Firebird 2.5 リリースノート

Helen Borrie （校正／編集）

2015年9月7日 — ドキュメント・バージョン0254_01-ja — Firebird 2.5.4 用

Firebird 2.5 リリースノート

2015年9月7日 — ドキュメント・バージョン0254_01-ja — Firebird 2.5.4 用
Helen Borrie （校正／編集）
Translation into Japanese: Tsutomu Hayashi, Yoshiyuki Iwasaki, Yasuro Tanigawa
Sponsored for translation into Japanese: Shinya Nakagawa

Table of Contents

List of Tables

Chapter 1

一般的注記事項

Firebird 2.5.4サブリリース

Firebird 2.5のこのサブリリースでは、一件の有用な新機能と、内部BLOB管理の改善が追加されています：

・ このサブリリースから、オンライン上にあるデータベースのテーブルとインデックスの検証を実行できるようになりました。

詳細は、Firebird APIとODSの変更の章のオンラインでの検証サービスの実行の項を参照してください。

・ (CORE-4671) ：メモリとディスク領域の解放のため、内部BLOBのリリースが早められました。

Vlad Khorsunが実装しました。

2.5.3サブリリース以降のバグフィックス集はこちら。

Firebird 2.5.3セキュリティアップデート1

スーパーサーバーとスーパークラシックサーバーには公開された脆弱性があり、不正な形式のネットワーク・パケットによってセグメンテーション違反を起こし、クラッシュする可能性がありました。クラシックサーバーではこの問題はありませんでした。

この脆弱性はAlex Peshkovによって修正されました。ビルド番号26778以下の全てのFirebirdスーパーサーバーとスパークラシックサーバーのバイナリ、また、2014年12月3日より前の全てのスーバーサーバーとスーパークラシックサーバーのスナップショットにはこの脆弱性が含まれます。

Firebird 2.5.3サブリリース

Firebird 2.5.1で作成またはリストアされたデータベースについての警告

Firebird 2.5.1から上位のサブリリースへアップグレードする場合、データベース移行の際に、gbakを使用してバックアップ／リストアを行うことを強く推奨します。これを行えない場合でも、データベース移行時には最低でも全ての複合インデックスを再作成して下さい。

Firebirdの比較的古いバージョン（ODS11.1以下）またはバージョン2.5.0からアップグレードするデータベースは、この不具合の影響を受けません。

このサブリリースでは新機能の追加はありませんが、バージョン2.5.2以降に累積した多くのバグフィックスを含んでいます。このサブリリースでの改善点はわずかで小さなものです。すなわち、

- 現在の接続と現在のトランザクションについての詳細情報を取得するため、SYSTEM名前空間に新しいコンテキスト変数が追加されました。

  追加された変数：現在の接続についてはSYSTEM::CLIENT_PIDおよびSYSTEM::CLIENT_PROCESS、現在のトランザクションについてはSYSTEM::LOCK_TIMEOUTおよびSYSTEM::READ_ONLY。

- 上限の拡大：

  - Windows上のスーパーサーバーとスーパークラシックサーバーへの最大接続数が1024から2048に拡大されました。

  - 外部関数（UDF）の入力パラメータ数の上限が15まで拡大されました。

- エラーレポートの改善：

  - “使用中のオブジェクト”のエラーに関するレポートが詳細になりました。

  - エラーを起こした文脈を特定しやすくするため、整合性制約検証エラーメッセージのテキスト中にリレーション名が追加されました。

  - インデックスと制約の違反に関するエラーレポートが拡張され、問題のあるキーの値を含むようになりました。

- 物理バックアップ（ALTER DATABASE BEGIN/END BACKUPコマンドやnBackupユーティリティを用いるもの）で、バックアップ状態がストールドからマージに切り替わる時、メインのデータベース・ファイルの伸張が高速化されるよう改善されました。

  - データベースの物理バックアップ状態がストールした時のアロケーションテーブル・ロックの競合が低減されました。

  - fallocate()をサポートするLinuxシステム上では、ファイルサイズのより高速な拡大が可能となりました。

  - アロケーションテーブルが初めて読み込まれる時にアタッチメントが他をブロックすることがなくなりました。

- SET STATISTICS INDEX文を実行しても同時接続中のアタッチメントがブロックされたり遅延したりしなくなりました。

- スイープ終了時のlimboトランザクションのスキャンが改善されました。

- エンベデッドSQL（ESQL）にUPDATE OR INSERT文とRETURNING句のサポートが実装されました。

Firebird 2.5.2セキュリティアップデート1

2013年3月にFirebirdサーバーでのリモートスタック・バッファオーバーフローが見つかりました。これにより、認証されていないユーザーがサーバーをクラッシュさせたり、リモートからコードを実行することが可能となっていました。

この脆弱性はAlex　Peshkovによって修正されました。ビルド番号26539以下の全てのFirebirdバイナリと、2013年3月8日より前の全てのスナップショットにはこの脆弱性があります。

Firebird 2.5.2サブリリース

RFC-4122の要請に正しく準拠するため、GEN_UUID()関数の実装に重大な変更が加えられました。詳細情報はこちらの項参照。

ビッグエンディアンのプラットフォーム上で関数CHAR_TO_UUIDとUUID_TO_CHARの結果、問題のあるバイトオーダーやキャラクタオーダーが生成されていたバグが修正されました。この修正により、Firebird　2.5や2.5.1では、ビッグエンディアンのホスト上でこれらの関数を呼び出していたコードに影響が出ます。

バージョン2.5.1以降数ヶ月での累積分以上のバグフィックスに加え、このサブリリースでは、少数の小さな改善、特に管理者支援に関する改善が施されました。概要：

- トレース・サービスにいくつかの歓迎すべき改善が施されました。すなわち、
  - 手動・自動スイープ活動のログを取るためのセッションが設定できるようになりました。このオプションに関するドキュメントは、トラッカー・チケットCORE-3656で見ることができます。
  - TRACEが、トランザクション終了後に起こるアクションの統計情報を生成するようになりました。トラッカー・チケットCORE-3598参照。
  - TRACEに、実行時に発生したエラー（ロックコンフリクトやキー違反など）のログを取る機能が提供されました。トラッカー・チケットCORE-3539参照。
- リモートのバックアップ／リストアを実行するAPIが利用できるようになりました。

  リモート・バックアップファイルによるバックアップとリストア参照。
- 自動スイープの開始時、firebird.log にログが書き込まれるようになりました。
- FirebirdのビルドシステムにCプリプロセッサフラグのサポートが追加されました。

Firebird 2.5.1サブリリース

このサブリリースでは、かなりの数のバグフィックスに加え、先のリリースでは見落とされていた少数の小さな改善と最適化が施されています。概要：

- MacOSX　10.7で、スーパーサーバーとスーパークラシックサーバーの起動に失敗するという重大なバグがありました。このバグは、トラッカー・チケットCORE-3589に記載され、このサブリリースのBug Fixes listにも載せられています。
- WHEN文中でエラー条件のテストに使われるコンテキスト変数GDSCODEやSQLCODEに相当するコンテキスト変数SQLSTATEがPSQLで利用できるようになりました。
- グローバル一時表（GTT）使用時のパフォーマンスを向上させる二三の改善が施されました。
  -

トランザクションをロールバックする際、ON COMMIT DELETE ROWSオプション付きで生成されたGTTに加えられた変更の“取り消し”は、不要なオーバーヘッドだったため、回避されました。

- グローバル一時表のガベージコレクションは他のアタッチメントでアクティブなトランザクションによって必要以上に遅くなっていました。このボトルネックは解消されました。

· 小さなチャンクの割り当てに関して、テンポラリ・スペース・マネージャに必要な最適化が施されました。

· DELETE FROM MON$xxxやfb_cancel_operationのいずれの要求にも応じない他のトランザクションが終了するまで、トランザクションがWAITモードのまま果てしなく待機し続けるというような状況を回避するため、待機状態をキャンセルできるロックマネージャが提供されました。

· クエリオプティマイザが実際のレコード圧縮比を推定することで、テーブル内に格納されたレコード数のより正確な推測が可能になりました。

· リモートインターフェースにいくつかの小さな改善が施されました：

- メッセージ・バッファ内の未使用のVARCHAR値のバイト数がゼロに設定されるようになりました。

- SO_KEEPALIVEオプションをクライアントのTCPソケットに設定して下さい。

· MON$STATEMENT_ID値の定数が監視中のスナップショットの間でも定数として維持されるようになりました。

· このサブリリースでは、Linux/HPPAおよびLinux/Alpha向けのポーティングも完了しています。

バグレポート

· このリリースで新たなバグを発見したと思った方は、Firebirdプロジェクトのウェブサイト中の記事[[有効なバグレポートの書き方]]で、バグレポートに関する説明をぜひお読み下さい。

· バグフィックスが無効だったり不具合を起こしていると思った方は、トラッカーの中からもとのバグレポートを見つけ出し、必要ならそれを再開した上、以下の説明に従って下さい。

発見したバグをご自身で分析する場合は、以下のガイドラインに従って下さい：

1. Firebirdの正確なビルド番号を記述し、詳細なバグレポートをお書き下さい。また、OSプラットフォームの詳細情報も記述して下さい。レポートには再現可能なテストデータを付けて、われわれのトラッカーにポストして下さい。

2. あなたがfield-testersメーリングリストに参加し、最善を尽したバグの記述をポストして、このプレリリース版のフィールドテスターとして名乗りを上げれば、温かく迎え入れられるでしょう。

3. バグや実装に関するディスカッションのスレッドを開始したい場合は、firebird-develメーリングリストに参加した上で行って下さい。フォーラム内でこのアルファ版に関するトラッカー・チケットをポストしておけば、フィードバックを受けられるかもしれません。

ドキュメント

このリリースノートで参照している全てのREADMEドキュメントは-参照されていない多くの他のドキュメントも-Firebird　　2.5がインストールされたディレクトリのサブディレクトリ内にあります。

トラッカー内に自動生成された"リリースノート"ページでは、このプレリリース・バージョンと他のプレリリース・バージョンに関連した全てのトラッカー・チケットのリストとリンクが提供されています。このリンクを利用して下さい。

--Firebirdプロジェクト

Chapter 2

Firebird 2.5の新機能

Firebird　2.5では、より低レベルな同期とスレッドの安全性を概ね確保できる新しいスレッド・アーキテクチャの基礎を確立し、これをスーパーサーバー、クラシックサーバーからエンベデッドモデルにまで全面的に適用していくことが大きな目標でした。

Firebird 2.5.4リリース（2015年3月）

このサブリリースについての一般的注記事項を参照して下さい。

Firebird 2.5.3セキュリティアップデート（2014年12月）

このパッチリリースについての一般的注記事項を参照して下さい。前回リリースのスーパーサーバーとスーパークラシックサーバーに存在した脆弱性が修正されています。

Firebird 2.5.3リリース（2014年7月）

このサブリリースについての一般的注記事項を参照して下さい。

Firebird 2.5.2セキュリティアップデート1（2013年3月）

このパッチリリースについての一般的注記事項を参照して下さい。

Firebird 2.5.2リリース（2012年10月）

このサブリリースについての一般的注記事項を参照して下さい。

Firebird 2.5.1リリース（2011年9月）

バージョン2.5では、最初のリリース以来数ヶ月にわたる数多くのトラッカー・イシューへの取り組みにより、「broken（壊れている）」「bent（挙動が怪しい）」「compromised（欠陥がある）」との報告を受けた事柄への対処が行われてきました。バグフィックスの長いリストに加え、少数の小さな改善が加えられています。概要：

- SQLSTATEコードがPSQLのコンテキスト変数として、WHEN　..　exception文中でのGDSCODEやSQLCODEと同様の方法で利用できるようになりました。

- 読み取り専用のデータベース中でグローバル一時表への書き込みが可能になりました。
- 内部トレース・エラーの診断が改善されました。
- fbtracemgrユーティリティが定期的に出力への書き込みを行うようになりました。
- データ書き込み段階でのgbakによるリストアのパフォーマンスが改善されました。
- BLOBと他のデータタイプとの変換がAPIの機能で可能になりました。
- サービスAPIが“metadata-only”リストアをサポートするようになりました。
- POSIXでのmake installに“サイレントインストール”スイッチが実装されました。
- メッセージ・バッファ内の未使用のVARCHAR値のバイト数がゼロに設定されるようになりました。
- オプティマイザで実際のレコード圧縮比が推定されるようになりました。
- MON$STATEMENT_ID値が監視中のスナップショットの間でも定数として維持されるようになりました。
- 最近のWindowsシステムによる攻撃的なソケットのタイムアウトから保護するための措置として、SO_KEEPALIVEオプションがクライアントのTCPソケットに設定されるようになりました。
- ロックマネージャの使用により果てしない待機状態をキャンセルできるようになりました。
- バージョン2.5.1のLinuxとAlpha向けのポーティングが行われました。

Firebird 2.5リリース（2010年10月）

SQLの拡張はこのリリースの主たる目的ではありませんでしたが、SQLにCREATE/ALTER/DROP USER文や、ALTER VIEWとCREATE OR ALTER VIEW向けの構文が実装されたことで、初めてユーザー管理が可能になりました。PSQLの改善には、自律型トランザクションの導入や、EXECUTE STATEMENTを介して別のデータベースにクエリを発行する機能などが含まれます。

その他の新機能

このリリースでの他の新機能と改善点は、以下の通り：

管理の強化

- サービスAPIを介したシステム監査トレースとユーザートレースのセッションにより、データベース内で動作中のあらゆる事柄をリアルタイムで監視・分析することが可能になりました。
- ODS 11.2データベースで追加された新しいシステムロールRDB$ADMINにより、SYSDBAがデータベースごとに自身の権限を他のユーザーへ譲渡できるようになりました。
- モニタリングテーブル内の情報が詳細になりました。

- 接続を非同期にキャンセルできるようになりました。
- 通常のユーザーが、CURRENT_CONNECTIONだけでなく自身のどのアタッチメントに対しても監視を行えるようになりました。

他のSQL言語の追加と拡張

- SIMILAR TO句を用いた正規表現がサポートされました。
- 計算項目に対するALTER COLUMNが実装されました。
- PSQLモジュール（ストアドプロシージャ、トリガ、動的に実行可能なPSQL文）内での自律型トランザクションが利用可能になりました。
- ビュー定義でのストアドプロシージャへのアクセスが拡張されました。
- GRANT文やREVOKE文のオプションGRANTED BY（またはAS）により、CURRENT_USER（デフォルト）以外のユーザーを権限付与者とすることが可能になりました。
- REVOKE ALL構文により、一度にユーザーやロールの全ての権限を取り消すことが可能になりました。
- WHERE SOME_COL = ? OR ? IS NULL句がサポートされました。
- 標準SQLの予約語ではないキーワードの内、一部のものを除いて“予約語”から解除しました。

データ処理の拡張

- 新しい組み込み関数により、UUID CHAR(16) OCTETS文字列からRFC4122-compliantフォーマットへの変換とその逆の変換が可能になりました。
- 32-bit整数と64-bit整数を、数値リテラル形式およびX-prefixedバイナリの文字列リテラル形式のhexadecimalとして渡せるようになりました。

APIの追加

- SQL文が、SQL-2003標準の英数5文字のSQLSTATEコードを返すようになりました。

> Tip
>
> バージョン2.5.1サブリリースで、SQLSTATEコードがPSQLのコンテキスト変数に追加され、WHEN .. exception文中でのGDSCODEやSQLCODEと同様の方法で利用できるようになりました。

- 新しい定数DSQL_unprepareをisc_dsql_free_statementで利用することによって、プリペアドステートメントを効率的にアンプリペアすることができるようになりました。

国際言語のサポート

- CREATE DATABASE文でデフォルトCOLLATE句を指定できるようになりました。
- 使用するキャラクタ・セットに合わせてデフォルトCOLLATEを変更できるようになりました。
- GBAKのリストア・スイッチFIX_FSS_DATAとFIX_FSS_METADATAにより、スクリプトや手作業に頼らずに、UNICODE_FSSデータおよび／またはメタデータを含むレガシー・データベースを正しくリストアできるようになりました。
- Unicodeのアクセントを区別しないコレーションが可能になりました。

Chapter 3

Firebirdエンジンの変更

このリリースの主な目的は、マルチプロセッサのハードウェアが持つ対称型マルチプロセッシング（SMP）の性能を活用するため、Firebirdのスレッド・アーキテクチャをリファクタリングすることでした。これにより、複数のデータベースが同時にアクセスされるようなスーパーサーバーのスケーラビリティに著しい効果が得られます。しかし、その重要な成果は、Firebird 3に向けて開発中の、きめ細かなマルチスレッディングを支える“スーパークラシックサーバー”モデルの登場です。

新しいスレッド・アーキテクチャ

Dmitry Yemanov
Vladyslav Khorsun
Alex Peshkov
さらに ―
Nickolay Samofatov
Roman Simakov

スーパーサーバーでは、新しいアーキテクチャの大きな利点として次の二点が挙げられます：

1. 複数データベース環境では、各データベースに対するスーパーサーバーのスレッドが、利用可能なプロセッサに均等に割り当てられます。

> Note
>
> デフォルトのCpuAffinity設定では、スーパーサーバーはシングルプロセッサにバインドされたままです。複数のデータベースを活用する際にこの改善の恩恵を受けるには、firebird.confの設定を変更する必要があります。

2. SMPハードウェア上では、単一のデータベースを利用する場合でも、スケーリングのわずかな改善が認められるはずです。

クラシックサーバーでは、さらに顕著な効果が得られます：

1. クラシックサーバーがマルチスレッド化できるようになりました。1プロセスに1ワーカースレッドというモデルは従来通りですが、非同期のシャットダウン、スイープ、ロックマネージャとのプロセス間通信など並行したタスクの処理に追加のスレッドを利用できるようになりました。

2. POSIX版では、クラシックサーバーでも、以前のようにフォークしたプロセスではなく、スレッドでサービスを運用できるようになりました。

> Note
>
> Windows版のクラシックサーバーでは、サービスはバージョン2.1ですでにスレッド化されていました。

3. エンベデッドライブラリ–POSIX版ではlibfbembed.so、Windows版ではfbembed.dll–は、マルチスレッドに対応し、スレッドセーフとなっています。そのため、これらはマルチスレッドアプリケーションから利用できます。

4. テストでは、このバージョンのクラシックサーバーのパフォーマンスは以前のバージョンに比べて相当速くなることが示されています。

“スーパークラシックサーバー”

この、クラシックサーバーのマルチスレッド・モードは、単一のサーバー・プロセス内で–割り当て中またはプーリング中–の複数のワーカースレッドを処理する能力から、“スーパークラシックサーバー”と名づけられています。通常の機能については全てクラシックサーバーと共通していますが、二三の相違点があります：

- どのプラットフォーム上でも、サーバーエンジンの安全かつ完全なシャットダウンが可能です。
- あるTPCベンチマークでは、クラシックサーバーを15〜20%ほど上回るパフォーマンスを示しています。
- カーネル・リソースの使用量が少なくなります（メモリ使用は減りませんが）。
- スーパークラシックサーバーのプロセスがクラッシュした場合、全ての接続が道連れになります。
- アタッチメント／アクティブユーザーのリストを取得できないなど、クラシックサーバー用サービスAPIに見られる制限は、スーパークラシックサーバーには適用されません。
- POSIX版のスーパークラシックサーバーが[x]inetdを必要としません。

> エンベデッドサーバーに関する注意
>
> - WindowsのDLLライブラリであるエンベデッドサーバーfbembed.dllが、以前のスーバーサーバーではなく、スーパークラシックサーバーを利用するようになり、このモデルをPOSIX上のスーパークラシックサーバーへのローカル接続に統合します。以前には単一のアプリケーション空間への接続を制限していたデータベースのファイルロックが、異なるエンベデッドサーバー・モジュールから同じデータベースへの同時アクセスを可能にするグローバルロックテーブルに置き換えられました。アプリケーションの同時デバッグやgbakやgstatのようなネイティブのユーティリティ・ツールの利用を容易にします。
> - 単一のアタッチメント・ハンドルが同時スレッドに共有されるようになりました。（トラッカー・リファレンス CORE-2498 A. dos Santos Fernandes）。

使用上の注意

Windows版

Windows版では、同じfb_inet_server.exe　バイナリが、スイッチの設定によって、クラシックサーバーとスーパークラシックサーバー、いずれの作業モードをも提供します。デフォルトはクラシックサーバー・モードです。

スーパークラシック・モードをサービスとして利用するには、次のように、コマンドラインでinstsvc.exeに-m[ulti-threaded] スイッチを付けます。

```
instsvc install -multithreaded
```

スーパークラシックサーバーをアプリケーションとして運用したい場合は、次のようにします。

```
fb_inet_server -a -m
```

POSIX版の新バイナリ

POSIX版では、スーパークラシックサーバー・モデルとして新しいバイナリfb_smp_server　が提供されました。これにはネットワークリスナーが含まれているため、アタッチメント・リクエストに関してfbserverと同様に働き、[x]inetdを必要としません。

fb_smp_server　が使用するマルチスレッド・エンジンは、OSRI要請に従い、libfbembed.so となっています。クラシックサーバーのパッケージにはfbguard (Guardian) も含まれます。これは、スーパークラシックサーバーのインストールの際に、fbserverではなくfb_smp_server　を開始します。スーパーサーバー・モデルがGuardianと一緒にインストールされた際に行われることと同様です。

> Important
>
> 従来のクラシックサーバーの運用中にはfbguardを使用しないで下さい。

スレッドセーフ・クライアント・ライブラリ

Dmitry Yemanov
Vladyslav Khorsun
Alex Peshkov

トラッカー・リファレンス CORE-707

エンベデッドライブラリを含むクライアント・ライブラリが、アプリケーションレベルでの同期なしでも、マルチスレッドアプリケーションで使用できるようになりました。

改善点

実装された改善点は以下の通り：

クラシックサーバーで接続の切れたクライアントの即時検出

Vladyslav Khorsun

クラシックサーバーが、クライアントの切断によって壊れたサーバーのプロセスを、すぐに検出できるようになりました。これに対し、未完了の活動を終了させたり、アクティブなトランザクションをロールバックしたり、ネットワークの接続を閉じたりといった対応をします。

トラッカー・リファレンス CORE-818

最適化

重要な最適化は以下の通り：

データの取得

Dmitry Yemanov

最適化により、どのフィールドもアクセスされていないテーブルについて、データ取得のパフォーマンスが改善されました。これは、例えば、SELECT COUNT(*)型のクエリに適用されます。

トラッカー・リファレンス CORE-1598

BLOBメモリの使用法

Adriano dos Santos Fernandes

最適化により、それぞれの一時的なBLOBに割り当てる際に生成された、<ページサイズ>バイトのメモリ消費を避けることができます。

トラッカー・リファレンス CORE-1658

更新パフォーマンスの改善

V. Khorsun

この改善の狙いは、“careful write”による更新手続きの際にエンジンが優先的に行う書き込みの量を減らすことでした。従来の手続きでは、大規模な更新、特に同一のトランザクションで同一のレコードへの複数回の更新を行う“updates-in-place”のパフォーマンスに目立った影響が出ていました。最悪の場合には、更新の際に生成された全ての単一の新しいレコードのバージョンのために、ディスクにページが書き込まれることもありました。

内部で何が起きているのかについての簡単な技術的解説は、トラッカー・リファレンス (CORE-2672)を参照して下さい。

この解決策では、正しい優先順位を維持するためレコードの新バージョンと旧バージョンとが置かれたページ間で起きる循環参照から、書き込み操作を保護することに関わるプロセスで多大な時間を要する可能性があったという事態に対処しています。

64bitサーバーでのキャッシュサイズ制限の拡大

V. Khorsun

トラッカー・リファレンス CORE-1687

以前の64bit版Firebirdサーバーでは、64bitアドレス空間の恩恵が受けられず、データベースのキャッシュを2GB（16K * 128K）以上に設定することができませんでした。この問題は今回のバージョンで修正されました。64bit版Firebirdでは、リソースが利用可能な場合、RAM内に5〜10GBのデータベースに完全に対応できる十分な大きさのキャッシュを設定することが可能になりました。

Firebirdのキャッシングは、ファイルシステムのキャッシュから多くの恩恵を受けることができますが、このことは、主に読み取りに負荷がかかる高スループット・システムにとって、おそらく重要な機能となっています。x64版Firebirdサーバーでのキャッシュの理論上の上限は2^31 -1（2,147,483,647）ページとなっています。

デフォルトでのデータベースの配置

A. Peshkov

トラッカー・リファレンス CORE-1643

設定パラメータDatabaseAccessに、さらに多くの“意味”が付け加えられました。特に指定がない場合、エンジンは、“Restrict”リスト内のDatabaseAccessで最初に登録された配置場所を、新規にデータベースを作成する際や、接続パラメータによってエイリアスもフルパスも指定されていないデータベースを見つけるための、デフォルトの配置として受け取ります。

この探索ロジックは、RestrictリストからExternalFileAccessパラメータに渡されて、外部テーブルを見つけるために使われるのと同様のものです。すなわち、

1. Restrctリスト内の全てのディレクトリが最初に探索されます。
2. 指定されたデータベースが見つからない場合：
   - CREATE DATABASEが含まれる場合、Restrictリスト内の最初の配置場所が使われます。
   - そうでない場合、アタッチは期待通りには行なわれません。

> 注意 :: 現在の作業ディレクトリ
>
> この機能は、直接ローカル接続するために指定されたデータベース・ファイルの暗黙の配置場所として現在の作業ディレクトリを使用することを禁ずるものではありません。これらの場合には、従来とまったく同様に、Y-valveがパス解決を処理します。
>
> データベースがどこに“ある”のか確かめる方法がないからといって、リモート・サブシステムを介して稼働中のスタンドアロンのサーバーが、パスの記述がないデータベース・ファイル名を使って接続しようとするというのは、まずありえない状況ですが、推奨されません。例えば、Windows版では、こうした状況での現在の作業ディレクトリは%system%となります。

Windows版エンベデッドエンジン向けDLLのロード

Adriano dos Santos Fernandes

アプリケーション構成のインストール時に、いわゆる“DLL地獄”に陥った場合に共通して起こる問題を避けるため、Windows版エンベデッドエンジン向けのルート決定メカニズムが変更されました。以前には、ユーザーアプリケーションのメインの実行ファイルを含むディレクトリが暗黙のルートディレクトリとなっていました。今後は、リネームされたfbembed.dllライブラリが置かれているディレクトリとなります。

トラッカー・リファレンス CORE-1814

大きな外部テーブルのサポートが有効に

Vlad Khorsun

Firebirdの以前のバージョンでは、外部テーブルを使用する際に32bit　I/Oを使っていたため、外部ファイルのサイズは2GB未満に制限されていましたが、64bit　I/Oがサポートされるファイルシステム上では、外部テーブルメカニズムがこれを利用するよう拡張されたことで、2GBの制限は事実上撤廃されました。

トラッカー・リファレンス CORE-2492

データベースの統計が64bit値で適切に動作するように

V. Khorsun
A. Peshkov

トラッカー・リファレンス CORE-2619

Firebirdの以前のバージョンでは、メモリその他の統計が64bit値を適切に処理していませんでした。このイシューには二つの要素があります：

a.　エンジンが統計に64bit整数を使えるように、AtomicCounterの内部が変更されました。

b.　isqlおよびqliのツールを64bit値に見合った動作をするよう改良する必要がありました。

> 古いクライアントとの非互換性
>
> 32bitのツールが64bitのサーバーで正しく動作するため、いくつかの新しい内部API機能（struct perf64やperf64_xxx）を導入し、またそれらを使用するisqlやqliを変更する必要がありました。このことは、バージョン2.5のisqlとqliのプログラムが古いFirebirdクライアントとの互換性を持たないことを意味します。

UDFのセーフガード

Adriano dos Santos Fernandes

トラッカー・リファレンス CORE-1937

Firebirdサーバーがアクセスしているのと同じランタイムによって割り当てられていないポインタを返すような文字列UDFが書かれている場合、その宣言中のFREE_ITキーワードの存在が、メモリを汚染し、サーバーをクラッシュさせます。そのような異常なUDFに対するセーフガードとして、エンジンは…

1. そのようなUDFを検出し、例外をスローします。
2. エンベデッド版を含む全てのサーバー・モデルで、更新されたib_utilライブラリがパスの中に存在するか否かに依存します。

診断

トランザクションの診断

Claudio Valderrama

TPBの内容に不正がある場合の診断とエラーレポートが改善されました。新たなTPB検証ロジックは、以下のものをリジェクトします：

- 同じカテゴリの中で明らかに矛盾するオプション。例えば、一緒に指定された{WAIT}と{NOWAIT}、あるいは、{READ COMMITTED}と{SNAPSHOT}、あるいは、{READ ONLY}と{WRITE}。
- 無意味なオプション。例えば、SNAPSHOT分離モードに指定された[NO] RECORD VERSION。
- テーブル予約オプションの誤った命令。例えば、{READ <TABLE> PROTECTED}と間違えて{PROTECTED READ <TABLE>}。

トラッカー・リファレンス CORE-1600

アクセス権のエラーメッセージ

Alex Peshkov

カラムに対するアクセス権の例外が起きた時、テーブル名とカラム名の両方が報告されるようになりました。

トラッカー・リファレンス CORE-1234

メッセージの改善

V. Khorsun

トラッカー・リファレンス CORE-2587

エンジンがWindowsの他のセッションで別のエンジンのプロセスによってすでにマッピングされた共有メモリを作成できない場合の診断メッセージが、少しだけユーザーフレンドリーになりました。従来は次のようなメッセージでした：

```
The requested operation cannot be performed on a file with a user-mapped section open.
```

（要求された操作はユーザーマップセクションで開いたファイルでは実行できません。）

これが、次のようなメッセージとなります：

```
Database is probably already opened by another engine instance in another Windows session.
```

（データベースはおそらく他のWindowsセッションの他のエンジンのインスタンスによりすでに開かれています。）

メタデータの改善

キャラクタ・セットをデフォルトのコレーションに保存

Adriano dos Santos Fernandes

システムテーブルRDB$CHARACTER_SETS内のRDB$DEFAULT_COLLATE_NAMEの現在の値がバックアップ／リストアのサイクルの中で維持されるように改善されました。このカスタマイズのためのメカニズムが新しいALTER CHARACTER SETコマンドです。

トラッカー・リファレンス CORE-789

Chapter 4

Firebird APIとODSの変更

ODS（On-Disk Structure）の変更

オンディスク構造体（ODS）の変更には以下のものが含まれます：

新たなODSバージョン数

Firebird 2.5はODS（オンディスク構造体）バージョン11.2のデータベースを作成します。

最大ページサイズ

最大ページサイズは従来通り16KB（16384バイト）です。

キャッシュ内の最大ページバッファ数

データベース・キャッシュとして設定できる最大のページ数は、データベースが64bit版Firebirdで運用されているか、32bit版で運用されているかによって異なります：

- 64bit版 :: 2^31 -1（2,147,483,647）ページ
- 32bit版 :: 128,000ページ、つまり、バージョン2.1と同じです

API（アプリケーションプログラミングインタフェース）の拡張

追加されたFirebird APIは、以下の通り―

APIの機能でBLOBの変換が可能に

A. dos Santos Fernandes

トラッカー・リファレンス CORE-3446

BLOBと他のデータタイプとの双方向の変換がAPIの機能で可能になりました（XSQLVARまたはblr messages）。

- BLOBから異なるタイプのデータへ、また、異なるタイプからBLOBへのデータの移動が、executeおよびfetchの呼び出しで可能になりました。

- 入力パラメータに関しては、パラメータに文字列を置くことができるので、クライアント側からBLOBの作成・記入を行う必要がありません。
- 出力（executeまたはfetch）に関しては、アプリケーションがそのデータを理解し、BLOBを最長の文字列として評価するのに役立ちます。

接続文字列とキャラクタ・セット

A. dos Santos Fernandes

以前のバージョンは、OSやファイルシステムで使用されるキャラクタ・セットと連携する手段を持っていませんでした。Firebird 2.5は、データベースその他のファイルの名称や文字列パラメータがAPI接続要求を通じてアクセスされたとき、および／または、API接続要求に渡されたとき、その全体を見て、“環境に応じた認識”を行うようになりました。この変更により、FirebirdがASCIIサブセット以外のキャラクタを含むファイル名や他のパラメータを受け付け、それらを使用する性能は大幅に改善されました。

> この機能をサポートするのはDPB接続のみ
>
> 現在の実装では、この機能をサポートしているのはDPB（データベースパラメータブロック）を通じた接続だけです。サービスAPI（isc_spb*）の機能はサポートされていません。

isc_dpb_utf8_filename

新しい接続オプションisc_dpb_utf8_filenameが導入され、これにより、Firebirdは、渡されているファイル名や他のキャラクタ・アイテムがUTF8（UTF-8）キャラクタ・セットであることを具体的に知ることができるようになりました。このオプションが使われない場合、デフォルトのキャラクタ・セットとしてOSのコードページが選択されます。

クライアントとサーバー間の互換性

新しいクライアント、古いサーバー
: バージョン2.5以降のクライアントで2.5より前のバージョンのリモート・サーバーに接続する場合、isc_dpb_utf8_filenameオプションを使用すると、クライアントは、ファイル名をサーバーへ渡す前に、これをUTF-8からクライアントのコードページへと変換します。isc_dpb_utf8_filenameオプションはDPBから削除されます。

  クライアントとサーバーのステーション間で同じコードページが使われている時、互換性は保証されます。

新しいクライアント、新しいサーバー、isc_dpb_utf8_filenameオプションなし
: バージョン2.5以降のクライアントで、isc_dpb_utf8_filenameオプションを使用せずに、バージョン2.5以降のリモートサーバーに接続する場合、クライアントはファイル名をOSのコードページからUTF-8へと変換し、isc_dpb_utf8_filenameオプションをDPBに挿入します。

  サーバーが受け取ったファイル名には特別な処理は施されません。しかし、古いクライアントの場合と違い、バージョン2.5のクライアントは、ファイル名を自動で変換し、DPBにisc_dpb_utf8_filenameオプションを自動で挿入します。いずれにせよ、ホストとクライアントが同じコードページを用いている場合には、互換性は保証されます。

新しいクライアント、新しいサーバー、isc_dpb_utf8_filenameオプション使用

isc_dpb_utf8_filenameが使用されている場合は、クライアントはファイル名を変更せずにサーバーに渡します。クライアントは常にUTF-8のファイル名をisc_dpb_utf8_filenameオプションとともにサーバーに渡します。

コードページの変換

Windowsでは、変換に使われるコードページはWindows　ANSIです。他の全てのプラットフォームではUTF-8が用いられています。

ファイル名にOSのコードページやUTF-8を用いるのは、必ずしもベストな選択とは言えません。例えば、スクリプトや他のテキストファイルを別の接続キャラクタ・セットを用いたisqlなどのスクリプト実行ツールで処理する場合、複数のキャラクタ・セット（コードページ）を使用していると、ファイルを正しく編集できないことがあります。

解決策：Unicodeコードポイントを使用します。これにより、クライアントのバージョンが2.5より前の場合でも、キャラクタの正しい解釈が可能になります。

Unicodeコードポイントの使用

接続文字列のファイル名にある任意のUnicodeキャラクタはエンコードによりASCIIキャラクタとして擬装できるようになりました。これは、Unicodeコードポイント番号（U+XXXXの表記法に似たhexadecimal形式で記述）にプレフィックスとしてシンボル#を付すことで実現されます。

#XXXXのように標記します。Xは0-9、a-f、A-Fです。

キャラクタの一つがリテラルな#だった場合、“二重の”ハッシュ・キャラクタ（##）か、またはコードポイント番号#0023を使用します。

> Note
>
> クライアントがバージョン2.5より古い場合でも、サーバーでのハッシュ・キャラクタの解釈にはこれら新しいセマンティクスが使われます。

SQLSTATEコードのサポート

W. Oliver
D. Yemanov

トラッカー・リファレンス CORE-1761

新しいクライアントサイドAPI関数fb_sqlstate()は、エラーのステータスベクター・アイテムをSQL-2003標準の英数5文字のSQLSTATEコードへと変換するために利用できます。

- SQLSTATEコードはSQL CLASSの2文字とSQL SUBCLASSの3文字を連結したものを表現しています。
- SQL文がSQLSTATEコードを返すようになりました。
- isqlユーティリティがエラーについてSQLCODEではなくSQLSTATEの診断を表示するようになりました。
- SQLCODEでの診断は非推奨となっています–将来のリリースで廃止されます。
- （バージョン2.5.1）PSQLにWHEN SQLSTATE型の例外処理構文が追加されました。

> SQLCODEの非推奨化
>
> SQLCODEは非推奨となっており、代わりにSQLSTATEを使用すべき所ですが、FirebirdではSQLCODEもこの先しばらく使うことができます。WHEN SQLCODEの例外処理のようなAPI関数isc_sqlcode()はまだサポートされています。

付録A：SQLSTATEには、このリリースで使用できるSQLSTATEコードのリストと対応するメッセージ・テキストが挙げられています。

“効率的なアンプリペア”

W. Oliver
D. Yemanov

トラッカー・リファレンス CORE-1741

APIルーチンisc_dsql_free_statement()の新たなオプションDSQL_unprepare（数値4）を使うと、DSQL文ハンドルはプリペアドステートメントを“アンプリペア”のままにしておくことができます。

従来のisc_dsql_free_statement()関数は、DSQL_close（名前付きカーソルを閉じる）とDSQL_drop（文ハンドルを解放する）のみサポートしていました。

追加されたAPIは次の通り：

```
#define DSQL_close 1
#define DSQL_drop 2
#define DSQL_unprepare 4
```

オペレーション関数のキャンセル

Alex Peshkov

新しいAPI呼び出しfb_cancel_operation()を使うことで、所与の接続中に、ある種のブロッキングAPI呼び出しが実行している現在のアクションをキャンセルすることができます。

構文

```
ISC_STATUS fb_cancel_operation(ISC_STATUS* status_vector,
                               isc_db_handle* db_handle,
                               ISC_USHORT option);
```

パラメータ

status vector (ISC_STATUS* status_vector)
　通常のステータスベクター・ポインタ構造体です。

db_handle (pointer to a isc_db_handle)
　通常の、有効なデータベースハンドルです。アタッチメントを特定します。

option (unsigned short: symbol)
　実行されるアクションを確定します。オプション・シンボルは以下の通り：

- fb_cancel_raise：第二のパラメータで指定されたdb_handleに関連する任意のアクションをキャンセルします。この効果として、できるだけ早い時点でエンジンが継続中のリクエストを止め、中断されたAPI呼び出しのステータスベクターを介して呼び出し元に例外を返せるようになります。

  “できるだけ早い時点”とは、通常は、次の再スケジューリング・ポイントのことです。

  例

```
Thread1:                              Thread2:
------------------------------        ------------------------------

isc_dsql_execute(status, ....)
........                              fb_cancel_operation(cancel_status, ...)
status[1] == isc_cancelled;           cancel_status[1] = 0;
```

- fb_cancel_disable：指定されたアタッチメントに関するfb_cancel_raiseリクエストの実行を無効にします。例えばcleanupなど、プログラムが重要なオペレーションを実行している時に役立つ可能性があります。
- fb_cancel_enable：前に無効化されたキャンセル実行の通知を再び有効にします。'cancel'状態はデフォルトで有効であり、アタッチメントが作成された時に初期化されます。
- fb_cancel_abort；クライアント側で接続を強制的に閉じます。接続をすぐに閉じる必要がある場合は役に立ちます。サーバーは実行中の全てのトランザクションをロールバックします。'Success'の場合は常にアプリケーションに返されます。注意して使用して下さい！

使い方

fb_cancel_disableとfb_cancel_enableリクエストのサイクルは、必要な頻度で繰り返すことができます。エンジンがすでにリクエストされた状態にある場合は、例外は発生しません：単に無視されます。

長時間にわたるリクエストを停止する必要がある場合、通常、fb_cancel_raiseが呼び出されます。これは非同期シグナルに対して安全ではないため、シグナルハンドラからではなく、個別のスレッドから呼び出されます。

> このAPI呼び出しが非同期な特性を持つことに注意して下さい！

非同期な実行の別の側面として、API呼び出しの終了時にアタッチメントの活動がキャンセルされることもあれば、されないこともあり得ます。後者の可能性は常にあります。また、非同期性により、返されるステータスベクターがほとんどの場合FB_SUCCESSを返すことになります。とはいえ、例外が発生することはあります：ネットワークパケット・エラーなど。

例

```
Thread A:
fb_cancel_operation(isc_status, &DB, fb_cancel_enable);
isc_dsql_execute_immediate(isc_status, &DB, &TR, 0, "long running statement", 3, NULL);
// waits for API call to finish...
```

```
  Thread B:
  fb_cancel_operation(local_status, &DB, fb_cancel_raise);

Thread A:
if (isc_status[1])
 isc_print_status(isc_status); // will print "operation was cancelled"
```

シャットダウン関数

Alex Peshkov

このリリースでは、エンベデッドサーバー・アプリケーションで有用となる二つのfb_shutdown*関数が公開されました：：fb_shutdown()とfb_shutdown_callbackです。

関連する二つのfb_shutdown* 関数

このリリースでは、埋め込みサーバー・アプリケーションで有用となる二つのfb_shutdown*関数が公開されました：：fb_shutdown()とfb_shutdown_callbackです。

