine leere Ergebnismenge wird zurückgeliefert.

Es werden die ersten *m* Zeilen der gewählten Ergebnismenge zurückgegeben. Durch die optionale SKIP Anweisung werden die ersten *n* Zeilen verworfen und es werden m Datensätze ab der Position n+1 zu-

rückgegeben. In der einfachsten Form sind *m* und *n* Integer, aber ebenso ist jegliche Firebird-Anweisung, die einen Integer zurückgibt, gültig. Ein Identifizierer, der einen Integer darstellt, kann in GDML verwendet werden, dies ist jedoch in SQL oder DSQL nicht zulässig.

Klammern sind für Ausdrucksargumente zwingend notwendig, ansonsten sind sie optional.

Es können auch Variablen eingebunden werden, so gibt z.B. SKIP ? * FROM ATABLE die verbliebene Ergebnismenge zurück, nachdem die ersten *n* Datensätze am Beginn ausgeschlossen wurden, wobei n den Wert, der in der Variable ? übergeben wurde, darstellt. SELECT FIRST ? COLUMNA, COLUMNB FROM ATABLE gibt die ersten *m* Datensätze zurück und verwirft den Rest.

Die FIRST Anweisung ist optional, d.h. sie können SKIP in einer Anweisung ohne FIRST verwenden, um eine Ergebnismenge, ohne die Datensätze die SKIP ausschließt, zu erhalten.

Verfügbar in SQL und DSQL mit Ausnahme der angegeben Einschränkungen.

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

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

SELECT FIRST 5 DISTINCT FIELD FROM MYTABLE;
```

Zwei Fallstricke bei SELECT FIRST

1. Diese Anweisung

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

wird alle Zeilen in der Tabelle löschen. Autsch! Die Unterabfrage wählt jeweils 10 Kandidaten für das Löschen aus, löscht sie, holt sich die nächsten 10 Sätze … etc., bis keine Sätze mehr vorhanden sind. Also: VORSICHT!

2. Abfragen wie z.B.:
```
...
WHERE F1 IN ( SELECT FIRST 5 F2 FROM TABLE2 ORDER BY 1 DESC )
```

Werden nicht wie erwartet abgearbeitet, da die Optimierung, die vom Datenbanksystem durchgeführt wird, korrelierende WHERE...IN (SELECT...) Prädikate in korrelierende EXITS Prädikate umsetzt. Es ist offensichtlich, dass in diesem Falle der Gebrauch von FIRST N nicht sinnvoll ist:

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

Erweiterungen der Zeichensätze

Neu in 1.5

- Die WIN1251_UA Collation (sowohl für die russische als auch die ukrainische Sprache) für den WIN1251 Zeichensatz wurde hinzugefügt.
- Korrektur bei Default Uppercase in WIN1251
- Hinzugefügt wurde ISO_HUN (für ungarische Sprache) für den ISO8859_2 Zeichensatz

Neu hinzugefügte Zeichensätze (keine nicht-binäre Collations)

Blas Rodriguez Somoza

- DOS737 PC Greek (Griechisch)
- DOS775 PC Baltic (Baltisch)
- DOS858 Variante des Cp850 mit dem Euro Zeichen (€)

- DOS862 PC Hebrew (Hebräisch)
- DOS864 PC Arabic (Arabisch)
- DOS866 MS-DOS Russian (Russsich)
- DOS869 IBM Modern Greek (modernes griechisch)
- WIN1255 Windows Hebrew (Hebräisch)
- WIN1256 Windows Arabic (Arabisch)
- WIN1257 Windows Baltic (Baltisch)
- ISO8859_3 Latin 3 (Esperanto, Maltekisch, Pinyi, Sami, kroatisch und andere)
- ISO8859_4 Latin 4 (Baltisch, grönländisch, lappländisch)
- ISO8859_5 Cyrillic (kyrillisch)
- ISO8859_6 Arabic (arabisch)
- ISO8859_7 Greek (griechisch)
- ISO8859_8 Hebrew (hebräisch)
- ISO8859_9 Turkish (türkisch)
- ISO8859_13 Baltic (baltisch)

Neu in 1.0

- Es wurde eine "case insensitive" ungarische Sortierung hinzugefügt, die von Sandor Szollosi (ssani@freemail.hu) entwickelt und getestet wurde.

- Firebird unterstützt jetzt den Zeichensatz ISO8859-2 (für die tschechische Sprache).

SPRACHERWEITERUNGEN IN FIREBIRD 1.0.x

Die folgenden Spracherweiterungen in Firebird 1.0.x werden der Vollständigkeit halber nochmals angeführt.

(1.0) CURRENT_USER und CURRENT_ROLE

Diese beiden neuen Kontextvariablen wurden hinzugefügt, um den aktuell angemeldeten Benutzer (USER) und, falls implementiert ([1]), die ROLE im Kontext der aktuellen Verbindung zurückzugeben.

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

COMMIT;

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

CURRENT_USER ist ein DSQL Synonym für USER, das im SQL Standard definiert ist. Diese sind identisch. CURRENT_USER besitzt keine Vorteile gegenüber USER.

[1] Wenn Sie beabsichtigen eine InterBase v.4.x oder 5.1 Datenbank mit Firebird zu verwenden, dann berücksichtigen Sie, dass ROLE nicht unterstützt wird, d.h. CURRENT_ROLE ist NONE (wie im SQL Standard vorgeschlagen, wenn keine explizite ROLE vorhanden ist), sogar wenn der Benutzer eine ROLE übergeben hat. Wenn Sie IB 5.5, IB 6 oder Firebird verwenden, dann wird die übergebene ROLE verifiziert. Falls die Role nicht existiert, dann wird diese auf NONE zurückgesetzt, ohne einen Fehler zurückzugeben.

Dies bedeutet, dass Sie mit FB niemals eine ungültige ROLE mit CURRENT_ROLE zurückbekommen, weil diese auf NONE zurückgesetzt wird. Dies ist unterschiedlich zu IB, wo dieser Wert intern mitgeführt wird, allerdings nicht über SQL zugänglich ist.

(1.0) DROP GENERATOR

Hiermit können nicht mehr benötigte Generatoren aus der Datenbank entfernt werden. Die Wiederverwendung des belegten Speicherplatzes wird mit dem nächsten Restore freigegeben. Verfügbar in SQL und DSQL.

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

(1.0) GROUP BY UDF

Es ist nun möglich ein SELECT nach einem Ergebnis einer UDF zu gruppieren.
z.B.

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

Ein Seiteneffekt dieser Neuerung ist, wo vormals die Verwendung einer internen Funktion in einem GROUP BY nicht möglich war, nun durch die Verwendung einer "Dummy UDF" (im Beispiel unten bin_or) möglich ist:

```
select count(*)
from rdb$relations r
group by bin_or((select count(rdb$field_name) from rdb$relation_fields f
where f.rdb$relation_name = r.rdb$relation_name),1)
```

(1.0) RECREATE PROCEDURE

Diese neue DDL Anweisung ermöglicht Ihnen nun eine Stored Procedure mit dem selben Namen einer bereits existierenden Stored Procedure zu erzeugen, die die alte Stored Procedure ersetzt, ohne vorher die Stored Procedure entfernen (DROP) zu müssen. Die Syntax ist identisch zu CREATE PROCEDURE.
Verfügbar in SQL und DSQL.

(1.0) RECREATE TABLE

Diese neue DDL Anweisung ermöglicht Ihnen eine neue Struktur für eine bestehenden Tabelle zu erstellen, ohne vorher die Tabelle entfernen (DROP) zu müssen.

Beachten Sie, dass RECREATE TABLE existierende Daten in der alten Tabelle nicht erhält.

Verfügbar in SQL und DSQL.

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

Interne Funktion, die die ANSI SQL SUBSTRING() Funktion implementiert. Diese Funktion gibt eine Zeichenkette, beginnend bei <*pos*> und bis zum Ende der übergegebenen Zeichenkette, zurück. Wurde die FOR <*length*> Option mitangegeben, dann wird die Anzahl der Bytes bis zu <*length*> bzw. bis zum Ende des Eingabestrings zurückgegeben.

Beim ersten Argument kann es sich um einen beliebigen Ausdruck, einer Konstante oder einen Identifizierer, der einen String darstellt, handeln.
<*pos*> muss ein Integer sein.
<*pos*> beginnt bei 1, wie andere SQL Kommandos auch.
Weder <*pos*> noch <*length*> können Abfrageparameter sein.

Da <pos> und <length> Byte-Positionen darstellen, kann der Identifzierer ein binärer BLOB, oder ein sub_type 1 Text-Blob mit zugrundeliegendem "Ein-Byte-Pro-Zeichen" Zeichensatz sein. Die Funktion kann derzeit keine Text-Blobs mit chinesischem (Maximum von 2 Byte/Zeichen) oder Unicode (Maximum von 3 Byte/Zeichen) Zeichensatz verarbeiten. Für ein String-Argument (im Gegensatz zu einem Blob) unterstützt die Funktion jeden Zeichensatz.

Verfügbar in SQL und DSQL.

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

Beachten Sie auch den Abschnitt für externe Funktionen (UDFs), um detaillierte Informationen zu Änderungen an den externen Substring-Funktionen in der Standard UDF Bibliothek zu erhalten.

(1.5) Erweiterung zu einzeiligen Kommentaren

Dmitry Yemanov

Einzeilige Kommentare können sich nun an einer beliebigen Stelle befinden, nicht nur am Anfang. D.h., in 1.5 kann der "--" Marker für einen Kommentar am Ende der Zeile in einem Skript, einer Stored Procedure, Trigger oder DSQL Anweisung verwendet wird. Dies kann dazu verwendet werden, um unerwünschte Teile von Anweisungen auszukommentieren. Jedes Zeichen nach dem "--" Marker bis zum nächsten Zeilenumbruch werden ignoriert.

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

(1.0) Neuer Kommentar-Marker

Claudio Valderrama

Für die Verwendung in Skripts, DSQL, Stored Procedures und Triggern.

Beispiel

