Firebird Version 1

Release Notes

March 2002

原文： http://www.firebirdsql.org/index.php?op=files&id=fb10
監修： 加藤大受
翻訳：（株）毎日コミュニケーションズ　書籍編集３課（book3@mycom.co.jp）
編集：（株）ドキュメントシステム

目次

一般的な情報

Firebird データベースエンジンは、2000 年 6 月 25 日に Borland Software 社によって公開された InterBase™のソースコードを基にして開発されました。現在でも、ボランティア開発者の独立したチームによって開発が続けられています。

このリリースは、約 1 年間のベータテストと、広範囲に渡るバグフィックスを経たソースコードからビルドされています。

バグフィックスのリストは、この文書の後半にあります。

いくつかの新しい言語のサポートが追加されています。構文については、この文書の後

半にある「言語の拡張」セクションを参照してください。

Firebird のバイナリは、FirebirdSQL.org からダウンロードすることができます。
http://www.firebirdsql.org/
（訳注：現在は、**http://firebird.sourceforge.net/**にリダイレクトされます）

参考となるドキュメントの場所については、このリリースノートの最後の「ドキュメント」を参照してください。

互換性

現在 InterBase データベースを使っており、Firebird を試しに使ってから再び InterBase に戻そうと考えているならば注意が必要です。必ず InterBase のバックアップをとり、データベースを復元するための考えうる限りの予防策をとってください。
この後の「言語の拡張」セクションの Ib_udf 関数ライブラリに記載されている関数が変更されていることに注意してください。
InterBase 4.x/5.x/server（10 以下の ODS）、あるいは viz 下で現在動作しているデータベースを移行するとき、次のソフトウェアは 100%の上位互換性があります。

- gbak（Windows では **gbak.exe**）の該当するバージョンを使用して、既存のデータベースの ODS に正確な InterBase のバックアップファイルを作成することができます。このファイルには、**gbk** という拡張子をつけることが推奨されています。Windows 用のバージョン 4 と 5 には、**ibmgr32.exe** というプログラムがあります。これは、［Take | Backup］メニューによって **gbak.exe** の GUI を使用することができます。このプログラムを Firebird や InterBase 6 のデータベースに対して動作させてはいけません
- 必要に応じて、バックアップファイルを Zip 形式で圧縮して、リムーバブルメディアに移動させることができます。保存されたファイルを CD-ROM から復元する場合には、読み取り専用属性を解除する必要があります
- インストールした Firebird の gbak を使って InterBase のバックアップを復元する場合は、Firebird が動作しているディレクトリに **gbk** ファイルを復元しなければなりません。ibserver プログラムとして、物理的に同じマシンに存在している必要があります
- コンバートしたデータベースは、すぐに SQL DIALECT1 の Firebird データベースとして動作させることができます

  このドキュメントの操作ガイドは、gbak のバックアップとリストアのコマンド構文を含んでいます。
  IBConsole のようないくつかのユーティリティプログラムは、ODS 10 のバックアップとリストアのためにグラフィカルユーザーインターフェイスのツールを提供します。おすすめのツールは、Martin Schmid 氏（**M.Schmid@EQUITANIA.de**）によるフリーソフトウェアのバイナリ「IBBackup」です。
  http://www.equitania.de/interbase/downloads/ibBackup.zip からダウンロードすることができます。

デスクトップクライアントプログラム

Firebird のリリースが独立して公開されるまでの約 1 カ月の間に、Borland の Interbase コミュニティは、オープンソースの IBConsole プログラムの開発に関して、わずかに要求を厳しくしました。しかし、IBConsole の主要な開発者たちは、IBConsole に Firebird のバグフィックスや拡張をサポートするパッチの導入を拒否しました。これによって、IBConsole のすべての機能が Firebird で動作する保証がなくなっ

てしまいました。

いくつかのすばらしい代替案が、http://www.ibphoenix.com/の寄贈ダウンロードページにリストアップされています。これらはオープンソースとフリーウェア、または商用プロダクトとして提供されています。
次の 2 つのフリーソフトウェアのデスクトッププログラムが推奨されています。

- Jason Wharton 氏の IB_SQL は、http://www.ibobjects.com/にて公開されています。IB_SQL は、スクリプティング、データポンピング、セキュリティメンテナンスなど、データデータベース管理ツールに必要な多くの機能をもっています
- Michael Mutl 氏の IBQuery も、洗練された GUI の SQL ツールを提供しています。これは、Torrys Web サイトから入手できます
  http://www.torry.net/apps/utilities/database/mitecibquery.zip

そのほかのフリーのツール：

- IBAccess は、IBX コンポーネントとしてビルドされ、Windows と Linux の両方のバージョンがあります。これは、www.ibaccess.org にて公開されています。何らかの互換性に関する問題に直面した場合には、作者の Toni Martir 氏（アドレスは Web サイトにあります）に連絡してください。それによって不具合が解決されるかもしれません。もちろんこれは Borland とは無関係なオープンソースプロジェクトです
- Marathon 2 は、Tilo Muetze 氏の指揮のもとでオープンソースで開発が進められています。http://www.alanti.net/firebird/marathon/の Web サイトでプロジェクトを見ることができます

Delphi/Borland C++Builder コンポーネント

Delphi や C++Builder 用の Borland InterBaseXpress（IBX）や DBExpress（DataSnap in versions 6）コンポーネントにおいても、同様の問題が予想されます。Firebird の言語や API の拡張をサポートする BDE InterBase 6 のドライバを、Borland が改修したり拡張してしまうといった、想像しにくい可能性も考えられます。
Delphi と C++Builder のユーザーには、IBX と BDE の両方の代わりに使用することができる 2 つのツールがあります。どちらも開発者から十分にサポートされています。

- http://www.ibobjects.com/にある、Jason Wharton 氏の「IB Objects」
- http://www.devrace.com/にある「FIBPLus」

Firebird のバージョン番号

バージョン番号は、Firebird 1.0 以前のバージョン情報の情報を含む多くの情報を返します。たとえば、「LI-V6.2.0.nnn Firebird」となります（nnn はビルド番号）。
これは、来るべき Firebird 2 へ向けて、2 つのコネクションモードの選択肢を用意しているためです。

- ネイティブの Firebird モードにおいては、サーバは異なるポート（デフォルトでは 3051）で通信し、「LI-V2.0.0.XXX Firebird」のように表記されるバージョン番号を返します
- 互換性オプションでは、3050 番ポートで通信し、互換性のあるバージョン番号の文字列（通常、LI-V6.2.0.368 Firebird）を返します。この互換のための文字列は、レジストリか **conf** ファイルの設定によって変更可能です

新しく加わった isc_database_info パラメータの isc_vendor 情報と isc_info_fb_

version 情報によって、いくつかの判別が可能になりました。クライアントアプリケーションが、Firebird または InterBase のデータベースのどちらに接続しているか、また接続している Firebird のバージョンは何か、などについて判別可能です。

InterClient と JDBC

Firebird-Java チームは、Firebird 用の Pure Java ドライバを実装しました。このドライバは、エンタープライズ情報システムのためのアプリケーションサーバコネクションである新しい JCA 標準と、従来の JDBC 標準のどちらにも準拠しています。

Linux 版

Linux 版のすべての Firebird は、glibc 2.2 でビルドされています。たとえば、Red Hat 6.2 や Debian などの古いバージョンの Linux は、アップグレードする必要があります。Paul Reeves 氏と Pavel Cisar 氏は、glibc 2.1 でも同じように動作する 32 ビット I/O ビルドを提供し、古い Linux 環境用での「追加的な」リリースとして使用できるか調査しています。

スレッド対応の gdslib.so（Linux 用クライアントライブラリ）には、最新の 1.0 ビルドを含んでいます。

新しい機能

依存性チェックの強化

UDF とジェネレータは、プロシージャか計算された列が使われた場合、データが失われないように監視しています。

大規模データベースファイルのサポート

論理的なデータベースは、一次ファイルといくつかの二次ファイル（単独の Firebird/InterBase のデータベースで有効なレコードは 980GB です）で構成されています。この拡張は、物理的なデータベースファイルのサイズに関係します。

最大ファイルサイズの制限は、次の 3 つの要因によるものです。

- ❏ Firebird/InterBase エンジンの I/O 関数の実装
- ❏ 64 ビットファイルオペレーションに対する OS のサポート
- ❏ ファイルサイズについてのファイルシステムの制限

64 ビットファイル I/O システムへのサポート

InterBase 6 を含む Firebird 1.0 以降のすべてのバージョンでは、32 ビットの API とシステムコール、32 ビット（整数）のファイルポインタストラクチャの両方を使用しています。このデータベースエンジンでは、最大 4GB のファイルサイズを扱うことができるように設計されています。UNIX のコードでは、ファイルポインタストラクチャは整数として定義されます。したがって、UNIX や Linux の実装では、ファイルサイズは 2GB までに制限されます。

Windows

Win32（Windows 9x/Me/NT/2000）のすべてのバージョンでは、64 ビットポインタのファイル操作がサポートされています。Firebird 用の大きなファイルサポートを追加するために、ファイルポインタストラクチャを **LARGE_INTEGER** に変更し、I/O コールをほ

んの少し変更しています。

Linux

Linux においては、64 ビットのファイル操作は、新しい 2.4kernel を使った glib 2.2 以降でサポートされています。glib 2.1.3 は、大きなファイルを扱うことについては、サポート外ながら制限付きで対応しています。Red Hat Linux 7.1 には、glib 2.2.2-10 が含まれており、2GB を超えるファイルも扱うことができます（ほぼすべてのディストリビューションでデフォルトになっている、ネイティブの Linux のファイルシステムである **ext2** を使った場合）。最近の Linux ディストリビューションであれば、2.4 Kernel、**ext2** ファイルシステム、glibc 2.2 などが含まれているので、従来の 32 ビット/2GB よりも大きなファイルを扱うことができるでしょう。

すべての UNIX/Linux で 64 ビットの大規模ファイルシステムがサポートされているわけではないので、32 ビット I/O 版と 64 ビット I/O 版の 2 種類が用意されています。使用している環境を確認してバージョンを選んでください。

基本的な動作条件は、glibc 2.2（加えてコアユーティリティをサポートしていること）と 2.4 kernel です。Red Hat 7.1、Mandrake 8.0、SuSE 7.2 とこれら以降のバージョンでは、この条件に一致しています。しかし、いくつかのディストリビューションでは、古いコンポーネントをインストールできてしまうので、注意してください。たとえば、Mandrake 8.x は、アドバンスセットアップで 2.2 kernel をインストールすることが可能となっています。

Mac OS X 10.0

Mac OS X 10.0 のバグのため、64 ビット I/O サポートの Firebird は Mac OS X 10.0 では動作しません。

ファイルシステム

現在、Firebird データベースは 64 ビット I/O をサポートしていますが、すべてのファイルシステムにおいてサポートされているわけではないことに注意してください。