プロトタイプ

```
typedef int (*FB_SHUTDOWN_CALLBACK)(const int reason, const int mask, void* arg);

int fb_shutdown(unsigned int timeout,
                const int reason);

ISC_STATUS fb_shutdown_callback(ISC_STATUS* status_vector,
                                FB_SHUTDOWN_CALLBACK callback_function,
                                const int mask,
                                void* arg);
```

fb_shutdown()

fb_shutdown()はさまざまなFirebirdサブシステム（yValve、エンジン、リダイレクタ）をスマートにシャットダウンさせます。これは主に内部エンジンが使うために設計されたもので、現在のプロセスにのみ適用できます。これはAPIによってエンベデッドサーバー環境でユーザーアプリケーションに利用していただくことが可能です。

今はエンベデッドエンジンでしか使えませんが、この関数は、現在の全ての活動を停止し、実行中のトランザクションをロールバックし、アクティブなアタッチメントを遮断し、エンベデッドエンジン・インスタンスを穏やかにシャットダウンします。

> アプリケーション開発者向けの注意
>
> fb_shutdown()は、アプリケーションが同時にアタッチされる可能性があるリモートサーバーのシャットダウンを実行しません。実際に、Firebirdの全てのクライアント・ライブラリ-エンベデッドサーバーを含む-は、最低一つでもデータベースまたはサービスにクライアントがアタッチされていれば、これをexit()で自動的に呼び出します。
>
> 従って、リモートのアタッチメントの文脈では、これがクライアントによって呼び出されることは決してありません。

パラメータ

fb_shutdown()は二つのパラメータを取ります：

1. ミリ秒でのタイムアウト

2. シャットダウンの理由

   理由コード（const int reason）は負の値を取りますが、ibase.hにリストが挙げられています： fb_shutrsnで始まる定数を参照して下さい。

> Note
>
> プログラムからfb_shutdown()を呼び出す際には正の値を渡す必要があります。これは、適切なアクションがコーディングされたコールバック関数ルーチンを通じて、fb_shutdown_callback()に引数として渡されます。

戻り値

- 戻り値ゼロはシャットダウンの成功を意味します。
- ゼロ以外の戻り値は、シャットダウン中に何らかのエラーが発生したことを意味します。詳細はfirebird.log に書き込まれます。

fb_shutdown_callback()

fb_shutdown_callback()はシャットダウン中に呼び出されるコールバック関数を設定します。この呼び出しはほとんどの場合正常終了の値を返しますが、メモリ不足の状態などでエラーが返されることもあります。

パラメータ

fb_shutdown_callback()は四つのパラメータを取ります：

status vector (ISC_STATUS* status_vector)
: 通常のステータスベクター・ポインタ構造体です。

pointer to callback function (FB_SHUTDOWN_CALLBACK callback_function)
: これは、コールバックが発生した時に取るべきアクション（あるならば）を実行するため記述しておいたコールバック関数を参照します。

  コールバック関数は三つのパラメータを取ることができます。第一と第二のパラメータはコールバックの際に取るべきアクションを決めるのに役立ちます：

  1. シャットダウンの理由

     シャットダウンの理由として二つのものが特に重要です：

     - fb_shutrsn_exit_called：exit()により、またはクライアント／エンベデッドライブラリがアンロードされることにより、Firebirdは終了します。
     - fb_shutrsn_signal、POSIXのみ適用：SIGINTまたはSIGTERMシグナルを受信した場合です。

> Note
>
> Firebirdは常に負の理由コードを使いますが、ユーザーがfb_shutdown()そのものを呼び出す際には正の値を使用することが求められます。

2. 呼び出しに用いられるマスクの実際の値

   このパラメータの用途は、コールバックを呼び出すのをエンジンのシャットダウンの前にするか後にするかを決めるのに役立つということです。

3. ユーザーアプリケーションによってfb_shutdown_callback()に渡される引数

   これは任意の目的で使用でき、NULLにすることもできます。

コールバック関数からの戻り値
コールバック関数がゼロを返した場合、これはジョブが正常に終了したことを意味します。ゼロ以外の戻り値は、コールマスク（下記のパラメータに関する項目を参照）に従って解釈されます：

- fb_shut_postproviders呼び出しの場合、何らかのエラーが発生してfb_shutdown()からゼロ以外の値が返されていることを意味します。エラー状態が返された正確な理由の通知はコールバック関数が担うことになります。
- fb_shut_preprovidersコールの場合、シャットダウンが実行されないことを意味します。

> Tip
>
> exit()が呼び出されてシャットダウンが実行される場合、ゼロ以外の値を返すのは良い考えではありません！ ;-)

コールマスク（const int mask）
以下のシンボル記号値を持つことができます：

- fb_shut_preproviders：コールバック関数はエンジンがシャットダウンされる前に呼び出されます
- fb_shut_postproviders：コールバック関数はエンジンがシャットダウンされた後に呼び出されます
- 両者を論理和で結合した場合は、シャットダウンの前後に同じ関数が呼び出されます

コールマスクの値

fb_shut_confirmation
　エンジンが質問：みんなシャットダウンの準備はできているかい？

fb_shut_preproviders
　プロバイダが終了する前に実行されるアクション

fb_shut_postproviders
　プロバイダが終了した時に実行されるアクション

fb_shut_finish
　最終的なクリーンアップ

fb_shut_confirmation（fb_shut_preprovidersではないのですが）に対してゼロ以外の値が返されることは、シャットダウンが実行されないことを意味します。

引数（void* arg）
　これは、コールバック関数に渡される引数です。

fb_shutdown関数の使い方

以下に挙げたのは、シャットダウンとシャットダウンのコールバック機能の使用サンプルです。これは、データベースのアタッチメントがある時に誰かがCtrl-Cを押すことでプログラムが終了されてしまうことを防ぐためのものです。

```
#include <ibase.h>

// callback function for shutdown
static int ignoreCtrlC(const int reason, const int, void*)
{
 return reason == fb_shutrsn_signal ? 1 : 0;
}

int main(int argc, char *argv[])
{
        ISC_STATUS_ARRAY status;
        if (fb_shutdown_callback(status, ignoreCtrlC, fb_shut_confirmation, 0))
 {
  isc_print_status(status);
  return 1;
 }
 // your code continues ...
}
```

シャットダウン用の新しいisc_spb_prp_*定数

新しいデータベースシャットダウンモードをサービスAPIへの呼び出しを使って設定できるようになりました。いくつかの新しいisc_spb_prp_*定数を引数として利用できます。

isc_spb_prp_shutdown_mode と isc_spb_prp_online_mode

これらの引数は、それぞれ、データベースをシャットダウンするためと、オンラインに戻すために使用します。いずれも、gfix -shutの設定と正確に一致する、新しいシャットダウンモードを設定するシングルバイト・パラメータをも持ちます：

- isc_spb_prp_sm_normal
- isc_spb_prp_sm_multi
- isc_spb_prp_sm_single
- isc_spb_prp_sm_full

シャットダウンのリクエストでは、シャットダウンのタイプも指定する必要があります。以下の中のいずれかとなります。

- isc_spb_prp_force_shutdown
- isc_spb_prp_attachments_shutdown
- isc_spb_prp_transactions_shutdown

いずれも4バイトの整数パラメータを取り、リクエストされたシャットダウン操作のタイムアウトを指定します。

> Note
>
> 古いスタイルのパラメータもサポートされており、デフォルトのシャットダウン（現在は'multi'）とオンライン（'normal'）モードに入るために使います。

使用例

以下に、fbsvcmgrユーティリティで新しいパラメータの使う例をいくつか挙げています。便宜上、ログインはすでに確立しているものと仮定しています。いずれの例もページ幅に合わせて改行していますが、実際には一行のコマンドです。

データベースをシャットダウンしてシングルユーザー・メンテナンスモードに：

```
fbsvcmgr service_mgr action_properties dbname employee
  prp_shutdown_mode prp_sm_single prp_force_shutdown 0
```

次に、マルチユーザー・メンテナンスモードを有効にする：

```
fbsvcmgr service_mgr action_properties dbname employee
  prp_online_mode prp_sm_multi
```

今度は完全なシャットダウンモードに入り、60秒間、新しいアタッチメントを無効にする：

```
fbsvcmgr service_mgr action_properties dbname employee
  prp_shutdown_mode prp_sm_full prp_attachments_shutdown 60
```

通常状態に戻す：

```
fbsvcmgr service_mgr action_properties dbname employee
  prp_online_mode prp_sm_normal
```

ヘッダレベルの変更に対する制御の厳格化

Alex Peshkov

危険な抜け穴を閉じることにより、いくつかのDPBパラメータに通常のユーザーからアクセスできなくなりました。それらの設定は、管理者の制御下で実行されたのでなければ、場合によっては、データベースヘッダの設定を変更し、破損を起こす可能性があるものです；つまり、それらは、本来ならSYSDBAに限定される操作を開始してしまうのです。すなわち—

- isc_dpb_shutdown and isc_dpb_online
- isc_dpb_gbak_attach, isc_dpb_gfix_attach と isc_dpb_gstat_attach
- isc_dpb_verify
- isc_dpb_no_db_triggers
- isc_dpb_set_db_sql_dialect
- isc_dpb_sweep_interval
- isc_dpb_force_write
- isc_dpb_no_reserve
- isc_dpb_set_db_readonly
- isc_dpb_set_page_buffers（スーパーサーバーで）

クラシックサーバーでは、パラメータisc_dpb_set_page_
buffers は、今でも通常のユーザーが使用できるようになっています。そのユーザーはそのセッションでのみ、一時的にバッファサイズを設定することができます。スーパーサーバーまたはクラシックサーバーでSYSDBAがこれを使用した場合は、データベースヘッダ内のバッファカウントが変更されます。つまり、デフォルトのバッファサイズは恒久的に変更されることになります。

> データアクセスドライバおよびツールの開発者とユーザー向けの重要な注意事項
>
> この変更は、リストに挙げられたDPBパラメータで明示的に設定されたもののいずれかに影響します。これは、デフォルトプロパティの値によるDPB実装にそれらを含める、または、通常のユーザーとしてデータベースにアクセスするツールやアプリケーションでそれらを有効にする、いずれの方法で設定した場合も当てはまります。例えば、データベースのParamsプロパティに'RESERVE PAGE SPACE=TRUE'や'FORCED WRITES=TRUE'を含むDelphiアプリケーションで、Firebird 1.x、2.0.1、2.0.3、2.0.4また2.1.0／2.1.1に接続した場合には問題を起こさなかったものが、ISC ERROR CODE 335544788、“Unable to perform operation. You must be either SYSDBA or owner of the database.（操作を実行できません。あなたはSYSDBAまたはデータベースの所有者でなければなりません。）”のメッセージを出してSYSDBA以外による接続をリジェクトするようになりました。

アプリケーション用の新しいトレースサービス

Vlad Khorsun

新しいユーザートレース・セッションの管理に関係する五つの新しいサービスがサービスマネージャに追加されました。いずれも対応するサービスAPIアクション関数を伴っています。

isc_action_svc_trace_start

ユーザートレース・セッションを開始します。

Parameter(s)

```
isc_spb_trc_name ： トーレスセッション名、文字列、オプション
isc_spb_trc_cfg  ： トレースセッションの設定、文字列、必須
```

必須パラメータは適切な設定用テキストを含む文字列です。この文字列の内容についてのガイドとして、テンプレートファイルfbtrace.conf がFirebirdのルートディレクトリ内に提供されています。

> Note
>
> 1. システム監査セッションとは違い、ユーザーセッションはファイルから設定を読み取ることはしません。クライアント側でローカルに設定を保存し、実行時に使用するためにそれらを取得する仕組みを考える責任はアプリケーション開発者が担うことになります。
> 2. 文字列内に余分なホワイトスペースがあっても問題ありません：単に無視されるだけです。

出力

・ 操作の状態を伝えるテキストメッセージは、次のいずれかとなります：

```
トレースセッションを開始できません。トレースのプラグインがロードされていません。
```

または、

```
トレースセッションID NNNが開始されました。
```

・ 第二の場合、トレースセッションの結果がテキスト形式で続けられます。

isc_action_svc_trace_stop

指定されたトレースセッションを停止します。

パラメータ

```
isc_spb_trc_id : トレースセッションID、整数、必須
```

出力

リクエストの結果（状態）を通知するテキストメッセージ：

- トレースセッションID NNNが停止されました。
- 他のユーザートレース・セッションを停止する権限がありません。
- トレースセッションID NNNが見つかりません。

isc_action_svc_trace_suspend

指定されたトレースセッションを中断します。

パラメータ

```
isc_spb_trc_id : トレースセッションID、整数、必須
```

出力

リクエストの結果（状態）を通知するテキストメッセージ：

- トレースセッションIDが中断されました。
- 他のユーザートレース・セッションを変更する権限がありません。
- トレースのセッションID NNNが見つかりません。

isc_action_svc_trace_resume

指定されたトレースセッションを中断から再開します。

パラメータ

```
isc_spb_trc_id : トレースセッションID、整数、必須
```

出力

リクエストの結果（状態）を通知するテキストメッセージ：

- トレースセッションID NNNが再開されました。
- 他のユーザートレース・セッションを変更する権限がありません。
- トレースセッションID NNNが見つかりません。

isc_action_svc_trace_list

既存のトレースセッションの一覧

パラメータなし

出力

トレースセッションとその状態を一覧するテキストメッセージです：

- セッションID：〈番号〉
- 名前：〈文字列〉。空でない場合、トレースセッション名を表示します
- ユーザー：〈文字列〉。トレースセッションを作成したユーザーのユーザー名を表示します
- 日付：YYYY-MM-DD HH:NN:SS、ユーザーセッションの開始日時
- フラグ：〈文字列〉、以下のものの一部または全部を含むコンマ区切りのセット：

  active | suspend
  : セッションの起動状態

  admin
  : 管理者ユーザーがセッションを作成した場合はadminを表示。通常のユーザーがセッションを作成した場合は表示せず。

  system
  : セッションがFirebirdエンジンによって作成された場合（システム監査セッション）はsystemを表示。通常のユーザーがセッションを作成した場合は表示せず。

  audit | trace
  : セッションの種類を示す：エンジンが作成した監査セッションの場合はaudit、ユーザートレース・セッションの場合はtrace。

  log full
  : ユーザートレース・セッションでセッションログファイルが満杯の場合に条件付きで表示。

> **Note**
>
> いずれのサービスの出力も、通常は、定期的なisc_service_query呼び出しをisc_info_svc_lineまたはisc_info_svc_to_eofのいずれかの情報アイテムと一緒に使うことで得ることができます。

リモート・バックアップファイルによるバックアップとリストア

Alex Peshkov

バージョン2.5.2では、サーバー側でgbakを起動し、リモートのクライアントに置かれたgbakバックアップファイルから読み取る、またはそれに書き込む機能が導入されました。これはインターネット接続を介したバックアップ／リストアの方法として非常に効果的です。

この機能を使う最も簡単な方法は、コマンドラインツールfbsvcmgrを利用することです。これはサービスAPI呼び出し用のインターフェースをとりあえず提供してくれるので、クライアントアプリケーションを自分で書かずに済みます。

リモートバックアップ

fbsvcmgrを使ってリモートのデータベースをバックアップするには、次のパターンでコマンドを入力します：

```
fbsvcmgr remotehost:service_mgr -user sysdba -password XXX ¥
  action_backup -dbname some.fdb -bkp_file stdout >some.fbk
```

バックアップファイルからのリストア

fbsvcmgrを使ってリモートに置かれたバックアップファイルからデータベースをリストアするには、次のパターンでコマンドを入力します。

```
fbsvcmgr remotehost:service_mgr -user sysdba -password XXX ¥
  action_restore -dbname some.fdb -bkp_file stdin <some.fbk
```

> Note
>
> バックアップ実行時に“詳細表示”（-v[erify]）スイッチは使えません。これは、サーバーからクライアントへのデータチャンネルが、バックアップファイルからデータのブロックを転送するのに使われているためです。
>
> データベースのリストア時には制限がなく、詳細表示モードが使えます。

独自のユーティリティ・コードを書く

リモートバックアップ／リストアを独自のプログラムで実行したい場合は、サービスAPIを利用して下さい。

バックアップ

バックアップは非常に単純です—isc_info_svc_to_eofのタグを付したisc_service_query()への呼び出しを繰り返すことで返されたデータがバックアップファイルのイメージを表すストリームとなります。isc_service_query()呼び出しでisc_info_svc_to_eofタグを使い、バックアップファイル名としてサーバーに“stdout”を渡して下さい。

リストア

リストアはやや複雑になります。クライアントはisc_service_query()呼び出しでサーバーに新しいspbパラメータisc_info_svc_stdinを送信します。サービスは、stdinのデータが必要な場合、クエリの結果としてisc_info_svc_stdinを返します。これには4バイト値が続き、クライアントから受け取る用意のあるバイト数を表します。（ゼロ値は現在これ以上のデータが必要ないことを意味します）。

> Important
>
> クライアントはサーバーからリクエストされた以上のデータを送信してはいけません：エラー“Size of data is more than requested”がスローされます。

データは次のisc_service_query()呼び出しで、従来のisc_info_svc_lineタグの形式を使い、send_itemsブロックで送信されます：isc_info_svc_lineは2バイト長のデータです。次の部分

が必要になると、サーバーは逆にisc_service_query()からisc_info_svc_stdinにゼロ以外の値を返します。

> Tip
>
> リモートバックアップ用のサービスAPIの使い方のサンプルについては、fbsvcmgrのソースコードを参照して下さい。

オンラインでの検証サービスの実行

Vlad Khorsun

データベースの検証はオンディスク構造体の整合性に関する低レベルでのチェックを可能にし、小さな破損を修正することもできます。データベースの健康ををを保つためにDBAが定期的に検証を行うことは、特に重要なデータベースを扱う上で推奨される手続きです。

これにはデータベースへの排他的なアクセスが必要です：検証作業中はその他のアクセスは禁止されます。時には、特に巨大で複雑なデータベースの場合、ユーザーのアクセスを遮断することで大規模な停滞が起こることがあります。

オンライン検証は新しい機能です。これにより、整合性チェックを排他的アクセスなしで実行できるようになります。

オンライン検証でできること

- データベースの一部の（または全部の）ユーザーテーブルを検証する。

  システムテーブルは検証されません。

- 一部の（または全部の）インデックスを検証する。

ヘッダ＼PIP＼TIP＼ジェネレータの各ページのような、他のODSのチェックは実行されません。

オンライン検証中の保護

テーブル（および／または、そのインデックス）の検証作業中でも、ユーザーのアタッチメントはそのテーブルの読み取りを行うことができます。データの変更（INSERT¥UPDATE¥DELETE）については、検証の終了まで待機させられるか、ユーザートランザクションのロックタイムアウトに従ってロックタイムアウト・エラーが返されるか、いずれかとなります。

検証作業中のテーブルまたはそのインデックスのガベージコレクションは実行できません：

- バックグラウンドモード、協調モードでのガベージコレクションは単にこのテーブルをスキップします。
- スイープはエラーを発生させて停止します。

テーブルのチェックを開始すると、オンライン検証はデータの変更を防止する二つのロックを取得します：

- PR（読み取り保護）モードでのリレーション・ロック

・ （新機能）PW（書き込み保護）モードでのガベージコレクション・ロック

いずれのロックも、ユーザー指定のロックタイムアウトを使うことで取得されます。ロック・リクエストが失敗した場合はエラーが報告され、そのテーブルはスキップされます。

一度ロックが取得されると、テーブルとそのインデックスは、完全な検証を実行した場合と同じ方法で検証されます。これが完了して同じ手続きが次のテーブルへと移行すると、ロックは解除されます。

新しいサービスAPIアクション：isc_action_svc_validate

オンライン検証はFirebirdサービスとして実装されており、サービスAPIを通じてアクセスできます。そのため、これをgfixから起動することはできません。

この呼び出しは以下の要素を含みます：

```
アクション：
 isc_action_svc_validate

パラメータ：
 isc_spb_dbname ：
  データベースファイル名、文字列、必須

 isc_spb_val_tab_incl, isc_spb_val_tab_excl,
 isc_spb_val_idx_incl, isc_spb_val_idx_excl ：
  テーブル＼インデックス名のパターン、文字列、オプション

 isc_spb_val_lock_timeout ：
  ロックタイムアウト、整数、オプション

出力：
 オンライン検証プロセスの進行状況を示すテキストメッセージ
```

isc_action_svc_validateの対話的な使用

fbsvcmgrユーティリティは、この新しいサービスを完全にサポートしています。構文は次の通り：

```
fbsvcmgr [host:]service_mgr [user <...>] [password <...>]
 action_validate dbname <ファイル名>
 [val_tab_incl <パターン>]
 [val_tab_excl <パターン>]
 [val_idx_incl <パターン>]
 [val_idx_excl <パターン>]
 [val_lock_timeout <番号>]
```

このうち、

val_tab_incl	検証の実行に含めるテーブル名のパターン
val_tab_excl	検証の実行から除外するテーブル名のパターン
val_idx_incl	

	検証の実行に含めるインデックス名のパターン。デフォルトは%、すなわち全インデックス
val_idx_excl	検証の実行から除外するインデックス名のパターン
val_lock_timeout	ロックタイムアウト、検証するテーブルのロックの取得に使用される、秒単位、デフォルトは10秒、0は待機なし、-1は無期限待機

使用上の注意

- パターンは正規表現、SIMILAR TO式と同じルールで処理されます。
- データベースのダイアレクトに関わりなく、全てのパターンで大文字小文字は区別されます。
- テーブル用のパターンが省略された場合、全てのユーザーテーブルは検証されます。
- インデックス用のパターンが省略された場合、指定されたテーブルの全てのインデックスが検証されます。
- システムテーブルは検証されません。
- テーブルまたはインデックスのリストを指定するには：
  1. パイプ・キャラクタ'|'で名前を区切ります。
  2. スペースを入れないこと：TAB1 | TAB2は誤りです。
  3. コマンド・インタープリタを混乱させないように、ダブルクォートでリスト全体を囲います。

例

1. データベース'c:¥db.fdb'の、名前が'A'で始まる全てのテーブルを検証。インデックスは検証しない。ロックの待機は実行しない。

```
fbsvcmgr.exe service_mgr user SYSDBA password masterkey
  action_validate dbname c:¥db.fdb
  val_tab_incl A%
  val_idx_excl %
  val_lock_timeout 0
```

2. テーブルTAB1とTAB2と、その全てのインデックスを検証。ロック待機のタイムアウトは10秒（デフォルト）：

```
fbsvcmgr.exe service_mgr user SYSDBA password masterkey
  action_validate dbname c:¥db.fdb
  val_tab_incl "TAB1|TAB2"
```

3. val_XXXオプションのデフォルトの挙動：データベース'c:¥db.fdb'の全てのユーザーテーブルとそのインデックスを検証。ロック待機はデフォルトの10秒：

```
fbsvcmgr.exe service_mgr user SYSDBA password masterkey
               action_validate dbname c:¥db.fdb
```

他のサービスAPIの追加

Alex Peshkov

他に追加されたサービスAPIには以下のものが含まれます：

サービスAPIでのロールRDB$ADMINのマッピング

セキュリティ・データベースへのアクセスをリクエストする際に権限のあるOSユーザーのロールRDB$ADMINを有効または無効にするため、サービスパラメータブロック（SPB）に二つのタグアイテムが追加されました。

> Note
>
> この機能は、新しい-mappingスイッチによってgsecユーティリティに実装されました。コマンドライン・ユーティリティの章の関連する節の注意事項を参照して下さい。

タグアイテムisc_action_svc_set_mapping

security2.fdbにアクセスするためのサービス・リクエスト用に、指定されたOSユーザーに対しロールRDB$ADMINを有効にします。

タグアイテムisc_action_svc_drop_mapping

security2.fdbにアクセスするためのサービス・リクエスト用に、指定されたOSユーザーのロールRDB$ADMINを無効にします。

パラメータisc_spb_sec_admin

新しいパラメータisc_spb_sec_adminは、SYSDBAまたは他の十分な権限を持つユーザーがセキュリティ・データベース（security2.fdb）のロールRDB$ADMINを通常のユーザーに対して付与したり削除したりできるようにするために導入された新しいDDL構文のSPB実装です。通常のユーザーがセキュリティ・データベースでユーザーを作成、変更、削除するには、このロールでSYSDBAと同じ権限を手にする必要があります。

isc_spb_sec_adminはspb_long型で、値として0（REVOKE ADMIN ROLEを意味する）またはゼロ以外の数字（GRANT ADMIN ROLEを意味する）を取ります。

詳細は、データ定義言語の章のCREATE/ALTER/DROP USERの項を参照して下さい。

タグアイテムisc_spb_bkp_no_triggers

この新しいSPBタグは、データベースレベルやトランザクションレベルでのトリガがバックアップ・リストアの最中に起動するのを防ぐため、バージョン2.1でgbakユーティリティに導入された-nodbtriggersスイッチのサービスAPIとしての側面を表すものです。これは、isc_spb_bkp_ignore_limboなどのアイテムを含むオプション命令のセットisc_spb_optionsの中での使用が意図されています。

タグアイテムisc_spb_res_metadata_only

トラッカー・リファレンス CORE-3462

SPB中のリストアサービス・オプションにisc_spb_bkp_metadata_onlyタグを渡すと、metadata-onlyリストアが実行されます。Firebirdのfbsvcmgrを含む多くのツールでは、リストアのリクエストにバックアップ・オプションを指定することはできません。

（バージョン2.5.1）混乱を避けるため、単にisc_spb_bkp_metadata_onlyを再実装しただけのタグisc_spb_res_metadata_onlyがパブリックヘッダとfbsvcmgrに定数として追加されました。

nBackupのサポート

サービスAPIでnBackupアクションをサポートするために三つのものが追加されました。

バックアップとリストア

トラッカー・リファレンス CORE-1758

nBackupユーティリティは二つの論理グループの操作を実行します：データベースのロックとロック解除、そして、バックアップとリストアです。そのうち、ロック／ロック解除の操作にサービスアクションを提供する正当な理由はありません–SQL言語でALTER DATABASEのリクエストを使うことでリモートからのそれらのリクエストは可能です–一方で、バックアップ／リストア操作用のサービスAPIのインターフェースを正当化するのは簡単です。

バックアップとリストアはホストのステーションで実行する必要があります。それらにアクセスする唯一の方法は、nBackupを起動することでした。

以下の二つの新しいサービスアクションによって、サービスAPIを通じてnBackupによるバックアップとリストアをリクエストできるようになりました：

- isc_action_svc_nbak – 増分バックアップ
- isc_action_svc_nrest – 増分データベースリストア

パラメータ・アイテムは、以下の通り：

- isc_spb_nbk_level – バックアップレベル（整数）
- isc_spb_nbk_file – バックアップファイル名（文字列）
- isc_spb_nbk_no_triggers – データベーストリガを抑制するためのオプション

使用例

以下に、fbsvcmgrユーティリティで新しいパラメータの使う例をいくつか挙げています。便宜上、ロングインはすでに確立しているものと仮定しています。いずれの例もページ幅に合わせて改行していますが、実際には一行のコマンドです。

レベル0バックアップを作成：

```
fbsvcmgr service_mgr action_nbak dbname employee
  nbk_file e.nb0 nbk_level 0
```

レベル1バックアップを作成：

```
fbsvcmgr service_mgr action_nbak dbname employee
  nbk_file e.nb1 nbk_level 1
```

ファイルからのデータベースのリストア：

```
fbsvcmgr service_mgr action_nrest dbname e.fdb
  nbk_file e.nb0 nbk_file e.nb1
```

ダイレクトI/O機能のサポート

isc_spb_nbk_direct on|off

新しいタグにより、コマンドラインからnbackup -D on|offを呼び出すのと同じアクションが可能となり、ダイレクトI/Oの強制的なオン・オフが行えます。これは他のnbackup関連のタグと同様に、action_nbakでのみ有効です。

> Note
>
> この機能の使用法については、ユーティリティの章の新しいnbackupのスイッチについての注意事項を参照して下さい。

サービスAPIでのFIX_FSS_DATAおよびFIX_FSS_METADATAオプションが可能に

A. Peshkov

トラッカー・リファレンス CORE-2439

Firebird 2.1でgbak -restoreスイッチに実装されたFIX_FSS_DATAとFIX_FSS_METADATA関数はコアエンジンに実装され、サービスAPIのisc_action_svc_restore構造体に対応するタグ定数として公開されています。従って、開発者には、古いFirebirdデータベースから新しいオンディスク構造体への移行を自動化するアプリケーションを書く道が開かれています。

新しいSPBタグはisc_spb_res_fix_fss_dataとisc_spb_res_fix_fss_metadataです。

新しいトレースAPI

作成中の新しいトレースAPIは、トレースされる任意のイベント発生時にエンジンから呼び出される外部プラグインモジュールとして実装可能なフックのセットを提供します。これはまだドキュメント化されておらず、今後のサブリリースで変更されることもあります。

これについてのより詳しい情報は、管理機能の章のトレースのプラグイン機能の項を参照して下さい。

Chapter 5

予約語とその変更

> Note
>
> アスタリスク（*）は、Firebirdの文法によって予約されたキーワードまたはそれ以外にキーワードとして認識されるもののうち、標準SQLでは予約語となっていないものを示しています。

予約語の刷新

A. Peshkov

トラッカー・リファレンス CORE-2638

Firebird独自の予約語の数はかなり減りました。これで、他のデータベースからFirebirdに移行する際にキーワードの競合で苦労することが少なくなるはずです。SQL標準で予約語となっていないものは、可能な限り、Firebirdの文法でも予約語ではなくなりました。

まだ少数、SQL標準の予約語でないものがFirebirdの予約語として残されています。次のリストの通り：

```
ADD *          DB_KEY *      GDSCODE *       INDEX *
LONG *         PLAN *        POST_EVENT *    RETURNING_VALUES *
SQLCODE *      VARIABLE *    VIEW *
```

他の非標準のキーワードで、以前に予約語だったものは全て、任意の目的で利用できるようになりました。

新しい予約語

```
SIMILAR                SQLSTATE
```

非予約語として追加されたキーワード

```
AUTONOMOUS *           BIN_NOT *          CALLER *
CHAR_TO_UUID *         COMMON *           DATA
FIRSTNAME *            GRANTED            LASTNAME *
MIDDLENAME *           MAPPING *          OS_NAME *
SOURCE *               TWO_PHASE *        UUID_TO_CHAR *
```

Chapter 6

設定パラメータの追加と変更

firebird.conf への以下の変更・追加に注意して下さい：

AuditTraceConfigFile

V. Khorsun

このパラメータは、Firebirdエンジンが次のシステム監査トレースに必要なイベントのリストを決定するために読むべきファイルの名前と配置を示します。デフォルトではこのパラメータの値は空で、どのシステム監査トレースも設定されていないことを示しています。

> Note
>
> Firebirdのルートディレクトリ内にあるテンプレートファイルfbtrace.conf には、監査トレース設定ファイルを記述するためのフォーマット、ルール、構文と、利用可能なイベントの完全なリストが含まれています。

詳細は、新しい管理機能に関する章、[トレースと監査サービス]の節のシステム監査トレースのセッションの項を参照して下さい。

ファイルシステムのキャッシュの利用に影響するパラメータ

Firebirdとファイルシステムのキャッシュとの相互作用の仕方を設定するパラメータは二つあります。

FileSystemCacheSize

N. Samofatov

Firebird 2.5で新たに導入されたFileSystemCacheSizeは、64bit Windows XPやService Pack 1以降のMicrosoft Server 2003ホストでWindowsファイルシステム・キャッシュが使用するRAMの最大量を制御します。

バージョン2.5の当初のリリースでは、これによるPOSIXホスト・システムへの影響はありません。

このパラメータの設定値は、OSが利用可能なトータルの物理RAMのパーセンテージを表す整数となります。有効な値となるためには設定値が10～95（%）の範囲に収まらなければなりませんが、明示的に0と設定することでホストのキャッシング設定を適用することもできます。これ以外の数値は、デフォルトの30（%）と見なされます。

firebird.conf の設定をしても、サーバーのプロセスが再起動されるまで、変更は有効になりません。

Windowsのセキュリティ権限

OSのユーザーは、ファイルシステムのキャッシュ設定の調整にSeIncreaseQuotaPrivilegeが必要となります。この権利は管理者権限を持つユーザーとサービスアカウントに組み込まれており、Firebirdサービスアカウントにも、Windowsインストーラーによって明示的に付与されます。

異なる状況、例えば、エンベデッド版や、Firebirdサーバーをアプリケーションとして稼働する場合、またはカスタムサービスによるインストールの場合、ユーザーはこの権限を持っていないことがあります。このような設定ミスがあっても結果的にプロセスのスタートアップが失敗することはありません：firebird.log に警告が書き込まれ、スタートアップは単純にホストOSの設定により進行します。

FileSystemCacheThreshold

V. Khorsun

このパラメータはバージョン2.1でMaxFileSystemCacheとして導入されていますが、名称が変更されましたので、アップグレードする方への注意喚起のため、ここで改めて説明します。

FileSystemCacheThresholdは、Firebirdがページキャッシュとファイルシステム・キャッシュとの重複を許可するか否かを決める敷居値を設定します。このパラメータがゼロより大きい任意の（整数の）値に設定されている場合の効果は、ページキャッシュの現在のデフォルトサイズによって異なります：デフォルトのページキャッシュ（ページ単位）がMaxFileSystemCacheの値（ページ単位）よりも小さい場合にはファイルシステムのキャッシングは有効、それ以外の場合は無効となります。

Note

このことは、ページキャッシュのバッファサイズが、DefaultDBCachePagesによって暗黙に設定されている場合にも、データベースのヘッダ属性として明示的に設定されている場合にも、いずれの場合にも適用されます。また、これは全てのプラットフォームに適用されます。

従って、

- ファイルシステムのキャッシングを常に無効にするには、FileSystemCacheThresholdの値にゼロを設定します。
- ファイルシステムのキャッシングを常に有効にするには、FileSystemCacheThresholdの値に、データベースのページキャッシュのサイズを超える充分大きな整数値を設定します。ただし、この値による効果がページキャッシュのサイズに対するその後の変更によって影響を受けることに留意して下さい。

Important

- FileSystemCacheThresholdのデフォルトの設定値は65536ページです。つまり、ファイルシステムのキャッシングは有効となっています。
- 特定のデータベースに対して設定されたキャッシュサイズがFileSystemCacheThresholdの値より大きい場合、FileSystemCacheSizeの設定値（上記）はそのデータベースに影響を与えません。

MaxFileSystemCache

Firebird 2.1で導入されたMaxFileSystemCacheは、もう有効なパラメータではありません。

ConnectionTimeout

D. Yemanov

高負荷のWindowsシステムでは、サーバーがxnet_response_eventを設定するのを待つ間にクライアントがタイムアウトし、ローカル接続（XNET）に失敗することがありました。この問題を解決するため、ConnectionTimeoutパラメータが、TCP/IPに加えてXNET接続にも作用するよう拡張されました。

> Note
>
> このパラメータ向けにドキュメント化された注意事項は、ネットワーク・トランスポートには今なお適用可能ですが、XNETのプロトコルには適用できません。

Authentication

A. Peshkov

Windowsサーバー・プラットフォームでは、バージョン2.1以来、デフォルト以外のサーバー認証モードの設定が必要な場合にAuthenticationが使われてきました。

バージョン2.5でのモード設定はこれと同じです。すなわち、

- trustedは、Windowsの“信頼された認証”を利用します。適切な条件下では、Windows上で最もセキュアな認証方法となります。
- nativeは、従来のFirebirdサーバーの認証モードを設定します。セキュリティ・データベースに登録されたユーザー名とパスワードを用いたログインをユーザーに要求します。
- mixedは、trustedとnativeの両方を利用します。

バージョン2.5での変更点

- バージョン2.5でも各モードに変更はありませんが、'mixed'または'trusted'モードに設定された場合、デフォルトでは、Windowsドメイン管理者へのSYSDBA権限の付与が自動で行なわれなくなりました。ODS　11.2データベースのロールRDB$ADMINやドメイン管理者へのSYSDBA権限の自動マッピングに関する管理機能の章の注意を参照して下さい。
- デフォルトの設定がmixedからnativeに変更されました。“信頼されたユーザー認証”を有効にする（mixedまたはtrusted）には、このパラメータの明示的な設定が必要になりました。

  トラッカー・リファレンス CORE-2376

MaxUserTraceLogSize

V. Khorsun

サービスAPIの新たなトレース機能を用いたユーザートレース・セッションによって生成されるテンポラリ・ファイル全体のサイズの最大値を設定します。デフォルト値は10MBです。一時的に出力を格納するファイル全体のサイズの最大値の上げ下げには、このパラメータを使用します。

OldSetClauseSemantics

D. Yemanov

Firebird　2.5より前のバージョンでは、UPDATE文のSET句によりユーザーが指定した順序に従ってカラムが割り当てられ、割り当てられた新しいカラムの値がその後の割り当てに即時に使用されていました。これは、カラムの開始値がSQL文の実行の間に維持されることを求めているSQL標準に準拠していませんでした。

現在は、SET句中のどの割り当てに対しても、元のカラムの値だけがアクセス可能となっています。

必要な場合は、OldSetClauseSemanticsの設定によって、このOldSetClauseSemanticsを介して、従来の挙動に戻すことができます。値を1に設定すると従来のもの、0（デフォルト）に設定すると修正後の挙動を示します。

> Warning
>
> - このパラメータを変更すると、サーバー上の全てのデータベースに影響が出ます。
> - このパラメータは後方互換性のための一時的な措置として提供されたものです。Firebirdの将来のバージョンでは非推奨となる予定です。

クラシックサーバー、スーパークラシックサーバーでのRemoteAuxPort

Dmitry Yemanov

トラッカー・エントリー：CORE-2263

クラシックサーバーとスーパークラシックサーバーで、RemoteAuxPortで指定された単一のポートからイベントの通知を受けるよう設定できるようになりました。スーパーサーバーではバージョン1.5から可能となっています。

待望されたこの改善により、アプリケーションが、使用するサーバー・モデルに関わらず、ファイアウォール越しであれ、セキュアなトンネル越しであれ、インターネット経由でデータベースに接続してイベントを利用することが可能になりました。

RemoteBindAddressでのホスト名の使用

Alex Peshkov

トラッカー・エントリー：CORE-2094

RemoteBindAddressの設定されたFirebirdサーバーが稼働しているホストのホスト名を使用できるようになりました。以前はIPアドレスしか使用できませんでした。

> Important
>
> RemoteBindAddressはユーザー接続をホスト・サーバー上の特定のNICカードに“固定的に割り当てる”ために使えます。指定したホスト名が一つ以上のIPアドレスに同時に関連づけられることのないよう注意して下さい！特に、ホスト・ステーション自体を含む全てのステーションで、etc/hosts ファイルをチェックして下さい。

RemoteFileOpenAbility

Nickolay Samofatov

トラッカー・エントリー：CORE-2263

POSIX版で、NFSデバイス上のデータベースへのアクセスを長時間許可する機能と合わせて、この極端なオプションをWindowsで利用可能にするため、また、ネットワーク上で共有されたデータベースを開けるようにするため、Red Soft社由来のコードが組み込まれました。

これは、異なるプラットフォーム間で機能の整合性を維持するための試みです。アーキテクチャの変更に関わるものではなく、実際にこれを使用したからといって従来より安全になると考えられるというようなものでもありません。しかし、明確で、充分にテストされた、安全な目的のために、マッピングされた配置場所にデータベースのシャドウイングを行ない、外部のファイルシステム上のデータベースに接続することを可能にします。例えば、ときおり単発的なセキュリティタスクを実行するためディスクレスのワークステーションに差し込まれたUSBデバイス上にある鍵付きデータベースなど。

> Warning
>
> これをアクティベートする前に、firebird.conf内の注意をお読み下さい！

Chapter 7

管理機能

Firebirdの管理機能にいくつか改善が施されました。多くの皆さんがこれを歓迎するでしょう。

新しいシステムロールRDB$ADMIN

Alex Peshkov

新たな定義済みシステムロールRDB$ADMINが追加され、SYSDBA権限を他のユーザーに譲渡できるようになりました。どのユーザーでも、特定のデータベースでこのロールが付与された場合、その指定されたロールRDB$ADMINを持つデータベースにアタッチする際にSYSDBAと同様の権限を手にします。

これを割り当てるには、SYSDBAはそのデータベースにログインしている必要がありますが、ロールRDB$ADMINをユーザーに付与する方法は、他のロールをユーザーに付与する仕方と同様です。このロールを付与されたユーザーが、そのデータベースでの“スーパーユーザー”権限にアクセスするためには、ログイン時にこれを明記する必要があります。

> Important
>
> ユーザーがアタッチする際に、ユーザー・データベース・ロールがDPB（接続パラメータ）に渡されている場合、これをRDB$ADMINと置き換えることはできません。つまりこの場合、SYSDBA権限を得ることはできません。

次に挙げる例では、SYSDBA権限がユーザー（User1とAdmins¥ADMINS）に譲渡されています。このうち二番目のユーザーは、“信頼された認証”を介してアクセスが有効とされたWindowsシステムユーザーとして典型的なものです：

```
GRANT RDB$ADMIN TO User1;
GRANT RDB$ADMIN TO "Admins¥ADMINS";
```