```
-- This is a comment
```

Dieser neue Marker kann verwendet werden, um eine einzelne Codezeile in einem Skript, einer DDL/DML Anweisung, Stored Procedure oder Trigger auszukommentieren.

Die Logik für das Ignorieren von Zeichen ist wie folgt:

1. Überspringe "--" falls dies als erstes Zeichenpaar nach einem End-Of-Line Marker (LF unter Linux/ Unix, CRLF unter Windows) angetroffen wird
2. Ignorieren weiterer Zeichen bis zum nächsten End-Of-Line Marker

Diese Logik sollte nicht mit der Logik des Blockkommentars (/* a comment */) vermischt werden. In anderen Worten, verwenden Sie "--" nicht innerhalb eines Blockkommentars, und umgekehrt.

INTERAKTIVE ISQL SITZUNGEN: Beachten Sie folgendes, wenn Sie in einer **isql** Sitzung arbeiten. **Isql** akzeptiert Teile von Anweisungen in weiterführenden Abschnitten durch Anzeige des "CON>" Prompt bis das Terminalsymbol (normalerweise "**;**") eingegeben wird. Wenn Sie "--" zu Beginn einer weiterführenden Eingabezeile eingeben, dann wird die Logik zum Ignorieren von Zeichen mit dem End-Of-Line Marker beendet.

Das Problem bei isql liegt darin, dass isql eigene spezielle Kommandos besitzt, die nur von isql geparst werden sollen. Falls diese, aufgrund einer möglichen Verwendung von "--" nicht als solche erkannt werden, so kann es vorkommen, dass diese an die Engine weitergeleitet werden. Dies hat zur Folge, dass die Engine die SET und SHOW Kommandos nicht versteht, und entsprechend zurückweist.

(1.0) Alter Trigger erhöht nun nicht mehr den Änderungszähler einer Tabelle

Wenn die Anzahl der Metadatenänderungen einer Tabelle das Limit von 255 erreicht hat, dann können keine weiteren Metadatenänderungen an dieser Tabelle vorgenommen werden. Ein Backup/Restore Zyklus ist notwendig, um diesen Änderungszähler für jede Tabelle zurückzusetzen. Danach können wieder Änderungen an der Tabellenstruktur vorgenommen werden. Dieses Verhalten hatte zur Absicht, die Datenbank den notwendigen Aufräumarbeiten (Backup/Restore) auszusetzen, falls eine Tabellenstruktur vielen Änderungen unterworfen war.

Früher wurde bei jedem Ändern des Status von ACTIVE | INACTIVE eines Triggers in einer ALTER TRIGGER Anweisung der Änderungszähler der zugrundeliegenden Tabelle erhöht. Dies hatte zur Folge, dass die Nützlichkeit des Deaktivieren und Aktivieren eines Triggers für reguläre Operationen in Frage gestellt werden konnte, da der Änderungszähler sich rasch erhöhte.

Neue reservierte Schlüsselwörter

Die folgenden **neuen Firebird Schlüsselwörter** sollten zur veröffentlichten InterBase 6.0.1 Liste der reservierten Schlüsselwörter hinzugefügt werden.

BIGINT (1.5)	CASE (1.5)	CURRENT_CONNECTION (1.5)
CURRENT_ROLE	CURRENT_TRANSACTION (1.5)	CURRENT_USER
RECREATE	ROW_COUNT (1.5)	RELEASE
SAVEPOINT		

Nachfolgende Schlüsselwörter sind für eine zukünftige Verwendung reserviert:

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

Nachfolgende Schlüsselwörter waren in Firebird 1.0 reserviert, sind dies jedoch in Firebird 1.5 nicht mehr:

BREAK	DESCRIPTOR	FIRST
IIF	SKIP	SUBSTRING

Nachfolgende, nicht-reservierte Wörter, werden in 1.5 abhängig vom verwendeten Kontext als Schlüsselwörter erkannt:

COALESCE	DELETING	INSERTING
LAST	LEAVE	LOCK
NULLIF	NULLS	STATEMENT
UPDATING	USING	

Die nachfolgenden **neuen InterBase 6.5 und 7 Schlüsselwörter** (nicht reserviert in Firebird) sollten aus Kompatibilitätsgründen ebenfalls als reserviert angesehen werden:

BOOLEAN	FALSE	GLOBAL
PERCENT	PRESERVE	ROWS
TEMPORARY	TIES	TRUE

ISQL Features

"readline" Fähigkeit in der isql Shell

Mark O'Donohue

Eine Anweisungshistorie ist der isql Shell hinzugefügt worden (vergleichbar mit Unix readline). Sie können nun die ausgeführten Anweisungen mit den Cursor Auf/Ab Tasten durchblättern.

Benutzerdefinierte Funktionen (User-Defined Functions)

In ib_udf

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

Füllt den String *instring* rechts bis zur angegebenen Länge *length* durch anhängen von *padcharacter*s auf. Der übergebene String kann eine beliebige Länge von weniger als 32766 Bytes aufweisen. Die Länge *length* darf 32765 Bytes nicht überschreiten.

Deklaration

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

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

Füllt den String *instring* links bis zur angegebenen Länge *length* durch voranstellen von *padcharacter*s auf. Der übergebene String kann eine beliebige Länge von weniger als 32766 Bytes aufweisen. Die Länge *length* darf 32765 Bytes nicht überschreiten.

Deklaration

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

log (*x, y***)**
Paul Vinkenoog

Diese Funktion wies einen alten Fehler auf, der die Argumente x und y irrtümlicherweise vertauscht hatte. Es sollte der Logarithmus von y zur Basis x zurückgeliefert werden, zurückgeliefert wurde jedoch der Logarithmus von x zur Basis y. Dies wurde nun berichtigt. Sollten Sie diese Funktion in der Vergangenheit verwendet haben, dann ÜBERPRÜFEN SIE BITTE IHREN ANWENDUNGS-CODE! Entweder es wurde ein falscher Wert zurückgegeben oder jemand behalf sich eines Workarounds, um die richtige Berechnung zu erhalten.

In fbudf

1. Die *NVL und *NULLIF Funktionen wurden aus Abwärtskompatibilitätsgründen beibehalten. Diese sind jedoch nun durch die Einführung der neuen internen Funktionen CASE, COALESCE und NULLIF osolet (deprecated).

2. Es sollte darauf hingewiesen werden, dass fbudf keine String Felder größer als 32 Kb – 1 Byte verarbeiten kann. Dieses Limit könnte sich auf Zeichenketten auswirken, die miteinander verkettet werden, bevor diese an die UDF übergeben werden. Falls die Summe der Felder dieses Limit überschreiten, dann ist das Verhalten undefiniert. Die Funktion könnte ein falsches Ergebnis zurückliefern, oder fbudf könnte eine nicht zulässige Operation durchführen.

3. Wenn Sie eine Datenbank, die mit Firebird 1.0.x erstellt wurde, portieren und Sie die fbudf Funktionen **truncate** und **round** verwenden, so beachten Sie, dass diese Deklarationen mit Firebird 1.5 nicht mehr funktionieren werden, da die Namen der Einsprungspunkte geändert wurden. Diese

Funktionen müssen aus der Datenbank entfernt und mit den Deklarationen aus dem fbudf.sql Skript im 1.5 / UDF Verzeichnis neu deklariert werden.

Neue Konfigurationsdatei – firebird.conf

Das Firebird Stammverzeichnis

Das Stammverzeichnis Ihrer Firebird-Installation wird auf unterschiedliche Art und Weise verwendet, nämlich während der Installation und des weiteren sind Serverroutinen, Konfigurationsparameter und Clients davon abhängig. Da unterschiedliche Möglichkeiten existieren um dem Server mitzuteilen, wo das Stammverzeichnis zu finden ist, sollten sich Entwickler und Systemadministratoren bewusst sein, welcher Reihenfolge der Server beim Hochstarten folgt, um das Stammverzeichnis korrekt zu ermitteln.

Win32 Superserver und Classic Builds (Server und Client):
1) FIREBIRD Umgebungsvariable
2) RootDirectory Parameter in firebird.conf
3) Registry:
   HKLM\SOFTWARE\Firebird Project\Firebird Server\Instances\DefaultInstance
   und sucht das Feld DefaultInstance.
4) Das Verzeichnis eine Ebene über dem Verzeichnis, in dem sich die ausführbaren Dateien des Servers befinden

Win32 Embedded:
1) FIREBIRD Umgebungsvariable
2) RootDirectory Parameter in firebird.conf
3) Das Verzeichnis, in dem sich fbembed.dll (fbclient.dll umbenannt) befindet

Linux Classic:
1) FIREBIRD Umgebungsvariable
2) RootDirectory Parameter in firebird.conf
3) Standardinstallationspfad (/opt/firebird)

Linux Superserver:
1) FIREBIRD Umgebungsvariable
2) RootDirectory Parameter in firebird.conf
3) Das Verzeichnis eine Ebene über dem Verzeichnis, in dem sich die ausführbaren Dateien des Servers befinden (via symlink "/proc/self/exe", falls unterstützt)
4) Standardinstallationspfad (/opt/firebird)

Parameter

Die meisten Parameter besitzen einen Standardwert. Unter Linux ist auf die Groß-/Kleinschreibung der Parameternamen zu achten. Dies ist bei Windows nicht erforderlich. Um einen Parameterwert auf einen, dem Vorgabewert unterschiedlichen Wert, zu setzen, entfernen Sie den Kommentarmarker (#) und ändern Sie den Wert. Sie können die Konfigurationsdatei während des Betriebs des Servers ändern. Geänderte Parameterwerten werden aber erst nach einem Neustart des Firebird Servers wirksam.
Einträge sind in der Form:

Parametername *Wert*

- Parametername ist eine Zeichenkette ohne Leerzeichen, und bezeichnet eine Eigenschaft des zu konfigurierenden Servers.

- *Wert* ist eine Zahl, Boolean (1=True, 0=False) oder eine Zeichenkette, die den Wert des Parameters spezifiziert.

Dateisystemspezifische Parameter

RootDirectory

Zeichenkette; der absolute Pfad zu einem Verzeichnis im lokalen Dateisystem. Sofern das Standardverhalten des Servers zur Ermittlung des Stammverzeichnis der Firebird Server Installation nicht überschrieben werden soll, sollte dieser Parameter auskommentiert bleiben.

DatabaseAccess

Unterstützt das Datenbank-Aliasing Feature. In früheren Versionen konnte der Server eine Verbindung zu einer beliebigen Datenbank im lokalen Dateisystem herstellen, und der Clientanwendung war der Zugriff zu dieser Datenbank durch Verwendung des absoluten Dateisystempfades aus Sicht des Servers möglich. Dieser Parameter stellt eine Option zur Verfügung, um dem Server nur den Zugriff auf Datenbanken zu gewähren, die durch einen Datenbank-Alias definiert sind, oder um den Zugriff auf Datenbanken in bestimmten Dateisystemverzeichnisbäumen einzuschränken.
DatabaseAccess kann None, Restrict oder Full sein.
Full (Standardwert) erlaubt den Zugriff auf eine beliebige Datenbank im lokalen Dateisystem.
None erlaubt dem Server nur über Datenbank-Aliase, die in **aliases.conf** definiert sein müssen, um auf eine Datenbank zugreifen zu können.
Restrict erlaubt Ihnen die Angabe einer Liste von Verzeichnissen, in denen sich zugreifbare Datenbankdateien befinden. Definieren Sie eine Liste von ein oder mehreren Verzeichnissen, durch Strichpunkte (;) getrennt, um erlaubte Zugriffsorte zu definieren.
Zum Beispiel,
Unix: `/db/databases;/userdir/data`
Windows: `D:\data`
Relative Pfade werden relativ zum Pfad des Stammverzeichnis des laufenden Servers betrachtet. Wenn zum Beispiel unter Windows das Stammverzeichnis C:\Program Files\Firebird lautet, dann bewirkt der folgende Wert, dass der Zugriff des Servers auf Datenbankdateien eingeschränkt wird, die sich im Verzeichnis C:\Program Files\Firebird\userdata befinden:

```
DatabaseAccess = Restrict userdata
```

HINWEIS: Datenbank-Shadowing – Die aktuelle DatabaseAccess Logik hat einen Bug, der zur Folge hat, dass die Restrict Option verwendet werden muss, falls Sie für irgendeine Datenbank am Server Shadowing einsetzen.

ExternalFileAccess

Vormals *external_file_directory* in isc_config/ibconfig. Es hat sich jedoch die Syntax geändert.

Stellt drei Sicherheitsebenen bezüglich EXTERNAL FILES (Textdateien mit fixer Datensatzlänge, die als Datenbanktabellen verwendet werden können) zur Verfügung. Der Wert ist eine Zeichenkette, welche None, Full oder Restrict sein kann.
None (Standardwert) deaktiviert jegliche Verwendung von externen Dateien am Server.
Restrict erlaubt den Zugriff auf externe Dateien für bestimmte Verzeichnisse einzuschränken. Definieren Sie eine Liste von ein oder mehreren Verzeichnissen, durch Strichpunkte (;) getrennt, in denen externe Dateien gespeichert sein können.
Zum Beispiel,
Unix: `/db/extern;/mnt/extern`
Windows: `C:\ExternalTables`
Relative Pfade werden relativ zum Pfad des Stammverzeichnis des laufenden Servers betrachtet. Wenn zum Beispiel unter Windows das Stammverzeichnis C:\Program Files\Firebird lautet, dann bewirkt der folgende Wert, dass der Zugriff des Servers auf externe Dateien eingeschränkt wird, die sich im Verzeichnis C:\Program Files\Firebird\userdata\ExternalModules befinden:

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

Full erlaubt den Zugriff auf beliebige externe Dateien am Server.

Beachten Sie den **HINWEIS** am Ende des nächsten Abschnitts, UdfAccess.

UdfAccess

Vormals *external_function_directory* in isc_config/ibconfig. Es hat sich jedoch die Syntax geändert.

Ersetzt nicht nur den Namen des früheren Parameters, sondern auch die Form, in denen Werte angegeben werden können. Ziel dieser Änderung waren optionale Ebenen des Schutzes für benutzerdefinierte Bibliotheken zu integrieren, da diese ein bekanntes Ziel für böswillige Attacken von außen darstellen. UdfAccess kann None, Restrict oder Full sein.

Restrict (Standardwert) besitzt die selbe Funktionalität wie der **external_function_directory** Parameter in Firebird 1.0, um den Ort von ausführbaren externen Bibliotheken auf bestimmte Verzeichnisse einzuschränken. Definieren Sie eine Liste von ein oder mehreren Verzeichnissen, durch Strichpunkte (;) getrennt, in denen UDFs, BLOB Filter und Zeichensatzdefinitionen gespeichert sein können.
Zum Beispiel,
Unix: `/db/extern;/mnt/extern`
Windows: `C:\ExternalModules`
Relative Pfade werden relativ zum Pfad des Stammverzeichnis des laufenden Servers betrachtet. Wenn zum Beispiel unter Windows das Stammverzeichnis C:\Program Files\Firebird lautet, dann bewirkt der folgende Wert, dass der Zugriff des Servers auf externe Bibliotheken eingeschränkt wird, die sich im Verzeichnis C:\Program Files\Firebird\userdata\ExternalModules befinden:

```
UdfAccess = Restrict userdata\ExternalModules
```

None unterbindet jegliche Verwendung von benutzerdefinierten, externen Bibliotheken.

Full erlaubt den Zugriff auf beliebige externe Bibliotheken am Server.

HINWEIS : Vermeiden Sie die Verwendung von Verzeichnissen für UdfAccess und ExternalFileAccess, die ein gemeinsames Stammverzeichnis besitzen. Die Standardwerte sind sicher. Wenn Sie nun Ihre eigenen Einstellungen vornehmen und diese keine separaten Verzeichnisbäume verwenden, dann kann der Server auf einfache Art und Weise gehackt werden, um unbefugten Code auszuführen. Ein Beispiel soll zeigen, was zu vermeiden ist:

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

UdfAccess & ExternalFileAccess haben hier ein gemeinsames Unterverzeichnis `/bad_dir/files`, in dem jemand eine externe Bibliothek `/bad_dir/files/hackudf.so` ablegen könnte, um dessen eigenen Code am gefährdeten System auszuführen.

Ressourcenspezifische Parameter

CpuAffinityMask

Vormals *cpu_affinity* in isc_config/ibconfig

Mit Firebird SuperServer unter Windows existiert das Problem, dass auf SMP Maschinen der komplette SuperServer Prozess zwischen den Prozessoren hin- und hergereicht wird. Dies zerstört die Performance. Dieser Parameter kann unter Windows dazu verwendet werden, um den Firebird SuperServer Prozess, durch setzen der Prozessoraffinität, an eine CPU zu binden.

HINWEIS: Firebird Superserver, bis und einschließlich Release 1.5 unterstützt nicht die Hyperthreading Technologie von neueren Motherboards unter Windows. Um Performanceprobleme zu vermeiden, sollten Sie Hyperthreading im BIOS komplett deaktivieren.

CpuAffinityMask erwartet einen Ganzzahlenwert, die CPU Mask.

Beispiel

```
CpuAffinityMask = 1
```
Läuft nur auf der ersten CPU (CPU 0).

```
CpuAffinityMask = 2
```
Läuft nur auf der zweiten CPU (CPU 1).