Win32 の場合、次のファイルサイズ（パーティションやボリュームと混同しないこと）に制限があります。

Windows 9x/Me 上の FAT16：最大ファイルサイズ ＝ 2GB – 1byte
Windows NT/2000 上の FAT16：最大ファイルサイズ ＝ 4GB – 1byte
Windows 9x/NT/2000 上の FAT32：最大ファイルサイズ ＝ 4GB – 1byte
Windows NT/2000 上の NTFS：最大ファイルサイズ ＝ 16,384GB – 1byte

Linux や UNIX の場合、さまざまなファイルシステムが存在します。次の URL で詳細を確認してください。

http://www.suse.de/~aj/linux_lfs.html

幸いなことに、すべてのメジャーな Linux ディストリビューションでは、大規模ファイルなを完全にサポートしている **ext**2 ファイルシステムを採用しています。
次の環境が必ず必要となります。

1.kernel 2.4 以降
2.glibc 2.2 以降
3.**ext2** ファイルシステム、あるいは kernel が LFS（Large File System）インターフェイスをサポートしているほかのファイルシステムを使用していること
4.おそらく **ReiserFS** と **Ext3** でも動作可能（まだ保証はされていない）

最大 16KB のページサイズのサポート

データベースエンジンと gbak では、従来の最大 8KB だった **PAGE_SIZE** が 16KB に変更されました。ハードウェア環境に合わせたサイズのデータベースを作成してください。

Win32 Forced Writes 設定

Win32 では、InterBase 6 の導入時に新たなデータベースを作る際のデフォルトの **FORCED WRITES** 設定が OFF になっていました。しかし、現在のデフォルト設定は ON です。この設定を OFF にすると、データベースのパフォーマンスが低下し、完全性やリカバリのコストが犠牲となります。

不正な接続文字列のために発生する問題は RC 1 で解消されました。この問題の回避を意図して、マルチユーザーモードでの強制書き込みに変わってしまうということはなくなりました。

外部テーブルハンドルの限界（NT）

従来では、NT 上で同時に使用することができる外部テーブルの数の限界は、**fopens** のデフォルト数である約 510 でした。

Firebird での変更によって、NT 上の **fopens** の総数は、OS が許可する最大値である 2048 まで増加しました。同時に開ける外部テーブル数の最大数は、まだテストされていません。その数は、開いているデータベースの数、データベースごとのファイル数などに依存しますが、おおよそ 2040 程度でしょう。

外部テーブルの破棄

Deferred Work Handler モジュール（dfw.e）の小さな変更によって、外部テーブルが破棄されると同時に、外部ファイルも閉じられます。この制約によって、サーバをシャットダウンせずにテーブルを破棄する場合でも、関連するファイルを削除することができます。

接続文字でのポート指定

Feature id 1468, SF ID 447400

ポートを指定する機能は、新しい Firebird の特長です。デフォルトの 3050 以外のポートでサーバに接続するために、接続文字に代替ポートを含めることができます。構文は次の通りです。

Win32:

```
Server/Port:Drive\Directory\Database.gdb
```

Linux および UNIX:

```
Server/Port:/device/Directory/Database.gdb
```

サーバ名とポートの間は:（コロン）ではなく/（スラッシュ）であることに注意してください。物理的なパス名の前には、従来通りにコロンが必要です。

デフォルトの 3050 ポートでの接続

サービスファイル中のエントリが見つからない場合、クライアントおよびサーバは 3050 ポートを使用します。これは、クライアント/サーバで同じようにインストールす

ることで解決できます。

gbak -V（erbose）オプション

gbak -V(erbose)オプションは、次のようにカウンタボリュームを指定することが可能になります。

```
GBAK ... -V 20000
```

バックアップかリストアを行うときに、このオプションを使用することで追加情報が表示されます。一度、gbak が（バックアップされたか、リストアされたか、インデックスが再構築されたかのいずれか）複数の列の定義を完了した場合、gbak は列の数を計算してメッセージを表示します。
デフォルトでは、カウンター数は 10,000 です。

曖昧な JOIN 文の拒否

InterBase では、このような命令に対してエラーを返しません:

```
SELECT A.FIELDA, B.FIELDA
FROM A JOIN B
ON FIELDX = FIELDY
WHERE FIELDA="99"
ORDER BY FIELDA
```

このような命令は、予測不能な出力結果を返します。SQL DIALECT3 の場合では、JOIN 文に定義されていないカラム識別子があるとエラーが返されます。SQL DIALECT1 の場合では、警告を返しますが、曖昧なクエリであれば実行されてしまいます。

言語の拡張

CURRENT_USER と CURRENT_ROLE

これらの新しい 2 つのコンテクスト変数は、USER と（もし準拠しているならば）現在の接続コンテクストの **ROLE** を参照します。

```
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 は、標準の SQL で使われている DSQL の **USER** とまったく同じです。

Firebird と同時に InterBase 4.x か 5.1 を使う場合は、ユーザーがロール名を渡しても **CURRENT_ROLE** は **NONE** となるので、ロールはサポートされません。InterBase 5.5、InterBase 6 あるいは Firebird を使用するときは、渡されたロールは確認されます。ロールが存在しない場合は、エラーを返さずに **NONE** にリセットされます。
NONE にリセットされるため、FireBird では **CURRENT_ROLE** に無効な **ROL**E を返しません。これは、SQL には認識されない場合でも、不正な値が内部に渡されてしまう InterBase とは対照的です。

ジェネレータの削除

未使用のジェネレータをデータベースから削除することができます。これによって、記憶領域を次のリストアで再使用するために解放できます。SQL と DSQL で利用可能です。

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

GROUP BY ユーザー定義関数

UDF の出力をグループ化することによって、SELECT 文をまとめることができます。

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

従来なら、**GROUP BY** 句で Firebird の内部関数を呼び出すことはできませんでしたが、ダミーの UDF ラッパーを作成することで可能になりました。

```
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)
```

RECREATE PROCEDURE

この新しい DDL 文によって、あらかじめ古いプロシージャを破棄することなく、同じ名前のストアドプロシージャを作成できます。構文は CREATE PROCEDURE と同じです。SQL と DSQL で利用可能です。

RECREATE TABLE

この新しい DDL 文によって、あらかじめ古いテーブルを破棄することなく、同じ名前のテーブルを作成できます。構文は CREATE TABLE と同じです。SQL と DSQL で利用可能です。
RECREATE TABLE では、古いテーブルのデータは保存されないことに注意してください。

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

先頭から *m* 列目までのデータを抽出します。SKIP を指定すると、*n+1* 列目から *m* 列

の出力を返します。もっとも単純な形式では、*m* と *n* が整数になります。Firebird で整数と評価されるすべての形式が有効です。SQL と DSQL に似ていますが、整数は GDML の中でも使用することができます。括弧は引数がないときには省略することができます。

さらに、**?**変数により ***n*** 列を除いた変数をバインドすることができます。例えば、**SKIP ?* FROM ATABLE** では、始めの ***n*** 列を除いた残りのデータを返します。**SELECT FIRST ? COLUMNA, COLUMNB FROM ATABLE** では、最初の ***m*** 列を返し、残りは除外されます。

FIRST はオプションです。SKIP は指定された列を除外するだけなので、FIRST がない場合でも正常に動作します。

上記のオプションを指定しない限り、SQL と DSQL で利用可能です。

```
SELECT SKIP (5+3*5) * FROM MYTABLE;
SELECT FIRST (4-2) SKIP ? * FROM MYTABLE;
SELECT FIRST 5 DISTINCT FIELD FROM MYTABLE;
```

SELECT FIRST の応用

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

これは、テーブルの中の列をすべて削除します。この SELECT 文は、指定されている 10 個の列をそれぞれ評価してから削除します。注意が必要です。

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

ANSI SQL の SUBSTRING()関数で実装されている内部関数です。**<*pos*>**バイト目から始まる文字列を返します。**FOR<*length*>**オプションが指定されている場合、**<*length*>** バイトの文字列を返します。

1 つ目の引数には任意の形式（文字列を評価する定数か識別子）を渡すことができます。**<*pos*>**は、整数されなければなりません。**<*pos*>**は、ほかの SQL コマンドと同様に 1 から始まります。**<*pos*>**も**<*length*>**も、クエリのパラメータにはなりません。

<*pos*>および**<*length*>**はバイトです。識別子は BLOB、あるいは、1 バイトのキャラクタセットを持つ **sub_type 1** のテキスト BLOB となります。この機能は、現在、漢字（2 バイト）と Unicode（3 バイト）のテキスト BLOB は扱えません。BLOB でない文字列については、すべてのキャラクタセットが対応しています。この関数は SQL と DSQL で利用可能です。

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

標準の UDF ライブラリの SUBSTRING 外部関数への追加については、このあとの「外部関数（UDF）」のセクションを参照してください。

大文字と小文字を区別しないハンガリー語キャラクタセット

大文字と小文字を区別しないハンガリー語キャラクタセットが追加されました。Sandor Szollosi 氏（**ssani@freemail.hu**）によって、テストと開発が行われました。

ISO8859-2 で追加されたキャラクタセット

Firebird では、ISO-8859-2 のキャラクタセット（チェコ語）をサポートしています。

新しいコメント記号

スクリプト、DSQL、ストアドプロシージャ、トリガーでは、次のコメント記号が使用できます。

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

この新しいコメント記号は、スクリプトや DDL/DML、ステートメント、ストアドプロシージャ、トリガーのコード中で 1 行を「コメントアウト」するために使用します。
ここでは、次のような仕組みで文字列を無視します。

1.改行コード（Linux/UNIX では LF、Windows では CRLF）の次の二文字が「--」であった場合無視する。
2.次の改行コードまで、間にある文字を無視する。

このコメント記号は、ブロックコメントロジック（**/* コメント */**）と混ぜて使用することは意図されていません。つまり、ブロックコメント内では「--」のコメントスタイルを使わず、「**--**」の行ではブロックスタイルのコメントを使ってはいけないということです。

INTERACTIVE ISQL SESSIONS :
対話型で isql セッションを使用する場合に、isql は、終端記号（通常「**;**」）を受け取るまで「**CON>**」プロンプトを表示し続けます。継続する行の始めに「--」と入力してしまうと、画面に表示されるか、Enter キーを押して出力ファイルの改行記号が現れるまで、入力を無視し続けます。「--」記号を使った場合は、エラーとなる可能性があります。
isql のみでしか認識されないコマンドがあるため、問題が生じることがあります。「--」が正常な配置にないため認識されない場合は、データベースエンジンに渡されます。データベースエンジン側では、isql の SET および SHOW コマンドが解釈できないので拒否します。

トリガーの変更ではテーブルのカウントはインクリメントされない

単一テーブル上のメタデータの変更の合計が 255 までに達すると、データベースは利用不可能となります。カウントをリセットし、データベースを使用できるようにするためにバックアップとリストアを使用します。これは、テーブルエンジンが機能しないほど多くの変更があった場合に、データベースのクリーンナップを実行することになります。
従来では、トリガーが、ALTER TRIGGER ステートメントによって、**ACTIVE|INACTIVE** に変更されると、関連するテーブルのカウントがインクリメントされました。しかし、カウントがインクリメントされるたびに負荷がかかるので、トリガーコードを破棄し、再利用する、という通常の操作に影響をおよぼしていました。

新しい予約語

以下に示す Firebird のキーワードを、InterBase 6.0.1 用に公表された予約語のリストに追加してください。アスタリスク（*****）が付いているものは、将来の使用に備えて予約されています。

ABS *	**BOTH ***	**BREAK**
CASE *	**CHAR_LENGTH ***	**CHARACTER_LENGTH ***
COALESCE *	**CURRENT_ROLE**	**CURRENT_USER**
DESCRIPTOR	**FIRST**	**IIF ***
LEADING *	**NULLIF ***	**OCTET_LENGTH ***
RECREATE	**SKIP**	**SUBSTRING**
TRIM *	**TRAILING ***	

次に示す InterBase 6.5 キーワード（Firebird でも予約されていない）についても、互換

性のために予約されていると考えておいてください。

PERCENT　　ROWS　　TIES

ISQL 機能

新しい PLANONLY オプション

PLANONLY オプションによって、ステートメントとクエリを実行せずに、ステートメントとクエリのデータベースエンジンへの登録とプランの検索ができるようになっています。これは isql のみが実行できる SET コマンドの 1 つです。