複数のデータベースとスーパーユーザー

RDB$ADMINロールが割り当てられている場合でも通常のユーザーがSYSDBAになるわけではないということは理解しておくべきです。正確には、あるデータベースでユーザーにこのロールが付与されている時、そのユーザーはデータベース内のオブジェクトに対してSYSDBAと同等の権限を与えられている、ということです。

- 同じユーザーに複数のデータベースでのスーパーユーザー権限が必要な場合、そのユーザーに対するロールRDB$ADMINは、それぞれのデータベースについて明示的に付与されなければなりません。

- 一つのデータベースに対して複数のユーザーにスーパーユーザー権限を持たせたい場合、それぞれのユーザーにロールRDB$ADMINを付与する必要があります。
- ユーザーがあるデータベースのロールRDB$ADMINに属している場合、これを他のユーザーに付与することができます。
- WITH ADMIN OPTION（このロールを他のユーザーに付与する権限のために）や、WITH GRANT OPTION（オブジェクトの所有者であることなしに他のユーザーにそれらのオブジェクトに対するパーミッションを付与する権限のために）を指定する必要はありません。ADMINやGRANTオプションは暗に含まれています。

システム “スーパーユーザー”

POSIXホストではrootユーザーが常にSYSDBA権限を持っていますが、このことは、Firebird 2.1まで、Windowsのドメイン管理者には当てはまりませんでした。バージョン2.1で設定パラメータAuthenticationが導入されたことにより、Windowsドメイン管理者としてログインしているユーザーは、“信頼されたユーザー認証”を通して、自動的にSYSDBA権限でサーバーにアクセスできるようになりました。POSIX版ではこの仕組みに変更はありませんが、Windows版では、ロールRDB$ADMINの導入により、WindowsのAdministratorsがSYSDBA権限を得る方法が変更されました。

> Windowsの“信頼されたユーザー認証”が、デフォルトでは利用できなくなりました！
>
> デフォルトでは、firebird.conf のAuthenticationパラメータはnativeに設定されています。“信頼されたユーザー認証”を有効にするには、trustedかmixedへと明示的に設定しなければなりません。

WindowsのAdministrators向けのグローバル管理者権限

“信頼されたユーザー認証”が設定されているFirebirdサーバーで、信頼されたドメイン管理者がSYSDBAアクセス権限を得るには、ドメイン管理者はロールRDB$ADMINに属している必要があります。通常のユーザー向けとして上で解説した手動による方法を使い、データベースごとにそれぞれ特定の管理者にロールRDB$ADMINを付与します。

しかし、SYSDBAがサーバーを設定することで、Windows管理者が任意のデータベースにログインする時にロールRDB$ADMINが自動的にマッピングされるようにする方法があり、これによって、POSIXシステムでroot権限を持つユーザーがSYSDBA権限と関連付けられているのと同様の状況にすることができます。新しいALTER ROLE文はこの目的のため（だけ）に用いられます。

ALTER ROLE文

“信頼されたユーザー認証”が有効になっているWindowsサーバーで、ロールRDB$ADMINを管理者に自動的にマッピングするようにデータベースを設定するには、SYSDBAは、任意のデータベースにログインし、以下のSQL文を発行します：

```
ALTER ROLE RDB$ADMIN
  SET AUTO ADMIN MAPPING;
```

デフォルト設定に戻して管理者が自動的にはSYSDBA権限を得ないようにするには、次のSQL文を発行します：

```
ALTER ROLE RDB$ADMIN
  DROP AUTO ADMIN MAPPING;
```

サービスAPIタグアイテム

同様の効果はサービスAPIに用意された二つのタグアイテムでサポートされます：自動マッピングを有効にするisc_action_svc_set_mappingとそれを無効にするisc_action_svc_drop_mappingです。

これらのタグはfbsvcmgrユーティリティでサポートされています。

ユーザー管理用に拡大するRDB$ADMINの範囲

新しいDDLコマンドALTER USERにより、“通常の”ユーザー（普通のFirebirdユーザーやPOSIX上での非rootユーザー、または“信頼されたユーザー認証”が有効になっているWindowsシステム上で信頼されたユーザー）が、任意のデータベースにログインしている間、パスワードおよび／または個人名要素を変更できるようになります。スーパーユーザーは、同じコマンドを使ってユーザーの作成と削除を行うこともできます。この新しいコマンドの詳細については、データ定義言語の章のCREATE/ALTER/DROP USERの項を参照して下さい。

security2.fdbはODS 11.2データベースとして（アップグレードが必要な場合もあります）作成されるので、ここにも定義済みロールRDB$ADMINがあります。どのユーザーも-SYSDBAでさえも-セキュリティ・データベースにログインすることができないため、SYSDBAまたはスーパーユーザーが、ユーザーの作成・削除する権限が必要な通常のユーザーに対してsecurity2.fdbのロールRDB$ADMINを適用できるという代替手段が提供されています。これには三つの方法があり、それぞれ同等の効果を持ちます。

1. CREATE USERまたはALTER USER文でオプションパラメータGRANT ADMIN ROLEを使う。

> 注意
>
> この場合のGRANT ADMIN ROLEやREVOKE ADMIN ROLEがGRANT／REVOKE文ではなく、CREATE USERやALTER USER文に対する3キーワードパラメータであることに注意して下さい。’ADMIN’という名のシステムロールは存在しません。
>
> データベースのロールRDB$ADMINを手にしている任意のユーザーは暗黙のうちに拡張権限WITH ADMIN OPTIONとWITH GRANT OPTIONを手にしています。

例

セキュリティ・データベースでユーザーalexにロールRDB$ADMINを付与する場合：

```
ALTER USER alex GRANT ADMIN ROLE;
```

セキュリティ・データベースでユーザーalexからロールRDB$ADMINを取り消す場合：

```
ALTER USER alex REVOKE ADMIN ROLE;
```

すべてのデータベースでユーザーalexを削除しその権限も削除する場合：

```
DROP USER alex;
```

2. gsecユーティリティに-adminスイッチを付けて使用します。このスイッチは引数を一つ取ります：YESの場合、ロールRDB$ADMINがユーザーに適用されます。NOの場合、このロールは取り消されます。詳細は、ユーティリティの章のgsecの節のロールRDB$ADMINを通常のユーザーに付与するの項を参照して下さい。

3. SPBパラメータisc_spb_sec_adminを使います。これはsecurity2.fdbで通常のユーザーにロールRDB$ADMINをSPB接続を介して割り当てる実装です。詳細については、Firebird APIとODSの変更の章のParameter isc_spb_sec_adminの項に記載されています。

   fbsvmgrユーティリティもこのパラメータの使用をサポートしています。

> Tip
>
> Firebird 2.5では、サーバーに複数のセキュリティ・データベースを作成することはできません。バージョン3.0以降は、データベースごとに個別のセキュリティ・データベースを持てるようになる予定です。現状では、サーバー上の任意のデータベース（employee.fdbも含め）に接続するごとに唯一のsecurity2.fdbが更新されることになります。
>
> 将来的に、これらのリクエストは影響を受ける各セキュリティ・データベースに対応するデータベースから送信することが必須となります。

トレースと監査サービス

Vlad Khorsun

バージョン2.5での新しいトレースと監査の機能は、当初、Nickolay Samofatov氏によってコントリビュートされたTraceAPIから開発されました。これはFirebirdコードをベースとする商業製品Red Soft Database用に彼が開発したものです。

機能の概要

新しいトレースと監査の機能は、SQL文の実行、接続、切断など、エンジン内で実行されるさまざまなイベントのログを取り、照合して、対応するパフォーマンス特性のリアルタイムな分析に使うことができます。

トレースは、トレース・セッションの文脈で起こります。それぞれのトレース・セッションが固有の設定、ステート、出力を持ちます。

Firebirdエンジンはトレース可能なイベントの固定リストを持ちます。システム監査トレースとユーザートレースという、二つの異なる種類のトレースを実行できます。エンジンがどのようにセッション用のイベントのリストを作成するかは、どのトレースが要求されているかによります。

Tip

全てのトレースセッションに一意なセッションIDが割り当てられます。任意のトレースセッションが始まると、サービスマネージャはこのIDをメッセージとして出力します。

```
Trace session ID nnnn started
```

もちろん、ここでの nnnn はIDです。

システム監査のセッション

システム監査のセッションはエンジン自ら開始します。セッションがどのイベントに“関心を持っている”か決定するため、エンジンはセッションを作成しに行く際にトレース設定ファイルの内容を読みます。

firebird.conf の新しいパラメータAuditTraceConfigFileは、ファイルの名前と配置を指示します。進行中のシステム監査トレースは最大で一つとなります。デフォルトでは、このパラメータの値は空で、どのシステム監査トレースも設定されていないことを示しています。

設定ファイルにはトレースされるイベントのリストが含まれ、各イベント用のトレースログの配置を指示しています。異なるイベントのセットがそれぞれ異なるデータベースのログを取り、ログファイルを切り離すことができるというような柔軟さは十分にあります。Firebirdのルートディレクトリ内にあるテンプレートファイルfbtrace.conf には、監査トレース設定ファイルを記述するためのフォーマット、ルール、構文とともに、利用可能なイベントの完全なリストが含まれています。

fbtrace.confファイルに関するTip

このファイルには各エントリの目的と構文を説明する大量のコメント付きテキストが含まれます。サブリリースを重ねるごとに、トレース性能の改善のため時々新たなイベントと機能が加えられますので、注意して下さい。

例えば、最近のプレリリースでの拡張では、サービスイベントを、名前を使ってトレースしたり、include、excludeフィルタをを使ってターゲットを絞ることができるようになっています。

他の例として、パス名マッチングアルゴリズムの改善により、設定ファイル内の文字列をプラットフォームのキャラクタ・セットに従って翻訳したり、UTF-8の文字列に基づいて、Windowsでは大文字小文字を区別しない、POSIXでは大文字小文字を区別する、というようなファイル名の取り扱い等に関するプラットフォーム固有のルールが適用できるようなっています。（トラッカー・リファレンス CORE-2404、A. dos Santos Fernandes）。

バージョン2.5.2で施された改善では、トレースセッションの設定により、手動・自動スイープ両方の詳細情報についてログを取ることができるようになりました。テンプレートファイルから“SWEEP_”オプションを見つけて下さい。

ユーザートレース・セッション

ユーザートレース・セッションはサービスAPIに対する新しい呼び出しを使ってユーザーが管理します。このための新しいサービス関数は五つあります：

- 開始：isc_action_svc_trace_start
- 停止：isc_action_svc_trace_stop
- 中断：isc_action_svc_trace_suspend
- 再開：isc_action_svc_trace_resume
- 既知の全トレースセッションの一覧: isc_action_svc_trace_list

サービスAPI呼び出しの構文については、Firebird APIとODSの変更の章のアプリケーション用の新トレース機能の項で議論されています。

ユーザートレース・セッションの働き

ユーザーアプリケーションがトレースセッションを開始すると、セッション名（オプション）とセッション設定（必須）が付けられます。セッション設定は、出力の配置に関する行は別として、fbtrace.conf をテンプレートとするルールと構文に準拠したテキストファイルです。このファイルはFirebirdのルートディレクトリにあります。

> Note
>
> これらのファイルはサーバーには残りません。ユーザートレースの要求に応えるためテキストの格納と取得にふさわしい仕組みを設計するのは、アプリケーション開発者の仕事になります。
>
> 例えば、コマンドラインユーティリティfbsvcmgrは、セーブファイル・パラメータtrc_cfgをサポートします。

ユーザーセッションの出力は、それぞれ1MBのテンポラリ・ファイルの一群に格納されます。アプリケーションによって一度完全に読み込まれたファイルは、自動的に削除されます。デフォルトでは、ファイルサイズの総計は最大で10MBに制限されています。この値はfirebird.conf 内のMaxUserTraceLogSizeを使って小さくも大きくもできます。

アプリケーションによって一度ユーザートレース・セッションのサービスが開始されると、アプリケーションは、isc_service_query()への呼び出しを使ってその出力を読み込む必要があります。このサービスは、アプリケーションが読み込めるより速く出力を生成することがあります。出力のサイズの総計がMaxUserTraceLogSizeの上限に達すると、エンジンは自動的にトレースセッションを中断します。アプリケーションがファイル（出力の1MB分）を読み込み終えるとそのファイルは削除され、容量に余裕が生まれてエンジンはトレースセッションを自動的に再開します。

アプリケーションは、そのトレースセッションの停止を決めると、単にサービスからのデタッチをリクエストします。別の方法として、isc_action_svc_trace_*関数を使うことで、アプリケーションが自由にトレースセッションを中断、再開、停止することもできます。

> Tip
>
> アタッチメントのキャラクタ・セット名は対応するいずれのトレースログのレコードにも含まれ、user:roleとprotocol:portの間に置かれています。例えば、
>
> A.FDB (ATT_36, SYSDBA:NONE, WIN1251, TCPv4:127.0.0.1)
>
> DPBにキャラクタ・セットが指定されていない場合、アタッチメントのキャラクタ・セットのスロットでトレースログのレコードにNONEが書き込まれます。
>
> (CORE-3008)

トレースのセッションを管理できるのは誰か？

どのユーザーでもトレースセッションを始動し、管理することができます。通常のユーザーは、自身の接続についてのトレースだけをリクエストすることができ、他のユーザーが開始したトレースセッションを管理することはできません。管理者はどのトレースセッションでも管理できます。

異常な終了

全てのFirebirdプロセスが停止されると、どのユーザートレース・セッションも保存されません。つまり、スーパーサーバーまたはスパークラシックサーバーのプロセスがシャットダウンされると、resumeを待機していたものを含め、進行中だったユーザートレース・セッションは完全に停止され、resumeもそれらを再開することができません。

> Note
>
> もちろん、クラシックサーバーでは、それぞれの接続がそれぞれ専用のサーバー・インスタンスを含むことから、この状況は当てはまりません。そもそも、クラシックサーバー・インスタンスを“シャットダウン”するという事態はありません。どのサービス・インスタンスもそれを引き起こした接続より長生きすることはできません。

ユーザートレースのサンプル設定テキスト

以下のサンプルは、ユーザートレース・セッションの設定テキストを記述するための基準を提供します。

a. 接続12345に含まれる全てのSQL文のプリペア、解放、実行をトレースします。

```
<database mydatabase.fdb>
 enabled                 true
 connection_id           12345
 log_statement_prepare   true
 log_statement_free      true
 log_statement_start     true
 log_statement_finish    true
 time_threshold          0
</database>
```

b. 実行されたINSERT、UPDATE、DELETE文と、プロシージャとトリガへのネストされた呼び出しのログを取りつつ、データベースmydatabase.fdbへの所定のユーザーの全ての接続をトレースし、対応するPLANとパフォーマンスの統計を示します。

```
<database mydatabase.fdb>
 enabled                 true
 include_filter          %(INSERT|UPDATE|DELETE)%
 log_statement_finish    true
 log_procedure_finish    true
 log_trigger_finish      true
 print_plan              true
 print_perf              true
 time_threshold          0
</database>
```

コマンドラインからユーザートレースサービスへのリクエスト

トレースサービスと対話的に動作する新しいコマンドラインユーティリティfbtracemgrが追加されました。これはスイッチとパラメータに独自の構文を持ちます。実例も含めた詳細はユーティリティの章で議論されています。

さらに、サービスユーティリティfbsvcmgrは、コマンドラインからのサービスリクエストの送信に使うことができます。以下に例を示します。

a. fbtrace.conf という名の設定ファイルを使って“My trace”と名づけたユーザートレースを開始し、画面上でその出力を読む。

```
fbsvcmgr service_mgr action_trace_start trc_name "My trace" trc_cfg fbtrace.conf
```

このトレースセッションを停止するには、fbsvcmgrコンソールプロンプトでCtrl+Cを押します。（下記(e)も参照）。

b. トレースセッションを一覧する：

```
fbsvcmgr service_mgr action_trace_list
```

c. ID 1のトレースセッションを中断する

```
fbsvcmgr service_mgr action_trace_suspend trc_id 1
```

d. ID 1のトレースセッションを再開する

```
fbsvcmgr service_mgr action_trace_resume trc_id 1
```

e. ID 1のトレースセッションを停止する

```
fbsvcmgr service_mgr action_trace_stop trc_id 1
```

> Tip
>
> 他のコンソールでセッションの一覧を得て（b参照）、関心あるセッションのIDを探し、現在のコンソールでそれを使ってそのセッションを停止します。

RDB$ADMINのユーザー管理スコープの昇格

トラッカー・リファレンス CORE-2588

Windowsでは、異なるWindowsセッションの複数のエンジン・インスタンスがグローバル名前空間でのtraceの起動を許可してしまうと、トレースツールは共有メモリの競合を引き起こします。その

ため、トレーススコープは、現在のWindowsセッションからアクセス可能なプロセスだけに限定されています。

使用例

一般に三つの場合が考えられます：

1. エンジン活動の常時監査

   システム監査トレースを使います。管理者はトレース設定ファイルを作成・編集し、firebird.conf内にあるAuditTraceConfigFileの設定を介してその名前を決め、Firebirdを再起動します。その後、管理者がこのセッションを中断、再開、停止するのにFirebirdを再起動する必要はありません。

   > Important
   >
   > 監査設定の変更をエンジンに通知するには、Firebirdを再起動する必要があります。

2. 一部（または全部）のデータベースの一部（または全部）の活動に対するオンデマンドな対話的インタラクティブ・トレース

   アプリケーション（fbtracemgrユーティリティでも可）がユーザートレース・セッションを開始し、その出力を読み込み、トレースしたイベントをリアルタイムで画面に表示します。ユーザーはそのトレースを中断・再開することができ、最終的にそれを停止することもできます。

3. 重要な期間（数時間または終日ということも）のエンジン活動を収集し後で分析する

   アプリケーションがユーザートレース・セッションを開始し、トレースの出力を定期的に読み込み、一つまたは複数のファイルに保存します。そのセッションの停止はその同じアプリケーションか別のアプリケーションによって手動で行う必要があります。複数のトレースセッションが起動している場合、関心あるセッションを特定するため一覧の取得が求められることがあります。

トレースのプラグイン機能

新しいトレースAPIは外部プラグインモジュールとして実装可能なフックのセットを提供します。これらは、トレースされる任意のイベントの発生時にエンジンから呼び出されます。こうしたイベントで適宜ログを取る役割はプラグインが担うことになります。

このトレースAPIはFirebird　2.5に含まれおり、使用することもできますが、今後のサブリリースでの変更が予定されておりますので、正式なものではなく、unstableと見て下さい。

実装された“標準の”トレースプラグインfbtrace.dll（.so） は、Firebird 2.5がインストールされたディレクトリ内の¥pluginsフォルダの中にあります。

モニタリングの改善

Dmitry Yemanov

Firebird　2.5では、バージョン2.1で導入された“MON$”データベースモニタリング機能が拡張され、コンテキスト変数とODS　11.2以降のデータベースでのメモリ使用に関するデータを通知する新

たなテーブルが備わりました。また、これらのデータベースでは、クライアントの接続をMON$ストラクチャーを通じて他の接続から終了できるようになりました。

通常のユーザー向けの拡張されたアクセス

もともとの設計では、特別な権限を持たないデータベースのユーザーは自身のCURRENT_CONNECTIONに関するモニタリング情報しか見ることができませんでした。現在は、同じユーザー名で認証された任意のアタッチメントの情報をリクエストできるようになっています。

トラッカー・リファレンス CORE-2233

> 注意
>
> 1. 異なるエンドユーザーのためにミドルウェア層で同じユーザー名による同時・複数回のログインを必要とするアプリケーションのアーキテクチャでは、エンドユーザーにモニタリング機能を公開する点について、パフォーマンスやプライバシーへの影響が考慮されるべきです。
> 2. 同様の拡張はバージョン2.1.2で実装されました。

ODS 11.2データベースの新しいMON$メタデータ

> Note
>
> ODS 11.1のメタデータについては、バージョン2.1用のドキュメントを参照して下さい。

> MON$メタデータ用キャラクタ・セットの変更
>
> ファイル仕様に関するMON$テーブル内のカラム定義に使われるシステムドメインRDB$FILE_NAME2が、CHARACTER SET NONEからCHARACTER SET UNICODE_FSSへと変更されました。現在影響を受けるカラムはMON$DATABASE_NAMEとMON$ATTACHMENT_NAMEとMON$REMOTE_PROCESSです。この変更で影響を受けるデータは、バージョン2.5で更新されたfilespecの取り扱い、および他のDPBのキャラクタパラメータ・アイテムと一致することになります。
>
> （トラッカー・エントリー：CORE-2551、A. dos Santos Fernandes）

MON$MEMORY_USAGE（現在のメモリ使用）

- MON$STAT_ID（統計 ID）
- MON$STAT_GROUP（統計グループ）
  0: データベース
  1: アタッチメント
  2: トランザクション
  3: SQL文
  4: 呼び出し
- MON$MEMORY_USED（現在使用中のバイト数）
  エンジンによって実行されるプールからの高レベルでのメモリ割り当て。
  Can be useful for tracing memory leaks and for investigating unusual memory consumption and the attachments, procedures, etc. that might be responsible for it.

メモリリークのトレース、また、それが原因となりうる異常なメモリ消費やアタッチメント、プロシー
- MON$MEMORY_ALLOCATED（OSレベルで現在割り当てられているバイト数）
  Firebirdメモリマネージャによって実行された低レベルでのメモリ割り当て。
  これらはOSによって実際に割り当てられているバイトであり、そのため、
  物理メモリの消費を監視することができます。

> Note
>
> すべてのレコードがゼロ以外の値を持つわけではありません。一般に、MON$DATABASEとメモリ・バインド・オブジェクトだけはゼロ以外の“割り当てられた”値を示します。小さな割り当てはこのレベルでは行われません。その代わり、データベースのメモリプールにリダイレクトされます。

- MON$MAX_MEMORY_USED（このオブジェクトで使用される最大のバイト数）
- MON$MAX_MEMORY_ALLOCATED（このオブジェクトによってOSから割り当てられた最大のバイト数）

MON$CONTEXT_VARIABLES（既知のコンテキスト変数）

- MON$ATTACHMENT_ID（アタッチメント ID）
  セッションレベルのコンテキスト変数用の有効なIDのみを含みます。
  トランザクションレベルの変数はこのフィールドにNULLをセットします。
- MON$TRANSACTION_ID（トランザクション ID）
  トランザクションレベルのコンテキスト変数用の有効なIDのみを含みます。
  セッションレベルの変数はこのフィールドにNULLをセットします。
- MON$VARIABLE_NAME（コンテキスト変数名）
- MON$VARIABLE_VALUE（コンテキスト変数値）

> MON$STATEMENTSとMON$STATEでのメモリ使用
>
> MON$STATEMENTSとMON$STATEでのメモリ使用の統計は、実際のCPU消費を表現しています。
>
> トラッカー・リファレンス CORE-1583

使用上の注意

例

SQL文のメモリ使用ランキング“トップ10”：

```
SELECT FIRST 10
  STMT.MON$ATTACHMENT_ID,
  STMT.MON$SQL_TEXT,
  MEM.MON$MEMORY_USED
FROM MON$MEMORY_USAGE MEM
  NATURAL JOIN MON$STATEMENTS STMT
  ORDER BY MEM.MON$MEMORY_USED DESC
```

現在の接続用の全てのセッションレベルのコンテキスト変数を列挙：

```
SELECT
  VAR.MON$VARIABLE_NAME,
  VAR.MON$VARIABLE_VALUE
FROM MON$CONTEXT_VARIABLES VAR
  WHERE VAR.MON$ATTACHMENT_ID = CURRENT_CONNECTION
```

クライアントの終了

MON$ストラクチャーは、設計上、読み取り専用となっています。そのため、それらに対するユーザーDMLオペレーションは禁止されています。しかし、MON$STATEMENTSとMON$ATTACHMENTSテーブルのレコードの削除（だけ）を許可する仕組みが組み込まれています。このメカニズムの効果として、それぞれ、SQL文実行のキャンセルや、ODS　　11.2データベース向けには、クライアントセッションの終了が可能になります。

指定された接続の現在の全活動をキャンセルする：

```
DELETE FROM MON$STATEMENTS
  WHERE MON$ATTACHMENT_ID = 32
```

“自分”以外の全てのクライアントの接続を切断する：

```
DELETE FROM MON$ATTACHMENTS
  WHERE MON$ATTACHMENT_ID <> CURRENT_CONNECTION
```

Note

1. ・ SQL文をキャンセルしようとしても、クライアントに現在実行中のSQL文がない場合は、無効なオペレーション（“no-op”）となります。
   ・ キャンセルと同時に、execute／fetchのAPI呼び出しはエラーコードisc_cancelledを返します。
   ・ その後のオペレーションは許可されます。
2. ・ 終了中の接続でアクティブなトランザクションはすぐにキャンセルされ、ロールバックされます。
   ・ 終了されると、クライアントセッションはエラーコードisc_att_shutdownを受け取ります。
   ・ その後にこの接続ハンドルを使おうとすると、ネットワークのread／writeエラーを引き起こします。

Chapter 8

セキュリティの強化

Windowsプラットフォーム

SYSDBAを自動マッピングしない（Windows）

バージョン2.1では、Windowsの管理者グループのメンバーは、デフォルトでSYSDBAにマッピングされていました。バージョン2.5以降では、SYSDBAの自動マッピングは次の新たなSQLコマンドを使うことでデータベースごとに制御されます。

```
ALTER ROLE RDB$ADMIN SET/DROP AUTO ADMIN MAPPING
```

> Note
>
> ロールRDB$ADMINの全容については、管理機能の章の新しいシステムロールRDB$ADMINの項を参照して下さい。

Chapter 9

データ定義言語（DDL）

バージョン2.5では、DDLに対して重要な追加と拡張が施されました。

Quick Links

- CREATE/ALTER/DROP USER
- ビュー変更の構文
- CREATE VIEW用のSSP拡張
- 計算項目用のALTERの仕組み
- GRANTおよびREVOKE用GRANTED BYの拡張
- ALTER ROLE
- REVOKE ALL
- データベースのデフォルトのコレーション属性
- ALTER CHARACTER SET
- CREATE DATABASE用の引数DIFFERENCE FILE

このリリースではFirebird 3を目指したアーキテクチャ変更が強調されていますが、多数の改善や拡張も実装されています。その多くはトラッカーでの機能要求に応えたものです。

> 一般的警告
>
> バージョン2.5以上では、カラムのデータ型の変更が可能になっています。そのカラムがストアドプロシージャやトリガで参照されていたとしても、例外がスローされることはありません。ただし、コンパイル済みのPSQLはBLOB内でバイナリ表現（“BLR”）として静的に格納されているため、オリジナルのBLRはバックアップやリストアを行なっても残っています。BLRは静的であるため、データ型を変更しても更新されません。
>
> このことは、テーブルやそのビューから影響を受けるカラムを参照しているBLRが、それらのカラムを参照するTYPE OF構文を使って任意の変数を宣言している場合、警告なしに破壊される可能性が高いことを意味します。最善の場合、BLRは“注意が必要”のフラグを付けられますが、テストでは、どんな条件でもフラグ付けが行なわれるわけではないことが分かっています。
>
> つまり、フィールドがコンパイル済みPSQLに何らかの依存関係を持っていたとしても、あなたがそのフィールドの型を変更するのをエンジンはもう止めてくれなくなった、ということです。変更された状況を管理するため、影響を受けるプロシージャやトリガを特定したり、変更に対応してそれらを再コンパイルしたりといった対応は、あなた自身の問題となります。

クラシックサーバーでのプロシージャ定義変更の可視性

Dmitry Yemanov

トラッカー・リファレンス CORE-2052

一件の変更により、クラシックサーバーで、変更されたストアドプロシージャに関する他のクライアントの接続からの可視性の問題への対処が行われました。変更中のトランザクションがコミットを完了するとすぐに、その変更がサーバー全体で確認できるようになりました。

CREATE/ALTER/DROP USER

Alex Peshkov

トラッカー・リファレンス CORE-696

バージョン2.5で、Firebirdはついに、通常のデータベースへのログイン時にSQL文を発行することでサーバーのユーザーアカウントを管理できる構文を手に入れました。

CREATE USERとALTER USER文はいずれもパラメータGRANT ADMIN ROLEとREVOKE ADMIN ROLEをを持ち、SYSDBA権限を持つユーザーがセキュリティ・データベースのロールRDB$ADMINを通常のユーザーに付与することができます。この使い方の解説とロールRDB$ADMINの全容については、管理機能の章の新しいシステムロールRDB$ADMINの項を参照して下さい。

構文パターン

SYSDBAまたはSYSDBA権限を持つユーザーは、現在のデータベースとセキュリティ・データベースのいずれにも新しいユーザーを追加できます：

```
CREATE USER <ユーザー名> {PASSWORD 'パスワード'}
  [FIRSTNAME 'ファーストネーム']
  [MIDDLENAME 'ミドルネーム']
  [LASTNAME 'ラストネーム']
  [GRANT ADMIN ROLE];
```

> Note
>
> 新しいユーザーを作成する際はPASSWORD句が必要です。これは新しいユーザーの初期パスワードとなります。ユーザーが後からこれを変更するにはALTER USERを使います。

SYSDBAまたはSYSDBA権限を持つユーザーは、現在のデータベースとセキュリティ・データベースのいずれでも、既存のユーザーのパスワードや固有名詞属性を変更できます。権限がないユーザーは、同じ文を使うことで自分自身の属性のみ変更できます：

```
ALTER USER <ユーザー名>
  [PASSWORD 'パスワード']
  [FIRSTNAME 'ファーストネーム']
  [MIDDLENAME 'ミドルネーム']
  [LASTNAME 'ラストネーム']
  [{GRANT | REVOKE} ADMIN ROLE];
```

> Note
>
> 少なくとも一つのオプションパラメータを与える必要があります。
>
> ALTER USERでは<ユーザー名>の変更はできません。別の<ユーザー名>に変更したい場合は、もとのユーザーを削除し新たなユーザーを作成しなくてはなりません。

SYSDBAまたはSYSDBA権限を持つユーザーは、現在のデータベースとセキュリティ・データベースのいずれでも、ユーザーを削除することができます：

```
DROP USER <ユーザー名>;
```

制限事項

現在のデータベースとセキュリティー・データベースのいずれでも、CREATE USERやDROP USER文とGRANT | REVOKE ADMIN ROLEを使えるのはSYSDBAまたはロールRDB$ADMINを手にしているユーザーだけです。

通常のユーザーは自身のパスワードや固有名詞の要素をALTERすることができます。別のユーザーの修正を試みても失敗します。

例

SYSDBAまたは同等の権限を持つユーザーは、現在のデータベースでもセキュリティ・データベースでも、以下のことができます：

```
CREATE USER alex PASSWORD 'test';

ALTER USER alex FIRSTNAME 'Alex' LASTNAME 'Peshkov';

ALTER USER alex PASSWORD 'IdQfA';
```

ビューを変更する構文

Adriano dos Santos Fernandes

従来は、ビュー定義を変更するためには、ビュー定義をオフラインのどこかに保存してからビューを削除し、その上で再作成するする必要がありました。これは、特に依存関係の問題がある場合には非常に煩わしいものでした。そこで、バージョン2.5では、ALTER VIEWとCREATE OR ALTER VIEWの構文が導入されました。

トラッカー・リファレンス CORE-770 および CORE-1640

ALTER VIEW

ALTER VIEWはビュー定義の変更を可能にします。ビューの古いバージョンと全ての依存関係を再作成（削除して作成）する必要はありません。

CREATE OR ALTER VIEW

CREATE OR ALTER VIEWを使うと、ビューが存在する場合はビュー定義が変更されます（ALTER VIEWと同様）。存在しない場合はビューが作成されます。

構文パターン

```
{create [ or alter ] | alter } view <ビュー名>
  [ ( <フィールドリスト> ) ]
as <select文>{
```

例

```
create table users (
    id integer,
    name varchar(20),
    passwd varchar(20)
);

create view v_users as
    select name from users;

alter view v_users (id, name) as
    select id, name from users;
```

CREATE VIEWの拡張

CREATE VIEWに以下のような拡張が追加されました。

FROM句にストアドプロシージャを指定

Adriano dos Santos Fernandes

トラッカー・リファレンス CORE-886.

ビュー定義のFROM句に選択型ストアドプロシージャを指定できるようになりました。

例

```
    create view a_view as
    select * from a_procedure(current_date);
```

カラムリストを含まないUNION句を用いたビューの作成

Dmitry Yemanov

トラッカー・リファレンス CORE-1402

セットがUNION句で定義されている場合、CREATE VIEWからカラムリストを省略できるようになりました。

例

```
recreate view V1 as
  select d.rdb$relation_id from rdb$database d
  union all
  select d.rdb$relation_id from rdb$database d

recreate view V2 as
  select d.rdb$relation_id as q from rdb$database d
  union all
  select d.rdb$relation_id as w from rdb$database d
```

推定されたカラム名

Adriano dos Santos Fernandes

トラッカー・リファレンス CORE-2424

CREATE VIEWが、GROUP BY句を含むビューまたは派生テーブルのカラム名を推定できるようになりました。

例

```
create view V as
  select d.rdb$relation_id from rdb$database d
    group by d.rdb$relation_id

create view V as
   select a from (select 1 a from rdb$database);
```

計算項目に対するALTERの仕組み

Adriano dos Santos Fernandes

トラッカー・リファレンス CORE-1454

COMPUTED BY 〈式〉と定義されたカラムをALTER TABLE...ALTER COLUMN構文を使って変更できるようになりました。この機能はカラム定義の〈式〉要素を異なる式に変更するためだけに使われます。計算項目を非計算項目に、またその逆の変換を行うことはできません。

構文パターン

```
alter table 〈テーブル名〉
  alter 〈計算項目名〉
  [type 〈データ型〉]
  COMPUTED BY (〈式〉);
```

例

```
create table test (
  n integer,
  dn computed by (n * 2)
);
commit;
alter table test
  alter dn computed by (n + n);
```

SQLパーミッションの拡張

Alex Peshkov

SQLパーミッション（権限）のエリアに以下の拡張が実装されました。

GRANTED BY句

GRANT文とREVOKE文にオプションでGRANTED　BY（またはAS）句を含めることで、CURRENT_USER（デフォルト）以外のユーザーを権限付与者にできるようになりました。

構文パターン

```
grant <権限> to <オブジェクト>
  [ { granted by | as } [ user ] <ユーザー名> ]
--
revoke <権限> from <オブジェクト>
  [ { granted by | as } [ user ] <ユーザー名> ]g
```

> Tip
>
> ```
> { granted by | as }
> ```
>
> GRANTED BYとASは同じ働きをします。SQL標準で推奨される形式はGRANTED BYです。われわれは他のサーバー（例えばInformix）との互換性のためにASをサポートしています。

例

SYSDBAとしてログイン：

```
create role r1; -- SYSDBAがこのロールを所持しています
  /* 次に、SYSDBAはuser1に対し、他のユーザーに同じロールを
     付与する権限を付けてこのロールを付与します */
grant r1 to user1 with admin option;
  /* SYSDBAがGRANTED BY句を使ってuser1の
     ADMIN OPTIONを行使します */
grant r1 to public granted by user1;
```

この効果はisqlでも得られます：

```
  SQL>show grant;
  /* このデータベースに対するを付与します */
  GRANT R1 TO PUBLIC GRANTED BY USER1
  GRANT R1 TO USER1 WITH ADMIN OPTION
  SQL>
```

ALTER ROLE

トラッカー・リファレンス CORE-1660

新しいALTER　　ROLE文の機能は、WindowsのAdministratorsに対する“信頼された認証”を通じたSYSDBAパーミッション割り当ての制御に特化しています。現在これに他の用途はありません。

> Note
>
> ALTER　ROLEの使い方の解説とRDB$ADMINの全容については、管理機能の章の新しいシステムロールRDB$ADMINの項を参照して下さい。

REVOKE ALL

トラッカー・リファレンス CORE-2113

ユーザーがセキュリティ・データベースまたは他の認証ソース（OSのACLなど）から削除される時、データベース内で関連するSQL権限のクリーンアップは手動で行う必要があります。この拡張により、特定のユーザーやロールの全ての権限を一度に取り消すことができるようになりました。

構文パターン

```
REVOKE ALL ON ALL FROM { <ユーザーリスト> | <ロールリスト> }
```

例

SYSDBAとしてログイン：

```
# gsec -del guest
# isql employee
fbs bin # ./isql employee
Database:  employee
SQL> REVOKE ALL ON ALL FROM USER guest;
SQL>
```

データベースのデフォルトのコレーション属性

Adriano dos Santos Fernandes

トラッカー・リファレンス CORE-1737 および CORE-1803

ODS　11.2以上のデータベースは、デフォルトのキャラクタ・セットに関連付けた、デフォルトのCOLLATION属性を持つことができ、異なるコレーション用のCOLLATE句が指定されていない限り、全てのテキスト・カラム、ドメイン、変数の定義を同じコレーションオーダーで作成できます。

COLLATION句はオプションです。これが省略された場合は、キャラクタ・セット用のデフォルトのコレーションオーダーが使われます。

> Tip
>
> ALTER CHARACTER SET用の構文が導入されたことで、データベースで使われるキャラクタ・セット用のデフォルトのコレーションオーダーも変更可能になっていることに注意して下さい。

構文パターン

```
create database <ファイル名>
    [ page_size <ページサイズ> ]
```

```
    [ length = <長さ> ]
    [ user <ユーザー名> ]
    [ password <ユーザーパスワード> ]
    [ set names <charset名> ]
    [ default character set <charset名>
      [ [ collation <コレーション名> ] ]
    [ difference file <ファイル名> ]
```

> Note
>
> CREATE DATABASE用のパラメータDIFFERENCE FILEは新しいものではありません。これは、バージョン2.0で、nBackupユーティリティに関連して静かに導入されており、ドキュメント化も三年間でひっそりと行われていました。詳細は、この章の終わりにあるCREATE DATABASEの進化をご覧ください。

例

```
create database 'test.fdb'
    default character set win1252
    collation win_ptbr;
```

ALTER CHARACTER SET コマンド

Adriano dos Santos Fernandes

トラッカー・リファレンス CORE-1803

このバージョンでは新しい構文が導入され、データベースにキャラクタ・セット用のデフォルトのコレーションを設定できるようになりました。

デフォルトのコレーションが使われるのは、（カラムまたはドメイン定義中のCHARACTER SET句を通じて明示的に、またはデータベースのデフォルトのキャラクタ・セット属性を通じて暗に）指定されたキャラクタ・セットを持つテーブルのカラムが作成され、そして、いかなるCOLLATION句も指定されていない場合となります。

> Note
>
> 文字列定数も、接続キャラクタ・セットのデフォルトのコレーションを使います。
>
> 既存のカラムのキャラクタ・セットとコレーションはALTER CHARACTER SETの変更によっては影響を受けません。

構文パターン

```
ALTER CHARACTER SET <charset名>
    SET DEFAULT COLLATION <コレーション名>
```

例

```
create database 'people.fdb'
  default character set win1252;
```

```
alter character set win1252
  set default collation win_ptbr;

create table person (
  id integer,
  name varchar(50)  /* データベースのデフォルトの
                       キャラクタ・セットとwin1252の
                       デフォルトのコレーションを使います */
);

insert into person
  values (1, 'adriano');
insert into person
  values (2, 'ADRIANO');

/* win_ptbrは大文字小文字を区別しないので、
   両方のレコードの取得します */
select * from person where name like 'A%';
```

> Tip
>
> また別の改善により、システムテーブルRDB$CHARACTER_SETS内のRDB$DEFAULT_COLLATE_NAMEの現在の値がバックアップ／リストアのサイクルの中で維持されるようになりました。

CREATE DATABASEの進化

nBackupの状態の登録と変更を行うために導入されたデータベースのヘッダ属性に対するDDLのサポートは、Firebird 2.0以来、進化を続けています。nBackupのユーザーには、完全なnBackupの間にメインのデータベースとは別のファイルに“増分”データの格納を開始、終了するALTER DATABASE文はお馴染みとなっています。

nBackup用の増分ファイルの命名

ALTER DATABASEはまたもう一つの引数を持ち、これによって増分データを格納するファイルに使われる名前を設定することができます。Paul Vinkenoog氏によるnBackupの素晴らしいマニュアルを引用すれば：

デフォルトでは、増分ファイルはデータベース自身と同じディレクトリ内に置かれます。このファイルには、データベースファイルと同じ名前に.deltaを加えた名前が付けられます。通常これを変更する理由はありませんが、必要なら変更できます。-ただし、それをnbackupから行うことはできません。

任意のクライアントからデータベースに接続したら、あなた自身でSQL文を入力し、コマンドを打って下さい：

```
alter database
  add difference file 'パスとファイル名'
```

カスタムの増分ファイル指定は、データベース内で保持されます；これはシステムテーブルRDB$FILESに格納されます。

デフォルトの挙動に戻すには、次のSQL文を発行して下さい：

```
alter database
  drop difference file
```

さらに詳しく知りたい方は、バージョン2.0のリリースノートまたはnBackupのマニュアルをお読み下さい。

CREATE DATABASEの引数DIFFERENCE FILE

C. Valderrama

Firebird 2.0で、増分ファイルのカスタム名を規定する構文は、CREATE DATABASEの追加のオプション引数として登場しました。データベースのデフォルトCOLLATION属性の項の上記の構文パターンの中でその配置を確認することができます。ALTER DATABASEの場合と同様に、引数用のキーワードはDIFFERENCE FILEであり、引数は有効なファイル指定となります。ALTER DATABASE BEGIN BACKUPが呼び出される場合、または、同等のnBackupのシェルコマンドが呼び出される場合はいつでも、作成される増分ファイルのカスタム名を指定できます。