```
CpuAffinityMask = 3
```
Läuft auf beiden, der ersten und der zweiten CPU.

Berechnung des Wertes für die Affinity Mask
Sie können dieses Flag verwenden, um Firebird's Affinität auf einen Prozessor oder (für Classic Server) auf eine Kombination der im System installierten CPUs zu setzen.
Betrachten Sie die CPUs als Array von 0 bis *n*-1, wo *n* die Anzahl der installierten CPUs und *i* den Array-index einer CPU definiert. M ist ein weiterer Array, der den Mask-Wert für jede ausgewählte CPU beinhaltet. Der Wert A ist die Summe der Werte in M.

Unter Verwendung der folgenden Formel lässt sich M und der Mask-Wert A wie folgt berechnen:

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

Zum Beispiel, um den ersten und den vierten Prozessor (Prozessor 0 und Prozessor 3) auszuwählen, lässt sich der Mask-Wert A wie folgt berechnen:

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

DeadlockTimeout

Vormals *deadlock_timeout* in isc_config/ibconfig

Anzahl von Sekunden (Integer) die der Lock Manager nach Auftreten eines Konflikts wartet, bevor Sperren von "toten" Prozessen entfernt werden und ein weiterer Deadlock-Scanzyklus gestartet wird. Normalerweise werden Deadlocks von der Engine sofort erkannt. D.h. dieser Parameter tritt nur dann in Kraft, wenn etwas schief gelaufen ist.

Der Standardwert von 10 Sekunden ist in der Regel ein guter Wert. Eine Verringerung dieses Wertes bedeutet nicht notwendigerweise einen Geschwindigkeitsgewinn, damit problematische Deadlocks eine entsprechende Deadlock-Meldung zurückliefern. Ist dieser Wert zu niedrig, so kann der Effekt von unnötigen Deadlock-Scans auftreten, der die Systemperformance negativ beeinträchtigt.

DefaultDbCachePages

Vormals *database_cache_pages* in isc_config/ibconfig

Serverweiter Standardwert (Integer) für die Anzahl der im Speicher zu allokierenden Datenbankseiten pro Datenbank. Der definierte Wert kann auf Datenbankebene überschrieben werden.
Der Standardwert für SuperServer ist 2048 Datenseiten. Für Classic ist dies 75.
SuperServer und Classic verwenden den Cache unterschiedlich. Superserver verwaltet den Cache für die Verwendung aller Verbindungen; Classic allokiert einen statischen Cache für jede Verbindung.

EventMemSize

Integer, repräsentiert die Anzahl der Bytes im Speicher, die für den Event Manager reserviert werden. Standardwert ist 65536 (64 Kb).

LockAcquireSpins

Vormals *lock_acquire_spins* in isc_config/ibconfig

Nur für Classic Server auf SMP Maschinen relevant. Bei Classic Server darf nur ein Clientprozess zu einem Zeitpunkt auf die Sperrtabelle zugreifen. Ein Mutex regelt den Zugriff auf diese Sperrtabelle. Clientprozesse dürfen den Mutex bedingt oder unbedingt anfordern. Falls dies bedingt erfolgt, so schlägt die Anforderung fehl und muss wiederholt werden. Falls dies unbedingt erfolgt, so wartet die Anforderung, bis diese zufriedengestellt wurde. LockAcquireSpins definiert die Anzahl der Versuche, die gemacht werden, falls die Mutex-Anforderung bedingt erfolgt.
Integer. Der Standardwert ist 0 (unbedingt). Es existiert kein empfohlener minimaler oder maximaler Wert.

LockHashSlots

Vormals *lock_hash_slots* in isc_config/ibconfig

Verwenden Sie diesen Parameter um die Lock-Hash-Liste zu tunen. Unter schwerer Serverlast kann der Durchsatz durch Erhöhen der Anzahl der Hash-Slots, um die Hash-Liste in kleinere Hash-Ketten zu zerlegen, erhöht werden. Integer, Primzahlen werden empfohlen. Der Standardwert ist 101.

LockGrantOrder

Vormals *lock_grant_order* in isc_config/ibconfig

Wenn eine Verbindung ein Objekt sperren möchte, so bekommt diese einen Lock-Request-Block, der das angeforderte Objekt und die Sperrebene (lock level) spezifiziert. Jedes gesperrte Objekt besitzt einen Sperrblock (lock block). Anforderungsblöcke sind mit denjenigen Sperrblöcken, entweder als Anfragen, die zugelassen wurden oder als bevorstehende Anfragen, verbunden.
Der LockGrantOrder Parameter ist ein Boolean. Der Standardwert (1=True) gibt an, dass Sperren nach dem First-Come-First-Serve Prinzip zugelassen werden. False (0) emuliert das Verhalten von InterBase v3.3, nämlich das Zulassen von Sperren ab dem Zeitpunkt zu dem diese verfügbar werden. Dies kann zur Folge haben, dass Sperranforderungen "verhungern".

LockMemSize

Dieser Integer Parameter gibt an, wie viele Bytes im Shared Memory, dem Lock-Manager zur Verfügung stehen sollen. Für einen Classic Server definiert LockMemSize den Initialwert für die Allokierung und wächst, bis der Speicher aufgebraucht ist (*"Lock manager is out of room"*). Wenn viele Verbindungen oder große Page Caches existieren, dann erhöhen Sie den Wert dieses Parameters, um diesen Fehler zu vermeiden.

In SuperServer wächst der für den Lock-Manager allokierte Speicher nicht.
Der Standardwert unter Linux und Solaris ist 98304 Bytes (96 Kb). Unter Windows ist dies 262144 (256 Kb).

LockSemCount

Dieser Integer Parameter definiert die Anzahl der verfügbaren Semaphoren für die Interprozess-kommunikation (interprocess communication; IPC). Der Standardwert ist 32. Setzen Sie diesen Parameter in Non-Threading Umgebungen, um die Anzahl der verfügbaren Semaphoren zu vergrößern oder zu verkleinern.

SortMemBlockSize

Dieser Parameter gibt die Größe jedes Speicherblocks in Bytes an, welcher vom In-Memory-Sorting-Module verwendet wird. Der Standardwert ist 1 Mb; sie können diesen Parameter beliebig setzen, jedoch auf einen Maximalwert, der durch den SortMemUpperLimit Parameter definiert ist (siehe darunter).

SortMemUpperLimit

Der maximale Speicher in Bytes, der vom In-Memory-Sorting-Module allokiert werden kann. Der Standardwert ist 67108864 Bytes (64 Mb) für SuperServer und 8388608 Bytes (8 Mb) für Classic Server.

> HINWEIS: Beachten Sie für Classic, dass eine Erhöhung des Speicherblocks oder des maximalen Speichers, jede Clientverbindung/Server Instanz beeinflusst, und entsprechend mehr Speicher vom Server verbraucht wird.

Kommunikationsspezifische Parameter

ConnectionTimeout

Vormals *connection_timeout* in isc_config/ibconfig

Anzahl der Sekunden, die gewartet werden, bevor ein Verbindungsversuch abgebrochen wird. Der Standardwert ist 180.

DummyPacketInterval

Vormals *dummy_packet_interval* in isc_config/ibconfig

Anzahl der Sekunden die der Server wartet, bevor an eine unbenutzte (ruhige) Clientverbindung Dummypakete versendet werden, um eine Rückantwort (Lebenszeichen) anzufordern.
VERWENDEN SIE DIESE OPTION NICHT für einen Win32 Server mit TCP/IP Clients. Dies bewirkt einen stetigen Anstieg von Kernel-Non-Paged-Memory, der zur Folge haben kann, dass Windows am Client hängen bleibt oder abstürzt, so wie es hier beschrieben ist:
http://support.microsoft.com/default.aspx?kbid=296265

Unabhängig von Win32 mit TCP/IP ist dies der einzige Weg um inaktive Clients zu erkennen und die Verbindung zu beenden, wenn entweder NamedPipes (NetBEUI), XNET oder IPC Protokolle verwendet werden. Es sind keine Probleme für POSIX Systeme bekannt.

Normalerweise verwendet Firebird die SO_KEEPALIVE Socket-Option um aktive Verbindungen zu überwachen. Wenn Sie das 2 Stunden Keepalive-Timeout nicht verwenden wollen, dann ändern Sie entsprechend die Servereinstellungen auf Betriebssystemebene:

- Unter UNIX-verwandten Betriebssystemen, ändern Sie den Inhalt von /proc/sys/net/ipv4/tcp_keepalive_*.
- Unter Windows, folgen Sie den Schritten im folgenden Artikel:

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

Der Standardwert sollte 0 und nicht 60 sein, wie dies der Fall für Firebird 1.0 und den meisten 1.5 Release Candidates war. Ein Wert von 60 sollte als Standardwert angesehen werden, wenn Systeme dieses Dummy-Packet-Polling verwenden sollen.

RemoteServiceName

Standardwert = gds_db

RemoteServicePort

Diese beiden Parameter ermöglichen das Überschreiben entweder des TCP/IP Servicenamen oder der TCP/IP Portnummer, die der Server verwendet, um Datenbankverbindungen entgegenzunehmen, wenn einer dieser Parameter von den Standardwerten (gds_db/tcp 3050) abweicht.

Ändern Sie einen dieser Parameter, aber nicht beide. RemoteServiceName wird zuerst mit einem übereinstimmenden Eintrag in der services Datei überprüft. Existiert eine Übereinstimmung, so wird die Portnummer für RemoteServicePort verwendet. Gibt es hier keine Übereinstimmung, dann wird der Standardport 3050 verwendet.

BEACHTE: Wird eine Portnummer in einem TCP/IP Verbindungsstring verwendet, so hat dieser Port immer Vorrang gegenüber dem Parameterwert von RemoteServicePort.

RemoteAuxPort

Das von InterBase geerbte Verhalten, dass Event-Benachrichtigungen über zufällige TCP/IP Ports zurück zum Client geschickt werden, war immer eine mögliche Quelle von Netzwerkproblemen und Konflikten mit Firewalls. Unter Umständen war es sogar möglich den Server zum Absturz zu bringen. Dieser Parameter erlaubt Ihnen einen TCP Port, der für jegliche Event-Kommunikation verwendet wird, zu konfigurieren.
Der Standardwert (0) beschreibt das alte Verhalten, also die Verwendung von zufälligen Ports.
Verwenden Sie einen ganzzahligen Wert, der einen frei verfügbaren Port spezifiziert, damit dieser Port für die Event-Kommunikation verwendet wird.

RemoteBindAddress

Per Default können sich Clients über ein beliebiges, vom Server akzeptiertes, Netzwerkinterface verbinden. Dieser Parameter erlaubt es Ihnen den Firebird Service für ankommende Anfragen an eine Netzwerkkarte (Network Interface Card, NIC) zu binden und Verbindungsversuche über andere Netzwerkkarten abzulehnen. Dies sollte hilfreich sein um Probleme zu vermeiden, wenn der Server Anfragen von Subnetzwerken über mehrere Netzwerkkarten abhandelen muss.
Der Wert muss eine gültige IP-Adresse sein. Es existiert kein Standardwert (nicht gebunden).

TcpRemoteBufferSize

Die Engine besitzt die Möglichkeit mehrere Datensätze "vorausschauend" zu lesen und in einem Paket an den Client zu senden. Umso größer die Paketgröße, desto mehr Datensätze können pro Paket an den Client übermittelt werden. Wenn Sie die TCP/IP Paketgröße vergrößern oder verkleinern möchten, dann verwenden Sie diesen Parameter mit Sorgfalt, und nur unter Kenntnis der möglichen Auswirkungen auf die Netzwerkperformance! Dieser Parameter hat Auswirkungen sowohl auf den Client, als auch auf den Server.
Der Wert ist ein Integer (Paketgröße in Bytes) im Bereich von 1448 bis 32768.
Der Standardwert ist 8192.

POSIX-spezifische Parameter

LockSignal

Integer Parameter, UNIX Signal für die Verwendung bei der Interprozesskommunikation.
Standardwert: 16

RemoteFileOpenAbility

VERWENDUNG MIT HÖCHSTER SORGFALT

Boolescher Parameter, falls True, dann wird der Engine erlaubt Datenbankdateien zu öffnen, die sich auf einer NFS gemappten Partition befinden. Da in diesem Fall das Dateisystem nicht unter der Kontrolle des lokalen Systems (Server) steht, ist dieser Parameter mit höchster Sorgfalt zu verwenden,

und sollte auf keinen Fall aktiviert werden, wenn sich damit unternehmenskritische Lese/Schreib-Datenbanken öffnen lassen.
Der Standardwert ist 0 (False, Deaktiviert) und Sie sollten diesen Parameter so belassen, solange Sie sich nicht im klaren sind, welche möglichen Effekte dies nach sich ziehen könnte.

TcpNoNagle

Vormals *tcp_no_nagle* in isc_config/ibconfig

Unter Linux minimiert die Socket-Bibliothek per Default physische Schreiboperationen durch eine Zwischenspeicherung (Buffering) dieser Operationen, bevor die Daten tatsächlich gesendet werden. Dies geschieht unter Verwendung eines internen Algorithmus (implementiert als TCP_NODELAY Option der Socketverbindung), auch als Nagle's Algorithmus bekannt, um Probleme mit kleinen Datenpaketen - genannt Tinygrams – bei langsamen Netzwerken zu vermeiden.
