Firebird™ Verze 1.5

Release Notes v.1.5

5 Únor 2004

Obsah

Úvod

Databázový systém Firebird™ byl vytvořen nezávislým týmem vývojářů ze zdrojových kódů InterBase™ uvolněných jako open source firmou Borland pod licencí InterBase Public License v.1.0 dne 25. července 2000.

Vývoj ve zdrojovém stromu Firebird 2 započal již v průběhu vývoje Firebirdu 1 převodem zdrojových textů Firebirdu 1 z jazyka C do C++ a prvním velkým „čištěním" kódu. Firebird 1.5 je první veřejná verze systému založená na zdrojovém stromu Firebird 2. Ačkoliv jde o důležitý milník pro vývojáře i celý projekt Firebird, nejedná se o konečné stádium vývoje Firebirdu. Současně s přípravou na zveřejnění verze 1.5 probíhá vývoj nové „menší" verze (1.6) jako dalšího milníku na cestě k Firebirdu 2.

Verze Firebird 1.0.x je nadále udržována, a důležité opravy a vylepšení jsou zpětně portována z verze 1.5.

Binární distribuce Firebirdu 1.5

Binární distribuce Firebirdu lze stáhnout z webových stránek projektu Firebird - http://sourceforge.net/project/showfiles.php?group_id=9028

Version Stringy pro Firebird 1.5

Win32: "WI-V1.5.0.nnnn Firebird 1.5"
Linux: "LI-V1.5.0.nnnn Firebird 1.5"
atd., kde nnnn je číslo sestavy (build number)

Přehled doporučené dokumentace naleznete v sekci Dokumentace.

Nové vlastnosti

Nový zdrojový strom, lepší optimalizace

Tato verze je vytvořena ze zdrojových textů přepsaných z původního jazyka C do C++, přičemž tento proces zahájil Mike Nordell již v roce 2000. Následovalo rozsáhlé „čištění" kódu a opravy chyb, spolu s vytvořením nové správy paměti a rozšířeními jazyka. V neposlední řadě pak Arno Brinkman a další provedli podstatné opravy a vylepšení optimalizátoru SQL dotazů, které vedly k zvýšení rychlosti dotazů v hlášeném rozsahu 30 až 60 (a více) procent.

Architektura

Pro platformu Windows přibyly dvě významné varianty v podobě Classic serveru a zapouzdřeného (embedded) serveru.

- Classic server nebyl na Windows k dispozici po více než osm let. Tato varianta serveru dokáže plně využít více procesorů, což se architektuře Super Server zatím nedaří. Ačkoliv je Classic server pro Windows použitelný, měl by být stále považován za experimentální.

- Zapouzdřený (embedded) server je sdílená dynamická knihovna (dll) která spojuje klientskou knihovnu a Firebird Super Server pro tvorbu rychlých a efektivních samostatných a „kufříkových" aplikací.

Od verze 1.0.x byly přidány nové důležité vlastnosti jazyka SQL, včetně podpory větvících funkcí CASE, COALESCE a NULLIF dle standardu SQL-92. Popis syntaxe a další informace o těchto nových vlastnostech naleznete v samostatné sekci tohoto dokumentu věnované rozšířením jazyka SQL

Instalované moduly a bezpečnost

Pokud jste doposud používali Firebird 1.0.x, zaregistrujete velké změny v názvech modulů a v pravidlech pro jejich vyhledávání a přístup k nim. Zde jsou uvedeny pouze význačné změny; detailní informace, umístění a způsob konfigurace naleznete v příslušné části dokumentu.

1. Většina modulů a konstant byla přejmenována. Ve většině případů obsahují nová jména variantu slov "firebird" nebo "fb". Například klientská API knihovna je nyní umístěna v dynamické sdílené knihovně nazvané"fbclient.dll" na Windows a "libfbclient.so" na ostatních platformách. Výjimkou která porušuje toto pravidlo je bezpečnostní databáze, dříve pojmenovaná jako"isc4.gdb", která se nyní nazývá"security.fdb".
2. Externí soubory používané serverem (UDF knihovny, BLOB filtry, knihovny znakových sad, externí tabulky) jsou nyní podřízeny úrovním ochrany na úrovni souborového systému, jejichž implicitní

hodnoty jsou v některých případech odlišné od úrovně která je standardní pro Firebird 1.0.x nebo InterBase.

3. Nový konfigurační soubor serveru, firebird.conf, který nahrazuje původní konfigurační soubor ibconfig (Windows) a isc_config (ostatní platformy) obsahuje několik nových konfiguračních parametrů, spolu s interní dokumentací v podobě vložených komentářů a lepší vnitřní organizací.
4. Verze 1.5 nabízí možnost vytvářet zástupná jména (alias) pro databáze. Tato vlastnost umožňuje nepřímou vazbu aplikace na databázi, kdy je specifikace cesty k databázi nahrazena zástupným jménem dle vašeho výběru. Skutečná cesta k databázi je uložena v textovém souboru aliases.conf na serveru. Nicméně hlavní účel této vlastnosti je ochránit informaci o fyzickém uložení databází před zneužitím.
5. Dle standardní (a dřívější) praxe je Firebird na Windows instalován jako služba pod účtem lokálního uživatele který spustil instalační program. To vytváří závažné bezpečnostní riziko v případě, že by byl Firebird server hacknut, protože by útočník získal přístup k systému se všemi právy uživatele který Firebird instaloval. Proto nyní program instsvc.exe verze 1.5 umožňuje specifikovat uživatelský účet pro instalovanou službu. Je doporučeno vytvořit separátní uživatelský účet pro Firebird server, a instalovat Firebird jako službu pod tímto účtem, obzvláště pokud je server připojen jakýmkoliv způsobem k Internetu.

Odstranění nepotřebných mezer u Varchar položek v přenosovém protokolu

Ve verzi 1.5 byla dokončena nová vlastnost, která u položek typu Varchar odstraňuje při přenosu po síti přebytečné mezery zprava. Po síti se tedy Varchar položky přenáší pouze v aktuální délce plus dva bajty.
POZNÁMKA: Protože jsou koncové mezery serverem odstraňovány na žádost klienta, je tato vlastnost dostupná i pro starší verze serveru pokud je k nim přistupováno prostřednictvím klienta verze 1.5 (knihovna fbclient.dll nebo libfbclient.so). Pokud k serveru verze 1.5 budete přistupovat prostřednictvím staršího klienta, koncové mezery nebudou při přenosu odstraňovány bez ohledu na verzi serveru ke které se připojujete.

Sémantika spouští pro více akcí

Nově lze vytvářet podmíněné spouště pro insert/update/delete akce v jediné Before nebo After spoušti, a pokrýt tak všechny DML akce jedinou spouští v dané fázi. Tato vlastnost redukuje složitost a náročnost údržby spouští bez ztráty možnosti vytvářet více spouští pro jednotlivé fáze zpracování řádku.

Rozšíření pojmenovaných integritních omezení

Nově lze uživatelsky pojmenovat indexy vytvářené automaticky pro realizaci integritních omezení. Upozornění: Pokud tuto vlastnost použijete, databázi nebude možné používat s InterBase nebo Firebirdem 1.0.x.

Zvýšení maximálního počtu indexů na tabulku

Nově lze v obou verzích – 1.5 i 1.0 – definovat 64-256 indexů na tabulku. Původně bylo možné vytvořit maximálně 64 indexů.

Pesimistické zamykání

Nově je k dispozici syntaxe umisťující zámek na řádky přečtené klientem pro zvláštní případy kdy potřebujete vytvořit pesimistický zámek. Používejte s rozumem.

Vyrovnávací paměť připojení k bezpečnostní databázi

Připojení k bezpečnostní databázi v architektuře Super Server jsou nyní uchovávána ve vyrovnávací paměti. To znamená, že databáze security.fdb je načtena při prvním připojení a zůstává interně připojena pro potřeby všech dalších připojení, až do ukončení všech klientských připojení.

Vylepšení chybových hlášení

Kdekoliv je to možné, obsahují chybová hlášení mnohem více podrobností u příčině chyby.
DŮLEŽITÉ UPOZORNĚNÍ: Pokud budete používat starší soubory interbase.msg nebo firebird.msg, můžete obdržet bizardní chybová hlášení.

Services API pro Classic Server na Linuxu

Classic server na platformě Linux má nyní omezenou podporu Services API. Fungují službu na bázi gbak (záloha/obnova) a gfix (kontrola databáze, shutdown/online atd.). Ostatní služby (gstat, server log atd.) nebyly testovány a pravděpodobně nepracují.

Změny v klientských knihovnách

Windows klient

Klientská knihovna se nyní nazývá "fbclient.dll". Všechny standardní nástroje serveru (gbak, gfix, atd.) používají pouze knihovnu fbclient.dll. Připojujte nové aplikace prostřednictvím knihovny fbclient.dll, bez potřeby knihovny gds32.dll (doporučeno).
Pro kompatibilitu s existujícími aplikacemi je možné vytvořit „klon" knihovny „fbclient.sll" pod názvem „gds32.dll" pomocí nového nástroje instclient.exe. Více informací naleznete v sekci o instalaci a nejaktuálnějších poznámkách k instalaci obsažených v distribučním balíku pro Windows.

Linux klient

Knihovna pro Super Server se nyní nazývá"libfbclient.so". Pro zpětnou kompatibilitu se staršími aplikacemi je při instalaci vytvářen symbolický link "libgds.so" na knihovnu libfbclient.so. Lokální klientská knihovna pro aplikace používající Classic server byla přejmenována na libfbembed.so.

Přejmenované soubory a moduly

Platforma	Modul	Firebird 1.0	Firebird 1.5	Poznámky
Všechny	Proměnné prostředí	INTERBASE INTERBASE_LOCK INTERBASE_MSG INTERBASE_TMP	FIREBIRD FIREBIRD_LOCK FIREBIRD_MSG FIREBIRD_TMP	Kořenový adresář instalace Umístění souboru zámků Umístění souboru hlášení Umístění dočasných souborů
Všechny	Bezpečnostní databáze	Isc4.gdb	security.fdb	
Všechny	Soubor zpráv	Interbase.msg	firebird.msg	
Všechny	Soubor protokolu serveru	interbase.log	firebird.log	
Všechny	ODS verze	10	10.1	Nová ODS (10.1). nezpůsobuje žádnou nekompatibilitu s předchozí ODS, ale verze není automaticky aktualizována. Jak Firebird 1.0 tak 1.5 mohou pracovat s databázemi s ODS 10.0 a 10.1. Nicméně je stále doporučeno provést backup/restore pro migraci databází mezi jednotlivými verzemi serveru.

Linux	Classic server program	gds_inet_server	fb_inet_server	
Linux	Classic manažer zámků	ib_lock_mgr	fb_lock_mgr	
Linux	Superserver kontroler	ibmgr.bin	fbmgr.bin	
Linux	Superserver program	ibserver	fbserver	
Linux	Konfigurační soubor	isc_config	firebird.conf	
Linux	Klientská knihovna	libgds.so	libfbclient.so libfbembed.so	Thread-safe vzdálený klient a klient TCP/IP lokální smyčky pro SuperServer Lokální klient (jednouživatelský, není thread-safe) pro Classic
Linux	Symlink klientské knihovny pro kompatibilitu	N/A	libgds.so	
Windows	Guardian	ibguard.exe	fbguard.exe	
Windows	SuperServer program	ibserver.exe	fbserver.exe	Bez podpory více procesorů.
Windows	Classic program	N/A	fb_inet_server.exe	Lokální připojení není podporováno. TCP/IP, NetBEUI OK. S podporou více procesorů.
Windows	Klientská knihovna	gds32.dll	fbclient.dll	Nástroje serveru v1.5, a všechny nové aplikace vyžadují pouze knihovnu fbclient.dll. Kompatibilita se staršími aplikacemi viz poznámka u gds32.dll.
Windows	Konfigurační soubor	ibconfig	firebird.conf	
Windows	Lokální IPC port	InterBaseIPI	FirebirdIPI	S implicitní konfigurací serveru se nelze lokálně připojit pomocí straší klientské knihovny (gds32.dll). V případě potřeby je možné konfigurovat server pro použití staršího jména pomocí firebird.conf.
Windows	Implicitní klíč Registry	HKLM\/SOFTWARE\Borland\InterBase	HKLM\SOFTWARE\Firebird Project\Firebird Server\Instances	Cesta je uložena v parametru "DefaultInstance". Tedy již žádný klíč "CurrentVersion", a parametr "RootDirectory" je nahrazen "DefaultInstance".
Nová jména služeb na Windows jsou "Firebird Guardian – DefaultInstance" a "Firebird Server – DefaultInstance"				

Kompatibilita

Interní struktura databáze (On-disk structure - ODS)

Interní struktura databáze u Firebirdu 1.5 je ODS 10.1. Tato malá aktualizace ODS byla nezbytná pro:
- tři nové indexy na systémových tabulkách
- drobné změny BLR u dvou systémových spouští
- vylepšené kódování RDB$TRIGGER_TYPE.

Některá vylepšení vyžadující změny ODS byly odloženy do verze 2. Zatím je možné přenášet vaše databáze z Firebirdu 1.0.x přímo. Nicméně vytvořte a otestujte zálohu vašich databází před jejich přenosem pod Firebird 1.5.

InterBase™ databáze

Pokud si plánujete „pohrát" s Firebirdem za použití vašich existujících databází InterBase se záměrem vrátit se později k InterBase, proveďte nezbytná bezpečnostní opatření vytvořením zálohy databází pomocí odpovídající verze programu gbak z InterBase. Pro převod databází pod Firebird 1.5 proveďte obnovu z vytvořené zálohy pomocí programu gbak z Firebirdu 1.5.
Příručka Operations Guide z InterBase® 6.0 beta documentation set nebo kniha InterBase a Firebird vydaná nakladatelstvím Computer Press obsahuje syntaxi programu gbak pro zálohování a obnovu databází.

Databáze z IB 7.x a pravděpodobně i IB 6.5 nemusí po migraci na Firebird 1.5 pracovat korektně, pokud používají specifické vlastnosti těchto verzí InterBase.

Názvy souborů a jejich umístění

V této verzi Firebirdu má mnoho souborů nová jména, v důsledku postupného odstraňování názvů zděděných z InterBase ® 6. Přečtěte si pozorně sekci Jména souborů a jejich umístění obsahující podrobnosti o změnách a další doporučení.

Současně běžící servery

Změny provedené v názvech některých systémových objektů umožňují aby byl FB 1.5 instalován a používán na systémech kterých již je instalována InterBase nebo Firebird 1.0.x. Na Windows používá, FB 1.5 také odlišný klíč Registry. Pokud nakonfigurujete servery pro práci na odlišných síťových portech, je možné současně provozovat více instancí serveru na jediném počítači, nebo provozovat FB 1.5 současně s IB nebo FB 1.0.x.

Návrat k Firebirdu 1.0.x

Z důvodu velkého množství opravených chyb, může se chování databází změnit po jejich přenosu z Firebirdu v.1.5 na v.1.0.x. Obzvláště, pokud vytvoříte primární, cizí nebo jedinečné klíče jako pojmenovaná omezení, nebudou implicitní názvy indexů kompatibilní s verzí 1.0.x. Čtěte budoucí README soubory pro podrobnosti o všech případných problémech zaznamenaných tak jak budou odhaleny.

Kompatibilita s Linuxem

Z důvodu historických problémů s překladačem GNU C++, vyžaduje Firebird 1.5 pro Linux vyšší verze knihovny glibc než předchozí verze. To naneštěstí znamená, že je těžké předpovědět schopnost jednotlivých distribucí Linuxu pro úspěšnou instalaci a provozování Firebirdu. Následující tabulka snad pomůže. Nicméně uvítáme jakékoliv další informace. Podělte se prosím o vaše zkušenosti s těmito a dalšími distribucemi v konferenci firebird-devel.

Distribuce	Verze	Classic	Super Server
Red Hat	7.x	Ne	Ne
	8.0	Ano	Ano
	9.0	Ano	Ano
Mandrake	8.x	Ne	Ne
	9.0, 9.1, 9.2	Ano	Ano
SuSE	7.3	Ano – Instalujte balíky libgcc-3.2-44.i586.rpm & libstdc++-3.2-44.i586.rpm před instalací Firebird 1.5.	Není známo
	8.0	Ano	Ano
	8.1	Ano	Není známo

Rozšíření jazyka

DATOVÉ TYPY

(1.5) Nový nativní SQL datový typ

BIGINT
Přesné číslo dle standardu SQL99, 64-bitové se znaménkem, bez desetinných míst. Dostupný pouze v SQL Dialektu 3.

Příklad(y)
I.)
```
DECLARE VARIABLE VAR1 BIGINT;
```
II.)
```
CREATE TABLE TABLE1 (FIELD1 BIGINT);
```

METADATA

(1.5) Rozšíření pojmenovaných integritních omezení

Dmitry Yemanov

Indexy které interně zajišťují realizaci pojmenovaných integritních omezení lze nyní uživatelsky pojmenovat.

Dříve, ačkoliv bylo možné vytvořit pojmenované PRIMARY, FOREIGN KEY a UNIQUE omezení, byly identifikátory automaticky vytvářených indexů definovány systémem, např., RDB$FOREIGN13, a nebylo je možné změnit. Pokud nejsou použita pojmenovaná omezení, chová se systém nadále uvedeným způsobem.

Nicméně nové rozšíření jazyka SQL umožňuje aby:
a) Systémem vytvořený index obdržel automaticky stejný název jako integritní omezení pro které je vytvořen.
b) Index který je vytvářen pro pojmenované nebo nepojmenované integritní omezení obdržel explicitně definovaný uživatelský identifikátor, a aby byl volitelně vytvořen jako SESTUPNÝ. POZNÁMKA V současnosti nelze pro integritní omezení specifikovat již existující index.

Syntaxe
```
...
[ADD] CONSTRAINT [<identifikátor-omezení>]
<typ-omezení> <definice-omezení>
[USING [ASC[ENDING] | DESC[ENDING]] INDEX <jméno-indexu>]
```

Upozornění: Ujistěte se, že cizí i jim odpovídající primární klíče mají **shodný způsob řazení** (DESC | ASC).

Příklady
I.) Pojmenované omezení a explicitně pojmenovaný index

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

```
COMMIT;
```
Následující příkaz vytvoří primární klíč nazvaný PK_ATEST a sestupný index pro toto omezení nazvaný IDX_PK_ATEST:

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

II.) Alternativně k i) uvedeném výše:

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

III.) Tento příkaz vytvoří tabulku ATEST s primárním klíčem PK_ATEST. Index pro omezení bude rovněž nazván PK_ATEST.

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

(1.5) Spouště pro více akcí

Dmitry Yemanov

Spouště byly rozšířeny o možnost obsluhovat více operací nad řádky, včetně podmíněného obsloužení.

Syntaxe

```
CREATE TRIGGER název FOR table
  [ACTIVE | INACTIVE]
  {BEFORE | AFTER} <více-akcí>
  [POSITION číslo]
AS tělo-spouště
```

<více-akcí> ::= <jediná-akce> [OR < jediná-akce > [OR < jediná-akce >]]
< jediná-akce > ::= {INSERT | UPDATE | DELETE}

Příklady
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
…;
```

Změny ODS
Kódování sloupce RDB$TRIGGER_TYPE (tabulka RDB$TRIGGERS) bylo rozšířeno pro podporu komplexních akcí spouští. Více informací naleznete v dokumentu readme.universal_triggers.txt v adresáři /doc/sql.extensions v CVS projektu.

Poznámky:
1. Spouště pro jedinou akci jsou na úrovni ODS plně kompatibilní s FB 1.0.
2. Kódování RDB$TRIGGER_TYPE je závislé na pořadí, tzn., BEFORE INSERT OR UPDATE a BEFORE UPDATE OR INSERT jsou kódovány odlišně, ačkoliv mají shodnou sémantiku a jsou zpracovávány shodným způsobem.
3. Ve spouštích pro více akcí jsou dostupné jak OLD tak NEW kontextové proměnné. Pokud kontext aktivace spouště nedovoluje některý druh kontextových proměnných (např.. OLD kontext pro operaci INSERT), jsou hodnoty všech těchto kontextových proměnných nastaveny na NULL. Pokud je jim v kódu spouště přiřazena hodnota v nesprávném kontextu, je vyvolána výjimka.
4. K zjištění typu operace ve které je spoušť aktuálně prováděna lze zjistit prostřednictvím nových logických kontextových proměnných INSERTING/UPDATING/DELETING. (Viz níže.)

(I.5) RECREATE VIEW

Pokud datový pohled neexistuje, je chování příkazu shodné s příkazem CREATE VIEW. Pokud datový pohled existuje, RECREATE VIEW se pokusí pohled zrušit a vytvořit zcela nový objekt. RECREATE VIEW selže, pokud je rušený pohled v používání.
Příkaz používá stejnou syntaxi jako CREATE VIEW.

(I.5) CREATE OR ALTER {TRIGGER | PROCEDURE }

Příkaz který vytvoří novou spoušť nebo uloženou proceduru (pokud již neexistuje) nebo ji změní (pokud již existuje) a opětovně ji přeloží. Syntaxe příkazu CREATE OR ALTER zachovává stávající závislosti a oprávnění.

Syntaxe je stejná jako u CREATE TRIGGER nebo CREATE PROCEDURE, s rozdílem dodatečné klauzule "OR ALTER".

(I.5) Hodnoty NULL v jedinečných indexech a integritních omezeních

Dmitry Yemanov

Nově je možné v souladu se standardem SQL-99 definovat omezení UNIQUE nebo jedinečný index pro sloupce které nemají definováno omezení NOT NULL. Buďte opatrní, pokud plánujete návrat k Firebirdu 1.0.x nebo InterBase.

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

kde <unique column list> může obsahovat jeden nebo více sloupců bez atributu NOT NULL, pokud je <unique specification> UNIQUE nebo UNIQUE INDEX index-name. Povšimněte si, že všechny sloupce v definici PRIMARY KEY musí být stále deklarovány jako NOT NULL.

Omezení povoluje pouze takové řádky, pro které je vyhledávací podmínka (i) nebo (ii) vyhodnocena jako pravdivá, podle následující logiky:

i) Pokud <unique specification> definuje PRIMARY KEY, pak bude vyhledávací podmínka:

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

ii) Jinak bude vyhledávací podmínka:

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

V takovém případě podmínka UNIQUE nebude True pokud (SELECT UCL FROM TN) vrátí dva řádky u kterých jsou shodné všechny korespondující ne-null segmenty.

Omezení povoluje existenci pouze takových řádků pro které je výše uvedená vyhledávací podmínka vyhodnocena jako pravdivá. Pro unikátní index nebo pro UNIQUE omezení budou dvě sady sloupců považovány za odlišné a tudíž povolené pokud:

a) Obě sady obsahují pouze null, nebo
b) Obsahuje alespoň jeden pár korespondujících sloupců kde jeden není null, a druhý je null nebo obsahuje jinou ne-null hodnotu.

Příklady

UNIQUE omezení:

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

nebo 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, null povoleny */
INSERT INTO t VALUES (1, 2); /* atak jako ne-nulls */
INSERT INTO t VALUES (1, NULL); /* a kombinace */
INSERT INTO t VALUES (NULL, NULL); /* ok, všechny null páry jsou odlišné */
```

ale

```
INSERT INTO t VALUES (1, NULL); /* selže, protože jsou všechny korespondující ne-null segmenty stejné */
```

To znamená, že ***PRIMARY KEY omezení nedovoluje NULL*****,** zatímco UNIQUE omezení a jedinečné indexy povolují libovolné množství NULL. Pro vícesloupcové množiny řádků (SELECT UCL FROM TN), je použito společné pravidlo pro NULL, tzn. (1, NULL) je odlišné od (NULL, 1) a jeden (NULL, NULL) je odlišný od jakéhokoliv dalšího (NULL, NULL).