使い方の例

```
 ]..¥bin> isql -user sysdba -pass masterkey

Use CONNECT or CREATE DATABASE to specify a database
SQL> create database 'ticks' difference file 'jaguar';
SQL> shell dir jaguar;
Volume in drive F is Firebird
Volume Serial Number is BCD9-4211

Directory of ..¥bin

File Not Found
```

これは正しく、ただファイル名が定義されただけです。次にこれを使います：

```
SQL> alter database begin backup;
SQL> shell dir jaguar;
Volume in drive F is Firebird
Volume Serial Number is BCD9-4211

Directory of ..¥bin

10-11-2009  00:59             8.192 jaguar
               1 File(s)          8.192 bytes
               0 Dir(s)  16.617.979.904 bytes free

SQL> alter database end backup;
SQL> shell dir jaguar;
Volume in drive F is Firebird
Volume Serial Number is BCD9-4211

Directory of ..¥bin

SQL> drop database;
SQL> ^Z
```

引数がファイル名なので、シングルクォートで囲います。ダブルクォートは無効となります：SQL文は失敗し、煩雑なエラーメッセージが返されます。

```
]..¥bin> isql -user sysdba -pass masterke

Use CONNECT or CREATE DATABASE to specify a database
SQL> create database 'ticks' DIFFERENCE FILE 'jaguar';
SQL> alter database add difference file 'leopard';

Statement failed, SQLCODE = -607
unsuccessful metadata update
-Difference file is already defined
```

このメッセージは正しいです。呼び出しALTER DATABASE END BACKUPによって増分が削除されたとしても、差分ファイルの名前はずっと格納されています。なお、存在して良いのは一つだけです。

```
SQL> alter database drop difference file;
SQL> alter database begin backup;
```

これは何も破壊したことになりません。エンジンがこの事態を救い、デフォルトの仕組みを使って増分を作成するためです。

```
SQL> alter database add difference file 'leopard';
SQL> alter database begin backup;
SQL> alter database drop difference file;

Statement failed, SQLCODE = -607
unsuccessful metadata update
-Cannot change difference file name while database is in backup mode
```

これは正しい検証です。

```
SQL> alter database end backup;
SQL> drop database;
SQL> ^Z
```

Chapter 10

データ操作言語（DML）

この章では、Firebird 2.5でSQLのデータ操作言語サブセットに施された追加と改善を扱います。

クイックリンク

SIMILAR TOを用いた正規表現検索のサポート

Adriano dos Santos Fernandes

トラッカー・リファレンス CORE-769

新たなSIMILAR TO句の導入により、正規表現がサポートされました。この演算子には、与えられたSQL標準の正規表現が引数の文字列にマッチするか検証する機能があります。これは、WHERE句やCHECK制約、PSQLのIF() test文など、ブール式を受け付ける任意の文脈で有効となります。

構文パターン

```
<similar句> ::=
 <値> [ NOT ] SIMILAR TO <similarパターン> [ ESCAPE <エスケープキャラクタ> ]

<similarパターン> ::= <キャラクタ値表現: 正規表現>

<正規表現> ::=
      <正規項>
    | <正規表現> <バーティカルバー> <正規項>

<正規項> ::=
      <正規因子>
    | <正規項> <正規因子>

<正規因子> ::=
      <正規一次子>
    | <正規一次子> <アスタリスク>
    | <正規一次子> <正符号>
    | <正規一次子> <疑問符>
    | <正規一次子> <繰り返し因子>
```

```
〈繰り返し因子〉::=
    〈左波括弧〉〈下限値〉[ 〈上限指定〉 ]〈右波括弧〉

〈上限指定〉::=〈コンマ〉[ 〈上限値〉 ]

〈下限値〉::=〈符号なし整数〉

〈上限値〉::=〈符号なし整数〉

〈正規一次子〉::=
      〈キャラクタ指定子〉
    | 〈パーセント記号〉
    | 〈正規キャラクタ・セット〉
    | 〈left paren〉〈正規表現〉〈右括弧〉

〈キャラクタ指定子〉::=
      〈エスケープされないキャラクタ〉
    | 〈エスケープされたキャラクタ〉

〈正規キャラクタ・セット〉::=
      〈アンダースコア〉
    | 〈左角括弧〉〈キャラクタ列挙〉... 〈右角括弧〉
    | 〈左角括弧〉〈サーカムフレックス〉〈キャラクタ列挙〉... 〈右角括弧〉
    | 〈左角括弧〉〈包含するキャラクタの列挙〉... 〈サーカムフレックス〉〈除外するキャラクタの列挙 exclude〉...
        〈右角括弧〉

〈包含するキャラクタの列挙〉::=〈キャラクタ列挙〉

〈除外するキャラクタの列挙〉::=〈キャラクタ列挙〉

〈キャラクタ列挙〉::=
      〈キャラクタ指定子〉
    | 〈キャラクタ指定子〉〈負符号〉〈キャラクタ指定子〉
    | 〈左角括弧〉〈コロン〉〈キャラクタクラス識別子〉〈コロン〉〈右角括弧〉

〈キャラクタ指定子〉::=
      〈エスケープされないキャラクタ〉
    | 〈エスケープされたキャラクタ〉

〈キャラクタクラス識別子〉::=
      ALPHA
    | UPPER
    | LOWER
    | DIGIT
    | SPACE
    | WHITESPACE
    | ALNUM
```

Note

1. 〈エスケープされないキャラクタ〉とは、〈左角括弧〉、〈右角括弧〉、〈左括弧〉、〈右括弧〉、〈バーティカルバー〉、〈サーカムフレックス〉、〈負符号〉、〈正符号〉、〈アスタリスク〉、〈アンダースコア〉、〈パーセント記号〉、〈疑問符〉、〈左波括弧〉、〈エスケープキャラクタ〉を除く任意のキャラクタです。

2. 〈エスケープされたキャラクタ〉とは、〈左角括弧〉、〈右角括弧〉、〈左括弧〉、〈右括弧〉、〈バーティカルバー〉、〈サーカムフレックス〉、〈負符号〉、〈正符号〉、〈アスタリスク〉,〈アンダースコア〉、〈パーセント記号〉、〈疑問符〉、〈左波括弧〉、〈エスケープキャラクタ〉のいずれか一つが続く〈エスケープキャラクタ〉です。

Table 10.1. キャラクタクラス識別子

識別子	説明	注意
ALPHA	単純なラテン文字の全てのキャラクタ（a-z, A-Z）	アクセントを無視したコレーションを使う場合はアクセント付きラテン文字を含む
UPPER	単純なラテン文字で大文字の全てのキャラクタ（A-Z）	大文字小文字を区別しないコレーションを使う場合は小文字を含む
LOWER	単純なラテン文字で小文字の全てのキャラクタ（a-z）	大文字小文字を区別しないコレーションを使う場合は大文字を含む
DIGIT	数字の全てのキャラクタ（0-9）	
SPACE	スペースキャラクタである全てのキャラクタ（ASCII 32）	
WHITESPACE	ボワイトスペースである全てのキャラクタ（垂直タブ（9）、改行（10）、水平タブ（11）、キャリッジリターン（13）、改頁（12）、スペース（32））	
ALNUM	単純なラテン文字（ALPHA）または数字（DIGIT）である全てのキャラクタ	

使用ガイド

<正規表現>または<正規項>にマッチする文字列に true を返します：

〈正規表現〉〈バーティカルバー〉〈正規項〉

```
'ab' SIMILAR TO 'ab|cd|efg'   -- true
'efg' SIMILAR TO 'ab|cd|efg'  -- true
'a' SIMILAR TO 'ab|cd|efg'    -- false
```

<正規一次子>のゼロ回以上の繰り返しにマッチします：〈正規一次子〉〈アスタリスク〉

```
'' SIMILAR TO 'a*'     -- true
'a' SIMILAR TO 'a*'    -- true
'aaa' SIMILAR TO 'a*'  -- true
```

<r正規一次子>の一回以上の繰り返しにマッチします：〈正規一次子〉〈正符号〉

```
'' SIMILAR TO 'a+'     -- false
'a' SIMILAR TO 'a+'    -- true
'aaa' SIMILAR TO 'a+'  -- true
```

ゼロ個または一個の<正規一次子>にマッチします：〈正規一次子〉〈疑問符〉

```
'' SIMILAR TO 'a?'     -- true
'a' SIMILAR TO 'a?'    -- true
'aaa' SIMILAR TO 'a?'  -- false
```

〈正規一次子〉のきっかり〈下限値〉回の繰り返しにマッチします：〈正規一次子〉〈左括波括弧〉〈下限値〉〈右波括弧〉

```
'' SIMILAR TO 'a{2}'     -- false
'a' SIMILAR TO 'a{2}'    -- false
'aa' SIMILAR TO 'a{2}'   -- true
'aaa' SIMILAR TO 'a{2}'  -- false
```

〈正規一次子〉の〈下限値〉以上の繰り返しにマッチします：〈正規一次子〉〈左波括弧〉〈下限値〉〈コンマ〉〈右波括弧〉

```
'' SIMILAR TO 'a{2,}'     -- false
'a' SIMILAR TO 'a{2,}'    -- false
'aa' SIMILAR TO 'a{2,}'   -- true
'aaa' SIMILAR TO 'a{2,}'  -- true
```

〈正規一次子〉の〈下限値〉回以上〈上限値〉回以下の繰り返しにマッチします：〈正規一次子〉〈左波括弧〉〈下限値〉〈コンマ〉〈上限値〉〈右波括弧〉

```
'' SIMILAR TO 'a{2,4}'       -- false
'a' SIMILAR TO 'a{2,4}'      -- false
'aa' SIMILAR TO 'a{2,4}'     -- true
'aaa' SIMILAR TO 'a{2,4}'    -- true
'aaaa' SIMILAR TO 'a{2,4}'   -- true
'aaaaa' SIMILAR TO 'a{2,4}'  -- false
```

（空ではない）任意のキャラクタにマッチします：〈アンダースコア〉

```
'' SIMILAR TO '_'    -- false
'a' SIMILAR TO '_'   -- true
'1' SIMILAR TO '_'   -- true
'a1' SIMILAR TO '_'  -- false
```

任意の長さの文字列（空の文字列を含む）にマッチします：〈パーセント記号〉

```
'' SIMILAR TO '%'         -- true
'az' SIMILAR TO 'a%z'     -- true
'a123z' SIMILAR TO 'a%z'  -- true
'azx' SIMILAR TO 'a%z'    -- false
```

〈正規表現〉を部分式としてグループ化し、一つの〈正規一次子〉として使います：〈左括弧〉〈正規表現〉〈右括弧〉

```
'ab' SIMILAR TO '(ab){2}'    -- false
```

```
    'aabb' SIMILAR TO '(ab){2}'  -- false
    'abab' SIMILAR TO '(ab){2}'  -- true
```

<キャラクタ列挙>中の一つのキャラクタにマッチします：<左角括弧> <キャラクタ列挙>... <右角括弧>

```
    'b' SIMILAR TO '[abc]'    -- true
    'd' SIMILAR TO '[abc]'    -- false
    '9' SIMILAR TO '[0-9]'    -- true
    '9' SIMILAR TO '[0-8]'    -- false
```

<キャラクタ列挙>中のものを除くキャラクタにマッチします：<左角括弧> <サーカムフレックス> <キャラクタ列挙>... <右角括弧>

```
    'b' SIMILAR TO '[^abc]'    -- false
    'd' SIMILAR TO '[^abc]'    -- true
```

<除外するキャラクタの列挙>中のキャラクタを除く<包含するキャラクタの列挙>中の一つのキャラクタにマッチします：<左角括弧> <包含するキャラクタの列挙>... <サーカムフレックス> <除外するキャラクタの列挙>...

```
    '3' SIMILAR TO '[[:DIGIT:]^3]'    -- false
    '4' SIMILAR TO '[[:DIGIT:]^3]'    -- true
```

<キャラクタクラス識別子>に含まれる一つのキャラクタにマッチします。上掲のキャラクタクラス識別子の一覧表を参照して下さい。<サーカムフレックス>と一緒に使い、ロジックを反転させることもできます（上記）：<左角括弧> <コロン> <キャラクタクラス識別子> <コロン> <右角括弧>

```
    '4' SIMILAR TO '[[:DIGIT:]]'  -- true
    'a' SIMILAR TO '[[:DIGIT:]]'  -- false
    '4' SIMILAR TO '[^[:DIGIT:]]' -- false
    'a' SIMILAR TO '[^[:DIGIT:]]' -- true
```

例

```
create table department (
  number numeric(3) not null,
  name varchar(25) not null,
  phone varchar(14)
  check (phone similar to '¥([0-9]{3}¥) [0-9]{3}¥-[0-9]{4}' escape '¥')
);

insert into department
  values ('000', 'Corporate Headquarters', '(408) 555-1234');
insert into department
  values ('100', 'Sales and Marketing', '(415) 555-1234');
insert into department
  values ('140', 'Field Office: Canada', '(416) 677-1000');

insert into department
  values ('600', 'Engineering', '(408) 555-123'); -- 制約違反をチェック
```

```
select * from department
  where phone not similar to '¥([0-9]{3}¥) 555¥-%' escape '¥';
```

16進数リテラルのサポート

Bill Oliver
Adriano dos Santos Fernandes

トラッカー・リファレンス CORE-1760

16進数値とバイナリ文字列リテラルのサポートが導入されました。

構文パターン

```
<16進数リテラル> ::=
    { 0x | 0X } <hexit> [ <hexit>... ]

<バイナリ文字列リテラル> ::=
    { x | X } <クオート> [ { <hexit> <hexit> }... ] <クオート>

<数字> ::=
    0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<hexit> ::=
    <数字> | A | B | C | D | E | F | a | b | c | d | e | f
```

16進数リテラル

- 文字列中の<hexit>の数字は16を超えることはできません。
- <hexit>の数字が8を超えた場合、定数データ型は符号付BIGINTとなります。8以下の場合のデータ型は符号付INTEGERです。

> **Tip**
>
> つまり、0xF0000000は-268435456であり、また0x0F0000000は4026531840である、ということです。

バイナリ文字列リテラル

結果として生じる文字列は、CHAR(n/2) CHARACTER SET OCTETSとして定義されます。nは<hexit>の数字です。

例

```
select 0x10, cast('0x0F0000000' as bigint)
  from rdb$database;
select x'deadbeef'
  from rdb$database;
```

GEN_UUID()関数に対する重要な変更

Adriano dos Santos Fernandes

Firebird 2.5.2より前のバージョンでは、組み込み関数GEN_UUID()は完全にランダムな文字列を返しており、RFC-4122（UUID仕様）非対応となっていました。2.5.2以降、GEN_UUID()はUUIDバージョン4に準拠した文字列を返します。一部のbitは予約されており、それ以外はランダムとなります。

> Note
>
> UUIDに準拠した文字列フォーマットは、次のようなものです。
>
> XXXXXXXX-XXXX-4XXX-YXXX-XXXXXXXXXXXX
>
> 4は固定（バージョンを示す）で、Yは8、9、AまたはBです。

トラッカー・アイテム CORE-3238参照。

新しいUUID変換関数

Adriano dos Santos Fernandes

トラッカー・リファレンス CORE-1656 および CORE-1682

新しい組み込み関数UUID_TO_CHARとCHAR_TO_UUIDにより、UUID CHAR(16) OCTETS文字列の形式とRFC4122-compliantの形式との相互の変換が可能になりました。

> ビッグエンディアンのサーバーに関する注意点
>
> ビッグエンディアンのサーバー上のFirebird 2.5と2.5.1で、CHAR_TO_UUIDとUUID_TO_CHARは正しく動作しませんでした。変換の際にバイトやキャラクタが入れ替わり、誤った位置に置かれてしまうという不具合がありました。このバグはバージョン2.5.2以上では修正されています：トラッカー・アイテム CORE-3887を参照して下さい。ただし、このことは、バージョン2.5.2以降のCHAR_TO_UUIDとUUID_TO_CHARが、同じ入力パラメータに対して、以前のバージョンとは異なる値を返すということを意味します。

CHAR_TO_UUID()

CHAR_TO_UUID()関数はUUID（XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX）に対するCHAR(32) ASCII表現をストレージに最適化されたCHAR(16) OCTETS表現に変換します。

構文モデル

```
CHAR_TO_UUID( <文字列> )
```

例

```
select char_to_uuid('93519227-8D50-4E47-81AA-8F6678C096A1')
```

```
  from rdb$database;
```

UUID_TO_CHAR()

UUID_TO_CHAR()関数は、CHAR(16)　OCTETS　UUID（GEN_UUID()関数で返されるもの）をCHAR(32) ASCII表現（XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX）に変換します。

構文モデル

```
UUID_TO_CHAR( <文字列> )
```

例

```
select uuid_to_char(gen_uuid())
  from rdb$database;
```

SOME_COL = ? OR ? IS NULL句

Adriano dos Santos Fernandes

トラッカー・リファレンス CORE-2298

特に多くのDelphiプログラマからの要望を受け、カラムとパラメータが同値であるか、また、パラメータに渡された値がNULLかどうかの両方について“OR”テストを行うことができる演算子のサポートが実装されました。この構造体は、一つのクエリをプリペアしてフィルタを用いた結果のセットをリクエストし、同じクエリでフィルタを用いない別の結果のセットをリクエストするというような状況を避けるための手段としてしばしば要望されていました。

Delphiなどのプログラミング・インターフェースはパラメータにクライアントサイドのオブジェクト名を適用するため、それらのユーザーは、DSQLエンジンが以下のような使い方を認めてくれるような機能を求めていました：

```
  WHERE col1 = :param1 OR :param1 IS NULL
```

APIレベルでは、言語インターフェースはこのクエリを次のように翻訳します。

```
  WHERE col1 = ? OR ? IS NULL
```

これには二つの問題がありました：

1. プログラマはパラメータ:param1を、二つの参照を持つ単一の変数のように扱いますが、APIにはそれができません：これは二つのパラメータで与えられます
2. 第二のパラメータのデータ型は未知のものとなり、プログラムがこれに割り当てる方法はありません。

この問題の解決に必要だったのは、条件句“? is NULL”を扱える新しいデータ型を導入し、このようなリクエストを受け取った時に正しい処理を行えるようにFirebirdに仕込むことでした。

この実装は、次のように動作します。第一の問題の解決のため、リクエストは二つのパラメータ（われわれのDelphiの例では）を与える必要があります：

```
WHERE col1 = :param1 OR :param2 IS NULL
```

- “param1”がNULLでない場合、言語インターフェースは第一のパラメータに正しい値を割り当て、XSQLVAR.sqlindをNOT　NULLに設定する必要があります。XSQLVAR.sqldataはNULLのままとします。
- “param2”がNULLの場合、言語インターフェースは両方のパラメータのXSQLVAR.sqlindをNULLに設定する必要があります。XSQLVAR.sqldataはNULLのままとします。

言い換えると、 ? IS NULL中のパラメータ（ ? ）について：

- XSQLVAR.sqlindは、パラメータのNULL/NON-NULLの状態に応じて設定される必要があります。これは新しい定数SQL_NULLによって記述されるパラメータの型です。
- パラメータのSQL_NULL型のXSQLVAR.sqldataは、常にクライアントのアプリケーションからNULLポインタとして渡される必要があり、決してアクセスされることはありません。

> 出力セットに指定されたNULL
>
> NULLが出力定数として指定された場合（select NULL from ...）は、SQL_NULLによってではなく、当面はCHAR(1)として記述されることになります。これは将来のバージョンで変更される可能性があります。

LIST()関数の拡張

Adriano dos Santos Fernandes

トラッカー・リファレンス CORE-1453

LIST()関数の区切り文字引数として文字列表現が認められるようになりました。

例

```
SELECT
  DISCUSSION_ID,
  LIST(COMMMENT, ASCII_CHAR(13))
FROM COMMENTS
  GROUP BY DISCUSSION_ID;
```

DATEADD()およびDATEDIFF()関数の拡張

Adriano dos Santos Fernandes

関数DATEADDとDATEDIFFにWEEKユニットが導入されました。

MILLISECOND、SECOND. MINUTE. HOURユニットはもう無効なユニットではなく、DATE引数と一緒に使用できます。

BIN_NOT()関数の追加

Adriano dos Santos Fernandes

新しい関数BIN_NOT()はその引数で実行されるbitごとのNOT演算の結果を返します。これで、バージョン2.1で追加された組み込みバイナリ関数のセットが揃ったことになります。

構文パターン

```
BIN_NOT( <数字> )
```

例

```
select bin_not(flags) from x;
```

読み取り専用データベースでの一時表への書き込み

Vladyslav Khorsun

（バージョン2.5.1）読み取り専用データベースでのグローバル一時表への書き込み操作が可能になりました。

トラッカー・リファレンス CORE-3399

オプティマイザの改善

認識済みの問題に対処するオプティマイザロジックの変更には以下のものを含みます：

CROSS JOINのロジック（D. Yemanov）
: CROSS　JOINが空のテーブルを含んでいる場合でも、オプティマイザには、クエリが無効であることを検出してすぐに空のセットを返すための特別なロジックがありませんでした。そのショートカット・ロジックが実装されました。

  トラッカー・リファレンス CORE-2200

> **Note**
>
> 同様の変更はv.2.1.2で実装されています。

派生テーブル（A. dos Santos Fernandes）
: 派生テーブルの使用中に利用できるコンテキスト数の上限が拡大されました。

  トラッカー・リファレンス CORE-2029

DEFAULT評価のタイミング（A. dos Santos Fernandes）
: まれな状況ですが、カラムに定義されたDEFAULTの値または式に対する評価が早いと、そのカラムに与えられた入力値の評価に関して、混乱した状況を引き起こす可能性があります。この可

能性については、DEFAULTの評価を延期し、実際の必要がない限り、実際には全く評価を実行しないことによって対処します。

トラッカー・リファレンス CORE-1842

NOT IN検索でのインデックスの使用（A. dos Santos Fernandes）
NOT IN句でインデックスの使用を可能にすることにより、パフォーマンスが改善されました。

トラッカー・リファレンス CORE-1137

Undoログのメモリ消費（D. Yemanov）
一度のトランザクションで長く続いた更新の後の、Undoログによる過剰なメモリ消費が回避されました。

トラッカー・リファレンス CORE-1477

その他の改善点

小さなイライラを解消するその他の変更点は、以下のものを含みます：

FREE_ITエラー検出（A. dos Santos Fernandes）
従来は、FREE_ITを使って宣言されたUDFが、返されたポインタがib_util_malloc()関数によって割り当てられなかった場合にクラッシュすることがありました。現在は、そのような状況が検出されると、例外がスローされ、ポインタは割り当てられた状態を維持します。

トラッカー・リファレンス CORE-1937

メッセージ“式の評価はサポートされていません”の改善（C. Valderrama）
多くのセカンダリGDSコードが導入されました。これにより、“式の評価はサポートされていません”の例外がスローされて失敗していた操作に関して、詳細情報が与えられるようになりました。例えば：

'Argument for @1 in dialect 1 must be string or numeric（ダイアレクト1での@1の引数は文字列か数値でなければなり
'Strings cannot be added to or subtracted from DATE or TIME types（文字列を追加したり、DATE型やTIME型から取り
'Invalid data type for subtraction involving DATE, TIME or TIMESTAMP types（DATE、TIME、TIMESTAMP型が必要な減
など

これらの詳細メッセージは、ステータスベクターでのisc_expression_eval_err（式の評価はサポートされていません）エラー用のGDSコードに従っています。

トラッカー・リファレンス CORE-1799

Chapter 11

手続き型SQL（PSQL）

Firebirdの手続き型言語（PSQL）にいくつかの重要な変更があります。すなわち、トリガ用の言語セット、ストアドプロシージャ、特にEXECUTE　STATEMENTの新しい拡張機能に関連した動的に実行可能な文、といったものです。また、このリリースでは“自律型トランザクション”の到来が告げられます。

クイックリンク

自律型トランザクション

Adriano dos Santos Fernandes

トラッカー・リファレンス CORE-1409

この新たな実装により、PSQLモジュール内の自律型トランザクションでコードの一部の実行ができるようになりました。これは、例外を発生させる必要があるけれどもデータベースの変更をロールバックしたくないという状況では便利です。

新しいトランザクションは、起動しているもののと同じ分離レベルで開始されます。自律型トランザクション内の文で発生した任意の例外は変更を引き起こし、ロールバックされます。その文が終わりまで実行された場合はトランザクションはコミットされます。

> Warning
>
> 自律型トランザクションは起動しているものから独立しているので、この機能は、デッドロックを避けるよう注意して使う必要があります。

構文パターン

```
IN AUTONOMOUS TRANSACTION
DO
  <単純な文 | 複合文>
```

使用例

```
create table log (
  logdate timestamp,
  msg varchar(60)
);

create exception e_conn 'Connection rejected';

set term !;

create trigger t_conn on connect
as
begin
  if (current_user = 'BAD_USER') then
  begin
    in autonomous transaction
    do
    begin
      insert into log (logdate, msg) values (current_timestamp, 'Connection rejected');
    end

    exception e_conn;
  end
end!

set term ;!
```

PSQLの変数にデータベースのカラムの型を借用

Adriano dos Santos Fernandes

トラッカー・リファレンス CORE-1356

この機能はバージョン2での実装を拡張したものです。これによって、PSQLで変数を宣言するための“データ型”としてドメインが利用できるようになりました。この用途のために、テーブルまたはビューからカラム定義のデータ型を借用できるようになりました。

構文パターン

```
    data_type ::=
        <組み込みデータ型>
      | <ドメイン名>
      | TYPE OF <ドメイン名>
      | TYPE OF COLUMN <テーブルまたはビュー>.<カラム>
```

> Note
>
> TYPE OF THEはカラムの型のみ取得します。カラムに定義された任意の制約またはデフォルト値は無視されます。

例

```
CREATE TABLE PERSON (
  ID INTEGER,
  NAME VARCHAR(40)
  );

CREATE PROCEDURE SP_INS_PERSON (
  ID TYPE OF COLUMN PERSON.ID,
  NAME TYPE OF COLUMN PERSON.NAME
  )
  AS
DECLARE VARIABLE NEW_ID TYPE OF COLUMN PERSON.ID;
BEGIN
  INSERT INTO PERSON (ID, NAME)
    VALUES (:ID, :NAME)
    RETURNING ID INTO :NEW_ID;
END
```

トラップに注意！

バージョン2.5以上では、カラムのデータ型の変更が可能になっています。そのカラムがストアドプロシージャやトリガで参照されていたとしても、例外がスローされることはありません。ただし、コンパイル済みのPSQLはBLOB内でバイナリ表現（“BLR”）として静的に格納されているため、オリジナルのBLRはバックアップやリストアを行なっても残っています。BLRは静的であるため、データ型を変更しても更新されません。

このことは、TYPE　OF構文や、テーブルから影響を受けるカラム、また、それらのテーブルから派生した任意のビューのカラムを使って変数が宣言されている場合、コンパイル済みBLRがデータ型の変更によって破壊されてしまうことを意味します。最善の場合、BLRは“注意が必要”のフラグを付けられますが、テストでは、どんな条件でもフラグ付けが行なわれるわけではないことが分かっています。

つまり、フィールドがコンパイル済みPSQLに何らかの依存関係を持っていたとしても、あなたがそのフィールドの型を変更するのをエンジンはもう止めてくれなくなった、ということです。変更された状況の管理のため、影響を受けるプロシージャやトリガを特定したり、変更に対応してそれらを再コンパイルしたりすることは、あなた自身の問題となります。

EXECUTE文の新たな拡張

われわれのリリースノートでは珍しく、この章はPSQLのEXECUTE　STATEMENT文の完全で新しく拡張された構文から始め、その後で、さまざまな新機能とその使い方の説明に移ります。

```
[FOR] EXECUTE STATEMENT <query_text> [(<input_parameters>)]
    [ON EXTERNAL [DATA SOURCE] <connection_string>]
    [WITH {AUTONOMOUS | COMMON} TRANSACTION]
    [AS USER <user_name>]
    [PASSWORD <password>]
    [ROLE <role_name>]
    [WITH CALLER PRIVILEGES]
    [INTO <variables>]
```

Note

オプションの句の順序は固定されておらず、例えば、次のモデルに基づく文も有効となります：

```
[ON EXTERNAL [DATA SOURCE] <接続文字列>]
[WITH {AUTONOMOUS | COMMON} TRANSACTION]
[AS USER <ユーザー名>]
[PASSWORD <パスワード>]
[ROLE <ロール名>]
[WITH CALLER PRIVILEGES]
```

句の重複はできません。

コンテキスト・イシュー

ON EXTERNAL DATA SOURCE句がない場合、EXECUTE STATEMENTは通常、CURRENT_CONNECTIONの文脈内で実行されます。AS USER句が省略されている場合、または、これがCURRENT_USERと同じ<ユーザー名>引数を持っている場合は、その通りです。

しかし、<ユーザー名>がCURRENT_USERと異なる場合、文は、同じエンジン・インスタンス内で、Y-Valveとリモートレイヤーなしに確立された別の接続で実行されます。

Note

AS USER <ユーザー名>句がない場合はCURRENT_USERがデフォルトとなります。

認証

CURRENT_CONNECTIONとは異なる接続にサーバー認証が必要な場合、例えば、外部のデータソースでEXECUTE STATEMENTコマンドを実行するには、AS USERやPASSWORD句が必要になります。しかし、いくつかの条件のもとで、PASSWORDが省略できることがあります。その効果は以下の通りです：

1. Windowsでは、“信頼された認証”がアクティブとなっており、AS USERパラメータが欠けているか無効、またはCURRENT_USERと同じ場合、CURRENT_CONNECTION（つまり、外部データソースではなく）に“信頼された認証”が実行されます。

2. 外部データソース・パラメータが与えられ、その<接続文字列>がCURRENT_CONNECTIONと同じデータベースを参照している場合は、有効なユーザーアカウントはCURRENT_USERのものとなります。

3. 外部データソース・パラメータが与えられ、その<接続文字列>が、CURRENT_CONNECTIONがアタッチしているものとは異なるデータベースを参照している場合、有効なユーザーアカウントは、Firebirdプロセスが現在稼働しているOSのアカウントとなります。

PASSWORD句が欠けている他の場合には、isc_dpb_user_nameだけはDPB（アタッチメントパラメータ）で与えられ、native認証が試行されます。

トランザクションの挙動

この新しい構文はトランザクションの挙動を設定するオプションの句を含んでいます：WITH AUTONOMOUS TRANSACTIONとWITH COMMON TRANSACTIONです。デフォルトはWITH COMMON TRANSACTIONであり、特に指定する必要はありません。トランザクションの有効期間はCURRENT_TRANSACTIONの有効期間によって決まり、CURRENT_TRANSACTIONに従ってコミットまたはロールバックされます。

WITH COMMON TRANSACTIONによる挙動は以下の通りです：

a. 外部データソースでの任意のトランザクションを、CURRENT_TRANSACTIONと同じパラメータで開始します；あるいは、

b. CURRENT_TRANSACTION内の文を実行します；または、

c. CURRENT_CONNECTION内で開始される別のトランザクションを使用します。

WITH AUTONOMOUS TRANSACTION設定は、新しいトランザクションをCURRENT_TRANSACTIONと同じパラメータで開始します。文が例外を発生させずに実行されればトランザクションはコミットされ、エラーがあった場合はロールバックされます。

アクセス権限の継承

Vladyslav Khorsun

トラッカー・リファレンス CORE-1928

設計上、EXECUTE STATEMENTのもともとの実装では、実行可能なコードは実行されるストアドプロシージャまたはトリガのアクセス権限から分離され、CURRENT_USERが利用できる権限へと戻されます。この戦略は、任意の文を実行されてしまう脆弱性を低減できるため、概ね賢いものです。しかし、堅牢な環境やプライバシーが問題にならない状況では、制約と感じられることがあります。

オプションの句WITH CALLER PRIVILEGESの導入により、実行可能な文に、実行されるストアドプロシージャやトリガのアクセス権限を継承させられるようになりました。文は呼び出し先のストアドプロシージャやトリガに適用される任意の追加権限を使用して作成されます。その効果は、文がストアドプロシージャやトリガによって直接実行された場合と同様のものです。

> Important
>
> WITH CALLER PRIVILEGESオプションはON EXTERNAL DATA SOURCEオプションとは互換性がありません。

PSQLからの外部クエリ

Vladyslav Khorsun

トラッカー・リファレンス CORE-1853

EXECUTE STATEMENTに<接続文字列>引数を付けたON EXTERNAL DATA SOURCE句を含めることで、外部データベースに対するクエリがサポートされました。

<接続文字列>引数

<接続文字列>のフォーマットはごく普通のもので、API関数isc_attach_database()に渡されます。すなわち、

[<ホスト名><プロトコル区切り文字>]データベースのパス

キャラクタ・セット

外部のデータソースへの接続には、CURRENT_CONNECTIONの文脈で使われているものと同じキャラクタ・セットが使われます。

アクセス権限

外部のデータソースが別のサーバー上にある場合、AS USER <ユーザー名>句とPASSWORD <パスワード>句が必要となります。

外部のデータソースが別のサーバ上にある場合、WITH CALLER PRIVILEGES句は無視されます。

情報が足りないよ！ロールはどうなってるの？

> Note
>
> 外部接続での二相トランザクションはバージョン2.5では利用できません。

EXECUTE STATEMENTでの動的パラメータ

Vladyslav Khorsun
Alex Peshkov

トラッカー・リファレンス CORE-1221

新たな拡張により、パラメータ化されたDSQL文に似た方法で、動的な入力パラメータ（プレースホルダ）を使って文を作成することができるようになりました。クエリのテキストそのものをパラメータとして渡すことも可能です。

構文の表記規則

この仕組みには、実行時のパーシングを容易にし、また、DSQLパラメータを扱うポピュラーなクライアント・ラッパー層（Delphiなど）と互換性のあるスタイルでのパラメータ“命名”オプションを許可するいくつかの表記規則が採用されています。API自体が持つ表記規則もサポートされており、定義された順序で名前なしパラメータを渡すことができます。ただし、名前付き、名前なしパラメータを混在させることはできません。

新しい拘束演算子

この点について、動的パラメータ機能の実装の中で、同等性テストでのクラッシュを避けるために、実行時の値を名前付きパラメータに拘束する新しい代入演算子を導入する必要がありました。この新しい演算子はPascalの代入演算子を真似たもの“:=”です。

パラメータを定義する構文

```
<input_parameters> ::=
  <named_parameter> | <input_parameters>, <named_parameter>

<named_parameter> ::=
  <parameter name> := <expression>
```

名前付き入力パラメータの例

例えば、次のPSQL文は<query_text>と<input_parameters>（<named_parameter>）の両方を定義します：

```
EXECUTE BLOCK AS
  DECLARE S VARCHAR(255);
  DECLARE N INT = 100000;
  BEGIN
  /* 通常のPSQL文字列としての<query_text>の割り当て */
    S = 'INSERT INTO TTT VALUES (:a, :b, :a)';

    WHILE (N > 0) DO
    BEGIN
    /* 各ループの実行は文字列の値と入力パラメータに
       拘束される値の両方に適用されます */

      EXECUTE STATEMENT (:S) (a := CURRENT_TRANSACTION, b := CURRENT_CONNECTION)
      WITH COMMON TRANSACTION;
      N = N - 1;
    END
  END
```

名前なし入力パラメータの例

同様の文で、今度は名前なし入力パラメータを使用し、定数の引数を直接渡します。

```
EXECUTE BLOCK AS
  DECLARE S VARCHAR(255);
  DECLARE N INT = 100000;
  BEGIN
    S = 'INSERT INTO TTT VALUES (?, ?, ?)';

    WHILE (N > 0) DO
    BEGIN
      EXECUTE STATEMENT (:S) (CURRENT_TRANSACTION, CURRENT_CONNECTION, CURRENT_TRANSACTION);
      N = N - 1;
    END
```

```
END
```

> Note
>
> 次の点に注意して下さい。<query_text>と<input_parameters>の両方を使用する場合、必ず<query_text>を括弧で囲みます。すなわち、
>
> ```
> EXECUTE STATEMENT (:sql) (p1 := 'abc', p2 := :second_param) ...
> ```

例外処理

例外の取り扱いは、ON EXTERNAL DATA SOURCE句が与えられているかどうかで異なります。

ON EXTERNAL DATA SOURCE句がある場合

ON EXTERNAL DATA SOURCE句があると、Firebirdは未知のデータソースから与えられるエラーコードを解釈することができず、エラー情報そのものを解釈し、文字列にラップして自身のエラー・ラッパー（isc_eds_connectionまたはisc_eds_statement）へと渡します。

リモートエラーを解釈したテキストには、エラーコードと対応するメッセージの両方が含まれます。

1. isc_eds_connectionエラーのフォーマット

```
テンプレート文字列
  Execute statement error at @1 :¥n@2Data source : @3
ステータスベクター・タグ
  isc_eds_connection,
  isc_arg_string, <失敗したAPI関数名>,
  isc_arg_string, <外部エラーを解釈したテキスト>,
  isc_arg_string, <データソース名>
```

2. isc_eds_statementエラーのフォーマット

```
テンプレート文字列
  Execute statement error at @1 :¥n@2Statement : @3¥nData source : @4
ステータスベクター・タグ
  isc_eds_statement,
  isc_arg_string, <失敗したAPI関数名>,
  isc_arg_string, <外部エラーを解釈したテキスト>,
  isc_arg_string, <クエリ>,
  isc_arg_string, <データソース名>
```

PSQLレベルでは、これらのエラーのシンボルは、それらを他のgdscodeと同様に扱うことで処理することができます。例えば、

```
WHEN GDSCODE eds_statement
```

> Note
>
> 現状では、エラーコード生成元にWHEN文ではアクセスできません。この状況は将来改善されるかもしれません。

ON EXTERNAL DATA SOURCE句がない場合

ON EXTERNAL DATA SOURCE句がない場合は、エラーの元のステータスベクターが呼び出し元であるPSQLコードにそのまま渡されます。

例えば、動的な文がisc_lock_conflictの例外を引き起こした場合、例外は呼び出し元に渡され、通常のハンドラで取り扱われます：

```
  WHEN GDSCODE lock_conflict
```

EXECUTE STATEMENTの使用例

以下の例は、EXECUTE　STATEMENTの拡張をアプリケーションに適用する方法のサンプルを提供しています。

テスト接続とトランザクション

設定のバリエーションを比較できるように、テストを二つ挙げておきます：

テスト　a）：同じトランザクションでこの文を数回実行する―現在のデータベースへの新しい接続を三つ作成し、呼び出しがあるたびにそれを再利用する。トランザクションは再利用される。

```
EXECUTE BLOCK
  RETURNS (CONN INT, TRAN INT, DB VARCHAR(255))
AS
  DECLARE I INT = 0;
  DECLARE N INT = 3;
  DECLARE S VARCHAR(255);
BEGIN
  SELECT A.MON$ATTACHMENT_NAME FROM MON$ATTACHMENTS A
   WHERE A.MON$ATTACHMENT_ID = CURRENT_CONNECTION
    INTO :S;

  WHILE (i < N) DO
  BEGIN
    DB = TRIM(CASE i - 3 * (I / 3)
      WHEN 0 THEN '¥¥.¥' WHEN 1 THEN 'localhost:' ELSE '' END) || :S;

    FOR EXECUTE STATEMENT
      'SELECT CURRENT_CONNECTION, CURRENT_TRANSACTION
      FROM RDB$DATABASE'
      ON EXTERNAL :DB
      AS USER CURRENT_USER PASSWORD 'masterkey' -- ただの例です
      WITH COMMON TRANSACTION
      INTO :CONN, :TRAN
```

```
    DO SUSPEND;

    i = i + 1;
  END
END
```

テスト　b）：同じトランザクションでこの文を数回実行する―呼び出しがあるたびに、現在のデータベースへの新しい接続を三つ作成する。

```
EXECUTE BLOCK
  RETURNS (CONN INT, TRAN INT, DB VARCHAR(255))
AS
  DECLARE I INT = 0;
  DECLARE N INT = 3;
  DECLARE S VARCHAR(255);
BEGIN
  SELECT A.MON$ATTACHMENT_NAME
    FROM MON$ATTACHMENTS A
  WHERE A.MON$ATTACHMENT_ID = CURRENT_CONNECTION
   INTO :S;

  WHILE (i < N) DO
  BEGIN
    DB = TRIM(CASE i - 3 * (I / 3)
      WHEN 0 THEN '¥¥.¥'
      WHEN 1 THEN 'localhost:'
      ELSE '' END) || :S;

    FOR EXECUTE STATEMENT
    'SELECT CURRENT_CONNECTION, CURRENT_TRANSACTION FROM RDB$DATABASE'
      ON EXTERNAL :DB
      WITH AUTONOMOUS TRANSACTION -- 自律型トランザクションに注意
      INTO :CONN, :TRAN
    DO SUSPEND;

    i = i + 1;
  END
END
```

入力の評価のデモ

一度だけ評価される入力式のデモ：

```
EXECUTE BLOCK
  RETURNS (A INT, B INT, C INT)
AS
BEGIN
  EXECUTE STATEMENT (
    'SELECT CAST(:X AS INT),
            CAST(:X AS INT),
            CAST(:X AS INT)
       FROM RDB$DATABASE')
      (x := GEN_ID(G, 1))
    INTO :A, :B, :C;

  SUSPEND;
```

```
END
```