```
SET PLANONLY; /* toggles the state */
SET PLANONLY ON|OFF;
```

クエリは、データベースエンジンに投入されて準備されますが、実行はまだされません。つまり、IB_SQL と IBConsole の選択されたプランを表示する Prepare コマンドと同じです。

PLANONLY は、エラーを起こさずに、値を割り当てないパラメータ化されたステートメントのテストができます。

```
SQL> SET PLANONLY ON;
SQL> select first ? fld from tbl;
```

注意：標準では、PLAN と PLANONLY はオフになっています

- **SET PLANONLY**（あるいは **SET PLANONLY ON**）は、**PLANONLY** と **PLAN** の両方を利用可能にします
- **SET PLANONLY**（あるいは **SET PLANONLY OFF**）は、**PLANONLY** を利用不可能にしますが、**PLAN** には影響しません（**PLAN** はオンのままです）

SET 文は、**PLANONLY** の状態も含む現在の状態リストを返します

Isql で対話的に行を追加する ADD コマンド

対話的にテーブルに列（BLOD を含む）を挿入しようとするときは、新しい isql キーワードの ADD を使用できます。

```
SQL> ADD aTable ;
```

このコマンドは、**aTable** に新しいレコードを加え、同時にそのフィールドに対する値を保持します。これは、ENTER を押して、フィールドのプロンプトが返されたときに終了します。

BLOB であることがわかった場合、ファイルを編集するか読み込むかを選択することができます。

- EDIT はテンポラリファイルを開き、**BLOBVIEW** 用に定義されたエディタを呼び出します。ファイルを保存すると、新しく作成された BLOB が挿入されたレコードにバインドされます。
- EDIT 以外で文字列を入力すると、isql の add_row()機能はそれを開いてデータベースエンジンに展開します。

注意：addrow()ファンクションは、いくつかの欠陥があります。

- エントリで ENTER を押した後、エントリにラインフィードを追加します。したがって、列の最後のフィールドエントリの後に ENTER を押した場合、最後の ENTER に加えて Ctrl+Z で入力を終える必要があります。
- 不正な日付データの入力によってデータベースエンジンに無効な日付が入力されてしまう可能性があります（Firebird-Bugs-518343）。addrow()は、日付フィールドのために tm 構造を使用します。しかし、この構造は 1900 年を基としているため、年数が 1900 だけシフトして格納されることあります。

```
SQL> create table t(d timestamp);
SQL> add t;
Enter data or NULL for each column. RETURN to end.
Enter D as M/D/Y>1/1/2002
Enter D as M/D/Y>
Bad date
SQL> select * from t;
D
==========================
3902-01-01 00:00:00.0000
```

-x と-a オプションでダブルクオートが不要な場合

オプションの-x と-a のいずれかを備えたスクリプトのオブジェクト名を抽出するとき、isql は常に簡単なパスを取得し、識別子をダブルクォートで囲みます。これは、一部のユーザーには不要なものと考えられていました。現在では、ダブルクオートなしでは名前を表現できない場合のみ、使用されます。

次のものを含んでいる場合、識別子をダブルクォートで囲む必要はありません。

a) A-Z の ASCII 文字（大文字のみ）だけの場合
b) 0..9 の数字の場合
c) アンダースコア（_）と$が最初に位置し、かつ a)を満たしてている場合

データベースエンジンは、**BLANK** 識別子および長さゼロの識別子を識別することができません。理由は、システムテーブルが **varchar** ではなく **CHAR** を使用しているためです。したがって、余計な空白は無視されます。DSQL は、空白として定義されたフィールドと、計算式に起因する空白の出力フィールド識別子を区別することができません。混乱を防ぐために、空白と長さゼロの識別子は禁止されています。すぐに計算するためのカラム用に人為的な識別子を生成しないので、Firebird は、このような空白を SQL 文のインライン形式でのみ生成します。

理想的には、AS キーワードを使用し、明示的に出力フィールドを指定するべきです。省略するときには、空白（あるいは 0）識別子を作成し、ダブルクォートで囲まなければなりません。

API の拡張

isc_info_database の追加と変更

次の 3 つのアイテムが追加されました。

isc_info_db_provider：値は以下の通りです。

```
enum info_db_provider
{
isc_info_db_code_rdb_eln,
isc_info_db_code_rdb_vms,
isc_info_db_code_interbase,
isc_info_db_code_firebird
```

```
  };
```

isc_info_db_class：値は以下の通りです。

```
  enum info_db_class
  {
  isc_info_db_class_access = 1,
  ....
  isc_info_db_class_cache,
  isc_info_db_class_classic_access,
  isc_info_db_class_server_access
  };
```

isc_info_firebird_version：値は inf.c で設定されています。

次のアイテムが変更されました。

```
  isc_info_isc_version
```

古い名前は新しいものに再定義され、すべてが完全に動作しています。将来的には、**isc_info_version** を **isc_info_firebird_version** に変更することになります。
Inf.c は、新しいアイテムとして扱われるコードを持っています。
GDS_VERSION は、Firebird への反映を有効にするために、6.2 と 1.0 で同時に **license.h** で再定義しています。
utl.c での実装リストは、InterBase/xxx から Firebird/XXX へと切り替えられています。**common.h** と **ibase.h** での実装数は、ほとんどが **utl.c** の実装リストに変更されています。

isc_database_info セットの拡張

Funded by Jason Wharton, CPS (IB Objects)

API では、4 つのリクエストバッファアイテムが、**isc_database_info** アイテム構造体に追加されました。これは、初期化やサービス API の呼び出しをする必要がなく、静的なトランザクションを取り出すことを呼び出すことができます。

アイテム

```
  65 isc_info_oldest_transaction
  66 isc_info_oldest_active
  67 isc_info_oldest_snapshot
  68 isc_info_next_transaction
```

すべてのアイテムは整数を返します。isc_database_info()の呼び出し方法の例は API ガイドを参照してください。この例では、新しく追加されたアイテムも整数であるため、完全に適合していることがわかります。
次の例では、アイテム宣言を置き換えています。

```
  char db_items[] = {
  isc_info_oldest_transaction,
  isc_info_oldest_active,
  isc_info_oldest_snapshot,
  isc_info_next_transaction,
  isc_info_end};
```

この例として、switch()を持つループがあります。ループの内部の **case** を新しい値に変更しなければなりません。つまり､2 つの **case**+デフォルトの代わりに、4 つの **case**+デフォルトにしなければなりません。

外部関数（UDF）

標準ライブラリ、ib_udf.dll、ib_udf.so

SUBSTR(<*string expr*>, <*pos1*>, <*pos2*>)

<pos1>から<pos2>までの文字で構成される文字列を返します。**<pos2>**が文字列の終端を過ぎている場合、この関数は、**<pos1>**から文字列の終端までの文字をすべて返します。このような振る舞いは、**<pos2>**が文字列の終端を過ぎている場合に **NULL** を返す、UDF ライブラリの **ib_udf.dll** の Borland や Firebird の以前のバージョンの外部関数 **SUBSTR** と異なることに注意してください。

```
UPDATE ATABLE
SET COLUMNB = SUBSTR(COLUMNB, 4, 32765)
WHERE...
```

SUBSTRLEN(<*string expr*>, <*pos*>, <*length*>)

<pos>から始まる**<length>**バイトの文字列を返します。文字列の長さは、**<length>**より短いか、**<pos>**から入力された文字列の終端までの長さより短いはずです。

```
UPDATE ATABLE
SET COLUMNB = SUBSTRLEN(COLUMNB, 4, 99)
WHERE...
```

ascii_char()

Claudio Valderrama 氏は、**ib_udf.sql** で提供されている **ascii_char** の古いバグを解消しました。以下のような（誤った）InterBase 用の **CHAR(1)**ではなく、1 文字の C 用の文字を返すように修正されました。