Per Default ist TCP_NODELAY aktiviert (Wert 0), falls Firebird SuperServer unter Linux installiert ist. Für langsame Netzwerke, kann das Deaktivieren einen Performancegewinn zur Folge haben. Setzen Sie den Parameter auf True (1) um TCP_NODELAY zu deaktivieren, und auf False (0) um dies zu aktivieren.
In Releases bis einschließlich v.1.5 ist dieses Feature nur für SuperServer aktiviert.

Windows-spezifische Parameter

CreateInternalWindow

Das "lokale Windows" Protokoll verwendet für die Interprozesskommunikation zwischen dem Client und dem Server ein verstecktes Fenster. Dieses IPC Fenster wird beim Server-Start erzeugt, falls CreateInternalWindow True (1, Standardwert) ist. Setzen Sie diesen Parameter auf 0 (Deaktiviert), um den Server ohne diesem Fenster zu betreiben, und somit das lokale Protokoll zu deaktivieren. Ein deaktiviertes lokales Protokoll erlaubt den Betrieb von mehreren gleichzeitigen Serverinstanzen.

DeadThreadsCollection

Eine Einstellung für den Thread-Scheduler unter Windows. Dieser Integer Parameter definiert die Anzahl der Priority-Switch-Zyklen (siehe auch unten PrioritySwitchDelay), die der Scheduler durchläuft, bis ein Thread zerstört (oder beendet) wird.
Eine unmittelbare Zerstörung (oder Beendigung) von Worker Threads würde eine Semaphore und einen blockierenden Aufruf benötigen, das einen signifikanten Overhead zur Folge haben würde. Darum verwaltet der Thread-Scheduler einen Pool von Threads. Wenn ein Thread seine Aufgabe erledigt hat, wird dieser als Idle (untätig) markiert. Ein Thread im Status Idle wird nach *n* Durchläufen der Scheduler-Schleife zerstört (oder beendet), wobei *n* für den Wert des Parameters DeadThreadsCollection steht.
Falls der Server eine große Anzahl von Verbindungen (einige hundert) verarbeiten muss, dann sollte der Standardwert von 50 erhöht werden.

GuardianOption

Boolescher Parameter, verwendet von einem Windows Server, um anzugeben, ob der Guardian nach einem unvorhergesehen Beenden des Servers diesen automatisch wieder starten soll. Der Standardwert ist 1 (True), also der automatische Neustart im Fehlerfall. Um dies zu deaktivieren, setzen Sie diesen Parameter auf 0 (False).

IpcMapSize

Vormals *server_client_mapping* in ibconfig

Größe des Anteils für einen Client, an der im Speicher abgebildeten Datei für die Interprozesskommunikation (IPC) bei einer "lokalen Windows" Verbindung in Bytes. Es gibt dafür keine Entsprechung unter anderen Plattformen. Ein Integer Wert im Bereich von 1024 bis 8192. Der Standardwert ist 4096.

Das Erhöhen der Map-Size kann die Performance bei der Verarbeitung von langen oder großen Datensätzen (z.B. wenn diese BLOBs zurückgeben) verbessern.

> HINWEIS: Dies kann nicht mehr über das Guardian Symbol im System Tray Dialog geändert werden.

IpcName

Standardwert: FirebirdIPI
Der Name des Shared-Memory-Bereichs, der als Transportkanal für das lokale Protokoll verwendet wird. Der Standardwert —FirebirdIPI— ist nicht kompatibel zu älteren Releases von Firebird und auch nicht zu InterBase®. Falls notwendig, verwenden Sie InterBaseIPI um die Kompatibilität wiederherzustellen.

MaxUnflushedWrites

Dieser Parameter wurde in Version 1.5 eingeführt, um einen Bug im Windows Betriebssystem zu umgehen, der verhinderte, dass asynchrone Schreiboperationen niemals auf die Festplatte zurückgeschrieben werden, ausgenommen, wenn der Firebird Server kontrolliert heruntergefahren wird. (Asynchrone Schreiboperationen werden unter Windows 9x oder ME nicht unterstützt). Somit würden auf 24/7 Systemen asynchrone Schreiboperationen niemals auf die Festplatte geschrieben werden.
Dieser Parameter definiert, wie oft zurückgehaltene Datenseiten auf die Festplatte geschrieben werden, falls Forced Writes deaktiviert (asynchrone Schreiboperationen sind aktiviert) ist. Der Wert ist ein Integer, der die Anzahl der Datenseiten definiert, die gepuffert werden, bevor ein Zurückschreiben beim nächsten Commit einer Transaktion durchgeführt wird. Der Standardwert ist 100 für Windows-Installationen und –1 (deaktiviert) für Installationen auf allen anderen Plattformen.
Falls das Ende des MaxUnflushedWriteTime Zyklus (siehe unten) erreicht wird, bevor die Anzahl der noch nicht geschriebenen Datenseiten den Wert von MaxUnflushedWrites erreicht haben, wird das Zurückschreiben auf die Festplatte unmittelbar durchgeführt, und die Anzahl der gepufferten Datenseiten auf 0 zurückgesetzt.

MaxUnflushedWriteTime

Dieser Parameter definiert die maximale Dauer, in deren Datenseiten bei asynchronen Schreiboperationen gepuffert werden, bevor diese auf die Festplatte zurückgeschrieben werden, falls Forced Writes deaktiviert (asynchrone Schreiboperationen sind aktiviert) ist. Der Wert ist ein Integer, der das Intervall - in Sekunden – definiert, bevor ein Zurückschreiben beim nächsten Commit einer Transaktion durchgeführt wird. Der Standardwert ist 5 Sekunden für Windows-Installationen und –1 (deaktiviert) für Installationen auf allen anderen Plattformen.

PrioritySwitchDelay

Eine Einstellung für den Thread-Scheduler unter Windows. Dieser Integer Parameter definiert die Zeit in Millisekunden, die vergeht, bevor die Priorität eines inaktiven Threads auf LOW verringert oder die Priorität eines aktiven Threads auf HIGH erhöht wird. Ein Durchlauf dieses Umschaltvorgangs repräsentiert einen Thread-Scheduler-Zyklus.
Der Standardwert ist 100ms, gewählt auf Basis von Versuchen auf einem Intel PIII/P4 Prozessor. Für Prozessoren mit einer geringeren Taktfrequenz ist ein längeres Zeitintervall notwendig.

PriorityBoost

Integer, setzt die Anzahl der zusätzlichen Zyklen die ein Thread erhält, wenn dessen Priorität auf HIGH umgeschaltet wird. Der Standardwert ist 5.

ProcessPriorityLevel

Vormals *server_priority_class* in ibconfig
Prioritäts-Level/Klasse für den Serverprozess. Dieser Parameter ersetzt den server_priority_class Parameter von Releases < 1.5, inklusive einer neuen Implementierung.

Die Werte sind Integer, wie folgt:

- 0 – Normale Priorität,
- Positiver Wert – Hohe Priorität (gleich wie der -B[oostPriority] Schalter für instsvc.exe *configure* und *start* Optionen)
- Negativer Wert – Niedrige Priorität.

Hinweis: Alle Änderungen dieses Wertes sollten sorgfältig getestet werden, um sicherzustellen, dass die Engine auf Anfragen besser reagiert.

RemotePipeName

Nur für NetBEUI Verbindungen anwendbar

String Parameter, der Name der Pipe, die als Transportkanal für das NetBEUI Protokoll verwendet wird. Eine "Named Pipe" ist gleichzusetzen mit einer Portnummer für TCP/IP. Der Standardwert interbas ist zu älteren Firebird und InterBase® Releases kompatibel.

Parameter zur Konfiguration des temporären Sortierungsspeichers

Wenn die Größe des internen Sortierpuffers zu klein wird um die Datensätze unterzubringen, die in einer Sortieroperation vorkommen, so muss Firebird temporäre Dateien im Dateisystem des Servers erzeugen. Firebird verwendet per Default das durch die Umgebungsvariable **FIREBIRD_TMP** spezifizierte Verzeichnis. Wenn diese Umgebungsvariable nicht vorgefunden wird, dann wird versucht, das Stammverzeichnis von **/tmp** unter Linux/UNIX, oder **C:\temp** unter Windows NT/2000/XP zu verwenden. Für keine dieser Verzeichnisse kann die Größe festgelegt werden.

Firebird stellt einen Parameter zur Verfügung, um den Festplattenspeicher zur Speicherung dieser temporären Dateien zu konfigurieren. Sie sollten sicherstellen, dass immer ausreichender Speicherplatz für temporäre Dateien vorhanden ist.

Alle CONNECT oder CREATE DATABASE Anweisungen teilen sich die Liste der Verzeichnisse für temporäre Dateien, und jede erzeugt dessen eigenen temporären Dateien. Sortierdateien werden wieder freigegeben, wenn die Sortieroperation oder die Anfrage selbst beendet ist.

In Release 1.5 wurde der Parametername von **tmp_directory** auf **TempDirectories** geändert. Die Syntax des Parameterwertes hat sich ebenfalls geändert.

TempDirectories

Ersetzt *tmp_directory* Einträge in isc_config/ibconfig

Spezifizieren Sie eine Liste von ein oder mehreren Verzeichnissen, durch einen Strichpunkt (;) voneinander getrennt, unter denen Sortierdateien gespeichert werden dürfen. Jeder Eintrag kann ein optionales Argument für die Größe (in Bytes) besitzen, um dessen Speichergröße zu limitieren. Ist dieses Argument nicht angegeben oder fehlerhaft, dann verwendet Firebird den komplett verfügbaren Speicherplatz in diesem Verzeichnis, bevor zum nächsten Verzeichnis in der Liste übergegangen wird.

Zum Beispiel,

Unix: `/db/sortfiles1 100000000;/firebird/sortfiles2`

Windows: `E:\sortfiles 500000000`

Relative Pfade werden relativ zum Pfad des Stammverzeichnis des laufenden Servers betrachtet. Wenn zum Beispiel unter Windows das Stammverzeichnis C:\Program Files\Firebird lautet, dann bewirkt der folgende Wert, dass der Server temporäre Dateien bis zu einer Größe von 500 Mb in C:\Program Files\Firebird\userdata\sortfiles speichert:

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

Die Pfade dürfen nicht durch Hochkommata eingeschlossen werden, wie dies in Firebird 1.0 noch der Fall war.

Kompatibilitätsparameter

CompleteBooleanEvaluation

Definiert die Methode zur booleschen Auswertung (Komplett oder Kurzschluss). Der Standardwert (0=False) steht für die Kurzschlussauswertung eines booleschen Ausdrucks, der AND oder OR Operatoren beinhalten kann, und die Auswertung mit der Rückgage eines booleschen Ergebnisses abbricht, sofern die Evaluierung weiterer Ausdrücke das Ergebnis nicht mehr beeinflussen kann.
Unter sehr seltenen (gewöhnlich vermeidbaren) Umständen kann es jedoch passieren, dass eine Operation inmitten einer OR oder einer AND Bedingung, die aufgrund der Kurzschlussauswertung nicht ausgewertet wird, obwohl diese das Ergebnis beeinflussen könnte. Sollten Sie in der Lage sein, dass eine Ihrer Anwendungen diese Charakteristika in deren SQL Logik aufweist, dann können Sie diesen Parameter verwenden, um die komplette boolesche Auswertung zu verwenden. Der Parametertyp ist ein Boolean.

> Übersehen Sie nicht, dass dieser Parameter jede boolesche Auswertung, für jede Datenbank am Server, beeinflusst.

OldParameterOrdering

Version 1.5 behob einen alten InterBase Bug der zur Folge hatte, dass Rückgabeparameter in nicht konsistenter Reihenfolge in der XSQLDA Struktur an den Client zurückgegeben werden konnten. Dieser Bug hatte eine derartige Langlebigkeit, dass viele existierende Anwendungen, Treiber und Zugriffskomponenten einen eingebauten Workaround enthalten, um dieses Problem clientseitig zu korrigieren.
Version 1.5 und später besitzen diesbezüglich einen Bugfix in der API, und werden mit OldParameterOrdering=0 (False) installiert. Setzen Sie diesen booleschen Parameter auf 1 (True), um das alte Verhalten zu verwenden, damit die Kompatibilität zu existierendem Code gewährleistet ist.

Datenbankdatei-Aliasing

Firebird 1.5 bietet nun ein neues Feature "Datenbankdatei-Aliasing" an, um die Portabilität von Anwendungen, und um die Kontrolle des internen Zugriffs und den Zugriff auf externe Datenbankdateien zu verbessern.

Aliases.conf

Konfigurieren Sie Datenbankdateialiase in einer Textdatei aliases.conf im Stammverzeichnis Ihrer Firebird Server Installation. Die vorinstallierte aliases.conf Datei sieht in etwa wie folgt aus:

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

Wie in jeder Konfigurationsdatei von Firebird, ist das "#" Symbol der Beginn eines Kommentars. Um einen Alias zu konfigurieren, entfernen Sie einfach das "#" Symbol und ändern Sie einfach die Zeile auf einen gültigen Datenbankpfad:

```
# fbdb1 is on a Windows server:
fbdb1 = c:\Firebird\sample\Employee.fdb
# fbdb2 is on a Linux server
fbdb2 = /opt/databases/killergames.fdb
#
```

Sie können während des Betriebs des Servers die aliases.conf Datei modifizieren. Es ist nicht notwendig den Server zu stoppen und zu starten, damit neue Einträge in aliases.conf berücksichtigt werden.

Unter Verwendung eines Alias zur Datenbank verbinden

Der modifizierte Verbindungsstring in Ihrer Clientanwendung sollte folgendes Aussehen haben:

```
Server_name:aliasname
```

Der folgende Verbindungsstring bewirkt, dass versucht wird sich zu einer Datenbank am Server "myserver" zu verbinden, wobei der Pfad zur Datenbank durch einen Alias "fbdb2" in aliases.conf definiert ist.:

```
myserver:fbdb2
```

Hinweis: Da das **gstat Tool** keine herkömmliche Datenbankverbindung verwendet um die Datenbankdatei zu lesen, ist nach wie vor ein vollständiger Pfad für die Verwendung von gstat notwendig. (Dies könnte sich in Zukunft ändern)

Benennung von Datenbanken unter Windows

Beachte Sie, dass ".fdb" nun die vorgeschlagene Dateierweiterung für eine Datenbankdatei unter Windows ME und XP ist, um mögliche Konflikte mit dem "System Restore" Feature von Windows zu vermeiden. Bleibt dies unbeachtet, so führt dies zu dem bekannten Problem von erhöhten Verbindungszeiten, sofern primäre und/oder sekundäre Datenbankdateien die bekannte ".gdb" Dateierweiterung verwenden.

Firebird Entwicklungs- Teams

Entwickler	Land	Hauptaufgaben
Dmitry Yemanov	Russische Föderation	Release Koordinator; DSQL und PSQL Erweiterungen; Implementierung des "Embedded Server", zahlreiche Metadaten-Erweiterungen, Serverseitige Datenbank-Aliase, Multi-Aktions-Trigger, BigInt Datentyp, neue Kontextvariablen, Windows Classic Server; zahlreiche Fehlerkorrekturen
Nickolay Samofatov	Russische Föderation	Implementierung und Entwicklung neuer SQL Features (Savepoints, Pessimistisches Locking); Metadaten Erweiterungen; Redesign von Teilen der Engine; Fehlersuche und Behebung; Verbesserung der Software-Architektur; Implementierung von Teilen der Service-Api für den Linux Classic-Server; Performanceverbesserungen; Linux Classic Builds
Arno Brinkman	Niederlande	Verbesserung des Optimierers; viele neue DSQL Erweiterungen
Claudio Valderrama	Chile	Kontrolle der Codeänderungen; Fehlersuche und Behebung; PSQL Erweiterungen; Fehlerbereinigung der UDF-Bibliothek, Design und Implementierung
Alex Peshkoff	Russische Föderation	PSQL und DSQL Erweiterungen; Koordinator und Autor für sicherheitsrelevante Änderungen; Fehlerbehebung; Linux Superserver Builds
Mike Nordell	Schweden	Konvertierung des Firebird Sourcecodes nach C++; Performanceverbesserungen; Portierung; Fehlersuche und Behebung
Blas Rodriguez Somoza	Spanien	Entwicklung neuer Zeichensätze; Weitgehende Modernisierung und Verbesserung des Source-Codes, Reorganisierung des Source-Verzeichnisses; MinGW Builds
Roman Rokytskyy	Deutschland	Entwicklung und Koordinierung von JayBird
David Jencks	U.S.A.	Entwicklung und Koordinierung von JayBird; Entwicklung der Firebird-Dokumentations-Tools
Carlos Guzman Alvarez	Spanien	Entwicklung und Koordinierung des .NET-Providers für Firebird
John Bellardo	U.S.A.	Entwicklung eines Plug-in Interfaces für Zeichensätze; Koordination, Darwin Builds; Entwurf und Implementierung eines neuen Speichermodells
Erik Kunze	Deutschland	Fehlersuche und Behebung; Verbesserung des Source-Codes; SINIX-Z Builds
Dmitry Sibiryakov	Russische Föderation	Bereinigung des Source-Codes ; MinGW Builds
Pavel Cisar	Tschechien	Linux Builds (Release 1.0); Entwicklung und Koordination von Qualitätstests

Entwickler	Land	Hauptaufgaben
Ann Harrison	U.S.A.	Fehlerbeseitigung; Mentor; Erhöhung der maximalen Indexanzahl
Mark O'Donohue	Australien	Einbindung der readline-Bibliothek in isql; Fehlerbeseitigung; Boot Builds (Release 1.0); Bereinigung des Source-Codes (Release 1.0)
Paul Reeves	Frankreich	Qualitätstests; Win32 Installationen; Standard Win32 Systemsteuerungsanwendung
Ignacio J. Ortega	Spanien	Explizite PLAN-Anweisung in Triggern; Bereinigung des Source-Codes
Konstantin Kuznetsov	Russische Föderation	Solaris Intel Builds
Olivier Mascia	Belgien	Reimplementation der Win32 Installations-Services
Peter Jacobi	Deutschland	Verbesserungen, Updating der Zeichensätze
Tilo Muetze	Deutschland	Koordination des Firebird-Dokumentations-Projekts
Paul Vinkenoog	Niederlande	Koordination, Firebird-Dokumentations-Projekt; Fehlerbeseitigung in den UDF-Bibliotheken
Artur Anjos	Portugal	Erweiterte Win32 Systemsteuerungsanwendung; Entwicklung und Internationalisierung des Firebird Konfigurationsmanagers
Achim Kalwa	Deutschland	Erweiterte Win32 Systemsteuerungsanwendung
Sean Leyne	Kanada	Organisation des Bugtrackers; Bereinigung des Source-Codes
Ryan Baldwin	U.K.	Entwicklung des Jaybird Type 2 Treibers
Sandor Szollosi	Ungarn	Implementation von Zeichensätzen und Sortierreihenfolgen
Dmitry Kuzmenko	Russische Föderation	Fehlerbeseitigung in GSTAT
Artem Petkevych	Ukraine	Fehlerbeseitigung in der Behandlung von ARRAY-Datentypen
Vlad Horsun	Ukraine	Geschwindigkeitserhöhung des Datenbank-Sweeps; Fehlerbeseitigung im Zwei-Phasen-Commit
Tomas Skoda	Slovakei	Fehlerbeseitigung
Evgeny Kilin	Russische Föderation	Fehlerbeseitigung
Oleg Loa	Russische Föderation	Fehlerbeseitigung
Erik S. La Bianca	U.S.A.	Fehlerbeseitigung
Tony Caduto	U.S.A.	Inoffizielle Win32 Installationen

Entwickler	Land	Hauptaufgaben
Juan Guerrero	Spanien	Neu UDFs
Chris Knight	Australien	FreeBSD Builds
Neil McCalden	U.K.	Solaris Builds
Grzegorz Prokopsi	Ungarn	Debian Builds
Paul Beach	U.K.	HP-UX Builds
Geoffrey Speicher	U.S.A.	FreeBSD Builds
Helen Borrie	Australien	Autorin der Release-Notes; Feld-Testerin und "Gedankenpolizistin"

"DIE FELD-TEST-HELDEN"

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

I N S T A L L A T I O N S H I N W E I S E

Installation von Firebird 1.5 auf Windows 32

HINWEISE – VOR DER INSTALLATION BITTE LESEN

Mit der Einführung von zwei neuen Server Modellen unter Win32 haben sich auch die Installations-möglichkeiten geändert.

- Stellen Sie sicher, dass Sie als Administrator angemeldet sind (gilt nicht für Win9x oder ME)

- Alle Servermodelle — Superserver, Classic oder Embedded Server, genauso wie die Server-Tools und die Nur-Client-Tools — können mit dem Windows Installationsprogramm installiert werden. Für die Vollversion ist es empfehlenswert, zur Installation das zugehörige Installationsprogramm zu benutzen, soweit verfügbar.

- Benutzen Sie **gbak** um Ihre alte **isc4.gdb** Benutzer Datenbank zu sichern. Die Datenbank kann dann später als **security.fdb** zurückgesichert werden.

- Wenn Sie in der Datei **ibconfig** spezielle Einstellungen gespeichert haben, sind darunter evtl. Werte, die Sie entsprechend in die **firebird.conf** übertragen möchten. Beachten Sie dazu die Anweisungen über die firebird.conf, um herauszufinden, welche Einstellungen 1:1 übernommen werden können und welche Parameter eine neue Syntax verwenden.

- Sofern spezielle Konfigurationsdateien im Installationsverzeichnis vorhanden sind, werden diese entweder gesichert wenn Sie das Installationsprogramm benutzen oder ÜBERSCHRIEBEN, wenn Sie gezippte Installationsdateien in das Stammverzeichnis entpacken.
  Diese Dateien sind:
  security.fdb
  firebird.log
  firebird.conf
  aliases.conf

- Jedes Servermodell kann mit einer ZIP-Datei installiert werden. Diese Methode ist schneller als die Installation mit dem Installationsprogramm, allerdings sollten Sie über ausreichend Erfahrung mit der Installation von Firebird 1.5 verfügen.

- Es wird vorausgesetzt, dass:
  1 Sie verstehen wie Ihr Netzwerk funktioniert.
  2 Sie verstehen warum ein Client/Server System beides, nämlich einen Client und einen Server braucht
  3 Sie den Rest dieser Informationen gelesen haben oder wenigsten verstehen, dass Sie diese lesen müssen, sobald etwas nicht wie erwartet funktioniert
  4 Sie wissen, dass Sie Hilfe in der Firebird-Support Gruppe finden können, wenn Sie nicht mehr weiter wissen. Abonnieren Sie hier:
  http://www.yahoogroups.com/groups/firebird-support

Wenn Sie bereits eine frühere Version von Firebird oder InterBase® auf Ihrem Server installiert haben und sie denken, dass Sie möglicherweise diese wieder heranziehen möchten, dann treffen Sie entsprechende Vorkehrungen, bevor Sie beginnen.

- Verwenden Sie die existierende Version von GBAK um Ihre Datenbanken im transportablem Format zu sichern

- Machen Sie ein Backup von der Datei gds32.dll im Systemverzeichnis. Evtl. möchten Sie die Dateien zu "gds32.dll.ib5" oder "gds32.dll.fb103" oder etwas ähnlich sprechendem umbenennen, oder verschieben Sie die Dateien in einem anderen Verzeichnis

- Sie sollten auch ein Backup von der Microsoft C++ runtime, msvcp60.dll, anfertigen. Obwohl das Installationsprogramm diese Datei nicht überschreiben sollte, sind hierbei schon etwaige Probleme aufgetreten.

- **STOPPEN SIE ALLE LAUFENDEN FIREBIRD ODER INTERBASE SERVER**
  Das Installationsprogramm wird versuchen, eine existierende oder laufende Version von Firebird oder InterBase zu erkennen. Wenn Sie nicht das Installationsprogramm verwenden, so sind Sie dafür selbst verantwortlich!

- Das Standard Stammverzeichnis von Firebird 1.5 ist C:\Programme\Firebird\Firebird_1_5. Wenn sich Ihre alte Version bereits in diesem Verzeichnis befindet und Sie für die neue Installation dieses Verzeichnis verwenden wollen, dann benennen Sie das existierende Verzeichnis um.

- Firebird Installation als Service: Wenn Sie das neue **Sichere Login Feature** nutzen möchten, legen Sie einen "Firebird Service Benutzer" (Name und Passwort bleibt Ihnen überlassen) mit den entsprechenden Rechten als normalen Benutzer auf Ihrem System an. Dazu sollten Sie zuerst das Dokument README.instsvc.txt lesen. Wenn Sie über die gezippten Installationsdateien verfügen, finden Sie die Datei im /doc Verzeichnis unterhalb des Wurzelverzeichnisses der ZIP-Datei. Wenn Sie über die gezippten Installationsdateien nicht verfügen, dann ist das Dokument erst nach der Installation verfügbar. Sie können das Dokument jedoch auch unter dieser URL nachlesen:
  http://cvs.sourceforge.net/viewcvs.py/firebird/firebird2/doc/README.instsvc

LESEN SIE WEITER!

Eines der Designziele von Firebird 1.5 ist die Möglichkeit der Installation von mehreren Servern zu ermöglichen. Dies wird es dem Benutzer erlauben, mehrere unterschiedliche Versionen nebeneinander zu betreiben. Firebird 1.5 unterstützt dies bereits. Es ist jedoch nicht gut dokumentiert, und ein Eingreifen eines erfahrenen Benutzers ist dazu notwendig. Zukünftige Versionen von Firebird werden diesen Prozeß jedoch erheblich vereinfachen. Dafür setzt Firebird 1.5 bereits die notwendigen Vorkehrungen. Dies hat jedoch Auswirkungen auf die Installation der Client-Bibliotheken. Zur selben Zeit hat auch Microsoft deren eigenen Schritte unternommen, um die Installation unterschiedlicher Bibliotheksversionen zu handhaben. Dies hat zur Folge, dass nun eine neue Vorgangsweise bei der Installation der benötigten Bibliotheken für Firebird 1.5 und später gewählt wurde.

Installation von Microsoft Systembibliotheken

Das Problem der Installation unterschiedlicher Versionen der Microsoft Systembibliotheken ist derart berüchtigt, dass es bereits einen eigenen Namen dafür gibt, nämlich 'DLL Hell'.

Beginnend mit Windows 2000 hat es Microsoft nahezu unmöglich gemacht, System DLL's upzugraden. Um dieses Problem zu lösen, empfiehlt Microsoft, dass jede Anwendung eine lokale Kopie aller benötigten Systembibliotheken verwendet.

Firebird 1.5 folgt dieser Empfehlung und platziert die benötigten Bibliotheken mit dem Server im \bin Verzeichnis.

Installation von fbclient.dll

Firebird 1.5 und später verwendet nicht mehr gds32.dll als Client-Bibliothek. Diese heißt nun fbclient.dll. Unter Berücksichtigung der Probleme von Microsoft hinsichtlich deren eigenen 'DLL Hell' würde es keinen Sinn ergeben, wenn wir weiterhin die Firebird Client-Bibliothek in das <system> Verzeichnis speichern würden. Und da wir die simultane Installation mehrerer Server ermöglichen

wollen, würden wir unsere eigene 'DLL Hell' erschaffen, wenn wir weiterhin das <system> Verzeichnis für die Client-Bibliothek verwenden würden. Dies bedeutet, beginnend mit Firebird 1.5, wird nun die Client-Bibliothek im \bin Verzeichnis mit allen anderen Binaries abgelegt.

Ein neuer Registry-Schlüssel wurde hinzugefügt und alle Firebird-kompatiblen Anwendungen müssen diesen nun verwenden, um die Firebird-Version korrekt zu ermittlen, die verwendet werden soll. Der neue Schlüssel lautet:

HKEY_LOCAL_MACHINE\SOFTWARE\Firebird Project\Firebird Server\Instances

Firebird garantiert, dass immer zumindest ein Eintrag unter diesem Schlüssel existiert. Dieser lautet:

"DefaultInstance"

und speichert den Pfad zum (ja, richtig erraten) Stammverzeichnis der Default-Installation. Diejenigen, die sich nicht um bestimmte Installationen (Instanzen) kümmern wollen, können immer die Default Instanz verwenden, um fbclient.dll zu lokalisieren.

Zukünftige Firebird-Versionen werden andere Einträge unter Instances vorfinden. Anwendungen werden somit in der Lage sein diese Registryeinträge durchzulaufen, um zu ermitteln, welche Bibliotheksinstanz geladen werden soll.

Support für existierende Anwendungen und Treiber

Traditionell haben Anwendungen, die InterBase oder Firebird verwenden, die gds32.dll Client-Bibliothek vom <system> Verzeichnis geladen. Firebird 1.5 kommt mit einem Tool "instclient.exe" das einen Klon von fbclient.dll im Windowssystemverzeichnis installieren kann. Dieser Klon wird On-The-Fly "gepatched", so dass die Dateiversionsinformation mit einem "6.3" beginnt, um die Kompatibilität für alte Applikationen zu gewährleisten, die die Dateiversion von GDS32.DLL überprüfen, und mit einem String wie zum Beispiel "1.5" nichts anfangen können.

Während des Installationsprozesses überprüft der Installer, ob eine InterBase oder Firebird Installation existiert. Falls keine der beiden vorgefunden wird, so wird gds32.dll in das <system> Verzeichnis kopiert. Wird eine InterBase oder Firebird Installation vorgefunden, dann wird gds32.dll nicht in das <system> Verzeichnis kopiert. Es kann jedoch das "instclient.exe" Tool zu einem späteren Zeitpunkt dazu verwendet werden.

Es ist geplant, dass zukünftige Versionen von Firebird nicht versuchen werden, gds32.dll in das <system> Verzeichnis zu installieren. Womöglich wird diese auch komplett von der Distribution entfernt.

Das "instclient.exe" Tool kann des weiteren auch FBCLIENT.DLL selbst in das Windowssystemverzeichnis installieren, falls benötigt. Somit wird sichergestellt, dass Tools und Applikationen funktionieren werden, sofern diese erwarten, dass sich die Client-Bibliothek fbclient.dll im Windowssystemverzeichnis befindet.

Das instclient.exe Utility sollte sich im "bin" Verzeichnis Ihrer Firebird-Installation befinden, und muss auch von dort ausgeführt werden.

Verwendung von instclient.exe:

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

wobei *library* ist: `fbclient | gds32`

'-z' kann mit jeder anderen Option verwendet werden, gibt die Version aus.

Die Versionsinformation und der "Shared Library Zähler" (shared library counts) werden automatisch gehandhabt. Sie können die –f[orce] Option verwenden, um Versionsüberprüfungen zu übergehen.

HINWEIS: Fall Sie -f[orce] für die Installation verwenden, so kann dies zur Folge haben, dass eine andere bereits installierte Firebird oder InterBase® Version danach nicht mehr funktionsfähig ist. Des weiteren ist es womöglich notwendig, dass Sie den Rechner neu starten müssen, um die Installation abzuschließen.

Für weitere Details wird auf das Dokument *README.Win32LibraryInstallation.txt* verwiesen, das sich entweder im Stammverzeichnis Ihrer Firebird-Installation oder in ..\doc befindet.

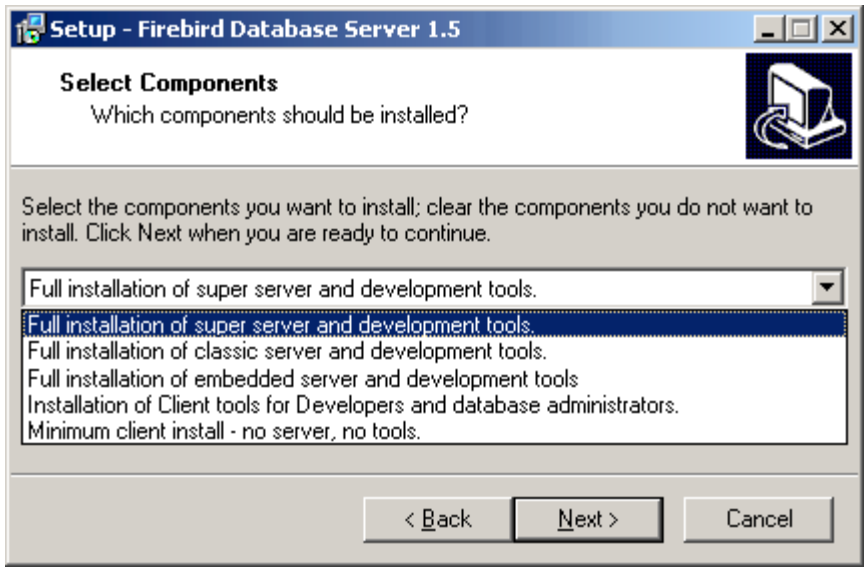
Aufräumarbeiten von Release Candidate Installationen

Es wird darauf hingewiesen, dass der Installer fbclient.dll vom <system> Verzeichnis entfernt, falls diese vorgefunden wird. Der Installer entfernt unter anderem auch den obsoleten Registry-Schlüssel HKLM\Software\FirebirdSQL.

Benutzung des Win32 Installationsprogramms

Dies ist der einfachste Weg.

Starten Sie einfach die ausführbare Datei und beantworten Sie die Dialoge. Nachdem Sie etwa 4 Dialoge beantwortet haben, sehen Sie eine Auswahlbox, die aufgeklappt etwa so aussieht. Dies ist die letzte Gelegenheit, um die Art der Installation auszuwählen.



Wählen Sie die gewünschte Installationsart aus, und klicken Sie auf "Next" um den nächsten Dialog angezeigt zu bekommen.

Service oder Anwendung?

Wenn Sie Superserver oder Classic zur Installation ausgewählt haben, und Ihr Betriebssystem Dienste unterstützt, so werden Sie gefragt, ob der Firebird Server als Dienst oder als Anwendung eingerichtet werden soll. Falls Sie keinen guten Grund haben den Server als Anwendung zu verwenden, so wählen Sie hier Service.

Manuell oder automatisch?

Mit der Option Automatic wird der Firebird Server jedes mal mitgestartet, wenn der Computer neu gestartet wird. Mit der Option Manual können Sie den Server selbst nach Bedarf starten.

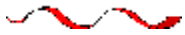
Guardian Option

Der Guardian ist ein Hilfsprogramm, das den Superserver überwacht und neu startet, wenn er aus irgendeinem Grund nicht mehr reagiert oder abgestürtzt ist. Während der Entwicklungsphase sollten Sie die Guardian Option deaktivieren. Im Produktionseinsatz kann der aktivierte Guardian dafür sorgen, dass der abgestürzte Server neu gestartet wird, ohne dass ein Eingreifen eines Datenbankadministrators notwendig ist.

Installationsverzeichnis (Stammverzeichnis)

Wenn Sie das Standard-Stammverzeichnis für die Installation nicht verwenden wollen, dann wählen Sie ein bereits angelegtes Verzeichnis aus, oder tippen Sie den kompletten Pfadnamen ein. Der Pfad, den Sie eingeben, muss nicht notwendigerweise existieren, da das Installationsprogramm in diesem Fall nachfrägt und das Verzeichnis anlegt.

Am Ende des Installationsvorganges wird der Firebird Server entweder im Hintergrund gestartet oder Sie werden gefragt, ob der Computer neu gestartet werden soll. Der Neustart ist dann notwendig, wenn das Installationsprogramm die Datei msvcp60.dll überschrieben hat, oder wenn eine alte Version der gds32.dll zum Zeitpunkt der Installation geladen war.



Installation des Superservers mit gezippten Installationsdateien

Die Installation von FB 1.5 ist prinzipiell der von früheren Versionen ähnlich.

Wenn Sie kein spezielles Installationsprogramm haben (nicht Teil der gezippten Version), dann sollten Sie den folgenden Schritten zur Installation folgen:

- Entpacken Sie das Archiv in ein separates Verzeichnis (da einige Dateinamen geändert wurden, macht es keinen Sinn, die Dateien in ein altes FB/IB Verzeichnis zu entpacken)
- Wechseln Sie in das Verzeichnis <root>\bin (wobei <root> das von Ihnen gewählte Verzeichnis ist, wohin Sie das Archiv entpackt haben)
- Starten Sie instreg.exe wie folgt:
  instreg.exe install
  Es wird der Pfad des übergeordneten Verzeichnisses in die Registry geschrieben
  (HKLM\Software\Firebird Project\Firebird Server\Instances\DefaultInstance)
- Wenn Sie den Dienst registrieren wollen, verwenden Sie instsvc.exe wie folgt:
  instsvc.exe install
- Optional sollten Sie die beiden Dateien fbclient.dll und gds32.dll in das Systemverzeichnis des Betriebssystems kopieren

Installation des Classic Servers mit gezippten Installationsdateien

Der Classic Server wird im Prinzip wie der Superserver installiert, nur dass die Installationsroutine mit einem zusätzlichen Parameter gestartet werden muss:

instsvc.exe install -classic

Bedenken Sie dabei, dass Sie nur eine Serverart – entweder fbserver.exe (Superserver) oder fb_inet_server.exe (der Elternprozess für Classicserver) – als Dienst installiert haben können.

Das Systemsteuerungs-Applet wird mit Classic nicht installiert. Versuchen Sie nicht, dieses zu installieren bzw. zu verwenden. Das Konzept zum Beenden des Service ist auf das Classic-Modell nicht anwendbar.

Vereinfachtes Setup

Wenn Sie keinen registrierten Dienst benötigen, so können Sie auf die Ausführung der Programme instreg.exe und instsvc.exe verzichten. In diesem Fall brauchen Sie nur die Dateien zu entpacken und den Server wie folgt zu starten:

```
fbserver.exe -a
```

Damit wird das übergeordnete Verzeichnis als Stammverzeichnis für den Server angesehen.

Deinstallation

Um FB 1.5 ohne Deinstallationsprogramm zu deinstallieren, gehen Sie wie folgt vor:

- Stoppen Sie den Server
- Starten Sie "instreg.exe remove"
- Starten Sie "instsvc.exe remove"
- Löschen Sie das Installationsverzeichnis
- Löschen Sie die Dateien fbclient.dll und gds32.dll aus dem Systemverzeichnis des Betriebssystems

Installation des Embedded Server mit gezippten Installationsdateien

Der Embedded Server ist ein Client mit einem voll funktionalen Server in Form einer DLL (fbembed.dll). Er hat die exakt gleiche Funktionalität wie der normale Server und exportiert auch die Standard-Einsprungspunkte der Firebird API.

Registry Die Registryeinträge für Firebird (wo der Server normalerweise nach dem Stammverzeichnis des Servers nachsieht) werden ignoriert. Das Stammverzeichnis des Embedded Servers ist das der Binärdatei (DLL) übergeordnete Verzeichnis.

Datenbankzugriff Nur ein rein lokaler Zugriff ist erlaubt. Der Embedded Server unterstützt keinen Fernzugriff. Daher ist auch der Zugriff über "localhost" nicht möglich.

Authentifizierung und Sicherheit Die Benutzerdatenbank (security.fdb) wird vom Embedded Server nicht benutzt und wird deshalb nicht benötigt. Jeder Benutzer kann sich zu jeder Datenbank verbinden. Da sowohl der Server, als auch der Client im selben (lokalen) Adressraum laufen, ergibt sich die Sicherheit aus dem physikalischen Zugriff auf den Computer.

SQL Privilegien werden wie bei anderen Servermodellen überprüft.

Kompatibilität Sie können eine beliebige Anzahl von Programmen mit dem Embedded Server ohne jegliche Konflikte laufen lassen. Selbst ein laufender IB/FB Server verursacht keine Probleme.

Sie sollten jedoch beachten, dass auf die gleiche Datenbank nicht von verschiedenen Embedded Servern gleichzeitig zugegriffen werden kann, da der Embedded Server auf der SuperServer Architektur aufbaut und deshalb die Datenbank exklusiv sperrt.

Dateistruktur für den Embedded Server

Kopieren Sie einfach die Datei fbembed.dll in das gleiche Verzeichnis, in dem sich Ihre Applikation befindet. Benennen Sie die Datei entweder in fbclient.dll oder gds32.dll, abhängig von der verwendeten Datenbankzugriffssoftware, um. Wenn Sie die Server-Hilfsprogramme (isql, gbak, etc.) benutzen möchten, so machen Sie eine Kopie der Datei fbembed.dll für fbclient.dll und für gds32.dll

Außerdem sollten Sie die Dateien firebird.msg, firebird.conf (wenn benötigt) und ib_util.dll in das gleiche Verzeichnis kopieren.

Falls externe Bibliotheken benötigt werden, z.B. internationale (INTL) Unterstützung (fbintl.dll) oder UDF Bibliotheken, so verwenden Sie eine Verzeichnisstruktur wie bei einer "regulären" Firebird Server Installation, z.B. in den Unterverzeichnissen \intl und \udf direkt unterhalb des Firebird Stammverzeichnis.

Beispiel

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

Starten Sie nun Ihre Anwendung. Diese wird den Embedded Server als eine Client-Bibliothek benutzen, und somit in der Lage sein, auf lokale Datenbanken zugreifen zu können.

HINWEIS: Angenommen Sie haben die Verzeichnisstruktur wie hier beschrieben verwendet, so ist es nicht notwendig RootDirectory in firebird.conf explizit zu setzen. Wenn Sie sich jedoch dazu entscheiden den Embedded Server und Ihre Applikation mit einer anderen Verzeichnisstruktur auszuliefern, dann lesen Sie zuerst das Dokument README_embedded.txt Ihrer Embedded Server Distribution, um Anweisungen für zusätzliche Konfigurationsmöglichkeiten zu erhalten.

Deinstallation

Die Firebird Deinstallationsroutine erhält die folgenden Dateien und benennt sie um:

- Erhält die Datei security.fdb oder benennt sie um zu security.fbnnnn
- Erhält die Datei firebird.log
- Erhält die Datei firebird.conf oder benennt sie um zu firebird.confnnnn
- Erhält die Datei aliases.conf oder benennt sie um zu aliases.confnnnn

"nnnn" ist die Build Nummer der alten Installation.
Es wird kein Versuch unternommen, Dateien zu deinstallieren, die nicht Teil der Originalinstallation waren.
Gemeinsam genutzte Dateien wie fbclient.dll und gds32.dll werden gelöscht, wenn ShareCount in der Registry signalisiert, dass kein anderes Programm diese Dateien noch weiter benötigt.
Alle erstellten Registryeinträge werden entfernt.

Zusätzliche Anmerkungen

Winsock2

Firebird benötigt Winsock2. Alle Win32 Plattformen sollten über WinSock2 verfügen, außer Win95. Die Verfügbarkeit der Winsock2 Bibliothek wird während der Installation überprüft. Die Installation wird fehlschlagen, wenn diese Bibliothek fehlt. Um auf Winsock2 upzudaten, folgen Sie diesem Link:
http://support.microsoft.com/default.aspx?scid=kb;EN-US;q177719

Windows ME und XP

Bei Windows ME und XP (Home und Professional Editionen) ein Feature "System Restore", das für ein wegsichern aller Dateien, unter anderem auch mit der Endung ".gdb", verantwortlich ist. Dies hat

merkliche Performanceeinbußen, bis zu einem kompletten Stillstand der Zugriffe zum Datenbankserver, wann immer eine I/O Operation stattfindet.

Eine Datei im Windows ME Verzeichnis, c:\windows\system\filelist.xml, enthält "geschützte Dateitypen". ".gdb" wird hier angeführt. Ursprünglich wurde von Charlie Caro empfohlen, diese Dateiendung in der Sektion "includes" komplett zu löschen. Seitdem wurde jedoch bekannt, dass u.U. diese Datei von ME wiederhergestellt wird. Unter XP ist es nicht möglich, diese Datei zu editieren.

Wenn Sie mit ME arbeiten, könnte eine mögliche permanente Lösung wie folgt aussehen:
- Verwendung der Erweiterung FDB (**F**irebird **DB**) für Datenbankdateien
- Als Speicherort der Datenbank das Verzeichnis "C:\Meine Dokumente" wählen. Dieses wird nämlich vom System Restore Feature vollständig ignoriert
- Komplette Deaktivierung von System Restore (konsultieren Sie hierfür die Windowsdokumentation).

Unter Windows XP Home and Professional Systemen können Sie Ihre Datenbanken auf eine separate Partition ablegen, und für die gewählte Partition das System Restore deaktivieren.

Windows XP benutzt ein "intelligentes Kopieren", so dass der Verwaltungsaufwand, der bei Systemen mit ME beobachtet wird, auf Systemen mit XP nicht so stark auffällt, zumindest nicht bei kleineren Dateien. Bei größeren Dateien wie z.B. Firebird Datenbankdateien scheint es z.Zt. keine bessere Lösung zu geben, als wenn man mit .gdb Dateien arbeiten muss.

Wir versuchen, eine akkurate Problembeschreibung und eine zuverlässige Lösung zu finden und bekannt zu machen. Wenn Sie dabei mitarbeiten können, dann senden Sie bitte eine Nachricht an die firebird-support oder firebird-devel Newsgroup, erreichbar unter news://news.atkin.com

Ein Problem existiert beim Herunterfahren des Rechners unter Windows XP, nämlich eine beträchtliche Pause beim des Stoppen des Serverdienstes. In diesem Zeitraum wird angezeigt, dass Firebird als Anwendung läuft. Dieses Problem scheint nur Windows XP zu betreffen, und nur wenn der Guardian zum Stoppen des Serverdienstes nicht verwenden wird. Die Verwendung des Guardian ist der momentane Workaround, bis eine Lösung für dieses Problem gefunden wurde.

Installation unter UNIX / Linux

(Im Original von Mark O'Donohue, überarbeitet für 1.5)

Firebird Server wird in zwei Versionen ausgeliefert; die Classic Version, die als Service läuft und SuperServer, die als Hintergrund-Daemon läuft. Classic ist der mehr traditionelle UNIX Service, während SuperServer Threads und keine Prozesse verwendet. Für Benutzer, die mit Firebird ihre ersten Gehversuche machen, sind beide geeignet, wobei Classic Server vermutlich eine bessere Plattform für das Experimentieren mit Firebird darstellt.

HINWEISE – VOR DER INSTALLATION BITTE LESEN

1) Sie müssen als root angemeldet sein um Firebird zu installieren.
2) Sie benötigen ein installiertes glibc Package größergleich glibc-2.2.5 und eine libstdc++.so größergleich libstdc++-5.0.
3) Orientieren Sie sich an dieser Tabelle für eine erste Bestandsaufnahme, ob Ihre Linux Installation geeignet ist. Trotzdem sind diese Angaben nicht als Garantie für die Lauffähigkeit von Firebird zu sehen, da Linux Installationen je nach Distribution und deren Build-Datum (stark) variieren können.
4) Vergewissern Sie sich, dass die "ed" und "vim" Editor Packages auf Ihrem System installiert sind. Falls der benötigte Editor nicht vorgefunden wird, dann wird zwar das Package installiert, aber das Installastionsskript schlägt anschließend fehl. (HINWEIS: Diese Abhängigkeit wird womöglich in der Zukunft durch "sed" ersetzt).

INSTALLATION UNTER LINUX

Die folgenden Anweisungen beschreiben die Classic Installation. Für die Installation von SuperServer muss das "CS" im Packagenamen durch ein "SS" ersetzt werden. Zum Beispiel, das Package FirebirdCS-1.5.0-nnnn.i686.rpm wird zu FirebirdSS-1.5.0-nnnn.i686.rpm.

Linux Installaion mit rpm

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

Linux Installation mit tar.gz

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

* or `FirebirdSS-1.5.0-nnnn`

Was die Linux Installtion macht

Die Linux Installation

1. Versucht einen bereits laufenden Server zu stoppen
2. Fügt einen Benutzer "firebird" und eine Gruppe "firebird" hinzu, sofern diese noch nicht existieren
3. Installiert die Software in das Verzeichnis /opt/firebird und erstellt Links für die Bibliotheken im Verzeichnis /usr/lib und für die Header-Dateien im Verzeichnis /usr/include
4. Fügt automatisch gds_db für port 3050 in /etc/services an, sofern dieser Eintrag noch nicht existiert
5. Fügt automatisch localhost.localdomain und HOSTNAME in /etc/host.equiv an
6. SuperServer installiert ein /etc/rc.d/init.d/firebird Server-Startskript
7. Classic Server installiert ein /etc/xinetd.d/firebird Server-Startskript oder für ältere inetd Systeme wird ein Eintrag in der Datei /etc/inetd hinzugefügt
8. Spezifisch für SuSE: Ein neuer rcfirebird Link wird in /usr/bin für das init.d Skript erzeugt. Außerdem wird ein /etc/rc.config Firebird-Einrag erstellt
9. Startet den Server/Service. Firebird sollte automatisch im Runlevel 2, 3 oder 5 gestartet werden
10. Generiert und setzt ein neues zufälliges SYSDBA Passwort und speichert dieses in der Datei /opt/firebird/SYSDBA.password
11. Erstellt für die Beispieldatenbank employee.fdb einen Eintrag in aliases.conf

Die Classic Installation konfiguriert automatisch den xinetd Eintrag, wenn das Verzeichnis /etc/xinetd.d gefunden wird. Andernfalls wird ein inetd Eintrag konfiguriert. Da einige Distributionen von xinetd Einträgen in anderen Verzeichnissen als /etc/xinetd.d ausgehen, kann womöglich eine manuelle Korrektur notwendig sein.

Testen Ihrer Linux Installation

Schritt 1 – Auf eine Datenbank zugreifen

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

SQL>connect localhost:employee.fdb /* this is an aliased path */

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

>quit;
```

* Ein Passwort wurde für Ihre Installation generiert. Dieses Passwort findet man in der /opt/firebird/SYSDBA.password Datei

Schritt 2 – Eine Datenbank erstellen

Ab Version 1.5 wird der Firebird Server per Default unter dem Benutzer "firebird" ausgeführt. Dies war immer die empfohlene Konfiguration. Vorherige Versionen wurden per Default unter dem Benutzer "root" ausgeführt. Wenn der Firebird Server unter root ausgeführt wurde, dann hatte der Server weitreichende Möglichkeiten um Datenbankdateien irgendwo im POSIX Dateisystem zu lesen, zu erzeugen und zu löschen. Aus Sicherheitsgründen sollte der Service eingeschränkte Möglichkeiten zum Lesen/Löschen/Erzeugen von Dateien besitzen.