DSQL

(1.5) Výrazy a proměnné jako argumenty procedur

Dmitry Yemanov

Volání EXECUTE PROCEDURE jméno-procedury(<seznam-argumentů>) a
SELECT <výstupní-sloupce> FROM jméno-procedury (<seznam-argumentů>) nyní akceptují lokální proměnné (v PSQL) a výrazy (v DSQL a PSQL) jako argumenty.

(1.5) Nové konstrukce pro podmíněné výrazy

Arno Brinkman

a) CASE

Umožňuje specifikovat konečnou hodnotu sloupce na základě výsledku skupiny samostatných podmínek.
Syntaxe

```
<case-výraz> ::=
   <case-zkratka>  | <case-specifikace>

<case-zkratka> ::=
   NULLIF <levá-závorka> <hodnota-výrazu> <čárka> <hodnota-výrazu> <pravá-závorka>
  | COALESCE <levá-závorka> <hodnota-výrazu> { <čárka> <hodnota-výrazu> }... <pravá-závorka>

<case-specifikace> ::=
   <jednoduchý-case>  | < podmínkový-case>

<jednoduchý-case> ::=
  CASE <hodnota-výrazu>  <jednoduchá-when-klauzule>...
   [ <else-klauzule> ]
  END

<porovnávací-case> ::=
  CASE <podmínková-when-klauzule>...
   [ <else-klauzule> ]
  END

<jednoduchá-when-klauzule> ::= WHEN <when-operand> THEN <výsledek>
< podmínková -when-klauzule> ::= WHEN <podmínka> THEN <výsledek>
<when-operand> ::= <hodnota-výrazu>
<else-klauzule> ::= ELSE <výsledek>
<výsledek> ::= <výsledkový-výraz>  | NULL
<výsledkový-výraz> ::= <hodnota-výrazu>
```

Příklady

I.) jednoduchý

```
SELECT
    o.ID,
    o.Description,
    CASE o.Status
      WHEN 1 THEN 'confirmed'
      WHEN 2 THEN 'in production'
```

```
      WHEN 3 THEN 'ready'
      WHEN 4 THEN 'shipped'
      ELSE 'unknown status ''' || o.Status || ''''
    END
FROM Orders o;
```

II.) podmínkový

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

b) COALESCE

Umožňuje definovat hodnotu pomocí více výrazů, přičemž první výraz v posloupnosti který není vyhodnocen jako NULL definuje návratovou hodnotu.

Formát

<case-zkratka> ::=
| COALESCE <levá-závorka> <hodnota-výrazu> { <čárka> <hodnota-výrazu> }... <pravá-závorka>

Syntaktická pravidla

I.) COALESCE (V1, V2) je shodný s následující <case-specifikací>:
CASE WHEN V1 IS NOT NULL THEN V1 ELSE V2 END

II.) COALESCE (V1, V2,..., Vn), pro n >= 3, je shodný s následující <case-specifikací>:
CASE WHEN V1 IS NOT NULL THEN V1 ELSE COALESCE (V2,...,Vn) END

Příklady

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

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

c) NULLIF

Vrací NULL pro podvýraz, pokud má specifickou hodnotu, jinak vrací hodnotu podvýrazu.

Formát

<case-zkratka> ::=
NULLIF <levá-závorka> <hodnota-výrazu> <čárka> <hodnota-výrazu> <pravá-závorka>

Syntaktická pravidla
NULLIF (V1, V2) je shodný s následující <case-specifikací>:
CASE WHEN V1 = V2 THEN NULL ELSE V1 END

Příklad

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

(1.5) Body návratu transakce dle SQL99

Nickolay Samofatov

Uživatelské body návratu (savepoints) (někdy také označované jako *vnořené transakce*) poskytují standardní metodu pro zpracování chyb obchodní logiky bez nutnosti odvolat celou transakci. Vlastnost je dostupná pouze v DSQL.

Pro definici bodu návratu v rámci transakce ke kterému je možné provést obnovu stavu (rollback) slouží příkaz SAVEPOINT.

```
SAVEPOINT <identifikátor>;
```

<identifikátor> definuje jméno vytvořeného bodu návratu. Po vytvoření bodu návratu lze pokračovat ve zpracování, potvrdit transakci, odvolat celou transakci, nebo obvolat změny provedené po definovaném bodu návratu.

Jména bodů návratu musí být v rámci transakce jedinečná. Pokud vytvoříte bod návratu se shodným jménem jako již existující bod návratu v rámci aktuální transakce, bude původní bod návratu zrušen.

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

Tento příkaz provede následující operace:
- Zruší (rollback) změny vytvořené v rámci transakce po vytvoření bodu návratu.
- Zruší všechny body návratu vytvořené po specifikovaném bodu návratu. Specifikovaný bod návratu zůstane zachován, takže je možné provést návrat ke stejnému bodu návratu více než jednou. Předcházející body návratu jsou rovněž zachovány.
- Uvolní všechny implicitní i explicitní zámky na řádcích vytvořené po specifikovaném bodu návratu. Ostatní transakce které požadovaly přístup k řádkům uzamčeným po vytvoření bodu návratu musí nadále čekat do ukončení transakce jejím potvrzením nebo odvoláním. Ostatní transakce které doposud přístup k uzamčeným řádkům nepožadovaly mohou k těmto řádkům okamžitě přistupovat. Poznámka: Toto chování se může změnit v budoucích verzích produktu.

Protokol změn vytvářený pro body návratu může zkonzumovat významné množství paměti serveru, obzvláště pokud aktualizujete stejný záznam vícekrát v rámci jediné transakce. Pro uvolnění systémových zdrojů blokovaných již nepotřebnými body návratu použijte příkaz RELEASE SAVEPOINT.

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

Příkaz RELEASE SAVEPOINT zruší bod návratu <identifikátor> z kontextu transakce. Pokud není specifikováno klíčové slovo ONLY, jsou rovněž zrušeny všechny body návratu vytvořené po specifikovaném bodu návratu.

Příklad použití bodů návratu

```
create table test (id integer);
commit;
insert into test values (1);
commit;
insert into test values (2);
savepoint y;
delete from test;
select * from test; -- nevrací žádné řádky
rollback to y;
select * from test; -- vrací dva řádky
rollback;
select * from test; -- vrací jeden řádek
```

Interní body návratu
Firebird používá interně systém bodů návratu automaticky vytvářených na úrovni transakce pro realizaci odvolání transakce. Pokud vykonáte příkaz ROLLBACK, jsou všechny změny provedené v rámci transakce odstraněny s pomocí bodu návratu na úrovni transakce, a transakce je potvrzena. Tato logika redukuje množství „úklidových prací" způsobených odvoláním transakce.

Pokud je množství změn provedených v rámci bodu návratu vytvářeného na úrovni transakce velmi velké (10^4-10^6 řádků), uvolní server automaticky bod návratu na úrovni transakce (pro úsporu zdrojů systému) a použije v případě odvolání transakce pro odstranění změn mechanizmus TIP (stránky stavu transakcí) stránek. Pokud předpokládáte velké množství změn v rámci transakce, můžete použít TPB (Transaction Parameter Block - blok parametrů transakce) příznak isc_tpb_no_auto_undo, aby nebyl bod návratu na úrovni transakce vůbec vytvářen.

Body návratu a PSQL
Implementace uživatelských bodů návratu v PSQL (spouště a uložené procedury) by porušilo pravidlo atomicity příkazů, včetně příkazů volání procedur. Firebird poskytuje pro PSQL mechanizmus zpracování výjimek pro odstranění změn provedených uloženými procedurami a spouštěmi. Každý SQL/PSQL příkaz je zpracováván pod systémem automatických, interních bodů návratu, kde je buď celý příkaz úspěšně proveden nebo jsou VŠECHNY změny odstraněny v případě vyvolání výjimky. Každý PSQL blok ošetření výjimek (WHEN blok) je rovněž zapouzdřen systémem automatických bodů návratu.

(1.5) Explicitní zámky

Nickolay Samofatov

Dodatečná volitelná klauzule WITH LOCK poskytuje omezenou možnost explicitního zamykání pro obezřetné použití v případech kdy zpracovávaná množina řádek je a) velmi malá (ideálně jediný řádek) a b) pod naprostou kontrolou aplikačního kódu.

Pesimistické zámky jsou u Firebirdu zapotřebí opravdu jen velmi vzácně, a celý mechanizmus by měl být dobře pochopen dříve než začnete o jejich použití uvažovat.

Syntaxe

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

Pokud klauzule WITH LOCK uspěje, zajistí zámek na vybraných řádcích a zabrání ostatním transakcím v získání oprávnění k zápisu těchto řádků, nebo na nich závislých objektech, až do ukončení transakce.

Pokud je použita klauzule FOR UPDATE, je zámek aplikován na každý řádek postupně, tak jak jsou postupně načítány do vyrovnávací paměti na straně serveru. Je tedy možné že uzamčení řádků které se jeví jako úspěšné může následně selhat, v případě že selže uzamknutí načítaného řádku v důsledku uzamčení řádku jinou transakcí.

Je nezbytné porozumět efektům izolace transakcí a dalších parametrů transakcí dříve než se pokusíte implementovat explicitní uzamykání ve vaší aplikaci.

Konstrukce SELECT... WITH LOCK je dostupná v DSQL a PSQL. Může uspět pouze u příkazů SELECT nejvyšší úrovně nad jedinou tabulkou. Není dostupná pro podvýběry ani spojované množiny. Nelze ji specifikovat spolu s operátorem DISTINCT, klauzulí GROUP BY nebo jakoukoliv agregační operací. Nelze ji použít v definici datového pohledu nebo s datovým pohledem, ani externí tabulkou nebo výstupem z dotazovatelné uložené procedury.

Pochopení klauzule WITH LOCK

Pro každý řádek který spadá pod příkaz s explicitním zamykáním vrací server buď nejaktuálnější potvrzenou verzi řádku bez ohledu na stav databáze v okamžiku vykonání příkazu, nebo výjimku.

Případné čekání a ohlášení konfliktu závisí na parametrech transakce specifikovaných v bloku TPB.

TPB mód	*Chování*
Isc_tpb_consistency	Explicitní zámky jsou přepsány implicitními nebo explicitními zámky na úrovni tabulky, a jsou tedy ignorovány.
Isc_tpb_concurrency + isc_tpb_nowait	Pokud je řádek změněn transakcí která byla potvrzena po zahájení transakce která se snaží získat explicitní zámek, nebo tento řádek změnila aktivní transakce, je okamžitě vyvolána výjimka o konfliktní aktualizaci.
Isc_tpb_concurrency + isc_tpb_wait	Pokud je řádek změněn transakcí která byla potvrzena po zahájení transakce která se snaží získat explicitní zámek, je okamžitě vyvolána výjimka o konfliktní aktualizaci. Pokud je řádek vlastněn aktivní transakcí (pomocí explicitního zámku nebo normálního optimistického zámku pro zápis), čeká transakce požadující explicitní zámek na výsledek blokující transakce, a po jejím ukončení se opětovně pokusí zámek získat. To znamená že pokud transakce potvrdila změnu řádku, je vyvolána výjimka o konfliktní aktualizaci.
Isc_tpb_read_committed + isc_tpb_nowait	Pokud je řádek vlastněn aktivní transakcí (pomocí explicitního zámku nebo normálního optimistického zámku pro zápis), je okamžitě vyvolána výjimka o konfliktní aktualizaci.

Pokračování →

TPB mód	*chování*
isc_tpb_read_committed + isc_tpb_wait	Pokud je řádek vlastněn aktivní transakcí (pomocí explicitního zámku nebo normálního optimistického zámku pro zápis), čeká transakce požadující explicitní zámek na ukončení blokující transakce, a po jejím ukončení se opětovně pokusí zámek získat. V tomto módu nikdy nemůže dojít k vyvolání výjimky konfliktní aktualizace.

Pokud příkaz UPDATE narazí na řádek uzamčený jinou transakcí, je v závislosti na módu transakce buď vyvolána výjimka nebo transakce čeká na výsledek blokující transakce. Chování serveru je stejné jako by byl řádek modifikován transakcí která řádek uzamkla. Chyba hlášená při konfliktu pesimistických zámků nevrací žádnou speciální hodnotu gdscode.

Server zaručuje, že všechny řádky vrácené příkazem s explicitními zámky jsou skutečně uzamknuty, a skutečně vyhovují výběrové podmínce v WHERE klauzuli, pokud vyhledávací podmínka nezávisí na jiných tabulkách prostřednictvím spojování, podvýběrů apod. je rovněž zaručeno, že řádky které nevyhovují výběrové podmínce nejsou příkazem uzamknuty. Server NEZARUČUJE, že neexistují neuzamčené řádky které vyhovují výběrové podmínce. Tato situace může nastat, pokud souběžná transakce potvrdí změny v průběhu zpracování příkazu s explicitními zámky.

Server uzamyká řádky v okamžiku načtení. To má důležité důsledky, pokud uzamykáte více řádků najednou. Mnoho knihoven pro práci s Firebirdem standardně načítá řádky v blocích ("načítání do vyrovnávací paměti"). Mnoho přístupových knihoven neumožňuje zpřístupnit řádky z naposled přečteného bloku řádků, kde došlo k chybě.

Klausule FOR UPDATE umožňuje zabránit dopředného načítání řádků do vyrovnávací paměti, volitelně s klauzulí OF <názvy-sloupců> pro poziční aktualizaci. Je rovněž možné, že vaše přístupová knihovna umožňuje nastavit velikost načítané bloku řádků na 1. To umožní zpracovat aktuálně uzamčený řádek před načtením a uzamčením následujícího řádku, nebo ošetření chyb bez nutnosti odvolat transakci.

Odvolání změn k implicitnímu nebo explicitnímu bodu návratu uvolní všechny zámky vytvořené v rámci daného bodu návratu (a případně následujících bodů návratu), ale neinformuje o této změně čekající transakce. Aplikace by ovšem neměly na tomto chování záviset, protože se může v budoucnu změnit.

Ačkoliv mohou explicitní zámky pomoci zabránit vzniku a/nebo ošetřit výjimečné konflikty aktualizací, dojde v případě že strategie zamykání nebude správně navržena a implementována k výraznému nárůstu chyb vzájemného zablokování. Většina aplikací vůbec nevyžaduje explicitní zámky. Explicitní zámky mají význam především pro jsou (1) odstranění nákladného ošetření konfliktních aktualizací v aplikacích s velkou zátěží a (2) pro údržbu integrity objektů mapovaných do relační databáze v distribuovaném prostředí. Pokud váš způsob použití explicitních zámků nespadá do některé z uvedených kategorií, pak jste zvolili špatný způsob pro dosažení požadovaného cíle.

Explicitní zámky jsou pokročilou vlastností, nezneužívejte je ! Ačkoliv taková řešení mohou být velmi důležitá pro webové servery obsluhující tisíce současných zápisů, nebo pro ERP/CRM systémy pracující v prostředí velkých firem, většina aplikací v takových podmínkách nepracovat nepotřebuje.

Příklady

I.) (jednoduchý)

```
SELECT * FROM DOCUMENT WHERE ID=? WITH LOCK
```

II.) (více řádků, zpracovány po jednom s DSQL kurzorem)

```
SELECT * FROM DOCUMENT WHERE PARENT_ID=? FOR UPDATE WITH LOCK
```

(1.5) Vylepšené zpracování agregací

Arno Brinkman

Původně bylo možné vytvářet skupiny pouze pro pojmenované sloupce. U Firebirdu 1.0 již bylo možné vytvářet skupiny pro UDF výrazy. Další rozšíření ve verzi 1.v zpracování agregačních funkcí a klauzule GROUP BY nyní umožňují vytvářet skupiny pomocí pořadí sloupců v definici výstupu (počítáno od 1, směrem zleva doprava, podobně jako u klauzule ORDER BY) nebo různých výrazů.
POZNÁMKA: V současnosti nejsou povoleny všechny výrazy. Například není dovoleno spojování řetězců.

Syntaxe Group By

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

group_by_seznam  : group_by_položka [, group_by_seznam];

group_by_položka : název_sloupce
                 | pozice (ordinal)
                 | udf
                 | group_by_funkce;

group_by_funkce  : numerická_funkce
                 | řetězcová_funkce
                 | case_výraz
                 ;

numerická_funkce        : EXTRACT '(' timestamp_part FROM hodnota ')';

řetězcová_funkce        :  SUBSTRING '(' hodnota FROM pos_short_integer ')'
                        | SUBSTRING '(' hodnota FROM pos_short_integer FOR
nezáporný_short_integer ')'
                        | KW_UPPER '(' hodnota ')'
                        ;
```

group_by_položka nemůže být odkazem na žádnou agregační funkci (včetně těch které jsou použity uvnitř výrazů) z toho samého kontextu.

HAVING
Klauzule HAVING připouští použití pouze agregačních funkcí nebo platných výrazů které jsou součástí klauzule GROUP BY. Původně bylo možné použít sloupce, které nebyly součástí klauzule GROUP BY a použití neplatných výrazů.

ORDER BY
Pokud je klauzule použita v rámci příkazu s agregací, připouští ORDER BY pouze použití platných výrazů které jsou agregačními funkcemi, nebo výrazů které jsou součástí klauzule GROUP BY. Dříve bylo možné použít neplatné výrazy.

Agregační funkce podvýběrech
Nyní je možné použít agregační funkce nebo výrazy obsahující klauzuli GROUP BY v podvýběru (vnořený select).

Příklady

```
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
```

Současné použití agregačních funkcí z různých kontextů
Nyní lze ve výrazech použít agregační funkce z odlišných kontextů.

Příklad
```
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)
```

Poznámka! Tento dotaz vrací výsledek i v FB1.0, ale tento výsledek je CHYBNÝ!

Podpora podvýběrů v agregačních funkcích
Nyní je možné použít SELECT výraz vracející jedinou hodnotu uvnitř agregační funkce.

Příklad
```
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
```

Zapouzdřené agregační funkce
Je možné použít agregační funkci uvnitř jiné agregační funkce, pokud je vnitřní agregační funkce z nižšího kontextu (viz příklad uvedený výše).

Vytváření skupin podle pořadí (pořadového čísla)
Použití pořadového čísla výstupní položky při specifikaci klauzule GROUP BY „kopíruje" výraz z definice výstupních polí do definice skupiny (stejně jako se tomu v takovém případě děje u ORDER BY klauzule). To znamená, že pokud pořadové číslo odkazuje na podvýběr, je tento podvýběr vykonán alespoň dvakrát.

(1.5) ORDER BY klauzule může definovat výraz a umístění NULL hodnot

Nickolay Samofatov

Klauzule ORDER BY umožňuje definovat platný výraz pro třídění výstupu. Pokud je výraz tvořen jediným číslem, je vyhodnocen jako pořadové číslo výstupního sloupce, stejně jako dříve.
Umístění null hodnot ve výstupu lze kontrolovat pomocí klauzule pro umístění null hodnot. Výsledek může být seřazen tak, že hodnoty null jsou umístěny na začátek (NULLS FIRST) nebo konec (NULLS LAST) seřazených ne-null hodnot.
Pokud umístění null hodnot není explicitně specifikováno, je NULLS LAST.

Syntaxe

```
SELECT ... FROM .... ORDER BY order_seznam ....;
order_seznam : order_položka [, order_seznam];
order_položka : <výraz> [směr_řazení] [umístění_null_hodnot]
směr_řazení : ASC | DESC;
umístění_null_hodnot : NULLS FIRST | NULLS LAST;
```

Omezení
- Pokud je specifikováno NULLS FIRST, nelze pro třídění použít index.
- Výsledek řazení podle hodnot vrácených UDF nebo uložené procedury bude nepředpověditelný, pokud hodnoty nelze použít pro určení jejich logického pořadí.
- Počet vyvolání procedury z třídění založeného na UDF nebo uložené procedury není předpověditelný, bez ohledu zda je použita definice třídění přímo pomocí výrazu nebo pořadového čísla určující výraz v seznamu výstupních položek.
- Klauzule pro třídění výsledku slučování výstupu (union) může používat pouze pořadová čísla výstupních položek.

Příklady
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 (Jazyk uložených procedur a spouští)

(1.5) EXECUTE STATEMENT

Alex Peshkov

Rozšíření PSQL které přebírá řetězec znaků obsahující platný dynamický SQL příkaz, a vykoná ho stejně jako by byl vykonán pomocí DSQL.
Dostupná v uložených procedurách a spouštích.

Syntaxe může mít tři podoby.

Syntaxe 1
Vykoná <řetězec> jako SQL operaci která nevrací žádné řádky dat, viz. INSERT, UPDATE, DELETE, EXECUTE PROCEDURE nebo jakýkoliv DDL příkaz mimo CREATE/DROP DATABASE.

```
EXECUTE STATEMENT <řetězec>;
```

Příklad

```
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
```

Syntaxe 2
Vykoná <řetězec> jako SQL operaci vracející jediný řádek dat. Tento formát příkazu umožňuje vykonat pouze singleton SELECT operace.