```
DECLARE EXTERNAL FUNCTION ascii_char
INTEGER
RETURNS CSTRING(1) FREE_IT
ENTRY_POINT 'IB_UDF_ascii_char' MODULE_NAME 'ib_udf';
```

Firebird の新しい UDF ライブラリ（FBUDF.dll）

Development by Claudio Valderrama C., funded by Craig L. Leonardi

さらに多くの汎用性を提供するため、**BY DESCRIPTOR** 構文を使用できる Firebird UDF ライブラリ（**FBUDF.dll**）が、このリリースで配布されています。SQL 文にて宣言されている関数では、同じ機能にマップされている多くの宣言の違いに気を付けてください。たとえば、INULLIF()は INULLIF()と I64NULLIF()の両方にマップされていることが挙げられます。

作者の Claudio Valderrama 氏は、まだいくつかのバグがあるために、このライブラリはベータ版の段階であるとコメントし、これまでのところ、これは Windows 以外のプラットフォームではコンパイルされていません。バグ報告やコメントは歓迎されています。

宣言のソースコードと DDL は、Firebird の CVS ツリーにあります。それらを見つけるために、**http://sourceforge.net/projects/firebird/**から「Browse the CVS tree」「Browse CVS Repository」をクリックして、「interbase | extlib | fbudf | fbudf.*」を選んでください。

NVL()関数の「invl」および「snvl」文字列パラメータ

これらの機能は、カラムが **NULL** 値を持っている場合に実値を出力するような

Oracle の NVL 関数を模倣しています。それらは、1 番目の引数にはカラムが **NULL** かどうかを調べる式を渡します。2 番目の引数には、1 番目の引数が **NULL** の場合に出力する値を渡します。

NVL は 1 番目の引数が **NULL** ではない場合は 1 番目を返し、**NULL** の場合は、2 番目の引数を返します。両方が **NULL** の場合は、**NULL** を得ます。

パラメータのペアは、2 つの数値（**smallint、int、int64**）、あるいは 2 つの文字列値（**char、varchar、cstring**）の組み合わせにしてください。数字と文字列を混ぜた引数を渡すと、誤った結果を出力する可能性があります。Firebird の UDF によって実行される場合、パラメータタイプを受け取りません。

NULLIF()の「snullif」文字、「inullif」整数、「i64nullif」INIT64 パラメータ

NULLIF()は 2 つの引数を持ち、引数が等価であった場合に **NULL** を返します。引数が等価でない場合には、1 番目の引数の値を返します。**NULL** が UDF から返されないというデータベースエンジンの欠点を補うため、これらの 3 つの関数は、0 に相当する値を返します。この標準的はない振る舞いは、ほかの関数において **NULL** と扱われません。

注意：整数と int64 関数の関数呼び出し（inullif）は同じです。

日付（Day-of-Week）関数には短い文字列（**SDOW**）を返すものと、タイムスタンプの入力から長い文字列（**DOW**）を返すものがあります。戻された文字列はローカライズすることが可能です。

addDay、AddWeek などのいくつかの関数は、タイムスタンプに時間のステートメントを追加します。

RIGHT()関数（基本的には Rstring も同様）は、入力された文字列から、右端から n 文字を返します。

GetExactTimestamp()関数は、ミリ秒でシステムのタイムスタンプを返します。

Truncate()と i64truncate()は、32 ビットと 64 ビットの整数を任意の範囲（SQL DIALECT 1 では 9 桁まで、SQL DIALECT 3 では 19 桁まで）で丸めて、整数部分を返します。浮動小数や倍精度のデータ型では動作しません。

```
truncate(14.76) returns 14
round(14.22) returns 14
```

Round()と i64Round()は、32 ビットと 64 ビットの整数をそれぞれで四捨五入します。任意の範囲（SQL DIALECT 1 では 9 桁まで、SQL DIALECT 3 では 19 桁まで）で丸めて、自然数として近い値を返します。小数点以下の桁数を指定することはできません。

```
round(14.76) returns 15
round(14.22) returns 14
```

これらの関数は数学を丸めるために使用します。つまり、n.5 は常に大きい値のほうになります。これは、絶対値で大きい値に丸める Borland 社のツールの機能と異なります。

```
round(1.5) => 2
round(-1.5) => -1
```

String2blob()関数は、**CHAR** 型と **VARCHAR** 型を BLOB データ型に変換します。FreeUdfLib の関数に似ていますが、内部的に処理されるために非常にシンプルです。

新しい設定パラメータ

変更は、ibconfig や iscconfig といった設定ファイルに影響します

Linux 環境用に-NONAGLE スイッチの追加

一般的に TCP/IP Nagle アルゴリズムを使用しないことで、ネットワーク速度が改善されます。

Nagle TCP/IP アルゴリズムは、低速のネットワークで「tinygrams」と呼ばれる小さなパケットに関する問題を回避するために設計されました。このアルゴリズムでは、TCP/IP 接続は認識されていない極めて小さなセグメントを 1 つだけ持つと定義されています。細かい定義はいろいろありますが、通常、ethernet 上に約 1,500 バイト未満のセグメントサイズとして定義されます。

ソケットライブラリは、デフォルトで、書き込みデータのバッファを行う Nagle のアルゴリズムと呼ばれる内部アルゴリズムを使用します。これは、データを送る前に、実際の物理的な書き込みを最小限にするためです。

Linux 環境では、開発者自身がこの代替パケットを取り扱う方法を採用するかどうかを決めることができます。

isc_config/ibconfig の初期設定では、Nagle アルゴリズムを使用します。

```
#TCP_NO_NAGLE    0
```

使用しない場合は、この行をコメントアウトするか、以下のように変更します。

```
TCP_NO_NAGLE     1
```

CPU_AFFINITY 設定パラメータ

Windows の SuperServer 版 Firebird では、SMP マシンのプロセッサ間で、連続的にサーバプロセスを前後にスワッピングしてしまう問題が発生します。これが実行されてしまうとデータベースを破滅してしまう可能性があります。

これまで、ibserver は単一の CPU と結合性を持っていたため、アプリケーションとしてサーバを実行したり、プログラムが動作しているサーバでユーティリティ（**IB_Affinity.exe**）を実行することが必要でした。しかし、この新しい構成パラメーターは、SMP の Windows システムの CPU との結合性を変更するような外部プログラムの削除する項目を **ibconfig** に加えることができます。これは、サービスを開始することで有効になります。

CPU の結合性は、CPU のマスクを表す 1 つの整数として、CPU_AFFINITY パラメータに設定します。

```
CPU_AFFINITY 1
only runs on the first CPU (CPU 0)
CPU_AFFINITY 2
only runs on the second CPU (CPU 1)
CPU_AFFINITY 3
runs on both first and second CPU
```

注意：このパラメータは、NT API を利用しているため、Windows 9x には影響しません。Windows 9x では、マルチプロセッサの利点を得ることはできません

外部ファイル位置の定義

外部ファイル位置を指定する新しい設定は、プラットフォームごとに特有のものです。現在、これは Windows 用にのみ実装されています。以下のように動作します。

ibconfig ファイルで、Firebird が外部ファイルを探索しようとする（既存の）物理的なディレクトリを指定することができます。この環境では、ダブルクォートでパスを囲まなければならないことに注意してください。

```
EXTERNAL_FILE_DIRECTORY "H:¥test"
EXTERNAL_FILE_DIRECTORY "H:¥external"
```

探索するディレクトリ数に制限はありません。Firebird は探索リストとして、一連のエントリとして扱います。

```
isql> create table foo1 external file 'foo1.txt' (afield(char1), crlf char(2));
```

この例では、Firebird は **foo1.txt** の位置を確認するために、**h:¥test** と **h:¥external** の両方の中を検索します。この場合には、外部ファイルテーブルが作成され、正しいディレクトリ（**rdb$relations** の **rdb$external_file** フィールドは、ディレクトリ構成を含まず、単に **foo1.txt** をストアするだけ）にあるファイルが指示されます。

foo1 テーブルにアクセスされるとき、外部ファイルのディレクトリから自動的に抽出されます。

ファイルが存在しない場合は、それがどの位置であっても、現在のディレクトリ中に作成されます。この場合、すでに **EXTERNAL_FILE_DIRECTORY** を定義したので、ディレクトリ名は rdb$relations の外部ファイル定義から消去されます。パスを指定されていなければ、以前に定義された外部ファイルのディレクトリ中の既存のファイルを参照したものと仮定されます。この場合には、目的のファイルのあるディレクトリから isql が起動します。

EXTERNAL_FILE_DIRECTORY が **config** ファイルに定義されていない場合に実行すると、外部ファイルはそれぞれの **ISC_expand_filename** を参照して、フルパスの処理を行います。

```
isql> create table foo1 external file 'h:\files\foo1.txt' (
   afield(char1),
   crlf char(2));
```

つまり、外部ファイルのディレクトリ名を指定する必要があるということです。外部ファイルが指定した場所に存在しない場合や、通常ファイルとして定義されている場合は、指定されたディレクトリに通常のファイルとして作成されます。

インストールノート

Windows へのインストール

目次

インストールの前に

重要！

このインストールパッケージは、Firebird または InterBase の既存のバージョンがインストールされているかをチェックします。既存のバージョンが存在する場合には、インストールの前にサーバを止めて、現在インストールされているバージョンを削除しなければなりません。

サービスの停止

- サービスとして動作しているなら、［コントロールパネル］の［サービス］で止めてください
- アプリケーションであれば終了してください

既存のサーバの削除

Firebied または InterBase の以前のバージョンをアンインストールすることを推奨しますが、必須ではありません。Firebird のアンインストールの詳細については、以下の「アンインストール」セクションを参照してください。

InterBase がインストールされている場合

Firebird は、InterBase と同時に動かすことはできません。Firebird をインストールする前に InterBase をアンインストールしなければなりません。これは将来のリリースでは変更になる可能性があります。

既存の InterBase を上書きして Firebird をインストールしないでください。必ず新しいディレクトリにインストールしてください。

セキュリティデータベース（ユーザーとパスワード）

- セキュリティデータベースである **isc4.gdb** を gbak でバックアップします
- 新しい名前でリストアします
- Firebird サーバを止めます
- **Isc4.gdb** の名前を変更して、サーバをリスタートします

ibconfig で特殊な設定をしている場合、新しい **ibconfig** にコピーしてください。これは、サーバの再起動後に反映されます。

インストール条件

- サービスとしてインストールするには管理者権限が必要です。この条件は、Windows 95/98/Me には該当しません
- **GDS32.dll** を上書きするかどうかたずねられたとき、サーバとしてインストールを行っているならば、「はい」と答えてください
- 既存のバージョンの **MSVCRT.DLL** が存在する場合、アップデートの必要はありません。システムに存在しない場合のみ、インストールします
- インストールするディレクトリに、次の既存の設定ファイルが存在する場合、保存されます

```
isc4.gdb
interbase.log
ibconfig
```

アンインストール

Firebird のアンインストールルーチンは、次の主要なファイルを保存します。

