Firebird 2.5 Release Notes

Helen Borrie (Составитель/Редактор)

21 Сентября 2010 - Документ v.0250_57 - для Firebird 2.5.0

Перевод на русский язык – 21 декабря 2010

Firebird 2.5 Release Notes

21 Сентября 2010 - Документ v.0250_57 - для Firebird 2.5.0
Перевод на русский язык – 21 декабря 2010
Helen Borrie (Составитель/Редактор)

Содержание

Список таблиц

Список примеров

Глава 1

Общие сведения

Сообщения об ошибках

- Если Вы думаете, что обнаружили ошибку в этой версии, пожалуйста, обратите внимание на инструкции по сообщению об ошибках, приведенные в статье How to Report Bugs Effectively на сайте проекта Firebird.
- Если Вы думаете, что исправление ошибки не работает или вызвало регрессию, найдите исходное описание ошибки в трекере, откройте его повторно, и следуйте приведенным ниже инструкциям.

Следуйте этим рекомендациям, когда будете пытаться анализировать ошибку:

1. Напишите подробный отчет, с указанием точной версии Firebird. Приведите подробную информацию об операционной системе. Включите воспроизводимый пример в Ваш отчет об ошибке и отправьте его в наш Трекер.
2. Мы будем Вам очень благодарны, если Вы станете постоянным участником тестирования Firebird, подписавшись на Список постоянных тестеров, и будете отправлять нам наилучшие описания ошибок.
3. Если Вы хотите начать обсуждение ошибки или новой функциональности, подпишитесь на форум разработчиков. В этом форуме Вы также можете найти обсуждение любой занесенной в трекер записи.

Документация

Все README-руководства, упомянутые в этом документе, Вы найдете в подкаталоге "doc" Вашей директории с установленной Firebird 2.5.

Автоматически генерируемая страница в трекере содержит ссылки на все ошибки, указанные для этой версии.

-- Команда проекта Firebird

Глава 2

Новое в Firebird 2.5

Главной целью при разработке версии 2.5 было создание основы новой многопоточной архитектуры с более низкоуровневой синхронизацией и большей безопасностью потоков, и которая была бы наиболее общей для архитектур Superserver, Classic Server и Embedded.

Хотя расширение языка SQL не было главной целью этой версии, впервые появилась возможность управлять пользователями посредством SQL-операторов CREATE/ALTER/DROP USER, были добавлены операторы ALTER VIEW и CREATE OR ALTER VIEW. Расширения языка PSQL включают добавление автономных транзакций и возможность выполнять запросы к внешним базам данных с помощью оператора EXECUTE STATEMENT.

Другие нововведения

Другие нововведения и усовершенствования в этой версии:

Усовершенствования администрирования

- Системный аудит и пользовательская трассировка с помощью Services API, позволяющие отслеживать и анализировать все, что происходит в базе данных в режиме реального времени
- Новая системная роль RDB$ADMIN в базах данных с ODS версии 11.2 позволяет SYSDBA передавать свои привилегии другому пользователю для каждой базы данных
- Больше информации в таблицах мониторинга базы данных
- Асинхронное закрытие соединений с базой данных
- Доступ обычным пользователям для мониторинга всех соединений под этим же пользователем (т.е. не только CURRENT_CONNECTION)

Дополнения и усовершенствования языка SQL

- Поддержка регулярных выражений с помощью предиката SIMILAR TO
- Поддержка ALTER COLUMN для вычисляемых полей (computed columns)
- Автономные транзакции в PSQL-модулях (хранимые процедуры, триггеры, анонимные PSQL-блоки)
- Поддержка хранимых процедур в качестве источника данных для представлений (view)

- Предложения GRANTED BY и GRANTED AS в операторах GRANT и REVOKE, позволяющие предоставлять права от имени другого пользователя, отличного от используемого по умолчанию CURRENT_USER
- Оператор REVOKE ALL, позволяющий разом отобрать все привилегии у пользователя/роли
- Поддержка нетипизированных параметров в предикате IS NULL
- Удаление из списка зарезервированных всех ключевых слов (за некоторыми исключениями), которые не указаны как зарезервированные в SQL-стандарте

Расширения возможностей обработки данных

- Новые встроенные функции конвертирования UUID CHAR(16) OCTETS строк в RFC4122-совместимый формат и наоборот
- Возможность обрабатывать 32-битные и 64-битные целые числа как шестнадцатеричные в числовых и строковых литералах

Дополнения API

- Запросы теперь возвращают код завершения (SQLSTATE) по стандарту SQL-2003
- Появилась новая константа «DSQL_unprepare» для использования в функции isc_dsql_free_statement, позволяющая «раскомпилировать» запрос (перевести в состояние «unprepared»)

Интернациональная языковая поддержка

- Предложение COLLATE для оператора CREATE DATABASE
- Возможность изменить последовательность сортировки (COLLATE), используемую по умолчанию для набора символов
- Утилита GBAK теперь поддерживает ключи FIX_FSS_DATA и FIX_FSS_METADATA для корректного восстановления баз данных с данными и/или метаданными в кодировке UNICODE_FSS без необходимости использования скриптов и ручного их исправления
- Нечувствительная к диакритическому знаку (Accent-insensitive) последовательность сортировки (collation) для Unicode

Глава 3

Изменения в ядре Firebird

Главной целью этой версии была переработка многопоточной архитектуры ядра Firebird, чтобы использовать преимущества симметричной многопроцессорной архитектуры (SMP) на многопроцессорных аппаратных средствах. Это значительно улучшает масштабируемость архитектуры Superserver, когда происходит одновременная работа с несколькими базами данных, но самое главное – это появление новой архитектуры «Superclassic», которая заложит основы многопоточной архитектуры Firebird 3.

Новая многопоточная архитектура

Дмитрий Еманов
Владислав Хорсун
Александр Пешков
а также
Николай Самофатов
Роман Симаков

В архитектуре SuperServer были сделаны два основных усовершествования:

1. При работе с несколькими базами данных потоки теперь равномерно распределяются на все доступные процессоры, т.е. потоки, работающие с разными базами данных могут выполняться на разных процессорах.

   > **Замечание**
   >
   > По умолчанию параметр «CpuAffinity» по прежнему привязывает сервер в режиме SuperServer к одному процессору. Для лучшей масштабируемости при работе с несколькими базами данных необходимо изменить значение этого параметра в файле `firebird.conf`.

2. Небольшое улучшение масштабируемости при работе с одной базой данных на аппаратных средствах с поддержкой SMP.

В архитектуре Classic Server изменения более существенны:

1. Сервер в режиме Classic Server теперь может быть многопоточным. По-прежнему остается один рабочий поток на каждый процесс, но теперь возможно использование отдельных потоков для параллельных задач, таких как асинхронное отключение, чистка (sweep), межпроцессные коммуникации с менеджером блокировок и т.д.

2.

На платформе POSIX сервисы в режиме Classic Server теперь тоже запускаются в потоках, а не в отдельных процессах, как раньше.

> **Замечание**
>
> На платформе Windows сервисы в режиме Classic Server стали запускаться в потоках уже в версии 2.1.

3. Embedded-библиотека (libfbembed.so - на платформе POSIX и fbembed.dll - на платформе Windows) теперь тоже поддерживает многопоточность и безопасность потоков и может быть использована в многопоточных приложениях.

4. Тесты подтверждают, что производительность архитектуры Classic Server в этой версии существенно выросла в сравнении с предыдущими версиями.

«Superclassic»

Созданная многопоточная модель для архитектуры Classic Server была выделена в новый режим «Superclassic» из-за возможности обрабатывать несколько работающих потоков (отдельных или объединенных в пул) внутри одного серверного процесса. Новая архитектура сохраняет все особенности архитектуры Classic Server с небольшими отличиями:

- Безопасное выключение (shutdown) сервера доступно для всех платформ
- Новая архиектура более производительна, чем Classic Server – примерно на 15-20%, согласно TPC-тестам
- Новая архитектура использует меньше ресурсов ядра (хотя расход памяти не уменьшился)
- При сбое основного процесса Superclassic происходит закрытие всех установленных соединений
- Известные ограничения Services API, существующие для архитектуры Classic Server (такие как невозможность получить список соединений и активных пользователей), не накладываются на архитектуру SuperClassic.
- На платформе POSIX режим Superclassic не требует использования [x]inetd.

> **Изменена архитектура встраиваемого сервера (Embedded)**
>
> - Встраиваемая библиотека сервера на платформе Windows (fbembed.dll) теперь использует архитектуру Superclassic, а не Superserver как раньше. Блокировка файла базы данных (которая запрещала устанавливать несколько соединений с одной и той же базой данных из разных процессов) заменена глобальной блокировкой, позволяющей одновременные соединения к одной и той же базе данных из разных процессов. Это позволяет облегчить отладку таких приложений, позволяет использовать стандартные утилиты (gbak, gstat) и т.д.
> - Одно и то же соединение может быть использовано в нескольких параллельных потоках (ссылка в трекере: CORE-2498).

Примечания к использованию

Windows

На платформе Windows архитектуры Classic Server и Superclassic представлены одним файлом `fb_inet_server.exe` и выбор архитектуры осуществляется параметром командной строки (по умолчанию используется Classic Server).

Чтобы запустить сервер в режиме Superclassic, используйте параметр `-m[ulti-threaded]`. Запуск сервера в качестве сервиса будет выглядеть следующим образом:

```
instsvc install -multithreaded
```

Запуск сервера в качестве приложения будет выглядеть следующим образом:

```
fb_inet_server -a -m
```

POSIX

Для платформы POSIX архитектура Superclassic представлена новым исполняемым файлом `fb_smp_server`. Он запускает прослушивающий процесс (listener), обрабатывающий запросы на соединение, но не требует наличия и использования *[x]inetd*.

Процесс `fb_smp_server` использует многопоточное ядро, представленное библиотекой `libfbembed.so`, в соответствии с требованиями OSRI. Пакет Classic Server также содержит файл `fbguard` (Guardian), который для архитектуры Superclassic запускает процесс `fb_smp_server` - аналогично тому, как в режиме Superserver запускается процесс `fbserver` при установленном Guardian-е.

Несмотря на то, что архитектуры SuperClassic и Classic Server поставляются в одном инсталляционном пакете, режимом по умолчанию (т.е. активным после установки пакета) является режим Classic Server. Для изменения архитектуры сервера на SuperClassic завершите все акивные процессы Firebird и воспользуйтесь интерактивным скриптом `changeMultiConnectMode.sh`, который находится в подкаталоге `bin`. На запрос

```
Which option would you like to choose: multi-(process|thread) [process] ?
```

введите символ «m» для выбора режима SuperClassic или символ «p» для выбора режима Classic Server. Более подробную информацию см. тут.

Потокобезопасная клиентская библиотека

Дмитрий Еманов
Владислав Хорсун
Александр Пешков

Ссылка в трекере: CORE-707

Клиентские библиотеки, включая библиотеку Embedded, теперь могут использоваться в многопоточных приложениях без необходимости какой-либо дополнительной синхронизации на уровне приложения.

Усовершенствования

Немедленное обнаружение разорванных соединений в режиме Classic Server

Владислав Хорсун

Сервер в режиме Classic теперь немедленно обнаруживает, что какой-то из запущенных дочерних процессов потерял связь с клиентским приложением. В таком случае он завершает любую незаконченную активность, откатывает все активные транзакции и закрывает сетевое соединение.

Ссылка в трекере: CORE-818

Оптимизации

Извлечение данных

Дмитрий Еманов

Усовершенствован механизм обработки данных при запросах, не обращающихся к полям таблиц, например, в запросах типа **SELECT COUNT(*) FROM TABLE**.

Ссылка в трекере: CORE-1598

Использование памяти для BLOB

Адриано дос Сантос Фернандес

Оптимизирован расход памяти, выделяемый под временные BLOB-ы.

Ссылка в трекере: CORE-1658

Повышение производительности Update

Владислав Хорсун

Ссылка в трекере: CORE-2672

Целью данного усовершенствования было уменьшить количество операций записи, выполняемых сервером при осуществлении процедуры «careful write» при изменениях данных. Ранее при выполнении массовых изменений (особенно при многократном редактировании одной и той же записи в одной и той

же транзакции) количество операций записи на диск было очень большим, а в худшем случае страница данных могла записываться на диск после создания каждой новой версии записи.

Решение проблемы заключается в предотвращении циклических ссылок (зависимостей, вызывающих необходимость незамедлительной записи на диск) между страницами с новыми и старыми версиями записей.

Увеличен максимальный размер кеша в 64-битных версиях сервера

Владислав Хорсун

Ссылка в трекере: CORE-1687

Предыдущие 64-битные версии Firebird не могли использовать более 2GB (16K * 128 K) кеша. Теперь 64-битные версии Firebird могут быть настроены так, чтобы поместить целиком в кеш даже базы данных более 2GB, при наличии соответствующих ресурсов (свободной памяти). Это может быть очень полезно в высоконагруженных информационных системах с большим количеством чтений.

Теоретический максимальный размер кеша для 64-битных версий Firebird равен 2^31 - 1 (2,147,483,647) страниц.

Размещение баз данных

Александр Пешков

Ссылка в трекере: CORE-1643

Конфигурационный параметр *DatabaseAccess* получил еще одно назначение: если значение параметра равно Restrict, то первый каталог из перечисленного списка допустимых каталогов будет использоваться как каталог по умолчанию при создании новых баз данных и при поиске файлов баз данных, если не указан полный путь к ним и нет соответствующего псевдонима (alias).

Алгоритм поиска аналогичен алгоритму поиска внешних таблиц (external tables) из списка каталогов параметра ExternalFileAccess и выглядит следующим образом:

1. Сначала файл базы данных ищется во всех каталогах из списка Restrict.
2. Если файл не найден:
    - Если выполняется оператор CREATE DATABASE, то база данных будет создана в первом каталоге из списка Restrict.
    - Иначе будет возвращена ошибка.

Текущий каталог

Текущий каталог по-прежнему используется при поиске файла базы данных, если при **локальном** соединении не был указан полный путь к нему. Но при сетевом соединении (TCP) не рекомендуется соединяться без указания псевдонима (alias) или полного пути файла, поскольку, например, в Windows текущим каталогом при определенных условиях может быть **%system%**.

Расположение DLL-файлов в Embedded под Windows

Адриано дос Сантос Фернандес

Ссылка в трекере: CORE-1814

Механизм определения корневого каталога в версии Embedded на платформе Windows изменен для избежания проблемы «Ад DLL». До сих пор корневым считался каталог, содержащий исполняемый файл самого приложения. Но теперь корневым считается каталог, в котором расположен файл `fbembed.dll`.

Поддержка больших внешних таблиц

Владислав Хорсун

Ссылка в трекере: CORE-2492

Предыдущие версии Firebird при работе с файлами внешних таблиц использовали 32-битные функции ввода/вывода, что ограничивало размер файла 2 GB. Теперь используются 64-битные функции ввода/вывода, если они поддерживаются файловой системой, что позволяет использовать файлы намного больше 2 GB.

Корректная обработка 64-битных значений в функциях получения статистики

Владислав Хорсун
Александр Пешков

Ссылка в трекере: CORE-2619

В предыдущих версиях в функциях получения статистики 64-битные значения обрабатывались некорректно. Решение проблемы заключается в двух изменениях:

a. Чтобы в функциях работы со статистикой использовать 64-битные числа, был изменен внутренний класс **AtomicCounter**.

b. В утилиты *isql* и *qli* были внесены изменения для поддержки 64-битных чисел.

> **Несовместимость со старыми версиями клиентских библиотек**
>
> Чтобы 32-битные утилиты могли работать с 64-битным сервером, были добавлены новые API-функции и структуры (**struct perf64** и **perf64_xxx**) и изменены утилиты *isql* и *qli* для поддержки нового API. Это значит, что утилиты *isql* и *qli* версии 2.5 несовместимы со старыми версиями клиентских библиотек.

Самозащита от некорректных UDF-функций

Адриано дос Сантос Фернандес

Ссылка в трекере: CORE-1937

Если строковая UDF-функция возвращает указатель на память, выделенную не тем же менеджером памяти, что используется сервером Firebird, использование ключевого слова FREE_IT в объявлении функции может вызвать порчу памяти и вызвать падение сервера. Теперь для самозащиты от таких некорректных UDF-функций сервер:

1. выявляет такие UDF-функции и возвращает ошибку при их вызове
2. требует наличия обновленной библиотеки *ib_util* во всех режимах работы сервера, включая Embedded.

Диагностика

Диагностика транзакций

Claudio Valderrama

Ссылка в трекере: CORE-1600

Улучшена диагностика и сообщения об ошибках при некорректно заполненном TPB (блок параметров транзакции) в следующих случаях:

- несовместимые параметры одного типа, указанные вместе, например, **WAIT** и **NOWAIT**, или **READ COMMITTED** и **SNAPSHOT**, или **READ ONLY** и **WRITE**
- неприменимые параметры, например, **[NO] RECORD VERSION** вместе с уровнем изоляции **SNAPSHOT**
- некорректный порядок параметров резервирования таблиц, например, **PROTECTED READ <таблица>** вместо **READ <TABLE> PROTECTED**

Сообщения об ошибках доступа

Александр Пешков

Ссылка в трекере: CORE-1234

В сообщениях об ошибках доступа теперь указываются и название таблицы, и название поля.

Усовершенствования сообщений

Владислав Хорсун

Ссылка в трекере: CORE-2587

В случаях, когда процесс сервера не может выделить область памяти, которая уже используется другим процессом сервера в параллельной сессии Windows, теперь выдается более подробное диагностическое сообщение:

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

Усовершенствования метаданных

Сохранение Default Collation для набора символов

Адриано дос Сантос Фернандес

Ссылка в трекере: CORE-789

Последовательность сортировки (collation), которая используется по умолчанию для набора символов, теперь можно изменить с помощью нового оператора ALTER CHARACTER SET. При этом установленное значение записывается в поле RDB$DEFAULT_COLLATE_NAME системной таблицы RDB $CHARACTER_SETS и сохраняется после backup/restore.

Глава 4

Изменения в Firebird API и ODS

Изменения ODS (структуры данных на диске)

Новый номер ODS

Firebird 2.5 создает базы данных с версией ODS 11.2.

Максимальный размер страницы

Максимальный размер страницы остался прежним - 16 КБ (16384 байт).

Максимальный размер кеша

Максимальное количество страниц, которые могут быть использованы для кеша базы данных, зависит от разрядности Firebird (64 или 32 бита):

- 64-битный :: 2^31 -1 (2,147,483,647) страниц
- 32-битный :: 128,000 страниц (как и в версии 2.1)

Расширения API (Application Programming Interface)

Строка подключения и набор символов

В предыдущих версиях сервер никак не учитывал набор символов, используемый в операционной системе и файловой системе. Начиная с версии 2.5 Firebird позволяет работать с именами файлов и строковыми параметрами, не входящими в подмножество ASCII.

Только DPB-соединения поддерживают эту возможность

В текущей реализации эта возможность поддерживается только при подключении через DPB (параметры соединения с базой данных). Функции Service API (*isc_spb**) эту возможность пока не поддерживают.

isc_dpb_utf8_filename

Новый параметр соединения **isc_dpb_utf8_filename** сообщает серверу Firebird, что имя файла в строке соединения передается в кодировке UTF8 (UTF-8). Если параметр isc_dpb_utf8_filename не указан, сервер будет использовать кодировку операционной системы.