```
EXECUTE STATEMENT <řetězec> INTO :proměnná_1, […, :proměnná_n] ;
```

Příklad

```
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
```

Syntaxe 3
Vykoná <řetězec> jako SQL operaci vracející více řádků dat. Tento formát příkazu může zpracovat jakoukoliv formu SELECT příkazu.

```
FOR EXECUTE STATEMENT <řetězec> INTO :proměnná_1, …, :proměnná_n DO
   <složený-příkaz>;
```

Příklad

```
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
```

Další poznámky k příkazu EXECUTE STATEMENT

DSQL řetězec příkazu 'EXECUTE STATEMENT' nesmí obsahovat žádné parametry v žádné variantě syntaxe. Všechny substituce ve statické části SQL příkazu musí provedeny před vykonáním příkazu EXECUTE STATEMENT.

Tato vlastnost je určena jen k obezřetnému použití, a před jejím použitím je nutné vzít v úvahu všechny faktory její činnosti. Příkaz EXECUTE STATEMENT by měl být používán pouze a pouze tehdy, pokud není možné použít jinou metodu, nebo tyto metody pracují hůře než EXECUTE STATEMENT.

EXECUTE STATEMENT je hned několika způsoby potenciálně nebezpečný:
1. Není žádná možnost ověřit korektnost syntaxe zapouzdřeného SQL příkazu.
2. Kontrola závislosti neodhalí žádné tabulky a sloupce které byly zrušeny.
3. Zpracování bude pomalejší, protože zapouzdřený SQL příkaz musí být zkompilován (prepare) před každým vykonáním.
4. U návratových hodnot je striktně kontrolován jejich datový typ k zabránění nepředvídaných výjimek přetypování. Například řetězec '1234' lze konvertovat na celé číslo, ale řetězec'abc' způsobí chybu konverze.
5. Pokud má uložená procedura zvláštní přístupová oprávnění k některým objektům, dynamický příkaz v řetězci předaném EXECUTE STATEMENT je nedědí. Oprávnění jsou omezena na oprávnění přidělená uživateli který uloženou proceduru spustil.

(1.5) Nové kontextové proměnné

Dmitry Yemanov

CURRENT_CONNECTION

a

CURRENT_TRANSACTION

Každá z těchto kontextových proměnných vrací systémový identifikátor aktivního připojení, respektive aktuálního transakčního kontextu. Hodnota je typu INTEGER. Proměnné jsou dostupné v DSQL a PSQL. Protože jsou tyto hodnoty uloženy v hlavičce databáze, jsou tyto hodnoty při obnově databáze resetovány.

Syntaxe

```
CURRENT_CONNECTION
CURRENT_TRANSACTION
```

Příklady

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

ROW_COUNT

Vrací celé číslo (integer) udávající počet řádků ovlivněných posledním vykonaným DML příkazem. Dostupná v PSQL, v kontextu uložené procedury nebo spouště. V současnosti vrací hodnotu 0 pro SELECT příkazy.

Syntaxe

```
ROW_COUNT
```

Příklad

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

Poznámka: Tuto proměnnou nelze použít pro zjištění počtu řádků ovlivněných příkazem EXECUTE STATEMENT.

SQLCODE a GDSCODE

Každá z kontextových proměnných vrací celé číslo (integer) představující chybový kód aktuální výjimky. Dostupná v PSQL, v rámci konkrétního bloku ošetření výjimek (WHEN blok). Mimo blok WHEN je hodnota obou proměnných 0.

Proměnná GDSCODE vrací číselnou reprezentaci chybového kódu GDS (ISC), např. pro '335544349L' vrací 335544349.

Blok 'WHEN SQLCODE' nebo 'WHEN ANY' zpracuje nenulovou hodnotu proměnné SQLCODE a vrací hodnotu 0 pro GDSCODE. Pouze blok 'WHEN GDSCODE' zpracuje nenulovou hodnotu proměnné GDSCODE (a vrátí nulovou hodnotu v SQLCODE). V případě vyvolání uživatelské výjimky obsahují obě proměnné hodnotu 0, bez ohledu na typ WHEN bloku.

Syntaxe

```
SQLCODE
GDSCODE
```

Příklad

```
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
```

Rovněž si přečtěte níže uvedenou stať VYLEPŠENÍ ZPRACOVÁNÍ VÝJIMEK, a dokument README.exception_handling v adresáři firebird2/doc/sql.extensions CVS stromu projektu Firebird.

INSERTING

UPDATING

DELETING

Tři pseudo-logické výrazy které je možno testovat k určení typu prováděné DML operace. Dostupné pouze v PSQL, pouze ve spouštích. Určeny pro použití s univerzálními spouštěmi (viz stať METADATA, uvedená výše).

Syntaxe

```
INSERTING
UPDATING
DELETING
```

Příklad

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

(1.5) Vylepšení zpracování výjimek v PSQL

Dmitry Yemanov

Obecná syntaxe pro příkaz EXCEPTION v PSQL je nyní:

```
  EXCEPTION [jméno [hodnota]];
```

Vylepšení ve verzi 1.5 vám umožňuje:

1) Definovat text hlášení pojmenované výjimky za běhu.
2) Reinicializovat (re-raise) zachycenou výjimku v rámci rozsahu bloku zpracování výjimek.
3) Získat číselný kód chyby pro zachycenou výjimku.

1) Definice textu hlášení výjimky za běhu

Syntaxe
```
EXCEPTION <jméno-výjimky> <hodnota-textu-hlášení>;
```

Příklady
i)
```
EXCEPTION E_EXCEPTION_1 'Error!';
```
ii)
```
EXCEPTION E_EXCEPTION_2 'Wrong type for record with ID=' || new.ID;
```

2) reinicializace výjimky
Poznámka – Nemá žádný efekt mimo blok zpracování výjimky.

Syntaxe
```
EXCEPTION;
```

Příklady
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) Kódy běhových chyb
Viz SQLCODE / GDSCODE (výše).

(1.5) Příkaz LEAVE | BREAK

Ukončuje zpracování smyčky, a způsobí přechod zpracování na příkaz následující za koncovým příkazem END smyčky. Dostupný pro pouze pro jazykové konstrukce WHILE, FOR SELECT a FOR EXECUTE, jinak bude hlášena chyba při překladu. Standardní klíčové slovo LEAVE standardu SQL-99 nahrazuje existující BREAK. Dostupné v uložených procedurách a spouštích.

Syntaxe

```
LEAVE;
```

Příklady

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

(ii)
```
WHILE (<podmínka>) DO
  BEGIN
    <příkazy>;
    WHEN ... DO
      LEAVE;
  END
```

POZNÁMKA Příkazy LEAVE | BREAK a EXIT lze nyní použít i v spouštích.

(1.5) Platné PLAN specifikace lze nyní použít v spouštích

Ignacio J. Ortega

Doposud byly spouště obsahující PLAN konstrukt odmítnuty překladačem. Nyní lze specifikovat platný prováděcí plán, který bude použit.

(1.5) Prázdné BEGIN..END bloky

Dmitry Yemanov

Prázdné BEGIN..END bloky v PSQL modulech jsou nyní legální. Například nyní lze definovat „prázdné" moduly jako

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

(1.5) Deklarace a inicializace lokálních proměnných v jediném příkazu

Claudio Valderrama

Zjednodušená syntaxe pro lokální proměnné nyní umožňuje deklarovat a zároveň inicializovat proměnné jediným příkazem.

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

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

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

(1.5) SELECT FIRST nyní akceptuje hodnotu 0 jako argument

FB 1.5 dovoluje použít hodnotu 0 jako argument pro FIRST. Bude vrácena prázdná množina.

Vybere prvních *m* řádků vybrané výstupní množiny. Volitelná SKIP klauzule způsobí přeskočení prvních *n* řádků, a vrácení datové množiny *m* řádků počínaje od *n + 1*. V zjednodušené formě jsou *m* an *n* celá čísla, ale je možné použít libovolný výraz který je vyhodnocen jako celé číslo. V GDML (nikoliv už v DSQL nebo SQL) lze použít i identifikátor který lze vyhodnotit.

Pokud je specifikován výraz, musí být uzavřen v závorkách, které jsou jinak volitelné.

Hodnoty *m* a *n* mohou být rovněž parametrické, např. SKIP ? * FROM ATABLE vrací zbytek datové množiny po přeskočení *n* řádků, přičemž *n* je předáno prostřednictvím proměnné (parametru) "?". SELECT FIRST ? COLUMNA, COLUMNB FROM ATABLE vrací pouze prvních *m* řádků.

Klauzule FIRST je rovněž volitelná, tzn. můžete v příkazu použití SKIP bez FIRST k získání výsledku který neobsahuje úvodní řádky specifikované pomocí SKIP.

Dostupné v SQL a DSQL mimo explicitně definovaných výjimek.

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

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

SELECT FIRST 5 DISTINCT FIELD FROM MYTABLE;
```

Dvě pasti příkazu SELECT FIRST

1. Příkaz

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

zruší všechny řádky v tabulce. Ouha! Podvýběr vyhodnotí prvních deset řádků pro výmaz, ty jsou vymazány, přesune se na dalších deset…a pořád dokola dokud není tabulka prázdná.
Pozor!

2. Dotazy jako:

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

nepracují tak jak lze předpokládat, protože optimalizace prováděná serverem transformuje souvztažný WHERE...IN (SELECT...) predikát na souvztažný EXISTS predikát. Je evidentní, že v takovém případě FIRST N nemá žádný význam:

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

Vylepšení znakových sad

Přidáno v 1.5

- Přidáno třídění WIN1251_UA (pro Ruštinu a Ukrajinštinu) pro znakovou sadu WIN1251.
- Korektní převod na velká písmena pro znakovou sadu WIN1251.

- Přidáno třídění ISO_HUN (pro Maďarštinu) pro znakovou sadu SO8859_2.

Nové znakové sady (žádné nebinární metody třídění)
Blas Rodriguez Somoza
- DOS737 PC Greek
- DOS775 PC Baltic
- DOS858 varianta Cp850 s Euro znakem (€)
- DOS862 PC Hebrew
- DOS864 PC Arabic
- DOS866 MS-DOS Russian
- DOS869 IBM Modern Greek
- WIN1255 Windows Hebrew
- WIN1256 Windows Arabic
- WIN1257 Windows Baltic
- ISO8859_3 Latin 3 (Esperanto, Maltese, Pinyi, Sami, Croatian a ostatní)
- ISO8859_4 Latin 4 (Baltic, Greenlandic, Lappish)
- ISO8859_5 Cyrillic
- ISO8859_6 Arabic
- ISO8859_7 Greek
- ISO8859_8 Hebrew
- ISO8859_9 Turkish
- ISO8859_13 Baltic

Přidáno v 1.0

- Přidáno case-insensitivní třídění pro Maďarštinu, vytvořil a otestoval Sandor Szollosi (ssani@freemail.hu).

- Firebird nyní podporuje znakovou sadu ISO8859-2 (s tříděním pro Češtinu).

ROZŠÍŘENÍ JAZYKA PŘEVZATÁ Z FIREBIRDU 1.0.x

Pro vaše pohodlí uvádíme i rozšíření uvedená verzi Firebird 1.0.x.

(1.0) CURRENT_USER a CURRENT_ROLE

Tyto dvě kontextové proměnné odkazují na jméno uživatele a (pokud je použita[1]) role v kontextu aktuálního připojení.

```
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 je DSQL synonymem pro proměnnou USER která je definována SQL standardem. Obě proměnné jsou identické.

[1] Pokud trváte na použití databáze InterBase v.4.x nebo 5.1 s Firebirdem, není ROLE podporována, takže current_role bude NONE (požadováno standardem SQL pokud není definována explicitní role) i v případě že uživatel roli definoval při přihlášení. Pokud používáte IB 5.5, IB 6 nebo Firebird, je předaná ROLE kontrolována. Pokud role neexistuje, je nastavena na NONE bez ohlášení chyby.

To znamená že Firebird nikdy nevrací neplatnou roli v proměnné CURRENT_ROLE, protože bude nastavena na NONE. Toto chování je odlišné od InterBase, kde je interně nastavena nesmyslná hodnota, ačkoliv není viditelná pro SQL.

(1.0) DROP GENERATOR

Umožňuje z databáze zrušit nepotřebné generátory. Prostor bude uvolněn pro další použití po následujíc obnově databáze ze zálohy. Dostupný v SQL a DSQL.

```
DROP GENERATOR <jméno-generátoru>;
```

(1.0) GROUP BY UDF

Nyní je možné definovat agregovaný SELECT specifikací skupiny na výstup UDF.
Např.

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

Vedlejším efektem možnosti skupinování podle UDF je že ačkoliv nemůžete volat vestavěné funkce Firebirdu v GROUP BY, můžete nyní s pomocí jednoduché zapouzdřovací funkce používat konstrukce jako:

```
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

Tento nový DDL příkaz dovoluje vytvořit novou uloženou proceduru se stejným názvem jako existující procedura, nahrazením této existující procedury, aniž by bylo nutné tuto proceduru nejprve zrušit.
Syntaxe je identická s příkazem CREATE PROCEDURE.
Dostupný v SQL a DSQL.

(1.0) RECREATE TABLE

Tento nový DDL příkaz dovoluje vytvořit novou tabulku se stejným názvem jako existující tabulka, nahrazením této existující tabulky, aniž by bylo nutné tuto nejprve zrušit. Syntaxe je identická s příkazem CREATE TABLE.

Příkaz RECREATE TABLE nezachovává data z původní tabulky.

Dostupný v SQL a DSQL.

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

Interní funkce implementující funkci ANSI SQL SUBSTRING(). Vrací proud bajtů počínaje bajtem na <*pos*> a všemi následujícími bajty až do konce řetězce. Pokud je specifikován parametr FOR <*length*>, vrací maximálně <*length*> bajtů.

První argument může být libovolný výraz, konstanta nebo identifikátor který lze vyhodnotit jako řetězec.
<*pos*> musí být vyhodnotitelný jako celé číslo (integer).
<*pos*> začíná hodnotou 1, jako u ostatních SQL příkazů.
Ani <*pos*> ani <*length*> nemohou být parametrizovány.

Protože <pos> a <length> jsou bajtové pozice, může být identifikátor binární blob, nebo textový blob sub_type 1 používající jednobajtovou znakovou sadu. Funkce v současnosti nezpracovává korektně znakové sady pro Čínštinu (maximálně 2 bajty/znak) nebo Unicode (maximálně 3 bajty/znak). Pro rětězcový argument (na rozdíl od blobu) se funkce vypořádá s libovolnou znakovou sadou.

Dostupná v SQL a DSQL.

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

Věnujte pozornost rovněž následující sekci věnované Externím Funkcím (UDF), obsahující informace o změnách v externí funkci pro výběr podřetězce ve standardní UDF knihovně.

(1.5) Vylepšení zpracování jednořádkového komentáře

Dmitry Yemanov

Jednořádkové komentáře mohou být na libovolné pozici v řádku, nejen na začátku.
Ve verzi 1.5 může být znak "--" použit pro komentář na konci řádku v skriptu, uložené proceduře, spouští nebo DSQL příkazu. Rovněž může být použit pro „odkomentování" nežádoucích částí příkazů. Všechny znaky po značce "--" do konce řádku (RC nebo LF) budou ignorovány.

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

(1.0) Nová značka pro komentář

Claudio Valderrama

Pro použití ve skriptech, DSQL, uložených procedurách a spouštích.

Příklad

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

Tato nová značka pro komentář může být použita pro „odkomentování" jednoho řádku kódu ve skriptu, DDL/DML příkazu, uložené proceduře nebo spoušti.

Logika pro ignorování znaků je následující:

1. Přeskočí '--' pokud je prvními dvěma znaky na řádku (hned po konci řádku - LF na Linux/Unix, CRLF na Windows)
2. Přeskakování znaků pokračuje do nalezení následujícího ukončení řádku

Tato logika není určeno pro současné použití s logikou blokových komentářů (/* komentář */). Jinak řečeno, nepoužívejte styl komentáře '--' v bloku komentáře a nepoužívejte blokový styl komentáře na řádku s '--'.

INTERAKTIVNÍ ISQL SEZENÍ: Při práci s **isql** mějte na paměti. **isql** akceptuje části příkazů v samostatných pokračovacích blocích, a zobrazuje výzvu 'CON>' dokud není zadán ukončovací znak (normálně znak '**;**'). Pokud napíšete dvojici '--' na začátku pokračovacího řádku, logika ignorování znaků skončí stiskem Enter. Může dojít k chybám, pokud zadáte pokračovací řádek a očekáváte že bude rovněž ignorován.

Problém s isql spočívá ve speciálních příkazech zpracovávaných pouze isql. Pokud nejsou rozpoznány díky neobratnému umístění "--", jsou předány ke zpracování serveru. Protože server nerozumí speciálním příkazům isql jako je SET a SHOW, ohlásí chybu.

(1.0) Alter Trigger již nadále nezvyšuje čítač změn tabulky

Pokud počet změn metadat jakékoliv jednotlivé tabulky dosáhne maxima 255, stane se databáze nedostupnou. Pro resetování počítadla změn je nutné provést zálohu a obnovu ze zálohy. Účelem tohoto omezení je vynutit vyčištění databáze v okamžiku kdy struktura databáze prodělala mnoho změn, nikoliv znemožnit používání serveru.

Pokud byl dříve byl změněn příznak spouště ACTIVE|INACTIVE prostřednictvím příkazu ALTER TRIGGER, byl pokaždé zvýšen čítač změn asociované tabulky. To ovlivnilo použitelnost zakázání a opětovného povolení zpracování spouští pro běžné operace, protože vedlo k rychlému navyšování počítadla změn.

Nová rezervovaná slova

Následující **nová klíčová slova Firebirdu** by měla být přidána do seznamu rezervovaných klíčových slov publikovaných pro InterBase 6.0.1.

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

Následující klíčová slova jsou rezervována pro budoucí použití:

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

Následující klíčová slova jsou byla rezervována v Firebirdu 1.0 a již nadále nejsou rezervována v Firebirdu 1.5:

BREAK	DESCRIPTOR	FIRST
IIF	SKIP	SUBSTRING

Následující nerezervovaná slova jsou rozpoznána jako klíčová slova u verze 1.5 pokud jsou použita v jim příslušejícím kontextu:

COALESCE	DELETING	INSERTING
LAST	LEAVE	LOCK
NULLIF	NULLS	STATEMENT
UPDATING	USING	

Následující nová klíčová slova **jsou rezervována v InterBase 6.5 a 7** (nejsou rezervována v Firebirdu) a měla by být rovněž považována za vyhrazená, z důvodu kompatibility:

BOOLEAN	FALSE	GLOBAL
PERCENT	PRESERVE	ROWS
TEMPORARY	TIES	TRUE

Vlastnosti ISQL

Řádkový editor prostředí isql

Mark O'Donohue

Do příkazového řádku isql byla přidána podpora historie příkazů (jako Unix readline). Nyní můžete používat klávesy pro šipky nahoru a dolů pro listování příkazy zadanými v rámci sezení v isql.

Uživatelsky definované funkce

V knihovně ib_udf

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

Juan Guerrero

Doplní zprava řetězec *instring* by přidáním znaku *padcharacter*s do specifikované délky *length*. Vstupní řetězec může být dlouhý až 32766 bajtů. Celková délka nesmí překročit 32765 bajtů.

Deklarace

```
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

Doplní zleva řetězec *instring* by přidáním znaku *padcharacter*s do specifikované délky *length*. Vstupní řetězec může být dlouhý až 32766 bajtů. Celková délka nesmí překročit 32765 bajtů.

Deklarace

```
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

Tato funkce obsahovala chybu, která způsobovala prohození argumentů x a y. Funkce má vracet logaritmus y o základu x, ale ve skutečnosti vracela logaritmus x o základu y. Funkce byla opravena. Pokud jste funkci používali ve svých aplikacích, PROVĚŘTE SVŮJ APLIKAČNÍ KÓD! Buď vrací chybné výsledky, nebo v určitém bodě úmyslně prohazuje předávané parametry pro dosažení správného výsledku.

V knihovně fbudf

1. Funkce *NVL a *NULLIF byly zachovány pro zpětnou kompatibilitu, ale jsou nyní považovány za zastaralé díky implementaci nových interních funkcí CASE, COALESCE a NULLIF.

2. Knihovna fbudf není schopna pracovat s řetězci delšími než 32Kb –1 bajt. Toto omezení může mít nepříjemné následky pokud použijete spojování řetězců před jejich předáním UDF funkci s řetězcovými argumenty. Pokud je celková délka větší než limit, je výsledek nepředvídatelný. Funkce může vrátit nesmyslný výsledek, nebo provést ilegální operaci.

3. Pokud přenášíte databázi která byla vytvořena pod Firebirdem 1.0.x, a definovali jste funkce **truncate** nebo **round** z knihovny fbudf, nebudou tyto deklarace pod Firebirdem 1.5 funkční, protože názvy vstupních bodů těchto funkcí byly změněny. Je nutné tyto deklarace zrušit a opětovně vytvořit pomocí skriptu fbudf.sql z podadresáře /UDF.

Nový konfigurační soubor – firebird.conf

Kořenový adresář Firebirdu

Kořenový adresář vaší instalace Firebirdu je používán mnoha způsoby, jak během instalace, stejně tak jako atribut pro rutiny serveru, configurační parametry a klienty které jsou na něm závislí. Protože existuje několik způsobů pro instruování serveru jakým způsobem má získat hodnotu tohoto atributu, musí si být vývojáři i administrátoři vědomi pořadí, v jakém server při startu vyhodnocuje jednotlivé varianty, aby byla správně vyhodnocena skutečná hodnota tohoto atributu.