```
isc4.gdb
interbase.log
ibconfig
```

インストールの際に追加されたファイル以外はアンインストールされません。

ほかのアプリケーションが使用してい場合、**gds32.dll** のような共有ファイルは削除されます。また、作成されたレジストリキーも削除されます。

そのほかの注意点

Winsock2

Firebird には、WinSock2 が必要です。Windows 95 を除く Win32 プラットフォームはすべてこのライブラリを持っています。インストール中に Winsock2 ライブラリの確認が行われ、見つからない場合にはインストールは失敗します。アップグレード方法は、次のページを参照してください。

http://support.microsoft.com/default.aspx?scid=kb;EN-US;q177719

Windows Me/XP

Windows Me および XP（Home Edition/Professional）では、**.gdb** 拡張子のすべてのファイルを自動的に更新する「システムの復元」と呼ばれる機能を搭載しています。この機能により、I/O 処理が起こるたびにファイルがバックアップされるので、データベースへのアクセスに遅延が生じます（Windows XP において、.NET サーバでの「システムの復元」はありません）。

Windows Me の Windows ディレクトリにあるファイル **filelist.xml** は、「保護されているファイルタイプ」が記述されています。**.gdb** はそこで指定されています。Charlie Caro 氏は、当初、このファイルの**[includes]**セクションから GDB を削除することを推奨していました。しかし、その後、Window Me は、このリストを再構築していることが判明しました。Windows XP では、いずれにしても **filelist.xml** を編集することはできません。

Windows Me においては、次のような代わりの方法があります。

- 主なデータベースファイルの拡張子として、FDB（Firebird DB）を使う
- 「システムの復元」からは除外される **C:¥My Documents** にデータベースを移動する
- 「システムの復元」のエントリをオフにする（指定するためには Windows のドキュメントを参照）

Windows XP（Home Edition/Professional）では、データベースを別のパーテーションに移動して、「システムの復元」からそのボリュームを除外することが可能です。

Windows XP はスマートコピーを使用しています。したがって、小さなファイルにおいては、Windows Me で見られるようなオーバーヘッドは、あまり問題にはなりません。大きなファイル（例えば、Firebird のデータベースはもちろん！）については、**.gdb** が一般的なシステムファイルに存在する限り、よりよい解決法は見つかっていません。

ユーザーがログインしているため書き込み可能となっている **isc4.gdb** セキュリティデータベースは、そのまま残されます。そのため、**isc4.gdb** ヘッダはトランザクションを更新します。したがって、ユーザーがログインするたびに、おそらく Windows Me はバックアップを作成します。

問題があった場合は、それを正しく提示し、その代わりとなる保証された手段をこの Web サイトに掲載したいと思っています。バグの指摘や代替手段の開発などで支援ができる場合は、Interbase のサポートリスト、あるいは、Firebird 開発ニュースグループ（news://news.atkin.com/）へメッセージを投稿してください。

Linux/UNIX へのインストール

（Mark O'Donohue）

Firebird サーバには、サービスとして動作する Classic 版と、バックグラウンドデーモンとして動作する SuperServer 版の 2 つの形式があります。将来的には SuperServer

を使うにしても、Firebird を使い始めたばかりのユーザーにとっては、Firebird の経験を積むために、Classic サーバを使用するのもよいでしょう。

注意：
1)Firebird をインストールするためには、root ユーザーである必要があります
2)SuperServer 版を正しくインストールするためには、**/etc/hosts.equiv** ファイルにローカルホストを追加する必要があります
3)リモートマシンからデータベースにアクセスするためには、**/etc/hosts.equiv** ファイルにリモートマシン名を追加する必要があります

Linux への Clasicc 版のインストールは以下のように行います。SuperServer 版のインストールは、`install` ファイルにある、SS タグが CS タグとなるだけです。

rpm を使った Linux へのインストール

```
$rpm -Uvh FirebirdCS-1.0.0-nnn.i386.rpm
```

.tar.gz ファイルでの Linux へのインストール

```
$tar -xzf FirebirdCS-1.0.0-nnn.tar.gz
$cd install
$./install.sh
```

もしくは、FirebirdSS-1.0.0-nnn

Linux のインストールで行うこと

Linux でのインストールは、以下のようになります。

1.現在、稼動しているサーバを停止します
2.以前にインストールした Firebird が存在する場合、**/usr/lib** と**/usr/include** に存在する関連ファイルは、**/opt/interbase_<datetimestamp>.tar.gz** としてアーカイブされ、削除されます
3.**/opt/interbase** ディレクトリにソフトウェアを、**/usr/lib** にライブラリを、**/usr/include** にヘッダファイルを、それぞれインストールします
4.3050 ポートとなっている **gds_db** を**/etc/services** に自動的に追加します
5.**localhost.localdomain** と **HOSTNAME** を、**/etc/host.equiv** に自動的に追加します
6.さらに、SuperServer 版の場合は**/etc/rc.d/init.d/firebird** にサーバスタートスクリプトをインストールします。新しい **rcfirebird** リンクは、**/usr/bin** の **init.d** スクリプトに作成されます。**/usr/bin** は、すべてのユーザーのサーチパスとなっているので、**Firebird initd** スクリプトを直接実行することができます
7.**/etc/rc.config** の Firebird エントリは、**SuSE** に作られます。

Firebird は、2、3、5 を自動的に行います。**.tar.gz** パッケージはアンインストールをサポートしていません。
すべてのインストールパッケージは、Red Hat 7.x、Madlake 7.x、SuSE 7.x 以降で開発・試験されています。
Classic 版では、**/etc/xinetd.d** ディレクトリが見つかった場合、自動的に **xinetd** エントリをインストールします。そうでない場合、**inetd** がインストールされます。いくつかのディストリビューションは、**/etc/xinetd.d** ディレクトリとは違う場所に **xinetd** があるので、手動でセットアップするときには、この設定をしなければなりません。

インストールの確認

以下のように、インストールが完了したか、ローカルからアクセスしてテストします。

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

>connect /opt/interbase/examples/employee.gdb;

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

>quit;
```

リモートアクセスによるテストは以下の通りです。

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

>connect '<hostname>:/opt/interbase/examples/employee.gdb';

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

>quit;
```

＊インストールによってパスワードが生成されたら、/opt/interbase/SYSDBA.file から取得してください。

Linux におけるアカウント

標準でインストールされるファイルに加えて、**/bin** ディレクトリに、以下のようなスクリプトがインストールされます。
（XX の 2 文字は、Firebird Classic 版の場合は SC に、Firebird SuperServer 版の場合には SS に読み替えてください）

XXchangeRunUser.sh	新しい Firebird unix ユーザーアカウントを作成します Firebird をインストールしたオーナーを変更し、Firebird ユーザーとして、バックグラウンドのタスクを起動します
XXrestoreRootRunUser.sh	Firebird のファイル自身をリストアし、バックグラウンドタスクのオーナーを、インストール時の root ユーザーにします

安全な Firebird のインストールのために、サーバプロセスは root として動作させないことが「強く」勧められています。
しかし、そうすることには、誰がどこに、最初の Firebird のデータベースを作成できるのかということに関して、いくつかの制限が存在します。

changeDBAPassword.sh	Firebird の SYSDBA ユーザーのパスワードを変更し、新しいパスワードと同時に/etc/rc.d/init.d/firebird の init スクリプトを変更しなければなりません

Solaris 2.7 Sparc への Firebird Classic 版と SuperServer 版のインストール

（Neil McCalden）

このリリースでは、以前と同じように 32 ビットファイルアクセスを使用します。つまり、2GB のファイルサイズ制限は依然として存在するということです。しかし、Web サイトのダウンロードページには 64 ビットの SuperServer 版があり、同様にバージョン 1.0 になっています。ただし、これは、テストおよびフィードバックが不十分なので、メインのリリースであるとはいえません。

基本的なインストール方法（Classic 版および Super Server 版）

root として、適当なディレクトリに**.tar** ボールを展開します。
次のようにリンクを作成します。

```
ln -s /ExtractDirPath/interbase /usr/interbase
ln -s /usr/interbase /opt/interbase

cd /usr/interbase
./install
```

これにより、ヘッダファイルとライブラリ用のリンクを作成し、**/etc/services** と **/etc/inetd.conf** ファイルをアップデートします。

Super Server 版の追加事項

/etc/hosts.equiv にローカルホストを追加し、**interbas** または **firebird** ユーザーとグループを作成します。サーバを開始するために**/etc/init.d|rc3.d** にスクリプトを作成します。
具体的な例は、**/usr/interbase/bin/firebird** を参照してください。
InterBase の初期バージョンがインストールされている場合、おそらく、**/opt/interbase** にインストールされています。パッケージを削除するか、適切な名前にリネームする必要があります。
サンプルファイルはこのリリースに含まれないことに注意してください。それらは **firebird.sourceforge.net** からダウンロードできます。さらに詳しい Firebird の情報に関しては、**README** ファイルあるいは **www.firebirdsql.org** を参照してください。

Mac OS X / Darwin への Firebird Classic のインストール

（John Bellardo）

（訳注：現在、Mac OS X 10.2 に対応し、GUI インストーラ形式の Firebird 1.0.1 がリリースされています。Mac OS X 10.2 以降は、Firebird 1.0.1 でないと動作しませんが、それ以前のバージョンの Mac OS X および Darwin では Firebird 1.0 を使うことが薦められています）

Firebird のインストールには、Mac OS X 10.1（Darwin カーネル 5.2）以降が必要です。**.tar.gz** アーカイブで配布されている Darwin 用の Firebird をダウンロードします。多くの Web ブラウザは、自動的に StuffIt Expander を呼び出して、アーカイブを展開するでしょう。Mac OS X ネイティブ版の StuffIt Expander を使わなければならないことに注意してください。もし、Classic 版を使った場合、インストール時に問題が発生します。
アーカイブを展開するために StuffIt Expander を使う代わりに、次のようにする方法もあります。

1.「ターミナル」を起動する
2.「**tar –zxf**」とタイプし、まだリターンは押さない
3.ファインダで、ダウンロードした **FirebirdCS-1.0.0-Beta2-Darwin.tar.gz** をド

ラッグし、ターミナルのウインドウにドロップする
4.ターミナルウインドウに戻り、リターンを押す

ソフトウェアをインストールするためには、管理者権限が必要です。もし、まだ管理者権限を持っていないなら、次のようにします。

1.［システム環境設定］→［ユーザ］
2.自分のユーザ名をダブルクリックする
3.［パスワード」タブを選択
4.左下にある鍵のアイコンをクリックし、管理者権限を持つユーザーにパスワードを入力してもらう
5.［ユーザがこのコンピュータを管理できるようにする］のチェックボックスをチェックする

管理者権限を取得したら、以下のように Firebird をインストールできます。

1.ターミナルを起動する(すでに開いているターミナルウインドウを使うことも可能です)
2.「**cd**」とタイプして、まだリターンは押さない
2.ファインダに戻り、ダウンロードしたファイルを展開してできた **firebird_install** フォルダを選び、これをターミナルウインドウにドラッグ＆ドロップする
5.ターミナルウインドウに戻り、リターンを押す
6.「**chmod a+x install**」とタイプし、リターンを押す
7.「**./install**」とタイプして、リターンを押す
8.「**no such process…**」というエラー表示がでるかもしれないが、これは無視しても構わない

インストールやそのほかのことで問題があった場合、Firebird は停止します。問題を解決するためには､技術サポートを提供する多くのオンラインフォーラムがあります｡特に、**http://groups.yahoo.com/**で運営されている「ibsupport」グループは注目です。

以上で、Firebird はインストールされ、起動準備が完了しました。
標準的な Firebird のコマンドラインユーティリィティは、**/Library/Frameworks/Firebird.framework/Resources/bin** にインストールされています。パスにこのディレクトリを加えたほうがよいでしょう。GUI ユーティリィティは、まだ完成していません。一般的な Interbase/Firebird のドキュメントは、すべて Mac OS X 版にも当てはまります。
興味を持つユーザーのために、インストールスクリプトがコンピュータにどのような変更を加えるのかを次に示しておきます。

1.**/Library/Frameworks** に Firebird のフレームワークをインストールする
2.System V のセマフォカーネルに CS が必要な拡張をインストールしロードする。この拡張は、コンピュータを再起動するときにはいつもロードされるように**/Library / StartupItems** にインストールされる
3.Firebird ユーザーをコンピュータに追加する。Firebird サーバが必要なときには、バックグラウンドプロセスは安全性を高めるために、このユーザーとして起動される。SS をインストールした場合、Firebird ユーザーは、すべてのデータベースに読み込み／書き込みの権限を持つ
4.**/etc/services** ファイルと **NetInfo** に **gds** エントリを追加する
5.次のように、ネットワークデータベースサーバをインストールする
   a. CS 版をインストールした場合、**inetd.conf** ファイルにエントリを作成する
   b. SS 版をインストールした場合、Firebird は、**/Library/StartupItems** に起動スクリプトを追加し、サーバを起動する

アンインストール

アンインストール用のスクリプトはありません。Firebird をアンインストールするためには、root として、以下のように作業します。

1.**/Library/Framework/Firebird.framework** と **/Library/StartupItems/FirebirdStartupItem**、**/Library/StartupItems/SysV Semaphores** を削除する
2.**/etc/inetd.conf** 中の Firebird エントリをコメントアウト（もしくは削除）し、HUP を送信（HUP の混乱を避けるために、再起動したほうがよい）
3.［システム環境設定］→［ユーザ］で、Firebird ユーザを削除する（オプション）
3.**/etc/services** と **NetInfo** から **gds_db** エントリを削除する（オプション）
4.コンピュータを再起動する。これにより、System V のセマフォカーネルをアンロードする。コンピュータを再起動しなければ、アンロードが有効にならない

John Bellardo
<bellardo@cs.ucsd.edu>

FreeBSD での Firebird のビルドとインストール

（Geoffrey Speicher, updated by Chris Knight）

FreeBSD 4.4 以前のシステムは、標準の暗号化ライブラリとして、DES 暗号化ライブラリをインストールする必要があります。次のようにチェックする方法が簡単です。