書き込み速度テスト

パラメータ化していない形式のEXECUTE　STATEMENTと比較するため、われわれが以前に挙げていた入力パラメータの使用例を再掲します：

```
RECREATE TABLE TTT (
  TRAN INT,
  CONN INT,
  ID INT);

-- 直接の書き込み：

EXECUTE BLOCK AS
  DECLARE N INT = 100000;
BEGIN
  WHILE (N > 0) DO
  BEGIN
    INSERT INTO TTT VALUES (CURRENT_TRANSACTION, CURRENT_CONNECTION, CURRENT_TRANSACTION);
    N = N - 1;
  END
END

-- プリペアされた動的な文を介した書き込み
-- 名前付き入力パラメータを使用：

EXECUTE BLOCK AS
  DECLARE S VARCHAR(255);
  DECLARE N INT = 100000;
BEGIN
  S = 'INSERT INTO TTT VALUES (:a, :b, :a)';

  WHILE (N > 0) DO
  BEGIN
    EXECUTE STATEMENT (:S)
      (a := CURRENT_TRANSACTION, b := CURRENT_CONNECTION)
    WITH COMMON TRANSACTION;
    N = N - 1;
  END
END

-- プリペアされた動的な文を介した書き込み
-- 名前なし入力パラメータを使用：

EXECUTE BLOCK AS
DECLARE S VARCHAR(255);
DECLARE N INT = 100000;
BEGIN
  S = 'INSERT INTO TTT VALUES (?, ?, ?)';

  WHILE (N > 0) DO
  BEGIN
    EXECUTE STATEMENT (:S) (CURRENT_TRANSACTION, CURRENT_CONNECTION, CURRENT_TRANSACTION);
    N = N - 1;
  END
END
```

PSQLのその他の改善点

既存のPSQL構文の改善には以下のものが含まれます：

PSQL式としてのサブクエリ

A. dos Santos Fernandes

トラッカー・リファレンス CORE-2580

従来は、PSQL式として使われるサブクエリは、SQLの項としてロジカルに有効な場合でも、例外を返していました。例えば、以下の構文ではすべてエラーが返されました：

```
var = (select ... from ...);
if ((select ... from ...) = 1) then
if (1 = any (select ... from ...)) then
if (1 in (select ... from ...)) then
```

このような有効と見られる式は認められるようになりました。これで、SELECT...INTOを使って何が何でもスカラサブクエリの出力を取得して中間変数に渡すといった必要はなくなりました。

コンテキスト変数としてのSQLSTATE

D. Yemanov

トラッカー・リファレンス CORE-2890

（バージョン2.5.1）SQLSTATEがPSQLコンテキスト変数として利用可能になりました。WHEN文中の例外処理部でGDSCODEやSQLCODEと同様に使われます。

Chapter 12

国際言語のサポート（INTL）

Adriano dos Santos Fernandes

このリリースでは、Firebirdによる国際言語環境の処理機能を厳格化し、また拡張するため、いくつかの改善が施されました。

データベースのデフォルトのコレーション属性

ODSバージョン11.2以上では、デフォルトのキャラクタ・セットと関連づけられたデフォルトのコレーション属性を持つデータベースを任意に作成できるようになりました。詳細は、DDLの章のデータベースのデフォルトのコレーション属性を参照して下さい。

ALTER CHARACTER SET コマンド

DDL構文が導入され、データベースレベルで設定されるキャラクタ・セットのコレーションがデフォルトで可能になりました。詳細は、DDLの章のALTER CHARACTER SET コマンドを参照して下さい。

接続文字列とキャラクタ・セット

ファイル名に非ASCIIキャラクタを含む場合に起きがちだった問題を避けるため、APIデータベース接続（DPB）エリアに、サーバーとクライアントのキャラクタ・セットおよび／またはコードページと連携する機能が実装されました。

Firebird APIとODSの変更の章の接続文字列とキャラクタ・セットの項を参照して下さい。APIに特に興味がない方でも、この種のの問題に悩まされたことがあれば、この項は読んでおく価値があるでしょう。

その他の改善点

Introducer構文の使用法

introducer構文、つまり、キャラクタ・セット名にアンダースコアを前置することにより、それに続くリテラルなテキストを強制的にそのキャラクタ・セットに変換させる手法を用いた場合、一つのSQL文が複数のキャラクタ・セットを扱わざるをえない状況で、問題が起きることがありまし

た。実際に起きる問題は、変換エラーや、不正な形式の文字列のエラー、または何らかの予期せぬ挙動を示すといったように、バージョンによって異なります。

二つの違った使用シーンで問題が起きるようです—

1. あるクエリがMON$STATEMENTSからselectを実行している時に、他のクエリがintroducer構文を使う場合

2. introducer構文がPSQLモジュールで使用された場合

このような問題を回避するため、リテラルな文字列をintroducerから渡されるASCIIキャラクタによる16進表現に変換できるようになりました。例えば、

```
select _dos850 '123áé456' from rdb$database
```

これが、次のように変換されます

```
select _dos850 X'313233A082343536' from rdb$database
```

不正な形式のUNICODE_FSSキャラクタの禁止

トラッカー・リファレンス CORE-1600

不正な形式のキャラクタがUNICODE_FSSカラムのデータとして認められなくあんりました。

不正な形式の文字列の修正

gbakユーティリティ・コードに新しいリストア・スイッチが追加され、影響を受けたデータベースのバックアップをリストアし、不正な形式のUNICODE_FSSデータやメタデータを修正できるようになりました。詳細は、ユーティリティの章のgbakの節をを参照して下さい。

数値ソート属性

トラッカー・リファレンス CORE-1945

UNICODEコレーション向けに、数値のソート用オーダーを指定するカスタム属性NUMERIC-SORTが使えるようになりました。

フォーマットと使用法

```
NUMERIC-SORT={0 | 1}
```

デフォルト値は0で、数値もアルファベットのオーダーでソートされます。例えば：

```
1
10
100
2
20
```

値を1に設定すると、数値のオーダーでソートされます。例えば：

```
1
2
10
20
100
```

例

```
create collation unicode_num for utf8
from unicode 'NUMERIC-SORT=1';
```

キャラクタ・セットとコレーション

UNICODE_CI_AI

トラッカー・リファレンス CORE-824

UNICODE_CI_AI：UTF8向けに大文字小文字、アクセント記号を無視したコレーションが追加されました。

WIN_1258

トラッカー・リファレンス CORE-2185

他のWIN*キャラクタ・セットとの整合性のため、キャラクタ・セットWIN1258にエイリアスWIN_1258が追加されました。

キャラクタ・セットSJISおよびEUCJ

トラッカー・リファレンス CORE-2103

キャラクタ・セットSJISとEUCJの文字列で適格性の検証が行われるようになりました。

キャラクタ・セットGB18030

トラッカー・リファレンス CORE-2636

GB18030は、中国国内のソフトウェアに必須の言語・キャラクタをサポートする中国語の国家規格です。ICUからアクティベートされています。

Chapter 13

コマンドライン・ユーティリティ

以前のクライアントとの非互換性

統計ルーチンを64bitサーバーで正しく動作させる新たなストラクチャーで、32bitツールが正しく動作できるようにするため、いくつかの新しい内部API関数（struct perf64とperf64_xxx）を導入し、それらを使用するisqlとqliを変更する必要がありました。このことは、バージョン2.5のisqlとqliのプログラムが以前のFirebirdクライアントと互換性がないことを意味します。

詳細は、エンジンの章のデータベースの統計が64bit値で適切に動作するようにの項を参照して下さい。

fbtracemgr

Vlad Khorsun

トラッカー・リファレンス CORE-2524

これは新たなトレース機能のインターフェースとなる新しいCLIユーティリティです。使用法は以下の通り―

```
fbtracemgr <アクション> [<パラメータ>]
```

アクションスイッチ

```
-STA[RT]      トレースのセッションを開始
-STO[P]       トレースのセッションを停止
-SU[SPEND]    トレースのセッションを中断
-R[ESUME]     トレースのセッションを再開
-L[IST]       トレースのセッションの一覧
```

パラメータ

アクションパラメータ

```
-N[AME]     <文字列>  セッション名
-I[D]       <番号>    セッションID
-C[ONFIG]   <文字列>   設定ファイル名
```

接続パラメータ

```
-SE[RVICE]  〈文字列〉   サービス名
-U[SER]     〈string〉   ユーザー名
-P[ASSWORD] 〈文字列〉   パスワード
-FE[TCH]    〈文字列〉   ファイルからパスワードを取得
-T[RUSTED]  〈文字列〉   フォース・トラステッド認証
```

fbtracemgrの使用例

```
fbtracemgr -SE remote_host:service_mgr -USER SYSDBA -PASS masterkey -LIST
fbtracemgr -SE service_mgr -START -NAME my_trace -CONFIG my_cfg.txt
fbtracemgr -SE service_mgr -SUSPEND -ID 2
fbtracemgr -SE service_mgr -RESUME -ID 2
fbtracemgr -SE service_mgr -STOP -ID 4
```

> 注意
>
> 1. 全てのスイッチ識別子とパラメータ識別子は大文字小文字を区別しません。
> 2. 対話型トレースセッションを停止するには、プラットフォームに関わらず、Ctrl-Cを押して下さい。

バージョン2.5.1での改善点

内部トレース・エラーの診断が改善されました（トラッカー・リファレンス CORE-3413）。

出力が定期的に書き込まれるようになりました（トラッカー・リファレンス CORE-3324）。

ファイル、プロンプトからのパスワード取得

Alex Peshkov

-passwordパラメータを取る任意のコマンドライン・ユーティリティには、パスワード盗聴に対する脆弱性があります。バージョン2.1以降、POSIXプラットフォームのプロセスリストで平文で示されていた引数［PASSWORD］はアスタリスク（＊）で表示されるよう改善されました。

認証のないユーザーからパスワードを秘匿する第二段階として、このリリースでは、パスワードをファイルまたは（POSIXでは）STDINから取得できるようになりました。

新しい -fetch_passwordスイッチ

Firebird　2.5では、認証用のパスワードを取得する全てのコマンドライン・ユーティリティ向けに、オプションで-pa[ssword]と置き換え可能な-fet[ch_password]スイッチが新たに導入されました。このスイッチは一定のルールに従って右側から短縮して使用できます。

注意して下さい

1. qliユーティリティはこのルールの例外で、有効なスイッチは-Fだけです。
2. この新しいスイッチをgsecユーティリティの-pwスイッチの代わりに使うことはできません。

-fetch_passwordの使い方

このスイッチは一つのパラメータを要求します。これはパスワードを含むファイルのファイルパスで、クオテーションのない文字列で表します。その呼び出しがスーパーユーザー／管理者の権限を持つシステムユーザーからのものでない場合、それは呼び出しを行うシステムユーザーがアクセスできる配置場所でなければなりません。

例えば、

```
isql -user sysdba -fet passfile server:employee
```

このコマンドは、現在の作業ディレクトリ内にある“passfile”という名前のファイルから一行目を抽出し、呼び出しの引数［PASSWORD］としてロードします。

filenameをstdinとして指定することができます：

```
isql -user sysdba -fet stdin server:employee
```

stdinがターミナルの場合は、次のようなプロンプトが現れます-

```
Enter password:
```

-このように、オペレータにパスワード入力を求めます。

Tip

POSIX版では、次のように指定した場合も、プロンプトが現れます。

```
    -fetch /dev/tty
```

このテクニックは、例えば、stdin（全てを一行で書く場合）からリストアする必要がある場合に役立つでしょう。

```
bunzip2 -c emp.fbk.bz2 | gbak -c stdin /db/new.fdb
   -user sysdba -fetch /dev/tty
```

gsec

gsecに施された改善は以下の通りです：

WindowsのAdministrators用マッピング・スイッチ

Alex Peshkov

バージョン2.1以来、Windowsドメイン管理者はユーザー管理機能へのフルアクセスが許可されていました。バージョン2.5では、SYDBAがセキュリティ・データベースをそのように設定しない限り、これらの権限は自動では付与されません。

管理機能の章には、新しいシステムロールRDB$ADMINについての詳しい解説があります。そこに、SYSDBAがWindowsのAdministratorsをデータベースのロールRDB$ADMINに自動マッピングする機能の有効・無効の切り替えに用いる新しいALTER ROLE構文についての説明があります。これには、セキュリティ・データベースでのユーザーの作成・変更・削除も含まれます。

自動マッピングはgsecコマンドライン呼び出しで新しい-mappingスイッチを使っても実現できます。

OS管理者をロールRDB$ADMINへマッピングする

新しい-mappingスイッチは、OSのユーザーとセキュリティ・データベースのロールRDB$ADMINとの関連付けを有効・無効にするために使用します。これは引数を一つ取ります：setで関連付けは有効に、dropで無効になります。構文は次の通り：

```
-mapping {set | drop}
```

ロールRDB$ADMINをFirebirdユーザーに付与する

システムロールRDB$ADMINの導入により、通常のユーザーの権限昇格を行なえるようになりました。しかし、SYSDBAも含めてどのユーザーも、どのユーザーに対しても、セキュリティ・データベースに直接アタッチして他のユーザーを管理するユーザーに必要なパーミッションを付与することはできませんでした（今もできません）。新しいCREATE USERとALTER USER分の構文に含まれるパラメータ-GRANT ADMIN ROLE-は、セキュリティ・データベースですでにロールRDB$ADMINを手にしているSYSDBAや他のユーザーが、ロールRDB$ADMINを通常のユーザーに、言わば“手の届く範囲で”、適用することを可能にしています。

同じことは、gsecで新しい-adminスイッチを使うことで実現できます。これは引数を一つ取ります：YES（security2.fdbで指定のユーザーにロールRDB$ADMINを付与）またはNO（削除）です。構文は次の通り：

```
-admin {YES | NO}
```

gsecのコマンドライン・ヘルプ

Claudio Valderrama

トラッカー・リファレンス CORE-756

gsecにはパラメータ・ヘルプ機能が実装されています。-helpまたは-?スイッチを使ってアクセスできます。

fbsvcmgr

システムロールRDB$ADMINに関連してgsecとサービスパラメータブロック（SPB）に追加された機能は、fbsvcmgrユーティリティを用いた適切なサポートによってカバーされます。

- gsec -mappingに対しては、二つの新しいタグアイテムがあります：isc_action_svc_set_mappingとisc_action_svc_drop_mappingです。
- gsec -adminに対しては：spb_longの新しいパラメータisc_spb_sec_admin。この値は0（REVOKE ADMIN ROLEを意味する）またはゼロ以外（GRANT ADMIN ROLEを意味する）となります。

ロールRDB$ADMINの全容については、管理機能の章の新しいシステムロールRDB$ADMINの項を参照して下さい。

gbak

不正な形式の文字列の修正

Adriano dos Santos Fernandes

トラッカー・リファレンス CORE-1831

gbakユーティリティに、不正な形式のUNICODE_FSSキャラクタを含むデータやメタデータをそれぞれ修正するための新しいリストア・スイッチが二つ追加されました。影響を受けたデータベースをバックアップからリストアする際に使用します。

Switch Syntax

```
-FIX_FSS_D(ATA) <charset> -- 不正な形式のUNICODE_FSSデータを修正
-FIX_FSS_M(ETADATA) <charset> -- 不正な形式のUNICODE_FSSメタデータを修正
```

> リストアのヒントと不正な形式の文字列の例外
>
> （CORE-2754、A. dos Santos Fernandes）
>
> 不正な形式の文字列のためリストアに失敗した場合、gbakは詳細表示出力でヒントを出し、ユーザーに-FIX_FSS_METADATAと-FIX_FSS_DATAスイッチの使用を促します。

キャラクタ・セットのデフォルトのコレーションを保存する

Adriano dos Santos Fernandes

システムテーブルRDB$CHARACTER_SETS内のRDB$DEFAULT_COLLATE_NAMEの現在の値がバックアップ／リストアのサイクルの中で維持されるよう改善されました。

トラッカー・リファレンス CORE-789

リストア時の書き込みパフォーマンスの改善

Adriano dos Santos Fernandes

（バージョン2.5.1）リストアのレコード書き込み段階でのパフォーマンスが改善されました。

トラッカー・リファレンス CORE-3433

nBackup

新たなスイッチの追加

NBackupに四つのスイッチが追加されました：

```
-FE <ファイル名>    ファイルからパスワードを取得
-Z                  バージョン情報の表示
-?                  ヘルプ
-D ON|OFF           ダイレクトI/Oの強制的なon/off
```

- -FE :: 認証パスワードをファイルから取得する機能をサポートします。詳細は、この章の新しい-fetch_passwordスイッチを参照して下さい。
- -Z :: 実行可能なnbackupの詳細なバージョン情報を表示します。
- -? :: スイッチとオプションの使い方を簡単なリストで表示します。
- -D :: オプションONで、ダイレクトI/O操作が有効になります；OFFで無効になります。デフォルトの設定は、以下の通り、OSとFirebirdのバージョンによって異なります：

  Windowsサービス対応プラットフォーム
  　全てのバージョンでONとなります。

  POSIX
  　バージョン2.0～2.0.5、2.1～2.1.2、2.1.4、2.5以降ではOFF

  　O_DIRECTを利用できる場合、2.1.3、2.0.6ではONとなります；それ以外の場合はOFFとなります。利用可能な場合は、POSIX_FADV_NOREUSEも設定されます。

> Note
>
> O_DIRECTもPOSIX_FADV_NOREUSEも利用できない場合、エラーや警告は出ませんが、-D ONは何の効果も持ちません。

> Note
>
> 新しい-Dスイッチのサポートは、このバージョンでのサービスAPIの変更の中にも含まれています。詳細は、Firebird APIとODSの変更の章のisc_spb_nbk_direct on|offを参照して下さい。

POSIX版でのI/Oリソース負荷の低減

POSIX版で、増分バックアップユーティリティnBackupのフルバックアップツールが大きなデータベースのバックアップの際にI/Oリソースを占有し、生産的な作業が停止してしまうという問題に対処するための改善が施されました。nBackupは、ディスクからの読み込みを行う前に、OSのキャッシュからの読み込みを試みるようになりました。これにより、I/O負荷はかなり低減されました。

> Note
>
> その“対価”として、高負荷状況下では、フルバックアップ完了までにかかる時間が10〜15%増加することがあります。

トラッカー・リファレンス CORE-2316

isql

対話型クエリツールisqlにいくつかの変更が施されました。

SQLCODEに代わるSQLSTATE

Claudio Valderrama

isqlはエラーの診断用に、非推奨になったSQLCODEに代わり、SQLSTATEコードを返すようになりました。詳細は、Firebird　　APIとODSの変更の章のSQLSTATEコードのサポートの項を参照して下さい。

数値の指数表示の改善

Claudio Valderrama

トラッカー・リファレンス CORE-1171

isqlは、Windows版とPOSIX版とで常に異なる数値フォーマットで出力していました。POSIX版では二桁の指数部を持ちますが、MicrosoftやIntelのコンパイラのデフォルトの挙動では、指数部はゼロパディングにより何であれ三桁となります。例えば、

```
select cast ('-2.488355210669293e+39' as double precision)
  from rdb$database;
```

- POSIXでは、結果は -2.488355210669293e+39 です。
- Windowsでは、結果は -2.488355210669293e+039 でした。

Windows版での出力を他のプラットフォームのものに合わせるようにisqlの出力は修正されました。

SHOW COLLATIONSのヘルプを追加

isqlのコマンドライン・ヘルプにSHOW　COLLATIONS用のヘルプが追加されました。（トラッカー・リファレンス CORE-2432、A. dos Santos Fernandes）。

SET ROWCOUNT文の追加

Mark O'Donohue

このコマンドは、主にテスターの支援のために追加されたもので、クエリが対話型isqlシェルに返す行数に制限をかけることができます。

使用例

次のisql文は、出力の10,000行目で．以降の行を返すのを止めます：

```
SQL>set rowcount 10000;
```

gpre（プリコンパイラ）

若干のアップデート

Stephen Boyd
Adriano dos Santos Fernandes

トラッカー・リファレンス CORE-1527

GPREが、コンテキスト変数CURRENT_*の完全なセットに加えて、IS　NOT　DISTINCT句とCASE/NULLIF/COALESCE/SUBSTRING関数をサポートするようになりました。

> 非推奨の機能がユーティリティに将来及ぼす影響
>
> Firebird　　3コードベースから組み込み関数Risc_ddlが削除されることを見越して、現在gdefとgpreツールで利用できるいくつかの機能が非推奨になりました-そのため、バージョン2.5では動作するかもしれませんが、Firebird　3では失敗するでしょう。詳細は、互換性の章ををご覧ください。

gstat

Claudio Valderrama

トラッカー・リファレンス CORE-1411

統計レポートユーティリティgstatに、利用可能なスイッチと引数に関するヘルプを参照するためのスイッチ-?と-helpが追加されました。

Chapter 14

インストールの注意

> Firebird 2.5.1で作成またはリストアされたデータベースについての警告
>
> Firebird 2.5.1から上位のサブリリースへアップグレードする場合、データベースの移行の際に、gbakを用いてバックアップ／リストアを行うことを強く推奨します。これが実行できない場合でも、データベース移行時に、最低でも全ての複合インデックスの再作成を行って下さい。
>
> 比較的古いバージョン（ODS11.1以下）またはバージョン2.5.0からアップグレードするデータベースは、この不具合の影響を受けません。

> インストールと移行のガイド
>
> Firebirdバージョン2.0.xと2.1.x向けのインストールと移行ガイド最新版の内容は、バージョン2.5シリーズにも適用できます。この文書のコピーが/doc/ディレクトリ内に存在しなくても、Firebirdのウェブサイトからダウンロードできます。

バイナリパッケージのインストールで問題が起きる可能性がありましたが、解決のためいくつかの改善が施されました。

Linux（POSIX）版

POSIX版のインストールに適用済みの改善内容は以下の通りです。

インストール・スクリプトの刷新

Alex Peshkov

- （CORE-2195）：Linux版クラシックサーバーのインストール・スクリプトが見直され、ドキュメントその他のコンポーネントの所有権とアクセス権の割り当てが改善されました。
- （CORE-2392）：アクティブなPOSIX版スーパーサーバー、スーパークラシックサーバー向けポートの全てのインストール・スクリプトが刷新され、これらのプラットフォーム上で起きていたGuardianに関する問題への対処が行われました。
- （CORE-2626）：/etc/init.d 中のスタートアップ・スクリプトは、これまでホスト・システム内の/var/run/firebirdディレクトリと/tmp/firebird ディレクトリの存在を考慮に入れていませんでした。スタートアップの失敗には様々なタイプのものがありますが、多くはこの欠陥に由来するものでした。

  バージョン2.5とそれ以降のビルドで配布されるスタートアップ・スクリプトではこの問題は対処済みで、解決しています。
- （バージョン2.5.1 CORE-3467）：make installにサイレントインストール・スイッチ（-silent）が追加され、シンプルなインストールと、ビルドとテスト実行の自動化が実現されまし

た。このスイッチを使うとFirebirdのインストールは一切プロンプトを出さずに進行し、SYSDBA用にランダムパスワードが生成されます。このランダムパスワードは、セキュリティ・データベースsecurity2.fdbと同じ配置場所にあるファイルに保存されます。

Firebirdの'configure'専用スイッチ

Alex Peshkov

POSIXプラットフォーム上でインストール・ディレクトリの微調整に使われる標準的なconfigureスイッチの多くは、Firebirdには役立ちません。

標準的なGNUのスイッチをデフォルト設定を変更せずに機能させるのは不可能に近く、簡単明瞭とはほど遠い煩雑なものでした。これらに代わり、configure用の新たなスイッチのセットが追加され、Firebirdのファイル群のきめ細かな配置の設定が可能になりました。

ib_utilローダーも改善され、設定されたレイアウトに応じてエンジンが正しく動作するようになりました。

利用可能なスイッチ

```
--with-fbbin 実行ファイルを含むディレクトリ (PREFIX/bin)
--with-fbsbin システム管理用実行ファイルを含むディレクトリ (PREFIX/bin)
--with-fbconf 設定ファイルを含むディレクトリ (PREFIX)
--with-fblib オブジェクトコード・ライブラリを含むディレクトリ (PREFIX/lib)
--with-fbinclude C/C++ヘッダファイルを含むディレクトリ (PREFIXinclude)
--with-fbdoc ドキュメント類を含むディレクトリ (PREFIX/doc)
--with-fbudf URFを含むディレクトリ (PREFIX/UDF)
--with-fbsample サンプルを含むディレクトリ (PREFIX/examples)
--with-fbsample-db サンプルデータベースを含むディレクトリ (PREFIX/examples/empbuild)
--with-fbhelp QLIのヘルプを含む (PREFIX/help)
--with-fbintl 国際化用ディレクトリ (PREFIX/intl)
--with-fbmisc その他のファイルを含むディレクトリ (PREFIX/misc)
--with-fbsecure-db セキュリティデータベースを含むディレクトリ (PREFIX)
--with-fbmsg メッセージファイルを含むディレクトリ (PREFIX)
--with-fblog ログファイルを含むディレクトリ (PREFIX)
--with-fbglock ガーディアン・ロックを含むディレクトリ (PREFIX)
--with-fbplugins プラグインを含むディレクトリ (PREFIX)
```

Firebirdバイナリのパス検出

Adriano dos Santos Fernandes

トラッカー・リファレンス CORE-2398

POSIXビルド向けに、バイナリがインストールされた場所のパスをFirebirdに正しく検出させる機能が実装されました。

> **Important**
>
> 現状、これは実験段階であり、配布版のバイナリでは無効になっています。これをアクティベートするには、Firebirdをソースからビルドする際に、autogen.shに--enable-binrelocを渡して下さい。

Windows版

エンベデッド版向けのデプロイ構造

Adriano dos Santos Fernandes
Vlad Khorsun

トラッカー・エントリー：CORE-1814

このリリースでは、エンベデッド版のアプリケーション・コンポーネントの配置を従来よりやや柔軟に設定できます。ここでの変更は、アプリケーション開発者の間で問題となっていた、相互に関連するいくつかのイシューに対処したものです。すなわち、

- ib_util.dll、またはfbembed.dll中のクライアントコード、または完全な外部ライブラリなど、他のDLLに依存する外部関数ライブラリをFirebirdの擬似ルートディレクトリ（<root>）下の..¥UDFフォルダにそのまま置くことはできませんでした。依存しているファイルが通常その配置場所には存在しないからです。そのため、%PATH%変数を使ってちょっとした工夫をする必要がありました。

- 従来の擬似<root>の決定ルールは、アプリケーションコードとしてそのフォルダにfbembed.dll（リネームしたものでも可）が存在することを要求していました。その影響で、これを必要とする各アプリケーション用にfbembed.dllの別々のコピーを確保するか、または、実行可能な全てのクライアントを一箇所に配置するか、いずれかの対応が必要でした。

バージョン2.5のエンベデッド版エンジンでは<root>の決定が変更され、もはやそこがアプリケーションのパスである必要はありません。それは、どこであれfbembed.dllの配置によって決まります。このことは、バージョン2.5のエンジンでは、かつてはfbembed.dllをロードしていた任意のエンベデッドアプリケーションやサードパーティ製UDFライブラリやコマンドラインツールのローカルコピーで利用できるFirebirdの擬似<root>のために標準の自己充足的なストラクチャーをセットアップできることを意味します。

Firebirdの擬似ツリーをアプリケーションから切り離したい場合は、必ず、そのストラクチャーのfirebird.confファイル内で<root>の配置の絶対パスを示すRootDirectoryパラメータを設定して下さい。

> Note
>
> 新たなルールが既存のエンベデッドストラクチャーを破壊することはありません。従来通りのやり方でエンベデッドアプリケーションを構築することができます。違いは、実行ファイルをエンベデッド版Firebirdの擬似<root>ストラクチャー専用のコピーと一緒にデプロイする必要がなくなったということです。

MSCV8アセンブリの管理

Vlad Khorsun

トラッカー・エントリー：CORE-2243

> Note
>
> この変更が施されたのがバージョン2.1.2からだったため、この議論もV.2 インストールと移行に関するドキュメント内に特別なトピックとして登場しています。

Firebird 2.5はVisual Studio 2005のMicrosoft MSVC8コンパイラでビルドされています。Firebirdの全バイナリはビルド時に動的リンキングを用いているため、それらは全てランタイム・ライブラリを必要とします。

dll地獄問題を避けるため、Microsoftは複数のアプリケーションで共有される可能性のあるコンポーネントの配布に新たなルールを導入しました。Windows XP以降、共有ライブラリ-Visual C++やVisual Cのランタイムmsvcp80.dll やmsvcr80.dll 、mscvcm80.dll などーは、共有アセンブリまたはプライベートアセンブリとして配布される必要があります。

- Microsoft MSIインストーラーは共有アセンブリを複数のアプリケーションで使用される共通の特別フォルダSxS内にインストールします。
- プライベートアセンブリはアプリケーションと一緒に配布され、アプリケーションフォルダに置かれます。アセンブリが¥system32 フォルダを使用することはXP、Server2003、Vistaプラットフォーム・ファミリーでは禁止されています。

共有アセンブリとしてのランタイムのインストール

ランタイムを共有アセンブリとしてインストールするには、デプロイするシステム上にMSI 3.0がインストールされている必要があり、また、ユーザーが管理者権限を持っていなければなりません。アプリケーションがエンベデッド版Firebirdと一緒にデプロイされていて、これが不可能となる場合がしばしばあります：エンベデッドサーバーはインストールされていて、実行可能な状態でなければなりません。この場合、ランタイムを共有アセンブリとしてインストールすることは計画しないで下さい。

プライベートアセンブリとしてのランタイムのインストール

MSVC8ランタイムライブラリをプライベートアセンブリとしてインストールするには、その内容物-上記の三つのDLLとアセンブリのマニフェストファイルMicrosoft VC80.CRT.manifest -を、これらに依存するバイナリ（.exeまたは.dll）が存在する全てのフォルダに置いておく必要があります。それらのバイナリには、コンパイル時に使用されたライブラリと同等のランタイムの配置が期待されるフォルダに対するチェックが組み込まれているためです。

典型的なエンベデッド版Firebirdのインストールでは、MSVC8ランタイムアセンブリの完全なコピーが三つ必要になります：アプリケーションフォルダに一つ、¥intlフォルダと¥udfフォルダにそれぞれ一つずつ。インストールの膨張問題を避けるため、バージョン2.1.2向けFirebirdバイナリの一部のビルド方法にいくつかの変更が加えられています。（トラッカー・エントリーCORE-2243も参照して下さい）。

これらは、アプリケーション構成がMSVC8ランタイムアセンブリを組み込んでいない場合でもエンベデッド版Firebirdが動作できるようにするための変更です：

a. ib_util.dll、fbudf.dll、ib_udf.dll、fbintl.dllはエンベデッドマニフェストなしでビルドされています。その効果として、ローダーが対応するDLLと同じフォルダでMSVC8アセンブリを検索するのを避けることができます。ホストのプロセスは、これらのセカンダリDLLをロードしようとする前に、マニフェストを介してMSVC8ランタイムをロードし終えていなければなりません。

b. 現在のfbembed.dllには、必要となる可能性がある任意のセカンダリDLLをロードする前に、自身のマニフェストからアクティベーションコンテキストを生成しアクティベートするためのコードが含まれています。

> 注意
>
> a. MSVC8ランタイムをインストールするMicrosoftの再配布パッケージの利用を強く推奨します！インストール実行ファイルvcredist_x86.exe またはvcredist_x64.exe （あなたが選択したFirebirdに適したもの）がフルインストール用またはエンベデッド版用のzipファイル一式の中にあるはずです。存在しない場合も、Microsoftのダウンロードサイトから落としてくることができます。
>
> b. MSVC8ランタイムアセンブリがプライベートアセンブリとしてインストールされる場合、サードパーティ製UDFは以下の条件のうち一つを満たしている必要があります。UDFライブラリをコンパイルする際に、MSVC8ランタイムは次のいずれかとなります：
>
> - 使用されない
> - 使用されても、エンベデッドマニフェストなしでビルドが行われる
> - 使用され、エンベデッドマニフェストがビルドに使われる-MSVC IDEのデフォルトオプション。この場合、MSVC8アセンブリは同じフォルダになければなりません

Chapter 15

互換性問題

Dmitry Yemanov

バージョン2.0またはバージョン2.1.xデータベースからFirebird 2.5に移行する際、既存のデータベースまたはアプリケーションに影響する可能性がある多くの非互換性に注意する必要があります。あなたがこれらの問題を解決するまで、移行の開始はお勧めできません。

> 古いクライアントとの非互換性
>
> 64bitサーバーで統計ルーチンが適切に動作する新しいストラクチャーで、32bitツールが正しく動作できるようにするため、いくつかの新しい内部API関数（struct perf64とperf64_xxx）を導入し、またこれらを使用するようにisqlとqliを変更する必要がありました。このことは、バージョン2.5のisqlとqliプログラムが古いFirebirdクライアントと互換性がないことを意味しています。
>
> 詳細は、エンジンの章のデータベースの統計が64bit値で適切に動作するようにの項を参照して下さい。

Unicodeメタデータへの影響

あなたがこれまでにデータベースのメタデータ中のテキストオブジェクトをUTF-8キャラクタ・セットへと更新していない場合、バージョン2.5以前のデータベースのリストアは“不正な形式の文字列”のエラーを起こして失敗します。この問題の解決には、あなたがインストールした/misc/upgrade/metadata ディレクトリ内のファイル群に注意し、また、gbakコマンドラインで新しい-fix_fss_data と-fix_fss_metadata スイッチを使う必要があります。

設定パラメータの削除

非推奨の設定パラメータOldParameterOrderingとCreateInternalWindowは今後サポートされず、firebird.conf からも削除されました。

以前のバージョンではLockSemCountとLockSignalという二つのパラメータを使ってロックマネージャの調整ができましたが、新しいロックマネージャの実装では必要なくなり、削除されました。

パラメータMaxFileSystemCacheはFileSystemCacheThresholdにリネームされました。

SQL言語の変更

SQL言語の実装にいくつか変更が加えられましたので、注意する必要があります。

予約語

いくつかの新しい予約語が導入される一方で、予約語リスト全体は劇的に縮小され、Firebirdのパーサ文法は、従来は予約語だった大部分の非標準キーワードを非予約語と見なすようになりました。従来からの予約語と新たに予約語となったSQL標準キーワードのリストについては予約語とその変更の章で確認できます。

実行結果

いくつかの変更により、gbakユーティリティのコード（バックアップとリストア）実行時に実行されるものを含むクエリのランタイム実行時に例外を発生させるようになりました。

不正な形式の文字列

UNICODE_FSS文字列とテキストBLOBに対して適格性のチェックが実行されるようになりました。これにより、新規のまたは既存のUNICODE_FSSが不正な形式だった場合、実行時に例外が発生します。

SET句でのロジックの変更

従来は、UPDATE文のSET句が新しい値をカラムに割り当てる際、ただちに新しい値が元の値に置き換わっていました。同じカラムが割り当てられたり、一度以上の割り当てがあった場合、現在の値は最後に行われた割り当ての値となっていました。言い換えると、従来は割り当ての順序が重要となっていました。

Firebirdを標準に合わせるため、このバージョン以降、SET句では、カラムのもとの値だけが任意の割り当てに対してアクセス可能となっています。

しばらくは、firebird.conf 内の一時的なパラメータOldSetClauseSemanticsの設定によって従来の挙動に戻すことができます。このパラメータは将来のリリースで非推奨となり、廃止されます。

ユーティリティ

Firebirdコマンドラインユーティリティに加えられた以下の変更の影響に注意して下さい。

fb_lock_print

バージョン2.5ではサーバー上の各データベースのセパレートロックストラクチャーが維持されているので、fb_lock_printがロックテーブルを表示するためにデータベースのパス名を要求するようになっています。コマンドラインで新しいスイッチ-d <path name>を付けて、分析したいデータベースのファイルシステム・パスを指定して下さい。

将来廃止予定の非推奨機能

Firebird 3コードベースから組み込み関数isc_ddlが削除されることを見越して、現在gdefとgpreツールで利用できるいくつかの機能が非推奨となりました–そのため、バージョン2.5では動作するかもしれませんが、Firebird 3では失敗するでしょう。

- gdefはもうサポートされていません。通常のDDLコマンドとしては、代わりにisqlを使用して下さい。
- gpreのプリプロセッシングとして、全てのDDLオペレーションを次のものと置き換えて下さい。

```
EXEC SQL
  EXECUTE IMMEDIATE "..."
```

- 全てのカスタムアプリケーションで、isc_ddlの呼び出しは、SQLのDDL文の要求に置き換えなければなりません。

APIの変更

クライアントライブラリに実装されるアプリケーションプログラミングインターフェース（API）に以下の変更が加えられましたので、注意して下さい。

矛盾するTPBオプションのリジェクト

API関数isc_start_transaction()とisc_start_multiple()が“一緒にあるべきではない”トランザクションパラメータバッファ（TPB）アイテムの組み合わせをリジェクトするようになりました。

例えば、ゼロではないwait timeoutはno waitオプションと矛盾します；また、no record versionは任意のトランザクション分離モードと矛盾します。本来のあいまいさについて、いくぶん恣意的で（そして恐らく不正確な）仮定を行う代わりに、エンジンはこのような組み合わせを無効としてリジェクトするようになりました。

詳細は、Firebirdエンジンの変更の章のトランザクションの診断の項を参照して下さい。

SQL_NULL定数の追加

SQL_NULL定数が導入されたことで、OR ? IS NULL句が認識され、“未知のデータ型”の例外を発生させることなく処理され、期待された結果を出せようになりました。このことは、XSQLVAR構造体がこの種のクエリに配置される仕方に影響を与えます。詳細は、DMLの章のSOME_COL = ? OR ? IS NULL Predicationの項を参照して下さい。

セキュリティの強化

以下の変更に注意して下さい。

SYSDBA自動マッピングの廃止（Windows版）

バージョン2.1では、Windows管理者グループのメンバーは、デフォルトでSYSDBAにマッピングされていました。バージョン2.5以降は、SYSDBAのマッピングは新しいSQLコマンドを使ってデータベースごとに制御されます。

```
ALTER ROLE RDB$ADMIN SET/DROP AUTO ADMIN MAPPING
```

詳細は、セキュリティに関する章を参照して下さい。

デフォルトの認証方法（Windows）

Windowsの“信頼された認証”が導入されたバージョン2.1では、デフォルトの認証方法はmixedです。つまり、DPBやSPBは、nativeのFirebirdログインもtrustedのユーザーログインもアクセプトします。そのため、firebird.conf 内のAuthenticationパラメータはデフォルトでmixedを示していました。

バージョン2.5以降、デフォルトはnativeとなりました。mixedまたはtrustedとするには、このパラメータを明示的に設定する必要があります。

トラッカー・リファレンス CORE-2376

サービスへのアクセスパス

Firebirdの以前の一部のバージョンでは、データベース名の引数としてサーバーのフルパスが与えられていれば、SYSDBAパスワードがクライアントとサーバーの両方で同じだった場合、サービスへのリモートアクセスが有効になっていたようです。これらの条件下では、例外がスローされず、アクセスは成功していました。

例えば、リモートのクライアントがリモートサーバーと同じSYSDBAパスワードを持ったFirebirdサーバーを稼動していた場合、gbakへのリモートの呼び出しのための以下の構文が有効となっていたようです。

```
gbak -b -se dbhost:service_mgr dbhost:dbalias
   /var2/backups/mydb.fbk -user SYSDBA -password masterke
```

一部の開発者は長い間これをドキュメント化されていない機能かと思っていましたが、残念ながら、この異常はバグでした。バージョン2.5では、どのような条件の下でも、この構文は例外となります。正しい構文は次の通り：

```
gbak -b -se dbhost:service_mgr dbalias
   /var2/backups/mydb.fbk -user SYSDBA -password masterke

gbak -b -se dbhost:service_mgr d:¥databases¥mydb.fdb
   x:¥backups¥mydb.fbk -user SYSDBA -password masterke
```

プラットフォーム固有の既知の問題

エンジンなどの変更により、一般的な問題以外に、いくつかのプラットフォームで固有の影響が出る可能性があります。既知の問題は以下に記載されています。

MacOSX版

マルチスレッド環境で新しいエンジンを正しく動作させるには、Grand Central Dispatchを使う必要があります。これはMacOSX 10.6 (Snow Leopard) で最初にリリースされたため、MacOSXのユーザーは、Firebird 2.5がMacOSX 10.6以上のバージョンでなければ起動しないことを認識して下さい。

> Note
>
> OSXの以前のバージョンを使いたい場合は、Firebirdも以前のバージョンのものを使う必要があります。

Chapter 16

プラットフォーム・ポート

この章の各項は、Firebirdのメインストリーム以外のプラットフォーム向けポーティングについてのものです。以前に各ポーティングに施されてきた変更・改善についてのメモも添えられています。

HPPA

Linux/HPPA

D. Ivanov
A. Peshkov

HPPA版Linux向けFirebird 2.5.1ポーティングです。

トラッカー・リファレンス CORE-3184

Linux/Alpha

D. Ivanov
A. Peshkov

Alpha版Linux向けFirebird 2.5.1ポーティングです。

トラッカー・リファレンス CORE-3184

IBM eServer z-Series

Linux/s390 (32-bit)