Sicherheitstechnisch ist diese neue Konfiguration besser, benötigt allerdings einige spezielle Betrachtungen wenn es darum geht, Datenbanken zu erzeugen:

a) Der Benutzer "firebird" benötigt Schreibrechte im Verzeichnis, in dem Datenbanken erstellt werden sollen.
b) Der empfohlene Wert für den DatabaseAccess Parameter in /opt/firebird/firebird.conf sollte None sein, um den Zugriff nur über Einträge in aliases.conf zu erlauben.
c) Verwenden Sie Einträge (Aliases) in aliases.conf um den physischen Pfad zur Datenbank vom Benutzer zu verbergen. Mehr Informationen über Aliases gibt es hier.

Die Vorgehensweisen für das Erzeugen von neuen Datenbanken können variieren, falls Sie eine unterschiedliche Konfiguration gewählt haben, aber die folgenden Schritte werden für die hier vorgestellte Konfiguration empfohlen:

1) Falls ein Verzeichnis, für das "firebird" der Eigentümer sein soll, nicht existiert, dann wechseln Sie zum root Benutzer und erzeugen Sie das Verzeichnis:

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

2) Erzeugen Sie eine neue physische Datenbank und definieren Sie dafür einen Alias, der auf diese Datenbank verweist. Führen Sie das folgende Skript unter root oder firebird aus:

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

(Verwendung ist: createDBAlias.sh <dbname> <pathtodb>)

3) Als Alternative (zu Schritt 2) können Sie die Aktionen im createDBAlias.sh Skript auch manuell ausführen:

```
$vi /opt/firebird/aliases.conf
```
und fügen Sie folgende Zeile am Ende der Datei hinzu:
```
test.fdb /var/firebird/test.fdb
```

4) Erstellen Sie nun die Datenbank:

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

5) Falls der Wert von DatabaseAccess in /opt/firebird/firebird.conf auf Full oder auf einen eingeschränkten (Restrict) Verzeichnispfad gesetzt ist, dann ist eine weitere Alternative zu Schritt 2 möglich, nämlich das direkte Erzeugen der physische Datenbank durch Angabe des absoluten Pfades mit dem Dateinamen:

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

Falls Sie diese Konfiguration verwenden, dann können Sie auf die Datenbankdatei auch direkt zugreifen, ohne dafür einen Eintrag in der aliases.conf Datei erstellen zu müssen.

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

* Ein Passwort wurde für Ihre Installation generiert. Dieses Passwort findet man in der /opt/firebird/SYSDBA.password Datei

Tipps und hilfreiche Skripts

Zusätzlich zu den Standardinstallationsdateien existieren die folgenden Skriptdateien im bin Verzeichnis Ihrer Installation:

- **changeDBAPassword.sh**—Ändert das Firebird SYSDBA Benutzer Passwort. Für SuperServer ändert dieses Skript auch das Init Skript /etc/rc.d/init.d/firebird mit dem neuen Passwort.

- **createAliasDB.sh**—Verwendung: createAliasDB.sh <dbname> <dbpath>.
  Dieses Skript erstellt eine neue physische Datenbank und erzeugt dafür des weiteren einen Eintrag in der aliases.conf Datei.

- **fb_config**—Ein Skript das in Makefiles verwendet werden kann, um die notwendigen Include-Pfade und Lib-Include-Direktiven für die installierte Firebird Version zu generieren. fb_config -help zeigt eine vollständige Liste der möglichen Optionen an.

- **changeGdsLibraryCompatibleLink.sh**—Nur Classic—Ändert den Client-Library-Link für libgds.so zwischen der multi-threaded libfbclient.so und der single-threaded libfbembed.so Bibliothek, die ein direktes (Embedded) Öffnen einer Datenbankdatei erlaubt. Aus Kompatibilitätsgründen mit vorherigen Installationen, zeigt libgds.so per Default auf libfbembed.so.

- **Embedded oder direkter Zugriff auf Datenbankdateien**
  Die Classic-Installation stellt einen Embedded Modus zur Verfügung der es Programmen gestattet, Datenbankdateien direkt zu öffnen. Um diesen Modus zu betreiben wird ein (database-enabled) Benutzer benötigt, der priviligierten Zugriff auf einige Firebird Konfigurations- und Statusdateien besitzt.

- **Zugriff auf Datenbanken erhalten:** Da es nun der "firebird" (nicht root) Benutzer ist, unter dem die Software läuft, ist es notwendig zu wissen, wie man einen **Benutzer zu einer Gruppe hinzufügt**, um den Zugriff auf Datenbanken zu ermöglichen. Dies ist dokumentiert in den Readme Hinweisen, aber die folgenden Schritte sollen Ihnen dabei helfen:
  Um einen Benutzer (z.B. skywalker) zu der firebird Gruppe hinzuzufügen, so muss der root Benutzer folgende Schritte ausführen:

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

  Wenn sich "skywalker" das nächste mal anmeldet, dann kann er mit Firebird-Datenbanken arbeiten.

Um die Gruppen aufzulisten zu denen der Benutzer gehört, ist folgendes möglich:

```
$ groups
```

- **NTPL Probleme unter höheren Linuxversionen:**

Die neue NPTL (Native POSIX Thread Library) in Red Hat 9 (bis jetzt) verursacht Probleme für SuperServer und lokal compilierte Programme, inklusive der mitinstallierten Firebird Utilities. Im speziellen gbak wirft einen Broken Pipe Fehler. Um dies zu beheben:

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

Diese Änderungen beheben die Probleme auf Serverinstanzebene. Weiters muss die Umgebungsvariable für die lokale Umgebung gesetzt sein, also:

2. Fügen Sie die folgenden Zeilen in /etc/profile hinzu, um zu gewährleisten, dass diese für jeden Benutzer gesetzt sind, damit dieser mit den Kommandozeilentools arbeiten kann.

Nach
```
  HISTSIZE=1000
```
Fügen Sie hinzu
```
  LD_ASSUME_KERNEL=2.25
```
Mit folgender Zeile wird diese exportiert:
```
  export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUT_RC LD_ASSUME_KERNEL
```

Deinstallation unter Linux

Deinstallieren Sie den Firebird Server als root Benutzer. Die folgenden Beispiele verwenden den Classic Server, aber die Vorgehensweise ist für SuperServer gleich. Es braucht nur das "CS" durch ein "SS" ersetzt werden.

Für rpm Packages:

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

oder für die .tar.gz Installation:

$/opt/firebird/bin/uninstall.sh

Installation von Firebird Classic & SuperServer unter Solaris 2.7 Sparc

Zur Zeit nicht verfügbar. Wir verweisen Sie auf die Release Notes der Version 1.0 als Referenz für die Installation von 1.5.

Installation Firebird Classic unter MacOS X / Darwin

Zur Zeit nicht verfügbar. Wir verweisen Sie auf die Release Notes der Version 1.0 als Referenz für die Installation von 1.5

Compilieren oder Installieren von Firebird auf FreeBSD

Zur Zeit nicht verfügbar. Wir verweisen Sie auf die Release Notes der Version 1.0 als Referenz für die Installation von 1.5

Konfiguration des Port Service am Client und am Server

Ein Firebird Server verwendet per Default Port 3050 um Verbindungsanfragen von Clients zu verarbeiten. Dessen registrierten Port-Service-Name ist **gds_db**. Wenn Sie mit diesen Vorgaben zufrieden sind, dann brauchen Sie weder am Server noch am Client irgendwelche Einstellungen für den Port Service vornehmen.

Sie können einen unterschiedlichen Port, einen unterschiedlichen Port-Service-Name oder beides verwenden. Dies ist z.B. notwendig, wenn ein anderer Service Port 3050 benötigt, zum Beispiel ein auf gds_db konfigurierter, simultan laufender Firebird oder InterBase® Server unterschiedlicher Version. Es existieren unterschiedliche Möglichkeiten, um die Voreinstellungen zu überschreiben. Sowohl der Server als auch der Client muss konfiguriert werden, um den Port-Service-Namen oder den Port oder beides zu überschreiben. Dies geschieht mit zumindest einen der hier angeführten Wege:

- Im Client-Verbindungsstring
- Bei der Anweisung wie die ausführbare Serverdatei gestartet wird
- Durch Aktivierung des RemoteServicePort oder RemoteServiceName Parameters in firebird.conf (seit v1.5)
- In der Daemon Konfiguration (für Classic unter POSIX)
- Durch einen Eintrag in der services Datei

Bevor man sich die hier angeführten Methoden näher ansieht ist es hilfreich zu wissen, welche Logik der Server verwendet, um den Port – über den Anfragen abgearbeitet werden sollen - zu setzen und wie für den Client der Port gesetzt werden kann, damit dieser den entsprechenden Port für die Kommunikation verwendet.

Wie der Server den abzuhörenden Port setzt

Die ausführbare Serverdatei besitzt einen optionalen Kommandozeilenschalter (-p) mit dem entweder die **Portnummer** oder der **Name des Portservices** definiert werden kann. Wird beim Start der ausführbaren Serverdatei dieser Schalter verwendet, so wird entweder die Portnummer 3050 oder der Port-Service-Name (gds_db) durch den mit dem von –p Schalter übergebenen Argument ersetzt.

Wird der –p Schalter nicht angegeben, so überprüft ein v.1.5 Server die firebird.conf Datei und schlägt die Parameter RemoteServiceName und RemoteServicePort nach:

- Falls beide mit "#" kommentiert sind, dann werden die Voreinstellungen verwendet, und es werden keine weiteren Änderungen vorgenommen. Ein –p Argument überschreibt entweder die Portnummer oder den Namen des Portservices, und das "fehlende" Argument bleibt in dessen Voreinstellung erhalten.

- Falls RemoteServiceName auskommentiert worden ist, RemoteServicePort hingegen nicht, dann wird *nur* der Port-Service-Name ersetzt und nur dann, wenn dieser nicht durch den –p Schalter überschrieben wurde.

- Falls RemoteServicePort auskommentiert worden ist, RemoteServiceName hingegen nicht, dann wird *nur* die Portnummer ersetzt und nur dann, wenn diese nicht durch den –p Schalter überschrieben wurde.

- Falls beide, RemoteServicePort und RemoteServiceName auskommentiert worden sind, dann hat RemoteServiceName den Vorrang, sofern dieser nicht durch den –p Schalter überschrieben wurde. Wenn der Port-Service-Name bereits überschrieben wurde, dann wird der Wert von RemoteServiceName ignoriert, und der Wert von RemoteServicePort überschreibt 3050.

- Falls ein Überschreiben der Portnummer oder des Port-Service-Namen stattgefunden hat, dann fahren sowohl v.1.0 als auch v.1.5.0 mit der Überprüfung der services Datei fort, um zu überprüfen, ob ein Eintrag mit einer korrekten Kombination der Portnummer und des Port-Service-Namen existiert. Wird eine Übereinstimmung gefunden, so ist alles in Ordnung. Falls keine Übereinstimmung gefunden wurde, und der Port-Service-Name nicht gds_db lautet, dann wirft der Server eine Exception und der Serverstart schlägt fehl. Wenn gds_db der Port-Service-Name ist, und dieser auf keinen anderen Port aufgelöst werden kann, dann wird dieser automatisch auf Port 3050 abgebildet.

Falls der Default-Servicename überschrieben wurde, dann müssen Sie einen Eintrag in der services Datei hinzufügen – Lesen Sie dazu den Abschnitt darunter, *Konfiguration der services Datei*.

Verwenden des –p Schalters zum Überschreiben

Beachten Sie, dass dieser Schalter bereits in Firebird 1.0.x verfügbar ist, jedoch nicht dokumentiert wurde.

Das Starten des Servers mit dem optionalen –p Schalter ermöglicht es Ihnen entweder die Voreinstellung für den Port (3050) oder für den Port-Service-Namen (gds_db), den der Server für eingehenden Anfragen abhört, zu überschreiben. Dieser Schalter kann einen Wert überschreiben, aber nicht beide. Ab Firebird 1.5 ist es nun möglich den –p Schalter in Kombination mit einer Konfiguration in firebird.conf zu verwenden, um sowohl die Portnummer, als auch den Port-Service-Namen zu überschreiben.

Syntax für TCP/IP

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

Um z.B. SuperServer als Anwendung zu starten, und den Port-Service-Namen gds_db mit fb_db zu überschreiben, gehen Sie wie folgt vor:

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

Oder um den Port von 3050 auf 3051 zu ändern:

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

Syntax für WNet Redirection

Ersetzen Sie das Argument des –p Schalters in einem WNet Netzwerk mit der folgenden "Backslash-Backslash-Dot-At" Syntax:

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

oder

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

Classic unter POSIX: der *inetd* oder *xinetd* Daemon

Für Firebird Classic Server unter Linux oder UNIX wird der **inetd** oder **xinetd** Daemon konfiguriert, um den Default-Port abzuhören und den Default-Service-Namen per Broadcast zu verteilen. Das Installationsskript schreibt den entsprechenden Eintrag in die Konfigurationsdatei /etc/inetd.conf oder /etc/xinetd.conf.

Falls notwendig, können Sie die aktuelle Konfiguration ändern. Folgendes Beispiel ist der Eintrag den Sie in xinetd.conf oder inetd.conf nach der Installation von Firebird Classic unter Linux vorfinden:

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

Wenn Sie den Portservice unterschiedlich zu den Voreinstellungen konfiguriert haben, dann müssen Sie xinetd.conf oder inetd.conf entsprechend den neuen Einstellungen ändern. Starten Sie xinetd (oder inetd) mit kill –HUP neu um sicherzustellen, dass die neue Konfiguration verwendet wird.