```
# ls -l /usr/lib/libcrypt.so lrwxr-xr-x 1 root wheel 14 Apr 24 2001
/usr/lib/libcrypt.so@ ->libdescrypt.so
```

->のあとに **libcrypt.so** があれば、デフォルトで MD5 暗号化ライブラリを使用しています。暗号ディストリビューションを追加するために、**sysinstall** を起動しなければなりません。これは、別のプログラムの問題を引き起こす可能性があるので、製品環境ではテストしないでください。そうでければ、暗号化ライブラリのインターフェイスが自動的に管理されているので、FreeBSD 4.4 以降に Firebird をインストールすることが推奨されます。
推奨される方法は、以下のように（root として）**ports** でビルドとインストールを行うことです。

```
# cd /usr/ports/databases/firebird
# make install
```

もう 1 つの方法は、以下のように、**pkg_add** で（root として）、ダウンロード可能な **package** をインストールすることです。

```
# pkg_add http://prdownloads.sourceforge.net/firebird/firebird-1.0_xx.tgz
```

xx の部分は、パッケージがビルドされている FreeBSD のバージョンです。

例）稼動しているのが、FreeBSD 4.5 の場合

```
# pkg_add http://prdownloads.sourceforge.net/firebird/firebird-1.0_45.tgz
```

Pure Java JCA-JDBC XA ドライバ

Firebird-Java チームは、Firebird 用の Pure Java ドライバを実装しました。このドライバは、エンタープライズの情報システムに接続するアプリケーションサーバ用の新しい JCA 標準と一般的な JDBC 標準の両方をサポートしています。
JCA 標準は、アプリケーションサーバがトランザクション、セキュリティ、資源のポーリングを管理するので、アプリケーションサーバはドライバに協調するアーキテクチャに

なり、仕様ではドライバは接続機能だけを提供します。JDBC 2 XADataSource のアイディアに似たようなところがありますが、JCA の仕様ではアプリケーションサーバとドライバ間の分担区分がかなり明確です。

実装されている機能

- 便利な JDBC 機能（開発者にとって「有用」）
- 完全な JCA SPI のサポート：Jboss のような JCA をサポートするアプリケーションサーバで直接利用される
- 管理された環境で JCA リソースアダプタとして利用されるときに、ツーフレーズコミットを行う XA トランザクション
- スタンドアローンで使うために、オプションの内部接続のポーリングを内蔵し、Tomcat 4 のような JCA 環境ではない場合での使用
- Tomcat 4 のようなトランザクションマネージャがない JNDI 環境で利用可能なオブジェクトファクトリの実装
- ポーリングの可否に関わらずデータソースを実装
- レガシーアプリケーションのために実装されているドライバ
- すべての Firebird のデータベースのパラメータブロックやトランザクションパラメータブロックの文字列への完全なアクセス
- オプションで統合された **log4j** を通じたログ取得
- データベース管理用の JMX mbean（データベースの **CREATE** と **DROP** だけに限らない）

インストール

すべての場合において、次のパッケージからの派生するクラスが利用可能です。

concurrent.jar
connector.jar
jaas.jar
jta-spec1_0_1.jar
log4j-core.jar
さらに JDBC 2/3 クラスの両方

Jmx エージェント（MbeanServer）の中、あるいはスタンドアローンのクラスとして、jmx 管理 mbean を利用可能です。Jmxri でのみのテストを行った限りでは、任意の jmx 実装系で特に問題はありませんでした。

JCA リソースアダプタとして管理された環境で使用するために、環境デプロイメントメカニズムによって、firebirdsql.rar を展開してください。たとえば、JBoss 3 を使用する場合には、展開するディレクトリにファイルを置いてください。アプリケーションサーバの必要条件によって、**ConnectionFactories** を設定する必要があります。JBoss 3 用の設定例は JBoss のオンラインマニュアルで参照できます。

JCA デプロイメントをサポートしていない環境では、アプリケーションで利用可能な **firebirdsql.jar** でクラスを作成してください。

JCA サポートまたは JNDI を備えた環境では、JNDI 中のデータソースへの参照をバインドするために **FBDataSourceObjectFactory** を利用します。Tomcat 4 は、このような場合のよい例です。JNDI の実装は、参照/参照可能の使用をサポートしていなければなりません。JNDI の実装がシリアライズされたオブジェクトのバインドのみをサポートする場合には、動作しません。

ポーリング接続が有効な複数の接続が必要なスタンドアローンのアプリケーションで使うためには、ポーリング用に設定された **FBWrappingDataSource** のインスタンスを使用します。

1 つの接続のみが必要なスタンドアローンのアプリケーションで使うためには、**FBWrappingDataSource** あるいは FBDriver のいずれかを使用します。

David Jencks, **d_jencks@users.sourceforge.net**

サポートの質問は、Firebird-Java 電子メールフォーラム（news://news.atkin.com/にミラーされています）で聞くのが一番の方法でしょう。

Firebird-Java@yahoogroups.com

InterClient JDBC ドライバ

Firebird チームは、Borland 社によってリリースされていたオリジナルの「Interclient 2」ドライバにあったいくつかのバグを修正しました。Firebird では、Windows と Linux 用のインストールパッケージを作成しました。これらは、ダウンロードエリアから利用可能です。

みなさんのニーズを満たすかどうかを確認するために、Pure Java ドライバ（上記参照）を試してみることを願っています。**Firebird-Java@yahoogroups.com** のサポート質問に、サポートおよび開発リストの質問をしてください。投稿する場合、InterClient に関しての質問であると明確に記述してください。

追加情報

さらなる情報は、以下の Firebird データベースエンジンフォーラムで参照することができます。

http://firebird.sourceforge.net/

それ以外の関連するサイトは次の通りです。

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

Firebird の開発に興味を持ったり、報告すべきバグを発見した場合には、Firebird 開発者リストに参加してください。登録するためには、単に次のアドレスに、タイトルを「subscribe」とした空の電子メールを送るだけです。

firebird-devel-request@lists.sourceforge.net

サポートの質問を投稿するために、Firebird 開発者リストを使わないようにしてください。テクニカルサポートが必要な場合、以下の「ib-support」リストに参加してください。

http://www.yahoogroups.com/groups/ib-support

InterClient と Java の開発やサポートのためには、以下のような特別なリストがあります。

http://www.yahoogroups.com/groups/Firebird-Java

ib-support リストでは、Firebird と InterBase の技術的な問題を扱います。Delphi やほかのクライアントの開発環境の質問は、関連するフォーラムでしてください。
オープンソースコミュニティは、Firebird の開発のさまざまな側面について、そのほかのディスカッションリストを運営しています。詳細に関しては、Firebird コミュニティサイトのメーリングリストとニュースグループを参照してください。
Firebird 開発者のリストおよび一般的なコミュニティリストは、開発者が興味を持つほ

かのいくつかのリストに加えて、次のところで、ニュースグループとしてミラーされています。

news://news.atkin.com/

Firebird に関して迅速なサポートを受けるには、IBPhoenix 社（連絡先と電話番号は http://www.ibphoenix.com/に記載されています）と契約するべきでしょう。Firebird チームの何人かのメンバーは、サポートとコンサルタントをすることも可能です。彼らに直接コンタクトしてください。

Firebird または InterBase のデータベース復旧サービスは、IBPhoenix 社が取り扱っています。

Firebird へのさらなる援助の要請・提供は、firebird-contact@lists.sourceforge.net 宛ての電子メールで、直接 Firebird チームに連絡を取ってください。

Firebird の拡張についての一般的な議論は、Firebird プライオリティリスト（http://www.yahoogroups.com/community/Firebird-priorities/）で扱われています。

IB-Architect（http://www.yahoogroups.com/community/ib-architect）は、技術的な設計の議論だけです。サポートなどの質問は、ほかの適切なところで行ってください。

ドキュメント

InterBase 6 用のドキュメントは、現在の Firebird のリリースにも当てはまります。

InterBase 6 マニュアルのベータ版は、次のサイトから、PDF で入手できます。

http://www.ibphoenix.com/downloads/60All.zip

構造化されたドキュメントのインデックスは、次の Firebird コミュニティサイト上でメンテナンスされています。

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

これは作業中であり、どのような追加でも歓迎されます。次のアドレスにメッセージを送ってください。

firebird-docs@lists.sourceforge.net

いくつかのインストールガイドラインとそのほかのガイドは、次のリンクのドキュメントエリアで見つけられるでしょう。

http://www.firebirdsql.org/

もしくは、次のサイトも参照してください。

http://sourceforge.net/projects/firebird/

ユーザーと技術的な問題に対する主なリポジトリは、以下の IBPhoenix 社のサイトです。

http://www.ibphoenix.com/

いくつかの追加ドキュメントは、以下の Borland 社の Techpubs エリアで発見できるかもしれません。

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

リリース以降のバグフィックス

SFID 492181　JDBC における BLOB のパフォーマンス問題

BLOBへのJDBCアクセスが非常に遅い(50KB/秒程度)ことが報告されています。BLOBバッファサイズ（**org.firebirdsql.jdbc.FBBlob.bufferlength**）を増やすことにより、この問題を解決できるかもしれません。例えば、2KB から 20KB に増加させるには、9 の倍数だけスループットが増やします。

修正済み

SFID 527669　Linux クライアントライブラリがスレッド接続できない

Linux クライアントライブラリ（**gdslib.so**）が各々のアプリケーションスレッドにおいて、open/use データベース接続の開始や使用する機能がサポートされていませんでした。リリース 1 で解決されました。

注：gdslib.so は依然としてスレッドセーフではありません。

SFID 217138.　複雑なビューを含む JOIN がサーバを停止させる

結合に使われるフィールドのインデックスを持つテーブルを含む複雑なビューの結合する場合、データベースエンジンがクラッシュします。

修正済み

SFID 216733　多すぎるジェネレータはデータベースを破壊する

InterBase が持つことができるジェネレータの数は、ページサイズ（オーバーヘッドは未知）とジェネレータのサイズに依存していました。この制限を越えたジェネレータも問題なく作成できますが、これらのジェネレータはランダムなデータを返し、インクリメントされればデータベースを破壊します。

修正済み

SFID 225283　ビューの ORDER BY がフィールドに NULL を作成する

この問題は、Claudio Valderrama 氏によって、リリース 1.0 で修正されました。

SFID 228135　単純なビューの left join の実行時に NULL ではなく 0 が返される

この問題は、Claudio Valderrama 氏によって、リリース 1.0 で修正されました。

SFID 518279　Left join が文字列が切り詰められたというエラーを返す

この問題は、Claudio Valderrama 氏によって、リリース 1.0 で修正されました。

SFID 514973　ソートファイル・テンポラリファイルが 2GB 以上になる問題

この問題は、Mike Grover 氏によって、リリース 1.0 で修正されました。

SFID 不明　Linux で gdslib.so クライアントがスレッドセーフではない

この問題は、リリース 1.0 で修正されました。

SFID 518273　SQL DIALECT 3 で数字を負の数で割るとデータが破壊される

2002 年 2 月に Guido Klapperich 氏によって発見された「Klapperich バグ」は、InterBase 6 のソースコードから継承されていました。Claudio Valderrama 氏によって Firebird 1.0 で修正が確認されました。

SFID 213462　Windows 上での「パスストリングバグ」の修正

InterBase は、InterBase 5 で確認されたものの InterBase 6 ベータで無視されたやっかいなバグを持っていました。これは、とても直しにくい方法でデータベースを破壊します。Windows サーバへの接続文字列の正確なパスは以下の通りです。