D. Ivanov
A. Peshkov

トラッカー・リファレンス CORE-2625

Linux/s390 (32bit) プラットフォーム向けポーティング。s390xアーキテクチャでビルドされています。

このポーティング向けのパッチでは、prefix.linux_s390xから-DS390Xが削除され、gccより提供される__s390__および__s390x__定義のチェックで置き換えられています。そのため、どちらのポーティングでも同じプレフィックス・ファイルを利用できます。

> Note
>
> s390にはアライメントの制限がありません。

Linux/s390x (64-bit)

D. Horak
A. Peshkov

トラッカー・リファレンス CORE-2559

Linux/s390x (64bit) プラットフォーム向けポーティングです。

Linux/sh4 (Renesas SH)

N. Iwamatsu
D. Ivanov
A. Peshkov

トラッカー・リファレンス CORE-2655

Linux/sh4 (Renesas SH) プラットフォーム向けポーティングです。

Linux/sh4 (Renesas SH) アーキテクチャにはリトルエンディアンのものとビッグエンディアンのものがあります。このポーティングはいずれもサポートしています。SHにはアライメントの制約があります。

HP-UX

HP-UX版のロックテーブルの改善

A. Peshkov

トラッカー・リファレンス CORE-2644

HP-UXを除くPOSIXプラットフォーム向けの全てのポーティングで、ロックテーブルは、最初にそれ用に割り当てられたスペースが使い果たされても伸張できるようになっています。HP-UXのポーティングでは、PA-RISCプラットフォームのハードウェア制限により、同一プロセス内で同一のファイル領域を異なる仮想アドレスへとリマッピングする機能がサポートされていないため、同様の動的リサイジングを実装することができません。そのため、ISC_remap_fileは機能せず、ロックテーブルは伸張できませんでした。

SAS社のチームは、プロジェクトVulcanのSAS版向けコードで、このプラットフォームでのロックテーブルの伸張を可能とするソリューションを開発しました。このソリューションはHP-UX向けFirebird 2.5ポーティングにインポートされています。

Windows 32bitプラットフォーム向けポーティング

のリクエストによる N. Samofatov

トラッカー・リファレンス CORE-2609

通知されていたように、Firebirdバージョン2.1.x以上ではMicrosoft Windows 98、ME、NT4はサポートされません。しかし、Red Soft社は今後もWin98/MEや、ロシア政府各部局でいまだに使われ続けメンテナンスされているNT4といったバージョンのサポートを行うとしており、別々のユニットに切り分けて再実装されたコンパイラコンディションのインクルージョンのパーミッションをリクエストし、これらの古い、あるいは年季の入ったWindowsプラットフォーム向けFirebird 2.5のポートをビルドができるようにしました。

これらのポートはFirebird 2.5のメインストリーム向け配布版には含まれていません！

- Firebirdプロジェクトで配布されているWindowsバイナリにはこのサポートを含まれません。
- これらのどのポートもプロジェクトのQAテストを受けていません。
- これらのポートに関するQAテスト、将来のメンテナンスや開発、またバグフィックスは、使用者の責任となります。

Chapter 17

Bugs Fixed

Firebird 2.5.4リリース

バージョン2.5.4リリース前に以下の改善とバグフィックスが報告されました：

コアエンジン

(CORE-4713)　テーブルに式インデックスを挿入した後のロールバックで“BLOB not found”のエラーが発生していました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4700)　GDS_DROP_DATABASEの実行時にシャドウのテストに誤りがありました。Firebird 3.0からのバックポートにより修正されました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4690)　DISTINCTと非DISTINCTの競合が、検索条件IN()を含むサブクエリの結果に影響することがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4678)　クラシックサーバーまたはスーパークラシックサーバー上のまだコミットされていないトランザクションで、“BLOB not found”のエラーが（誤って）スローされることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4673)　計算項目に基づいた計算インデックスが全てのキーにNULLを格納していました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4670)　制約違反のエラーが取り消される場合がありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4644)　　同時接続による高負荷状況下でデータベースのオープンエラーが発生することがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-4631)　　ロックテーブルをバックアップする共有メモリ領域がリマップされなかった時、ステータスベクターがエラーメッセージ“Lock manager out of room”を返していました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4627)　　ON CONNECTトリガがモニタリングテーブルにアクセスする時に、TIPページロックとモニタリングロックの間でデッドロックが発生することがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4618)　　MERGE文が同じターゲット行を複数回更新し、また、PLAN MERGEが使われた場合、ロールバックが変更を取り消せませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4615)　　クラシック・サーバーで、ある条件下でASTプロセスがループ化し、CPU負荷が100%となってハングアップすることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4599)　　REPLACE()関数がマルチバイトのキャラクタ・セットで正しく機能していませんでした。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-4530)　　DB_KEYによる二つのテーブルの結合が、SQL文の式の順序によっては実行に失敗することがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4475)　　fb_shutdown()の後では、end-of-processクリーンアップは何もすべきではありませんが、クリーンアップ中にファイルを作成できるらしい状況があることが報告されました。これは修正されました。

A. Personが修正しました。

~ ~ ~

(CORE-4384)　テーブルのサイズが65535ポインタページを超えた場合に問題が発生していました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4383)　update_in_place()で、インデックスとBLOBのガベージコレクションが動作していませんでした。

D. Yemanov、D. Sibiryakovが修正しました。

~ ~ ~

(CORE-4382)　ユーザーによるセーブポイントがコミットの際に解放されていませんでした。

D. Sibiryakovが修正しました。

~ ~ ~

サーバークラッシュ

(CORE-4616)　同じコンテキスト変数が別のスレッドから同時にアクセスされている場合にサーバーがクラッシュすることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4588)　クライアントがインデックスのナビゲーションスキャン中に異常切断された場合、スーパーサーバーがダウンしていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4500)　ロックテーブルの共有メモリのリマッピングが失敗した後でFirebirdがクラッシュしていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4075)　計算済みのインデックスで例外が引き起こされた場合にサーバーのバグチェックまたはクラッシュが起きていました。

D. Yemanovが修正しました。

~ ~ ~

ストアドプロシージャ／トリガ言語（PSQL）

(CORE-4604)　　EXECUTE STATEMENTの間にUTF8 varcharのchar_length()サイズが予期せず増加していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-4566)　　実行可能な文、プロシージャ、または関数がメタデータのキャラクタ・セットでシステムフィールドを使用する場合、出力パラメータまたは引数が不正なサイズとなっていました。

A. Peshkovが修正しました。

~ ~ ~

コマンドライン・ユーティリティ

isql

(CORE-4578)　　isqlのINPUTファイルが適切に閉じられていませんでした。

A. Peshkovが修正しました。

~ ~ ~

(CORE-4452)　　AUTODDL=OFFの場合、isqlで二つのコレーションを異なる名前で作成することができませんでした。

A. dos Santos Fernandesが修正しました。

~ ~ ~

nBackup

(CORE-4461)　　nBackupがstderrではなくstdoutにエラーメッセージを表示していました。

A. Peshkovが修正しました。

~ ~ ~

POSIX固有のバグ

(CORE-4624)　　FirebirdはPOSIXのマウントテーブルエントリーでコロン文字の処理に誤りがありました。

A. Peshkovが修正しました。

~ ~ ~

Firebird 2.5.3アップデート1リリース

以下のバグフィックスは、以前のリリースのスーパーサーバーとスーパークラシックサーバー・モデルで発見された脆弱性に対処したものです：

(CORE-4630)　　サーバーが不正な形式のネットワークパケットによりセグメンテーション違反を起こし、クラッシュすることがありました。

A. Peshkovが修正しました。

~ ~ ~

Firebird 2.5.3リリース

バージョン2.5.3リリース以前に修正されたものとして、以下の改善とバグフィックスが報告されました：

コアエンジン

(CORE-4460)　　いくつかの組み込み関数を含む式がうまく最適化されないことがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4444)　　物理バックアップ中のout-of-disk-space状態で、エンジンがハングアップし、全てのアタッチメントをブロックすることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2648)　　nBackupの増分ファイルへの書き込みがデータベースのForced Writesの設定を無視していました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4433)　　誰かが読み取りを行っている場合、グローバルな読み取り／書き込みロック(GlobalRWLock) がEXロックをSHへとダウングレードできませんでした。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4372)　　二つのデータページが相互参照するレコードフラグメントを含んでいる場合、デッドロックが起こることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4353)　　ソーティングレコードが必要以上に大きくなっていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4302)　　降順インデックスでの検索（またはスキャン）が、いくつかの検索キーで非常に非効率となることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4299)　　EXTRACT()関数でWITH THE EXTRACTを使用した場合、エラー“Inappropriate self-reference of column”が発生することがありました。

~ ~ ~

(CORE-4283)　　複数イベントの一斉登録中にエラー“Resource temporarily unavailable”が発生ことがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-4251)　　Guardianサービスがイベントログのメッセージの末尾の後に不要なデータを書き込むことがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4250)　　Guardianでプロセスのシャットダウンの時にアクセス違反が発生することがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4216)　　TRIGGER ON TRANSACTION COMMITでメモリリークが起こることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4214)　　グローバル一時表（GTT）が、できないはずの永続的リレーションへの参照ができていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4211)　　シャットダウンプロセスのタイムアウトとfirebird.log に書き込まれる無効なmutexに関するエラーで、エンベデッドエンジンが終了中に5秒間ハングアップすることがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-4198)　　SQL文が16進数リテラルで終わっている場合に不正な“token unknown”のエラーが発生していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-4145)　　ドメインを使用するEXECUTE BLOCKのプリペア中にメモリリークが起きていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-4144)　　UNIONを用いたクエリのプリペア中にエラー“context already in use (BLR error)”が発生していました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4143)　　fbembed.dll でメモリリークが起きていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4139)　　計算インデックスのマッチング中にエラー“invalid stream”が発生する場合がありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4135)　　スーパーサーバーで、スイープが複数のアタッチメントが同時に確立するのをブロックしていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4134)　　自動スイープが開始された時に競合状態が発生することがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4118)　　派生フィールドまたはビュー・フィールドに式インデックスが使われないことがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4117)　　COMPUTED BYフィールドが、例外パラメータとして直接使用された場合、NULLとして評価されていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4113)　　EXECUTE BLOCKのプリペアに失敗していました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4102)　　OR句がunionに適用されている場合の最適化が不十分でした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4101)　　無効な“I/O error during write operation”エントリーが、そのようなデータベースエラーがない場合にもfirebird.log に現れていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4100)　　自動スイープが必要のない時に起動することがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4083)　　COALESCE（またはIIF）を使った派生テーブルでの完全な外部結合が正しくNULLを返していませんでした。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-4058)　　サーバーにリモートスタックバッファオーバーフローがありました。

A. Peshkovにより修正され、特別リリースFirebird 2.5.2アップデート1としてリリースされました（詳細は下記details below）。

~ ~ ~

(CORE-4050)　　SQLダイアレクトがセキュリティ・データベースへの内部接続に設定されていませんでした。

A. Peshkovが修正しました。

~ ~ ~

(CORE-4051)　　32KB以上のレコードのソートを行った場合にメモリリークが起きていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4038)　　格納されたDBKEYの最適化に失敗していました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4036)　　長く圧縮不能なデータをテーブルに格納しようとした場合にバグチェックまたはデータベースの破損が起こることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4005)　　再帰CTEの失敗が誤ったエラーメッセージを返していました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4004)　　時間のかかる操作が時々、非同期シャットダウン／キャンセル・リクエストで中断されないことがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4002)　　データベース・トリガON COMMIT TRANSACTIONにエラーメッセージ“index unexpectedly deleted”が現れることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3989)　　多数の同時ソートが実行されている時にパフォーマンスの悪化またはレスポンスの遅延が起きていました。

D. YemanovとV. Khorsunが修正しました。

~ ~ ~

(CORE-3981)　　ビューからのselect実行中に演算子のチェックが最適なものとなっていませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3978)　　スイープ中に無効なトランザクション・カウンターがfirebird.log にレポートされることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3940)　　ファイルXXXXへのGetFileSize操作中にI/Oエラーが起きていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3924)　　グローバル一時表（GTT）が最低一つのリード・コミッティッドな読み取り専用トランザクションが含まれている場合に並行した変更が行われると、バグチェック291（cannot find record back version）が起きていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3921)　　エラー“record disappeared (186), file: vio.cpp line: 408”が発生し、CPU負荷100%となることがありました（bugcheckabort=1で、スイープがgap ~21000で開始した場合）。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3916)　　大きなテーブルで誤った検証のエラーが発生することがありました。すなわち、“Index x is corrupt (missing entries) in table ...”

V. Khorsunが修正しました。

~ ~ ~

(CORE-3902)　　派生フィールドがインデックスの使用で最適化されないことがありました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3874)　　左結合の結果セットで存在しない行にも関わらず計算項目が現れていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3817)　　ゼロ以外の遅延の値が指定された場合、データベースの強制シャットダウンが動作しませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3360)　　UPDATE ... RETURNING ... がRETURNING句だけに存在するカラムに対してエラー-551（更新許可がありません）を引き起こしていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3305)　　無効なトリガを作成・変更すると、その後に“BLOB not found”のエラーが起きていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2848)　競合が比較的高い場合、“lock conversion denied”または“lock denied”のエラーが起きていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2648)　nBackupの増分ファイルへの書き込みがデータベースのForced Writesの設定を無視していました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2165)　厳密な不等価条件が使われている場合、不必要なインデックスの読み込みが起こることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1604)　ユーザーの名前に非asciiキャラクタが含まれている場合、データベースの作成時にエラーが起きていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1475)　アクティブなアタッチメントを持つデータベースが、データベースがシャットダウンした後でもgbakのバックアップファイルで置き換えられないことがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1295)　DB_KEYを持つクエリの最適化が不十分でした。

D. Yemanovが修正しました。

~ ~ ~

改善点

(CORE-4445)　物理バックアップ（ALTER DATABASE BEGIN/END BACKUPまたはnBackupユーティリティを用いるもの）で、バックアップ状態がストールドからマージに切り替わる時、メインのデータベース・ファイルの伸張が高速化されるよう改善されました。

V. Khorsunが実装しました。

~ ~ ~

(CORE-4443)　fallocate()をサポートするLinuxシステムでファイルの高速な伸張が可能になりました。

V. Khorsunが実装しました。

~ ~ ~

(CORE-4432)　アロケーションテーブルが初めて読み込まれる時にアタッチメントが他をブロックすることがなくなりました。

V. Khorsunが実装しました。

~ ~ ~

(CORE-4431)　物理バックアップ状態がストールした時のアロケーションテーブル・ロックの競合が低減されました。

V. Khorsunが実装しました。

~ ~ ~

(CORE-4386)　“object in use”のエラーに関するレポートが詳細になりました。

D. Yemanovが実装しました。

~ ~ ~

(CORE-4252)　エラーを起こした文脈を特定しやすくするため、整合性制約検証エラーメッセージのテキスト中にリレーション名が追加されました。

V. Khorsunが実装しました。

~ ~ ~

(CORE-4215)　SET STATISTICS INDEX文を実行しても並行するアタッチメントがブロックされたり遅延したりしなくなりました。

V. Khorsunが実装しました。

~ ~ ~

(CORE-3994)　スイープ終了時のlimboトランザクションのスキャンが改善されました。

D. Yemanovが実装しました。

~ ~ ~

(CORE-3881)　インデックスと制約の違反に関するエラーレポートが拡張され、問題のあるキーの値を含むようになりました。

D. Yemanovが実装しました。

~ ~ ~

(CORE-3704)　現在の接続と現在のトランザクションについての詳細情報を取得するため、SYSTEM名前空間に、新しいコンテキスト変数が追加されました。

追加された変数：現在の接続についてはSYSTEM::CLIENT_PIDおよびSYSTEM::CLIENT_PROCESS、現在のトランザクションについてはSYSTEM::LOCK_TIMEOUTおよびSYSTEM::READ_ONLY。

D. Yemanovが実装しました。

~ ~ ~

(CORE-4438)　　エンベデッドSQL（ESQL）にUPDATE OR INSERT文のサポートが実装されました。

D. Yemanovが実装しました。

~ ~ ~

(CORE-4437)　　エンベデッドSQL（ESQL）にRETURNING句のサポートが実装されました。

D. Yemanovが実装しました。

~ ~ ~

(CORE-4047)　　外部関数（UDF）への入力パラメータ数の上限が15まで拡大されました。

A. dos Santos Fernandesが実装しました。

~ ~ ~

サーバークラッシュ

(CORE-4319)　　トレース設定に“connection_id=NN”の行が含まれている時に存在しないデータベース／エイリアスへの接続が試みられた場合、エンジンがクラッシュすることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4300)　　バッファサイズ = 0でiscDatabaseInfo()が呼び出された場合、サーバーが異常終了していました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4267)　　データベースのスイープ中にサーバーがクラッシュすることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4212)　　グローバル一時表（GTT）で外部キー制約を削除するとサーバーがクラッシュしていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4127)　　サイズオーバーのキーに遭遇した時、エラー“key size exceeds implementation restriction”をレポートせずにサーバーがクラッシュしていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4123)　　文字列を大文字に変換するトリガから呼び出されたストアドプロシージャを実行する時にサーバーがクラッシュすることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4121)　エンジンがUDFから呼び出されたBLOB GET/PUT関数内でシャットダウンする時にセグメンテーション違反が起こることがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-4093)　巨大な数値を文字列に変換している間にサーバーがクラッシュしていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-4045)　データベースのシャットダウン中にサーバーがクラッシュすることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3996)　制限されたパスでデータベースを作成しようとした時にFirebirdがクラッシュしていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3993)　進行中のアタッチメントを持つデータベースのシャットダウン中にサーバーが終了するかクラッシュすることがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3988)　トレースまたは監査がアクティブな時にエンジンがクラッシュすることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3979)　自律型トランザクションで変更をロールバック中にサーバーがクラッシュすることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3944)　アタッチメントを削除してオフラインでデータベースを移動させるスクリプトを実行すると、サーバーがクラッシュしていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3908)　　多数の自律型トランザクションが開始・終了している時にエンジンがメモリリークを起こしてクラッシュすることがありました。

A. Peshkovが修正しました。

~ ~ ~

ストアドプロシージャ／トリガ言語（PSQL）

(CORE-4247)　　DELETE WHERE CURRENT OF {カーソル} が新しく追加されたフィールドを持つテーブルで失敗していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-4244)　　DOS864キャラクタ・セットの連結したテキストを含むプロシージャの作成に問題がありました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-4233)　　宣言済みのカーソルを持つPSQLモジュールで、エンジンが誤った変数に値を割り当てることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4210)　　出力パラメータに付けたコメントがプロシージャの変更後に保存されていませんでした。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-4204)　　IF (x = (SELECT ...)) 文を含むプロシージャのコンパイル時にエラーが発生していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3054)　　ユーザーのロールが外部のEXECUTE STATEMENT呼び出しを通じて渡されていませんでした。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3998)　　パラメータ化されたEXECUTE STATEMENT呼び出しが失敗することがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3895)　　PSQLコードが修正されたデータを含むストアドプロシージャからselectを実行する時に高いメモリ使用量を示していました。

V. Khorsunが修正しました。

~ ~ ~

国際言語のサポート

(CORE-4136)　　Sharp-Sのキャラクタが誤ってUNICODE_CI_AIコレーションで取り扱われていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-4125)　　COLLATE UNICODE_CI_AIをインデックスなしのWHERE句で使用すると極端に遅くなっていました。

A. dos Santos Fernandes、T. Martirが修正しました。

~ ~ ~

(CORE-3949)　　ICU 49でUNICODEコレーションが動作しませんでした。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2912)　　小文字のy-トレマ（ISO8859_1コード0xFF）を含む文字列を大文字化する時に例外が発生していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

コマンドライン・ユーティリティ

gfix

(CORE-4297)　　RDB$TRANSACTIONSでのlimboトランザクションの説明が1KBを超えた場合にgfixがクラッシュしていました。

V. Khorsunが修正しました。

~ ~ ~

fbsvcmgr

(CORE-4298)　　fbsvcmgrがsts_record_versionsなどのstsスイッチの認識に失敗していました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3969)　　fbsvcmgrをaction_trace_start、_list、_stop付きで何度も繰り返し実行すると、メモリリークが起きていました。

V. Khorsunが修正しました。

~ ~ ~

fbtracemgr

gbak

(CORE-4417)　　gbakがドイツ語のウムラウト付きのキャラクタを含むインデックスまたは主キーをコミットすることができませんでした。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3995)　　バージョン2.5.2、スイッチ-Vと-Yが一部で失敗していました。

A. Peshkovが修正しました。

~ ~ ~

gsec

(CORE-3932)　　gsecを使用した場合、ユーザー名にダブルクォートを含むユーザーの作成はできましたが、削除ができませんでした。

A. Peshkovが修正しました。

~ ~ ~

qli

(CORE-4327)　　qliでNULLのBLOBをデータベース間でコピーするとエラーが起きていました。

A. Peshkovが修正しました。

~ ~ ~

isql

(CORE-4137)　　isqlが-extractから誤ったメタデータを返していました。具体的には、データ型とPSQL変数定義のキャラクタ・セット引数の間にスペースを入れておらず、そのため、出力スクリプトで構文エラーを引き起こしていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

データベースの監視／管理

(CORE-4010)　　取得操作をDELETE FROM MON$STATEMENTSを介して後から中断することができませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3977)　　DELETE FROM MON$STATEMENTSが、そこそこ長い取得を行っているSQL文を中断しませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3935)　　同じマシンの別のデータベースにDELETE FROM MON$ATTACHMENTSを発行した後に、TCP/IPを介したデータベースへの接続ができなくなっていました。

A. Peshkovが修正しました。

~ ~ ~

トレース／監査

(CORE-4225)　　データベースレベルのトリガを持つデータベースで活動のトレースを行おうとすると、サーバーがクラッシュすることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-4094)　　トレースの出力で誤ったパラメータの順序が示されていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3970)　　POSIXで、トレースが不正確なタイマーを使用していました。

A. Peshkovが修正しました。

~ ~ ~

サービスマネージャ

(CORE-4303)　　サービスの破棄中に競合状態が起こることがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-4224)　　サービスAPIを通じたデータベースの置換に失敗していました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3942)　　サービスAPIを用いたgbakバックアップからリストアを行う場合のエラーメッセージでいくつかの問題が明らかになりました。

A. Peshkovが修正しました。

~ ~ ~

Windows固有の問題

(CORE-3243)　　CURRENT_USERとMON$USERが“信頼された認証”でエラーを起こしていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3183)　　fbembed.dll が自身と同じフォルダからicuin30.dll　　をロードしていませんでした。

V. Khorsunが修正しました。

~ ~ ~

Improvements

(CORE-4439)　　Windows上のスーパーサーバーとスーパークラシックサーバーへの最大接続数が1024から2048に拡大されました。

P. Beachが実装しました。

~ ~ ~

POSIX固有の問題

(CORE-4031)　　Debian Ubuntu 64のmake installに誤りがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-4011)　　Red Hat、Mageiaで稼働するスーパークラシックサーバーとスーパーサーバーで、開始／停止状態が検出されていませんでした。

P. Makowskiが修正しました。

~ ~ ~

Firebird 2.5.2セキュリティアップデート1、2013年3月

(CORE-4058)　　2013年3月にFirebirdサーバーでのリモートスタック・バッファオーバーフローが見つかりました。これにより、認証されていないユーザーがサーバーをクラッシュさせたり、リモートからコードを実行することが可能となっていました。

ビルド番号23539以下の全てのFirebirdバイナリと、2013年3月8日より前の全てのスナップショットにはこの脆弱性があります。

A. Peshkovが修正しました。

~ ~ ~

Firebird 2.5.2リリース

バージョン2.5.2リリース以前に修正されたものとして、以下の改善とバグフィックスが報告されました：

改善点

(CORE-3727)　　改善点 :: FirebirdのビルドシステムにCプリプロセッサフラグのサポートが追加されました。

A. Peshkovが実装しました。

~ ~ ~

(CORE-3656)　　改善点 :: トレースAPIでスイープの情報がサポートされました。

V. Khorsunが実装しました。

~ ~ ~

(CORE-3598)　　改善点 :: TRACEが、トランザクション終了後に起こるアクションの統計情報を生成するようになりました。

V. Khorsunが実装しました。

~ ~ ~

(CORE-3539)　　改善点 :: TRACEに、実行時に発生したエラー（ロックコンフリクトやキー違反など）のログを取る機能が提供されました。

V. Khorsunが実装しました。

~ ~ ~

(CORE-2668)　　改善点 :: 自動スイープの開始時、firebird.logにログが書き込まれるようになりました。

V. Khorsunが実装しました。

~ ~ ~

(CORE-2666)　　改善点 :: リモートのバックアップ／リストアを実行するAPIが利用できるようになりました。

See also /doc/README.services_extension　in your Firebird installation.

A. Peshkovが実装しました。

~ ~ ~

コアエンジン

(CORE-3877)　　ビッグエンディアンのプラットフォーム上で、関数CHAR_TO_UUIDとUUID_TO_CHARが正しく動作しませんでした。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3855)　　GLOBAL TEMPORARY TABLE ON COMMIT DELETE ROWSに書き込まれるBLOBが、既存のデータページに十分な空き領域がある場合でも、新しく割り当てられたページに配置されることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3853)　　Firebird 2.5.1で作成されたデータベースをFirebird 2.5.0で使用する場合、“IS NULL”句に異なる評価が与えられることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3844)　　検証が、特定の場合にインデックス破損の検出に失敗していました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3841)　　行が挿入された際のデータベースの破損が放置されることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3825)　　トランザクションがisc_tpb_autocommitオプション付きで開始された時に、EXECUTE STATEMENTを用いたDDLを起動しようとすると、バグチェック287（“too many save-points”）を生成していました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3799)　　AUTONOMOUS TRANSACTIONオプションと一緒に使うとWITH CALLER PRIVILEGESオプションが動作していませんでした。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3792)　　バッチ挿入のパフォーマンスが低下していました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3791)　　エンジンがRAMの容量より大きなデータベースで活発に作業している時にパフォーマンスが低下していました。

N. Samofatov、D. Yemanovが修正しました。

~ ~ ~

(CORE-3761)　　BLOBをEXCEPTION文の引数として使用した時に変換エラーが起こることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3730)　　関数isc_dsql_exec_immed2()がRETURNING句に入力パラメータの値を渡す際にそれを失ってしまいました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3722)　　IS NOT DISTINCT FROM NULLがインデックスを利用できる場合にそれを使っていませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3697)　　UNION句を含むビューからselectを実行する場合に文字列切り捨てのエラーが発生することがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3692)　　UNIQUE制約に関わっているカラムでNOT NULL制約を削除できませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3690)　　いくつかの曖昧なクエリに対して誤った警告メッセージが返されていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3677)　　外部関数がエンベデッドモードで使われた時に発生することのあるバグによりユーティリティがエントリーポイントをエクスポートするのを防ぐ必要がありました。

> Note
>
> ISC APIをエクスポートする必要があるスーパーサーバーのバイナリは唯一の例外です。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3675)　　CREATE INDEXが複合インデックスでNULL値と空の文字列を同じもののよう扱っていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3631)　　NULL値を持つ重複レコードが正しくチェックされていませんでした。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3610)　　一意なインデックスに重複キーを書き込むことが可能となっていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3601)　　トリガを持つビューでTEXT BLOB charsetの不正な変換が起きていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3599)　　システムロールRDB$ADMINの削除が可能になっていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3579)　　計算項目が後から作成された別のカラムに依存している場合にテーブルを削除できませんでした。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3557)　　削除中のテーブルに対するクエリのプリペア中にサーバーがクラッシュしていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3490)　　名前付きカーソルの使用中に並列処理上の問題が起こることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3238)　　GEN_UUID()にRFC-4122準拠のバイナリUUIDを返させること。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3092)　　ROW_COUNTが単体でのINSERT文の前にクリアされていませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2457)　　UNICODE_CIコレーションが内部gdsソフトウェアの整合性チェックを引き起こしていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1997)　　マルチレベルのコレーションを使用するマルチセグメントのインデックスへの外部キー処理が失敗していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1992)　　エラー“bad BLR -- invalid stream for union select”が不適切にスローされていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-927)　　ビュー内で使用されるプロシージャで、権限付与が動作しませんでした。

D. Yemanovが修正しました。

~ ~ ~

サーバークラッシュ

(CORE-3884)　　トレースが有効な場合、空のクエリのプリペア中にサーバーがクラッシュすることがありました。

D. Starodubov、V. Khorsunが修正しました。

~ ~ ~

(CORE-3873)　　キャッシュへの書き込み中にディスクI/O違反が起こった場合、シャドウへの切り替え中にサーバーがクラッシュすることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3834)　　派生テーブルでのNATURAL JOINの使用がサーバーをクラッシュさせることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3814)　　Forced Writes offでのデータベースのシャットダウンの実行中にスーパークラシックサーバーがクラッシュすることがありました。

V. Khorsun、D. Yemanovが修正しました。

~ ~ ~

(CORE-3636)　　トレースAPIがバージョン2.5.1のサーバーをクラッシュさせていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3627)　　一意なインデックスを持つテーブルに行を挿入すると、アクセス違反でサーバーがクラッシュすることがありました。

A. Peshkovが修正しました。

~ ~ ~

データ操作言語

(CORE-3807)　　マルチバイト接続の際にGROUP BY句内で文字列リテラルが使われている場合、エラー“Invalid expression in the select list”が予期せず発生することがありました。

D. Yemanov、a. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3806)　　ORDER BY句内でサブクエリまたは計算フィールドがベーステーブルを参照している場合に誤ったデータが返されていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3777)　　GROUP BYを使っている場合に予期せぬ“conversion error from string”のエラーが起こることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3683)　　再帰クエリがエンベデッドなGROUP BY句を含んでいる場合、誤った結果が生成されていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3736)　　いくつかのテーブルで、読み取り専用権限を持つユーザーがWITH LOCK句を使い、そのテーブルの更新をブロックすることができていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3611)　　同じカラム名が繰り返し現れるCTEまたは派生テーブルからの取得中に誤った結果が起こることがありました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

コマンドライン・ユーティリティ

fb_lock_print

(CORE-3686)　　fb_lock_printの出力での“acquire blocks”と“mutex wait”のカウンターで不正な値（ゼロ）がレポートされていました。

D. Yemanovが修正しました。

~ ~ ~

fbsvcmgr

(CORE-3658)　　fbsvcmgrユーティリティがISC_USER環境変数の値ではなくOSユーザー名でサーバーに接続していました。

A. Peshkovが修正しました。

~ ~ ~

fbtracemgr

(CORE-3770)　　fbtracemgrユーティリティが全く活動していない時に55%ほどまでのCPU負荷を掛けていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3769)　　fbtracemgrがCtrl-Cで中断された場合にメッセージ“Unknown tag (4) in isc_svc_query() results”が現れていました。

A. Peshkovが修正しました。

~ ~ ~

gbak

(CORE-3875)　　gbakがパラメータの正しいチェックに失敗しており、-B “:”を指定した場合、ランダムなデータベースをバックアップしていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3802)　　Firebird 2.5.1で、データベースのバックアップからのリストア中にメモリを使い果たすことがありました。

A. dos Santos Fernandes、D. Yemanovが修正しました。

~ ~ ~

(CORE-3733)　　gbakリストア中にシステムジェネレータの修復に失敗していました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3649)　　gbakが、すでに正常に閉じられた後でエラーが発生した場合でも、バックアップファイルを削除していました。

A. Peshkovが修正しました。

~ ~ ~

gsec

(CORE-3762)　　gsecがいくつかのエラーに対してゼロのリターンコード（“no error”）を返していました。

A. Peshkovが修正しました。

~ ~ ~

isql

(CORE-3810)　　isqlで、“-pag 0”コマンドスイッチがSET HEADING命令を含むSQLスクリプトで使われた場合、ゼロ除算またはコアダンプが起きていました。

V. Khorsunが修正しました。

~ ~ ~

データベースの監視／管理

(CORE-3625)　　MON$IO_STATSが（ASTレベルで）非同期で実行されたページ書き込みを報告していませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2286)　　PSQLモジュール内でのMON$CALL_STACKからのselectが、時々ゼロの行を返していました。

D. Yemanovが修正しました。

~ ~ ~

トレース／監査

(CORE-3860)　　トレースAPIの不完全なデータベースフィルタがサーバーをクラッシュさせることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3845)　　“重い”クエリが中断された時、その時間がトレース統計にゼロミリ秒として記録されていました。

V. Khorsunが修正しました。

~ ~ ~

サービスマネージャ

(CORE-3612)　　gfix関連のサービスが、isc_service_start()のステータスベクターのエラー値を失うことがありました。

A. Peshkovが修正しました。

~ ~ ~

POSIX固有の問題

(CORE-3912)　　スーパークラシックサーバーで、変数INET_select のGlobalPtr<>が欠けていたため、セグメンテーション違反が起きていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3750)　　さまざまな上限の拡大がPOSIXではエラーを起こすことがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3721)　　マルチユーザーのサーバーで、ISC_変数が設定されていた場合、スタートアップスクリプト（/etc/init.d）がこれらを拾っていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3646)　　Linux上で、FreePascalのマルチスレッドプログラムがバージョン2.5.xのクライアントライブラリを使っている場合にセグメンテーション違反が起こっていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3609)　　オプション-tがfb_inet_server -hによって二度表示されていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3607)　　Solarisで、RLIMIT_NPROC制限が定義されていなかったため、このプラットフォームでのsrc/remote/inet_server.cppのコンパイルに失敗していました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3606)　　Solarisでのコンパイルの問題を回避するため、CまたはC++コンパイラを使うリンカコマンドが、それぞれCFLAGS、CXXFLAGSを適用する必要がありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3605)　　GLOB_OPTIONSがCFLAGSとCXXFLAGSを混在させ、これらがプラットフォーム間で互換性を持つかのように見せていました；しかし、コンパイラ実装の違いにより、両者の違いが排他的となることがありました。

A. Peshkovが修正しました。

~ ~ ~

Mac OS X Lion

(CORE-3911)　　Mac OSXで、APIエントリーポイントBopenとBLOB_open が見えなくなっていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3786)　　Firebird 2.5.1のクラシックサーバー（32bit版）が、再起動直後、isqlでのデータベース作成中にハングアップしていました。そこでisqlをkillして操作をやり直すと今度は成功し、次の再起動まで再び失敗することはありませんでした。この問題は約90%の再現性がありました。

P. Beach、A. Peshkovが修正しました。

~ ~ ~

(CORE-3740)　　標準の64bitバイナリ版で、引数に153以上の要素を持つIN()演算子を使用するSELECTがクラッシュを起こしていました。これは、コンパイラで使われる最適化セット（フラグ-03）の問題であることがわかりました。

P. Beach、A. Peshkovが修正しました。

~ ~ ~

(CORE-3682)　　Firebird 2.5.1が第二のデータベースへのアタッチ中にハングアップしていました。

P. Beach、A. Peshkovが修正しました。

~ ~ ~

リモートインターフェース／API

(CORE-3819)　　データベース接続文字列内のポートアドレスに対するサービス名の解決が誤って実行されることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3812)　　セッション中に膨大な数の主キーが削除および／または変更されている時、クライアントがデータベースへの接続を失うことがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3801)　　ステータスベクターで警告が二度現れることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3778)　　接続のシャットダウン中にアクセス違反が起こることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3732)　　データベースへのアタッチメントを閉じるとセグメンテーション違反を起こすことがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3569)　　CHAR(32767)が長さ32765でXSQLVARに現れていました。

D. Yemanovが修正しました。

~ ~ ~

Firebird 2.5.1リリース

バージョン2.5.1リリース以前に修正されたものとして、以下の改善とバグフィックスが報告されました：

コアエンジン／API

(CORE-3537)　　改善点　::　ON　COMMIT DELETE　ROWSオプション付きで作成されたグローバル一時表に加えられたトランザクションの変更のロールバックの“取り消し”は不必要だったため、削除されました。

V. Khorsunが実装しました。

~ ~ ~

(CORE-3536)　　改善点 :: 他のアタッチメントでアクティブなトランザクションにより、グローバル一時表でのガベージコレクションに不必要な遅延が起きていました。

V. Khorsunが実装しました。

~ ~ ~

(CORE-3457)　　改善点 :: 小さなチャンクの割り当てに関して、テンポラリ・スペース・マネージャに一層の最適化がなされました。

D. Yemanovが実装しました。

~ ~ ~

(CORE-3323)　　改善点 :: ロックマネージャに待機状態をキャンセルする機能が提供されました。

以下の例を考えてみます：

```
tx1: update table t ... where id = 1
tx2: update table t ... where id = 1
```

トランザクションtx2がWAITモードにある場合、tx1の終了までえんえんと待機し続けることになり、この待機状態はDELETE FROM MON$xxxまたはfb_cancel_operationのリクエストを用いても解消されませんでした。

この改善により、ロックマネージャに、このような果てしない待機状態を取り消す機能が提供されました。

V. Khorsunが実装しました。

~ ~ ~

(CORE-3295)　　改善点 :: オプティマイザで実際のレコード圧縮比を推定し、テーブルの濃度（格納されたレコード数）に関するより正確な推測ができるようになりました。

D. Yemanovが実装しました。

~ ~ ~

(CORE-3560)　　バージョン2.5のクラシックサーバーがメタデータのキャッシングの際にバージョン2.1.5より多くのメモリを使用していました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3549)　　セッション終了時にデータベースの破損が起こり、エラー“page xxx is of wrong type expected 4 found 7”がスローされることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3547)　　浮動小数点の負のゼロがインデックスでは正のゼロと同じになっていませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3535)　　nbackupの状態の変更時にエラーが発生した場合、不正なページの書き込み対象が未定義となることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3533)　　スーパーサーバーで、クライアントのアプリケーションによってプリペアされキャッシュされた全てのSQL文のハンドルを明示的にリリースせずに接続が終了された場合に、メモリリークが起きていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3532)　　トレースの実行中に新たなセッションの開始されるとサーバーがハングアップしていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3525)　　自律型トランザクションが"親"トランザクション実行時のフラグを誤って継承していました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3515)　　インデックスが更新されて同期が外れ、エントリーを失わせる複合的な条件下で、インデックスの破損が発生することがありました。これは検証によってピックアップされ、"missing entries"のメッセージがfirebird.logへと書き込まれていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3512)　　トレースの実行中にサーバーがハングアップすることがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3509)　　ALTER PROCEDUREが既存のものと同じ名前を持つパラメータの追加を許可していました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3502)　　DROP VIEWがカラム以外の既存の依存関係を無視していました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3494)　　シャットダウンがfb_shutdown_callback()にインストールされたハンドラによってリジェクトされた後では、アタッチがどうしてもうまく行きませんでした。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3491)　　PSQLモジュールのTYPE OF COLUMNパラメータを変更すると、元のカラムに影響が出ていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3461)　　バックアップ・リストアの後でDDLの操作が失敗していました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3443)　　UDFライブラリの検索中に競合状態が発生することがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3418)　　INACTIVEとして作成されたデータベースのトリガがアクティブになっていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3398)　　GRANT ADMIN ROLEが受け付けられませんでした。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3397)　　intlとトレース・ライブラリに未解決のシンボルが見つかっていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3394)　　一意性制約違反を起こした場合に、ステータスベクターに不要な“lock conflict”エラーが残されることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3341)　　自律型トランザクション内にポストされたイベントが消失し、リスニング中のクライアントに通知されていませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3340)　　空の例外ハンドラを持つ自律型トランザクションで、一意なキーカラムのプライマリに重複した値が挿入され、バックアップをリストアできなくなることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3327)　　ネットワークサーバーのスレッドプールが必要以上のスレッドを作成することがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3326)　　速いmutexが、死んだプロセスによってロックされた状態のままになることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3325)　　高負荷状況下で、新たなプロセスが共有メモリのマッピングに失敗する可能性がありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3315)　　監査プラグインが“正規の”ものの後に第二の　*失敗した* EXECUTE_STATEMENT_FINISH を記録していました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3314)　　プロシージャとそれが依存しているテーブルが同じトランザクションで削除された後で、依存関係が解消されていませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3312)　　従属テーブルがOR句によって親テーブルに依存している場合、結合のplanが最適なものとなっていませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3306)　　インバリアントなサブクエリがバリアントなものとして扱われており、そのため、ネストされたストアドプロシージャに対する複数回の呼び出しが起こっていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3283)　　サブクエリでLEFT OUTER JOINを使うクエリで不正なplanが生成されていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3282)　　EXECUTE STATEMENTがSQLのテキスト引数をパーシングする際に誤ったキャラクタ・セットを使用していました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3266)　　非同期サービスによるデタッチのリクエストと起動中のユーザートレースサービスとの間で競合状態が起こることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3256)　　明示的なplanを持つビューに対する選択クエリのパーシング中にエラー“request depth exceeded”が現れることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3237)　　ストアドプロシージャのコンパイルに時間がかかりすぎていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3207)　　トランザクションの開始時にFB_965910463_Class.isc_start_multiple内にAccessViolationExceptionがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3205)　　isc_dsql_exec_immed2()がエラーコードisc_stream_eofとisc_sing_select_errを、これらのエラー発生時に返していませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3188)　　エラー“Page 0 is of wrong type (expected 6, found 1)”が予期しない場合に起こっていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3176)　　サブクエリから派生したカラムを持つビューが、インデックスを使わずに、サブクエリの出力をテーブルに結合していました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3168)　　トレース機能の<サービス>セクションでexclude_filterが動作していませんでした。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3157)　　ストアドプロシージャにパラメータの説明を加えると、メモリが大量に消費され、実行時間が長くなることがありました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3151)　　sqlda_supに割り当てられたメモリがいつまでもリリースされない場合がありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3148)　　SQZ_apply_differencesで危険なコードが見つかりました。