Обратная совместимость

Новый клиент, старый сервер
: При подключении к серверу с версией ниже 2.5 клиентом от версии 2.5 или выше с использованием параметра **isc_dpb_utf8_filename** имя файла конвертируется из UTF-8 в кодировку **операционной системы клиента** и параметр **isc_dpb_utf8_filename** удаляется из DPB. Совместимость гарантируется только, если кодировки операционных систем на сервере и клиенте совпадают.

Новый клиент, новый сервер, без параметра isc_dpb_utf8_filename
: При подключении клиента с версией 2.5 и выше к серверу с версией 2.5 и выше без использования параметра **isc_dpb_utf8_filename** имя файла автоматически конвертируется из кодовой страницы операционной системы в UTF-8 и в DPB добавляется параметр **isc_dpb_utf8_filename**. Совместимость гарантируется только, если кодировки операционных систем на сервере и клиенте совпадают.

Новый клиент, новый сервер, с параметром isc_dpb_utf8_filename
: Когда используется параметр **isc_dpb_utf8_filename** клиент передает имя файла на сервер, не изменяя его. Но клиентское приложение **уже** должно передать имя файла в кодировке UTF-8.

Преобразования кодовой страницы

В ОС Windows используется кодировка Windows ANSI, на всех остальных платформах используется кодировка UTF-8.

Кодировка UTF-8 может быть не лучшим выбором для имени файла. Например, если скрипт или текстовый файл загружается в *isql* или другое приложение, использующее другую кодировку при подключении, то могут возникнуть проблемы с использованием нескольких кодировок в одном файле.

Данная проблема может быть решена с помощью кодов символов Unicode, в том числе клиентами версий ниже 2.5.

Использование кодов символов Unicode

Любой символ Unicode в имени файла в строке подключения теперь может быть закодирован как ASCII символ. Это достигается подстановкой символа # в качестве префикса перед кодом символа из таблицы Unicode, т.е. **#XXXX**, где X - шестнадцатеричный код символа (символы 0-9, a-f, A-F).

Если один из символов в строке подключения оказывается равным #, то можно «экранировать» (удвоить) символ (##) или использовать его Unicode-код **#0023**.

Замечание

Символ # интерпретируется на сервере по новым правилам, даже при использовании клиента версий ниже 2.5.

Поддержка кодов завершения SQLSTATE

Билл Оливер
Дмитрий Еманов

Ссылка в трекере: CORE-1761

Добавлена новая API-функция **fb_sqlstate()** для преобразования статус-вектора ошибки в 5-ти символьный буквенно-цифровой код завершения (SQLSTATE) по стандарту SQL-2003.

- Код завершения SQLSTATE представляет собой соединение 2-х символьного SQL CLASS и 3-х символьного SQL SUBCLASS.
- SQL Операторы теперь возвращают код завершения SQLSTATE.
- Утилита *isql* в случае ошибки теперь выводит код SQLSTATE, а не один код SQLCODE, как было ранее
- Диагностика по кодам SQLCODE устарела и не будет поддерживаться в следующих версиях

Устаревшие коды SQLCODE

Хотя коды SQLCODE являются устаревшими и использование кодов SQLSTATE предпочтительнее, коды SQLCODE пока остаются в Firebird. Функция API *isc_sqlcode()* все еще поддерживается, так же как обработка исключений с помощью **WHEN SQLCODE**.

Приложение A: SQLSTATE содержит список всех кодов завершения SQLSTATE, используемых в этой версии, вместе с соответствующими текстами сообщений.

«Эффективный Unprepare»

Билл Оливер
Дмитрий Еманов

Ссылка в трекере: CORE-1741

Новый параметр *DSQL_unprepare* (числовое значение 4) для функции API *isc_dsql_free_statement()* позволяет переводить DSQL-запросы в состояние «unpreparing».

Ранее функция *isc_dsql_free_statement()* поддерживала только параметры *DSQL_close* (для закрытия именованного курсора) и *DSQL_drop* (для освобождения дескриптора запроса).

Функция fb_cancel_operation

Александр Пешков

Новая API-функция *fb_cancel_operation()* позволяет отменить выполнение текущих операций в конкретном соединении с базой данных путем блокирования API-вызовов.

Синтаксис

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

Параметры

status vector (ISC_STATUS status_vector)*
Указатель на статус-вектор.

db_handle (pointer to a isc_db_handle)
Дескриптор соединения с базой данных.

option (unsigned short: symbol)
Определяет тип действия. Значения параметра:

- *fb_cancel_raise*: отменяет любые операции в указаннном соединении. При этом сервер попытается остановить выполнение текущей операции и вернуть исключение со статус-вектором прерванного API-вызова.

  Пример 4.1.

```
Поток1:                                 Поток2:
------------------------------          ------------------------------

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

- *fb_cancel_disable*: запрещает выполнение команды *fb_cancel_raise* в указанном соединении. Это может быть полезно при выполнении критичеких операций, например, сборки мусора.
- *fb_cancel_enable*: позволяет вновь разрешить выполнение команды отмены операции (fb_cancel_raise), если ранее она ранее была запрещена (fb_cancel_disable). Во вновь создаваемом соединении с базой данных выполнение команды *fb_cancel_raise* разрешено по умолчанию.
- *fb_cancel_abort*: принудительно закрывает клиентское соединение. Полезно при необходимости срочно закрыть соединение. Все активные транзакции будут отменены (Rollback), а клиентское соединение будет закрыто без возврата ошибок (со статусом «Sucess»). **Используйте с осторожностью!**

Использование

Повторять вызов команд *fb_cancel_disable* и *fb_cancel_enable* можно много раз.

Обычно вызов команды *fb_cancel_raise* нужен для остановки "долгоиграющего" запроса, функция при этом должна вызываться из отдельного потока.

> **Обратите внимание на асинхронный характер вызова команды *fb_cancel_raise*!**

С другой стороны, асинхронность выполнения подразумевает, что к концу API-вызова соединение может быть неактивным. Асинхронность также означает, что возвращаемый статус-вектор почти всегда будет возвращать значение *FB_SUCCESS*, хотя возможны и исключения: например, ошибка при передаче пакетов.

Пример

```
Поток A:
// разрешаем выполнение fb_cancel_operation в соединении
fb_cancel_operation(isc_status, &DB, fb_cancel_enable);

// запускаем на выолнение долгоиграющий запрос
isc_dsql_execute_immediate(isc_status, &DB, &TR, 0, "запрос", 3, NULL);

// ожидание завершения API-вызова

    Поток B:
    // отменяем текущую операцию
    fb_cancel_operation(local_status, &DB, fb_cancel_raise);

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

Функция Shutdown

Александр Пешков

Добавлены функции для остановки сервера из клиентских приложений.

Две взаимосвязанные функции fb_shutdown*

Две функции *fb_shutdown** могут использоваться в приложениях для встраиваемого (embedded) сервера: *fb_shutdown()* и *fb_shutdown_callback*.

Прототипы

```
   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()

Функция **fb_shutdown()** выполняет остановку различных подсистем сервера Firebird (yValve, engine, redirector) и разрабатывалась в первую очередь для встраиваемого сервера, так как действие функции распространяется только на текущий процесс.

В настоящее время функция работает только для Embedded-режима. Она прерывает выполнение всех активных запросов, отменяет все активные транзакции, разрывает активные соединения и останавливает сервер.

> **Важно для прикладных разработчиков**
>
> Функция **fb_shutdown()** не останавливает удаленный сервер, к которому может быть подключено приложение, а завершает всю активность в текущем соединении. Все клиентские библиотеки, включая и библиотеку встраиваемого сервера, автоматически выполняют эту команду перед завершением приложения, если клиент подключен хотя бы к одной базе данных или сервису. Следовательно, функция никогда не должна вызываться для удаленного соединения.

Параметры

Функция **fb_shutdown()** имеет два параметра:

1. время ожидания в миллисекундах
2. причина остановки

   Коды причины (**const int reason**) отрицательны и приведены в файле `ibase.h`: имена констант начинаются с **fb_shutrsn**.

> **Замечание**
>
> При вызове функции **fb_shutdown()** из приложения нужно задавать **положительное** значение аргумента.

Возвращаемые значения

- При успешной остановке сервера возвращает ноль.
- Ненулевое значение говорит об ошибках при остановке сервера. Более подробная информация будет записана в файл `firebird.log`.

fb_shutdown_callback()

fb_shutdown_callback() указывает функцию обратного вызова, которая должна быть вызвана при остановке сервера. Этот вызов почти всегда проходит успешно, хотя существуют условия, приводящие к ошибкам - например, отсутствие свободной памяти.

Параметры

fb_shutdown_callback() имеет четыре параметра:

status vector (ISC_STATUS status_vector)*
Указатель на структуру статус-вектора.

pointer to callback function (FB_SHUTDOWN_CALLBACK callback_function)
Указатель на функцию обратного вызова. Функция обратного вызова может иметь три параметра. Первый и второй параметры позволяют определить, какие действия должны быть выполнены:

1. Причина остановки сервера.

   Коды причин приведены в файле `ibase.h`: имена констант начинаются с **fb_shutrsn**. Наиболее интересны две из них:

   - fb_shutrsn_exit_called: Firebird завершается командой exit() или выгружается библиотека встраиваемого сервера
   - fb_shutrsn_signal: применим только для POSIX ОС: сигнал SIGINT или SIGTERM

   > **Замечание**
   >
   > Нужно использовать **положительные** значения констант при вызове вызовов команды **fb_shutdown()**.

2. Актуальное значение маски для вызова

   Этот параметр определяет, когда запускается функция обратного вызова - до или после остановки сервера.

3. Параметр для прикладного применения

   Определяется приложением и может быть использован для любых целей (может иметь значение NULL).

Возвращаемые значения
При успешном выполнении функция обратного вызова вовращает ноль. Отличное от нуля интерпретируется по маске вызова (смотрите далеее описание следующего параметра):

- При вызове *fb_shut_postproviders* это приводит к возврату ненулевого значения функцией **fb_shutdown()**. Функция обратного вызова отвечает за возврат точных причин ошибки.
- При вызове fb_shut_preproviders возврат ненулевого значения говорит о невозможности остановки сервера.

> **Подсказка**
>
> Возврат ненулевого значения при остановке сервера командой exit() является плохой идеей.

Маска вызова - call mask (const int mask)
Может принимать следующие символические значения:

- fb_shut_preproviders: функция обратного вызова выполняется до остановки сервера

- fb_shut_postproviders: функция обратного вызова выполняется после остановки сервера
- Сочетание этих параметров

Значения для маски вызова

fb_shut_confirmation
Сервер запросит подтверждение готовности к остановке

fb_shut_preproviders
Операции дожны быть выполнены до остановки сервера

fb_shut_postproviders
Операции дожны быть выполнены после остановки сервера

fb_shut_finish
Заключительная очистка

Возврат ненулевого значения при fb_shut_confirmation (без флага fb_shut_preproviders) означает, что остановка сервера не будет выполняться.

argument (void arg)*
Это параметр для передачи функции **callback_function**.

Использование функции fb_shutdown

Ниже приводится пример остановки сервера и вызова функции обратного вызова для защиты от некорректного завершения программы при наличии активных соединений по нажатию комбинации клавиш Ctrl-C.

```
#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;
  }
  // другой прикладной код
}
```

Новые константы isc_spb_prp_* для функции Shutdown

Services API поддерживает новые параметры команды и режимы выключения (shutdown) базы данных с помощью констант **isc_spb_prp_***.

isc_spb_prp_shutdown_mode и isc_spb_prp_online_mode

Эти команды используются для выключения (shutdown) и включения (bring online) базы данных и принимают на вход однобайтовые параметры, указывающие режим выключения базы данных, аналогично параметрам команды **gfix -shut**:

- isc_spb_prp_sm_normal
- isc_spb_prp_sm_multi
- isc_spb_prp_sm_single
- isc_spb_prp_sm_full

При выключении базы данных также необходимо задать тип операции:

- isc_spb_prp_force_shutdown
- isc_spb_prp_attachments_shutdown
- isc_spb_prp_transactions_shutdown

После типа операции указывается 4-х байтовый (integer) параметр, определяющий задержку (в секундах) при выполнении операции.

> **Замечание**
>
> Старый формат команд также поддерживается и может использоваться с режимами по умолчанию - выключения (режим 'multi') и включения ('normal').

Примеры использования

Ниже приводятся несколько примеров использования новых параметров утилиты fbsvmgr. Предполагается, что авторизация уже выполнена. Примеры надо набирать одной строкой.

Выключение базы данных и перевод в однопользовательский режим доступа:

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

Выключение базы данных и перевод в многопользовательский режим доступа:

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

Запрет на новые соединения и полное выключение базы данных через 60 секунд:

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

Возврат базы данных в нормальное состояние:

```
fbsvcmgr service_mgr action_properties dbname employee
```

```
prp_online_mode prp_sm_normal
```

Более жесткий контроль изменения заголовка базы данных

Александр Пешков

Для закрытия опасных лазеек несколько параметров DPB были сделаны недоступными обычным пользователям. Это параметры, меняющие заголовок базы данных, и которые потенциально могут повредить ее, если выполняются не под контролем администратора, и параметры, инициирующие операции, разрешенные только для пользователя SYSDBA:

- isc_dpb_shutdown и isc_dpb_online
- isc_dpb_gbak_attach, isc_dpb_gfix_attach и 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 (для архитектуры SuperServer)

Параметр `isc_dpb_set_page_buffers` может использоваться обычным пользователем в архитектуре Classic Server для установки временного размера страничного буфера для текущего соединения. При использовании его пользователем SYSDBA во всех архитектурах размер страничного буфера записывается в заголовок базы данных, т.е. устанавливается его новый постоянный размер.

> **Важное замечание для разработчиков и пользователей драйверов доступа к данным и приложений**
>
> Это изменение затронуло все перечисленные параметры DPB, значения которых были явно установлены (включением в DPB по умолчанию или настройкой на уровне приложения). Например, приложение с параметрами подключения к базе данных 'RESERVE PAGE SPACE=TRUE' и 'FORCED WRITES=TRUE' корректно работало с Firebird до версий 1.5.6, 2.0.5 и 2.1.2, но теперь при соединении обычных (не-SYSDBA) пользователей с ошибкой ISC ERROR CODE 335544788, «Unable to perform operation. You must be either SYSDBA or owner of the database.»

Новые сервисы трассировки

Владислав Хорсун

В Services Manager были добавлены пять новых сервисов для управления сессиями трассировки, каждому соответствует функция Services API.

isc_action_svc_trace_start

Начинает сессию трассировки

Параметры:

```
isc_spb_trc_name : имя сессии трассировки, строка, необязателен
isc_spb_trc_cfg  : конфигурация сессии трассировки, строка, обязателен
```

Второй параметр содержит строку с текстом конфигурации сессии. Файл `fbtrace.conf` находится в корневом каталоге Firebird и содержит шаблон конфигурации (с комментариями и описанием синтаксиса и правил).

> **Замечание**
>
> 1. В отличие от сессий системного аудита, сессии пользовательской трассировки не читают конфигурацию из файла. Разработчик приложения самостоятельно отвечает за корректность текста конфигурации, его сохранение и загрузку.
> 2. Лишние пробелы в строке игнорируются.

Вывод:

- Текстовое сообщение рапортует о статусе операции:

```
Can not start trace session. There are no trace plugins loaded
```

или

```
Trace session ID NNN started
```

- Во втором случае вслед за сообщением начнут выводиться результаты трассировки.

isc_action_svc_trace_stop

Остановка запущенной сессии трассировки

Параметры:

```
isc_spb_trc_id : ID сессии трассировки, integer, обязателен
```

Вывод:

Текстовое сообщение со статусом операции:

- Trace session ID NNN stopped

- No permissions to stop other user trace session
- Trace session ID NNN not found

isc_action_svc_trace_suspend

Приостанавливает активную сессию трассировки

Параметры:

```
isc_spb_trc_id : ID сессии трассировки, integer, обязателен
```

Вывод:

Текстовое сообщение со статусом операции:

- Trace session ID NNN paused
- No permissions to change other user trace session
- Trace session ID NNN not found

isc_action_svc_trace_resume

Возобновляет приостановленную сессию трассировки

Параметры:

```
isc_spb_trc_id : ID сессии трассировки, integer, обязателен
```

Вывод:

Текстовое сообщение со статусом операции:

- Trace session ID NNN resumed
- No permissions to change other user trace session
- Trace session ID NNN not found

isc_action_svc_trace_list

Список существующих сессий трассировки

Параметров не имеет.

Вывод:

Текстовое сообщение показывает список сессий трассировкии и их состояние:

- Session ID: <число>
- name: <строка>. Выводит имя сессии, если оно не пустое
- user: <строка>. Выводит имя пользователя, запустившего сессию

- date: YYYY-MM-DD HH:NN:SS, Дата и время старта сессии
- flags: <строка>, набор флагов через запятую:

  active | *suspend*
  : Состояние запущенной сессии.

  admin
  : Выводит «admin», если сессию запустил администратор. Для сессий, созданных обычными пользователями, ничего не выводит.

  system
  : Показывает «system», если сессия создана сервером (сессия системного аудита). Отсутствует для сессий пользовательской трассировки.

  audit | *trace*
  : Показывает тип сессии: «audit» для сессий системного аудита или «trace» для сессий пользовательской трассировки.

  log full
  : Выводится для сессий трассировки, если файл лога достиг предела.

> **Замечание**
>
> Вывод каждого сервиса может быть получен с помощью вызова isc_service_query с параметром isc_info_svc_line или isc_info_svc_to_eof.

Другие дополнения Services API

Александр Пешков

Использование роли RDB$ADMIN Role в Services API

В SPB (Services Parameter Block - блок параметров сервиса) были добавлены два параметра для предоставления и лишения пользователей Windows прав на роль RDB$ADMIN в базе данных пользователей *security2.fdb*.

> **Замечание**
>
> Эта возможность также доступна с помощью нового параметра **-mapping** утилиты gsec.

Флаг isc_action_svc_set_mapping

Выдает права на роль RDB$ADMIN указанному пользователю Windows в базе данных пользователей *security2.fdb*.

Флаг isc_action_svc_drop_mapping

Забирает права на роль RDB$ADMIN у указанного пользователя Windows в базе данных пользователей *security2.fdb*.

Параметр isc_spb_sec_admin

Новый параметр **isc_spb_sec_admin** позволяет пользователю с правами SYSDBA предоставлять или лишать пользователей прав на роль RDB$ADMIN в базе данных пользователей (*security2.fdb*). Это позволяет дать другим пользователям (кроме SYSDBA) права на создание, изменение и удаление пользователей Firebird.

Параметр **isc_spb_sec_admin** имеет тип spb_long и может принимать значение 0 (для REVOKE ADMIN ROLE) или любое ненулевое число (для GRANT ADMIN ROLE). Более подробная информация приведена в разделе CREATE/ALTER/DROP USER.

Флаг isc_spb_bkp_no_triggers

Новый флаг SPB позволяет использовать в Services API параметр **-nodbtriggers**, добавленный в утилиту gbak в версии 2.1 для отключения триггеров базы данных во время резервного копирования и восстановления базы данных. Он может использоваться в наборе необязательных параметров **isc_spb_options**.

Поддержка nBackup

Ссылка в трекере: CORE-1758

Утилита NBackup выполняет две логические группы операций: блокировку или разблокировку базы данных и ее резервное копирование или восстановление. Ранее блокировку и разблокировку базы данных можно было выполнять удаленно с помощью SQL-запроса ALTER DATABASE, теперь - еще и с помощью новых сервисов Services API:

- isc_action_svc_nbak - инкрементальное резервное копирование
- isc_action_svc_nrest - инкрементальное восстановление базы данных

Параметры:

- isc_spb_nbk_level - уровень резервного копирования (integer)
- isc_spb_nbk_file - имя файла для резервной копии (string)
- isc_spb_nbk_no_triggers - флаг отключения триггеров базы данных

Примеры использования

Ниже приведены примеры использования новых параметров с утилитой fbsvcmgr. Для простоты предполагается, что авторизация уже произведена. Примеры надо набирать одной строкой.

Создание резервной копии уровня 0:

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

Создание резервной копии уровня 1:

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

Восстановление базы данных из этих файлов:

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

Параметры FIX_FSS_DATA и FIX_FSS_METADATA доступны в Services API

Александр Пешков

Ссылка в трекере: CORE-2439

Функции FIX_FSS_DATA и FIX_FSS_METADATA, добавленные как ключи утилиты **gbak -restore** в 2.1 теперь доступны и как параметры команды **isc_action_svc_restore** в Services API.

Новые флаги SPB: **isc_spb_res_fix_fss_data** и **isc_spb_res_fix_fss_metadata**.

Новое API для трассировки

Новое API для трассировки предоставляет набор хуков (обработчиков прерываний), которые выполнены как внешние плагины и могут вызываться сервером для отслеживания любых событий. API пока находится в стадии разработки и не документировано, т.к. может измениться в дальнейших субрелизах.

Дополнительная информация находится в разделе Плагины для трассировки.

Глава 5

Зарезервированные слова

Замечание

Звездочками (*) отмечены зарезервированные ключевые слова, остальные ключевые слова не являются зарезервированными.

Сокращение списка зарезервированных ключевых слов

Александр Пешков

Ссылка в трекере: CORE-2638

Значительно сокращен список зарезервированных ключевых слов, что должно облегчить процесс миграции баз данных с других СУБД. Список зарезервированных ключевых слов был по возможности приближен к требованиям SQL-стандарта.

Список ключевых слов, которые зарезервированы в Firebird, но не являются зарезервированными в SQL-стандарте:

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

Новые зарезервированные ключевые слова:

```
SIMILAR
```

Новые незарезервированные ключевые слова:

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

Глава 6

Изменения в параметрах конфигурации

В файле конфигурации `firebird.conf` были сделаны следующие изменения:

AuditTraceConfigFile

Владислав Хорсун

Этот новый параметр позволяет задать файл с настройками системного аудита. По умолчанию значение параметра не установлено, что значит, что системный аудит не сконфигурирован и не работает.

> **Замечание**
>
> В корневом каталоге Firebird находится файл `fbtrace.conf`, содержащий полный список событий, доступных для трассировки и аудита, с их форматом, правилами и синтаксисом.

Более подробное описание смотрите в разделе Системный аудит.

Параметры настройки использования системного файлового кеша

Добавлены два новых параметра для настройки использования системного файлового кеша.

FileSystemCacheSize

Николай Самофатов

Параметр *FileSystemCacheSize* устанавливает максимальный размер оперативной памяти, используемый системным файловым кешем 64-битными Windows XP или Windows Server 2003 с Service Pack 1 или выше. В версии 2.5.0 на платформах POSIX параметр не используется.

Параметр содержит целое число, представляющее собой количество (в процентах) оперативной памяти, которое может быть использовано под файловый кеш. Значение может быть от 10 до 95 процентов. Если задать значение 0, операционная система сама будет определять размер файлового кеша, отводимый

Firebird. Некорректные значения будут проигнорированы и будет использовано значение по умолчанию равное 30 процентам. Как и другие параметры, значение параметра считывается сервером один раз при запуске.

Привилегии безопасности Windows

Windows требует обладания привилегией SeIncreaseQuotaPrivilege для управления настройками файлового кеша. Эта привилегия доступна по умолчанию администраторам и службам, а также выдается учетной записи Firebird при установке из дистрибутива Windows Installer.

Если Firebird запущен как приложение или в режиме Embedded или установлен не из официального дистрибутива, учетная запись может не иметь данной привилегии. Процесс не выдаст ошибку при запуске, а просто запишет соответствующее сообщение в файл `firebird.log` и будет работать с настройками операционной системы.

FileSystemCacheThreshold

Владислав Хорсун

Параметр был добавлен в версии 2.1 под названием MaxFileSystemCache. Но поскольку он переименован, описание продублировано в этом документе, чтобы обратить внимание обновляющих версию Firebird.

FileSystemCacheThreshold позволяет настроить использование кеша файловой системы. Если параметр установлен в любое положительное целое число, большее чем размер (в страницах) кеша базы данных (указанного в файле конфигурации или заданного на уровне базы данных), то кеш файловой системы будет использоваться и будет ограничен указанным значением, иначе - не будет:

- Чтобы запретить использование системного файлового кеша, установите значение параметра FileSystemCacheThreshold в ноль
- Чтобы разрешить использование системного файлового кеша, установите значение параметра FileSystemCacheThreshold в целое число, значительно превосходящее размер страничного кеша базы данных. Не забывайте, что это поведение зависит от последующих изменений размера страничного кеша базы данных .

Важно

- Это поведение не зависит от платформы (ОС) и способа задания размера страничного кеша - с помощью параметра DefaultDBCachePages или непосредственно на уровне базы данных.
- Параметр FileSystemCacheThreshold по умолчанию установлен в 65536 страниц, т.е. системный файловый кеш по умолчанию разрешен.
- Помните, что если размер страничного кеша базы данных больше значения параметра FileSystemCacheThreshold, то значение параметра будет проигнорировано.

MaxFileSystemCache

Параметр MaxFileSystemCache, добавленный в Firebird 2.1, больше не используется.

ConnectionTimeout

Дмитрий Еманов

В высоконагруженных системах на платформе Windows локальное соединение (XNET) могло завершиться ошибкой из-за таймаута при ожидании от сервера результата вызова функции `xnet_response_event`. Теперь параметр ConnectionTimeout действует и на локальные (XNET) соединения.

Authentication

Александр Пешков

В версии 2.1 был добавлен параметр *Authentication*, позволяющий задать режим аутентификации пользователей. В версии 2.5 параметр имеет то же самое назначение, но произошли и некоторые изменения:

- В версии 2.5 режимы 'mixed' и 'trusted' больше не предоставляют администраторам Windows привилегии SYSDBA автоматически. Более подробное описание приведено в разделе Автоматическое получение прав администраторами Windows.
- Значение параметра по умолчанию изменено с *mixed* на *native*. Для использования Trusted Authentication нужно изменить значение параметра на *mixed* или *trusted*.

  Ссылка в трекере: CORE-2376

MaxUserTraceLogSize

Владислав Хорсун

Задает максимальный суммарный размер временных файлов, создаваемых сессией пользовательской трассировки Services API. По умолчанию установлен лимит в 10MB.

OldSetClauseSemantics

Дмитрий Еманов

До версии 2.5 предложение SET оператора UPDATE присваивало новые значения столбцам в порядке перечисления в операторе, и новые значения были доступны для использования этим же оператором. Это не соответствует SQL-стандарту, который требует, чтобы первоначальные значения столбцов не менялись в процессе выполнения оператора. Теперь в предложении SET доступны только старые (первоначальные) значения столбцов.

Параметр *OldSetClauseSemantics* позволяет восстановить предыдущее поведение в целях совместимости: значение 1 - старый вариант, 0 (по умолчанию) - новый вариант (соответствующий SQL-стандарту).

Внимание

- Этот параметр действует на **все** базы данных на сервере.
- Этот параметр является временным решением для обеспечения обратной совместимости и будет удален в следующих версиях Firebird.

RemoteAuxPort

Дмитрий Еманов

Ссылка в трекере: CORE-2263

Режимы Classic Server и SuperClassic теперь могут быть настроены на использование для событий (events) одного фиксированного порта, заданного параметром RemoteAuxPort. Для режима SuperServer это было доступно начиная с версии 1.5. Это особенно важно для приложений, соединяющихся с базами данных в сети Internet через брандмауэр (firewall) или туннель, поскольку позволяет использовать в таких системах любые режимы работы сервера.

RemoteBindAddress

Александр Пешков

Ссылка в трекере: CORE-2094

Появилась возможность использовать имя компьютера в параметре RemoteBindAddress (раньше допускалось только использование IP-адреса).

Важно

С помощью RemoteBindAddress также можно «привязать» пользовательские соединения к определенному интерфейсу (сетевой карте). Но при использовании имени компьютера ему не должно соответствовать несколько IP-адресов! В частности, проверьте файл `etc/hosts` на всех компьютерах, включая сервер.

RemoteFileOpenAbility

Николай Самофатов

Ссылка в трекере: CORE-2263

Компания Red Soft предоставила код, добавляющий на платформе Windows возможность работать с базами данных, расположенными на сетевых ресурсах. Напомним, на платформе POSIX давно существует возможность работать с базами данных на NFS-устройствах.

Нет никаких гарантий, что использование этой возможности стало более безопасным и надежным, чем раньше. Однако, в определенных случаях это позволяет повысить безопасность базы данных. Например,

можно хранить файл базы данных на USB-диске с ключом, подключаемом к изолированному компьютеру без жестких дисков.

Внимание

Перед тем, как решите использовать эту возможность, обязательно прочитайте примечание к параметру в файле firebird.conf!

Глава 7

Административные возможности

Новая системная роль RDB$ADMIN

Александр Пешков

Для предоставления привилегий SYSDBA другим пользователям была добавлена новая предопределенная системная роль RDB$ADMIN. Любой пользователь, получивший права на роль RDB$ADMIN в конкретной базе, при подключении к этой базе с ролью RDB$ADMIN приобретает все права SYSDBA.

Для выдачи прав на эту роль нужно подключиться к базе данных под учетной записью SYSDBA и дать права на роль RDB$ADMIN пользователю так же, как и на любую другую роль. После этого пользователь может использовать ее при соединении с базой данных как и любую другую роль.

> **Важно**
>
> Если при соединении с базой данных в DPB (параметрах соединения) была указана другая роль, то она **не** будет автоматически заменена на роль RDB$ADMIN и, соответственно, пользователь в этом соединении не будет обладать правами SYSDBA.

Следующий пример показывает предоставление прав SYSDBA пользователям User1 и Admins\ADMIN. Второй пользователь в данном примере - это обычный пользователь Windows (имеет смысл при использовании Trusted Authetication):

```
GRANT RDB$ADMIN TO User1;
GRANT RDB$ADMIN TO "Admins\ADMIN";
```

Несколько баз данных и суперпользователи

Следует иметь в виду, что владение ролью RDB$ADMIN не превращает обычного пользователя в SYSDBA. Она лишь дает пользователю те же права над объектами, что и у SYSDBA, **в той базе данных, в которой ему были выданы права на эту роль**.

- Если пользователю нужны права администратора в нескольких базах данных, то права на роль RDB $ADMIN должны быть ему явно даны в **каждой** из этих баз.
- Если права администратора нужны нескольким пользователям, то каждому из них должны быть выданы права на роль RDB$ADMIN.

- Пользователь, получивший в базе данных роль RDB$ADMIN, может дать права на нее другим пользователям. При этом необязательно указывать WITH ADMIN OPTION (для права предоставлять эту роль другим) или WITH GRANT OPTION (для выдачи разрешений на объекты другим пользователям не будучи владельцем этих объектов) - эти опции устанавливаются неявно (автоматически).

Суперпользователи ОС

На платформах POSIX пользователь *root* всегда имеет права SYSDBA, но в ОС Windows администраторы домена не имели таких прав до версии Firebird 2.1. В версии Firebird 2.1 был добавлен новый конфигурационный параметр *Authentication*, позволяющий администраторам Windows автоматически получать права SYSDBA при использовании Trusted Authentication (аутентификации средствами ОС). На платформах POSIX механизм не поменялся, а в ОС Windows, начиная с версии 2.5, пользователи ОС получают права SYSDBA, только если соединились с ролью RDB$ADMIN (если она указана явно или предоставлена автоматически - см. ниже).

> **Trusted Authentication теперь по умолчанию отключена!**
>
> Параметр *Authentication* в `firebird.conf` теперь (начиная с версии 2.1.3) по умолчанию имеет значение *native*. Для разрешения Trusted Authentication его надо явно установить равным *trusted* или *mixed*.

Автоматическое получение прав администраторами Windows

Для получения привилегий SYSDBA администраторами домена Windows необходимо разрешить Trusted Authentication на сервере Firebird, а сам администратор домена должен получить роль RDB$ADMIN. Метод, описанный выше для обычных пользователей, применим и для администраторов - каждому из них надо дать права на роль RDB$ADMIN в каждой из баз данных.

Тем не менее можно сконфигурировать сервер таким образом, чтобы роль RDB$ADMIN автоматически предоставлялась администраторам при подключении к базе данных (как на POSIX серверах для пользователей с правами root). Новый оператор ALTER ROLE предназначен для этой (и только для этой) цели.

Оператор ALTER ROLE

Для автоматического предоставления роли RDB$ADMIN администраторам Windows при включенной Trusted Authentication нужно подключиться к любой базе данных с правами SYSDBA и выполнить следующий запрос:

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

Чтобы вернуть режим по умолчанию (без автоматического предоставления роли RDB$ADMIN администраторам Windows), выполните следующий запрос:

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

Команды Services API

Эти же действия поддерживаются и в Services API двумя командами: **isc_action_svc_set_mapping** для включения автоматического предоставления роли RDB$ADMIN и **isc_action_svc_drop_mapping** для

отключения автоматического предоставления роли. Эти команды также поддерживаются в утилите *fbsvcmgr*.

Использование RDB$ADMIN для управления пользователями

Новая команда DDL **ALTER USER** позволяет обычному пользователю Firebird (не обладающему правами SYSDBA) изменять свой пароль и/или персональные данные (имя пользователя и др.) Пользователи с правами SYSDBA могут также создавать и удалять пользователей. Более подробно это описано в разделе CREATE/ALTER/DROP USER.

Поскольку база данных *security2.fdb* имеет ODS 11.2 (или должна обновиться до нее), то в ней уже определена также и роль RDB$ADMIN. Так как никто (включая SYSDBA) не имеет возможности подключиться к базе данных пользователей, существуют следующие способы, позволяющие предоставлять роль RDB$ADMIN (и возможность создавать и удалять пользователей) в базе данных *security2.fdb* обычным пользователям:

1. Использовать параметр GRANT ADMIN ROLE в операторах CREATE USER или ALTER USER.

   > **Примечания**
   >
   > Обратите внимание, что GRANT ADMIN ROLE и REVOKE ADMIN ROLE являются не операторами GRANT/REVOKE, а параметрами операторов CREATE USER и ALTER USER.

   Примеры

   Предоставление прав на роль RDB$ADMIN пользователю *alex* в базе данных пользователей:

   ALTER USER alex GRANT ADMIN ROLE;

   Аннулирование прав на роль RDB$ADMIN у пользователя *alex* в базе данных пользователей:

   ALTER USER alex REVOKE ADMIN ROLE;

   Удаление пользователя *alex*

   DROP USER alex;

2. Использование утилиты *gsec* с новым ключом **-admin**. Ключ имеет один аргумент: YES для предоставления пользователю прав на роль RDB$ADMIN или NO для ее аннулирования прав. Более подробное описание смотрите в разделе Выдача прав на роль RDB$ADMIN пользователям Firebird главы *Утилиты командной строки*.

3. Использование нового SPB параметра **isc_spb_sec_admin**, назначающего роль RDB$ADMIN пользователям в базе данных *security2.fdb*. Более подробное описание смотрите в разделе Параметр isc_spb_sec_admin главы *Изменения в Firebird API и ODS*. Утилита *fbsvmgr* также поддерживает использование этого параметра.

Сервисы трассировки и аудита

Владислав Хорсун

Новые средства трассировки и аудита первоначально основывались на TraceAPI, разработанном Николаем Самофатовым - разработчиком Red Soft Database (коммерческого продукта, основанного на коде Firebird).

Обзор возможностей

Новые средства трассировки и аудита позволяют серверу отслеживать и записывать в лог такие события, как подключение и отключение к базе данных, старт и завершение транзакций, выполнение запросов и другие операции.

Трассировка всегда происходит в отдельной сессии, каждая сессия имеет собственную конфигурацию, состояние и вывод.

Сервер Firebird имеет фиксированный список событий, которые могут отслеживаться. Существует два способа трассировки: *системный аудит* и *пользовательская трассировка*, и в зависимости от способа трассировки сервер формирует список событий для сессии. Каждой сессии присваивается уникальный номер - ID, и при старте сессии Services Manager выводит сообщение с этим ID: «Trace session ID nnnn started».

Системный аудит

Сессию системного аудита запускает сам сервер. События, которые будут отслеживаться в этой сессии, задаются в конфигурационном файле и читаются при старте сессии.

Новый параметр AuditTraceConfigFile в файле конфигурации `firebird.conf` задает имя и расположение файла с настройками системного аудита. По умолчанию этот параметр пустой, что означает отсутствие сессий системного аудита. Запущено может быть не более одной системной сессии аудита.

Файл конфигурации системного аудита содержит список отслеживаемых событий и указывает размещение логов трассировки для каждого события. Это позволяет достаточно гибко настроить параметры аудита различных событий для любой базы данных, при этом логирование будет осуществляться в отдельные файлы. Файл с шаблоном настроек `fbtrace.conf` находится в корневом каталоге Firebird и содержит полный список доступных событий с форматом, правилами и синтаксисом для написания файла конфигурации аудита слежения.

> **Подсказка**
>
> Файл `fbtrace.conf` содержит большое количество комментариев, разъясняющих назначение и синтаксис каждого параметра. По мере выпуска субрелизов будут добавляться новые события (и параметры для них) и средства для улучшения возможностей трассировки.

Пользовательская трассировка

Для управления сессиями пользовательской трассировки добавлены пять новых функций Services API:

- isc_action_svc_trace_start - запуск
- isc_action_svc_trace_stop - остановка
- isc_action_svc_trace_suspend - пауза
- isc_action_svc_trace_resume - возобновление
- isc_action_svc_trace_list - получение списка запущенных сессий

Синтаксис функций подробно описан в разделе Новые сервисы трассировки главы *Изменения в Firebird API и ODS*.

Сессии пользовательской трассировки

При запуске сессии пользовательской трассировки из приложения задаются ее имя (необязательный параметр) и конфигурация (обязательный параметр). Конфигурация сессии представляет собой текстовый файл, составленный в соответствии с правилами и синтаксисом (за исключением параметров, связанных с выводом), приведенными в файле шаблона `fbtrace.conf`.

> **Замечание**
>
> Очевидно, файл конфигурации хранится не на сервере, а на клиентском компьютере, с которого запускаеся сессия пользовательской трассировки. Например, утилита командной строки fbsvcmgr позволяет указать файл конфигурации с помощью параметра trc_cfg.

Вывод сессии пользовательской трассировки сохраняется во временные файлы, каждый размером в 1 МБ. После прочтения файла приложением он автоматически удаляется. По умолчанию максимальный размер файла вовода ограничен 10 МБ. Он может быть изменен в большую или меньшую сторону с помощью параметра *MaxUserTraceLogSize* в файле `firebird.conf`.

После запуска сессии пользовательской трассировки чтение ее вывода осуществляется вызовом из приложения функции *isc_service_query()*. Сервис может генерировать вывод быстрее, чем приложение может прочитать его. Если общий размер вывода достигает значения ограничения MaxUserTraceLogSize, то сервер автоматически приостанавливает сессию слежения. После того, как приложение завершит чтение файла (размером 1 МБ), он удаляется, работоспособность восстанавливается и сервер автоматически запускает приостановленную ранее сессию.

Когда приложению нужно остановить сессию, достаточно просто послать запрос на отсоединение от сервиса. В качестве альтернативы приложение может использовать функции **isc_action_svc_trace_*** для остановки, паузы или возобновления сессии трассировки.

Управление сессиями пользовательской трассировки

Любой пользователь может инициировать и управлять сессией трассировки. Обычный пользователь может управлять сессиями только в своих соединениях и не может управлять сессиями, начатыми другими пользователями. Администраторы могут управлять любыми сессиями.

Аварийное завершение

Если все процессы Firebird остановлены, то ни одна из сессий не сохраняется. Это означает, что при закрытии процесса Superserver или Superclassic (остановки службы или завершения сервера, работающего как приложение) все запущенные сессии пользовательской трассировки будут полностью остановлены и команда «resume» не сможет их возобновить. Эта ситуация не распространяется, конечно, на Classic Server, так как каждое подключение к нему создает отдельный серверный процесс.

Пример конфигурации для сессии пользовательской трассировки

Следующие примеры служат основой для составления текстов конфигурации для сессий пользовательской трассировки.

a. Трассировка событий компиляции, выполнения и освобождения всех операторов для соединения с номером 12345

```
<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. Трассировка выполнения операторов INSERT, UPDATE, DELETE и вложенных вызовов процедур и триггеров с показом соответствующих планов (PLAN) и статистики во всех подключениях заданного пользователя к базе данных «mydatabase.fdb».

```
<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>
```

Пользовательская трассирова с помощью командной строки

Для трассировки в интерактивном режиме была добавлена новая утилита командной строки fbtracemgr. Синтаксис ее ключей и параметров подробно описан в главе Утилиты командной строки.