```
C:¥patha¥mydatabase.gdb
```

しかし、Windows は、次のような記述も可能です。

```
C:patha¥mydatabase.gdb
```

2 人のユーザーが接続するとき、一方が前者の構文を使用し、他方が後者の構文を使用した場合、サーバは、Windows から誤ったメッセージを受け取り、2 人のユーザーが 2 つの異なったデータベースへ接続していると判断します。ユーザーのトランザクションはお互いが動作していることに気づかず、その結果、データはひどく破壊されてしまいます。
修正では、最初のユーザーの接続によりそのユーザーが接続したパスの文字列に従ってデータベースファイルに排他的なアクセス制限を確立します。
別のパスの文字列で接続を試みるほかのユーザーには、次のようなエラーが表示されます。

```
I/O error for file "C:patha¥mydatabase.gdb"
Error while trying to open file
The process cannot access the file because it is being
used by another process.
```

この修正によって生じた RC1 での例外と、データベースを強制的に書き込みに設定してしまう機能は修正されました。

SFID 227760　長さゼロの識別子が禁止されている

クライアントアプリケーションにデータを伝えるために使われていた **XSQLVAR** 構造は、ident(NULL)および空白の ident を識別できません。したがって、空白のフィールド名（長さゼロの名前）は禁止されていました。また、「""」（空白のみ）のような名前も、長さゼロの識別子なるので禁止されていました。同様に、**CREATE** および **ALTER** 構文でも、すべて異なるオブジェクトタイプとして扱われます。さらに、SQL DIALECT 3 の識別子で、データベースエンジンが先頭にある空白を扱えないという問題も修正されました。
この問題は Jason Wharton 氏によって修正されました。

SFID 428889　カラムの位置が 1 ではなく、0 を基準として扱っていた

```
alter table...alter column...position <n>;
```

この構文は、Firebird では 1 を基準としています。内部的には、エンジンは C の慣習を用いて、0 から始まっているという事実にも関わらず、論理的なレベルでは、**ORDER BY <n>** と同じ方法で扱われていました。
したがって、次のコマンドは、最初の位置にいくつかのフィールドを設置します。

```
alter table tbl
alter column cln position 1;
```

現在では、Firebird は、標準および InterBase 6 ドキュメントの両方に一致するように修正されたため、非互換性が存在しています。EmbedSQL.pdf の第五章から引用すると、以下のような記述があります。

> ...
> ALTER TABLE ALTER コマンドは、カラムの位置と名前を同時に変更することが可能である。例えば、以下の命令は、EMPLOYEE テーブルで、EMP_NO のカラムを 2 番目から 3 番目に移動する。
>
> ```
> ALTER TABLE EMPLOYEE ALTER EMP_NO POSITION 2;
> ```
>
> ...

この例は、手動で行えば Firebird で動作します。ゼロを基準としているかのように、カラムの位置が扱われるバグを依然として持つ InterBase 6 では、2 番目が InterBase 6 における 3 番目の位置にあるため、何も起こりません。

SFID 228526　JOIN 構文の曖昧さが解消された

InterBase は、次のような構文にはエラーを返しません。

```
SELECT A.FIELDA, B.FIELDA
FROM A JOIN B
ON FIELDX = FIELDY
WHERE FIELDA="99"
ORDER BY FIELDA
```

このような命令文は予測不能な出力結果を返します。現在、SQL DIALECT 3 の Firebird では、結合命令文の中に何らかの絶対的なカラム識別子がある場合、エラーを返します。SQL DIALECT 1 では警告を返しますが、曖昧なクエリのまま、処理を進めてしまいます。

SFID 223133　奇妙な結果を生じさせる join 構文自体の不具合

修正済み

SFID 460261　BLOB の API における空白を含む名前の問題

次の API の呼び出しは、空白を埋め込んだ SQL DIALECT 3 の名前が存在した場合、エラーを返します。

```
isc_blob_default_desc
isc_blob_lookup_desc
isc_blob_set_desc
```

原因となった、**blob.e** の **get_name()**関数は修正されました。

SFID 436462　ビューの BEFORE UPDATE トリガーで、列が不正な影響を受ける

列の影響を受けた値は、物理的な操作ではなく仮想的な操作として報告されます。これは、更新不可能なビューでは問題なく動作しますが、トリガーを使って更新しているビューでは、物理的な操作を行うごとにカウントがインクリメントされてしまうことに注意してください。

SFID 444463　BEFORE トリガーが CHECK の後に実行される

BEFORE トリガー（挿入または更新）は、CHECK 制約の後に実行されます。この CHECK 制約は、テーブルの CHECK 制約が失敗するような値にフィールドを変更することが可能です。これは間違いであり、データは復元可能となります。

SFID 229009　CREATE VIEW が構文エラーを返さない

ビューカラムの数が、選択したステートメントでのカラムの数に相当しない場合でも、InterBase 6 は CREATE VIEW コマンドを許可してします。SQL92 規格によれば、このようなステートメントは構文エラーを返すべきです。現在では修正されています。

SFID 458888　REFERENCES 特権が制限され、クラッシュを引き起こす

依存関係にあるテーブルが参照されているテーブルについてのパーミッションを得るために、外部キートリガー用に明示的に宣言されている REFERENCES 特権を要求したというバグが報告されました。これは、内部トリガーを持つテーブルが、別のテーブルに暗黙の REFERENCES 特権を得るように修正されました。REFERENCES 特権をチェックするコードが、コメントアウトされ、このような条件が発生しないようにトリガーのアクセスリクエストの反応は柔軟なものに変更されました。

このバグを分析することで、さらに 2 つのバグが発見され、それらも修正されました。

- 「設計時」（DDL）にチェックされる REFERENCES 特権は、バイパスする対象となりました。現在では、外部キーの作成者は、マスターテーブルに REFERENCES 特権を持っているか、あるいは、マスターテーブルの所有者であることが義務となっています。

- 最初のバグの修正は、議論はされていたものの記録されていなかったバグも修正しました。このバグは、任意のテーブルから選択することで、ただちにクラッシュを引き起こすような複雑なデータベースをどこにバックアップし、リストアすべきなのかという、議論のきっかけともなりました。

SFID 446237　'Column not found'エラーが起こるべきでないところで起こる

修正されました。空白のフィールドを持つ SQL DAILECT 3 のテーブルがあるビューで、この問題は発生していました。これは、埋め込まれた空白に関するいくつかのバグのうちの 1 つです。

SFID 229860　誤ったエラーメッセージ

「DATE data type is now called TIMESTAMP」というエラーメッセージは、タイプミスのような無関係のエラーのときに返されます。

修正済み

SFID 460624　SQL DIALECT 3 で、isql のプロシージャパラメータを抽出するとデータを破壊する

名前にはダブルクォートが必要ですが、それにも関わらず、SQL DIALECT 3 では、プロシージャのパラメータを正確に抽出しないという isql のバグを修正しました。ダブルクォートを行うルーチンでの正しくないパラメータ宣言も修正しました。

SFID 451798　FIRST が集約の前に適用される
SFID 451810　SKIP が 1 によってオフになる

ともに修正済み

SFID 412417　CHAR から VARCHAR への変更時のエラー

CHAR を **VARCHAR** カラムに変更することは、フィールド長に 2 バイトを加えることでした。さらに、**CHAR** をより小さいサイズの **VARCHAR** へ変更することも可能です。損傷したテーブルから選択した場合、変換された文字列がトランザクションエラーメッセージの原因となります。

修正済み

SFID 231998　ダイアレクト 1 において、CAST された数式の前の空白

修正済み

SFID 212177　英語以外のカラムがデフォルトのときのエラー

デフォルトで英語ではないキャラクタセットを使っているデータベースに格納されたデータを復元すると、**GBAK** はトランザクションエラーを起こしてしまいます。

SFID 221589　数値のフィールドと数学処理