Win32 Superserver a Classic (jak server, tak klient):
1) Proměnná prostředí FIREBIRD
2) Parametr RootDirectory v souboru firebird.conf
3) Registrační databáze Windows:
   HKLM\SOFTWARE\ Firebird Project\Firebird Server\Instances\DefaultInstance
   a zde hledá položku DefaultInstance
4) Adresář o jednu úroveň výše než adresář ve kterém je uložen program serveru

Win32 Embedded:
1) Proměnná prostředí FIREBIRD
2) Parametr RootDirectory v souboru firebird.conf
3) Adresář, ve kterém je uložena knihovna fbembed.dll (přejmenovaný fbclient.dll)

Linux Classic:
1) Proměnná prostředí FIREBIRD
2) Parametr RootDirectory v souboru firebird.conf
3) Implicitní instalační cesta (/opt/firebird)

Linux Superserver:
1) Proměnná prostředí FIREBIRD
2) RootDirectory parameter in firebird.conf
3) Adresář o jednu úroveň výše než adresář ve kterém je uložen program serveru (adresář uložení serveru je získán pomocí symbolického odkazu "/proc/self/exe", pokud je podporován)
4) Inplicitní instalační cesta (/opt/firebird)

Parametry

Většina parametrů má implicitní hodnoty. Názvy i hodnoty parametrů jsou na Linuxu case-sensitivní, zatímco na Windows není velikost písmen rozlišována. Pro změnu hodnoty parametru na jinou než implicitní hodnotu, vymažte znak pro komentář (#) a změňte hodnotu. Konfigurační soubor je možné měnit za běhu serveru. Pro aktivaci provedených změn je nezbytné restartovat server.
Zápisy v konfiguračním souboru mají následující tvar:

jméno_parametru *hodnota*

- jméno_parametru je řetězec neobsahující žádné mezery, a označující proměnnou serveru která má být konfigurována.

- *hodnota* je číslo, logická hodnota (1=True, 0=False) nebo řetězec znaků specifikující hodnotu parametru.

Parametry souborového systému

RootDirectory

Řetězec znaků, absolutní cesta ke kořenovému adresáři instalace v lokálním systému souborů. Pokud si nepřejete, aby byla jinak automaticky detekovaná cesta přepsána, měl by tento parametr zůstat zakomentován.

DatabaseAccess

Podpora pro přístup k databázím prostřednictvím aliasu. V předchozích verzích se server připojoval k databázím umístěným na lokálním souborovém systému prostřednictvím předané absolutní cesty k databázi. Tento parametr umožňuje omezit přístup buď pouze na databáze specifikované pomocí aliasu, nebo na databáze umístěné ve specifikovaném adresářovém stromu.
Hodnota parametru DatabaseAccess může být None, Restrict nebo Full.

Full (implicitní) dovoluje přistupovat k databázím uloženým kdekoliv v lokálním souborovém systému.

None dovoluje serveru připojit pouze databáze specifikované v konfiguračním souboru **aliases.conf**.

Restrict dovoluje specifikovat umístění připojitelných databází prostřednictvím specifikace kořenových adresářů. K definici jedné nebo více přípustných lokací zadejte jeden nebo více kořenových adresářů oddělených středníkem.
Například,
Unix: `/db/databases;/userdir/data`
Windows: `D:\data`
Relativní cesty jsou chápány jako relativní ke kořenovému adresáři serveru. Například, pokud je na Windows kořenovým adresářem C:\Program Files\Firebird, pak následující hodnota omezí přístup pouze na databáze uložené v adresáři C:\Program Files\Firebird\userdata a jeho podadresářích:

```
DatabaseAccess = Restrict userdata
```

POZNÁMKA Stínování (shadowing) databáze – nynější zpracování DatabaseAccess obsahuje chybu, kvůli které je nezbytné použít volbu Restrict pokud na serveru používáte stínování.

ExternalFileAccess

Původně *external_file_directory* v konfiguračním souboru isc_config/ibconfig, ale syntaxe byla změněna.

Poskytuje tři úrovně zabezpečení EXTERNÍCH SOUBORŮ (textové soubory s pevnou strukturou, k kterým je přistupováno jako k databázovým tabulkám). Hodnota parametru je typu řetězec, a může nabývat hodnot None, Full nebo Restrict.

None (implicitní) zakazuje jakékoliv použití externích souborů na serveru.

Restrict umožňuje omezit umístění externích souborů na specifické kořenové adresáře. K definici jedné nebo více přípustných lokací pro externí soubory zadejte jeden nebo více kořenových adresářů oddělených středníkem.
Například,
Unix: `/db/extern;/mnt/extern`
Windows: `C:\ExternalTables`
Relativní cesty jsou chápány jako relativní ke kořenovému adresáři serveru. Například, pokud je na Windows kořenovým adresářem C:\Program Files\Firebird, pak následující hodnota omezí přístup pouze

na externí soubory uložené v adresáři C:\Program Files\Firebird\userdata\ExternalTables a jeho podadresáře:

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

Full umožňuje přistupovat k externím souborům umístěných kdekoliv v systému.

Přečtěte si **VAROVÁNÍ** pod následujícím parametrem, UdfAccess.

UdfAccess

Původně *external_function_directory* v konfiguračním souboru isc_config/ibconfig, ale syntaxe byla změněna.

Nahrazuje nejen název původního parametru, ale rovněž způsob specifikace hodnoty. Účelem této změny je umožnit volitelné úrovně ochrany pro knihovny uživatelsky definovaných funkcí, které jsou častým cílem zlomyslných útoků. Hodnota parametru UdfAccess může být None, Restrict nebo Full.

Restrict (implicitní) zachovává funkčnost parametru **external_function_directory** z Firebirdu 1.0, k omezení lokace knihoven externích modulů na specifikované adresáře v systému soborů. K definici jedné nebo více přípustných lokací pro UDF knihovny, BLOB filtry a rozšiřující znakové sady zadejte jeden nebo více kořenových adresářů oddělených středníkem.
například,
Unix: `/db/extern;/mnt/extern`
Windows: `C:\ExternalModules`
Relativní cesty jsou chápány jako relativní ke kořenovému adresáři serveru. Například, pokud je na Windows kořenovým adresářem C:\Program Files\Firebird, pak následující hodnota omezí přístup pouze na rozšiřující moduly uložené v adresáři C:\Program Files\Firebird\userdata\ExternalModules:

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

None znemožní jakékoliv použití uživatelsky deinovaných externích knihoven.

Full dovoluje pracovat s externími knihovnami umístěnými kdekoliv v systému souborů.

VAROVÁNÍ: Vyhněte se takové specifikaci uživatelských stromů adresářů pro parametry UdfAccess a ExternalFileAccess, které by sdílely stejné kořenové adresáře. Implicitní nastavení je bezpečné. Pokud provedete vlastníl nastavení, a nevytvoříte separátní adresářové stromy pro oba parametry, může být server velmi snadno donucen k spuštění neautorizovaného kódu. Příklad nastavení, kterému je třeba se vyhnout:

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

UdfAccess & ExternalFileAccess zde mají stejný podstrom, `/bad_dir/files`, kam někdo může umístit externí soubor `/bad_dir/files/hackudf.so` a spustit svůj vlastní kód.

Parametry zdrojů

CpuAffinityMask

Původně *cpu_affinity* v konfiguračním souboru isc_config/ibconfig

U Firebird SuperServeru na Windows na SMP počítačích existuje problém, kdy operační systém neustále prohazuje proces databázového serveru tam a zpět mezi procesory. To má velmi negativní dopad na

výkon. Tento parametr by měl být použit na SMP systémech s Windows pro pevné přidělení procesu databázového serveru pouze jediném procesoru.

> VÝSTRAHA Server Firebird do verze 1.5 včetně nemusí podporovat Hyperthreading dostupný na Windows u některých novějších modelů základních desek. Aby nedocházelo k problémům s balancováním, může být nezbytné zakázat hyperthreading na úrovni BIOSu.

CpuAffinityMask akceptuje jediné celé číslo, CPU masku.

Příklad

```
CpuAffinityMask = 1
```
Běh pouze na prvnímu CPU (CPU 0).

```
CpuAffinityMask = 2
```
Běh pouze na druhému CPU (CPU 1).

```
CpuAffinityMask = 3
```
Běh na obou CPU.

Výpočet hodnoty masky
Tento parametr lze použít pro přiřazení Firebirdu (CPU affinity) jakémukoliv jednotlivému procesoru, nebo (Classic serveru) libovolnou kombinaci procesorů instalovaných v systému.
Považujte procesory jako vektor číslovaný od 0 do *n*-1, kde *n* je počet instalovaných procesorů a *i* je pořadové číslo konkrétního procesory ve vektoru. M je další pole, obsahující hodnotu masky pro každý vybraný procesor. Hodnota A je součtem hodnot ve vektoru M.
Použijte následující vzorec pro zjištění hodnot M a výpočet hodnoty masky A:

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

Například, pro první a čtvrtý procesor (procesor 0 a procesor 3) bude výpočet následující:

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

DeadlockTimeout

Původně *deadlock_timeout* v konfiguračním souboru isc_config/ibconfig

Počet vteřin (integer) po které bude manažer zámků čekat po zjištění konfliktu, než budou odstraněny zámky mrtvých procesů a proveden další cyklus detekce vzájemného zablokování. Za normálních podmínek detekuje server vzájemné zablokování okamžitě. Deadlock timeout je použit pouze v mimořádných situacích.

Implicitní hodnota 10 vteřin je optimální pro většinu situací. Nastavení nižší hodnoty nemusí nezbytně zlepšit rychlost kterou jsou hlášeno vzájemné zablokování. Pokud je hodnota příliš nízká, může mít za následek zbytečné prohledávání, které sníží výkon systému.

DefaultDbCachePages

Původně *database_cache_pages* v konfiguračním souboru isc_config/ibconfig

Implicitní počet (integer) databázových stránek alokovaných serverem v paměti pro každou databázi. Nastavená hodnota platí pro celý server, a může být změněna na úrovni databáze.
Implicitní hodnota pro SuperServer je 2048 stránek. Pro Classic je implicitní hodnota 75.

SuperServer a Classic používají vyrovnávací paměť odlišně. U SS se vyrovnávací paměť sdílí pro všechna připojení k dané databázi, zatímco Classic alokuje vyrovnávací paměť pro každé připojení zvlášť.

EventMemSize

Celé číslo reprezentující počet bajtů paměti rezervovaných pro manažer událostí. Implicitní hodnota je 65536 (64 Kb).

LockAcquireSpins

Původně *lock_acquire_spins* v konfiguračním souboru isc_config/ibconfig

Relevantní pouze pro SMP stroje na kterých je provozován Classic server. U Classic server může k tabulce zámků přistupovat v daném okamžiku pouze jediný klient. Přístup k tabulce zámků je řízen mutexem, a klientské procesy mohou tento mutex požadovat podmíněně nebo nepodmíněně. Pokud je žádost podmíněná, pak selže a musí být opakována. Pokud je nepodmíněná, bude proces čekat až do uspokojení požadavku. Parametr LockAcquireSpins definuje počet podmíněný pokusů o získání mutexu, než bude proveden nepodmíněný požadavek.
Celé číslo. Implicitní hodnota je 0 (nepodmíněný). Není žádná doporučená minimální nebo maximální hodnota.

LockHashSlots

Původně *lock_hash_slots* v konfiguračním souboru isc_config/ibconfig

Tento parametr slouží k vyladění hash-tabulky zámků. Pod velkou zátěží je možné zlepšit propustnost systému zvýšením počtu hash slotů, a tedy zkrácením hash řetězců. Celé číslo —doporučena prvočísla. Implicitní hodnota je 101.

LockGrantOrder

Původně *lock_grant_order* v konfiguračním souboru isc_config/ibconfig

Pokud připojení vyžaduje zámek na objektu, získává blok požadavku na zámek který specifikuje požadovaný objekt a požadovanou úroveň zámku. Každý uzamčený objekt má vlastní blok zámku. Bloky požadavků jsou propojeny na tyto bloky zámků prostřednictvím bloku požadavku který je získal, nebo který je požaduje.
Parametr LockGrantOrder je logická hodnota (Boolean). Implicitní hodnota (1=True) určuje, že požadavky mají být vyřizovány metodou FIFO. Opačné nastavení (0=False) emuluje chování InterBase v3.3, a přiřazuje zámky ihned jak jsou k dispozici (podle typu požadavku). Toto nastavení může vést k „vyhladovění" některých požadavků na zámky.

LockMemSize

Tento celočíselný parametr reprezentuje počet bajtů sdílené paměti alokované pro manažer zámků. Pro Classic server určuje hodnota LockMemSize počáteční velikost, která může narůstat až do vyčerpání paměti (*"Lock manager is out of room")*. Pokud je v systému aktivní velké množství připojení nebo jsou vyrovnávací paměti velké, předejdete zvýšením hodnoty tohoto parametru vzniku chyb.

U architektury SuperServer se velikost paměti alokované pro manažer zámků nemění.
Implicitní hodnota pro Linux a Solaris je 98304 bajtů (96 Kb). Na Windows je to 262144 bajtů (256 Kb).

LockSemCount

Celočíselný parametr specifikující počet semaforů dostupných pro meziprocesovou komunikaci (IPC). Implicitní hodnota je 32. Změňte hodnotu tohoto parametru u systémů bez vláken pro zvýšení nebo snížení počtu dostupných semaforů.

SortMemBlockSize

Tento parametr umožňuje konfigurovat (v bajtech) velikost každého paměťového bloku použitého pro třídění v paměti. Implicitní hodnota je 1 MB. Lze nastavit libovolnou hodnotu až do hodnoty definované parametrem SortMemUpperLimit (viz níže).

SortMemUpperLimit

Maximální množství paměti (v bajtech) alokované pro třídění v paměti. Implicitní hodnota je 67108864 bajtů (64 MB) pro SuperServer a 8388608 bajtů (8 MB) pro Classic server.

> VÝSTRAHA pamatujte, že zvýšení hodnoty u Classic serveru odpovídajícím způsobem zvýší paměťové nároky každého aktivního připojení..

Komunikační parametry

ConnectionTimeout

Původně *connection_timeout* v konfiguračním souboru isc_config/ibconfig

Počet vteřin čekání před upuštěním od požadavku na vytvoření spojení. Implicitně 180.

DummyPacketInterval

PŮvodně *dummy_packet_interval* v konfiguračním souboru isc_config/ibconfig

Počet vteřin (integer) po které server čeká u neaktivního spojení, než zašle prázdný paket pro potvrzení existence spojení.
NEPOUŽÍVEJTE TENTO PARAMETR na Win32 serveru používajícím TCP/IP klienty. Může způsobit nestálý nárůst používané neodstránkovávané paměti jádra,což může vést k zamrznutí nebo pádu Windows na straně klienta, tak jak je vysvětleno v:
http://support.microsoft.com/default.aspx?kbid=296265

Bez ohledu na Win32-s-TCP/IP, toto je jediná metoda pro detekci a odpojení neaktivních klientů pokud je použit některý z protokolů NamedPipes (NetBEUI), XNET nebo IPC. Na POSIX systémech nejsou známy žádné problémy.

Standardně používá Firebird parametr SO_KEEPALIVE TCP/IP soketů pro sledování aktivity připojení. Pokud vám nevyhovuje dvouhodinový standardní timeout, nastavte požadovaným způsobem váš operační systém:

- Na systémech typu UNIX modifikujte obsah /proc/sys/net/ipv4/tcp_keepalive_*.
- Na Windows postupujte podle instrukcí z článku:

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

Implicitní hodnota by měla být 0 – nikoliv 60 jako starší implicitní hodnota u Firebirdu 1.0 a většiny vývojových verzí Firebirdu 1.5. Pro systémy u kterých vyžadujete testování pomocí prázdných paketů, je doporučeno nastavena hodnotu 60.

RemoteServiceName

Implicitně = gds_db

RemoteServicePort

Tyto dva parametry umožňují přepsat název služby TCP/IP nebo číslo TCP/IP portu používané pro příjem požadavku na vytvoření databázového připojení, pokud se některá z hodnot liší od instalované implicitní hodnoty (gds_db/tcp 3050).

Změňte pouze jeden z těchto parametrů, nikoliv oba. Parametr RemoteServiceName je testován jako první, a je hledána služba odpovídajícího jména v souboru services. Pokud je taková služba nalezena, je použita hodnota portu nastavená pro RemoteServicePort. Pokud služba daného jména není nalezena, je použit implicitníčíslo portu 3050.

POZNÁMKA Pokud je číslo portu uvedeno přímo v řetězci specifikace připojení, má přednost před hodnotou specifikovanou pro RemoteServicePort.

RemoteAuxPort

Chování zděděné po InterBase, kdy byl pro zasílání zpráv aplikacím používán náhodně vybraný TCP/IP port byl častým zdrojem problémů na síti a konfliktů s firewally, a za určitých okolností mohlo dojít i k pádu serveru. Tento parametr umožňuje nastavit jediný TCP port pro veškerou zpětnou komunikaci. Implicitní hodnota (0) zachovává původní náhodný výběr portu. Pro vyhrazení specifického portu pro rozesílání zpráv definujte celočíselnou představující číslo volného portu.

RemoteBindAddress

Standardně se mohou klienti připojovat přes jakékoliv síťové rozhranní síťového uzlu serveru. Tento parametr umožňuje svázat službu Firebirdu s jedinou síťovou kartou (NIC). Server tak nebude přijímat spojení z žádného jiného síťového rozhranní. To umožňuje vyřešit problém na některých podsítích, kdy server zpracovává provoz skrz několik síťových rozhranní.
Hodnota parametru je řetězec, reprezentující platnou IP adresu ve standardním formátu. Implicitní hodnota je prázdný řetězec (nesvázáno s konkrétním rozhranním).

TcpRemoteBufferSize

Server načítá data dopředu, a může klientovi předat více řádků dat v jediném paketu. Čím je velikost paketu větší, tím více řádků je v rámci jednoho přenosu odesláno. Tento parametr použijte – s maximální opatrností a při plném pochopení jeho dopadu na efektivitu síťového přenosu – pokud potřebujete zvětšit nebo zmenšit velikost vyrovnávacích pamětí pro přenos. Hodnota parametru ovlivňuje komunikaci klienta i serveru.
Hodnotou parametru je celé číslo (velikost paketu v bajtech) v rozsahu 1448 až 32768. Implicitní hodnota je 8192.

Pouze POSIX

LockSignal

Integer, UNIX signál používaný pro meziprocesovou komunikaci. Default: 16

RemoteFileOpenAbility

POUŽÍVAT POUZE S MAXIMÁLNÍ OPATRNOSTÍ

Boolean parametr který, pokud je nastaven na True, umožňuje serveru otevřít databázové soubory, které se nacházejí na souborovém systému připojeném prostřednictvím sítě (NFS). Protože je takový souborový systém mimo kontrolu lokálního systému, jde o *velice riskantní vlastnost*, která by neměla být použita pro práci s databázemi, na jejichž obsahu vám záleží.
Implicitní hodnota je 0 (False, zakázáno) a měli by jste ji tak nechat, pokud si nejste zcela vědomi všech důsledků.

TcpNoNagle

Původně *tcp_no_nagle* v konfiguračním souboru isc_config/ibconfig

Na Linuxu minimalizuje síťová komunikační knihovna standardně množství fyzických zápisů použitím vyrovnávací paměti před skutečným odesláním dat, s použitím interního algoritmu (implementovaném jako TCP_NODELAY příznak soketového spojení) známého jako Naglův Algoritmus. Ten je navržen k odstranění problému s malými pakety, nazývanými tinigramy, na pomalých sítích.
Při instalaci SuperServeru na Linuxu je standardně TCP_NODELAY aktivní (hodnota 0). Na pomalých sítích může jeho potlačení ve skutečnosti zlepšit výkon přenosu. Pozor na obrácenou logiku – nastavte parametr na True pro potlačení TCP_NODELAY a False pro jeho zapnutí.
U Firebirdu do verze 1.5 včetně je tato vlastnost aktivní pouze u SuperServeru.

Pouze Windows

CreateInternalWindow

Lokální Windows protokol používá skryté okno pro meziprocesovou komunikaci mezi lokálním klientem a serverem. Toto IPC okno je vytvořeno serverem při jeho startu pokud je hodnota parametru CreateInternalWindow 1 (implicitní). Nastavte hodnotu na 0 pokud chcete provozovat server bez tohoto okna, a tedy bez možnosti využít lokálního přenosového protokolu. S vypnutým lokálním protokolem je možné provozovat více instancí serveru současně.

DeadThreadsCollection

Nastavení plánovače vláken na Windows. Tento parametr definuje počet cyklů změny priority (viz níže uvedený parametr PrioritySwitchDelay) které plánovač provede před zrušením (nebo uzavřením) vlákna.
Okamžité zrušení (nebo uzavření) pracovního vlákna by vyžadovalo práci se semaforem a blokující volání, které má nezanedbatelnou režii. Místo toho udržuje plánovač kolekci vláken. Pokud vlákno ukončí práci, je označeno jako čekající. Čekající vlákno je zrušeno (nebo uzavřeno) po *n* iteracích smyčky plánovače, kde *n* je hodnota parametru DeadThreadsCollection.
Pro servery obsluhující velké množství připojení – 100 a více – je nutné hodnotu tohoto parametru zvýšit z implicitních 50.

GuardianOption

Logická hodnota určující, zda má Guardian restartovat server po každém jeho abnormálním ukončení. Implicitní hodnota 1 (True) způsobí restart serveru. Pro potlačení restartu nastavte parametr na hodnotu 0 (False).

IpcMapSize

Původně *server_client_mapping* v konfiguračním souboru ibconfig

Velikost klientské části paměťového okna jednoho klienta pro meziprocesovou komunikaci (IPC) používaného pro Lokální přenosový protokol na Windows. Nemá ekvivalent na jiných platformách. Celé číslo v rozsahu 1024 až 8192. Implicitní hodnota je 4096.
Zvětšení velikosti okna může zlepšit výkon při přenosu rozsáhlých datových množin, nebo velkých BLOB položek.

> POZNÁMKA Tuto hodnotu již nelze modifikovat v dialogu programu Guardian.

IpcName

Implicitní hodnota: FirebirdIPI

Jméno sdílené oblasti paměti používané pro přenos dat v lokálním protokolu.
U Firebirdu 1.5 není implicitní hodnota – FirebirdIPI – kompatibilní se staršími verzemi Firebirdu ani s InterBase. V případě potřeby použijte původní hodnotu InterBaseIPI pro zajištění kompatibility.

MaxUnflushedWrites

Tento parametr byl přidán do verze 1.5 pro ošetření chyby v operačním systému Windows, kdy asynchronní zápisy nebyly nikdy na explicitní žádost serveru uloženy na disk (flush), mimo případ kdy byl Firebird řízeně ukončen. (Asynchronní zápis není podporován systémy Windows 9X a ME). U systémů s 24/7 provozem tak nikdy nedocházelo ke kontrolovanému uložení dat na disk, a v případě pádu systému tak docházelo k poškození databáze a/nebo ztrátě dat.
Tento parametr určuje jak často jsou zadržované stránky skutečně uloženy na disk v případě vypnutého parametru Forced Writes (zapnut asynchronní zápis). Hodnota je celé číslo, které definuje počet stránek které mohou být neuloženy, než bude nastaven příznak požadavku na fyzický zápis při následujícím potvrzení transakce. Implicitní hodnota na Windows je 100, a na všech ostatních platformách –1 (vypnuto).
Pokud je dosaženo konce cyklu MaxUnflushedWriteTime (viz níže) před dosažením limitu daného tímto parametrem, je příznak požadavku na zápis nastaven okamžitě, a čítač zadržovaných stránek je resetován na nulu.