Кроме того, для отправки запросов из командной строки может быть использована общая утилита сервисов fbsvcmgr, как показано в примерах ниже.

a. Запуск сессии с названием «My trace», использованием конфигурационного файла `fbtrace.conf` и выводом результатов на экран:

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

Для остановки этой сессии нажмите Ctrl+C в окне консоли *fbsvcmgr*.

b. Получение списка запущенных сессий пользовательской трассировки:

```
fbsvcmgr service_mgr action_trace_list
```

c. Приостановка сессии с ID 1

```
fbsvcmgr service_mgr action_trace_suspend trc_id 1
```

d. Возобновление ранее приостановленной сессии с ID 1

```
fbsvcmgr service_mgr action_trace_resume trc_id 1
```

e. Остановка сессии с ID 1

```
fbsvcmgr service_mgr action_trace_stop trc_id 1
```

> **Подсказка**
>
> Чтобы узнать ID нужной сессии, выведите список сессий в другом окне консоли.

Область видимости сессий в Windows

Ссылка в трекере: CORE-2588

В Windows использование трассировки вызывает конфликты, если запущено несколько экземпляров сервера в разных сессиях Windows и разрешено запускать трассировку в глобальном пространстве имен. Поэтому область видимости трассировки ограничена только процессами, доступными в текущей сессии Windows.

Примеры использования

Существуют три основных варианта использования:

1. **Постоянный аудит активности сервера**

   Это серверный системный аудит. Администратор создает или редактирует файл конфигурации аудита, указывает его имя в параметре *AuditTraceConfigFile* в файле `firebird.conf` и перезапускает

сервер Firebird. Позже администратор может приостановить, возобновить или остановить эту сессию аудита без перезапуска Firebird.

> **Важно**
>
> Чтобы внесенные в конфигурацию аудита изменения стали доступны серверу, необходим перезапуск Firebird.

2. **Интерактивная трассировка некоторых или всех событий в некоторых или во всех базах данных**

   Приложение (например, утилита *fbtracemgr*) запускает сессию пользовательской трассировки, читает ее вывод и отображает на экране в реальном времени. Пользователь может приостанавливать, восстанавливать и остановить трассировку.

3. **Сбор сведений об активности сервера за длительный период времени (за несколько часов и даже за целый день) для дальнейшего анализа.**

   Приложение, запустившее сессию пользовательской трассировки, регулярно читает ее вывод и сохраняет его в один или несколько файлов. Сессия должна быть остановлена вручную из этого приложения или из другого.

Плагины для трассировки

Новое API для трассировки находится в стадии разработки и будет предоставлять возможность обработки событий с помощью внешних плагинов. API аудита уже есть в Firebird 2.5, но оно будет меняться в следующих суб-релизах, поэтому пока не документировано. Стандартный плагин трассировки представлен файлом `fbtrace.dll (fbtrace.so)`, расположенном в каталоге `\plugins`.

Улучшения мониторинга

Дмитрий Еманов

Firebird 2.5 содержит ряд улучшений мониторинга базы данных, новые таблицы с данными о контекстных переменных и использовании памяти в базах данных с ODS 11.2 и выше. Кроме того, стало возможно закрытие клиентского соединения из другого подключения через MON$-таблицы.

Расширенный доступ для обычных пользователей

Ранее непривилегированные пользователи могли получать информацию, относящуюся только к своему текущему соединению. Теперь они могут получать информацию о всех соединениях, выполненных под своей учетной записью.

Ссылка в трекере: CORE-2233

Примечания

1. В приложениях, конечные пользователи которого осуществляют соединение с базой данных под одним и тем же пользователем Firebird (например, сервера приложений в трехзвенной архитектуре или клиент-серверные приложения с реализацией разделения прав на прикладном уровне), нужно обратить внимание на вопросы производительности и конфиденциальности данных при предоставлении результатов мониторинга конечным пользователям.
2. Это же расширение было реализовано в версии 2.1.2.

Изменения в метаданных мониторинга баз данных с ODS 11.2

Замечание

Метаданные для ODS 11.1 описаны в документации к версии 2.1.

Изменение кодировки в метаданных мониторинга

Набор символов (CHARACTER SET) системного домена RDB$FILE_NAME2, который используется в MON$-таблицах для определения столбцов, относящихся к спецификации файлов, был изменен с NONE на UNICODE_FSS. Это касается столбцов MON$DATABASE_NAME, MON$ATTACHMENT_NAME и MON$REMOTE_PROCESS. Это изменение соответствует новому способу обработки строковых параметров (в т.ч. имен файлов) в DPB.

Ссылка в трекере: CORE-2551

MON$MEMORY_USAGE (использование памяти)

- MON$STAT_ID (ID статистики)
- MON$STAT_GROUP (вид статистики)
  0: база данных
  1: соединение
  2: транзакция
  3: оператор
  4: вызов
- MON$MEMORY_USED (количество используемой памяти, байт)
  Информация о высокоуровневом распределении памяти, выполненном сервером. Может быть полезна для отслеживания утечек памяти и чрезмерного потребления памяти в соединениях, процедурах и т.д.
- MON$MEMORY_ALLOCATED (количество памяти, выделенной ОС, байт)
  Информация о низкоуровневом распределении памяти, выполненном менеджером памяти Firebird - объем памяти, выделенный операционной системой, что позволяет контролировать физическое потребление памяти.

Замечание

Не все записи этой таблицы имеют ненулевые значения, только MON$DATABASE и связанные с выделением памяти объекты имеют ненулевое значение. Малые выделения памяти здесь не фиксируются, а вместо этого добавляются к пулу памяти базы данных.

- MON$MAX_MEMORY_USED (максимальное количество байт, используемое данным объектом)
- MON$MAX_MEMORY_ALLOCATED (максимальное количество байт, выделенное ОС данному объекту)

MON$CONTEXT_VARIABLES (контекстные переменные)

- MON$ATTACHMENT_ID (ID соединения)
  Содержит корректное значение только для контекстных переменных уровня соединения, для переменных уровня транзакции устанавливается в NULL.
- MON$TRANSACTION_ID (ID транзакции)
  Содержит корректное значение только для контекстных переменных уровня транзакции, для переменных уровня соединения устанавливается в NULL.
- MON$VARIABLE_NAME (имя контекстной переменной)
- MON$VARIABLE_VALUE (значение контекстной переменной)

Использование памяти в MON$STATEMENTS и MON$STATE

Статистика использования памяти в MON$STATEMENTS и MON$STATE представляет фактическое потребление ее процессором.

Ссылка в трекере: CORE-1583

Примеры использования

Примеры

TOP-10 операторов, отсортированных по расходу памяти:

```
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
```

Список всех контекстных переменных текущего соединения:

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

Завершение запросов и соединений

Таблицы мониторинга доступны только для чтения. Однако в сервер встроен механизм для удаления (и только удаления) записей в таблицах MON$STATEMENTS и MON$ATTACHMENTS, что позволяет, соответственно, завершить активный запрос и закрыть соединение с базой данных (последнее доступно только в базах данных с ODS 11.2).

Отмена всех активных запросов для заданного соединения:

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

Замечание

- Попытка отмены запросов не выполняется, если в соединении в настоящее время нет никаких выполняющихся операторов.
- После отмены запросов их вызов API-функции вернет ошибку с кодом *isc_cancelled*.
- Последующие запросы в данном соединении не запрещены.

Отключение всех соединений, за исключением своего (текущего):

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

Замечание

- Вся текущая активность в соединении немедленно прекращается и все активные транзакции откатываются.
- Закрытое соединение вернет приложению ошибку с кодом *isc_att_shutdown*.
- Последующие попытки использовать это соединение (т.е. использовать его handle в API-вызовах) вернут ошибки.

Глава 8

Усиление безопасности

Отключено автоматическое предоставление прав SYSDBA администраторам Windows

В версии 2.1 пользователи, состоящие в группе Администраторы, автоматически получали права SYSDBA при соединении в режиме Trusted Authentication. В версии 2.5 по умолчанию администраторам Windows права SYSDBA не предоставляются - это регулируется отдельно для каждой базы данных с помощью оператора SQL:

ALTER ROLE RDB$ADMIN { SET | DROP } AUTO ADMIN MAPPING

> **Замечание**
>
> Более подробное описание роли RDB$ADMIN читайте в разделе Новая системная роль RDB$ADMIN.

Глава 9

Язык определения данных - Data Definition Language (DDL)

В V.2.5 добавлено несколько существенных дополнений и улучшений в DDL.

Ссылки

- Операторы CREATE/ALTER/DROP USER
- Синтаксис изменения представлений (Alter View)
- Поддержка селективных процедур в CREATE VIEW
- Поддержка ALTER COLUMN для вычисляемых полей
- Расширение операторов GRANT и REVOKE
- Оператор ALTER ROLE
- Оператор REVOKE ALL
- Параметр DEFAULT COLLATION для базы данных
- Оператор ALTER CHARACTER SET
- Параметр DIFFERENCES FILE для базы данных

Видимость изменений хранимых процедур в архитектуре Classic Server

Дмитрий Еманов

Ссылка в трекере: CORE-2052

Больше нет проблемы с видимостью измененных хранимых процедур для параллельных соединений в архитектуре Classic Server - теперь изменения видны всем соединениям сразу же после подтверждения транзакции изменения.

Операторы CREATE/ALTER/DROP USER

Александр Пешков

Ссылка в трекере: CORE-696

В версии 2.5 добавлен синтаксис управления пользователями с помощью SQL.

Операторы CREATE USER и ALTER USER также могут включать предложения GRANT ADMIN ROLE и REVOKE ADMIN ROLE, чтобы SYSDBA мог предоставить обычному пользователю роль RDB$ADMIN для доступа к базе данных пользователей. Более подробная информация приведена в разделе Новая системная ролиRDB$ADMIN.

Синтаксис:

Пользователи с правами SYSDBA в текущей базе данных и в системной базе данных пользователей могут добавлять новых пользователей:

```
CREATE USER <username> {PASSWORD 'password'}
  [FIRSTNAME 'firstname']
  [MIDDLENAME 'middlename']
  [LASTNAME 'lastname']
  [GRANT ADMIN ROLE];
```

> **Замечание**
>
> Параметр PASSWORD обязателен при создании нового пользователя. Это первоначальный пароль нового пользователя и пользователь сможет изменить его позже, с помощью оператора ALTER USER.

Пользователи с правами SYSDBA в текущей базе данных и в системной базе данных пользователей могут изменять пароли и атрибуты существующих пользователей. Непривилегированные пользователи могут использовать этот оператор только для изменения своих атрибутов.

```
ALTER USER <username>
  [PASSWORD 'password']
  [FIRSTNAME 'firstname']
  [MIDDLENAME 'middlename']
  [LASTNAME 'lastname']
  [{GRANT | REVOKE} ADMIN ROLE];
```

> **Замечание**
>
> Должен быть указан как минимум один из параметров PASSWORD, FIRSTNAME, MIDDLENAME или LASTNAME.
>
> ALTER USER не позволяет изменять имя пользователя. Если необходимо поменять имя пользователя, надо удалить старого пользователя и добавить нового.

Пользователь с привилегиями SYSDBA в текущей базе данных и в базе данных пользователей может удалять пользователей:

```
DROP USER <username>;
```

Ограничения

Операторы CREATE USER, DROP USER и GRANT/REVOKE ADMIN ROLE доступны только пользователю SYSDBA или пользователям, включенным в роль RDB$ADMIN в текущей базе данных **и** в системной базе данных пользователей. Обычные (не имеющие привилегий SYSDBA) пользователи могут менять пароль и другие атрибуты только своей учетной записи - при попытке изменить атрибуты другого пользователя будет выдана ошибка.

Примеры:

```
CREATE USER alex PASSWORD 'test';

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

ALTER USER alex PASSWORD 'IdQfA';
```

Синтаксис для изменения представлений (view)

Адриано дос Сантос Фернандес

Ранее для изменения представления надо было удалить его и создать заново с новым определением. Это делало изменение представления очень неудобным, особенно при наличии зависимостей. В версии 2.5 добавлены новые операторы ALTER VIEW и CREATE OR ALTER VIEW.

Ссылки в трекере: CORE-770 и CORE-1640

ALTER VIEW

ALTER VIEW позволяет изменять определение представления без его пересоздания.

CREATE OR ALTER VIEW

Оператор CREATE OR ALTER VIEW изменяет определение представления (как ALTER VIEW), если оно существует, или создает его, если оно не существует.

Синтаксис:

```
create [ or alter ] | alter } view <имя представления>
  [ ( <список полей> ) ]
as <оператор select>
```

Пример

```
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;
```

Расширения оператора CREATE VIEW

Поддержка селективных процедур в CREATE VIEW

Адриано дос Сантос Фернандес

Ссылка в трекере: CORE-886

Селективные хранимые процедуры могут использоваться в предложении FROM при определении представления.

Пример

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

Создание представления на основе UNION без списка столбцов

Дмитрий Еманов

Ссылка в трекере: CORE-1402

При создании представления на основе запроса с UNION теперь необязательно указывать список столбцов.

Пример

```
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
```

Извлечение имен столбцов из запроса

Адриано дос Сантос Фернандес

Ссылка в трекере: CORE-2424

Оператор CREATE VIEW теперь может извлекать имена столбцов из запросов, включающих группировку или производную таблицу.

Примеры:

```
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);
```

Поддержка ALTER COLUMN для вычисляемых полей

Адриано дос Сантос Фернандес

Ссылка в трекере: CORE-1454

Поле таблицы, определенное как вычисляемое (COMPUTED BY), может быть изменено с помощью оператора ALTER TABLE...ALTER COLUMN. Изменить можно только выражение, заданное при определении поля.

Синтаксис:

```
alter table <table-name>
  alter <computed-column-name>
  [type <data-type>]
  COMPUTED BY (<expression>);
```

Пример:

```
create table test (
  n integer,
  dn computed by (n * 2)
);
commit;
alter table test
  alter dn computed by (n * 5);
```

Расширения для SQL-привилегий

Александр Пешков

Предложение GRANTED BY

В операторы GRANT и REVOKE добавлены необязательные предложения GRANTED BY и GRANTED AS, позволяющие предоставить права от имени другого пользователя, отличного от CURRENT_USER (используемого по умолчанию).

Синтаксис:

```
grant <right> to <object>
```

```
  [ { granted by | as } [ user ] <имя пользователя> ]

revoke <right> from <object>
  [ { granted by | as } [ user ] <имя пользователя> ]
```

> **Подсказка**
>
> Предложения GRANTED BY и GRANTED AS эквивалентны. GRANTED BY рекомендуется стандартом SQL, но для совместимости с другими серверами (например, Informix) поддерживаем и GRANTED AS.

Пример

При соединении с учетной записью SYSDBA:

```
create role r1; -- владелец роли - SYSDBA
  /* SYSDBA дает права на роль пользователю user1 с правом
     предоставления прав на роль другим пользователям */
grant r1 to user1 with admin option;
  /* SYSDBA дает права на роль от имени пользователя
     user1 с помощью предложения GRANTED BY */
grant r1 to public granted by user1;
```

в isql это выглядит так:

```
SQL>show grant;
/* Grant permissions for this database */
GRANT R1 TO PUBLIC GRANTED BY USER1
GRANT R1 TO USER1 WITH ADMIN OPTION
SQL>
```

ALTER ROLE

Ссылка в трекере: CORE-1660

Для управления присвоения администраторам Windows привелегий SYSDBA при использовании Trusted Authentication добавлен новый оператор ALTER ROLE (для других целей оператор не используется).

> **Замечание**
>
> Описание использования ALTER ROLE и полное описание роли RDB$ADMIN приведены в подразделе Новая Системная Роль RDB$ADMIN главы «Административные функции».

REVOKE ALL

Ссылка в трекере: CORE-2113

Когда пользователь удален из базы данных пользователей (или удален в операционной системе при использовании Trusted Authentication), указанные для него SQL-привилегии в базах данных должны бы-

ли удаляться вручную. Теперь появилась возможность удалить все привилегии одной командой для конкретного пользователя или роли.

Синтаксис:

```
REVOKE ALL ON ALL FROM { <user list> | <role list> }
```

Пример (при соединении с учетной записью SYSDBA):

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

Параметр COLLATION базы данных

Адриано дос Сантос Фернандес

Ссылки в трекере: CORE-1737 и CORE-1803

При создании баз данных с ODS 11.2 и выше теперь можно указывать необязательный параметр COLLATION, связанный с набором символов, что позволяет создавать все текстовые столбцы, домены и переменные с указанной последовательностью сортировки (collation), если не указан другой COLLATE.

> **Подсказка**
>
> Отметим также, что COLLATION, используемый по умолчанию для набора символов в базе данных, может быть изменен с помощью нового синтаксиса ALTER CHARACTER SET.

Синтаксис:

```
create database <имя файла>
    [ page_size <размер страницы> ]
    [ length = <длина> ]
    [ user <имя пользователя> ]
    [ password <пароль> ]
    [ set names <набор символов соединения> ]
    [ default character set <набор символов по умолчанию>
      [ collation <collation по умолчанию> ] ]
    [ difference file <имя файла> ]
```

> **Замечание**
>
> Параметр **DIFFERENCE FILE** не является новым для оператора CREATE DATABASE. Он был добавлен вместе с утилитой nBackup в версии 2.0, но не был документирован. Для получения дополнительной информации смотрите раздел Эволюция оператора CREATE DATABASE в конце данной главы.

Пример

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

Оператор ALTER CHARACTER SET

Адриано дос Сантос Фернандес

Ссылка в трекере: CORE-1803

Как известно, при определении строковых полей таблицы без явного указания предложения COLLATION будет использована последовательность сортировки, которая используется по умолчанию для набора символов поля. В версии 2.5 появилась возможность изменять эту последовательность сортировки с помощью нового оператора ALTER CHARACTER SET.

> **Замечание**
>
> Строковые константы также используют DEFAULT COLLATION.
>
> Последовательность сортировки уже существующих полей таблиц при выполнении оператора ALTER CHARACTER SET не изменяется.