```
select field1 * field2 from mytable
```

あるいは

```
select field1 * (1+field2/100) from mytable
```

両方のフィールドが数値のとき、(9,2)は正しくない結果を生成します。

SFID 450301　SUBSTRING が動作しない

Where 節（例えば、with、in、=)、あるいは **stringconcatenation** の中で使用されたとき、新しい機能が動作していませんでした。

修正済み

SFID 223059　VARCHAR の更新時に古いデータをクリアしない

InterBase 6 が **VARCHAR** に格納された文字列を更新する場合、残される文字列の長さをゼロにせず、更新した値に古い値を連結します。**VARCHAR** が文字列の長さを含んでいるので、クライアントアプリケーションはいかなる問題（つまり、常に正確な結果を参照しているかどうか）も判別できません。しかし、**gdb** ファイルは予想するよりも速く増大します（追加データは rle 圧縮されないため)。また、データベースは遅くなってしまいます。

Firebird で修正済み

SFID 223512　テーブルが DROP VIEW によって破壊される

テーブルとビューが同じ名前空間を共有するということを意味しています。しかし、その名前のビューがある場合のみ、**DROP VIEW** は動作するべきです。同じ名前のテーブルがあっても、コマンドはあたかもそれが存在しないかのように動作するべきです。

修正済み

SFID 226456.- SELECT / PLAN は SQL インデックス名で区切られていると解釈されない

修正済み

SFID 419964　remote / interface.c ライブラリにおけるバッファオーバーフロー

これは、長さがわずか 64 バイトのバージョンストリングバッファーによって引き起こされました。現在、このバッファ容量は増やされています。

SFID 453686　Firebird 1 βで SQL DIALECT 3 データベースを作成できない

この問題は、5 以降のメジャーバージョン番号に接続する必要のあるクライアント（例えば IBConsole や IBExpert)を使用するときに生じていました。Firebird RC 1 で修正されました。

SFID 233124　不正なコードの実行中に接続が失われる

修正済み

SFID 425799　名前を変更されたドメインに前の次元が残る

配列であるドメインで、ドメイン名を変更しても、rdb$field_dimensions は変更されません。したがって、ドメインとその次元指定の関係は崩れてしまいます。

修正済み

SFID 450405　複雑な ROLL は基本的な SQL セキュリティに失敗する

これは複雑な問題でしたが、現在では、この血管は修正されています。

SFID 462800　RDB$FORMATS に一意でないペアが存在する

修正済み

SFID 447377　GDS エラー（TIP を見つけることができない）

InterBase 5.6/6.01 にはバグがあります。初期の Firebird β は、トランザクションページが 32,767 以上ある場合、トランザクションインベントリページの検査に失敗しまし

た。データベース用に、次のような最大の安全なトランザクション ID を作成します。

```
1024 byte pages 131,596,287.
2048 byte pages 265,814,016.
4096 byte pages 534,249,472.
8192 byte pages 1,071,120,384.
```

非常に大きな数字ですが、6 か月で 1 億 3,100 万のトランザクションを越えるような特殊なデータベースもあります。このような大量のトランザクションが発生するデータベースに接続するとき、TIP が見つからずに GDS の内部エラーが発生します。

提案：
1) 1,024B のページサイズを使わない
2) 次のトランザクション番号をときどきチェックする
3) 次のトランザクション番号がリミットに近づいていることが確認できた場合、データベースをバックアップ／リストアする

SFID 229231　REVOKE でのユーザー名の大文字小文字

```
grant all on config to "admin";
revoke all on config from "admin";
```

上の例では、**config** 上に特権を持っていない admin という結果になると思われるでしょう。しかし、**RDB$USER_PRIVILEGE** テーブル上での選択は、「admin」ユーザーなら、依然としてフルアクセス可能であることを示しているのです！　特権の削除を行うために、次のように実行しなければなりません。

```
revoke all on config from "ADMIN"
```

さらに、InterBase 4 以降に存在する曖昧さをなくしました。

```
grant update on tbl to <unknown>
```

InterBase では、「unknown」がロールであるかユーザーであるかがわかりません。ロールが存在する場合、ロールには特権が与えられます。ロールがない場合でも、ユーザーには特権が与えられてしまいます。**USER** キーワードを指定しても、InterBase では何もしません。Firebird では、**GRANT** が作り出すのは、ロールであるのか、ユーザーであるのかを指定することができますが、上記のステートメントは依然として曖昧です。

- **ROLE** が存在していたとしても、ユーザー「unknown」にユーザー特権が与えられます
- **ROLE** が存在しない場合はエラーを返し、「unknown」**ROLE** には ROLE 特権だけが与えられます

これは、依然として SYSDBA にロールが存在していない **GRANT** を許してしまうという、修正されていないバグ（SFID 223128）です。
Ivan Prenosil 氏によって報告された、isql が「**grant ... TO GROUP UNIX_group**」構文でコピーすることができなかった問題も修正されました。

SFID 421260　文字長が UDF を満たしていない

ODS 10 の場合、スタブとしてのみ書き込まれます。
修正済み

SFID 227375　NULL データ上で派生したフィールドをグループ化すると InterBase が停止する

GROPU BY 句によって派生されたカラムから引き算を行う計算フィールドをビュー内

の SELECT 句で設定すると、InterBase が停止するバグが発生しました。現在、このバグは修正され、Firebird では適切に動作します。

SFID 425949　データベースエンジンをクラッシュさせるエラー

```
select count(*),adresy.rdb$db_key from adresy
```

このような記述は、InterBase をクラッシュさせます。**adresy** は任意のテーブルです。修正パッチを Borland のツリーから Firebirdhe にコピーしてください。

SFID 228467　isc4.gdb で全権限を持つハードコードされたユーザーによるセキュリティバグ

これは、InterBase 4.x 以前のすべてのバージョンに影響すると発見されたセキュリティ上の脆弱性でした。Firebird で修正され、その後、2001 年 1 月以後の InterBase のソースコードとバイナリも Borland によって修正されました。

SFID 229121　TEMP ディレクトリがいっぱいになる

修正済み

SFID 213708　宣言されたカーソルが依然として残っていた

ローカルで AIX の上の InterBase 6 の Classic 版に接続している場合、および、Windows NT 上の SuperServer にリモートで接続している場合に、Microfocus に問題が生じました。

修正済み

SFID 214298　テーブルが空のとき、select count(*)式が不正な動作をする

テーブルが列を持たないとき、**select count(*) + 1** はゼロを返します。ただし、**count(*)**が 0 以上の場合には正しい結果を返します。

修正済み

SFID 216579　COMPUTED BY のコラムでジェネレータを使った場合、不正な結果を返す

COMPUTED BY でジェネレータを使った場合、不正な結果を返し、データベースを使用不可能にします。

修正済み

SFID 222476　AVG と SUM が、SQL DIALECT 3 で空のフィールド名を返す

修正済み。同様の問題が **CAST** と **UPPER** においても修正されました。

SFID 227717　COBOL プログラムが、ランダムに 901 request sync. Error を返す

COBOL プログラムが、ランダムに**-901 request synchronization error** を返します。COBOL プログラムが **UPDATE** 中に SQL 文の値を失ってしまいます。

```
UPDATE SET ... WHERE x=..
```

更新によって返される「**Mass Update**」エラーコードは、プログラムに返されず、代わりに、プログラムには**-901 request synchronization error** が返されます。

このエラーは、GPRE によって生成される不正なコードによって引き起こされていました。

修正済み

SFID 212328　IB Guardian がハンドルをリークしていた

修正されました。しかし、シャットダウンするか、トレイのアイコンをダブルクリックするまで、Firebird がクラッシュするたびに、さらに多くのスレッドを作成しています。サービスとして動作しているときには、トレイにアイコンを置かないので、このバグは、アプリケーションとして動作している Guardian だけに起こります。

SFID 212263　コマンドラインの isql が、–a もしくは-x と同時に指定した場合に-user と-password を無視していた

修正済み

SFID 421262　ISQL reports UDF BLOB parameter BY VALUE

Firebird の公開で、Ibudf 中の BLOB の UDF が修正されました。

SFID 222563　ISQL が UNICODE の不正なストアドプロシージャのパラメータを抽出する

ISQL は、プロシージャのパラメータとして **rdb$character_length** の代わりに **rdb$field_length** を読み出していました。テーブルのフィールドの場合には、正確な情報が読み出されて表示されます。データベースエンジン自体は正しい処理を行っていました。

修正済み

SFID 223516　rdb$types で見当たらないタイプがある

追加されましたが、すべての提案されたタイプが追加されたわけではありません。

SFID 451944　トリガのアクティブ化/非アクティブ化がメタカウンタを増加する

修正済み

既知の問題

未解決の問題

外部キーインデックスが使用不能

ALTER INDEX を使って FOREIGN KEY インデックスを不活発化を可能にする拡張は、低選択性のインデックスを維持することに関連するいくつかのパフォーマンスの問題を解決するために、関連項目の未解決問題として延期されました。

SFID 446227　Linux 上で、gdslib.so クライアントライブラリがスレッドセーフではない

スレッド可能な gdslib.so が 1.0 リリースに含まれていたが、このスレッドセーフ問題は解決されていない

コアエンジン

SFID
479483 サブセレクトにおける FIRST/SKIP の誤った処理
224810 DISTINCT が VIEW の外部に伝搬する
442140 VIEW が動作しないロールを許可する
222376 多くの OR 条件がある大変なプラン
223514 InterBase は 2 つのプロシージャが内部結合されている場合にクラッシュする
217042 InterBase が誤った構文の妥当性を検査しない
419065 異なるデータタイプで結合できる
223058 マルチホップサーバの機能は、確認済みのバグで破壊される

219525 フェッチ処理のカレントレコードがない
221921 ORDER BY が効かない
213460 イベント w/特定の設定で Registering Events が IB サーバをクラッシュさせる
233025 2 回以上のストアドプロシージャを実行したとき、サーバが停止する
221649 NULLABLE フィールドで、一意なインデックスが許可される
211781 Win32: サーバがスレッドハンドルをクローズできない
233644 UPDATE ステートメントにおけるプランの特定が不可能

DSQL

GBAK
SFID
228431 gbak が InterBase5 によるバックアップをリストアできない

GPRE
SFID
416228 gpre が無効な isc_vtov 呼び出しを生成する
223513 isql の SHOW コマンド中のテーブルおよびビュー間の両義性
223126 isql を備えたメタデータを抽出する場合の誤った照合
225219 isql –a：テーブルのフィールドに基づいたドメインにおける誤ったオーダー
450404 コマンドラインでの isql の大文字のロール

InterClient
SFID
227414 ときどき Interbase/Interserver が接続を許可しなくなる

セキュリティ問題
SFID
229237 空白パスワードの不完全なサポート
229894 クライアントプログラムが任意のユーザーとしてログオン可能
222375 Grant 文が従来の **rdb$security_classes** エントリに上書きする