HINWEIS: Verbindungsanfragen werden fehlschlagen, wenn beide xinetd (oder inetd) und fbserver (or ibserver) versuchen den selben Port abzuhören. Falls Ihr System diese Konfiguration aufweist, dann ist es notwendig, dass jede Server-Version dessen eigenen Serverport erhält.

Verwenden eines Konfigurationsdateiparameters

Sie können entweder RemoteServiceName oder RemoteServicePort in firebird.conf dahingehend konfigurieren, um den Default-Port (3050) oder den Default-Port-Service-Namen (gds_db) zu überschreiben.

Die Engine wird einen RemoteService* Parameter verwenden, aber nicht beide. Wenn Sie beide verwenden, dann wird RemoteServicePort immer ignoriert, ausgenommen wenn die Serverstartanweisung den –p Schalter verwendet, um den Port-Service-Namen zu überschreiben. Somit ist es unter Verwendung einer Kombination des –p Schalters und einem RemoteService* Parameter möglich, beide Portnummer und Port-Service-Name zu ändern.

Wenn die Voreinstellung für den Port-Service-Namen überschrieben werden soll, dann ist es notwendig, dass ein entsprechender Eintrag in der services Datei erstellt wird.

ACHTUNG! Falls Sie RemoteServiceName oder RemoteServicePort verwenden, jedoch die Voreinstellungen beibehalten, dann werden diese als überschrieben betrachtet. Somit ist es notwendig, dass ein Eintrag in der services Datei mit den Voreinstellungen für die Port-Service-Einstellungen vorhanden ist.

Konfiguration des Clients um den Serviceport zu finden

Wenn Sie einen Server mit den Voreinstellungen (Service gds_db auf Port 3050) aufgesetzt haben, dann sind keine weiteren Schritte notwendig. Falls der Server einen unterschiedlichen Port oder Port-Service-Namen verwendet, dann benötigt die Clientapplikation und/oder die Host-Maschine bestimmte Einstellungen, damit die Firebird-Clientbibliothek den richtigen Port findet. Der von einem Client verwendete Verbindungsstring kann den zu verwendenden Port beinhalten. Firebird 1.5 Clients können optional eine lokale Kopie von firebird.conf verwenden. Änderungen müssen dann womöglich auch in der services Datei am Client durchgeführt werden

Verbindungsstring verwenden

Wenn nur die Portnummer oder der Servicename geändert wurde, dann können Sie den alternativen Port oder Servicenamen im Verbindungsstring mitangeben. Dies funktioniert für alle Firebird Versionen.

Syntax für TCP/IP Verbindungen

Wollen Sie sich zu einer Datenbank am Server alice mit Port 3050 über den Servicenamen fb_db verbinden, dann würde der Verbindungsstring wie folgt aussehen:

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

Oder, falls der Servicename gds_db und der Port 3051 ist:
```
alice/3051:/data/teaparty.fdb
```

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

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

Beachten Sie, dass das Trennzeichen zwischen dem Servernamen und dem Port ein Slash und kein Doppelpunkt ist. Der Doppelpunkt vor dem physikalischen (absoluten) Pfad ist nach wie vor erforderlich.

Syntax für WNet Verbindungen

Verwenden Sie in einem WNet Netzwerk die UNC-Notation:

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

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

Falls der konfigurierte Port oder Servicename am Server überschrieben wurde, dann benötigen Sie einen Eintrag in der services Datei.

Verwendung der Portsyntax mit Datenbankaliase

Um sich mit einem Datenbankalias über einen Port, unterschiedlich zur Voreinstellung, zu verbinden, so geben Sie den Port nach dem Servernamen und nicht nach dem Alias an. Angenommen Sie haben den folgenden Alias in aliases.conf gespeichert:

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

Der Verbindungsstring in Ihrer Applikation würde für einen Server "alice" wie folgt aussehen:

```
alice/fb_db:rabbit
```

oder
```
alice/3051:rabbit
```

Verwendung einer Kopie von firebird.conf

Ab Firebird 1.5 können Sie nun optional auf dem Client eine Kopie von firebird.conf im Firebird Stammverzeichnis speichern, um RemoteServiceName oder RemoteServicePort zu konfigurieren, damit der Client in der Lage ist den Serverport zu lokalisieren.

- Sie können *einen* der beiden Parameter konfigurieren, um das Überschreiben durch Verwendung des Verbindungsstring dahingehend zu erweitern, dass der andere Parameter ebenfalls überschrieben wird, oder um nur RemoteServiceName oder RemoteServicePort zu überschreiben, ohne dass dies durch den Verbindungsstring gemacht werden muss.

- Wenn Sie vermeiden möchten den Port-Service-Namen oder die Portnummer im Verbindungsstring angeben zu müssen und der Server Einstellungen für beide Parameter verwendet, die von den Voreinstellungen abweichen, dann können Sie dies über die Parameter RemoteServiceName und RemoteServicePort definieren. Diese Vorgehensweise sollten Sie allerdings vermeiden, wenn Sie sich zu einem InterBase oder Firebird 1.0 Server verbinden wollen.

Speicherort von Firebird-Dateien am Client

Wenn Ihre Clientinstallation auf firebird.conf basiert, dann ist es wichtig, dass die Clientbibliothek die Konfigurationsdatei findet. Sie müssen hierzu ein Firebird-Stammverzeichnis erstellen, und dem System den Speicherort bekanntgeben. Verwenden Sie hierzu die FIREBIRD Umgebungsvariable. Windows Clients können dazu alternativ auch den Firebird-Registryeintrag verwenden. Mit einer korrekten Clientinstallation können Sie des weiteren auch eine lokale Version der Meldungsdatei verwenden.

Konfiguration der services Datei

Es ist nicht notwendig einen Serviceporteintrag für den Firebird Server oder dem Client zu erstellen, sofern der Server die Installationsvoreinstellungen gds_db mit Port 3050 verwendet. Fall gds_db der Port-Service-Name ist und dieser auf keinen anderen Port aufgelöst werden kann, dann wird dieser automatisch auf Port 3050 abgebildet.

Wenn Sie den Serviceport auf einen unterschiedlichen Port oder Servicenamen einstellen, dann müssen beide, **Server** und **Client** explizit entsprechend diesen Änderungen rekonfiguriert werden. Sowohl unter Linux als auch unter Windows ist diese Information in der **services** Datei gespeichert.

Lokalisierung der services Datei

- Unter Windows NT/2000/XP/S2003 ist diese Datei C:\windows\system32\drivers\etc\services.

- Unter Windows 95/98/ME ist dies C:\windows\services.

- Unter Linux/UNIX ist dies /etc/services.

Ein Serviceeintrag sieht wie folgt aus:

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

Öffnen Sie diese Datei mit einem Texteditor und fügen Sie entweder eine neue Zeile hinzu oder editieren Sie den existierenden gds_db Eintrag, wie folgt:

- Für einen Firebird 1.0.x Server oder Client, ändern Sie entweder den Servicenamen oder die Portnummer um anzugeben, wie der Server gestartet wird.

- Für einen Firebird 1.5 Server oder höher, ändern Sie den existierenden Eintrag oder fügen Sie einen Neuen hinzu. Falls Sie auf dem selben Hostsystem einen Firebird 1.0 oder InterBase Server installiert haben, dann ändern Sie nicht den existierenden Eintrag, sondern fügen einen Neuen hinzu, um anzugeben, wie der Firebird 1.5 Server gestartet wird.

Weitere Informationen

Weiterführende Informationen über die Firebird Datenbank Engine finden Sie unter:
http://firebird.sourceforge.net

oder verwandte Seiten:
http://firebirdsql.org
http://www.ibphoenix.com
http://www.cvalde.net

Wenn Sie sich für die Mitarbeit an der Entwicklung von Firebird interessieren oder einen möglichen Fehler zur Diskussion stellen wollen, treten Sie unserer firebird-devel Liste bei. Um diese zu abonnieren, senden Sie eine leere E-Mail an:

firebird-devel-request@lists.sourceforge.net

mit dem Wort "subscribe" im Betreff Feld.

Verwenden Sie die firebird-devel Liste bitte nicht für Support-Fragen

Für technischen Support treten Sie bitte der Liste firebird-support bei:
http://www.yahoogroups.com/groups/firebird-support

Für InterClient und Java Entwicklung und Support existiert eine eigene Liste:
http://www.yahoogroups.com/groups/Firebird-Java

Die firebird-support Liste kümmert sich um technische Probleme rund um Firebird und Interbase(R). Bitte richten Sie Delphi- und andere programmiersprachenspezifische Fragen an die dafür vorgesehenen Foren.

Die Open-Source-Community betreibt verschiedene andere Diskussionsgruppen bzgl. unterschiedlicher Aspekte der Firebird Entwicklung. Für weitere Details wenden Sie sich bitte an die Mailing Liste und die Newsgroups der Firebird Community Site.

Die Firebird-Entwickler-Liste und die generelle Community-Liste, zusammen mit einigen anderen Listen, interessant für Firebird und InterBase Entwickler, werden gespiegelt unter
news://news.atkin.com

Bezahlter weltweiter Support kann durch IBPhoenix arrangiert werden (Kontaktadressen und Nummern unter http://www.ibphoenix.com). Verschiedene Mitglieder des Firebird-Teams sind außerdem verfügbar für Support und Consulting. Bitte kontaktieren Sie diese Mitglieder direkt.

Datenbank-Restaurierungs-Service für Firebird und InterBase Datenbanken kann von IBPhoenix bezogen werden. Für die Analyse und mögliche Reparatur beschädigter Datenbanken durch Sie selbst, versuchen Sie IBSurgeon von IBase.ru (www.ibase.ru)
IBase.ru bietet auch einen Restaurierungsservice in Russland und Europa an.

Anfragen/Angebote für bezahlte Verbesserungen von Firebird können direkt an die FirebirdSQL Foundation Inc. gerichtet werden. Senden Sie eine E-Mail an foundation@firebirdsql.org. Womöglich bevorzugen Sie den Erstkontakt mit den Projektadministratoren, dann Senden Sie eine E-Mail an firebirds@users.sourceforge.net

Allgemeine Diskussionen über Firebird Verbesserungen können in der Firebird-Prioritäten Liste besprochen werden http://www.yahoogroups.com/community/Firebird-priorities.

IB-Architect (http://www.yahoogroups.com/community/ib-architect) ist ausschließlich für Diskussionen über **technisches Design**. Supportanfragen sind dort definitiv off-topic.

Tools und Treiber

Database Desktop Client Programs

Unter http://www.ibphoenix.com finden Sie ein Reihe exzellenter grafischer Administrationstools für den Firebird Server. Es handelt sich hierbei um Open Source, Freeware, und kommerzielle Produkte.

Borland's IBConsole wird als Administrations-Client für Firebird 1.5 nicht empfohlen.

Treiber und Komponenten

JAVA: Das JayBird JDBC Treiber Projekt wird aktiv als ein Teil des Firebird Projektes entwicklelt. Das Projekt hat einen Type 4 JDBC/JCA Treiber und einen Type 2 Treiber freigegeben. Quellcode und Binaries können hier heruntergeladen werden:
http://sourceforge.net/project/showfiles.php?group_id=9028

Für Hilfestellung und Teilnahme an Entwicklung und Tests, werden Sie Mitglied im Firebird-java Forum unter: http://www.yahoogroups.com/community/firebird-java.

.NET: Firebird hat ein .NET Treiber Projekt in der Entwicklung. Quellcode und Binaries können hier heruntergeladen werden:
http://sourceforge.net/project/showfiles.php?group_id=9028

Für Hilfestellung und Teilnahme an Entwicklung und Tests, werden Sie Mitglied im Firebird .NET Forum unter: http://lists.sourceforge.net/lists/listinfo/firebird-net-provider

Delphi and C++Builder: Entwickler haben die Wahl zwischen zwei Alternativen für den direkten Zugriff über die Firebird 1.5 API. Beide mit gutem Support durch die Hersteller und anderen Entwicklern, die diese Produkte verwenden:

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

C++: Die Freeware C++ Zugriffsbibliothek, IBPP (http://www.ibpp.org), MPL Lizenz, gehostet auf sourceforge.net, vollständig portierbar zwischen Win32 und Linux und möglicherweise anderen POSIX Plattformen. Diese Bibliothek ist nützlich, wenn Sie einen Low-Level C-API Zugriff, allerdings mit einer leichten C++ Abstraktionsschicht, die an keine Entwicklungsumgebung gebunden ist, benötigen.

ODBC: Eine Liste mit ODBC Treibern ist hier verfügbar: http://www.ibphoenix.com. In folgendem Forum wird die Weiterentwicklung des Open Source ODBC/JDBC Treibers diskutiert:
http://lists.sourceforge.net/lists/listinfo/firebird-odbc-devel

PHP: Eine Gruppe von Entwicklern arbeitet an der Anpassung der alten InterBase PHP Erweiterung für Firebird. Wenn Sie mehr über das Projekt erfahren möchten, dann werden Sie Mitglied im Firebird-PHP Forum unter: http://www.yahoogroups.com/community/firebird-php

PYTHON: KInterbasDB ist ein Python Erweiterungspaket das den Support für die Python Database API 2.0 für Firebird implementiert. Voll-native API Unterstützung für den Firebird Client. Verfügbar als stabiles Release und unter aktiver Entwicklung. BSD Open Source Lizenz. Downloads und News unter http://kinterbasdb.sourceforge.net/

Dokumentation

Die Dokumentation für InterBase Version 6.0 gilt auch noch für die aktuelle Firebird Version. Eine Beta Version der InterBase(tm) 6 Handbücher ist im Adobe PDF Format verfügbar unter:

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

Ein strukturierter Index für die Dokumentation wird auf der Firebird Community Seite gepflegt unter:

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

Dabei handelt es sich um ein Projekt in der Entwicklung. Anregungen sind herzlich willkommen. Senden Sie dazu eine E-Mail an: firebird-docs@lists.sourceforge.net

Einige Installationsanleitungen und andere HowTos können in der Documentation Area über folgenden Link gefunden werden:
http://www.firebirdsql.org
oder direkt von
http://sourceforge.net/projects/firebird

Die Hauptquelle für Benutzer- und technische Fragen ist die IBPhoenix Website-
http://www.ibphoenix.com
IB Phoenix gibt in regelmäßigen Abständen eine umfassende Abo-CD heraus (auch einzelne Editionen sind erwerbbar), die unter anderem von IBPhoenix publizierte Handbücher enthält – "Using Firebird" und "The Firebird Reference Guide".

Zusätzliche Dokumentationen können auch in der Borland Techpubs Area gefunden werden:
http://www.borland.com/techpubs/interbase/

Bugfixes und Erweiterungen seit dem Release 1.0 (im Original)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Anmerkungen zu dieser Übersetzung:

Diese Übersetzung wurde erstellt von

Thomas Steinmaurer ts@iblogmanager.com
Lucas Franzen luc@rol3.com
Frank Schlottmann-Gödde fsg@users.sourceforge.net
Florian Hector FHector@web.de
Dominik Fässler d.faessler@performa-software.ch

Hilfreiche Tipps kamen wie immer von:

Volker Rehn volker.rehn@bigpond.com