D. Kovalenko、A. Peshkovが修正しました。

~ ~ ~

(CORE-3140)　　パラメータに付けたコメントがプロシージャの変更後に保存されていませんでした。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3137)　　データを修正している選択型プロシージャで、部分的なロールバックが可能となっていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3131)　　WIN1257_LV（ラトヴィア語）コレーションで、A、E、I、Uの四つの文字に誤りがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3125)　　ルーチンWorker::shutdown()でアクセス違反が起きていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3058)　　前に32,767以上のジェネレータが作成されていた場合、新しく作成されたジェネレータは誤った値を持っていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3029)　　例外ハンドラを含むEXECUTE BLOCKから例外が発生した後のロールバックで、バグチェック“Too many savepoints (287)”が起きていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3024)　　ALTER VIEWの後で、エラー“no current record for fetch operation”が起きていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2835)　　selectに主キー・インデックスを使う必要がある場合に、ナチュラルキーが使われていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2827)　　多くのトリガと間接的に関わる操作のプリペア中に、相互に関連する複雑なメタデータのプリペアに非常に時間がかかっていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2709)　　NULL値を持つ複合インデックスで、インデックス付きの読み込みが過剰に行なわれていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1752)　　異なるコレーションを持つ結合の結果が、使用するexecution planによって違うものになっていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1274)　　PLAN MERGEが選ばれて、等価演算子の引数のデータ型が異なっている場合、結果が誤っていました。

D. Yemanovが修正しました。

~ ~ ~

サーバークラッシュ

(CORE-3554)　　リモートで空のSQLクエリを渡すと、プリペア中にサーバーがクラッシュするか、または不正なパーシング・エラーがスローされることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3557)　　削除プロセスに入っているテーブルに対するクエリのプリペア中にサーバーがクラッシュすることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3524)　　使用中のストアドプロシージャの再コンパイル中にサーバーがクラッシュしていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3503)　　古いバージョンでは正規の文脈となる位置に、新しいバージョンでは不自然な(aggregateまたはunionの) ストリームがある場合に、ALTER VIEWがサーバーをクラッシュさせていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3477)　　存在しないSQLパラメータが渡された場合にサーバーがクラッシュしていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3440)　　isc_que_events()のqueueに入っているイベントが0の場合、サーバーがクラッシュしていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3247)　　BLOBにキャラクタ・セットがUTF8のデータがある場合にサーバーがクラッシュしていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3419)　　自律型トランザクションがロールバックされるとサーバーがハングアップまたはクラッシュすることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3400)　　FreeBSD8.2R上でしばしばサーバークラッシュが起きていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3374)　　最新のフォーマットではないレコードに対してSELECT WITHが発行された場合にサーバーがクラッシュしたり、データが破損することがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3320)　　特定のMERGE構文でサーバーがクラッシュすることがありました。詳細は公表されていません。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3255)　　GROUP BY句を持つビューを使うとサーバーがクラッシュすることがありました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3219)　　DSQL_unprepareを使うと、トレースマネージャがサーバーをクラッシュさせていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3217)　　複数の接続が同時にアタッチまたはデタッチしている時にロックマネージャ内部でサーバーがクラッシュしていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3202)　　execute_immediate API呼び出しファミリーが、リモートサーバーをクラッシュさせることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3180)　　宣言や選択中にマッチングしないカラムを持つALTER VIEWがサーバーをクラッシュさせていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3138)　　MON$テーブルに、その構造を変更した後でアクセスすると、内部エラーまたはクラッシュが起きていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3064)　　明示的なplanの中でプロシージャの識別子とそのエイリアスの両方を使用すると、サーバーがクラッシュしていました。

D. Yemanovが修正しました。

~ ~ ~

データ操作言語

(CORE-3523)　　SIMILAR TOが降順の範囲で誤ったマッチングを行っていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3493)　　タイムスタンプに'16.11.1858 00:00:01'以前の値を入力すると、エラー"value exceeds the range for valid timestamp"をスローしていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3489)　　union内でのBLOBの変換が時々失敗していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3479)　　ASCII_VAL()関数が空の文字列に対して0を返す所でエラーを起こしていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3355)　　インデックスが使われている場合にDATEとTIMESTAMPの比較が誤っていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3353)　　演算子 (blob_field LIKE ?) がパラメータをBLOBとしてではなくVARCHAR(30)として記述していました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3335)　　マルチバイトBLOBのSUBSTRING関数とその境界引数で内部ラッピングが発生し、誤った結果が出ていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3311)　　パラメータ化したROWS句を用いたUPDATEまたはDELETE文のプリペア中にエラー“data type unknown”がスローされていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3302)　　DISTINCTによる集約で、誤った（重複した）データが返されていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3277)　　キャラクタ・セットUTF8のvarcharで、RIGHT()関数が誤った結果を出していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3245)　　長いテキストからなるBLOBに対するSUBSTRING()の操作で、オプションの三番目の引数がなかった場合、返されたBLOBに32767文字しか含まれていませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3244)　　三番目の引数があった場合には、空の文字列（''）に対するPOSITION()の結果が誤っていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3233)　　第二のオペランドが32KB以上だった場合、LIKE、STARTING、CONTAININGが失敗していました。

A. dos Santos Fernandes、D. Yemanovが修正しました。

~ ~ ~

(CORE-3228)　　マルチバイトのテキストからなる長さ1024バイト以上のBLOBで、RIGHT()関数が失敗していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3227)　　引数がマルチバイトのキャラクタをどこかに含んでいる場合、ASCII_VAL()関数が失敗していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3222)　　“WITH CHECK OPTION”を持つビューが、WHERE句で、TRIM()関数の呼び出しに適合しませんでした。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3211)　　NOT IN条件を含むビューからselectを行ったときに文字列切り捨てのエラーが起きていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3210)　　SELECTクエリで、予期しないエラー“no current record for fetch operation”が発生していました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3208)　　再帰クエリが重大なメモリリークを起こしていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3203)　　UPDATE OR INSERTでRETURNINGを使うと、“Invalid Cursor”のエラーを起きていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3173)　第二のものにGROUP BY句と内部結合がある、二つの共通テーブル式を含むストアドプロシージャからselctを行うと、誤って何も結果を返しませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3164)　クライアントがキャラクタ・セットUTF8を使用して接続している場合、BLOBフィールドを含むパラメータ化されたリクエストが失敗していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3141)　ビューの最後のカラムが値を持っていてもNULLとして返されていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3091)　引数Xが負の数で、Yの値がスケール0でスケーリングされた数値だった場合、組み込み関数POWER(X, Y)が動作しませんでした。

A. dos Santos Fernandesが修正しました。

~ ~ ~

コマンドライン・ユーティリティ

gbak

(CORE-3236)　サービスマネージャスイッチとlocalhost:dbの両方が指定された場合に、gbakが“unavailable database”のエラーをスローしていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3249)　gbak -bの出力ファイルとして使用されていているファイルがすでに存在し、新たなバックアップファイルのサイズが既存のものより小さい場合、バックアップファイルが切り捨てられませんでした。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3232)　“-se service_mgr”スイッチを付けずにnon-transportableなバックアップを行った場合、元のデータベースより約50%大きいバックアップファイルが生成されていました。

A. Peshkovが修正しました。

~ ~ ~

nBackup

(CORE-3521)　　nBackupの増分ファイルの内容がディスクに書き込まれていませんでした。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3482)　　Linux版のnBackupがCtrl-Cでセグメンテーション違反を起こし、データベースがロック状態になり、増分ファイルが増大し続けていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3297)　　firebird.confがない場合、nBackupが終了の通知を行っていませんでした。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3199)　　POSIX版で、リクエストを行なっているユーザーがrootまたは所有者ではない場合、O_NOATIMEフラグが開いていることで、nBackupが失敗していました。

A. Peshkovが修正しました。

~ ~ ~

fbtracemgr

(CORE-3487)　　fbtracemgrユーティリティが、Ctrl-Cでの終了時に、時々セグメンテーション違反を起こいsていました。

A. Peshkovが修正しました。

~ ~ ~

fb_lock_prt

(CORE-3454)　　“fb_lock_print -c”がサーバーをハングアップさせていました。

A. Peshkovが修正しました。

~ ~ ~

gpre

(CORE-3486)　　GPRE言語モジュールがgcc 4.4でコンパイルできませんでした。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3022)　　gpreをGCC 4.4.1でコンパイルするとC++コンパイラ警告が出ていました。

D. Dodsonが修正しました。

~ ~ ~

データベースの監視／管理

(CORE-2305)　　改善点 :: 監視中のスナップショットの間でMON$STATEMENT_ID値を定数としました。

D. Yemanovが実装しました。

~ ~ ~

(CORE-3508)　　MON$DATABASE_NAMEとMON$ATTACHMENT_NAMEフィールドで、どの接続キャラクタ・セットでも、非ASCIIキャラクタがクエスチョン・マークで代替されていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3218)　　SQL文キャンセルのリクエストが、現在実行中のSQLコードによって警告なく無視されることがありました。

D. Yemanovが修正しました。

~ ~ ~

サービスマネージャ

(CORE-3261)　　リストア・サービス実行中にアサーションが実行されることがありました。

A. Peshkovが修正しました。

~ ~ ~

POSIX固有の問題

(CORE-3589)　　MacOSXとFreeBSDで、共有ファイルが閉じられると、セマフォがその共有ファイルからデタッチされず、内部リソースリークを起こしていました。このバグは、ISC_event_init()関数が返すエラーに対するチェックが抜けていたため、たまたま見落とされていたものです。これは、MacOSX　10.7でスーパーサーバーとスーパークラシックサーバーの起動に失敗していたことから明らかになりました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3544)　　Linuxでmake installに失敗していました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3447)　　4.2より上のバージョンのicuを持つLinuxで、いくつかのコレーションがインストールされていませんでした。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3377)　　POSIXでのFirebirdビルド中に、ビルドされたディレクトリではなく既定のディレクトリで、欠けているfbintl.confについての記録をfirebird.logに書き込もうとしていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3259)　　POSIXで、ユーザーコードでCtrl-C（終了）を処理する時にデッドロックやセグメンテーション違反が起きていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3257)　　Linuxで'make install'が失敗していました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3229)　　何らかの原因により、エラー“Operating system directive open failed, Too many links”がfirebird.logに記録されていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3212)　　FreeBSDでのコンパイルがエラーを起こして失敗していました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3194)　　Linuxのスーパークラシックサーバーへの接続数の上限が508になっていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3185)　　すでにFirebirdサーバーが稼働しているPOSIXボックスで、root以外のユーザーによるFirebirdのコンパイルを行うと、テンポラリ・ロックスペースでアクセス競合を起こしていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3166)　　POSIXでクラシックサーバーからスーパークラシックサーバーへとモードを切り替えるためのスクリプトchangeMultiConnectMode.shがスーパーサーバーのインストールに誤って含まれていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3150)　　Linuxで、gbakがCtrl-Cで中断された時にセグメンテーション違反を起こしていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3143)　　Linuxで、ユーザーがgstatを中断した時にセグメンテーション違反を起こすことがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2921)　　FreeBSDで'make install'が動作していませんでした。

A. Peshkovが修正しました。

~ ~ ~

Windows固有の問題

(CORE-3329)　　WindowsのAdministratorsがロールRDB$ADMINを予期せず手にしていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3059)　　WindowsサーバーでRemoteFileOpenAbilityにTrueを設定すると失敗していました。

D. Yemanovが修正しました。

~ ~ ~

リモートインターフェース／API

(CORE-3248)　　改善点 :: メッセージ・バッファ内の未使用のVARCHAR値のバイト数がゼロに設定されるようになりました。

A. Peshkovが実装しました。

~ ~ ~

(CORE-2752)　　改善点 :: SO_KEEPALIVEオプションがクライアントのTCPソケットに設定されるようになりました。

D. Yemanovが実装しました。

~ ~ ~

(CORE-3511)　非ASCIIキャラクタを持つ、クォートで囲まれていないロール名がデータベースパラメータバッファ（DPB）に渡された時、誤って大文字化されていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3389)　ゼロのトランザクションハンドルを持つisc_dsql_exec_immed2がBUGCHECK(147)を起こすことがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3387)　クライアントライブラリが強制的に切断されたサーバーソケットでの応答パケットを待ち続けていつまでもハングアップしていることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3351)　Windowsクライアントが接続時にエラーメッセージ10054をfirebird.logに書き込むことがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3328)　クライアントがデータベースのシャットダウン時にエラーメッセージ“Unsuccessful detach from database”をfirebird.logに書き込んでいました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3220)　API関数isc_info_svc_get_usersが結果のクラスタにエラーメッセージを返していました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3170)　EVENTがポストされてもサブスクライバが存在しない場合、エンジンが無限ループに入ることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3119)　イベント処理に関連したリモートでのプロトコルコードが無限ループを引き起こし、CPU使用率が100%となっていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3095)　　同じトランザクションでイベントが何回ポストされても、クライアントが一回分のイベントしか受け取っていませんでした。

V. Khorsunが修正しました。

~ ~ ~

Firebird 2.5.0リリース

バージョン2.5.0の最終リリース以前に修正されたものとして、以下のバグが報告されました：

コアエンジン／API

(CORE-3115)　　内部レコードの圧縮ルーチンにいくつかのバグがありました。

A. Peshkov、D. Kovalenkoが修正しました。

~ ~ ~

(CORE-3103)　　バージョン2.5の第三のリリース候補で、SELECT文がインデックスなしの読み込みを、バージョン2.1.3での同じ文よりも、より多く行っていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-3101)　　以前のバージョンから移行したデータベースで、ALTER DOMAINができませんでした。

A. dos Santos Fernandes、D. Yemanovが修正しました。

~ ~ ~

(CORE-3100)　　EXECUTE STATEMENTの外部トランザクションで、WAITモードとロックのタイムアウトのパラメータが、対応するローカルなトランザクションのパラメータとマッチングしていませんでした。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3096)　　CORE-2893の修正で入り込んだ不具合のため、SQL文のプリペア中にノードの二重処理が起こり、デバッグ・ビルドでアボートが頻発していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3094)　　選択型ストアドプロシージャからのNOT INでパラメータが動作していませんでした。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3090)　テーブルと派生した定数を返すサブクエリを用いたLEFT JOINの結果が不正なものでした。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-3089)　外部のデータソースがInterBase 4.1 (ODS 8) のデータベースだった場合、そのデータソースに対してEXECUTE STATEMENTを実行しようとすると失敗していました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3079)　バッチ挿入が全て単一のトランザクション中で実行され、イベントをポストしたトリガを含んでいる場合に極端に遅くなっていました。

V. Khorsunが修正しました。

~ ~ ~

サーバークラッシュ

(CORE-3109)　NULLのトランザクションでisc_dql_exec_immed3_m()　がCREATE DATABASE ...のために呼び出された場合、サーバーがクラッシュしていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

コマンドライン・ユーティリティ

gsec

(CORE-3116)　gsecユーティリティがユーザーの出力リストをstdoutではなくstderrに送っていました。

A. Peshkovが修正しました。

~ ~ ~

旧版でのバグフィックス

バージョン2.5開発中に修正されたものとして、以下のバグが報告されました：

コアエンジン／API

(CORE-3034)　式インデックスへのキーの挿入中に、データベースのシャットダウンやユーザーからのリクエストなど、何らかの理由でリクエストがキャンセルされた場合、バグチェック 300 (can't find shared latch) を起こすことがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3016)　接続の切断時に“Fatal lock manager error: invalid lock id (0), errno: 0”がfirebird.log に出ることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3015)　さまざまな“Cannot initialize the shared memory region”のエラーが報告されていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-3003)　SELECT文を介して呼び出されているプロシージャ内にSUSPEND文が存在するかどうかについての新たなチェックは適切に動作していました：プロシージャがSELECT文を介して呼び出された時、SUSPENDが存在しなかった場合、エラー“Procedure ... is not selectable (it does not contain a SUSPEND statement)”がスローされていました。

ところが、同じエラー状態がRESTORE中にも発生し、リストアが失敗することがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2995)　ステータスベクターで、単一のエラーが二回報告されていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2993)　高負荷なシステムで、モニタリングテーブルの使用中に、ロックマネージャの致命的なエラー“Invalid lock id (NNN)”が起こることがありました。

V. Khorsun、D. Yemanovが修正しました。

~ ~ ~

(CORE-2900)　DISTINCTによる集約を含むリクエストを使うと定期的に、ただしランダムなメモリアクセス違反が起きることがありました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2977)　　ODSバージョンが10より前のデータベースの、DATE型のインデックス付きフィールドでサーバーが不正な動作をしていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2965)　　SINGULAR句を含むサブクエリの後で返されたROW_COUNTの値が規則に反していました。そもそもROW_COUNTは更新または削除の影響を受けた行のカウントを意味するものなので、サブクエリからゼロ以外の値が返されるはずがありませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2956)　　プロシージャのパラメータを処理するリクエストで問題が起きていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2943)　　二つの再帰部分を持つ再帰クエリのパーシング中にエラーが起きていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2936)　　二つの連続するリーフインデックスページが、二つの異なる接続によって、インデックスから同時に削除（ガベージコレクション）された場合、別のインデックスページで兄弟ポインタが解放されたインデックスページを参照したままになり、リンクしている兄弟ページのリストが壊れることがありました。解放されたページが再び割り当てられると、“Wrong page type (expected 7 found N)”として、インデックスの破損が報告されていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2916)　　インデックス作成中に変換エラーが発生した場合、エラー処理に失敗していました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2893)　　サブクエリ中の式がインバリアントなものとして扱われ、不正な結果を生成することがありました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2879)　　スイープがエラーpage 0 is of wrong type (expected 6, found 1)を起こすことがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2876)　　ALTER DATABASE ADD DIFFERENCE FILE使用時に、おかしなエラー処理が行なわれていたため、エンジンがデータベースのバックアップモードに関して混乱することがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2875)　　4096バイトより長いCHARカラムと文字列定数との比較を行った場合に、文字列右端の切り捨てエラーが起きていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2871)　　派生テーブルまたはビューが左／右結合とORDER BY句の両方を含み、また、外部クエリもORDER BY句を含む場合、外部のORDER BY句が効果を持ちませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2858)　　セキュリティチェックの失敗を通知する例外が発生した場合にメモリの破損が起こることがありました。

C. Valderramaが修正しました。

~ ~ ~

(CORE-2856)　　キーが削除された場合、一意なインデックス中でNULL以外のキーが見つからないことがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2833)　　データがnullデータフィールドへの参照を含んでいた場合、式インデックスに影響を及ぼすデータの変更に失敗していました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2826)　　UTF-8のデータベースで、結合条件が失敗していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1089)　　ORDER BY句が（メインではない）右側のテーブルのカラムを含んでいない場合、DISTINCTとLEFT JOINを使ったビューからのselectで、誤った順序のレコードが返されていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-195)　　BEFORE UPDATEトリガですでにアクションを起こしているいくつかのレコードを更新した際に、すでにバージョン1.5.1で修正されていた古いバグによる不具合により、バグチェッ

ク291（cannnot find back record version）が起きていました。この不具合がバージョン2.0で再び入り込みましたが、以前ほど破壊的ではなく、物理的にテーブルの最初にあるレコードのみ影響を受けていました。

A. Peshkovが修正しました。

~ ~ ~

（CORE-2822）　サブクエリが自明ではない派生テーブルを含んでいた場合、エラー“no current row for fetch operation”がスローされていました。

D. Yemanovが修正しました。

~ ~ ~

（CORE-2820）　初期に施された大きな修正の副作用として、PLAN ORDERを持つクエリが小さなメモリリークを起こしていました。

V. Khorsunが修正しました。

~ ~ ~

（CORE-2815）　ページインベントリページの変更済み後に変更のマーキングが行なわれていた論理条件の整理。

V. Khorsunが修正しました。

~ ~ ~

（CORE-2785）　BLOBの変換が不適切に処理される問題が特殊な状況下で現れていました。すなわち、オブジェクト定義付きCOMMENTとして保存される1バイトのキリル文字テキストによる入力引数文字列が、変換時に、先頭以外のBLOBセグメントの最大64KBというサイズ制限を超過していました。この余剰バイトは後続のチャンクへと回されるはずのところ、変換エラーを引き起こしていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

（CORE-2783）　再帰クエリがSELECTリスト内のサブクエリとして使われ、その結果がORDER BY基準として指定された場合、アクセス違反が起きていました。

V. Khorsunが修正しました。

~ ~ ~

（CORE-2730）　RISCマシンで、DB_KEYを使った動作中にバスエラーが起こることがありました：（アライメント要求を持たない）リテラルからdb_keyへとキャストする時、QWORD境界ではアライメントが必要であり、強制的に行なわれるべきでした。

A. Peshkovが修正しました。

~ ~ ~

（CORE-2722）　BLOBからNONE/OCTETSキャラクタ・セットでコピーする時に不正な形式のBLOBの保存が許可されていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2660)　外部結合状態でマッチングが見つからなかった場合にCOUNT(*)が誤って0を返していました。現在は正しくNULLが返されます。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2659)　複雑なビューを含む外部結合クエリのplanが、利用可能なインデックスを利用していなかったことで、最適なものとなっていないことがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2640)　ある条件下で、ロックマネージャが通常のデッドロックの検出に失敗しサーバーをハングアップさせることがありました。

V. Khorsun、D. Yemanovが修正しました。

~ ~ ~

(CORE-2635)　一意なインデックスが、多くのNULLキーを含んでいた場合にレベル1で破損することがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2632)　モニタリングテーブル使用時に、予期しない“Invalid BLOB ID”のエラーが起こることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2616)　高負荷下でエラー“page <N> is of wrong type (expected 7, found 5)”が起こり、何かがデータベースを破損したような印象を与えていましたが、再起動すると、破損した痕跡はありませんでした。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2608)　Windowsの最近のバージョン（64bitのXP以降、32bitのVista以降）で、Firebirdで大規模なデータベースを稼働していると、OSが全RAMをファイルシステムのキャッシュに費やして、応答しなくなりました。Windowsがクラッシュする場合もありました。

このイシューはドキュメント化されています。Firebirdでは、ファイルシステムのキャッシュでのRAMの使用量を制御するfirebird.conf中の新たなパラメータFileSystemCacheSizeの実装によって対処しています。

N. Samofatovが修正しました。

~ ~ ~

(CORE-2602)　　キャラクタ・セットNONEを使用するアタッチメントがモニタリングテーブルからの読み込みに失敗することがありました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2591)　　通常のパフォーマンスがしばらく続いた後で、高いmutex待機率とパフォーマンス低下を示し始めました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2581およびCORE-2582)　　ある状況では、内部・外部関数の呼び出しを含む式の結果としての無限大とNANを、エンジンが例外として捉えていませんでした。

C. Valderramaが修正しました。

~ ~ ~

(CORE-2578)　　結合した一つ以上のテーブルを持つビューからRDB$DBKEYをselectすると、変換エラーを返していました。

A. dos Santos Fernandes、A. Peshkovが修正しました。

~ ~ ~

(CORE-2514)　　'temp'ドライブの空きスペースが不十分だった場合に、CreateFileに関するエラーが報告されていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2422およびCORE-2321)　　サーバーがTempDirectoriesで設定された複数の配置間の切り替えを行いませんでした。そのため、最初に設定されたテンポラリ・ディレクトリの空きスペースが不十分だった場合、ソートに失敗していました。たいていの場合、ユーザーは大量のテンポラリ・スペースをリクエストする、ソートまたは他のクエリが'operating system directive write failed. Invalid argument.'のようなエラーを起こして失敗するのを目の当たりにしていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2315)　　FirebirdによるFLOATのサポートはオリジナルのInterBaseの仕様に従っていませんでした。IBのドキュメントによれば、FLOATの値は1.175E-38から3.402E38までの範囲になければなりませんでした。クロスプラットフォームで行われたテストでは、オーバーフローを起こさない最大値は < 3.4E38だと証明されています。

W. Oliverが修正しました。

~ ~ ~

(CORE-1991)　　UDFがBLOBパラメータ付きで宣言された場合、エンジンはRDB$FUNCTION_PARAMETERS.RDB$FIELD_LENGTHにnullを格納していました。これにより、実行時に'message length error'が起きていました。

D. Sibiryakovが修正しました。

~ ~ ~

(CORE-1781)　　サブクエリが集約関数によって別の文脈に指示された場合、エンジンが整合性チェックのエラーをスローしていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2564)　　RISCマシンで、モニタリングテーブル使用時に、バスエラーがスローされていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2550)　　ビッグエンディアンのマシンで、DB_KEYの使用時にバスエラーがスローされていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2538)　　ストアドプロシージャがEXECUTE PROCEDUREを使って呼び出された場合、PSQLが単体のクエリ結果をそのプロシージャへの入力パラメータとして使用するのを許可しませんでした。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2532)　　マルチボリューム・データベースのボリュームサイズが不正に割り当てられていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2526)　　サービスへの接続に関わらず、サーバーがシャットダウンすることがありました。

A. Peshkov、D. Kovalenkoが修正しました。

~ ~ ~

(CORE-2505)　　組み込み三角関数が、NaNや無限大を生成することがありました。

C. Valderramaが修正しました。

~ ~ ~

(CORE-2501)　　バイナリのshift関数が負のshift値を取るという、誤った結果を出していました。

C. Valderramaが修正しました。

~ ~ ~

(CORE-2499)　　DISTINCTアイテムの実装制限が行われておらず、不正なBLRが生成されていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2482)　　データベースがアタッチメントまたはデタッチメントを受け付けた時にモニタリングテーブルによるデータ収集が不安定になっていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2475)　　外部テーブルのデータが、そのテーブルにアクセスする最初のもの以外のクラシックサーバーのセッションから見えなくなっていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2449)　　予期された例外の代わりに予期しない“lock conflict”エラーがスローされることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2444)　　複数のアタッチメントが同時に関心をイベントに登録し、イベントテーブルの空きスペースが使い果たされた場合、エンジンがハングアップすることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2426)　　ALTER TABLEがコレーションを尊重していませんでした。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2416)　　派生テーブルに関する集約を持つクエリの準備がアクセス違反を起こしていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2411)　　オプティマイザが特定のタイプのクエリで、バージョン2.0.4と2.1.1でよりも遅い方のPLANを選んでいました（このバグはバージョン2.0.5と2.1.2にも同様に影響していました）。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2397)　　同じトランザクション内で一つのテーブルの一つ以上のインデックスの削除すると、破損が起こることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2359)　　番号を割り当てる際に、マルチバイト文字列の論理的な最大長が尊重されていませんでした。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2331)　　ALTER DOMAINの結果として、無効なRDB$FIELD_SUB_TYPEの値が格納されていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2272)　　イベント接続の試行をkillした際に、サーバーがガベージを返し始めていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1971)　　WHERE句や他の演算子に、修正済みでドキュメント化もされている通りの評価順序（常に左から右）を設定すること。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1346)　　LPAD()とRPAD()関数が一つのSQL文で一つ以上のカラムに適用された場合、実装制限に掛かることがありました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2356)　　Windowsで、任意の作業プロセスが存在している場合、クラシックサーバーのリスナープロセスが再起動後に必要なリソースを作成できませんでした。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2355)　　結果の文字列のバイト長が縮小した場合、LOWER/UPPERの不正な処理が行なわれていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2351)　　データベースの<ファイル指定>がエイリアスだった場合、そのエイリアスが存在したとしても、そのデータベースを作成できませんでした。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2349)　　“Invalid SQLDA”のエラーが誤ってスローされていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2348)　　トランザクション番号が32bitの符号付整数を超過することから、さらなるデータベース破損の問題が起こりました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2340)　　同時に高い負荷が掛かる状況下で、バグチェック258 (page slot not empty) が起こることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2320)　　複雑な再帰クエリが常に全ての行を返すわけではありませんでした。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2313)　　境界条件により、INF_*関数が最初にisc_info_truncatedを持つ出力バッファの全体を無効化することがありました。

C. Valderramaが修正しました。

~ ~ ~

(CORE-2311)　　WITH RECURSIVEクエリがメモリリークを起こすことがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2300)　　SUBSTRING()の二番目の評価が、予期せぬ算術例外、数値オーバーフロー、または文字列切り捨てのエラーをスローしていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2289)　　外部キー作成中に外部キー違反が起こった時、参照された主キーについて、誤った名前が報告されていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2242) エンジンが整数コンテナを、マシンのローカルなフォーマットの整数で、BLOBパラメータバッファ (BPB) に不正に配置していました。このため、ビッグエンディアンのプラットフォームでエラーが起きていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2241) テーブルに対するALTER TABLE ALTER COLUMN.. の操作が大量の挿入操作の過程で行われた場合、インデックスのわずかな破損が起こり、それ以降のクエリが誤ったレコード数を返す原因となっていました。このバグは、BTR ¥compress_root()のレガシーコードに由来するものでした。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2230) EXECUTE BLOCKの入力パラメータがドメインチェックを受けていませんでした。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2186) Windowsのエンベデッドサーバーで、CREATE DATABASEの処理中に、isc_dsql_execute_immediate()の後でfbintl.dllがアンロードされていませんでした。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2182) 新しい組み込み関数と名前が重複してしまった既存のUDFを削除できませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2154) 最後のレコードがEXECUTE PROCEDUREで取得された後で、isc_dsql_sql_info()をisc_info_sql_recordsパラメータ付きで呼び出すと、“request synchronization error”が起きていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2132) 比較演算子でストアドプロシージャの呼び出しが使われている場合、インデックス付き検索を選択できませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2117) インデックス付き検索とサブクエリで不正なROW_COUNTの値が返されていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2115)　　クエリが長い場合、クエリのplanが失われることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2101)　　オープンなPSQLカーソルのend-of-streamマークを越えて取得を行おうとした場合に、バグチェック249 (pointer page vanished) のエラーがスローされていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2098)　　グローバル一時表からselectしたビューを作成できませんでした。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2078)　　インデックスなしの選択演算子が結合に含まれていない場合、その結合のplanに可能なだけの最適化がなされませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2075)　　外部結合を使った場合、ビューのRDB$DB_KEYの各部が逆にされることがありました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2073)　　式インデックスのバグ：逆にされたブール式の結果が不正なものとなっていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2069)　　RDB$DB_KEYがビュー本体の中で使われた際に、不正なビューの伸張が行なわれていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2068)　　<サブクエリ式>の引数がRDB$DB_KEYを含む場合、IN <サブクエリ式>オペランドで、比較が誤った結果を返していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2067)　　GROUP BYとRDB$DB_KEYの問題。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2066)　　SQL_TEXT／SQL_VARCHARからSQL_TIMESTAMP／SQL_TIME／SQL_DATEへの変換。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2053)　　計算式がINSERT文のRETURNING句の中で使われた場合、最適化が不十分となることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2045)　　バージョン2.1での不具合が元に戻っていました。これにより、blr_field で存在しないシステムフィールドへの参照がNULLとして解決されていませんでした。一方で、並行して施されていたblr_fld に関わる変更は、適切で正しい挙動を示していました。

dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2044)　　UPDATE OR INSERT ... RETURNING OLDとnullableでないカラムの結果が不正なものとなっていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2041)　　UPDATE OR INSERTでGEN_ID()を使うと、ジェネレータが3ずつステップしていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2039)　　ドメインレベルでのCHECK制約が誤ってNULL値を処理していました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2031)　　RDB$DB_KEYにある条件を加えると、最初のレコードにNULLが現れていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2027)　　システムフィールドを含むORDER BY式のバッファサイズが誤って計算されていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2026)　　読み取り専用としてマーキングされたデータベースで問題がありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2008)　　システムスキーマで、プロシージャのパラメータにNOT NULLのフラグがありました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2002)　　UDFの結果にFREE_ITがマーキングされていた場合、その結果からの変換エラーでメモリリークが起きていました。

C. Valderramaが修正しました。

~ ~ ~

(CORE-2001)　　時々、変換エラーが表示されずに、算術例外または文字列切り捨てのメッセージが現れていました。

C. Valderramaが修正しました。

~ ~ ~

(CORE-2000)　　ロックマネージャが高負荷下で誤ってデッドロックをレポートすることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1986)　　ドメイン名を変更するとドメインへの依存関係が削除されてしまいました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1984)　　パーティシパントの一つがタイムアウトを許可されて待機している場合に、ロックマネージャが誤ってデッドロックをレポートすることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1980)　　スイーパによるCPU使用率がいつまでも100%となっていることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1970)　　Lock conversion denied (215) のエラーが起こることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1962)　　関数EXTRACT (MILLISECONDS FROM aTimeStampOrTime)が不正な結果を返していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1958)　　トランザクションが同じレコードを複数回更新しようとした場合、バグチェック179 (decompression overran buffer) による整合性チェックがスローされていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1957)　　長いアクセス制御リスト (ACL) が切り捨てられ、権限が消される原因となっていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1943)　　RAND()式で集約を行うSQL文が無限に行を返していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1938)　　別の接続で削除または再作成されているテーブルを参照するSQL文のプリペア中または実行中に、バグチェック243 (missing pointer page) スローされていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1936)　　組み込み関数LOG(base, number)がパラメータをチェックしておらず、例外処理を行わずにNANの値をout-of-rangeの入力に通知していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1914)　　テーブル作成中に問題が起こった場合、データベースが矛盾した状態に置かれることがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1812)　　ダイアレクト1によるいくつかのdate/time式で、使われるはずのインデックスが使われていませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1650)　　GROUP BY操作を使ったSELECT GEN_ID(..) FROM RDB$DATABASEで行が無限に生成されるという、ありそうもないことが起こっていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1607)　　UNIONのストリームに依存する相関サブクエリが充分に最適化されていませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1606)　　親レコードがロックされていても、外部キーのターゲットが変更されていない場合、子レコードを挿入することができていました。

A. Potapchenko、V. Khorsunが修正しました。

~ ~ ~

(CORE-1575)　　単一のトランザクションで、テーブルに複数回の更新が行なわれると、深刻なメモリバグが起きていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1544)　　ユーザーアプリケーションが、実行時の何らかの用途で、実行時に“一時”ストアドプロシージャを作成すると、RDB$PROCEDURES.RDB$PROCEDURE_IDカラムの内部ジェネレータが、その内部ジェネレータの32Kの制限（符号付SMALL　INT）をすぐに超えてしまい、新たなストアドプロシージャ作成の試みに対して“数値オーバーフロー”の例外を起こしていました。

この修正により、32Kの境界で生成された値は包み込まれ、ID番号の既存のギャップを再利用することが許可されます。同様の修正はRDB$GENERATORSとRDB$EXCEPTIONSにも適用されています。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1343)　　単純な場合とサブクエリでのバグ。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1246)　　派生テーブルとの外部結合が不正な結果を返していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1245)　　ビューとの外部結合が不正な結果を返していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-903)　　UPDATE文のSET句で、同じカラムを参照する複数の割り当てに関するFirebirdの挙動が標準に準拠していませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-501)　　COALESCEが最適化の問題を起こしていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-216)　　データベースでの権限付与が多すぎる場合、それらの権限が失われることがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1421)　　スーパーサーバーで、ログインの試みが失敗した後でシャットダウンがリクエストされた場合、すぐにシャットダウンできませんでした。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1907)　　同じトランザクションでドメイン制約の削除・追加を行うと、不正な依存関係が残っていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1905)　　aliases.confで、ファイル名に含まれるハッシュ記号 (#) が不正に処理されていました。

C. Valderramaが修正しました。

~ ~ ~

(CORE-1887)　　新たに作成されたデータベースが誤ったアクセス権を持っていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1869)　　ロールの付与／削除のロジックがバージョン2.0とバージョン2.1で違っていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1841)　　いくつかのビューが派生テーブルと長いテーブル名／エイリアスを使っていた場合、RDB$VIEW_RELATIONS.RDB$CONTEXT_NAMEをオーバーフローすることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1840)　　個々のDDLの実行で小さなメモリリークが起きていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1838)　　GTTのインデックスに対するSET STATISTICS INDEXが、インデックスidをデータベースのページサイズで利用できる最大の番号で誤って変更することがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1830)　　セーブポイントを使った同じトランザクションで同じレコードを複数回更新すると、インデックスが破損することがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1817)　　RelaxedAliasCheckingパラメータがRDB$DB_KEYに関して効果を持っていませんでした。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1811)　　キーワード“VALUE”をクォートで囲まずに使用しても、パーサが反応していました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1798)　　RDB$DB_KEYがINSERT ... RETURNINGの中でNULLとして評価されていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1797)　　トリガで、OLD/NEW.RDB$DB_KEYが不正な結果を返していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1784)　　EXECUTE STATEMENT内のEXECUTE PROCEDUREでエラーが起きていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1777)　　TPBで、テーブル予約に競合する指定が許可されていました。

C. Valderramaが修正しました。

~ ~ ~

(CORE-1775)　　プリペア中のセキュリティチェックが十分に行われていませんでした。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1770)　　DDLでバグチェック291が起きていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1735)　　参照整合性のトリガでSET DEFAULTアクションの引数の挙動に問題がありました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1731)　　時々、エンジンが何もI/O活動を行わないままCPU使用率100%で数分間ハングアップしていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1730)　　TempDirectoriesの設定で指定されたディレクトリの一つが利用できない場合、問題が起こっていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1724)　　計算項目または限定子（IN／ANY／ALL）で、共通テーブル式が使えないことがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1694)　　CREATE／ALTERデータベーストリガにバグがありました（ここのコメントはロシア語です）。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1693)　　CONNECT／TRANSACTION STARTトリガ内のEXECUTE STATEMENTでエラーが起きていました。

A. dos Santos Fernandes、D. Yemanovが修正しました。

~ ~ ~

(CORE-1689)　　“There are <n> dependencies”のエラーメッセージが依存するオブジェクトの数を誤って示していました。

C. Valderramaが修正しました。

~ ~ ~

(CORE-1357)　　DummyPacketIntervalの仕組みが壊れていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1307)　　fb_inet_serverのスイッチ-sが正しく処理されていませんでした。

A. Peshkovが修正しました。

~ ~ ~

(CORE-479)　　RDB$SECURITY_CLASSESで、権限付与により、もとのエントリが上書きされていました。

A. Peshkovが修正しました。

~ ~ ~

サーバー／クライアントのクラッシュ

(CORE-3011)　　アタッチとデタッチが繰り返し行なわれている接続の監視中にサーバーがハングアップまたはクラッシュすることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2908)　　ODS 8.xのデータベースでの稼働中に、エンジンがクラッシュまたは予期しないエラーを起こすことがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2888)　　メモリ破損が原因となり、クエリが不正に評価されたり、サーバーがクラッシュすることもありました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2576)　　誤った、または切り捨てられたBLRのパース時にサーバーがクラッシュすることがありました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2455)　　統計関数でのエラー発生直後にDROP DATABASEを実行すると、サーバーが故障していました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2441)　　UPDATE OR INSERT文の実行時にサーバーがクラッシュすることがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2306)　　いくつかのワーカースレッドが開始に失敗すると、スーパーサーバーが異常終了することがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2368)　　イベントが見つからない場合、isc_cancel_events()への呼び出しの後でアクセス違反が起きていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2222)　　BLOB変換フィルタを使ってテキストBLOBを保存すると、エンジンでアクセス違反が起こることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2137)　　設定パラメータDummyPacketIntervalが明示的に設定されていると、データベースのリストアで、サーバーがクラッシュすることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2121)　　クライアントのライブラリが、BLOBに関わる操作を行なっている間にクラッシュすることがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1983)　　OSでのout-of-memory状態によりアクセス違反が起きていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1965)　　DDLに同時に負荷がかかる状況下で、無効なロックIDにより、ロックマネージャがクラッシュしていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1894)　　計算フィールド間で循環した依存関係により、サーバーがクラッシュしていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1963)　　複数の接続から同時に権限の付与／削除を行うと、サーバーがクラッシュすることがありました

D. Yemanovが修正しました。

~ ~ ~

(CORE-1506)　　isc_dsql_execute_immediate()とゼロ長の文字列によりサーバーがクラッシュしていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-210)　　同じストアドプロシージャが二つの別々のプロセスで変更されている場合、クラシックサーバーがクラッシュしていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1930)　　プロシージャが出力を持たないよう変更され、依存するプロシージャが再コンパイルされなかった場合、サーバーがクラッシュすることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1919)　　EXECUTE STATEMENTでのメモリ破損によりサーバーがクラッシュすることがありました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1884)　　入力パラメータのデフォルト値として式を持つストアドプロシージャを使用すると、ランダムにクラッシュが起きていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1839)　　再帰CTEを使って計算されたフィールドによってソートを行うと、サーバーがクラッシュすることがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1793)　　パラメータ化された未使用のCTEを持つクエリの準備中にサーバーがクラッシュすることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1512)　　DEFAULT句に誤ったパース行っていたため、サーバーがクラッシュしていました。