Синтаксис:

```
 ALTER CHARACTER SET <набор символов>
  SET DEFAULT COLLATION <последовательность сортировки>
```

Пример

```
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)  /* будет использован набор символов win1252 и collation win_ptbr */
);

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

/* вернет обе записи поскольку win_ptbr регистронезависимый */
select * from person where name like 'A%';
```

> **Подсказка**
>
> Смотрите также раздел Сохранение Default Collation для набора символов.

Эволюция оператора CREATE DATABASE

DDL поддерживает регистрацию и изменение состояния nBackup в атрибутах заголовка базы данных начиная с версии Firebird 2.0. В этом разделе описывается использование оператора ALTER DATABASE для старта и завершения сохранения «дельта» (промежуточных) данных в отдельном файле.

Дельта-файл базы данных

По умолчанию дельта-файл сохраняется в той же директории, что и база данных. Он имеет то же имя, что и файл базы данных, но с добавкой расширения **.delta**. Хотя обычно нет причин менять имя дельта-файла, это можно сделать с помощью оператора alter database add difference file:

```
alter database add difference file 'путь и имя файла'
```

Указанное имя дельта-файла сохранится в системной таблице `RDB$FILES`.

Для возврата умолчательного имени дельта-файла достаточно выполнить следующую команду:

```
 alter database drop difference file
```

Дополнительную информацию можно прочитать в Release Notes к версии 2.0 или в Руководстве по **nBackup**.

Параметр DIFFERENCES FILE для оператора CREATE DATABASE

C. Valderrama

В Firebird 2.0 появился синтаксис для задания имени дельта-файла в виде дополнительного параметра оператора CREATE DATABASE. Синтаксис команды такой же, что и приведенный в подразделе Параметр COLLATION базы данных. В операторе ALTER DATABASE ключевое слово DIFFERENCE FILE служит параметром для задания файла. Данный параметр позволяет задать имя дельта-файла, который будет создаваться при выполнении команды ALTER DATABASE BEGIN BACKUP или запуске утилиты **nBackup** из командной строки.

Пример использования:

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

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
```

Это не ошибка - мы только задали имя дельта-файла. Теперь выполним следующие команды:

```
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
```

Так как аргумент - имя файла, заключаем его в одинарные кавычки. Использование двойных кавычек приведет к ошибке.

```
]..\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
```

Сообщение корректно. Хотя дельта-файл и был удален командой ALTER DATABASE END BACKUP, его имя было сохранено. Допускается задавать только одно имя дельта-файла.

```
SQL> alter database drop difference file;
SQL> alter database begin backup;
Statement failed, SQLCODE = -607
unsuccessful metadata update
-STORE RDB$FILES failed
-message length error (encountered 278, expected 276)
```

Сообщение об ошибке говорит нам о том, что дельта-файл не создан (т.к. не задано его имя). Сервер в этой ситуации не может использовать и имя дельта-файла по умолчанию.

```
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
```

Это корректное сообщение.

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

Глава 10

Язык манипулиования данными (DML)

В этой главе описаны изменения в языке манипулирования данными SQL.

Ссылки

- SIMILAR TO: поиск по регулярным выражениям
- Поддержка шестнадцатеричных литералов
- Новые функции конвертирования UUID (GUID)
- Расширение функции LIST()
- Поддержка нетипизированных параметров в IS NULL
- Усовершенствования оптимизатора

SIMILAR TO: поиск по регулярным выражениям

Адриано дос Сантос Фернандес

Ссылка в трекере: CORE-769.

Добавлен новый предикат SIMILAR TO, позволяющий проверять строковый аргумент на соответствие регулярному выражению. Предикат применим везде, где уместны логические (boolean) выражения, например, в предложении WHERE, в условиях ограничения целостности CHECK, в PSQL-операторе IF() и т.д.

Синтаксис

```
<предикат SIMILAR> ::= <строка> [ NOT ] SIMILAR TO <регулярное выражение>
    [ ESCAPE <специальный символ> ]

<регулярное выражение> ::=
      <regular term>
    | <регулярное выражение> <символ вертикальной черты> <regular term>

<regular term> ::=
      <regular factor>
    | <regular term> <regular factor>

<regular factor> ::=
      <regular primary>
    | <regular primary> <символ звездочка>
```

```
    | <regular primary> <символ плюс>
    | <regular primary> <вопросительный знак>
    | <regular primary> <repeat factor>

<repeat factor> ::= <фигурная скобка> <нижняя граница>
    [, [ <верхняя граница> ] ] <фигурная скобка>

<нижняя граница> ::= <неотрицательное целое>

<верхняя граница> ::= <неотрицательное целое>

<regular primary> ::=
      <символьный спецификатор>
    | <символ процента>
    | <regular character set>
    | <круглая скобка> <регулярное выражение> <круглая скобка>

<символьный спецификатор> ::=
      <обычный символ>
    | <escaped-символ>

<regular character set> ::=
      <символ подчеркивания>
    | <квадратная скобка> <перечисление символов> ... <квадратная скобка>
    | <квадратная скобка> <символ циркумфлекс> <перечисление символов> ... <квадратная скоб
    | <квадратная скобка> <перечисление включения> ... <символ циркумфлекс>
                                          <перечисление исключения> ... <квадратная скобк

<перечисление включения> ::= <перечисление символов>

<перечисление исключения> ::= <перечисление символов>

<перечисление символов> ::=
      <символьный спецификатор>
    | <символьный спецификатор> <символ минус> <символьный спецификатор>
    | <квадратная скобка> <символ двоеточие> <иденификатор класса символа>
                                          <символ двоеточие> <квадратная скобка>

<иденификатор класса символа> ::=
      ALPHA
    | UPPER
    | LOWER
    | DIGIT
    | SPACE
    | WHITESPACE
    | ALNUM
```

Замечание

1. <обычный символ> - это любой символ кроме открывающих и закрывающих квадратных, круглых и фигурных скобок, символов вертикальной черты, циркумфлекса, минуса, плюса, звездочки, подчеркивания, процента, вопросительного знака и символа <специальный символ>.

2. <escaped-символ> это <специальный символ>, за которым следует один из следующих символов: открывающая или закрывающая квадратная, круглая или фигурная скобка, символ вертикальной черты, циркумфлекса, минуса, плюса, звездочки, подчеркивания, процента, вопросительный знак или <специальный символ>.

Таблица 10.1. Идентификаторы класса символа

Идентификатор	Описание	Примечание
ALPHA	Все латинские буквы (a-z, A-Z)	Включает латинские буквы с диакритическим знаком, если используется нечувствительная к диакритическому знаку (accent-insensitive) последовательность сортировки (collation)
UPPER	Все прописные латинские буквы (A-Z)	Включает строчные буквы, если используется регистронезависимая (case-insensitive) последовательность сортировки (collation)
LOWER	Все строчные латинские буквы (a-z)	Включает прописные буквы, если используется регистронезависимая (case-insensitive) последовательность сортировки (collation)
DIGIT	Все арабские цифры (0-9)	
SPACE	Пробел (ASCII 32)	
WHITESPACE	Все символы-разделители (горизонтальная табуляция (9), перевод строки (10), вертикальная табуляция (11), возврат каретки (13), перевод страницы (12), пробел (32))	
ALNUM	Все латинские буквы (ALPHA) и арабские цифры (DIGIT)	

Руководство к использованию

Проверяет, соответствует ли строка хотя бы одному из перечисленных элементов регулярного выражения <регулярное выражение> <символ вертикальной черты> <regular term>

```
'ab' SIMILAR TO 'ab|cd|efg'   -- истина
'efg' SIMILAR TO 'ab|cd|efg'  -- истина
'a' SIMILAR TO 'ab|cd|efg'    -- ложь
```

Проверяет нулевое или большее количество вхождений элемента в строку: <regular primary> <символ звездочка>

```
'' SIMILAR TO 'a*'         -- истина
'a' SIMILAR TO 'a*'        -- истина
'aaa' SIMILAR TO 'a*'      -- истина
'b' SIMILAR TO 'a*'        -- ложь
'ab' SIMILAR TO 'a*b'      -- истина
'abab' SIMILAR TO 'ab*'    -- ложь
'abab' SIMILAR TO '(ab)*'  -- истина
```

Проверяет одинарное или большее количество вхождений элемента в строку: <regular primary> <символ плюс>

```
'' SIMILAR TO 'a+'     -- ложь
'a' SIMILAR TO 'a+'    -- истина
'aaa' SIMILAR TO 'a+'  -- истина
```

Проверяет одинарное или нулевое количество вхождений элемента в строку: <regular primary> <вопросительный знак>

```
'' SIMILAR TO 'a?'     -- истина
'a' SIMILAR TO 'a?'    -- истина
'aaa' SIMILAR TO 'a?'  -- ложь
```

Проверяет точное (<значение>) количество вхождений элемента в строку: <regular primary> <фигурная скобка> <значение> <фигурная скобка>

```
'' SIMILAR TO 'a{2}'     -- ложь
'a' SIMILAR TO 'a{2}'    -- ложь
'aa' SIMILAR TO 'a{2}'   -- истина
'aaa' SIMILAR TO 'a{2}'  -- ложь
```

Проверяет точное (<значение>) или большее количество вхождений элемента в строку: <regular primary> <фигурная скобка> <значение> <запятая> <фигурная скобка>:

```
'' SIMILAR TO 'a{2,}'     -- ложь
'a' SIMILAR TO 'a{2,}'    -- ложь
'aa' SIMILAR TO 'a{2,}'   -- истина
'aaa' SIMILAR TO 'a{2,}'  -- истина
```

Проверяет количество вхождений элемента в строку от <нижняя граница> до <верхняя граница> раз: <regular primary> <фигурная скобка> <нижняя граница> <запятая> <верхняя граница> <фигурная скобка>

```
'' SIMILAR TO 'a{2,4}'       -- ложь
'a' SIMILAR TO 'a{2,4}'      -- ложь
'aa' SIMILAR TO 'a{2,4}'     -- истина
'aaa' SIMILAR TO 'a{2,4}'    -- истина
'aaaa' SIMILAR TO 'a{2,4}'   -- истина
'aaaaa' SIMILAR TO 'a{2,4}'  -- ложь
```

Проверяет наличие любого (не пустого) символа в строке: <символ подчеркивания>

```
'' SIMILAR TO '_'    -- ложь
'a' SIMILAR TO '_'   -- истина
'1' SIMILAR TO '_'   -- истина
```

```
'a1' SIMILAR TO '_'  -- ложь
```

Проверяет наличие любой строки любой длины (включая пустую строку): <символ процента>

```
'' SIMILAR TO '%'         -- истина
'az' SIMILAR TO 'a%z'     -- истина
'a123z' SIMILAR TO 'a%z'  -- истина
'azx' SIMILAR TO 'a%z'    -- ложь
```

Использование цельного регулярного выражения в качестве подвыражения: <круглая скобка> <регулярное выражение> <круглая скобка>

```
'ab' SIMILAR TO '(ab){2}'    -- ложь
'aabb' SIMILAR TO '(ab){2}'  -- ложь
'abab' SIMILAR TO '(ab){2}'  -- истина
```

Проверяет символ на соответствие одному из перечисленных: <квадратная скобка> <перечисление символов> ... <квадратная скобка>

```
'b' SIMILAR TO '[abc]'    -- истина
'd' SIMILAR TO '[abc]'    -- ложь
'9' SIMILAR TO '[0-9]'    -- истина
'9' SIMILAR TO '[0-8]'    -- ложь
'8' SIMILAR TO '[7-9]'    -- истина
'8' SIMILAR TO '[9-7]'    -- истина
```

Проверяет символ на отсутствие в перечислении: <квадратная скобка> <символ циркумфлекс> <перечисление символов> ... <квадратная скобка>

```
'b' SIMILAR TO '[^abc]'    -- ложь
'd' SIMILAR TO '[^abc]'    -- истина
```

Проверяет символ на наличие в перечислении <перечисление включения>, но отсутствие в перечислении <перечисление исключения>: <квадратная скобка> <перечисление включения> ... <символ циркумфлекс> <перечисление исключения> ... <квадратная скобка>

```
'b' SIMILAR TO '[abc^d]'        -- истина
'b' SIMILAR TO '[a-d^b]'        -- ложь
'b' SIMILAR TO '[abc^b]'        -- ложь
'3' SIMILAR TO '[[:DIGIT:]^3]'    -- ложь
'4' SIMILAR TO '[[:DIGIT:]^3]'    -- истина
```

Проверяет символ на принадлежность классу <идентификатор класса символа> (допустимые идентификаторы классов приведены в таблице Идентификаторы класса символа), может быть использовано с символом циркумфлекса для инвертирования логики: <квадратная скобка> <двоеточие> <идентификатор класса символа> <двоеточие> <квадратная скобка>

```
    '4' SIMILAR TO '[[:DIGIT:]]'  -- истина
    'a' SIMILAR TO '[[:DIGIT:]]'  -- ложь
    '4' SIMILAR TO '[^[:DIGIT:]]' -- ложь
    'a' SIMILAR TO '[^[:DIGIT:]]' -- истина
```

Пример использования специального символа: ESCAPE <специальный символ>

```
    '(' SIMILAR TO '#(' ESCAPE '#'  -- истина
    '#' SIMILAR TO '##' ESCAPE '#'  -- истина
    '#' SIMILAR TO '#'  ESCAPE '#'  -- ошибка
```

Примеры

```
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'); -- нарушение условия check

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

Поддержка шестнадцатеричных литералов

Билл Оливер
Адриано дос Сантос Фернандес

Ссылка в трекере: CORE-1760.

Добавлена поддержка шестнадцатеричных числовых и строковых литералов.

Синтаксис

```
<числовой шестнадцатеричный литерал>  ::= { 0x | 0X } <шестнадцатеричный знак> [ <шестнадца

<строковой шестнадцатеричный литерал> ::= { x | X } <апостроф> [ { <шестнадцатеричный знак>
```

```
<шестнадцатеричный знак> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F
```

Числовые шестнадцатеричные литералы

- Количество шестнадцатеричных знаков в строке не может быть больше 16.
- Если количество шестнадцатеричных знаков больше 8, константа имеет тип данных BIGINT, иначе - тип данных INTEGER. Т.е., например, 0xF0000000 - это -268435456, а 0x0F0000000 равно 4026531840.

Строковые шестнадцатеричные литералы

Строка будет иметь тип данных CHAR(*n*/2) CHARACTER SET OCTETS, где *n* - количество шестнадцатеричных знаков.

Примеры

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

select x'4669726562697264'
  from rdb$database;

select x'CFF0E8E2E5F22C20F7E8F2E0F2E5EBFC21'
  from rdb$database;
```

Новые функции конвертирования UUID (GUID)

Адриано дос Сантос Фернандес

Ссылки в трекере: CORE-1656 и CORE-1682.

Добавлены две новые встроенные функции (UUID_TO_CHAR и CHAR_TO_UUID), позволяющие преобразование UUID (GUID) между форматом CHAR(16) OCTETS и форматом RFC4122.

CHAR_TO_UUID()

Функция CHAR_TO_UUID() преобразовывает UUID (GUID), представленный в формате CHAR(32) ASCII (XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX), в формат CHAR(16) OCTETS, оптимизированный для хранения.

Синтаксис

```
CHAR_TO_UUID( <строка> )
```

Пример

```
select char_to_uuid('93519227-8D50-4E47-81AA-8F6678C096A1')
  from rdb$database;
```

UUID_TO_CHAR()

Функция UUID_TO_CHAR() преобразовывает UUID (GUID), представленный в формате CHAR(16) OCTETS (как результат функции GEN_UUID()), в формат CHAR(32) ASCII (XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX).

Syntax Model

```
UUID_TO_CHAR( <строка> )
```

Example

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

Поддержка нетипизированных параметров в предикате IS NULL

Адриано дос Сантос Фернандес

Ссылка в трекере: CORE-2298

По многочисленным просьбам (в частности, от программистов, использующих Delphi) добавлена возможность использовать «один и тот же» параметр как условие фильтра по полю таблицы и проверять его (параметр) на равенство NULL. Это позволяет разработчикам клиентских приложений (использующим Delphi и другие языки программирования) использовать «отключаемые фильтры» (т.е. условия с использованием параметров, которые «отключаются» присвоением параметру NULL), т.е. использовать запросы следующего типа:

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

На уровне API этот запрос преобразовывается в следующий:

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

Вследствие чего возникает две проблемы:

1. В то время как разработчик рассматривает параметр **:param1** как одну переменную с двумя «ссылками», на уровне API запрос содержит два отдельных и независимых параметра.

2. Тип второго параметра неизвестен и сервер не может его самостоятельно определить.

Для решения этих проблем специально для предиката «? is NULL» добавлен новый тип данных **SQL_NULL**. Работает это следующим образом:

- Если параметр «param1» не равен NULL, клиентская библиотека доступа должна присвоить параметру нужное значение и установить поле **XSQLVAR.sqlind** в NOT NULL.
- Если параметр равен NULL, клиентская библиотека доступа должна присвоить полю **XSQLVAR.sqlind обоих** параметров значение NULL и «очистить» (присвоить NULL) поле **XSQLVAR.sqldata**.

Иными словами, для параметра (?) в предикате **? IS NULL**
:

- Поле **XSQLVAR.sqlind** должно быть установлено в соответствии с NULL/не-NULL состоянием параметра.
- Поле **XSQLVAR.sqldata** параметра с типом SQL_NULL всегда должно быть пустым (NULL).

> **NULL в выходных параметрах**
>
> Если запрос возвращает константу NULL (например, **select NULL from ...**), поле результирующего набора данных будет иметь тип CHAR(1), а не SQL_NULL. Но это может быть изменено в следующих версиях.

Расширение функции LIST()

Адриано дос Сантос Фернандес

Ссылка в трекере: CORE-1453

Добавлена возможность в функции LIST() использовать строковое выражение в качество разделителя.

Синтаксис

```
<Функция LIST> ::= LIST '(' [ { ALL | DISTINCT } ] <выражение> [ ',' <разделитель> ] ')'

<разделитель>  ::= любое строковое выражение
```

> **Замечание**
>
> Если разделитель не указан, то в качестве разделителя будет использована запятая.

Пример

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

Усовершенствования оптимизатора

В оптимизаторе сделаны следующие усовершенствования:

CROSS JOIN
Ссылка в трекере: CORE-2200.

При выполнении операции CROSS JOIN с пустой таблицей оптимизатор немедленно возвращает пустой набор данных.

> **Замечание**
>
> Это усовершенствование добавлено и в версию 2.1.2.

Производные таблицы (derived tables)
Ссылка в трекере: CORE-2029.

Увеличено максимальное количество контекстов для производных таблиц.

Отложенное вычисление DEFAULT-значения
Ссылка в трекере: CORE-1842.

В некоторых случаях преждевременное вычисление значения или выражения, указанного в качестве DEFAULT для поля таблицы, может привести к путанице в отношении правильности значения, указанного при вставке (INSERT) для этого поля. Усовершенствование состоит в том, что DEFAULT-выражение не вычисляется - до тех пор, пока это не станет необходимо (т.е. до попытки вставки записи, без присвоения значения этому полю).

Использование индекса для NOT IN
Ссылка в трекере: CORE-1137.

Для предиката NOT IN стало возможным использование индекса.

Утечка памяти в Undo Log
Ссылка в трекере: CORE-1477.

При большом количестве операций UPDATE в одной и той же транзакции был возможен чрезмерный расход памяти в Undo log. Потребление памяти уменьшено в версии 2.5.

Другие усовершенствования

Другие изменения и исправления:

Обнаружение некорректных UDF-функций, объявленных с помощью FREE_IT
Ссылка в трекере: CORE-1937.

Ранее, функция UDF, объявленная с помощью FREE_IT, могла вызвать падение сервера, если память была выделена не функцией **ib_util_malloc()**. Теперь подобные ситуации корректно обнаруживаются и обрабатываются, но выделенная память остается **неосвобожденной**.

Дополнительная информация при ошибке «Expression evaluation not supported»

Ссылка в трекере: CORE-1799.

Добавлены новые GDS-коды (и тексты ошибок) для обеспечения более подробной информации при возникновении ошибки «Expression evaluation not supported». Например:

```
'Argument for @1 in dialect 1 must be string or numeric'
'Strings cannot be added to or subtracted from DATE or TIME types'
'Invalid data type for subtraction involving DATE, TIME or TIMESTAMP types'
 и т.д.
```

Эта информация следует после GDS-кода в статус-векторе (status vector) ошибки **isc_expression_eval_err (expression evaluation not supported)**.

Глава 11

Процедурный язык (PSQL)

В процедурный язык (PSQL) Firebird (используемый в триггерах, хранимых процедурах и динамически выполняемых блоках) внесено несколько существенных изменений; наиболее важные - новые возможности оператора EXECUTE STATEMENT и автономные транзакции.

Ссылки

- Автономные транзакции
- Использование типа поля таблицы для переменной PSQL
- Использование подзапросов в PSQL-выражениях
- Расширение оператора EXECUTE STATEMENT
  - Особенности использования
    - Аутентификация
    - Режим работы транзакции
    - Наследование привилегий
  - Запросы к внешним базам данных
  - EXECUTE STATEMENT с динамическими параметрами
  - Примеры использования EXECUTE STATEMENT

Автономные транзакции

Адриано дос Сантос Фернандес

Ссылка в трекере: CORE-1409

Это нововведение позволяет выполнять часть кода в PSQL-модуле в автономной (отдельной) транзакции. Это может быть удобно, например, когда необходимо вызвать исключение, но не хочется отменять произведенные в базе данных изменения. Также автономные транзакции позволяют журналировать (логировать) все действия - независимо от способа завершения клиентской транзакции.

Новая транзакция создается с таким же уровнем изоляции, как и у запустившей ее транзакции. Любое исключение, возникшее в блоке кода, выполняющегося в автономной транзакции, вызовет откат всех произведенных в нем изменений. Если выполнение этого блока кода доходит до конца без ошибок, то автономная транзакция подтверждается.

> **Внимание**
>
> Поскольку автономная транзакция независима от запустившей ее транзакции, нужно очень осторожно использовать эту функциональность во избежание блокировок (deadlocks).

Синтаксис

```
IN AUTONOMOUS TRANSACTION
DO
  <простой оператор | составной оператор>
```

Пример использования

```
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 ;!
```

Использование типа поля таблицы для переменной PSQL

Адриано дос Сантос Фернандес

Ссылка в трекере: CORE-1356

В версии 2.0 появилась возможность использования доменов в качестве типов переменных в PSQL. Теперь стало возможным для этих целей использовать тип поля таблицы или представления.

Синтаксис

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

```
        | TYPE OF <domain_name>
        | TYPE OF COLUMN <table or view>.<column>
```

> **Замечание**
>
> TYPE OF COLUMN получает только тип поля. Любые ограничения (constraints) или значения по умолчанию этого поля игнорируются.

Примеры

```
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
```

Расширение оператора EXECUTE STATEMENT

В начале этой главы дается подробное описание нового расширенного синтаксиса PSQL-оператора EXECUTE STATEMENT, а далее описываются новые возможности и особенности их использования.

```
[FOR] EXECUTE STATEMENT <query_text> [(<входные параметры>)]
    [ON EXTERNAL [DATA SOURCE] <строка соединения>]
    [WITH {AUTONOMOUS | COMMON} TRANSACTION]
    [AS USER <имя пользователя>]
    [PASSWORD <пароль>]
    [ROLE <роль>]
    [WITH CALLER PRIVILEGES]
    [INTO <переменные>]
```

> **Замечание**
>
> Приведенный порядок необязательных предложений не является строго фиксированным. Предложения не могут повторяться.

Особенности использования

Если не указаны предложения ON EXTERNAL DATA SOURCE и AS USER, то оператор EXECUTE STATEMENT будет выполнен в контексте текущего соединения (CURRENT_CONNECTION). В остальных случаях оператор будет выполнен в отдельном соединении.

Аутентификация

Для выполнения оператора в отдельном соединении (например, для выполнения запроса к внешнему источнику данных) требуется произвести аутентификацию, для настройки которой предназначены предложения AS USER и PASSWORD. Однако, в некоторых случаях они могут быть пропущены:

1. На платформе Windows для запросов в текущем соединении (т.е. CURRENT_CONNECTION, без обращения к внешним источникам данных) будет использована Trusted Authentication, если она уже используется в текущем соединении и параметр <имя пользователя> не указан, равен Null или совпадает с CURRENT_USER.

2. Если используется соединение с внешним источником данных и параметр <строка соединения> указывает на текущую базу данных (CURRENT_CONNECTION), то будет использована учетная запись текущего пользователя (CURRENT_USER).

3. Если используется соединение с внешним источником данных и параметр <строка соединения> указывает на базу данных, отличную от текущей, будет использована учетная запись операционной системы, под которой запущен сам процесс Firebird.

Во всех остальных случаях, когда предложение PASSWORD пропущено, в DPB (блок параметров соединения) будет присутствовать только параметр isc_dpb_user_name и будет использован традиционный механизм аутентификации.

Режим работы транзакции

В новом синтаксисе есть необязательное предложение, позволяющее управлять транзакцией, в которой будет выполняться оператор: WITH COMMON | AUTONOMOUS TRANSACTION, режим WITH COMMON TRANSACTION используется по умолчанию.

Поведение оператора при использовании WITH COMMON TRANSACTION будет следующим:

a. При соединении с внешним источником данных:

   - при первом (в контексте текущей транзакции) соединении с внешним источником данных будет запущена новая транзакция с такими же параметрами (уровень изоляции и т.д.), как у текущей транзакции (CURRENT_TRANSACTION);

   - при следующих (в контексте текущей транзакции) соединениях с этим же внешним источником данных будет использована ранее запущенная транзакция.

b. При выполнении оператора в контексте текущего соединения будет использована текущая транзакция (CURRENT_TRANSACTION).

При использовании предложения WITH AUTONOMOUS TRANSACTION оператор всегда будет выполняться в новой транзакции, запущенной с такими же параметрами, как у текущей транзакции (CURRENT_TRANSACTION). Эта транзакция будет подтверждена, если оператор выполнится без ошибок, или отменена, если во время выполнения оператора возникли ошибки.

Наследование привилегий

Владислав Хорсун

Ссылка в трекере: CORE-1928

По замыслу исходная реализация оператора EXECUTE STATEMENT изолирует исполняемый код от привилегий доступа вызвавшей его хранимой процедуры или триггера и работает с привилегиями пользователя CURRENT_USER. В целом стратегия правильная, поскольку уменьшает уязвимость при выполнении произвольных операторов. Однако в защищенном окружении или в случаях, когда безопасность является некритичной, эта стратегия может накладывать нежелательные ограничения.

Добавление необязательного предложения WITH CALLER PRIVILEGES позволяет выполняемому коду (оператору EXECUTE STATEMENT) наследовать привилегии доступа вызвавшей его хранимой процедуры или триггера. При подготовке (prepare) оператора будут учитываться все имеющиеся привилегии хранимой процедуры или триггера. Результат будет таким же, как если бы исполняемый оператор был вызван хранимой процедурой или триггером непосредственно (т.е. без оператора EXECUTE STATEMENT).

> **Важно**
>
> Предложение WITH CALLER PRIVILEGES не может использоваться совместно с предложением ON EXTERNAL DATA SOURCE.

Запросы к внешним базам данных

Владислав Хорсун

Ссылка в трекере: CORE-1853

Оператор EXECUTE STATEMENT теперь позволяет выполнять запросы к внешним базам данных с помощью предложения ON EXTERNAL DATA SOURCE.

Параметр <строка соединения>

Параметр <строка соединения> представляет собой обычную строку соединения с базой данных, аналогично передаваемой в API-функцию isc_attach_database(), и имеет следующий формат:

```
[ <имя сервера> [ / <номер порта> ] : ] <путь к базе данных>
```

Набор символов

Соединение с внешней базой данных использует тот же самый набор символов, что и в текущем соединении (CURRENT_CONNECTION).

Привилении доступа

Если внешний источник данных находится на другом сервере, то предложения AS USER <имя пользователя> и PASSWORD <пароль> являются обязательными, а предложение ROLE <роль> является опциональным. Предложение WITH CALLER PRIVILEGES недопустимо, если внешний источник данных находится на другом сервере.

> **Замечание**
>
> В версии 2.5 нет возможности использовать двухфазные транзакции для внешних соединений.

Использование внешнего соединения

При использовании предложения ON EXTERNAL [DATA SOURCE], параметр <строка соединения> которого содержит значение отличное от Null, оператор всегда будет выполняться в отдельном соединении (отличном от CURRENT_CONNECTION) – независимо от того, с какой базой данных и на каком сервере происходит соединение. Кроме того, внешнее соединение будет установлено и при выполнении оператора EXECUTE STATEMENT с использованием предложения AS USER, параметр <имя пользвоателя> которого отличается от CURRENT_USER.

Установленное внешнее соединение будет «жить» (т.е. существовать и при необходимости использоваться) до тех пор, пока существует хотя бы одна транзакция, использующая его, и закрывается независимо от способа завершения последней транзакции – COMMIT или ROLLBACK. Если последняя транзакция, использующая внешнее соединение, подтверждается с помощью CommitRetaining или отменяется с помощью RollbackRetaining, то внешнее соединение не закрывается.

Все локальные коннекты к серверу (в рамках одного процесса сервера) используют общий пул внешних соединений, т.е. установленные внешние соединения могут повторно использоваться. При этом, если оператор EXECUTE STATEMENT выполняется в режиме WITH COMMON TRANSACTION, возможно также повторное использование транзакции – подробнее читайте раздел Режим работы транзакции.

Внешние соединения с одной и той же базой данных на одном и том же сервере, но с разной строкой соединения (параметром предложения ON EXTERNAL), считаются разными соединениями – т.е. при использовании разных протоколов, портов, имен или IP-адресов сервера, а также алиасов баз данных – будут установлены отдельные внешние соединения. При соединении с внешним источником с одной и той же строкой соединения, но под разными учетными записями (AS USER), также будут созданы отдельные внешние соединения.

EXECUTE STATEMENT с динамическими параметрами

Владислав Хорсун
Александр Пешков

Ссылка в трекере: CORE-1221

Новые расширения позволяют использовать оператор EXECUTE STATEMENT с динамическими параметрами аналогично параметризованному DSQL-оператору. Текст самого запроса тоже может быть передан в качестве параметра.

Синтаксис

Механизм использует некоторые соглашения, чтобы облегчить парсинг (синтаксический разбор) запроса и дать возможность использования именованных параметров в стиле аналогично некоторым популярным клиентским инструментам (типа Delphi). Родной синтаксис API, использующий неименованные параметры в указанном порядке, тоже поддерживается. Но совместное использование именованных и неименованных параметров невозможно.

Новый оператор связывания

При реализации поддержки динамических параметров появилась необходимость ввести новый оператор присваивания для связывания во время выполнения именованных параметров с их значениями. Новый оператор подобен оператору присваивания в языке Pascal: «:=».

Синтаксис присвоения параметрам значений

```
<входные параметры> ::=
  <именованный параметр> | <входные параметры>, <именованный параметр>

<именованный параметр> ::=
  <имя параметра> := <выражение>
```

Пример использования **именованных входных параметров**

Следующий PSQL-блок передает текст запроса через параметр и использует именованные входные параметры:

```
EXECUTE BLOCK AS
  DECLARE S VARCHAR(255);
  DECLARE N INT = 100000;
  BEGIN
  /* Обычное присвоение текста запроса строковой переменной */
    S = 'INSERT INTO TTT VALUES (:a, :b, :a)';

    WHILE (N > 0) DO
    BEGIN
      /* Каждая итерация цикла передает текст запроса и
         связывает значения с именованными параметрами */

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

Пример использования **неименованных входных параметров**

Следующий PSQL-блок использует неименованные входные параметры и передает их значения непосредственно по порядку:

```
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
```

Замечание

Обратите внимание, что если текст запроса передается через параметр и одновременно используюся динамические параметры, то текст запроса должен быть заключен в скобки:

```
EXECUTE STATEMENT (:sql) (p1 := 'abc', p2 := 0) ...
```

Примеры использования EXECUTE STATEMENT

Следующие примеры демонстрируют различные варианты применения новых возможностей оператора EXECUTE STATEMENT.

Тесты соединений и транзакций

Повторное использование соединений и транзакций

Выполните этот PSQL-блок несколько раз в одной и той же транзакции – он создает три новых соединения к текущей базе данных и повторно использует их при каждом следующем вызове. Транзакции также используются повторно.

```
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 'sysdba' PASSWORD 'masterkey' -- просто пример
      WITH COMMON TRANSACTION
      INTO :CONN, :TRAN
    DO SUSPEND;

    i = i + 1;
  END
END
```

Открытие нового соединения

Выполните этот PSQL-блок несколько раз в одной и той же транзакции – он создает три новых соединения с текущей базой данных при каждом вызове.

```
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 -- обратите внимание на автономную транзакцию
      INTO :CONN, :TRAN
    DO SUSPEND;

    i = i + 1;
  END
END
```

Пример вычисления входного параметра

Демонстрация того, что выражение вычисляется только один раз:

```
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
```

Тестирование скорости вставки записей

Циклическое выполнение вышеприведенных примеров использования входных параметров для сравнения с непараметризованной формой оператора EXECUTE STATEMENT:

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

-- Прямая вставка записей:

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

-- Вставка записей с помощью подготовленного динамического
-- оператора с использованием именованных входных параметров:

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

-- Вставка записей с помощью подготовленного динамического
-- запроса с использованием неименованных входных параметров:

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
```

Использование подзапросов в PSQL-выражениях

Адриано дос Сантос Фернандес

Ссылка в трекере: CORE-2580

Появилась возможность использовать подзапросы в PSQL-выражениях (ранее для этого приходилось использовать SELECT...INTO и промежуточную переменную). Примеры:

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

Глава 12

Интернациональная языковая поддержка (INTL)

Адриано дос Сантос Фернандес

В Firebird 2.5 были добавлены некоторые усовершенствования, направленные на улучшение интернациональной языковой поддержки.

Параметр COLLATION для базы данных

Базы данных с версией ODS 11.2 и выше теперь могут быть созданы с последовательностью сортировки (collation), связанной с набором символов по умолчанию. Подробности описаны в разделе Параметр COLLATION для базы данных.

Оператор ALTER CHARACTER SET

Синтаксис языка DDL был дополнен возможностью указания последовательности сортировки (collation) по умолчанию для набора символов на уровне базы данных. Подробности описаны в разделе Оператор ALTER CHARACTER SET.

Кодировка строки соединения

В параметры соединения с базой данных (DPB) добавлен параметр, указывающий кодировку (набор символов) строки соединения, что позволяет без проблем соединяться с базами данных, названия файлов которых содержат не-ASCII символы.

Прочитайте раздел Кодировка строки соединения - даже если не сталкивались с такими проблемами.

Другие усовершенствования

Использование маркера набора символов

При использовании маркера (introducer) набора символов, т.е. названия набора символов с префиксом «_» (подчеркивание), для преобразования следующего за маркером строкового литерала в указанный набор символов, могли возникать проблемы, если в одном SQL-операторе были использованы несколько наборов символов. В разных версиях присутствовали разные ошибки - от ошибок транслитерации до просто неожиданного (неправильного) результата.

Проблемы могли возникнуть в двух случаях:

1. Один запрос использует маркер набора символов, в то время как другой запрос читает данные из таблицы MON$STATEMENTS

2. Маркер набора символов используется в PSQL-модуле

В качестве обходного решения этих проблем можно использовать шестнадцатеричное представление ASCII-символов с маркером. Например, вместо

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

можно написать

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

Запрет на некорректные UNICODE_FSS символы

Ссылка в трекере: CORE-1600

Некорректные символы больше не допускаются в данных UNICODE_FSS полей.

Новые ключи для исправления некорректных строк

В утилиту *gbak* добавлены новые ключи для исправления некорректных данных и метаданных в кодировке UNICODE_FSS. Более подробное описание приведено в разделе gbak.

Числовая сортировка строк

Ссылка в трекере: CORE-1945

Для последовательностей сортировки (collation) набора символов UNICODE можно указать параметр NUMERIC-SORT, который задает численный порядок сортировки строк.

Синтаксис и примеры использования:

NUMERIC-SORT={0 | 1}

По умолчанию используется значение 0, обозначающее обычную сортировку в алфавитном порядке:

```
1
10
100
2
20
```

Значение 1 устанавливает сортировку в числовом порядке:

```
1
2
10
20
100
```

Пример создания collation:

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

Наборы символов и последовательности сортировки

UNICODE_CI_AI

Ссылка в трекере: CORE-824

Для набора символов UNICODE добавлена последовательность сортировки (collation) UNICODE_CI_AI, не зависящая от регистра и диакритических знаков.

WIN_1258

Ссылка в трекере: CORE-2185

Для набора символов WIN1258 добавлен псевдоним WIN_1258 для соответствия с остальными наборами символов WIN*.

Наборы символов SJIS и EUCJ

Ссылка в трекере: CORE-2103

Строки в наборах символов SJIS и EUCJ теперь проверяются на корректность.

GB18030

Ссылка в трекере: CORE-2636

Добавлен набор символов GB18030 для поддержки символов и правил китайского языка.

Глава 13

Утилиты командной строки

Несовместимость со старыми версиями клиентских библиотек

Чтобы 32-битные утилиты могли работать с 64-битным сервером, были добавлены новые API-функции и структуры (**struct perf64** и **perf64_xxx**) и изменены утилиты *isql* и *qli* для поддержки нового API. Это значит, что утилиты *isql* и *qli* версии 2.5 несовместимы с более старыми версиями клиентских библиотек.

Более подробное описание приведено в разделе Корректная обработка 64-битных значений в функциях получения статистики.

fbtracemgr

Владислав Хорсун

Ссылка в трекере: CORE-2524

Это новая утилита для работы с новыми возможностями трассировки и аудита. Формат вызова утилиты следующий:

```
fbtracemgr <команда> [<параметры>]
```

Команды:

```
-STA[RT]      запуск новой сессии
-STO[P]       остановка сессии
-SU[SPEND]    приостановка сессии
-R[ESUME]     возобновление сессии
-L[IST]       отображение списка сессий
```

Параметры команд:

```
-N[AME]     <string>  имя сессии
-I[D]       <number>  ID сессии
-C[ONFIG]   <string>  конфигурационный файл сесии
```

Параметры соединения:

```
-SE[RVICE]  <string>  имя сервиса
```

```
-U[SER]     <string>  имя пользователя
-P[ASSWORD] <string>  пароль
-FE[TCH]    <string>  считать пароль из файла
-T[RUSTED]  <string>  использовать Trusted Authentication
```

Примеры:

```
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
```

Примечания

1. Все команды и параметры регистронезависимы.
2. Чтобы выйти из сессии трассировки нажмите Ctrl-C.

Считывание пароля из файла

Александр Пешков

Все утилиты с параметром **-password** уязвимы для перехвата пароля, особенно если запускаются из скрипта. Начиная с версии 2.1, в списке процессов на платформах POSIX пароль отображается звездочками (*), что уже было лучше, чем отображать его в открытом виде.

Следующим шагом на пути скрытия пароля от чужих глаз стала возможность считывания пароля из файла или (на платформах POSIX) из входного потока STDIN.

Новый параметр -fetch_password

В версии 2.5 добавлен новый параметр *-fet[ch_password]*, который можно использовать вместо параметра *-pa[ssword]* во всех утилитах командной строки, требующих аутентификации.

Важно

1. Утилита *qli* поддерживает только сокращенное название параметра «-F».
2. Параметр *-fet[ch_password]* **не может** быть использован для замены параметра *-pw* утилиты *gsec*.

Использование -fetch_password

Параметр принимает одно значение - строку без кавычек и апострофов, содержащую путь и имя файла с паролем. Файл должен быть доступен текущему пользователю операционной системы для чтения. Например:

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

считывает первую строку файла «passfile» в текущем каталоге и использует ее в качестве пароля.

В качестве имени файла можно указать «stdin»:

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

При вызове из командной строки такой команды будет запрошен пароль:

```
Enter password:
```

Подсказка

На платформе POSIX запрос пароля появится также при указании в качестве файла «/dev/tty». Это может быть удобно, например, при восстановлении (restore) из stdin (все в одной строке):

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

gsec

Управление выдачей прав администраторам Windows

Александр Пешков

В версии 2.1 на платформе Windows администраторы домена автоматически получали привилегии SYSBDA. В версии 2.5 администраторы домена получают привилегии SYSBDA автоматически, только если сделана соответствующая настройка в системной базе данных пользователей.

В главе «Возможности администрирования» приведено подробное описание новой системной роли RDB $ADMIN и оператора ALTER ROLE, который позволяет SYSDBA включить или выключить автоматическую выдачу прав на роль RDB$ADMIN администраторам Windows (в том числе в системной базе данных пользователей, к которой происходит неявное обращение при создании/изменении/удалении пользователей).

Автоматической выдачей администраторам Windows прав на роль RDB$ADMIN можно также управлять с помощью утилиты gsec с помощью нового параметра **-mapping**.

Mapping an OS Administrator to the RDB$ADMIN Role

Новый параметр **-mapping** предназначен для включения или выключения автоматической выдачи прав на роль RDB$ADMIN в системной базе данных пользователей администраторам Windows. У параметра только один аргумент: set (для включения) или drop (для выключения):

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

Выдача прав на роль RDB$ADMIN пользователям Firebird

С появлением системной роли RDB$ADMIN появилась возможность наделения пользователей Firebird привилегиями SYSDBA. Однако, ни один пользователь (даже SYSDBA) не может подключаться к системной базе данных пользователей для управления пользователями. Но в операторах CREATE USER и ALTER USER есть предложение GRANT ADMIN ROLE, которое позволяет пользователю с привилегиями SYSDBA выдать другому пользователю права на роль RDB$ADMIN.

То же самое можно сделать с помощью параметра **-admin** утилиты gsec. У этого параметра есть только один аргумент YES (чтобы выдать указанному пользователю права на роль RDB$ADMIN в базе данных security2.fdb) или NO (чтобы забрать права):

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

Консольная справка для gsec

Клаудио Вальдеррама

Ссылка в трекере: CORE-756

У утилиты gsec появились два новых параметра для вызова справки: **-help** и **-?**.

fbsvcmgr

В утилиту fbsvcmgr и Service Parameter Block (SPB) внесены изменения в соответствии с новыми возможностями gsec.

- по аналогии с **gsec -mapping** добавлены два новых параметра: isc_action_svc_set_mapping и isc_action_svc_drop_mapping
- по аналогии с **gsec -admin** новый параметр isc_spb_sec_admin типа spb_long со значением 0 (REVOKE ADMIN ROLE) или любым другим значением (GRANT ADMIN ROLE).

Подробное описание роли RDB$ADMIN приведено в разделе Новая системная роль RDB$ADMIN.

gbak

Параметры для исправления некорректных строк

Адриано дос Сантос Фернандес

Ссылка в трекере: CORE-1831

У утилиты gbak появились два новых параметра для исправления некорретных строк (в данных и метаданных) в кодировке UNICODE_FSS при восстановлении базы данных из резервной копии (restore).

Синтаксис:

```
-FIX_FSS_DATA <кодировка>     -- исправить некорректные данные
-FIX_FSS_METADATA <кодировка> -- исправить некорректные метаданные
```

> **Подсказка**
>
> CORE-2754
>
> Если процесс восстановления базы из резервной копии заканчивается неудачно из-за некорректных данных/метаданных, то утилита gbak выдаст соответствующее сообщение об ошибке.

Сохранение параметра Collation для наборов символов

Адриано дос Сантос Фернандес

Ссылка в трекере: CORE-789

Последовательность сортировки (collation), используемая по умолчанию для набора символов (хранится в поле RDB$DEFAULT_COLLATE_NAME таблицы RDB$CHARACTER_SETS) сохраняется после цикла резервного копирования и восстановления (backup/restore).

nBackup

Ссылка в трекере: CORE-2316

На платформе POSIX создание с помощью утилиты nBackup полной резервной копии большой по размеру базы данных вызывало большой объем дискового ввода/вывода (I/O), что могло мешать работе пользователей. Теперь nBackup сначала пытается считать данные из кеша операционной системы до чтения данных с диска, что значительно снижает объем дискового ввода/вывода.

> **Замечание**
>
> В результате этого нововведения время создания полной резервной копии в условиях высокой нагрузки может возрасти на 10-15%.

isql

Коды SQLSTATE вместо SQLCODE

Клаудио Вальдеррама

Утилита isql теперь возвращает коды завершения SQLSTATE вместо кодов SQLCODE, которые теперь являются устаревшими. Более подробное описание приведено в разделе Поддержка кодов завершения SQLSTATE.

Изменен вывод чисел в экспоненциальной форме

Клаудио Вальдеррама

Ссылка в трекере: CORE-1171

Утилита isql на платформах Windows и POSIX выводила разное количество цифр в экспоненциальной части числа. По умолчанию компиляторы от Microsoft и Intel дополняют экспоненциальную часть ведущим нулем до трех знаков. Например:

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

- В POSIX будет выведено **-2.488355210669293e+39**
- В Windows выводилось **-2.488355210669293e+039**

Теперь isql на платформе Windows тоже выводит экспоненциальные части чисел без ведущего нуля.

Справка по команде SHOW COLLATIONS

Адриано дос Сантос Фернандес

В справку утилиты isql добавлена информация о команде SHOW COLLATIONS.

Ссылка в трекере: CORE-2432

gpre (Precompiler)

Новые функции и переменные

Стефен Бойд
Адриано дос Сантос Фернандес

Ссылка в трекере: CORE-1527

GPRE теперь поддерживает предикат IS NOT DISTINCT, функции CASE, NULLIF, COALESCE, SUBSTRING и контекстные переменные CURRENT_*.

> **Устаревшие функции**
>
> В связи ожидаемым с удалением функции **isc_ddl** в Firebird 3.0, некоторые возможности, существующие в утилитах gdef и gpre, объявляются устаревшими, т.е. они будут работать в версии 2.5, но не будут работать в Firebird 3. Более подробное описание приведено в главе Совместимость.

gstat

Клаудио Вальдеррама

Ссылка в трекере: CORE-1411

Утилита gstat теперь поддерживает параметры **-?** и **-help** для вызова справки по командам и параметрам.

Глава 14

Руководство по установке и миграции

Руководство по установке и миграции

Последняя версия Руководства по установке и миграции для версий 2.0 и 2.1 применима и для версии 2.5. Если этот файл отсутствует в подкаталоге «doc», Вы можете скачать его с сайта Firebird.

Linux (POSIX)

В установке на платформе POSIX были сделаны следующие изменения:

Усовершенствование инсталляционных скриптов

Александр Пешков

- CORE-2195: Скрипты установки для архитектуры Classic Server были усовершенствованы в части назначения прав доступа к документации и другим компонентам.
- CORE-2392: Скрипты инсталляции для архитектур Superserver и SuperClassic для всех поддерживаемых POSIX-систем были усовершенствованы в части установки и настройки Guardian-а.
- CORE-2626: Скрипты автозагрузки в `/etc/init.d` не учитывали наличие в системе каталогов `/var/run/firebird` и `/tmp/firebird`, вследствие чего могли возникать различные ошибки при загрузке системы. В версии 2.5 и выше эти ошибки исправлены.

Ключи для configure

Александр Пешков

Большое количество стандартных ключей утилиты **configure** для настройки каталогов установки на платформе POSIX не работали при установке Firebird. Было почти невозможно использовать стандартные ключи без изменения их значений по умолчанию, из-за их неочевидности и сложности. Вместо них в утилиту **configure** добавлены новые ключи для настройки расположения файлов Firebird. В библиотеку **ib_util** также были внесены изменения для корректной обработки заданной конфигурации.

Доступные ключи