MaxUnflushedWriteTime

Tento parametr určuje maximální dobu, po kterou mohou být stránky zadržovány operačním systémem při asynchronním zápisu (vypnuto Forced Writes), před jejich fyzickým zápisem na disk. Hodnota parametru je celé číslo představující počet vteřin mezi posledním fyzickým zápisem a nastavením příznaku požadavku na fyzický zápis při následujícím potvrzení transakce. Implicitní hodnota na Windows je 5 vteřin, a –1 (vypnuto) na ostatních platformách.

PrioritySwitchDelay

Nastavení plánovače vláken na Windows. Celé číslo definující časový interval v milisekundách, po jehož uplynutí je priorita neaktivního vlákna snížena (na LOW), nebo priorita aktivního vlákna zvýšena (na HIGH). Jedna iterace této přepínací sekvence reprezentuje jeden cyklus plánovače vláken.
Implicitní hodnota je 100 ms, vybraná na základě experimentů na strojích s procesory Intel PIII/P4. Pro stroje s procesory s nižší frekvencí je nutné nastavit větší prodlevu.

PriorityBoost

Celé číslo, nastavuje počet extra cyklů které vlákno obdrží po zvýšení jeho priority na HIGH. Implicitní hodnota je 5.

ProcessPriorityLevel

Původně *server_priority_class* v konfiguračním souboru ibconfig
Prioritní úroveň/třída procesu serveru. Tento parametr nahrazuje parametr server_priority_class starších verzí – viz níže - novou implementací.
Hodnota parametru je celé číslo, kde:

- 0 – normální priorita
- Kladné číslo – vyšší priorita (stejné jako přepínač –B[oostPriority] programu instsvc.exe
- Záporné číslo – nižší priorita

Poznámka: Jakákoliv změna hodnoty tohoto parametru by měla být pečlivě testována, zda skutečně způsobuje lepší odezvu serveru na požadavky.

RemotePipeName

Platné pouze pro připojení přes NetBEUI

Řetězcový parametr, jméno trubky použité jako přenosový kanál v protokolu NetBEUI. Název trubky je ekvivalentní číslu portu u protokolu TCP/IP. Implicitní hodnota – Interbas – je kompatibilní se staršími verzemi Firebirdu a s InterBase.

Parametry pro konfiguraci pracovního prostoru pro třídění

Pokud je velikost interní třídící vyrovnávací paměti příliš malá pro zpracování všech řádků určených pro třídění, musí Firebird vytvořit dočasné třídící soubory na souborovém systému serverového uzlu.
Standardně server hledá specifikaci cesty definovanou proměnnou prostředí **FIREBIRD_TMP** . Pokud tato proměnná není definována, pokusí se server použít adresář **/tmp** na systémech Linux/UNIX, nebo **C:\temp** na Windows NT/2000/XP. Množství použitého místa z žádné z výše uvedených lokací nemůže být konfigurováno.
Firebird poskytuje parametr pro konfiguraci diskového prostoru který bude použit pro uložení těchto dočasných souborů, přičemž server dbá při použití specifikovaných lokací na to, aby bylo pro třídění za všech okolností dostatek prostoru.
Všechna připojení k databázi sdílejí stejný seznam adresářů pro dočasné soubory, a všechna připojení vytvářejí své vlastní dočasné soubory. Třídící soubory jsou zrušeny po ukončení třídění nebo uvolnění požadavku.
U verze 1.5 je změněn nejen název parametru z **tmp_directory** na **TempDirectories,** ale rovněž jeho syntaxe.

TempDirectories

Nahrazuje *tmp_directory* záznamy v konfiguračním souboru isc_config/ibconfig

Zadejte seznam jednoho nebo více adresářů oddělených znakem středník, ve kterých mají být vytvářeny dočasné soubory pro třídění. Každá položka může obsahovat volitelný argument definující velikost v bajtech, pro omezení množství dat umístěných v daném adresáři. Pokud je argument vynechán nebo je neplatný, bude Firebird využívat prostor v adresáři dle potřeby až do jeho zaplnění. Pokud je specifikovaný nebo dostupný prostor v adresáři plně využit, použije Firebird následující adresář.
Například,
Unix: `/db/sortfiles1 100000000;/firebird/sortfiles2`
Windows: `E:\sortfiles 500000000`
Relativní cesty jsou chápány jako relativní k adresáři který server rozpoznal jako kořenový adresář instalace Firebirdu. Například na Windows, pokud je kořenový adresář C:\Program Files\Firebird, pak následující hodnota specifikuje uložení dočasných třídících souborů v adresáři C:\Program Files\Firebird\userdata\sortfiles, až do limitu 500 MB:

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

POZNÁMKA: Cesty nejsou uzavřeny v uvozovkách, jak bylo vyžadováno u Firebirdu 1.0

Parametry pro kompatibilitu

CompleteBooleanEvaluation

Definuje způsob vyhodnocování logických výrazů (kompletní nebo zkrácenou). Implicitní hodnota (0=False) specifikuje zkrácené vyhodnocování výrazů s AND a OR, vrácením hodnoty v okamžiku kdy zjištěná hodnota již nemůže být ovlivněna vyhodnocením zbytku výrazu.
Ve zvláštních případech (kterým se lze obvykle vyhnout) se může stát, že výsledek operace uvnitř ANO nebo OR která zůstane nevyhodnocena může ovlivnit celkový výsledek (např. vedlejší efekty volání UDF apod.). Pokud máte tu smůlu, že jste zdědili aplikaci která má takové vlastnosti v logice SQL, můžete tento parametr použít pro nastavení kompletního vyhodnocování výrazů (jako u předchozích verzí Firebirdu a InterBase), než budete mít možnost chování aplikace změnit. Parametr má logickou hodnotu.

Nepřehlédněte, že parametr má vliv na vyhodnocování všech logických výrazů prováděných nad jakoukoliv databází na serveru.

OldParameterOrdering

Verze 1.5 opravuje velmi starou chybu InterBase, která způsobuje že výstupní parametry vrácené klientovi v XSQLDA struktuře mají idiosynkratické uspořádání. Tato chyba byla v produktu tak dlouho, že má řada existujících aplikací, ovladačů a komunikačních knihoven zabudované speciální ošetření pro nápravu tohoto problému na klientovi.
Verze 1.5 a novější zohledňují opravu v API a jsou instalovány s hodnotou parametru OldParameterOrdering=0 (False). Nastavte logickou hodnotu tohoto parametru na True, pokud se potřebujete vrátit k původnímu chybnému chování kvůli kompatibilitě s existujícím kódem.

Alias databází

Firebird verze 1.5 nabízí nově možnost zástupných jmen (alias) databází pro zvýšení portability aplikací a pro zvýšení bezpečnostní kontroly nad interním i externím přístupem k databázovým souborům.

Soubor aliases.conf

Pro konfiguraci zástupných jmen databází slouží textový soubor aliases.conf, umístěný v kořenovém adresáři instalace Firebirdu. Instalovaný soubor aliases.conf vypadá podobně jako následující:

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

Jako v ostatních konfiguračních souborech serveru Firebirdu reprezentují znaky '#' začátek řádkového komentáře. Pro konfiguraci alias jednoduše odstraňte znak '#' a změňte dummy řádek na odpovídající cestu k databázi:

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

Změny v souboru aliases.conf můžete provádět za běhu serveru, a pro zpřístupnění nových alias není zapotřebí server restartovat.

Připojování k databázi prostřednictvím alias

Upravený řetězec pro připojení z klientské aplikace k databázi s pomocí alias bude vypadat následovně:

```
Server_name:aliasname
```

Podle výše uvedeného příkladu požádá následující řetězec Firebird běžící na Linuxovém serveru nazvaném "myserver" o vyhledání a připojení klienta k databázi specifikované v souboru aliases.conf jako "fbdb2":

```
myserver:fbdb2
```

Poznámka: protože program **gstat** pracuje přímo s databázovými soubory, je stále nutné specifikovat plnou cestu k databázi. (Může se změnit).

Pojmenování databází na Windows

Na platformě Windows XP a ME je doporučeno používat u databází extenzi ".fdb" k zamezení případného konfliktu a vlastností "System Restore" těchto verzí Windows.

Vývojový tým Firebirdu

Vývojář	Země	Hlavní činnosti
Dmitry Yemanov	Ruská Federace	Release koordinátor; vylepšení DSQL a PSQL; implementace Embedded Serveru, mnoho vylepšení metadat, alias databází, spouště pro více akcí, datový typ BigInt, nové kontextové proměnné, Windows Classic server; oprava bezpočtu chyb
Nickolay Samofatov	Ruská Federace	Design/implementace vlastností SQL (Savepointy, pesimistické zámky); vylepšení metadat; zásadní reimplementace jádra serveru; hledání a oprava chyb; řešení problémů architektury; zpřístupnění Services API pro Linux Classic; zlepšení výkonu; Linux Classic builds
Arno Brinkman	Nizozemí	Vylepšení optimalizátoru; mnoho nových vlastností DSQL
Claudio Valderrama	Chile	Kontrola kódu; hledání a oprava chyb; vylepšení PSQL; oprava, návrh a implementace UDF
Alex Peshkoff	Ruská Federace	Nové vlastnosti PSQL a DSQL; tvorba a koordinace nových bezpečnostních vlastností; oprava kódu; Linux Superserver builds
Mike Nordell	Švédsko	Převod zdrojového kódu Firebirdu do C++; zlepšení výkonu; portování vlastností; hledání a oprava chyb
Blas Rodriguez Somoza	Španělsko	Tvorba nových znakových sad; zásadní čištění a restrukturalizace kódu; MinGW builds
Roman Rokytskyy	Německo	Implementace a koordinace vývoje ovladače Jaybird
David Jencks	U.S.A.	Návrh a koordinace vývoje ovladače JayBird designer; návrh nástrojů pro tvorbu dokumentace
Carlos Guzman Alvarez	Španělsko	Vývoj a koordinace vývoje o.NET ovladače pro Firebird
John Bellardo	U.S.A.	Implementace plug-in rozhranní pro znakové sady; koordinátor, Darwin builds; původní implementace nového paměťového modelu
Erik Kunze	Německo	Hledání a oprava chyb; čištění kódu; SINIX-Z builds
Dmitry Sibiryakov	Ruská Federace	Čištění kódu; MinGW builds
Pavel Cisar	Česká Republika	Linux builds (Release 1.0); koordinace QA a návrh nástrojů pro QA
Ann Harrison	U.S.A.	Oprava chyb; technický poradce; zvýšení maximálního počtu indexů
Mark O'Donohue	Austrálie	Editační řádka v isql; oprava chyb; boot builds (Release 1.0); oprava chyb (Release 1.0)
Paul Reeves	Francie	QA; instalátory pro Win32; standardní Win32 control panel applet
Ignacio J. Ortega	Španělsko	Přidání použití PLAN uvnitř spouští; čištění kódu

Developer	Country	Major tasks
Konstantin Kuznetsov	Ruská Federace	Solaris Intel builds
Olivier Mascia	Belgie	Reimplementace Win32 služby pro instalaci
Peter Jacobi	Německo	Vylepšení a aktualizace znakových sad
Tilo Muetze	Německo	Koordinace projektu pro tvorbu dokumentace
Paul Vinkenoog	Nizozemí	Koordinace projektu pro tvorbu dokumentace; opravy UDF
Artur Anjos	Portugalsko	Vylepšený Win32 control panel applet; Vývoj a lokalizace nástroje Firebird Configuration Manager
Achim Kalwa	Německo	Vylepšený Win32 control panel applet
Sean Leyne	Kanada	Organizace databáze chyb; čištění kódu
Ryan Baldwin	U.K.	Vývoj ovladače JayBird
Sandor Szollosi	Maďarsko	Implementace znakových sad a definic třídění
Dmitry Kuzmenko	Ruská Federace	Oprava chyb v programu GSTAT
Artem Petkevych	Ukrajina	Oprava chyb v typu ARRAY
Vlad Horsun	Ukrajina	Zrychlení sweep procesu; oprava chyby v dvoufázovém potvrzovacím protokolu
Tomas Skoda	Sloenská Republika	Oprava chyb
Evgeny Kilin	Ruská Federace	Oprava chyb
Oleg Loa	Ruská Federace	Oprava chyb
Erik S. La Bianca	U.S.A.	Oprava chyb
Tony Caduto	U.S.A.	Neoficiální Win32 instalátory
Juan Guerrero	španělsko	Nové UDF funkce
Chris Knight	Austrálie	FreeBSD builds
Neil McCalden	U.K.	Solaris builds
Grzegorz Prokopsi	Maďarsko	Debian builds
Paul Beach	U.K.	HP-UX builds
Geoffrey Speicher	U.S.A.	FreeBSD builds
Helen Borrie	Austrálie	Release notes; testování a Myšlenková Policie

"THE FIELD TEST HEROES"

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

POZNÁMKY K INSTALACI

Instalace Firebirdu 1.5 na Windows

ČTĚTE JAKO PRVNÍ!

S uvedením dvou nových variant serveru pro Win32 se rozšířilo i množství možností pro instalaci Firebirdu.

- Ujistěte se, že jste přihlášeni jako Administrator (neplatí pro Win9x nebo ME)

- Všechny moduly serveru—SuperServer, Classic a Embedded stejně jako serverové a klientské nástroje—mohou být instalovány pomocí instalační aplikace pro Windows. Pro úplnou instalaci je doporučeno použít právě instalátor, pokud je k dispozici.

- Použijte **gbak** pro zálohování vaší stávající bezpečnostní databáze **isc4.gdb**. Můžete ji později obnovit jako **security.fdb**

- Pokud máte zvláštní nastavení v souboru **ibconfig,** budete pravděpodobně chtít přenést některé hodnoty do odpovídajících parametrů v souboru **firebird.conf**. Prostudujte si příslušnou část tohoto dokumenty věnovanou souboru firebird.conf k zjištění které hodnoty lze přímo přenést, a které parametry vyžadují novou syntaxi.

- Pokud v instalačním adresáři již existují nějaké konfigurační soubory, budou při použití instalační aplikace zachovány, ale budou PŘEPSÁNY, pokud k instalaci použijete rozbalení distribučního archivu. Jde především o soubory
  security.fdb
  firebird.log
  firebird.conf
  aliases.conf

- Každá součást může být instalována z komprimovaného .zip souboru. Tato metoda instalace bude rychlejší než s použitím instalátoru, pokud máte dost zkušeností s instalací Firebirdu 1.5 z distribučních .zip archivů. Bude ovšem velmi rozčilující, pokud s Firebirdem teprve začínáte.

- Předpokládá se, že
  1 Rozumíte jak funguje vaše síť.
  2 Chápete proč client/server systém vyžaduje server a klienty.
  3 Přečetli jste zbytek tohoto dokumentu—nebo alespoň chápete že je třeba si jej přečíst v případě že něco nepůjde jak jste očekávali.
  4 Víte jak se přihlásit do mailové konference firebird-support v případě, že budete potřebovat radu. Přihlásit se můžete na adrese http://www.yahoogroups.com/groups/firebird-support

Pokud již máte instalovánu některou ze starších verzí firebirdu nebo InterBase®, a předpokládádte že se budete potřebovat vrátit k těmto instalacím, připravte si cestu k obnově původní instalace dříve než začnete s instalací Firebirdu 1.5.

- Použijte existující verzi nástroje GBAK pro vytvoření záložní kopie vašich databází v přenositelném formátu.

- Ve vašem systémovém adresáři vytvořte záložní kopii souboru gds32.dll. Můžete záložní kopii nazvat "gds32.dll.ib5" nebo "gds32.dll.fb103" nebo obdobným informativním způsobem; nebo zkopírovat ji do jiného adresáře.

- Šikovnou se může rovněž stát i záloha Microsoft C++ runtime knihovny, msvcp60.dll. Instalátor by neměl přepsat vaši verzi knihovny, ale jak známo, divné věci se občas stávají.

- **ZASTAVTE JAKÝKOLIV BĚŽÍCÍ SERVER FIREBIRD NEBO INTERBASE**
  Instalátor se pokusí zjistit, zda je instalována a/nebo běží starší verze Firebirdu nebo InterBase. Při instalaci bez instalátoru je to na vás!

- Implicitní umístění kořene instalace pro Firebird 1.5 je adresář C:\Program Files\Firebird\Firebird_1_5. Pokud je v tomto adresáři již umístěna vaše původní instalace, a přejete si instalovat verzi 1.5 podle implicitního nastavení, přejmenujte tento existující adresář.

- Při instalaci Firebirdu jako služby: Pokud si přejete využít nové možnosti **bezpečného přihlášení**, vytvořte „uživatele služby Firebird" na vašem systému—libovolné jméno a heslo dle vašeho výběru—jako běžného uživatele s odpovídajícími privilegii. Měli by jste si nejdříve přečíst dokument nazvaný README.instsvc.txt. Pokud máte distribuční zip archiv, naleznete jej v adresáři /doc uvnitř .zip archivu. Pokud distribuční .zip archiv nemáte, bude soubor dostupný až po instalaci. Stejný dokument si ale můžete přečíst i na adrese:
  http://cvs.sourceforge.net/viewcvs.py/firebird/firebird2/doc/README.instsvc

DALŠÍ DŮLEŽITÉ ČTENÍ!

Jedním z cílů pro Firebird 1.5 je vytvoření cesty pro vícenásobné instalace serveru, a umožnit tak uživatelům provozovat vedle sebe různé verze. Firebird 1.5 již tuto možnost nabízí, ačkoliv není dobře zdokumentována, a vyžaduje zásah zkušeného uživatele. U budoucích verzí Firebirdu bude vícenásobná instalace mnohem méně komplikovaná. Verze 1.5 je určitý přípravný mezistupeň, především ve způsobu instalace různých verzí knihoven. Zároveň Microsoft podnikl své vlastní kroky pro řešení problému instalace různých verzí knihoven. Dohromady tyto dva nezávislé problémy znamenají zcela nový přístup k instalaci knihoven počínaje Firebirdem 1.5.

Instalace systémových knihoven fy Microsoft

Problém spojený s instalací odlišných verzí systémových knihoven Microsoftu je natolik známý, že si vydobyl označení „DLL Hell".

Počínaje Windows 2000, učinil Microsoft téměř nemožným aktualizovat systémové knihovny. Jako řešení doporučuje Microsoft, aby si každá aplikace instalovala vlastní privátní kopie všech potřebných systémových knihoven.

Firebird 1.5 postupuje podle tohoto doporučení, a instaluje potřebné knihovny do podadresáře \bin společně s serverem.

Instalace knihovny fbclient.dll

Počínaje verzí 1.5 Firebird nepoužívá klientskou knihovnu gds32.dll. Tato knihovna je nyní nazývána fbclient.dll. Protože chceme umožnit současnou instalaci vícera serverů, nemá smysl nadále ukládat klientskou knihovnu v systémovém adresáři, a a vytvořit tak vlastní DLL hell. Proto je klientská knihovna počínaje Firebirdem 1.5 umístěna v podadresáři \bin spolu s ostatními binárními soubory.

Byl zaveden nový klíč v Registry, a všechny aplikace využívající Firebird musí tento klíč používat pro nalezení odpovídající verze Firebirdu, který chtějí používat. Tento nový klíč je:

HKEY_LOCAL_MACHINE\SOFTWARE\Firebird Project\Firebird Server\Instances

Firebird zaručuje, že pod tímto klíčem vždy existuje jeden záznam, který bude identifikován jako

"DefaultInstance"

a bude obsahovat cestu ke kořenovému adresáři (ano, můžete hádat) implicitní instalace. Aplikace kterým nezáleží na konkrétní verzi/instalaci mohou vždy používat implicitní instanci pro vyhledání knihovny fbclient.dll.

Budoucí verze Firebirdu budou vytvářet další záznamy pod klíčem Instances. Aplikace tak budou schopné zpracováním těchto klíčů nalézt odpovídající klientskou knihovnu dle svých potřeb.

Podpora zděděných aplikací a ovladačů

Aplikace které používají InterBase nebo Firebird tradičně předpokládají, že se klientská knihovna nazývá gds32.dll, a je zaváděna ze systémového adresáře. Firebird 1.5 je dodáván s nástrojem „instclient.exe" který nainstaluje klon knihovny fbclient.dll do systémového adresáře Windows. Na tento klon je při kopírování aplikována záplata, takže informace o verzi souboru začínají „6.3", pro kompatibilitu se staršími aplikacemi které kontrolují verzi GDS32.DLL, a nejsou schopné správně zpracovat verzi s číslem „1.5".

V průběhu instalace ověřuje instalační program, zda na počítači již není instalována InterBase nebo Firebird. Pokud žádnou instalaci nenalezne, zkopíruje gds32.dll do systémového adresáře. Pokud bude zjištěna možnost, že na počítači již existuje nějaká verze Firebirdu nebo InterBase, gds32.dll do systémového adresáře zkopírována nebude. Nicméně je později možné k tomu použít program „instclient.exe".

Budoucí verze Firebirdu se nebudou pokoušet zapsat gds32.dll do systémového adresáře, a konečným cílem je tuto knihovnu zcela vypustit z distribuce serveru.

Program „instclient.exe" může rovněž v případě potřeby instalovat knihovnu FBCLIENT.DLL do systémového adresáře Windows. Tím jsou ošetřeny nástroje a aplikace které ji vyžadují v tomto adresáři.
Program instclient.exe by měl být umístěn v ‚bin' adresáři instalace Firebirdu, a musí být z tohoto adresáře také spouštěn.

Použití programu instclient.exe:

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

Kde *knihovna* je: `fbclient | gds32`
S libovolným přepínačem lze použít přepínač '-z', který vypíše číslo verze.

Informace o verzi a čítání sdíleného použití knihovny je zpracováváno automaticky. Pro ignorování kontroly verze lze použít parametr –f[orce].

POZNÁMKA: Pokud si vynutíte ignorování verze parametrem –f, můžete poškodit již existující instalaci Firebirdu nebo InterBase®. Pro dokončení kopírování může být nezbytné restartovat počítač.

Více informací naleznete v dokumentu *README.Win32LibraryInstallation.txt* který je umístěn v kořenovém adresáři instalace nebo podadresáři \doc.