D. Yemanovが修正しました。

~ ~ ~

データベースの監視／管理

(CORE-2209)　　高負荷状況下では、監視のリクエストが非常に遅くなったり、その間の他の活動をブロックすることもありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2171)　　テーブルMON$CALL_STACKのカラムMON$CALLER_IDが無効なIDを報告していました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2017)　　ストアドプロシージャのI/O統計がモニタリングテーブルに記録されていませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1944)　　ビッグエンディアンのプラットフォームで、モニタリングテーブルが誤ったデータを含んでいました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1890)　　高負荷下で、データベース監視プロセスがハングアップすることがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1881)　　データベースの監視により、サーバーがクラッシュしたり、ページロックのロジックに悪影響を及ぼすことがありました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1728)　　Linuxの新規インストール後にデータベース監視が動作しませんでした。

A. Peshkovが修正しました。

~ ~ ~

データ操作言語

(CORE-1910)　　MERGE文のINSERT句で、無効なフィールドが受け付けられていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1859)　　MAX()関数で、算術オーバーフローまたはゼロ除算が起こることがありました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1828)　　ダイアレクト1で、ABS()関数にエラーが起きていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

コマンドライン・ユーティリティ

isql

(CORE-2831)　　スクリプトが抽出される場合、出力にデータベース名とユーザー名が含まれるべきではありません。

C. Valderramaが修正しました。

~ ~ ~

(CORE-2741)　　CHECKキーワードが全ての大文字・小文字以外の任意のキャラクタが混在したものだった場合、メタデータの抽出で、CHECK制約のDDLが誤って解釈されていました。

C. Valderramaが修正しました。

~ ~ ~

(CORE-915)　　PSQL本体がWindowsのサードパーティ製テキストエディタで書かれていた場合、isqlのメタデータ抽出ツールがPSQL本体のコードで改行を二重にしていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2408)　　isqlによるメタデータ抽出プロセスで、NOT NULLとCOLLATEフラグの前に、プロシージャのパラメータのデフォルト値を置いていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2407)　　isqlによるメタデータ抽出プロセスで、CREATE DATABASE文からPAGE_SIZE句が抜けていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2370)　　2048以上のキャラクタを含むSQLのplanが、isqlで全く表示されませんでした。

C. Valderramaが修正しました。

~ ~ ~

(CORE-2270)　　isqlをzloginコンソールで実行すると、全メモリを消費してクラッシュしていました。

J. Swierczynski、A. Peshkovが修正しました。

~ ~ ~

(CORE-1891)　　SHOW VIEWが、式を持つビューのフィールドについて無意味な情報を示していました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1875)　　CURRENT_DATEを含むスクリプトでエラーが起きていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1862)　　Firebird 2.1で、抽出されたスクリプトが相互に依存する選択型プロシージャで使用できませんでした。

C. Valderramaが修正しました。

~ ~ ~

(CORE-1782)　　30以上のキャラクタからなるエイリアスを持つカラムのデータ取得時に、isqlがクラッシュしていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1749)　　AUTODDL ONを含むDDL文が統計を表示していませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1507)　　ISQLによるスクリプトでの行数カウント機能が、INPUTコマンドの後で同期しなくなっていました。

C. Valderramaが修正しました。

~ ~ ~

(CORE-1363)　　double型から変換された文字列が23バイトを超えた場合、ISQLがクラッシュしていました。

C. Valderramaが修正しました。

~ ~ ~

gsec

(CORE-2928)　　gsecでバッファオーバーフローが起きていました。

A. Peshkovが修正しました。

理由は不明ですが、gsecのコードが、表示操作中に、パスワードハッシュの値を内部ユーザーデータ構造体へとコピーしていました。バージョン2.0以降、新しいハッシュアルゴリズムによって以前よりハッシュが長くなっていますが、その保存操作に使えるバッファが短すぎた可能性があります。

ハッシュの値はどこにも移動しないため、これが脆弱性となることはありません。ともかく、無害ではあります：ファーストネーム、ミドルネーム、ラストネームはパスワードの直後に記入されるので、このバッファオーバーフローが悪用されることはあり得ません。

現在、これは修正されているので、このオーバーフローを検出するためにglibcの新しいバージョンを採用する必要はありません。

~ ~ ~

(CORE-2528)　　gsecユーティリティがOSにエラーコードを返していませんでした。

C. Valderramaが修正しました。

~ ~ ~

(CORE-1680)　　セキュリティ・データベースのユーザー数が50以上だった場合、gsecのDISPLAYで、最初の少数のユーザーしか表示されませんでした。

A. Peshkovが修正しました。

~ ~ ~

gbak

(CORE-2914)　　存在しないUDFを参照している式インデックスを持つデータベースのリストアを行うと、サーバーにクラッシュが起きていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2793)　　同じデータの複数回のバックアップ／リストアのサイクルでテストした結果、一つのバックアップから別のバックアップへ、バックアップファイルのバイナリ表現に一貫性がないことが明らかになりました。このことは、gbakのバージョン1.5.4以降全てのバージョンに当ては

まることが報告されました。これは、バージョン2.1.4で、またバージョン3.0のアルファ版で、現在までに修正されています。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2634)　　大量のメタデータを持つデータベースのリストア時に、パフォーマンスが低下する不具合が起きていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2291)　　エンジンがエラーを検出して適切な例外をスローすべき場合に、[FOR] SELECTを含む間違ったトリガのコードに対してエラーバグチェック284 (cannot restore singleton select data) がスローされていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2285)　　多数の権限付与を行っているデータベースがバックアップ／リストア後に破損することがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2245)　　長い例外メッセージが定義されたデータベースで、バックアップからのリストア時にエラーが起きていました。

C. Valderramaが修正しました。

~ ~ ~

(CORE-2223)　　gbakが、SQLの権限を保存しているアクセス制御リスト (ACL) での作業中にいくつかのバグに遭遇していました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2214)　　セキュリティクラスのリストアが不正に行われていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1911)　　サービスAPI使用時に、バックアップとリストアがスレッドセーフとなっていませんでした。

C. Valderramaが修正しました。

~ ~ ~

(CORE-1843)　　サービスマネージャを使用した場合、gbakがスペースを含むパスを許可しませんでした。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1703)　　gbakの出力が別のプロセスにリダイレクトされた場合、遅延／ロックアップが起こっていました。

D. Yemanovが修正しました。

~ ~ ~

nBackup

(CORE-2750)　　明示的な差分ファイルが削除された後で、物理バックアップが操作を再開できませんでした。

C. Valderramaが修正しました。

~ ~ ~

(CORE-2648)　　nBackupの増分ファイルがデータベースの“Forced Writes”設定を尊重していませんでした。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2266)　　nBackupによるデータベースのロックが正しく動作しておらず、データベースへの書き込みが中断状態にあるべき時でもデータベースファイルの伸張が続いていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1696)　　nBackupユーティリティ使用時に、ロックマネージャでデッドロックが発生していました。

R. Simakovが修正しました。

~ ~ ~

(CORE-1876)　　バージョン2.1で、nBackupを用いた増分バックアップに失敗していました。

N. Samofatovが修正しました。

~ ~ ~

gfix

(CORE-2846)　　接続がまだアクティブであるために、gfix -shut <モード> -attach <タイムアウト>が指定されたタイムアウトの後で失敗した場合、データベースへの接続ができませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-97)　　gfix -shut -forceによってデータベースファイルがロックされたままとなり、その後にリストアできなくなっていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2268)　　有効でないトランザクション番号が原因となって、gfixがBUGCHECKのエラーをスローしていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2271)　　gfixユーティリティにはレガシーなバグがあり、大きなデータベースでのデータベース検証／修正ルーチン中に現れていました。こうしたルーチンを実行するユーザーの権限レベルに関するチェックが操作の遅い段階で行なわれていたため、権限を持たない（つまり、SYSDBAや所有者ではない）ユーザーが検証操作を開始できてしまいました。いったん権限チェックが始まると、データベースの検証は作業中でも停止されるため、未完状態となり、避けられたはずのロジカルな破損が起こっていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1961)　　データベースの権焼酎に整合性チェックのエラーバグチェック210 (page in use during flush) がスローされていました。

D. Yemanov、R. Simakovが修正しました。

~ ~ ~

gstat

(CORE-2519)　　20億以上のレコードを持つテーブルで、gstatからの出力が不正なものとなっていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1412)　　長く放置されてきた、gstatのパラメータ処理に関するいくつかのバグを修正する必要がありました。

C. Valderramaが修正しました。

~ ~ ~

fb_lock_print

(CORE-2598)　　fb_lock_print -c[onsistency]スイッチがWindowsで動作していませんでした。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2354)　　“fb_lock_print -ia”の出力が、繰り返し中にファイルに書き込まれていませんでした。

A. Peshkovが修正しました。

~ ~ ~

GDMLのqliクエリユーティリティ

(CORE-2247)　　QLIユーティリティで、メッセージと記述子のバッファが適切に配列されていませんでした。

A. Peshkovが修正しました。

~ ~ ~

サービスマネージャ

(CORE-1982)　　サービスAPIを通じて呼び出された同時のバックアップまたはリストアが互いに干渉しあうことがありました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

リモートインターフェース／API

(CORE-2563)　　不正な形式のパケットをいくつかの特殊なフォーマットで送信することで、スーパーサーバーのメインポート（デフォルトで3050）をシャットダウンすることが可能でした。これにより、新規の接続に対するサービス妨害（DoS）状態に陥っていました。これは認証されていないクライアントによって悪用される可能性がありました。

2009年7月15日、Core Security Technologies社によって報告されました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2437)　　クライアントがイベントを受け取った時にバッファオーバーフローが起こることがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2307)　　API情報のリクエストの結果として不完全な値が返されていました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-2234)　　Windowsのクラシックサーバーで、Firebirdサーバー側でのチェックが不適切だったため、時々、終了した作業プロセスが終了後もまだ生きていると見なされていました。同じバグにより、Firebirdサーバーで高負荷時に誤作動や長いデッドロックが起こることがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2151)　　テンポラリディレクトリのパスにスペースが含まれていた場合、右端のスペースで（誤って）切り捨てられていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2033)　　クライアントライブラリ内のシンボル_Unwind_GetIPが、静的ライブラリのリンクが失われていたため、未解決のままになっていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2018)　　読み取り専用データベースに単一のクライアントしかアクセスできませんでした。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1972)　　非SYSDBAユーザーが、SYSDBAに限定すべき他のいくつかのデータベースの特性とともに、任意のデータベースのForced　Writesモードを変更できていました。この、長く放置されてきた、DPBパラメータの処理でのレガシーなループホールにより、データベースの破損が起きたり、通常のユーザーがSYSDBA専用の操作にアクセスできるようになっていました。

この変更により、いくつかの既存のアプリケーションやデータベースツール、コネクティヴィティ・レイヤー（ドライバ、コンポーネント）が影響を受けます。

いくつかの修正はバージョン2.1.2とバージョン2.0.5にバックポートされました。.

A. Peshkovが修正しました。

~ ~ ~

(CORE-1868)　　クライアントライブラリがisc_dsql_free_statement()内でクラッシュしていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1763)　　クライアントライブラリが、接続時、ソケットにオプションSO_KEEPALIVEもTCP_NODELAYも設定していませんでした。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1755 and)CORE-1756 isc_start_transaction()で、二種類のクラッシュが起こることがありました。

D. Kovalenko、A. Peshkovが修正しました。

~ ~ ~

(CORE-1726) isc_service_start()で失敗していました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1079) fbclient/fbembedライブラリによるホストプロセスへの各アタッチで、64KBのメモリリークが起きていました。

A. Peshkovが修正しました。

~ ~ ~

セキュリティ

(CORE-2657) きちんとドキュメント化されていないSPBタグにより、任意の信頼されたユーザー名をサービスに渡し、そのユーザーがSYSDBAのものを含む任意のパーミッションを手にすることができるという、望ましくない機能が提供されていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2087) 設定パラメータRemoteBindAddressがIPアドレスではなく、ホスト名や、存在しないIPアドレスを指定した場合、それは警告なく無視され、サーバーはfirebird.logまたはシステムログには何も通知せずに全てのインターフェースにバインドします。システムがインターネットにポートを開いている場合、これは潜在的なセキュリティ上のリスクと見なされます。現在は、無効な、または利用できないIPアドレスはlocalhost（127.0.0.1）として解決されることになります。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2055) Firebirdのクライアントライブラリでバッファオーバーフローが起きていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1845) 標準の呼び出しの一部が、通常のユーザーに対してサーバーのインストールディレクトリを示していました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1810)　　キャラクタ'.'を含むユーザー名に関して、いくつかのイシューが出されました。

A. Peshkovが修正しました。

~ ~ ~

国際言語のサポート

~ ~ ~

(CORE-2642)　　Windowsでの不明瞭なICU初期化の問題により、マルチスレッド環境で誤作動が起きていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2607)　　introducer構文（_charset）が監視クエリやPSQLモジュールと関連して使われた場合、問題が起きていました。

この問題と回避方法については、国際言語のサポートの章、その他の改善点中のIntroducer構文の使用法の項を参照して下さい。

A. dos Santos Fernandesが修正しました。

~ ~ ~

~ ~ ~

(CORE-2361)　　isc_dsql_fetch()をUTF8接続で使用して、キャラクタ・セット8859_1のスペイン語のカラムを読み込む時に、文字列切り捨てのエラーが起きていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2227)　　いくつかの環境で、アクセント付きキャラクタを含むカラム名を参照するトリガの作成時に問題が起きていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-2123)　　CP943Cの接続charsetでUNICODE_FSSのデータを取得する際に問題がおきていました。

A. dos Santos Fernandes、D. Kovalenkoが修正しました。

~ ~ ~

(CORE-2122)　　UNICODE_FSS／UTF8と他のcharsetの間での大きなテキストBLOBの翻訳。

A. dos Santos Fernandes、D. Kovalenkoが修正しました。

~ ~ ~

(CORE-2095)　　CVJIS_eucj_to_unicode()のバグ。

A. dos Santos Fernandes、D. Kovalenkoが修正しました。

~ ~ ~

(CORE-2019)　　UTF-8の変換エラー（文字列切り捨て）が予期せずにスローされていました。

dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1989)　　UTF8用のUNICODE_CIコレーションを使ったカラムが外部キー制約で使用できませんでした。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1927)　　プロシージャsp_register_character_setがRDB$CHARACTER_SETS.RDB$CHARACTER_SET_IDに負の値を生成することがありました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1690)　　テーブルがUTF8のテキストを含んでいる場合に、算術例外、数値オーバーフロー、または文字列切り捨てのエラーが起きていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1596)　　CsConvert::convertのバグ。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1432)　　レコードのフォーマット間でカラムのコレーション属性が伝達されていませんでした。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-316)　　名前にマルチバイトのキャラクタを含むデータベースを開けないことがありました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1802)　　PXW_CSYのコレーションを使う最大のキーサイズに関して、いくつかのイシューがレポートされていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1774)　　ES_ES_CI_AIのコレートで問題が起きていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

(CORE-1254)　　DISTINCTと、大文字小文字・アクセントを無視したコレーションで問題が起きていました。

A. dos Santos Fernandesが修正しました。

~ ~ ~

POSIX固有の問題

(CORE-3067)　　HP-UXを除く64bitのPOSIXで、共有メモリが閉じられた時にオブジェクトがアンマッピングされていませんでした。

64bitのポインタが32bitのマスクでマスキングされていました。これは全くひどいやり方だとわかりました：Object == 0x7FAB12345678、page size == 4K、bit数の上限は無くなり、開始アドレスは意図した0x7FAB12345000ではなく0x12345000となっていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3019)　　Gentoo　Linuxの最近のバージョンで、関数ライブラリ/etc/init.d/functions.shを利用しようとすると、メッセージ“* ERROR: firebird does not have a start function”を出して失敗し、スーパーサーバーもスーパークラシックサーバーも開始しませんでした。

A. Peshkovが修正しました。

~ ~ ~

(CORE-3001)　　インストール時に、firebirdユーザーとグループの作成に失敗していました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2919)　　Linuxのインストール・スクリプトが標準以外のポートを無視していました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2845)　　Solarisでのビルドのプロセスが、メッセージ“SFIOを使う必要があります”を出して停止していました。そもそも、Solaris　10はもはやSFIOを必要としていません。64bitビルドでは、ファイル記述子の上限255での問題は起こりませんし、32bitのビルドでは簡単な回避方法があります。詳細はこちらの記事を参照して下さい。

しかし、Solaris 10 3/05からSolaris 10 11/06まではこのイシューの影響を受けます。問題の修正にはいくつかのパッチが必要となります。

当面の最も簡単な解決法は、コードにdefineを残しておくことです。ただし、Solaris 10に関連する情報でコメントを付けておいて下さい。パッチを適用するか、ユーザーがSolaris 10でこの問題の影響を受けないバージョンを使っていれば、defineは削除できます。

このイシューに関する更新情報はcommon.hに挿入されています。Solarisで32bitのFirebirdをビルドする方は、SFIO defineのコメントを解除する前に、Soloarisのバージョンやビルド／パッチレベルに応じて十分な情報に基づいた決定を行う必要があります。

P. Beach、A. Peshkovが修正しました。

~ ~ ~

(CORE-2814) SPARCでは、ルーチンmap_sort_dataがバスエラーを起こしてサーバーがクラッシュしていました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2601) POSIXプラットフォーム上でインストール・ディレクトリの微調整に使われる標準的なconfigureスイッチの多くは、Firebirdには役立ちません。

標準的なGNUのスイッチをデフォルト設定を変更せずに機能させるのは不可能に近く、簡単明瞭とはほど遠い煩雑なものでした。これらに代わり、configure用の新たなスイッチのセットが追加され、Firebirdのファイル群のきめ細かな配置設定が可能になりました。

これらのスイッチのリストはインストールの章、Firebirdの'configure'専用スイッチの項で挙げられています。

A. Peshkoffが修正しました。

~ ~ ~

(CORE-2572) タイプLCK_page_spaceのロックがビッグエンディアンのマシンで不正に処理されていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-2221) POSIXプラットフォームで、security2.fdbへのアクセス権が0660から0666に修正されると、どのデータベースへのどのアタッチメントも失敗するようになっていました。

P. Beach、A. Peshkovが修正しました。

~ ~ ~

(CORE-2093) 64bit版Solarisで、スーパーサーバーのスタートアップに失敗していました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1909) linux/amd64でfirebird.logにガベージがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1885)　　POSIX下で、CREATE COLLATIONによって接続が失われていました。

A. dos Santos Fernandes、A. Peshkovが修正しました。

~ ~ ~

(CORE-1854)　　Unix OSの認証方法を利用する時、CURRENT_USERの値が大文字にならないことがありました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1826)　　changeRunUser.shとrestoreRootRunUser.shスクリプトがinit.dスクリプトで実行ユーザーを変更していませんでした。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1818)　　POSIXプラットフォームで、テンポラリ・ページスペースで使われるテンポラリ・ファイルが使用後に削除されませんでした。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1807)　　サーバーが異常終了後に標準的でないポートに割り当てられていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1766)　　Linux版クラシックサーバーの所有者とisc_monitor1ファイルのグループが不正なものとなっていました。

A. Peshkovが修正しました。

~ ~ ~

(CORE-1671)　　ライブラリがdlopen'edモジュールで使われた場合、クライアントライブラリのatexit()呼び出しがセグメンテーション違反を起こしていました。

A. Peshkovが修正しました。

~ ~ ~

Windows固有の問題

(CORE-2769)　　高負荷のWindowsシステム上で、サーバによるxnet_response_eventの設定を待機している間に、ローカル接続（XNET）がクライアントのタイムアウトによって失敗することがありました。この問題の解決のため、firebird.conf 中のConnectionTimeoutパラメータがXNET接続用にアクティベートされています。

D. Yemanovが改善しました。

~ ~ ~

(CORE-2108)　Windowsローカルプロトコル（XNET）の使用時に、次に利用可能なマップ番号が不正に計算され、その結果、サーバーが既存のマップ番号を再利用しようとすることができていました。“新しい”マップのタイムスタンプが既存のマップのタイムスタンプと等しい場合、get_free_slot()関数が失敗していました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-2107)　高負荷下で、Windows版クラシックサーバーへのTCP¥IP接続の確立に失敗することがありました。

V. Khorsunが修正しました。

~ ~ ~

(CORE-1923)　instsvc.exeの削除が正常に行われた時に、完了コードとして0ではなく1を返していました。

D. Yemanovが修正しました。

~ ~ ~

(CORE-1820)　セットアッププログラムが稼働中のサーバーの検出に失敗していました。

P. Reeves、D. Yemanovが修正しました。

~ ~ ~

(CORE-1105、 CORE-1390、 CORE-1566 および CORE-1639)　XNET接続でエイリアスが適切に機能していませんでした。

D. Yemanovが修正しました。

~ ~ ~

MacOSX固有の問題

(CORE-2065)　MacOSXのインストールパッケージが、動的ローダの検索パスにクライアントライブラリを含んでおらず、プラットフォームのルールに違反していました。

P. Beach、A. Peshkovが修正しました。

~ ~ ~

Miscellaneous Bugs

(CORE-2282)　-1より小さい負の数で、UDFの切り捨てが失敗していました。

C. Valderramaが修正しました。

~ ~ ~

(CORE-2281)　　負の数でUDFを丸めるのに失敗していました。

C. Valderramaが修正しました。

~ ~ ~

Chapter 18

Firebird 2.5 プロジェクトチーム

Table 18.1. Firebird開発チーム

開発者	出身国	担当作業
Dmitry Yemanov	ロシア連邦	フルタイムDBエンジニア／実装、コアチームリーダー
Alex Peshkov	ロシア連邦	フルタイムセキュリティ機能コーディネータ；ビルドマスター；移植オーソリティ
Claudio Valderrama	チリ	コード検査；バグ発見、修正；ISQL拡張；UDF修正、設計、実装
Vladyslav Khorsun	ウクライナ	フルタイムDBエンジニア、SQL機能設計／実装
Adriano dos Santos Fernandes	ブラジル	新国際キャラクタ・セット処理；テキストおよびテキストBLOB拡張；新DSQL機能；コード検査
Nickolay Samofatov	ロシア連邦	エンジンへの寄与
Roman Simakov	ロシア連邦	エンジンへの寄与
Paul Beach	フランス	リリースマネージャ；HP-UXビルド；MacOSビルド；Solarisビルド
Pavel Cisar	チェコ共和国	QAツール設計／コーディネート
Philippe Makowski	フランス	QAテスト；Linuxディストリビューションビルド
Pavel Zotov	ロシア連邦	QAテスト／ツール開発
Paul Reeves	フランス	Win32インストーラおよびビルド
Mark Rotteveel	オランダ	Jaybird実装およびコーディネート
Jiri Cincura	チェコ共和国	.NETプロバイダの開発およびコーディネート
Alexander Potapchenko	ロシア連邦	Firebird用ODBC/JDBCドライバの開発およびコーディネート
Stephen Boyd	カナダ	GPREへの寄与
Paul Vinkenoog	オランダ	コーディネート、Firebirdドキュメントプロジェクト；ドキュメント執筆およびツール開発／実装
Norman Dunbar	イギリス	ドキュメント執筆

開発者	出身国	担当作業
Pavel Menshchikov	ロシア連邦	ドキュメント翻訳
Tomneko Hayashi	日本	ドキュメント翻訳
Umberto (Mimmo) Masotti	イタリア	ドキュメント翻訳
Helen Borrie	オーストラリア	リリースノート編集；思想警察長官
さらに		
リトラル・コート・ドパール大学の学生たち	フランス	QAテスト開発

Appendix A: SQLSTATE

SQLSTATEコードとメッセージ

この付録では、現在サポートされている全SQLSTATEコードを網羅しています：

1. SQL CLASS（2文字）とSQL SUBCLASS（3文字）からなる状態配列として返される5文字のSQL-STATEコードです。
2. 対応するものが知られている場合は、非推奨のSQLCODEへの一対一のマッピングが含まれます。
3. 多くの場合、SQLCODE対SQLSTATEのマッピングは一対一ではありませんが、これはSQL標準委員会が意図した通りです。SQLCODEの使用を完全に非推奨とすることが長年にわたる彼らの目標でした。

SQLSTATEコード	マッピングされたメッセージ	SQLCODEへのマッピング
SQLCLASS 00 成功（Success）		
00000	成功（Success）	
SQLCLASS 01 警告（Warning）		
01000	一般的警告（General Warning）	
01001	カーソル操作の競合（Cursor operation conflict）	
01002	接続切断エラー（Disconnect error）	
01003	NULL値がset関数から除外されました（NULL value eliminated in set function）	
01004	文字列データの右端が切り捨てられました（String data, right-truncated）	
01005	アイテム記述子エリアが不足しています（Insufficient item descriptor areas）	
01006	権限を削除できません（Privilege not revoked）	
01007	権限を付与できません（Privilege not granted）	
01008	暗黙のゼロbitパディング（Implicit zero-bit padding）	
01100	SQL文がunpreparedにリセットされました（Statement reset to unprepared）	
01101		

SQLSTATEコード	マッピングされたメッセージ	SQLCODEへのマッピング
	実行中のトランザクションがコミットされました（Ongoing transaction has been committed）	
01102	実行中のトランザクションがロールバックされました（Ongoing transaction has been rolled back）	
SQLCLASS 02 データがありません（No Data）		
02000	データが見つかりません、または、影響を受ける行がありません（No data found or no rows affected）	
SQLCLASS 07 動的SQLエラー（Dynamic SQL error）		
07000	動的SQLエラー（Dynamic SQL error）	
07001	入力パラメータの数が正しくありません（Wrong number of input parameters）	
07002	出力パラメータの数が正しくありません（Wrong number of output parameters）	
07003	カーソル指定を実行できません（Cursor specification cannot be executed）	
07004	動的パラメータにUSING句が必要です（USING clause required for dynamic parameters）	
07005	プリペアされたSQL文がカーソル指定のものではありません（Prepared statement not a cursor-specification）	
07006	制限付データ型の属性違反（Restricted data type attribute violation）	
07007	結果フィールドにUSING句が必要です（USING clause required for result fields）	
07008	無効な記述子カウント（Invalid descriptor count）	
07009	無効な記述子インデックス（Invalid descriptor index）	
SQLCLASS 08 接続の例外（Connection Exception）		
08001	クライアントが接続を確立できません（Client unable to establish connection）	
08002	接続名が使用中です（Connection name in use）	
08003	接続が存在しません（Connection does not exist）	
08004	サーバーが接続をリジェクトしました（Server rejected the connection）	

SQLSTATEコード	マッピングされたメッセージ	SQLCODEへのマッピング
08006	接続に失敗しました (Connection failure)	
08007	トランザクションの解決が不明です (Transaction resolution unknown)	
SQLCLASS 0A 非サポート機能 (Feature Not Supported)		
0A000	機能がサポートされていません (Feature Not Supported)	
SQLCLASS 0B 無効なトランザクション初期化 (Invalid Transaction Initiation)		
0B000	無効なトランザクションの初期化 (Invalid transaction initiation)	
SQLCLASS 0L 無効な権限付与者 (Invalid Grantor)		
0L000	無効な権限付与者 (Invalid grantor)	
SQLCLASS 0P 無効なロール指定 (Invalid Role Specification)		
0P000	無効なロール指定 (Invalid role specification)	
SQLCLASS 0U 更新不可のカラムへの割り当て試行 (Attempt to Assign to Non-Updatable Column)		
0U000	更新できないカラムへ割り当てを行おうとしています (Attempt to assign to non-updatable column)	
SQLCLASS 0V オーダリングカラムへの割り当て試行 (Attempt to Assign to Ordering Column)		
0V000	オーダリングカラムへ割り当てを行おうとしています (Attempt to assign to Ordering column)	
SQLCLASS 20 case文でcaseが見つからない (Case Not Found For Case Statement)		
20000	case文でcaseが見つかりません (Case not found for case statement)	
SQLCLASS 21 濃度違反 (Cardinality Violation)		
21000	濃度違反 (Cardinality violation)	
21S01	挿入値リストがカラムリストにマッチしません (Insert value list does not match column list)	
21S02	派生テーブルの次数がカラムリストにマッチしません (Degree of derived table does not match column list)	
SQLCLASS 22 データの例外 (Data Exception)		
22000	データの例外 (Data exception)	

SQLSTATEコード	マッピングされたメッセージ	SQLCODEへのマッピング
22001	文字列データの右側が切り捨てられました (String data, right truncation)	
22002	NULL値またはインジケータパラメータがありません (Null value, no indicator parameter)	
22003	数値が範囲内にありません (Numeric value out of range)	
22004	NULL値は許可されていません (Null value not allowed)	
2205	割り当てエラー (Error in assignment)	
2206	フィールド参照にNULL値があります (Null value in field reference)	
2207	無効なデータ日時フォーマット (Invalid datetime format)	
22008	日時フィールドのオーバーフロー (Datetime field overflow)	
22009	無効なタイムゾーンのディスプレースメント値 (Invalid time zone displacement value)	
2200A	参照先がNULL値です (Null value in reference target)	
2200B	エスケープキャラクタの競合 (Escape character conflict)	
2200C	エスケープキャラクタ使用が無効です (Invalid use of escape character)	
2200D	無効なエスケープoctet (Invalid escape octet)	
2200E	配列ターゲットにNULL値があります (Null value in array target)	
2200F	ゼロ長のキャラクタ文字列 (Zero-length character string)	
2200G	明確な型の不一致 (Most specific type mismatch)	
22010	無効なインジケータパラメータ値 (Invalid indicator parameter value)	
22011	部分文字列エラー (Substring error)	
22012	ゼロ除算 (Division by zero)	
22014	無効な更新値 (Invalid update value)	
22015	インターバルフィールドのオーバーフロー (Interval field overflow)	

SQLSTATEコード	マッピングされたメッセージ	SQLCODEへのマッピング
22018	castのキャラクタ値が無効です (Invalid character value for cast)	
22019	無効なエスケープキャラクタ (Invalid escape character)	
2201B	無効な正規表現 (Invalid regular expression)	
2201C	テーブルでNULL行は許可されていません (Null row not permitted in table)	
22020	無効な制限値 (Invalid limit value)	
22021	キャラクタがレパートリにありません (Character not in repertoire)	
22022	インジケータのオーバーフロー (Indicator overflow)	
22023	無効なパラメータ値 (Invalid parameter value)	
22024	キャラクタ文字列が適切に終了していません (Character string not properly terminated)	
22025	無効なエスケープシークケンス (Invalid escape sequence)	
22026	文字列データ、長さの不一致 (String data, length mismatch)	
22027	切り捨てエラー (Trim error)	
22028	行がすでに存在しています (Row already exists)	
2202D	mutator関数にNULLインスタンスがあります (Null instance used in mutator function)	
2202E	配列要素エラー (Array element error)	
2202F	配列データ、右端切り捨て (Array data, right truncation)	
SQLCLASS 23 整合性制約違反 (Integrity Constraint Violation)		
23000	整合性制約違反 (Integrity constraint violation)	
SQLCLASS 24 無効なカーソル状態 (Invalid Cursor State)		
24000	無効なカーソル状態 (Invalid cursor state)	
24504	UPDATE、DELETE、SET、またはGET文で指定されたカーソルが行に配置されていません (The cursor identified in the UPDATE, DELETE, SET, or GET statement is not positioned on a row)	

SQLSTATEコード	マッピングされたメッセージ	SQLCODEへのマッピング
SQLCLASS 25 無効なトランザクション状態 (Invalid Transaction State)		
25000	無効なトランザクション状態 (Invalid transaction state)	
25	xxxx	
25S01	トランザクション状態 (Transaction state)	
25S02	トランザクションがまだアクティブです (Transaction is still active)	
25S03	トランザクションがロールバックされています (Transaction is rolled back)	
SQLCLASS 26 無効なSQL文名 (Invalid SQL Statement Name)		
26000	無効なSQL文名 (Invalid SQL statement name)	
SQLCLASS 27 トリガデータの変更違反 (Triggered Data Change Violation)		
27000	トリガデータの変更違反 (Triggered data change violation)	
SQLCLASS 28 無効な認証指定 (Invalid Authorization Specification)		
28000	無効な認証指定 (Invalid authorization specification)	
SQLCLASS 2B まだ存在する依存検眼記述子 (Dependent Privilege Descriptors Still Exist)		
2B000	依存権限記述子がまだ存在します (Dependent privilege descriptors still exist)	
SQLCLASS 2C 無効なキャラクタ・セット名 (Invalid Character Set Name)		
2C000	無効なキャラクタ・セット名 (Invalid character set name)	
SQLCLASS 2D 無効なトランザクション終了 (Invalid Transaction Termination)		
2D000	無効なトランザクション終了 (Invalid transaction termination)	
SQLCLASS 2E 無効な接続名 (Invalid Connection Name)		
2E000	無効な接続名 (Invalid connection name)	
SQLCLASS 2F SQLルーチンの例外 (SQL Routine Exception)		
2F000	SQLルーチンの例外 (SQL routine exception)	

SQLSTATEコード	マッピングされたメッセージ	SQLCODEへのマッピング
2F002	SQLデータの修正は許可されていません (Modifying SQL-data not permitted)	
2F003	SQL文の試行は禁止されています (Prohibited SQL-statement attempted)	
2F004	SQLデータの読み込みが許可されていません (Reading SQL-data not permitted)	
2F005	関数がreturn文を実行しません (Function executed no return statement)	
SQLCLASS 33 無効なSQL記述子名 (Invalid SQL Descriptor Name)		
33000	無効なSQL記述子名 (Invalid SQL descriptor name)	
SQLCLASS 34 無効なカーソル名 (Invalid Cursor Name)		
34000	無効なカーソル名 (Invalid cursor name)	
SQLCLASS 35 無効な条件番号 (Invalid Condition Number)		
35000	無効な条件番号 (Invalid condition number)	
SQLCLASS 36 カーソル感度の例外 (Cursor Sensitivity Exception)		
36001	リクエストがリジェクトされました (Request rejected)	
36002	リクエストに失敗しました (Request failed)	
SQLCLASS 37 無効な識別子 (Invalid Identifier)		
37000	無効な識別子 (Invalid identifier)	
37001	識別子が長すぎます (Identifier too long)	
SQLCLASS 38 外部ルーチンの例外 (External Routine Exception)		
38000	外部ルーチンの例外 (External routine exception)	
SQLCLASS 39 外部ルーチン呼び出しの例外 (External Routine Invocation Exception)		
39000	外部ルーチン呼び出しの例外 (External routine invocation exception)	
SQLCLASS 3B 無効なセーブポイント (Invalid Save Point)		
3B000	無効なセーブポイント (Invalid save point)	
SQLCLASS 3C あいまいなカーソル名 (Ambiguous Cursor Name)		
3C000	カーソル名があいまいです (Ambiguous cursor name)	

SQLSTATEコード	マッピングされたメッセージ	SQLCODEへのマッピング
SQLCLASS 3D 無効なカタログ名 (Invalid Catalog Name)		
3D000	無効なカタログ名 (Invalid catalog name)	
3D001	カタログ名が見つかりません (Catalog name not found)	
SQLCLASS 3F 無効なスキーマ名 (Invalid Schema Name)		
3F000	無効なスキーマ名 (Invalid schema name)	
SQLCLASS 40 トランザクションロールバック (Transaction Rollback)		
40000	実行中のトランザクションがロールバックされました (Ongoing transaction has been rolled back)	
40001	シリアライズの失敗 (Serialization failure)	
40002	トランザクション整合性制約違反 (Transaction integrity constraint violation)	
40003	SQL文の完了が不明です (Statement completion unknown)	
SQLCLASS 42 構文エラーまたはアクセス違反 (Syntax Error or Access Violation)		
42000	構文エラーまたはアクセス違反 (Syntax error or access violation)	
42702	カラム参照があいまいです (Ambiguous column reference)	
42725	関数参照があいまいです (Ambiguous function reference)	
42818	演算子または関数のオペランドに互換性がありません (The operands of an operator or function are not compatible)	
42S01	ベースのテーブルまたはビューがすでに存在しています (Base table or view already exists)	
42S02	ベースのテーブルまたはビューが見つかりません (Base table or view not found)	
42S11	インデックスがすでに存在しています (Index already exists)	
42S12	インデックスが見つかりません (Index not found)	
42S21	カラムがすでに存在しています (Column already exists)	
42S22	カラムが見つかりません (Column not found)	

SQLSTATEコード	マッピングされたメッセージ	SQLCODEへのマッピング
SQLCLASS 44 WITH CHECKオプション違反 (With Check Option Violation)		
44000	WITH CHECKオプション違反 (WITH CHECK OPTION Violation)	
SQLCLASS 45 処理されないユーザー定義の例外 (Unhandled User-defined Exception)		
45000	ユーザー定義の例外が処理されません (Unhandled user-defined exception)	
SQLCLASS 54 プログラム制限の超過 (Program Limit Exceeded)		
54000	プログラム制限を超過しています (Program limit exceeded)	
54001	SQL文が複雑すぎます (Statement too complex)	
54011	カラムが多すぎます (Too many columns)	
54023	引数が多すぎます (Too many arguments)	
SQLCLASS HY CLI固有の状態 (CLI-specific Condition)		
HY000	CLI固有の状態です (CLI-specific condition)	
HY001	メモリ割り当てエラー (Memory allocation error)	
HY003	アプリケーション記述子に無効なデータ型があります (Invalid data type in application descriptor)	
HY004	無効なデータ型 (Invalid data type)	
HY007	関連するSQL文がプリペアされていません (Associated statement is not prepared)	
HY008	操作がキャンセルされました (Operation canceled)	
HY009	NULLポインタの使用が無効です (Invalid use of null pointer)	
HY010	関数シーケンスエラー (Function sequence error)	
HY011	現在、属性の設定はできません (Attribute cannot be set now)	
HY012	無効なトランザクション操作コード (Invalid transaction operation code)	
HY013	メモリ管理エラー (Memory management error)	
HY014	ハンドル数の制限を超過しています (Limit on the number of handles exceeded)	

SQLSTATEコード	マッピングされたメッセージ	SQLCODEへのマッピング
HY015	利用可能なカーソル名がありません (No cursor name available)	
HY016	実装行記述子を修正できません (Cannot modify an implementation row descriptor)	
HY017	自動で割り当てられた記述子ハンドルの使用が無効です (Invalid use of an automatically allocated descriptor handle)	
HY018	サーバーがキャンセルのリクエストを拒否しました (Server declined the cancellation request)	
HY019	文字列以外のデータは分割して送信できません (Non-string data cannot be sent in pieces)	
HY020	NULL値を連結しようとしています (Attempt to concatenate a null value)	
HY021	記述子の情報が矛盾しています (Inconsistent descriptor information)	
HY024	無効な属性値 (Invalid attribute value)	
HY055	文字列以外のデータは文字列ルーチンで使えません (Non-string data cannot be used with string routine)	
HY090	無効な文字列長またはバッファ長 (Invalid string length or buffer length)	
HY091	無効な記述子フィールド識別子 (Invalid descriptor field identifier)	
HY092	無効な属性識別子 (Invalid attribute identifier)	
HY095	無効な関数ID指定 (Invalid FunctionId specified)	
HY096	無効な情報型 (Invalid information type)	
HY097	カラム型が範囲内にありません (Column type out of range)	
HY098	スコープが範囲内にありません (Scope out of range)	
HY099	nullable型が範囲内にありません (Nullable type out of range)	
HY100	一意性オプション型が範囲内にありません (Uniqueness option type out of range)	

SQLSTATEコード	マッピングされたメッセージ	SQLCODEへのマッピング
HY101	accuracyオプション型が範囲内にありません（Accuracy option type out of range）	
HY103	無効な検索コード（Invalid retrieval code）	
HY104	無効なLengthPrecision値（Invalid LengthPrecision value）	
HY105	無効なパラメータ型（Invalid parameter type）	
HY106	無効な取得操作（Invalid fetch orientation）	
HY107	行の値が範囲内にありません（Row value out of range）	
HY109	無効なカーソル配置（Invalid cursor position）	
HY110	無効なドライバ完了（Invalid driver completion）	
HY111	無効なブックマーク値（Invalid bookmark value）	
HYC00	オプション機能が実装されていません（Optional feature not implemented）	
HYT00	タイムアウトしました（Timeout expired）	
HYT01	接続がタイムアウトしました（Connection timeout expired）	
SQLCLASS XX 内部エラー（Internal Error）		
XX000	内部エラー（Internal error）	
XX001	データが破損しています（Data corrupted）	
XX002	インデックスが破損しています（Index corrupted）	

Appendix B: Licence Notice

このドキュメントの内容はPublic Document Licence バージョン1.0（the “License”）の下で公開されています；あなたはこのライセンスの条項を遵守する限りにおいて、このドキュメントを使用することができます。このライセンスのコピーはhttp://www.firebirdsql.org/pdfmanual/pdl.pdf （PDF）およびhttp://www.firebirdsql.org/manual/pdl.html （HTML）より入手可能です。

オリジナルのドキュメントのタイトルはFirebird 2.5 Release Notesです。

オリジナルのドキュメントの筆頭執筆者は、Helen Borrieです。各担当で名前の挙げられた人たちが寄稿しています。