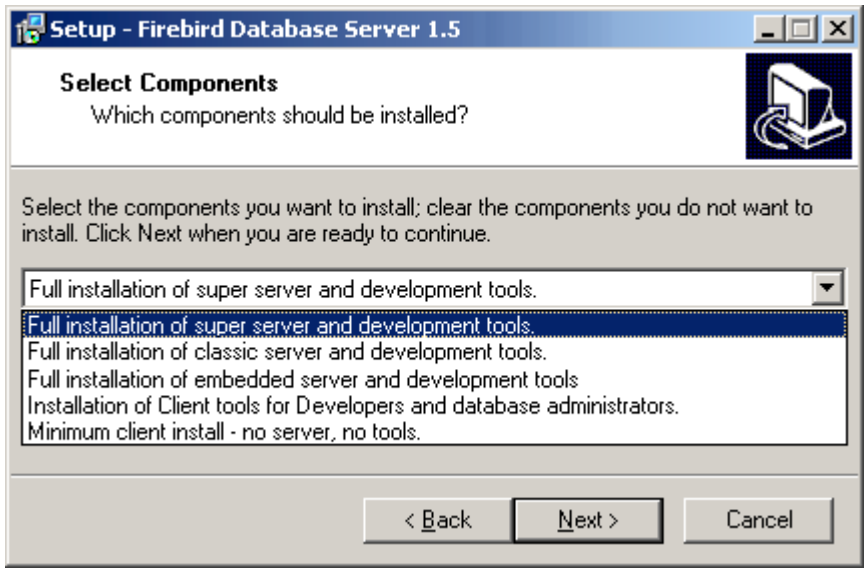
Čistka instalace starších release candidate

Upozorňujeme, že instalační program odstraní knihovnu fbclient.dll ze systémového adresáře, pokud ji zde nalezne. Instalátor rovněž odstraní již nepoužívaný klíč registry HKLM\Software\FirebirdSQL.

Použití Win32 instalátoru Firebird

Použití je velmi snadné.

Jednoduše spusťte instalační program a odpovězte na dialogy. Po projití zhruba čtyř dialogů, uvidíte dialog s rozbalovacím seznamem, který—když je rozbalen—vypadá podobně jako na následujícím obrázku. Toto je opravdu poslední možnost, kdy máte možnost zvolit typ instalace který požadujete.



Vyberte požadovaný typ instalace a klepněte na "Next", který vás přenese k následujícím dialogům.

Služba nebo aplikace?

Pokud si vyberete k instalaci SuperServer nebo Classic, a váš operační systém podporuje služby, budete dotázání, zda si přejete provozovat Firebird jako službu nebo jako aplikaci. Pokud nemáte dobrý *důvod*, proč server provozovat jako aplikaci, vyberte službu.

Manuální nebo automatický?

V automatickém režimu bude Firebird spuštěn při každém startu počítače. V manuálním režimu musíte server dle potřeby spustit sami.

Guardian

Guardian je nástroj který pracuje „nad" Superserverem, a automaticky ho restartuje v případě jeho abnormálního ukončení. Tuto vlastnost může využít především v průběhu vývoje. V ostrém provozu umožňuje předejít situacím, kdy server přestane pracovat, a nikdo nemůže najít administrátora, aby jej restartoval.

Instalační (kořenový) adresář

Pokud se rozhodnete nepoužít implicitní adresář, nalistujte na adresář který jste již vytvořili, nebo vypište plnou cestu. Specifikovaný adresář nemusí existovat. V takovém případě vás o tom instalátor uvědomí, a požadovaný adresář vytvoří.

Instalace SuperServeru z distribučního .zip archivu

Instalace Firebirdu 1.5 je v zásadě obdobná instalaci předchozích verzí.
Pokud nemáte speciální instalační program (distribuovaný samostatně), je postup následující:

- Rozbalte archív do samostatného adresáře (protože byly některé soubory přejmenovány, nemá smysl archiv rozbalovat do adresáře s IB/FB1)
- Změňte aktuální adresář na <kořen>\bin (v adresáři <root> a podadresářích jsou soubory v1.5)
- Spusťte program instreg.exe:
-

  instreg.exe install

  Tím bude nadřízeného adresář uložen do registry jako instalační cesta.
  (HKLM\Software\Firebird Project\Firebird Server\Instances\DefaultInstance)
- Pokud si přejete provozovat server jako službu, spusťte rovněž program instsvc.exe:
  instsvc.exe install
- Měli by jste překopírovat soubory fbclient.dll a gds32.dll do systémového adresáře Windows, ale není to nezbytné, pokud zajistíte aby vaše aplikace tytou soubory nalezla jiným způsobem.

Instalace Classic serveru z distribučního .zip archivu

Postup při instalaci Classic serveru je shodný s instalací SuperServeru s jediným rozdílem, dodatečným parametrem pro program instsvc.exe:

```
instsvc.exe install -classic
```

Povšimněte si, že jako službu můžete provozovat pouze jedinou variantu serveru, SuperServer (fbserver.exe) nebo Classic (fb_inet_server.exe, základní proces pro Classic).

Instalace Classic serveru **úmyslně** neobsahuje **Control Panel Applet**. Nepokoušejte se ho instalovat a používat. Koncept ukončení služby není pro architekturu Classic použitelný.

Zjednodušená instalace

Pokud nepotřebujete provozovat server jako službu, nemusíte spouštět programy instreg.exe a instsvc.exe. V takovém případě můžete jednoduše rozbalit archiv do samostatného adresáře, a spustit server:

```
fbserver.exe -a
```

Server by měl v takovém případě chápat nadřízený adresář jako kořenový.

Odinstalace

K odstranění FB 1.5 bez použití odinstalačního programu pro Windows by jste měli:

- Zastavit server
- Spustit "instreg.exe remove"
- Spustit "instsvc.exe remove"
- Smazat instalační adresář
- Smazat soubory fbclient.dll a gds32.dll ze systémového adresáře Windows

Instalace Embedded serveru z distribučního .zip archivu

Embedded server je klient s plně funkčním serverem v jediné dynamicky připojované knihovně (fbembed.dll). Má naprosto stejné vlastnosti jako server architektury SuperServer, a exportuje standardní vstupní body Firebird API.

Registry Záznamy v Registry pro Firebird (kde server normálně hledá umístění kořenového adresáře instalace) jsou ignorovány. Kořenový adresář pro embedded server je adresář andřízený adresáři, v němž je uložena knihovna serveru.

Přístup k databázi Je dovolen pouze "skutečný lokální" přístup. Embedded server nemá žádnou podporu pro síťové protokoly, tedy nebude fungovat ani přístup přes "localhost".

Autentizace a bezpečnost Embedded server nepoužívá bezpečnostní databázi (security.fdb), a tato databáze tedy není zapotřebí. Jakýkoliv uživatel se může připojit k libovolné databázi. Protože jak server, tak klient pracují ve stejném (lokálním) adresním prostoru, je bezpečnost věcí fyzického přístupu.

SQL privilegia jsou kontrolována stejně jako v jiných variantách serveru.

Kompatibilita Bez jakýchkoliv konfliktů lze provozovat libovolný počet aplikací s embedded serverem. Rovněž není problémem běžící IB/Fb server.

Nicméně je nutné mít na paměti, že nemůžete současně přistupovat k stejné databázi z více embedded serverů, protože mají architekturu SuperServer, a tedy exkluzivně uzamykají připojené databáze.

Struktura souborů pro Embedded Server
Jednoduše nakopírujte soubor fbembed.dll do adresáře k vaší aplikaci. Následně ji přejmenujte na fbclient.dll nebo gds32.dll, v závislosti na tom, jaké jméno klientské knihovny očekává váš systém/knihovna pro připojení k databázi. Pokud si přejete používat i standardně dodávané nástroje (isql, gbak, atd.), musíte knihovnu přejmenovat na fbclient.dll (není problém vytvořit dvě kopie knihovny s odlišným jménem).

Rovněž by jste měli do stejného adresáře nakopírovat soubory firebird.msg, firebird.conf (pokud je potřebný) a ib_util.dll.

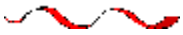
Pokud vaše aplikace vyžaduje přítomnost uživatelských modulů jako jsou např. UDF nebo podpora znakových sad (fbintl.dll), měly by být umístěny odděleně od samotné aplikace. Umístěte tyto moduly do adresářů, které emulují adresářovou strukturu Firebirdu, tzn. podadresáře pojmenované /intl a /udf přímo pod adresářem ve kterém jsou umístěny základní soubory Firebirdu.

Příklad

```
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_utils.dll
D:\my_app\gbak.exe
D:\my_app\firebird.msg
D:\my_app\intl\fbintl.dll
D:\my_app\udf\fbudf.dll
```

Nakonec restartujte svou aplikaci. Nyní bude používat embedded server jako klientskou knihovnu a bude schopná pracovat s lokálními databázemi.

POZNÁMKA: Pokud jste vytvořili adresářovou strukturu přesně podle uvedených pravidel, není nutné explicitně konfigurovat RootDirectory v souboru firebird.conf. Nicméně pokud se rozhodnete instalovat embedded server a vaši aplikaci do odlišné adresářové struktury, přečtěte si dokument README_embedded.txt z distribuce Embedded Serveru obsahující instrukce pro další konfigurace.



Odinstalace

Odinstalační rutina Firebirdu zachovává a přejmenovává následující klíčové soubory:

- zachová security.gdb nebo ho přejmenuje na security.fbnnnn
- zachová firebird.log
- zachová firebird.conf nebo ho přejmenuje na firebird.confnnnn
- zachová aliases.conf nebo ho přejmenuje na aliases.confnnnn

"nnnn" je číslo sestavy (build number) předchozí instalace.
Soubory které nejsou součástí originální instalace nejsou nijak dotčeny.
Sdílené soubory jako je fbclient.dll a gds32.dll budou smazány, pokud čítač sdílení indikuje, že soubor již není využíván žádnou aplikací.
Klíče Registry které byly vytvořeny budou odstraněny.

Ostatní poznámky

Winsock2

Firebird vyžaduje WinSock2. Všechny Win32 platformy by tuto podmínku měly splňovat, mimo Windows 95. V průběhu instalace je prováděn test na přítomnost knihovny Winsock2. Pokud není nalezena, nebude instalace dokončena. Informace jak aktualizovat váš systém naleznete např. na adrese: http://support.microsoft.com/default.aspx?scid=kb;EN-US;q177719

Windows ME a XP

Na systémech Windows ME a XP (Home a Professional) je k dispozici vlastnost nazývaná System Restore, která provádí automatické zálohování (a případnou obnovu) změněných systémových souborů, bohužel včetně souborů s příponou ".gdb". Výsledným efektem je výrazné zpomalení práce s databázemi. (System Restore není součástí .NET Serverů).

Na Windows ME je seznam „chráněných" souborů uložen v souboru c:\windows\system\filelist.xml. Charlie Caro původně doporučoval odstranit příponu GDB ze sekce "includes" tohoto souboru. Od té doby bylo ovšem prokázáno, že WinME mohou tento seznam obnovit. U Windows XP není možné soubor filelist.xml vůbec změnit.

Doporučená permanentní řešení pro Windows ME jsou:

- Použít extenzi FDB (Firebird DB) pro vaše primární databázové soubory
- Přesunout databáze do adresáře C:\My Documents, který je systémem System Restore ignorován
- Zcela vypnout System Restore (návod naleznete v dokumentaci Windows).

Na Windows XP Home a Professional můžete přesunout databáze na samostatný diskový oddíl, a vypnout System Restore pro tento oddíl.

Windows XP používají „chytré kopírování", takže zpomalení práce s databázemi není tak výrazné jako u Windows ME, přinejmenším pro menší databáze.

Ukončování (shutdown) Windows XP je známo jako "temné území". Dochází k poměrně dlouhé prodlevě kdy je ukončována služba serveru. V průběhu zastavování je indikováno, že Firebird běží jako aplikace. Problém podle všeho postihuje pouze Windows XP a pouze v případě, že k zastavení služby není používán Guardian. Použití Guardian aplikace je jediným dočasným řešením, než bude nalezena a odstraněna podstata problému.

Instalace na systémech UNIX / Linux

(Původní text Mark O'Donohue, revize pro 1.5)

Firebird server je distribuován ve dvou formách, Classic který běží jako služba, a SuperServer který běží jako démon na pozadí.

POZNÁMKY – ČTĚTE JAKO PRVNÍ

1) Pro instalaci Firebirdu musíte být přihlášení jako uživatel root.
2) Instalace na Linuxu vyžaduje knihovnu glibc verze 2.2.5 nebo novější a knihovnu libstdc++ verze 5.0 nebo novější.
3) Hrubý přehled kompatibilních distribucí Linuxu naleznete zde.
4) Ujistěte se, že máte instalovány editory 'ed' a 'vim'. Pokud editory nainstalovány nejsou, bude sice Firebird instalován, ale instalační skripty selžou. (POZNÁMKA: tato závislost může být v budoucnu nahrazena editorem 'sed').

INSTALACE NA LINUXU

Následující instrukce popisují instalaci verze Classic. Pro instalaci verze SuperServer nahraďte „CS" v názvech balíků za „SS". Například balík FirebirdCS-1.5.0-nnnn.i686.rpm je nahrazen názvem FirebirdSS-1.5.0-nnnn.i686.rpm.

Linux rpm install

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

Linux .tar.gz install

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

Co dělá Linuxová instalace

Linuxová instalace:

1. Pokusí se zastavit jakýkoliv právě běžící server Firebird nebo InterBase
2. Vytvoří uživatele 'firebird' a skupinu 'firebird' , pokud neexistují.
3. Firebird je instalován do adresáře /opt/firebird, vytvoří symbolický odkaz na knihovny do /usr/lib a hlavičkové soubory do /usr/include
4. Automaticky je přidána služba gds_db na portu3050 do /etc/services pokud ještě neexistuje.
5. Automaticky je přidán localhost.localdomain a HOSTNAME do /etc/host.equiv
6. SuperServer rovněž instaluje startovací skript /etc/rc.d/init.d/firebird.
7. Pro Classic server je vytvořena xinetd konfigurace /etc/xinet.d/firebird, nebo pro starší systémy konfigurace inetd v souboru /etc/inetd.
8. Na SuSE je pro SuperServer vytvořen záznam v /etc/rc.config (specifická konfigurace SuSE pro service startup management) a symbolický odkaz /usr/bin/rcfirebird na startovací skript.
9. Spustí službu serveru. Firebird by měl být spouštěn automaticky v runlevelu 2, 3 nebo 5.
10. Vygeneruje nové náhodné heslo pro SYSDBA a uloží ho do souboru /opt/firebird/SYSDBA.password.
11. Přidá záznam do souboru aliases.conf pro ukázkovou databázi employee.fdb.

Classic install automaticky konfiguruje záznam pro xinetd pokud je nalezen adresář /etc/xinetd.d, jinak je vytvořen záznam pro inetd. Protože některé distribuce ppoužívají odlišný způsob konfigurace xinetd, může být nutné provést manuální nastavení.

Testování vaší Linuxové instalace

Krok 1 – Přístup k databázi

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

>connect localhost:employee.gdb; /* Použití aliasu */

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

>quit;
```

*Heslo bylo vygenerováno v průběhu instalace, a je dostupné v souboru /opt/firebird/SYSDBA.password.

Krok 2 – Vytvoření databáze

Počínaje verzí 1.5 běží server implicitně pod uživatelským účtem 'firebird'. Ačkoliv se jedná o preferovaný způsob, byl dříve server provozován pod účtem uživatele 'root'. Při běhu pod tímto účtem měl server možnost číst, vytvářet a rušit databázové soubory kdekoliv v rámci POSIXového systému souborů. Z bezpečnostních důvodů by měly být možnosti serveru pro čtení, rušení a vytváření souborů omezeny.

Ačkoliv je nová konfigurace lepší z pohledu bezpečnosti, je při vytváření nových databází nutné vzít v úvahu, že:

a) Uživatel 'firebird' musí mít oprávnění k zápisu pro adresář, v kterém chcete databázi vytvořit.
b) Doporučená hodnota pro parametr DatabaseAccess v souboru /opt/firebird/firebird.conf je None, aby byl přístup omezen pouze na soubory definované v souboru aliases.conf.
c) Použijte záznamy v aliases.conf pro odstínění uživatelů od fyzického umístění databáze. Více informací o aliasech databází naleznete zde.

Postup při vytvoření nové databáze se může odlišovat pokud použijete odlišnou konfiguraci, ale následující kroky jsou doporučené s výše uvedenou konfigurací:

1. Pokud adresář ve kterém má být databáze vytvořena doposud neexistuje, přihlaste se jako uživatel root a adresář vytvořte:

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

2. Vytvořte novou databázi a definujte záznam v souboru aliases.conf, který na ni odkazuje. Jako uživatel root nebo firebird proveďte následující skript:

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

(Použití je: createDBAlias.sh <název-databáze> <cesta-k-databázi>)

3. Alternativně (pro krok 2) lze pro vytvoření aliasu místo skriptu použít manuální úpravu pomocí:

```
$vi /opt/firebird/aliases.conf
a přidat řádek na konec souboru:
test.fdb /var/firebird/test.fdb
```

4. Vytvořte databázi:

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

5. Pokud je hodnota parametru DatabaseAccess v /opt/firebird/firebird.conf nastavena na Full nebo omezena na konkrétní adresář(e) (např.: DatabaseAccess=/var/firebird) lze alternativně k kroku 2 databázi vytvořit přímo specifikací fyzické cesty souboru:

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

Pokud použijete takovou konfiguraci, lze k databázi přistupovat přímo bez záznamu v alias souboru:

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

*Heslo bylo vygenerováno v průběhu instalace, a je dostupné v souboru /opt/firebird/SYSDBA.password.

Tipy a pomocné skripty

V adresáři bin instalace jsou vedle standardních souborů i následující skripty:

- **ChangeDBAPassword.sh** – Změní heslo uživatele SYSDBA. Pro SuperServer tento skript rovněž změní heslo v inicializačním skriptu /etc/rc.d/init.d/firebird.
- **CreateAliasDB.sh** - Použití: createDBAlias.sh <název-databáze> <cesta-k-databázi>
  Skript vytvoří novou fyzickou databázi a přidá pro ni záznam v souboru aliases.conf.
- **Fb_config** – Skript který lze použít v make souborech k generování požadovaných cest pro include a direktivy vložení knihoven aktuálně instalované verze Firebirdu. Parametr –help skriptu vypíše kompletní seznam parametrů.
- **ChangeGdsLibraryCompatibleLink.sh** – Pouze Classic – Změní odkaz na klientskou knihovnu libgds.so mezi vícevláknovou knihovnou libfbclient.so (standardní SS klient) a jednovláknovou knihovnou libfbembed.so (standardní CS klient – embedded server). Pro kompatibilitu s předchozími instalacemi odkazuje libgds.so implicitně na libfbembed.so.
- **Zapouzdřený nebo přímý přístup k databázovým souborům**
  Classic server nabízí zapouzdřený (embedded) přístup k lokálním databázím, který umožňuje klientským aplikacím otevřít databázové soubory přímo. Pro správnou funkci aplikací v tomto režimu musí mít uživatel provozující takovou aplikaci přístupová oprávnění nejen k databázím, ale i k některým konfiguračním a stavovým souborům Firebirdu.
- **Získání přímého pořístupu k databázím:** Protože je nyní server provozován poud uživatelským účtem firebird místo root, je nezbytné aby byli uživatelé aplikací používajících lokální (embedded) přístup zařazeni do uživatelské skupiny firebird. Postup je zdokumentován v readme, ale následující kroky vás dostanou tak kam potřebujete:
  Pro přidání uživatele (např.: skywalker) do skupiny firebird musí uživatel root provést příkaz:

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

Při následujícím přihlášení bude uživatel skywalker oprávněn přistupovat lokálně k databázím vytvořeným serverem pod uživatelem firebird.

K výpisu seznamu skupin do kterých náleží aktuálně přihlášený uživatel použijte příkaz *groups*.

- **Problémy s NTPL na vyšších verzích Linuxu:**
  Nová knihovna NTPL (Native POSIX Thread Library) na Red Hatu 9 (a novější) způsobuje problémy u SuperServeru a lokálně kompilovaných programů, včetně nástrojů. Speciálně program gbak ohlašuje chybu Broken Pipe. Náprava:

  1. Do souboru /etc/init.d/firebird doplňte za úvodní komentáře
  LD_ASSUME_KERNEL=2.2.5
  export LD_ASSUME_KERNEL

  Tím je zajištěn běh serveru Firebird. Nyní je nutné zajistit nastavení této proměnné rovněž v lokálním prostředí, takže:

  2. přidejte následující do souboru /etc/profile, aby jste zajistili že každý uživatel bude mít toto nastavení.

  po
  HISTSIZE=1000
  přidejte
  LD_ASSUME_KERNEL=2.2.5
  na následujícím řádku exportu doplňte:
  export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE INPUT_RC LD_ASSUME_KERNEL

Odinstalace na Linuxu

Pokud potřebujete provést odinstalování, proveďte je jako uživatel root. Následující příklad je platný pro Classic, a lze jej adaptovat pro SuperServer nahrazením CS za SS.

Pro rpm instalaci:

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

Nebo pro .tar.gz instalaci:

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

Instalace Firebird Classic & SuperServer na Solaris 2.7 Sparc

V současnosti není k dispozici. Jako referenci můžete použít releasenotes v.1 jako referenci pro instalaci verze 1.5.

Instalace Firebird Classic na MacOS X / Darwin

V současnosti není k dispozici. Jako referenci můžete použít releasenotes v.1 jako referenci pro instalaci verze 1.5.

Kompilace nebo Instalace Firebird na FreeBSD

V současnosti není k dispozici. Jako referenci můžete použít releasenotes v.1 jako referenci pro instalaci verze 1.5.

Konfigurace služby na portu na klientovi a serveru

Firebird standardně přijímá žádosti klientů na spojení na portu 3050. registrovaný název služby portu je **gds_db**. Dobrá zpráva je, že pokud vám vyhovuje toto standardní nastavení, nemusíte již provádět žádnou další konfiguraci služeb portu u klientů a serveru.

Můžete ovšem použít jiný port, jiné jméno služby nebo obojí. Odlišné nastavení může být nezbytné, pokud je port 3050 vyžadován jinou službou, například souběžně pracující jiná verze Firebirdu nebo InterBase® serveru. Existuje několik způsobů, jak změnit standardní nastavení. Jak server, tak klienti musí být konfigurování pro změnu jména služby nebo čísla portu, nebo obojího, alespoň jedním z následujících způsobů:

- V připojovacím řetězci specifikovaném klientem
- V příkazu použitém pro spuštění serveru
- Aktivací a změnou parametru RemoteServicePort nebo RemoteServiceName v souboru firebird.conf (počínaje verzí 1.5)
- V konfiguraci démona služeb portů (pro Classic na POSIX platformách)
- Záznamem v souboru Services

Před bližším studiem technik technik je vhodné seznámit se s logikou použitou Firebirdem pro určení portu pro komunikaci jak na straně serveru, tak u klienta.

Jak Server určuje port pro příjem připojení

Při spuštění serveru je možné specifikovat volitelný přepínač (-p) kterým lze definovat buď **číslo portu**, nebo **jméno služby** pro příjem příchozích žádostí o připojení. Pokud je tedy přepínač specifikován, je port 3050 nebo jméno služby (gds_db) nahrazeno argumentem specifikovaným s přepínačem –p.

Následně – nebo prvně pokud není specifikován –p přepínač – kontroluje server verze 1.5 nastavení parametrů RemoteServicePort a RemoteServiceName v souboru firebird.noc:

- Pokud jsou oba zakomentovány znakem „#", pak jsou předpokládány implicitní hodnoty a další změny nejsou provedeny. Pro hodnoty nepředefinované argumentem –p tedy zůstavávají implicitní hodnoty.
- Pokud je aktivní parametr RemoteServiceName, ale nikoliv RemoteServicePort, je název služby nahrazen názvem definovaným parametrem *pouze* pokud již nebyl přepsán argumentem –p.
- Pokud je aktivní parametr RemoteServicePort, ale nikoliv RemoteServiceName, je číslo portu nahrazeno číslem definovaným parametrem *pouze* pokud již nebylo přepsán argumentem –p.
- Pokdu jsou aktivní oba parametry RemoteServiceName a RemoteServicePort, pak má parametr RemoteServiceName přednost, pokud nebyl přepsán argumentem –p. V takovém případě je hodnota RemoteServiceName ignorována a je použita hodnota RemoteServicePort.
- V tomto okamžiku, pokud byla indikována změna čísla portu nebo názvu služby, jak server v1.0 tak v1.5 pokračují kontrolou souboru Services, kde hledají záznam s korektní kombinací názvu služby a čísla portu. Pokud je takový záznam nalezen, je vše vpořádku. Pokud záznam nalezen

není, a název služby není gds_db, vyvolá server výjimku a odmítne se spustit. Pokud je název služby gds_db a nelze jej vyhodnotit na jiné číslo portu, bude mapováno na port 3050.

Pokud má být redefinován název služby, je nutné vytvořit příslušný záznam v souboru Services – viz níže, *Configurace souboru Services*.

Použití přepínače –p

Tento parametr je k dispozici rovněž u Firebirdu 1.0.x, ale nebyl dříve dokumentován.
Spuštění serveru s přepínačem –p umožňuje předefinovat implicitní číslo portu (3050), nebo implicitní název služby (gds_db), ale nikoliv obojí najednou. Počínaje Firebirdem v1.5 lze tento přžepínač použít v kombinaci s konfiguračním souborem firebird.conf pro redifinici obou hodnot (viz výše).

Syntaxe pro TCP/IP

```
server-command <ostatní přepínače> -p číslo-portu | jméno-služby
```

například k spuštění SuperServeru jako aplikace a redefinici jména služby z gds_db na fb_db:

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

Nebo k přepsání čísla portu z 3050 na 3051:

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

Syntaxe pro přesměrování WNet

Pro Wnet sítě nahraďte syntaxi –p přepínače za následující syntaxi:

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

nebo

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

Classic na POSIX platformě: *inetd nebo xinetd démon*

U Classic serveru na platformě Linux nebo UNIX je konfigurace jména služby a portu součástí konfigurace démona **inetd** nebo **xinetd**. Instalační skript zapisuje odpovídající záznam do souboru /etc/inetd.conf nebo /etc/xinetd.conf (případně jako samostatný soubor s názvem firebird v adresáři /etc/xinetd.d).

Tuto konfiguraci můžete v případě potřeby změnit. Následující příklad ukazuje záznam který by jste měli nalézt v souboru xinetd.conf po instalaci Classic serveru na Linuxu:

```
# 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
}
```

Pokud chcete změnit číslo portu, je nezbytné přslušně upravit konfiguraci xinetd nebo inetd, a restartovat démona (pomocí kill –HUP nebo pomocí příslušného initd skriptu), aby byla nová konfigurace zohledněna.

POZNÁMKA Žádosti o spojení selžou, pokud xinetd (nebo inetd) a fbserver (nebo ibserver) budou naslouchat na stejném portu. Pokud používáta takovou vícenásobnou konfiguraci, je nezbytné nastavit každý server pro práci na jiném portu.

Použití parametru v konfiguračním souboru

Pomocí parametrů RemoteServiceName nebo RemoteServicePort v souboru firebird.conf lze předefinovat implicitní číslo portu (3050) nebo implicitní název služby (gds_db) které server používá pro příchozí spojení.

Server použije jedne z RemoteService* parametrů, ale nikoliv oba. Pokud jsou definovány oba, bude server ignorovat RemoteServicePort ve všech případech, s výjimkou spuštění serveru s přepínačem –p který redefinuje název služby. Lze tedy použít přepínač –p v kombinaci s RemoteServicePort parametrem k předefinování jak čísla portu, tak názvu služby.

Pokud má být redefinován název služby, je nutné vytvořit záznam s souboru Services.

VÝSTRAHA! Pokud odkomentujete RemoteServiceName nebo RemoteServicePort, ale ponecháte jejich implicitní hodnotu, budou přesto považovány za předefinování. V takovém případě bude nezbytné vytvořit záznam v souboru Services i pro implicitní hodnoty.

Nastavení klienta pro nalezení čísla portu

Pokud váš server používá implicitně definované hodnoty (služba gds_db na portu 3050), pak již není třeba žádná další konfigurace na straně klienta. Pokud server používá odlišné číslo portu nebo odlišný název služby, pak klientská aplikace a/nebo klientská stanice vyžaduje dodatečnou konfiguraci umožňující klientské knihovně Firebirdu nalézt port pro komunikaci.

Spojovací řetězec použitý klientem může v různé podobě obsahovat informaci pro zjištění čísla portu. Klienti Firebirdu 1.5 mohou volitelně použít lokální kopii souboru firebird.conf. Mohou být rovněž vyžadovány změny v lokálním souboru Services.

Použití připojovacího řetězce

Pokud bylo předefinováno pouze jméno služby nebo číslo portu, pak vložte nové jméno služby nebo číslo portu do připojovacího řetězce. Tatotechnika funguje se všemi verzemi Firebirdu.

Syntaxe pro TCP/IP spojení

Pro připojení k serveru jménem alice naslouchajícím jako služba fb_db na portu 3050 bude spojovací řetězec:

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

nebo pokud je jméno služby gds_db na portu 3051:

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

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

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

Povšimněte si, že oddělovač mezi názvem serveru a portem je lomítko, nikoliv dvojtečka. Dvojtečka před fyzickým umístěním databáze je stále vyžadována.

Syntaxe pro Wnet spojení

Na Wnet síti použijtenotaci ve stylu UNC:

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

nebo

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

Pokud je port nebo služba na které server pracuje předefinována, je nezbytné vytvořit záznam v souboru Services.

Použití definice portu s alias databáze

Pro připojení přes jiný než implicitní port k databázi specifikované pomocí alias, přidejte specifikaci portu k specifikaci serveru, nikoliv k aliasu. Například pokud bude databázový alias definován v souboru aliases.conf jako:

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

Bude připojovací řetězec pro tuto databázi na serveru 'alice' vypadat následovně:

```
alice/fb_db:rabbit
```

nebo

```
alice/3051:rabbit
```

Použití kopie souboru firebird.conf

Počínaje Firebirdem 1.5, můžete volitelně na klienstkou stanici umístit kopii souboru firebird.conf do kořenového adresáře Firebirdu, a definovat hodnoty parametrů RemoteServiceName nebo RemoteServicePort pro specifikaci portu použitých klientem pro komunikaci.

- Můžete specifikovat *jeden* z těchto dvouparametrů samostatně, a případně můžete toto nastavení skombinovat s specifikací zbývajícího parametru ve spojovacím řetězci k redefinici obou hodnot.

- Pokud se potřebujete připojit k serveru který používá nestandardní nastavení obou parametrů, a chcete se vyhnout specifikaci jednoho z parametrů v připojovacím řetězci, můžete v konfiguračním souboru specifikovat oba parametry.

Lokalizace souborů Firebirdu na klientovi

Pokud chcete na klientovi použít soubor firebird.conf, musíte zajistit aby ho klientská knihovna mohla nalézt. Je nutné definovat kořenový adresář Firebirdu, a informovat systém o jeho umístění. K tomu použijte proměnnou prostředí FIREBIRD. Klienti na platformě Windows mohou alternativně použít klíč Registry. Při správném nastavení můžete rovněž použít lokální soubor hlášení (firebird.msg).

Konfigurace souboru Services

Není nutné specifikovat záznam v souboru Services pro klienta nebo server, pokud server používá implicitní hodnoty (gds_db a 3050). Pokud je název služby gds_db a nelze ji vyhodnotit pro jiný port, je automaticky mapována na port 3050.

Pokud provádíte konfiguraci pro jiný port nebo název služby, musí být jak **server**, tak i **klient** explicitně aktualizovány na příslušné nastavení. Pro systémy Linux i Windows je tato informace uložena v souboru **Services**.

Umístění souboru Services

- Na Windows NT/2000/XP/S2003 jde o soubor c:\winNT\system32\etc\services
- Na Windows 95/98/ME jde o soubor c:\widnows\services
- Na Linuxu jde o soubor /etc/services

Záznam o službě vypadá následovně:

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

Otevřete soubor v textovém editoru a přidejte nový řádek nebo upravte existující záznam pro gds_db:

- Pro Firebird 1.0.x server nebo klient upravte název služby nebo číslo portu tak, aby odpovídalo nastavení pod kterým bude server pracovat.
- Pro Firebird 1.5 a novější změňte nebo přidejte řádek dle potřeby. Pokud máte na stejném počítači instalovaný Firebird 1.0.x nebo InterBase, ponechte původní nastavení které potřebují a přidejte nový řádek s odpovídajícím nastavením pro práci Firebirdu.

Další informace

Více informací o relačním databázovém systému Firebird naleznete na:
http://firebird.sourceforge.net

nebo některém ze spřízněných serverů:
http://firebirdsql.org
http://www.ibphoenix.com
http://www.cvalde.net/

Pokud se chcete zapojit do vývoje systému Firebird, nebo chcete upozornit na možnou chybu v produktu, můžete se přihlásit do diskusního fóra firebird-devel. Pro přihlášení jednoduše zašlete prázdný e-mail na adresu:

firebird-devel-request@lists.sourceforge.net

se slovem 'subscribe' v subjektu zprávy.

Fórum firebird-devel není určeno pro technickou podporu!

Pro technickou podporu použijte fórum firebird-support. Přihlásit se můžete na adrese:
http://www.yahoogroups.com/groups/firebird-support

pro vývoj a podporu produktů InterClient a JayBird je určena specializovaná e-mailová konference:
http://www.yahoogroups.com/groups/Firebird-Java

Fórum firebird-support je určeno pro dotazy týkající se serverů Firebird a InterBase(R). Dotazy týkající se Delphi a dalších vývojových prostředí směřujte prosím do odpovídajícího fóra určeného pro tyto produkty.

Open Source komunita provozuje několik dalších diskusních konferencí zaměřených na různé aspekty vývoje Firebirdu. Více informací o těchto diskuzních fórech naleznete na webových stránkách projektu Firebird.

Konference Firebird-devel, Firebird-support a řada dalších konferencí je rovněž dostupná v podobě newsgroups na serveru
news://news.atkin.com

Placená technická podpora je dostupná celosvětově prostřednictvím společnosti IBPhoenix (kontaktní adresy a telefonní čísla naleznete na http://www.ibphoenix.com). Několik členů vývojového týmu Firebirdu je rovněž k dispozici pro technickou podporu a konzultace. Kontaktujte je prosím přímo.

Služby obnovy databází pro Firebird nebo InterBase zajišťuje firma IBPhoenix. Pokud si chcete analyzovat a případně opravit poškozené databáze sami, zkuste produkt IBSurgeon dostopný na IBase.ru (www.ibase.ru)
IBase.ru rovněž nabízí služby obnovy dat pro Ruskou Federaci a v Evropu.

Žádosti/nabídky na sponzorování vylepšení Firebirdu můžete směřovat přímo na FirebirdSQL Foundation Inc. E-maily adresujte na foundation@firebirdsql.org. Pokud chcete nejdříve kontaktovat administrátory projektu Firebird- adresujte e-mail na firebirds@users.sourceforge.net.

Obecná diskuse o vylepšeních FB je realizována skrze konferenci Firebird-priorities (http://www.yahoogroups.com/community/Firebird-priorities.

Konference IB-Architect (http://www.yahoogroups.com/community/ib-architect) je určena POUZE pro **diskuse nad technickým návrhem.**

Nástroje a ovladače

Aplikace pro administraci databází

Několik vynikajících administrátorských nástrojů pro Firebird s grafickým rozhranním naleznete například na stránce Contributed Downloads na http://www.ibphoenix.com. Některé jsou open source, některé freeware, ostatní jsou zavedené komerční produkty.

IBConsole firmy Borland není doporučeným administrátorským nástrojem pro Firebird 1.5.

Ovladače a komponenty

JAVA: JDBC ovladač Jaybird je vyvíjen jako součást projektu Firebird. K dispozici je Type 4 JDBC/JCA driver a driver Type 2 je ve fázi beta. Zdrojové texty i binární distribuce lze stáhnout ze stránek projektu Firebird na-

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

Pokud potřebujete poradit, nebo se chcete zapojit do vývoje a testování, přihlaste se do fóra Firebird-java na http://www.yahoogroups.com/community/firebird-java.

.NET: V rámci projektu Firebird probíhá rovněž vývoj.NET ovladače. Zdrojové texty a binární distribuce lze stáhnout ze stránek projektu Firebird na -

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

Pokud potřebujete poradit, nebo se chcete zapojit do vývoje a testování, přihlaste se do fóra pro Firebird .NET na: http://lists.sourceforge.net/lists/listinfo/firebird-net-provider

Delphi a C++Builder: Uživatelé mají na výběr ze dvou mocných alternativ pro úplnou, přímou konektivitu přes Firebird 1.5 API, obě s výbornou podporou:

- IB Objects (autor Jason Wharton) na http://www.ibobjects.com
- FIBPLus na http://www.devrace.com

C++: Freeware knihovna pro C++ 'IBPP' (http://www.ibpp.org), MPL licence, k dispozici na sourceforge.net, plná přenositelnost mezi Win32 a Linuxem a pravděpodobně dalšími POSIX platformami. Užitečná, pokud vyžadujete nízkoúrovňový přístup přes C-API, ale s odlehčenou abstrakcí na úrovni C++, která není přímo svázána s konkrétním vývojovým prostředím.

ODBC: Seznam ODBC ovladačů naleznete na stránce Contributed Downloads na http://www.ibphoenix.com. Laboratoř pro probíhající vývoj open source ODBC/JDBC ovladače naleznete na: http://lists.sourceforge.net/lists/listinfo/firebird-odbc-devel

PHP: V současnosti probíhají práce na adaptaci staré rozšiřující knihovny PHP určené pro InterBase na přímou podporu Firebirdu 1.5. Více informací o tomto projektu naleznete ve fóru Firebird-PHP na http://www.yahoogroups.com/community/firebird-php

PYTHON: KInterbasDB je rozšiřující balík pro Python který implementuje podporu pro Firebird v souladu s Python Database API 2.0. Plná nativní podpora Firebird API; ve stabilní verzi a aktivně udržovaný. BSD open source licence. Download novinky na http://kinterbasdb.sourceforge.net/

Dokumentace

Dokumentace pro InterBase v 6.0 lze aplikovat rovněž na aktuální verzi Firebirdu. Betaverzi manuálů k InterBase(tm) 6 ve formátu PDF lze získat na

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

Strukturovaný seznam dokumentace je spravován na stránkách projektu Firebird na

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

Tento seznam je ve výstavbě, a jakákoliv rozšíření jsou vítána – zašlete zprávu na firebird-docs@lists.sourceforge.net

Různé návody a postupy lze rovněž najít v části vyhrazené dokumentaci na stránkách
http://www.firebirdsql.org
nebo mnohem příměji
http://sourceforge.net/projects/firebird

Hlavní repozitoří pro uživatelskou a technickou dokumentaci je server IBPhoenix -
http://www.ibphoenix.com
IBPhoenix rovněž pravidelně vydává komplexní „subscription" CD (jednotlivá vydání jsou rovněž dostupná) které rovněž obsahuje dokumentaci publikovanou touto firmou– Using Firebird a The Firebird Reference Guide.

Další dokumentaci můžete nalézt prozkoumáním oblasti pro technické publikace firmy Borland:
http://www.borland.com/techpubs/interbase/

Opravy chyb a rozšíření od verze 1.0

Evidenční číslo	Popis	Autor
(bez čísla)	Opravena drobná nekonzistence v názvech znakových sad.	P. Jacobi
(bez čísla)	Zhroucení GSTAT při určité kombinaci přepínačů.	D. Yemanov
(bez čísla)	Opraveno chybné zpracování bodů návratu v BREAK|LEAVE/EXIT.	D. Yemanov
(bez čísla)	Optimalizátor upraven tak, aby upřednostňoval jednoduché indexy před kompozitními, a preferoval unikátní indexy s úplnou shodou klíčů.	A. Brinkman
(bez čísla)	Vylepšení Win32 instalačních nástrojů instsvc.exe a instreg.exe	O. Mascia
Vylepšení	Maximální počet indexů na tabulku zvýšen z 64 na (DB_PAGE_SIZE/16)-2	A. Harrison, Portace do 1.5 N.Samofatov
721792	Dlouhotrvající transakce způsobuje mem. leak na úrovni zařízení jádra OS	N. Samofatov
775003	UDF log(x, y) ve skutečnosti vrací log(y, x)	P. Vinkenoog, N. Samofatov
774987	UDF ltrim(") a rtrim(") vrací NULL; rtrim zapomíná 1 znak	P. Vinkenoog, N. Samofatov
(bez čísla)	Oprava zhroucení serveru způsobeného ztrátou transakčního kontextu.	A. Peshkoff
(bez čísla)	Oprava zhroucení serveru při jakékoliv kombinaci sub-select & between.	A. Peshkoff
736318	"<value> STARTING WITH <field>" selže při použití indexů.	D. Yemanov
(bez čísla)	Ohlášen neexistující deadlock po provedení pre-(update/delete) spouště.	A. Peshkoff
Vylepšení	Definice klíčových slov INSERTING/UPDATING/DELETING jako nerezervovaných.	N. Samofatov
Vylepšení	Přidána nová (více specifická) chybová hlášení pro některé změny u v1.5.	D. Yemanov, A. Brinkman, A. Peshkoff
Bezpečnostní záplata	Přidán přepínač -login do instsvc dovolující instalovat služby FB jako pro jiný než lokální systémový účet.	A. Peshkoff
Vylepšení	Znovuzavedení odstraňování mezer u VARCHAR položek v síťovém protokolu.	D. Yemanov

Evidenční číslo	Popis	Autor
(bez čísla)	Náhodné hroucení serveru při přípravě velkých dotazů.	D. Yemanov
Vylepšení	Vylepšení konfigurace – uzpůsobení zprávy adresářových cest v souboru firebird.conf požadavkům OS.	A. Peshkoff
(bez čísla)	Chybné argumenty UDF typu DATE/TIME (dialekt 3).	Oleg Loa
(bez čísla)	Možné porušení referenční integrity.	Vlad Horsun, D. Yemanov
745090 a ostatní instalační problémy RC 2	Problémy s oprávněními u firebird.conf (SF #745090). generování aliases.conf při instalaci; použití rpmbuild pro vytvoření Linuxových balíků.	Erik S. LaBianca, N. Samofatov
(bez čísla)	Umožnění snadné úpravy LockSemCount na POSIX platformách. Není nutné použít gds_drop nebo restartovat počítač, aby nové nastavení vstoupilo v platnost.	N. Samofatov
Vylepšení	Definice klíčových slov FIRST/SKIP jako nerezervovaných.	N. Samofatov
(bez čísla)	Chybný odkaz na připojení po výjimce v PSQL.	A. Peshkoff
(bez čísla)	Příkazy BREAK/LEAVE a EXIT lze nyní používat ve spouštích.	D. Yemanov
(bez čísla)	Možné porušení indexů v průběhu odstraňování smetí.	Vlad Horsun, D. Yemanov
(bez čísla)	Vyřešen problém se správou dočasných souborů: 1) Bezpečnostní díra na všech POSIX platformách mimo FREEBSD/OPENBSD vázaná na použití mktemp (možnost DoS útoků nebo elevace oprávnění) Generováno pouze 27 unikátních jmen na win32 (což může způsobit nepředvídatelné chování u SS)	N. Samofatov
(bez čísla)	Změna manažeru událostí: zakázáno použití konkrétního aux portu v CS kvůli zjištěným problémům.	D. Yemanov
(bez čísla)	Umožnění použití agregačních funkcí z různých kontextů v rámci jiné agregační funkce. Příklad: SELECT MAX((SELECT COUNT(*) FROM RDB$RELATIONS)) FROM RDB$RELATIONS	A. Brinkman
(bez čísla)	Možné zhroucení při odpojování databáze při použití událostí.	D. Yemanov
(bez čísla)	Změny správce služeb: vlastnosti GSTAT/GSEC nejsou dostupné přes Services API u win32 CS (až do verze v1.6).	D. Yemanov
(bez čísla)	Hlášení chybných statistik pokud operace z nějakého důvodu selže.	D. Yemanov

Evidenční číslo	Popis	Autor
(bez čísla)	U Win32 verze GBAK nelze použít stdin/stdout k přesměrování výstupu.	A. Peshkoff
(bez čísla)	Nefunkční změna velikosti tabulky zámků u CS. Už žádné chyby "lock manager out of room" (Win32 CS 1.5 RC1) nebo zhroucení (možné ve všech ostatních CS verzích Interbase/Firebird).	N. Samofatov
Vylepšení	Vylepšení INTL: funkce UPPER funguje pro znakovou sadu WIN1251 bez explicitní specifikace collations.	N. Samofatov, D. Yemanov
BUGCHECK(291)	Možné poškození databáze při změně/zrušení stejného řádku v BEFORE spouští.	A. Peshkoff
(bez čísla)	Přepsání bufferu ve volání isc_database_info().	Oleg Loa
(bez čísla)	Změna konfiguračního manažeru: nyní je server ukončen při chybějícím / chybné firebird.conf s chybovým hlášením v systémovém logu.	A. Peshkoff
(bez čísla)	Oprava Services API: povolení statistického Services API pro POSIX CS.	N. Samofatov
(bez čísla)	Změna parseru. 1) ROWS_AFFECTED přejmenováno na ROW_COUNT 2) CONNECTION_ID/TRANSACTION_ID přejmenováno na CURRENT_CONNECTION/CURRENT_TRANSACTION 3) některá nová klíčová slova definována jako nerezervovaná	D. Yemanov
(bez čísla)	Oprava Services API: částečné povolení Services API pro win32 CS.	D. Yemanov
(bez čísla)	Chybný způsob zasílání zpráv (zbytečné použití OOB paketů).	Jim Starkey, Paul Reeves
(bez čísla)	Vylepšení správce zámků: vzájemné zablokování je nyní detekováno a ohlášeno v okamžiku kdy všechny blokující procesy dostanou oznámení, tedy okamžitě ve všech normálních případech	N. Samofatov
(bez čísla)	Server havaruje při některých operacích Services API.	A. Brinkman
(bez čísla)	Pokročilé techniky zabezpečení: implementován konfigurovatelný přístup k databázím, externím souborům a UDF knihovnám.	A. Peshkoff
(bez čísla)	Oprava resource/memory leaks.	Mike Nordell, A. Peshkoff, N. Samofatov, D. Yemanov
(bez čísla)	Přepsání bufferu u multidimenzionálních polí.	D. Yemanov

Evidenční číslo	Popis	Autor
213460, 678718	Různé problémy s událostmi na multihomed hosts. POZNÁMKA Nyní je rovněž možné definovat specifický port pro zasílání zpráv.	D. Yemanov
(bez čísla)	Odstranění několika resource leaks.	Mike Nordell, A. Peshkoff
(bez čísla)	Oprava Services API: povolení Services API pro posix CS. Poznámky: 1. příslušné změny u Win32 CS nejsou hotové 2. Služba backup/restore je opravena, otestována a měla by fungovat 3. validace databáze je z části opravena a měla by fungovat 4. Ostatní služby v CS pravděpodobně nefungují	N. Samofatov
(bez čísla)	Vylepšení SQL: povolení NULL v omezení na jedinečnost a jedinečných indexech (SQL-99 spec).	D. Yemanov, N. Samofatov
(bez čísla)	Zlepšení výkonu: VIO undo log nyní používá pro uložení savepoint záznamů B+ strom. To zlepšuje výkon při vícenásobných změnách záznamů v jediné transakci (zhruba 2-3 násobně pro 100000 záznamů).	N. Samofatov
(bez čísla)	Poškození databáze při návratu odstranění změn savepoint po velkém množství DML operací (takže savepoint na úrovni transakce je zrušen) a záznam je aktualizován _nikoliv_ pod savepointem a smazán pod savepointem.	N. Samofatov
(bez čísla)	Vylepšení EXECUTE STATEMENT. Nyní je možné vracet hodnoty z dynamického SQL. Syntaxe: EXECUTE STATEMENT <value> INTO <var_list>; (singleton form) nebo FOR EXECUTE STATEMENT <value> INTO <var_list> DO <stmt_list>;	A. Peshkoff
(bez čísla)	Server zamrzne během odpojování od databáze po masivní aktualizaci.	D. Yemanov
(bez čísla)	Vylepšení optimalizátoru: Podváběry v SET klauzuli příkazu UPDATE nyní mohou používat indexy.	A. Brinkman
(bez čísla)	Chyba "Context already in use" v případě DISTINCT s podvýběry.	A. Brinkman
(bez čísla)	Vylepšení zjišťování databázových informací: seznam právě aktivních transakcí je nyní dostupný přes volání isc_database_info.	N. Samofatov

Evidenční číslo	Popis	Autor
(bez čísla)	Zlepšení výkonu: zkrácené vyhodnocování logických výrazů. POZNÁMKA Chování je řízeno parametrem "CompleteBooleanEvaluation" v firebird.conf. Implicitní hodnota je 0 (zkrácené vyhodnocení).	Mike Nordell
(chyba Beta 2)	Přetečení zásobníku v průběhu přípravy příkazu.	D. Yemanov, Mike Nordell
(bez čísla)	Zlepšení výkonu na IA32 CPU: zrychlení operací s indexy	Mike Nordell
(bez čísla)	Změna v univerzálních spouštích: povolen přístup k oběma (OLD a NEW) kontextům.	D. Yemanov
(bez čísla)	Vylepšení optimalizátoru: Pokud je k dispozici uzel ekvivalence spolu s ostatními uzly (geq, leq, between...) pro výběr podle indexu, použije se vždy uzel ekvivalence místo jiných uzlů.	A. Brinkman
(bez čísla)	Dlouhé pauzy při připojování/odpojování na WinXP.	A. Brinkman
(bez čísla)	Obecné čištění: odstraněno velké množství nepoužitého kódu.	Blas Rodriguez Somoza, Erik Kunze
523589	View ovlivňuje výsledek dotazu. Poznámka: problém byl, že RSE uzly (uvnitř view) nebyly označeny jako proměnlivé.	A. Brinkman
(bez čísla)	Změna chování forced writes módu: nyní je možné při vypnutém FW určit jak často se mají změněné stránky zapisovat na disk (flush) (zvyšuje spolehlivost na Win32 při vypnutém FW).	Blas Rodriguez Somoza
(bez čísla)	Bezpečnostní databáze byla přejmenována na security.fdb.	D. Yemanov
(bez čísla)	Nový konfigurační soubor: firebird.conf.	D. Yemanov
(bez čísla)	Do knihovny IB_UDF přidány nové uživatelské funkce LPAD a RPAD.	Juan Guerrero
(bez čísla)	GFIX někdy nedovoluje specifikovat přepínače "-user" a "-password" (chyba "incompatible swiches").	D. Yemanov
(bez čísla)	Vyrovnávací paměť připojení k bezpečnostní databázi: připojení jsou nyní ve vyrovnávací paměti, což snižuje čas na přihlášení k databázi.	D. Yemanov
Vylepšení	1. Snížena spotřeba paměti serverem. 2. Přímé externí I/pokud není k dispozici paměť pro třídění. Zvýšen počet proudů a predikátů zpracovávaných optimalizátorem.	D. Yemanov
508594	LEFT JOIN s VIEW: jednoduchý LEFT JOIN na VIEW s jedinou ON klauzulí nepoužíval index ačkoliv to bylo možné.	A. Brinkman

Evidenční číslo	Popis	Autor
(bez čísla)	Nové znakové sady: 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 POZNÁMKA Collations pro výše uvedené znakové sady nejsou zatím k dispozici.	Blas Rodriguez Somoza
(bez čísla)	Změny v CREATE VIEW: zakázána PLAN klauzule.	D. Yemanov
(bez čísla)	Změna chování zpracování agregací -- zavedena zpětná kompatibilita s agregacemi. Nejvnitřnější pole v agregaci určuje kam patří agregační kontext.	A. Brinkman
(bez čísla)	Vylepšený optimalizátor: lepší optimalizace "komplexních" JOIN dotazů (LEFT JOIN, pohledy, SP, atd.).	A. Brinkman
(chyba alpha 5)	Odstraněny závažné memory leaks.	D. Yemanov
(bez čísla)	Nové API funkce: funkce pro zjištění verze klientské knihovny v souladu s IB7 -- isc_get_client_version(), isc_get_client_major_version(), isc_get_client_minor_version()	D. Yemanov
(bez čísla)	Vylepšení třídění/spojování: spojování (SORT MERGE plány) je nyní prováděno přes modul třídění v paměti.	D. Yemanov
(bez čísla)	Změna vnitřností nového správce paměti pro vyšší výkon.	N. Samofatov
(bez čísla)	Změny vytváření Win32: 1. Změna jmen USER32 objektů pro umožnění souběžného běhu s IB/FB1. 2. Změněno jméno mapované oblasti pro lokální (IPC) protokol, takže v1.5 klientské knihovny již není kompatibilní s předchozí verzí při přístupu přes IPC. 3. Jména všech transportních protokolů (INET port a service, WNET pipe, IPC map) jsou nyní konfigurovatelná přes firebird.conf.	D. Yemanov
(bez čísla)	Poškození RDB$FIELD_LENGTH u pohledů které obsahují spojování dlouhých CHAR/VARCHAR položek.	D. Yemanov

Vylepšení	Vylepšení spouští: přidána kontrola za běhu (predikáty INSERTING/UPDATING/DELETING).
	
	Příklad:
	
	if (INSERTING) then
	
	new.OPER_TYPE = 'I';
	
	else
	
	new.OPER_TYPE = 'U';
		D. Yemanov

Evidenční číslo	Popis	Autor
(bez čísla)	Kurzory (klauzule WHERE CURRENT OF) nemohou být použity ve spouštích.	D. Yemanov
221921	ORDER BY nemá žádný efekt.	A. Brinkman
213859	Podvýběr spojený s 'IN' klauzulí.	A. Brinkman
Vylepšení	Povolení libovolných výrazů v klauzuli ORDER BY.	N. Samofatov
(bez čísla)	Server havaruje pokud je použit UNION ve VIEW a tento VIEW je použit ve WHERE klauzuli podvýběru.	A. Brinkman
(bez čísla)	Obecné čištění kódu: struktury uvnitř Y-ventilu.	A. Peshkoff, N. Samofatov
Vylepšení	Jednořádkové komentáře (--) jsou nyní povoleny na jakékoliv pozici v SQL příkazu.	D. Yemanov
(bez čísla)	"Request sycnhronization error" s příkazem BREAK.	D. Yemanov
625899	Bugcheck 291.	A. Peshkoff
(bez čísla)	Změna PSQL: EXECUTE VARCHAR přejmenován na EXECUTE STATEMENT.	A. Peshkoff
521952	Žádný aktuální záznam pro fetch operaci.	A. Brinkman
(bez čísla)	QLI nezná typ BIGINT.	D. Yemanov
(bez čísla)	Délka textové proměnné v proceduře/spoušti není kopírována do popisné struktury.	A. Brinkman
(bez čísla)	Změny FIRST/SKIP a ORDER BY -- 1. Implementace klauzule ORDER BY v podvýběrech. 2. Zakázání FIRST/SKIP pro views. 3. Povolení hodnoty nula jako platného argumentu pro FIRST.	D. Yemanov
(bez čísla)	Přetečení bufferu (MAXPATHLEN).	Erik Kunze
(bez čísla)	Sladění SQLDA mapování parametrů s pořadím a počtem parametrů ve zdrojovém SQL řetězci. POZNÁMKA Lze povolit původní způsob mapování (pro zpětnou kompatibilitu) použitím konfiguračního parametru "OldParameterOrdering".	N. Samofatov
Vylepšení	Vylepšený optimalizátor: podvýběry rovněž používají indexy pokud je nadřízená množina uloženou procedurou.	A. Brinkman
(bez čísla)	Odstraněno omezení velikosti požadavku.	D. Yemanov

Evidenční číslo	Popis	Autor
(bez čísla)	Řazení Null první/poslední a zpracování collation v "order by" klauzuli unionů	N. Samofatov
(bez čísla)	Obecné čištění kódu; přejmenování, nová bezpečná makra, podpora pro mingw.	Erik Kunze, Ignacio J. Ortega, D. Sibiryakov
(bez čísla)	Dokončena implementace explicitního zamykání. Nyní by mělo být funkční a konzistentní.	N. Samofatov
Vylepšení	Vylepšený optimalizátor: lepší zpracování AND uzlů uvnitř OR uzlu.	A. Brinkman
(bez čísla)	Výjimky uvnitř for/while smyček nejsou zpracovány korektně.	A. Peshkoff
623992	Dvojité lomítko v připojovacím řetězci.	Paul Reeves, Mark O'Donohue
(bez čísla)	Vzájemné zablokování při některých operacích s metadaty.	A. Peshkoff
Vylepšení	Vylepšený optimalizátor: Pokud může být použito více indexů s odlišnou selektivitou, je použit pouze nejlepší a ostatní ignorovány.	D. Yemanov
(bez čísla)	Problém s Quoted identifiers ve specifikaci plánu.	N. Samofatov
Vylepšení	Architektura CS je nyní podporována I na Win32, ale stále nemůže být považována za stabilní.	D. Yemanov
(bez čísla)	Uložené procedury již nejsou před smazáním rekompilovány.	N. Samofatov
(bez čísla)	Nový collation pro WIN1251: WIN1251_UA pro ukrajinštinu a ruštinu.	D. Yemanov
(bez čísla)	Změna klientské knihovny: API rutiny již nejsou exportovány podle čísel.	D. Yemanov
Vylepšení	Nový konfigurační manažer: zpřístupňuje stejnou konfiguraci prostým textovým souborem na všech platformách.	D. Yemanov
Vylepšení	Vylepšený optimalizátor: přidána lepší podpora zpracování indexů s "OR". Vybere nejlepší dostupný složený index ze všech "AND" uzlů.	A. Brinkman
Vylepšení	Přidána podpora správy explicitních savepointů v DSQL.	N. Samofatov
(bez čísla)	Čištění protokolu: IPX/SPX již není podporován.	Sean Leyne
(bez čísla)	Čištění nepodporovaných platforem: některé platformy již nejsou současným kódem podporovány: DELTA, IMP, DG_X86, M88K, UNIXWARE, Ultrix, NeXT, ALPHA_NT, DGUX, MPE/XL, DecOSF, SGI, HP700, Netware, MSDOS, SUN3_3	Sean Leyne
Vylepšení	Vylepšený optimalizátor: přidána podpora detekce použití indexu s podvýběrem v agregačním výběru.	A. Brinkman

Evidenční číslo	Popis	Autor
Vylepšení	Vylepšený plánovač vláken pro Win32 SS: nyní by měl server pod vysokou zátěží lépe reagovat.	A. Peshkoff
Vylepšení	Přidána podpora explicitního zamykání. Syntaxe: SELECT <...> [FOR UPDATE [OF col [, col ...]] [WITH LOCK]]	N. Samofatov
558364	Kompilace spouště selže pokud je specifikován PLAN.	Ignacio J. Ortega
(bez čísla)	Distribuované (2PC) transakce nemohou být korektně odvolány pro chyby na síti.	Vlad Horsun, Erik Kunze
(bez čísla)	Obecné čištění kódu: makra ISC_STATUS_LENGTH a MAXPATHLEN.	Erik Kunze
496784	Pokud optimalizátor nalezne index pro LEFT JOIN, pracuje jako INNER JOIN. Opraven problém který způsoboval chybné výsledky u komplexních spojování.	N. Samofatov
(bez čísla)	Podtyp BLOB je ignorován v systémových doménách generovaných pro vypočítané položky u view.	D. Yemanov
(bez čísla)	Opravena instalační chyba: instreg.exe nevytvoří hodnotu "GuardianOptions" v registry.	D. Yemanov
(bez čísla)	Resource leaks ve zpracování DDL rekurzivní procedury který způsoboval selhání některých DDL.	N. Samofatov
(bez čísla)	Check omezení které používá pouze jedinou položku tabulky je nyní zrušeno při zrušení této položky.	N. Samofatov
451927	Nová systémové proměnná ROWS_AFFECTED v PSQL: vrací počet řádků ovlivněných posledním příkazem INSERT/UPDATE/DELETE. Pra příkazy jiné než INSERT/UPDATE/DELETE vrací vždy hodnotu nula.	D. Yemanov
446240	Dynamická hlášení výjimek: dovoluje vyvolat výjimku s jiným než hlášením definovaným při vytvoření výjimky. Syntaxe: EXCEPTION name [value];	D. Yemanov
547383	Nové systémové proměnné SQLCODE a GDSCODE poskytují přístup ke kódu zachycené chyby v WHEN-bloku v PSQL. Mimo WHEN-blok vždy vrací hodnotu 0 (úspěch).	D. Yemanov
(bez čísla)	Sémantika reinicializace výjimky: dovoluje opětovné vyvolání již zachycené výjimky v PSQL z WHEN-bloku. Syntaxe: EXCEPTION; Mimo WHEN-blok nemá efekt.	"Digitman"
(bez čísla)	Server havaruje během odstraňování smetí pod velkou zátěží.	N. Samofatov

Evidenční číslo	Popis	Autor
Vylepšení	Odložená kompilace metadat: řeší mnoho případů dobře známé chyby "object in use".	N. Samofatov
Vylepšení	Nové zpracování řazení NULL: dovoluje uživatelsky definované řazení NULL.	N. Samofatov
(bez čísla)	gstat zobrazuje chybné číslo pro maxdup.	D. Kuzmenko
(bez čísla)	Nový klíč registry pro win32: SOFTWARE\FirebirdSQL\Firebird.	--
451925	Uživatelsky definovaná jména indexů pro omezení: umožňuje pojmenovat index zajišťující omezení stejně jako omezení nebo uživatelským jménem.	D. Yemanov
Vylepšení	Nový příkaz RECREATE VIEW: zkratka pro spojení DROP VIEW / CREATE VIEW. Syntaxe: RECREATE VIEW name <view_definition>;	D. Yemanov
(bez čísla)	Spouště jejichž jméno začíná 'RDB$' nelze změnit ani zrušit.	D. Yemanov
(bez čísla)	Přejmenování distribučních souborů protože jsme Firebird. Nyní se jmenují fbserver, fbclient, firebird.msg atd. Klientská knihovna je nyní fbclient a měla by být používána ve všech nových projektech. gds32 neobsahuje nic jiného než přesměrování exportů a je k dispozici pouze pro zpětnou kompatibilitu.	Various
(malá aktualizace ODS)	Přidány nové systémové indexy (RDB$INDEX_41, RDB$INDEX_42, RDB$INDEX_43), ODS verze je nyní 10.1.	D. Yemanov, N. Samofatov
451935	Nový příkaz CREATE OR ALTER pro spouště a uložené procedury, umožňuje vytvořit nebo změnit databázové objekty v závislosti na tom , zda existují nebo ne. Syntaxe: CREATE OR ALTER name <object_definition>;	D. Yemanov
(bez čísla)	Po změnách metadat se v databázi objevují porušené závislosti (jako DB$34).	D. Yemanov
(bez čísla)	Vylepšená deklarace lokálních proměnných: zjednodušuje syntaxi a dovoluje současně proměnnou deklarovat a definovat její hodnotu. Syntaxe: DECLARE [VARIABLE] name <variable_type> [{'=' | DEFAULT} value]; příklad: DECLARE my_var INTEGER = 123;	Claudio Valderrama

Evidenční číslo	Popis	Autor
(bez čísla)	Zakázán příkaz BREAK pro spouště (jako EXIT) kvůli známým interním omezením.	D. Yemanov
555839, 546274	Vylepšené skupinování: umožňuje použít interní funkce a podvýběry v GROUP BY. Rovněž dovoluje definovat GROUP BY ordinal (tzn.. pozice sloupce).	A. Brinkman
451917	Nová interní funkce COALESCE.	A. Brinkman
451917	Nová interní funkce NULLIF.	A. Brinkman
451917	Nová interní funkce CASE.	A. Brinkman
545725	Zaseknutí automatického sweep na pozadí.	A. Peshkoff
(bez čísla)	Server havaruje pokud nejsou XSQLDA strujtury inicializovány pro všechny parametry příkazu.	D. Yemanov
(bez čísla)	PSQL: přidána podpora pro prázdné bloky BEGIN...END.	D. Yemanov
567931	Částečná oprava bezpečnostní díry v metadatech.	D. Yemanov
(bez čísla)	BigInt arrays nefungují.	Artem Petkevych
437859	Implementováno spojení volání procedury a řetězce, dovoluje použití libovolného výrazu jako parametru SP.	D. Yemanov
562417	Agregace spojovaného prázdného řetězce.	D. Yemanov
Vylepšení	Přidána podpora readline (historie příkazů) do ISQL.	M. O'Donohue
446206	Nový datový typ BIGINT umožňuje nativní SQL použití 64-bitových přesných čísel (pouze dialekt 3).	D. Yemanov
451922	Univerzální spouště umožňují aby byla jediná spoušť spuštěna pro několik typů operací.	D. Yemanov
446238, 446243	Nové systémové proměnné CONNECTION_ID a TRANSACTION_ID v PSQL. Vracejí odpovídající interní identifikátor uložený na hlavičkové stránce databáze.	D. Yemanov
446180	Alias databází na straně serveru: umožňuje připojení k databázi prostřednictvím "alias" jména místo specifikace fyzického jména/cesty. Seznam známých alias databází je uložen v souboru aliases.conf v kořenovém adresáři instalace. příklad: Záznam alias v konfiguračním souboru: my_database = c:\dbs\my\database.gdb Připojovací řetězec v aplikaci: localhost:my_database	D. Yemanov
(bez čísla)	Nový plugin manažer a INTL rozhranní.	John Bellardo

Evidenční číslo	Popis	Autor
Vylepšení	Třídění v paměti: Pokud je v SQL příkazu použit SORT plán, je třídění realizováno v paměti. Pokud není dostatek paměti, je použito původní třídění s použitím dočasných souborů.	D. Yemanov
538201	Havárie při funkci extract z null jako datumu.	Claudio Valderrama
446256	Nové rozšíření PSQL EXECUTE VARCHAR umožňuje provádět dynamické SQL příkazy v SP/spoušti. (následně přejmenováno na EXECUTE STATEMENT).	A. Peshkoff
(bez čísla)	Zásadní čištění kódu.	Sean Leyne, Erik Kunze
(bez čísla)	Nový manažer paměti.	John Bellardo
(bez čísla)	Nová logika zpracování výjimek.	Mike Nordell, John Bellardo
(bez čísla)	Nová konfigurace kompilace s použitím autoconf.	John Bellardo, M. O'Donohue, Erik Kunze
(bez čísla)	Převod kódu z C do C++.	Mike Nordell, John Bellardo, M. O'Donohue