```
--with-fbbin - каталог исполняемых файлов (PREFIX/bin)
```

```
--with-fbsbin - каталог системных исполняемых файлов (PREFIX/bin)
--with-fbconf - каталог файлов конфигурации (PREFIX)
--with-fblib - каталог файлов библиотек (PREFIX/lib)
--with-fbinclude - каталог заголовочных файлов C/C++ (PREFIXinclude)
--with-fbdoc - корневой каталог документации (PREFIX/doc)
--with-fbudf - каталог библиотек UDF (PREFIX/UDF)
--with-fbsample - каталог файлов примеров (PREFIX/examples)
--with-fbsample-db - каталог базы данных примеров (PREFIX/examples/empbuild)
--with-fbhelp - каталог справки к утилите QLI (PREFIX/help)
--with-fbintl - каталог файлов языковой поддержки (PREFIX/intl)
--with-fbmisc - каталог дополнительных файлов (PREFIX/misc)
--with-fbsecure-db - каталог базы данных пользователей (PREFIX)
--with-fbmsg - каталог базы данных сообщений (PREFIX)
--with-fblog - каталог файлов журналов и логов (PREFIX)
--with-fbglock - каталог файлов блокировки Guardian (PREFIX)
--with-fbplugins - каталог plugin-библиотек (PREFIX)
```

Обнаружение исполняемых файлов Firebird

Адриано дос Сантос Фернандес

Ссылка в трекере: CORE-2398

На платформе POSIX обнаружение каталога установки исполняемых файлов Firebird теперь выполняется корректно.

> **Важно**
>
> На данный момент это экспериментальная функциональность. Она отключена в официальных дистрибутивах. Чтобы включить ее, скомпилируйте Firebird из исходных кодов файлом autogen.sh с использованием ключа --enable-binreloc.

Windows

Владислав Хорсун

Ссылка в трекере: CORE-2243

> **Замечание**
>
> Изменения были также внесены в версию 2.1.2, поэтому также описаны в документе «Установка и миграция».

Управление библиотеками MSCV8-runtime

Firebird 2.5 скомпилирован компилятором Microsoft MSVC8 из Visual Studio 2005. Поскольку исполняемые файлы Firebird используют динамическое связывание, они требуют наличия run-time библиотек. Во избежание «Ада DLL» Microsoft ввела новые правила распространения run-time библиотек. Начиная с Windows XP, run-time библиотеки (`msvcp80.dll, msvcr80.dll` и `mscvcm80.dll`) могут распространяться либо как общие библиотеки для всей системы, либо отдельно для каждого приложения.

- Инсталлятор Microsoft MSI Installer устанавливает run-time библиотеки в специальный общий для всех каталог `SxS`, доступный для использования всеми приложениями.
- Второй способ распространения - вместе с приложениями, т.е. файлы run-time библиотек должны располагаться в каталоге исполняемых файлов. Использование каталога `\system32` для файлов run-time библиотек запрещено в операционных системах Windows XP, Server2003 и Vista.

Установка библиотек на уровне системы

Чтобы установить runtime-библиотеки общими для всех приложений, в операционной системе должен быть установлен MSI 3.0 и у пользователя должны быть административные привилегии. Но при распространении приложений с Firebird Embedded это может быть невозможно, поскольку требуется работоспособность приложения без установки (в том числе запуск со сменного носителя). В таких случаях распространяйте runtime-библиотеки вместе с приложением, как описано в следующем разделе.

Распространение библиотек вместе с приложением

Чтобы распространять runtime-библиотеки MSVC8 вместе с приложением, нужно скопировать три DLL-файла (msvcp80.dll, msvcr80.dll и mscvcm80.dll) и файл манифеста (Microsoft VC80.CRT.manifest) в каждый каталог с приложением или DLL-файлом (.exe or .dll), который требует Microsoft MSVC8 runtime.

Типичное приложение, использующее с Firebird Embedded, требует наличия этих трех файлов в каталоге приложения и в подкаталогах intl и udf. Чтобы избежать увеличения общего размера приложения (вследствие трехкратного дублирования файлов), в версии 2.1.2 были изменены пути, по которым ищутся файлы runtime-библиотеки (см. также CORE-2243).

Эти изменения позволяют Firebird Embedded корректно работать, даже если структура каталогов приложения не соответствует правилам распространения runtime-библиотек MSVC8:

a. Библиотеки ib_util.dll, fbudf.dll, ib_udf.dll, fbintl.dll скомпилированы без встроенного манифеста и не ищут файлы runtime-библиотек в своем каталоге. Для корректной работы с ними нужно, чтобы программа уже загрузила runtime-библиотеки MSVC8 до первого обращения к этим DLL-файлам.

b. В библиотеку fbembed.dll добавлен код активации и загрузки своего runtime-манифеста до загрузки других необходимых DLL.

Примечания

a. Настоятельно рекомендуется использование официального дистрибутива Microsoft для установки MSVC8 runtime-библиотек! Установочный файл `vcredist_x86.exe` или `vcredist_x64.exe` (в зависимости от выбранного пакета) должен быть в пакетах установки полной и Embedded версий Firebird. Также он доступен по ссылке на сайте Microsoft.

b. Разрабатываемые UDF-библиотеки должны соблюдать одно из следующих требований, если MSVC8 runtime-библиотеки распространяются отдельно. При компиляции UDF-библиотеки MSVC8 runtime-библиотеки должны **либо**:

- вообще НЕ использоваться
- использоваться, но компилироваться без встроенного манифеста
- использоваться и компилироваться со встроенным манифестом (вариант по умолчанию в IDE MSVC) - в этом случае runtime-библиотеки должны располагаться в каталоге с UDF-файлом.

Глава 15

Обратная совместимость

Дмитрий Еманов

При миграции баз данных с версии 2.0.x или 2.1.x на версию 2.5 необходимо иметь в виду проблемы с совместимостью, которые могут возникнуть в базах данных и использующих их приложениях. Не рекомендуется начинать миграцию до решениях этих проблем.

Несовместимость со старыми версиями клиентских библиотек

Чтобы 32-битные утилиты могли работать с 64-битным сервером, были добавлены новые API-функции и структуры (**struct perf64** и **perf64_xxx**) и изменены утилиты *isql* и *qli* для поддержки нового API. Это значит, что утилиты *isql* и *qli* версии 2.5 несовместимы с более старыми версиями клиентских библиотек.

Более подробное описание приведено в разделе Корректная обработка 64-битных значений в функциях получения статистики.

Проверка метаданных на корректность

Если строковые метаданные базы данных не были предварительно преобразованы в кодировку UTF8, то восстановление (restore) базы данных сервером версии 2.5 выдаст ошибку о некорректных строках в метаданных. Для устранения проблемы нужно использовать файлы в подкаталоге `/misc/upgrade/metadata` и параметры `-fix_fss_data и -fix_fss_metadata` утилиты gbak.

Удаленные параметры конфигурации

Устаревшие параметры конфигурации `OldParameterOrdering` и `CreateInternalWindow` больше не поддерживаются и удалены из файла `firebird.conf`. Кроме того, благодаря новой реализации менеджера блокировок два параметра (`LockSemCount` и `LockSignal`), использующиеся для его настройки в предыдущих версиях, стали ненужными и также удалены. Параметр MaxFileSystemCache переименован в FileSystemCacheThreshold.

Изменения в языке SQL

Зарезервированные слова

Хотя были добавлены новые зарезервированные слова, общее количество зарезервированных слов значительно сокращено путем переноса большого количества несоответствующих SQL-стандарту зарезер-

вированных слов в разряд незарезервированных ключевых слов. Список старых и новых зарезервированных слов приведен в главе Зарезервированные слова.

Проверка UNICODE-строк на корректность

Строки и текстовые BLOB-ы в кодировке UNICODE_FSS теперь проверяются на корректность.

Обработка операций в предложении SET

Ранее в предложении SET оператора UPDATE новые (измененные) значения полей заменяли старые значения и были доступны для последующего изменения этим же оператором. Если одно поле менялось или использовалось несколько раз, то после первого изменения старое значения терялось, что не соответствует требованиям SQL-стандарта.

Теперь на всем протяжении обработки записи старые значения полей доступны для чтения в предложении SET, независимо от количества и порядка обращения к ним.

Но в целях обратной совместимости в файл конфигурации добавлен параметр OldSetClauseSemantics. Этот временный параметр и будет удален в будущих версиях.

Утилиты

fb_lock_print

Поскольку в версии 2.5 базы данных имеют отдельные таблицы блокировок, утилита `fb_lock_print` теперь требует указывать путь к файлу базы данных с помощью параметра `-d <путь>`.

Устаревшие функции

В связи ожидаемым с удалением функции **isc_ddl** в Firebird 3.0, некоторые возможности, существующие в утилитах gdef и gpre, объявляются устаревшими, т.е. они будут работать в версии 2.5, но не будут работать в Firebird 3:

- gdef больше не будет поддерживаться. Вместо нее следует использовать утилиту isql с обычными DDL-запросами.
- В случае использования прекомпилятора gpre замените все DDL-операторы на «EXEC SQL EXECUTE IMMEDIATE "..."»
- Во всех прикладных программах вызовы **isc_ddl** должны быть заменены DDL-запросами.

Изменения в API

Отклонение некорректных параметров TPB

API-функции **isc_start_transaction()** и **isc_start_multiple()** теперь выдают ошибку, если TPB (блок параметров транзакции) содержит несовместимые параметры.

Например, ненулевое значение параметра «wait timeout» несовместимо с параметром «no wait»; параметр «no record version» совместим только с уровнем изоляции транзакции ReadCommitted и т.д. Теперь вместо самостоятельного выбора правильной комбинации параметров сервер выдаст ошибку о некорректном содержимом TPB.

Более подробное описание приведено в разделе Диагностика транзакций.

Константа SQL_NULL

Добавлена константа SQL_NULL для обработки предиката **OR ? IS NULL** без вызова исключения «Data type unknown». Более подробное описание приведено в разделе Поддержка нетипизированных параметров в IS NULL.

Усиление безопасности

Отключено автоматическое предоставление прав SYSDBA администраторам Windows

В версии 2.1 учетные записи, состоящие в группе Администраторы, автоматически получали права SYSDBA по умолчанию. В версии 2.5 предоставление прав SYSDBA регулируется для каждой базы данных отдельно с помощью SQL-оператора **ALTER ROLE RDB$ADMIN { SET | DROP } AUTO ADMIN MAPPING**

Более подробное описание роли RDB$ADMIN читайте в разделе Новая системная роль RDB$ADMIN.

Режим аутентификации по умолчанию (Windows)

В версии 2.1.0 была добавлена возможность аутентификации средствами ОС Windows (Trusted Authentication) и введен конфигурационный параметр Authentication, который по умолчанию имел значение *mixed*, т.е. по умолчанию была разрешена и аутентификация средствами ОС, и обычная аутентификация с помощью базы данных пользователей Firebird. В версии 2.5 параметр по умолчанию имеет

значение *native*. Чтобы разрешить использование Trusted Authentication необходимо вручную установить значение параметра в *mixed* или *trusted*.

Ссылка в трекере: CORE-2376

MacOSX

Для корректной работы сервера в многопоточном режиме необходимо Grand Central Dispatch, который впервые появился в MacOSX 10.6 (Snow Leopard). Таким образом, Firebird 2.5 будет работать только в версии MacOSX 10.6 и выше.

> **Замечание**
>
> Для использования более старых версий MacOSX используйте более старые версии Firebird.

Глава 16

Дистрибутивы для различных платформ

В этой главе описываются существующие дистрибутивы Firebird для редких (неосновных) операционных систем и изменения и усовершенствования в них.

IBM eServer z-Series

Linux/s390 (32-bit)

Дамиан Иванов
Александр Пешков

Ссылка в трекере: CORE-2625

Дистрибутив для платформы Linux/s390 (32-bit) построен на архитектуре s390x. Патч удаляет параметр -DS390X из файла prefix.linux_s390x и заменяет его параметрами __s390__ и __s390x__ , которые поддерживаются компилятором gcc. Это позволяет использовать один префиксный файл для обеих платформ.

> **Замечание**
>
> s390 не имеет ограничений на выравнивание.

Linux/s390x (64-bit)

Дэн Горак
Александр Пешков

Ссылка в трекере: CORE-2559

Дистрибутив для платформы Linux/s390x (64-bit).

Linux/sh4 (Renesas SH)

Нобухиро Ивамацу
Дамиан Иванов
Александр Пешков

Ссылка в трекере: CORE-2655

Дистрибутив для платформы Linux/sh4 (Renesas SH) поддерживает варианты выравнивания little-endian и big-endian.

HP-UX

Таблица блокировок на платформе HP-UX

Александр Пешков

Ссылка в трекере: CORE-2644

Дистрибутивы для всех POSIX-платформ, кроме HP-UX, позволяют таблице блокировок увеличиваться в размерах при нехватке памяти. В HP-UX аппаратные ограничения платформы PA-RISC не позволяют динамически менять размер таблицы, поскольку платформа не поддерживает отображение одного файла на разные виртуальные адреса в одном процессе, поэтому функция **ISC_remap_file** не работает и таблица блокировок не могла увеличиваться в размерах. Разработчики SAS решили данную проблему в своей редакции СУБД Vulcan и этот патч был импортирован и в дистрибутив Firebird 2.5 для HP-UX.

Поддержка старых платформ Windows 32-bit

Николай Самофатов

Ссылка в трекере: CORE-2609

Поддержка операционных систем Microsoft Windows 98, ME и NT4 в версии Firebird 2.5 не планировалась, однако компания Red Soft, которая осуществляет поддержку систем Win98/ME и NT4 для государственных организаций в России, запросила разрешение для включения соответствующих условий компиляции (выделенных в отдельные файлы), позволяющих скомпилировать дистрибутив Firebird 2.5 для этих старых версий Windows.

Эти дистрибутивы не являются основными для Firebird 2.5!

- Эти дистрибутивы не проходят QA-тестирование в рамках проекта.
- Официальные Windows-дистрибутивы Firebird не поддерживают вышеперечисленные старые версии Windows.
- Любая техническая поддержка, разработка и исправление ошибок в этих дистрибутивах осуществляется не командой проекта Firebird, а теми, кто их использует.

Глава 17

Исправленные ошибки

Ошибки версии Firebird 2.5.0

Ядро сервера и API

CORE-3115 А. Пешков, Д. Коваленко

Ошибки в процедурах сжатия записей.

CORE-3103 Д. Еманов

Оператор SELECT выполняет больше неиндексированных чтений в 2.5 RC3, чем в 2.1.3.

CORE-3101 А. дос Сантос Фернандес, Д. Еманов

Выполнение оператора ALTER DOMAIN было невозможно в базах данных, мигрированных с более ранних версий.

CORE-3100 В. Хорсун

Параметры режима WAIT и таймаута блокировки внешней транзакции, создаваемой оператором EXECUTE STATEMENT, не соответствовали параметрам локальной транзакции.

CORE-3096 А. дос Сантос Фернандес

Исправление ошибки CORE-2893 вызвало регрессию при компилировании запросов.

CORE-3094 А. дос Сантос Фернандес

При использовании в предикате NOT IN селективной хранимой процедуры не обрабатывались ее входные параметры.

CORE-3090 А. дос Сантос Фернандес

Неправильный результат соединения LEFT JOIN таблицы и константного подзапроса.

CORE-3089 В. Хорсун

Ошибка при выполнении запросов к внешнему источнику данных InterBase 4.1 (ODS 8) с помощью оператора EXECUTE STATEMENT.

CORE-3079 В. Хорсун

Падение производительности при большом количестве вставок записей в одной транзакции при наличии триггеров, рассылающих события (events).

Сбои сервера

CORE-3109 А. дос Сантос Фернандес

При вызове функции `isc_dql_exec_immed3_m()` с запросом **CREATE DATABASE** с дескриптором транзакции равным NULL происходил сбой сервера.

Утилиты командной строки

gsec

CORE-3116 А. Пешков

Утилита `gsec` выводила список пользователей в поток `stderr` вместо of `stdout`.

Ошибки предыдущих версий

Ядро сервера и API

CORE-3034 В. Хорсун

Отмена запроса (например, из-за выключения (shutdown) базы данных или по команде пользователя) во время вставки ключа в индекс по выражению могла привести к ошибке «bugcheck 300 (can't find shared latch)».

CORE-3016 В. Хорсун

В некоторых случаях при закрытии соединения с базой данных в файл `firebird.log` записывалось сообщение об ошибке «Fatal lock manager error: invalid lock id (0), errno: 0».

CORE-3015 В. Хорсун

Исправлены ошибки «Cannot initialize the shared memory region».

CORE-3003 Д. Еманов

Добавлена проверка на наличие оператора SUSPEND в процедурах, вызываемых с помощью оператора SELECT и если оператор SUSPEND отсутствует, будет выдано сообщение об ошибке «Procedure ... is not selectable (it does not contain a SUSPEND statement)». Но если эта ошибка обнаруживается во время восстановления базы данных из резервной копии, то процесс восстановления теперь не прерывается, а вместо ошибки выводится предупреждение (Warning) с тем же текстом.

CORE-2995 В. Хорсун

Дублирование ошибки в статус-векторе в некоторых случаях.

CORE-2993 В. Хорсун, Д. Еманов

В высоконагруженных системах при работе с таблицами мониторинга (MON$) могла появляться ошибка «Fatal lock manager: Invalid lock id (NNN)».

CORE-2900 А. дос Сантос Фернандес

Запрос, содержащий DISTINCT, может приводить к регулярным, но случайным ошибкам доступа к памяти (access violation).

CORE-2977 В. Хорсун

Индексированные поля типа `DATE` некорректно обрабатывались в базах данных с ODS < 10.

CORE-2965 Д. Еманов

Функция `ROW_COUNT` возвращает некорректное значение после запроса с предикатом `SINGULAR`, поскольку `ROW_COUNT` не должна учитывать количество строк, обработанных подзапросом.

CORE-2956 В. Хорсун

Некорректная обработка параметров процедур в ядре сервера.

CORE-2943 В. Хорсун

Ошибки при разборе запросов с двумя рекурсивными частями.

CORE-2936 В. Хорсун

Удаление двух последовательных листовых страниц индекса при сборке мусора одновременно двумя различными соединениями могло вызвать порчу индекса. Когда освобожденные страницы использовались вновь, возвращалась ошибка «Wrong page type (expected 7 found N)».

CORE-2916 Д. Еманов

Не обрабатывались ошибки при преобразовании во время создания индекса.

CORE-2893 А. дос Сантос Фернандес

Выражение с подзапросом могло быть ошибочно принято за константное.

CORE-2879 В. Хорсун

Чистка (sweep) могла вызвать ошибку «page 0 is of wrong type (expected 6, found 1)».

CORE-2876 А. Пешков

Некорректная обработка ошибок при выполнении оператора **ALTER DATABASE ADD DIFFERENCE FILE** могла привести к неправильному состоянию базы данных.

CORE-2875 А. дос Сантос Фернандес

Сравнение поля типа CHAR длиннее 4096 байт со строковой константой могло привести к ошибке.

CORE-2871 Д. Еманов

Если представление (view) или производная таблица (derived table), используют внешнее соединение (LEFT/RIGHT JOIN) с сортировкой и внешний запрос тоже содержит предложение ORDER BY, то предложение ORDER BY внешнего запроса не будет использовано.

CORE-2858 К. Вальдеррама

Порча памяти при формировании исключений (exception) о нарушении прав доступа.

CORE-2856 В. Хорсун

При удалении ключа из уникального индекса он мог быть не найден.

CORE-2833 Д. Еманов

Изменение данных, использованных в индексе по выражению, могло вызвать ошибку, если эти данные содержали ссылки на поля с типом date, содержащие NULL-значения.

CORE-2826 А. дос Сантос Фернандес

Ошибки при проверке условий Join в базах данных с кодировкой UTF-8.

CORE-1089 Д. Еманов

Выборка из представления, основанного на запросе с использованием DISTINCT и LEFT JOIN, возвращает записи в неправильном порядке, если предложение ORDER BY не включает поля из присоединенной (JOIN) таблицы.

CORE-195 А. Пешков

Рецидив старой ошибки, исправленной в версии 1.5.1, вызывающей «bugcheck 291 (cannnot find back record version)» при изменении одной и той же записи, измененной в триггере BEFORE UPDATE. В версии 2.0 ошибка затрагивала только первую (физически) запись таблицы.

CORE-2822 Д. Еманов

При использовании в подзапросе сложных производных таблиц (derived table) происходила ошибка «no current row for fetch operation».

CORE-2820 В. Хорсун

Запросы с PLAN ORDER содержали небольшую утечку памяти.

CORE-2815 В. Хорсун

При определенных условиях служебная страница со списком страниц (page inventory page) отмечалась как измененная уже после того, как ее фактически изменили.

CORE-2785 А. дос Сантос Фернандес

В определенных условиях возникала ошибка транслитерации BLOB-полей: при использовании строки в однобайтовой кодировке в качестве описания (COMMENT) объекта базы данных превышался максимальный размер (64KB) BLOB-сегмента. Эти лишние байты вызывали ошибку транслитерации вместо перенесения их в следующий сегмент.

CORE-2783 В. Хорсун

При использовании рекурсивных запросов в списке выбираемых полей оператора SELECT и сортировке по этому полю происходила ошибка доступа (access violation).

CORE-2730 А. Пешков

На платформе RISC возникала ошибка при преобразовании литерала (который не имеет требований к выравниванию) к DB_KEY, который требует выравнивания до 8 байт (QWORD).

CORE-2722 А. дос Сантос Фернандес

Сохранение некорректных BLOB при копировании из BLOB с кодировкой NONE/OCTETS.

CORE-2660 А. дос Сантос Фернандес

Если при использовании внешнего соединения (OUTER JOIN) с запросом, содержащим функцию COUNT(*), условие соединения возвращало ложь, то запрос использовал в качестве результата соединения 0, а не NULL, как должен был.

CORE-2659 Д. Еманов

План запроса с внешним соединением с участием представлений (views) мог быть неоптимальным, не используя существующий индекс.

CORE-2640 В. Хорсун, Д. Еманов

При определенных условиях менеджер блокировок мог не обнаружить deadlock, что могло привести к сбою сервера .

CORE-2635 В. Хорсун

Уникальный индекс мог быть поврежден, если содержал много NULL-ключей.

CORE-2632 В. Хорсун

При работе с таблицами мониторинга (MON$) могли возникать ошибки «Invalid BLOB ID».

CORE-2616 В. Хорсун

При высокой нагрузке могла возникать ошибка «page N is of wrong type (expected 7, found 5)» с ложным сообщением о порче базы данных (после перезапуска никаких признаков порчи не обнаруживлось).

CORE-2608 Н. Самофатов

В последних версиях Windows (64-bit XP и выше, 32-bit Vista и выше) при работе с большими базами данных операционная система могла использовать всю оперативную память для кеша файловой системы и перестать отвечать на запросы, что могло также привести к сбою (BSOD) самой Windows.

Это известная проблема Windows. В Firebird она решена с помощью нового параметра FileSystemCacheSize в файле firebird.conf, позволяющего контролировать объем оперативной памяти, используемый для кеша файловой системы.

CORE-2602 А. дос Сантос Фернандес

Ошибки при работе с таблицами мониторинга (MON$) в соединениях с кодировкой NONE.

CORE-2591 Д. Еманов

Падение производительности при высокой нагрузке из-за высокого коэффициента ожидания мьютекса.

CORE-2581 и CORE-2582 К. Вальдеррама

Некорректная обработка выражений, возвращающих бесконечность или NAN.

CORE-2578 А. дос Сантос Фернандес, А. Пешков

Ошибка преобразования при выборке RDB$DBKEY из представления (view), основанного на соединении нескольких таблиц .

CORE-2514 Д. Еманов

Улучшено сообщение об ошибке при нехватке места на диске для временных файлов.

CORE-2422 и CORE-2321 Д. Еманов

Сервер не переключался между различными каталогами, перечисленными в параметре TempDirectories. Вместо этого при нехватке места в первом каталоге из перечисленных сортировка (а также другие операции, требующие большое количество памяти) возвращалась ошибка 'Operating system directive write failed. Invalid argument.'

CORE-2315 Б. Оливер

Поддержка полного диапазона FLOAT-значений. Ранее при сохранении значений 3.4E38 и больше возвращались ошибки переполнения.

CORE-1991 Д. Сибиряков

Если UDF объявлена с параметром BLOB, в поле RDB$FUNCTION_PARAMETERS.RDB $FIELD_LENGTH записывалось значение NULL, что приводило к ошибке 'message length error'.

CORE-1781 A. дос Сантос Фернандес

Ошибка проверки целостности при использовании в подзапросе сортировки по агрегатной функции из внешнего запроса.

CORE-2564 A. Пешков

Ошибки при работе с таблицами мониторинга (MON$) на платформе RISC.

CORE-2550 A. Пешков

Ошибки при работе с DB_KEY в системах, использующих порядок байт big-Endian.

CORE-2538 A. дос Сантос Фернандес

Невозможность использования запроса в качестве входного параметра хранимой процедуры в PSQL-операторе EXECUTE PROCEDURE.

CORE-2532 B. Хорсун

Размер файла в многофайловых базах данных устанавливался некорректно.

CORE-2526 А. Пешков, Д. Еманов

Выключение сервера не учитывало соединения с сервисами.

CORE-2505 К. Вальдеррама

Встроенные тригонометрические функции могут вернуть NaN и бесконечность.

CORE-2501 К. Вальдеррама

Двоичные функции сдвига могут вернуть неправильный результат при сдвиге на отрицательное число бит.

CORE-2499 А. дос Сантос Фернандес

Ограничение на количество элементов функции DISTINCT не обрабатывалось, что могло привести к генерированию некорректного BLR-кода.

CORE-2482 А. Пешков

Получение данных из таблиц мониторинга (MON$) было нестабильным в моменты соединения и отсоединения от базы данных.

CORE-2475 В. Хорсун

Данные внешних (external) таблиц недоступны соединениям при архитектуре Classic Server (кроме первого доступа к таблице).

CORE-2449 Д. Еманов

Вместо реального исключения могла быть возвращена ошибка «lock conflict on no wait transaction».

CORE-2444 В. Хорсун

Если много соединений одновременно зарегистрировались на события (events) и свободное место в таблице событий исчерпано, возможен сбой сервера.

CORE-2426 А. дос Сантос Фернандес

При выполнении оператора ALTER TABLE не учитывались collation.

CORE-2416 В. Хорсун

Компилирование запроса с агрегированием результата производной таблицы (derived table) могло вызвать ошибку доступа (access violation).

CORE-2411 Д. Еманов

Для определенных типов запросов оптимизатор мог выбрать план, более медленный чем в версиях 2.0.5 и 2.1.2.

CORE-2397 В. Хорсун

Порча базы данных при удалении нескольких индексов в одной транзакции.

CORE-2359 А. дос Сантос Фернандес

Максимальная длина мультибайтовой строки не учитывалась при присвоении чисел.

CORE-2331 А. дос Сантос Фернандес

Оператор ALTER DOMAIN записывает в поле RDB$FIELD_SUB_TYPE неправильное значение.

CORE-2272 А. Пешков

Некорректная работа сервера после завершения (kill) процесса, работающего с событиями (events).

CORE-1971 Д. Еманов

Установлен и документирован однозначный порядок проверки условий (всегда слева направо) в предложении WHERE и в других предикатах.

CORE-1346 А. дос Сантос Фернандес

Ошибки в функциях LPAD() и RPAD() при их применении с несколькими различными полями в одном операторе.

CORE-2356 В. Хорсун

На платформе Windows прослушивающий процесс (listener) в архитектуре Classic Server мог не работать после перезапуска при наличии активных процессов (соединений).

CORE-2355 А. дос Сантос Фернандес

Неправильная работа функций LOWER/UPPER, если результирующая длина (в байтах) результирующей строки уменьшилась.

CORE-2351 А. Пешков

Невозможно создать базу данных с использованием алиаса вместо имени файла.

CORE-2349 В. Хорсун

Ошибка «Invalid SQLDA» в новом сетевом протоколе.

CORE-2348 В. Хорсун

Порча базы данных при превышении номера транзакции 32-битного целого.

CORE-2340 В. Хорсун

При высокой нагрузке могла возникать ошибка «Bugcheck 258 (page slot not empty)».

CORE-2320 В. Хорсун

Сложный рекурсивный запрос не всегда возвращал все записи результата.

CORE-2313 К. Вальдеррама

Функции INF_* могут вернуть неверный результат.

CORE-2311 В. Хорсун

Утечки памяти в запросах с WITH RECURSIVE.

CORE-2300 А. дос Сантос Фернандес

При втором вызове функции SUBSTRING() могла возникнуть ошибка «arithmetic exception, numeric overflow, or string truncation».

CORE-2289 В. Хорсун

При ошибке создания ограничения внешнего ключа (foreign key constraint) в сообщении может быть указано неверное название первичного ключа.

CORE-2242 А. Пешков

В блоке параметров BLOB (BPB) целые числа обрабатывались согласно порядку байт сервера, что вызывало ошибки на платформах с порядком байт Big Endian.

CORE-2241 В. Хорсун

Оператор ALTER TABLE ALTER COLUMN, выполняемый над таблицей при большом количестве вставок записей в нее, может привести к порче индекса.

CORE-2230 А. дос Сантос Фернандес

Входные параметры оператора EXECUTE BLOCK игнорируют ограничения домена.

CORE-2186 А. дос Сантос Фернандес

На платформе Windows библиотека Embedded-сервера fbintl.dll не выгружалась после вызова команды isc_dsql_execute_immediate() в ходе выполнения оператора CREATE DATABASE.

CORE-2182 Д. Еманов

Невозможно удалить из базы данных существующую UDF-функцию при наличии одноименной встроенной функции.

CORE-2154 В. Хорсун

При вызове функции **isc_dsql_sql_info()** с параметром **isc_info_sql_records** после извлечения (fetch) последней записи оператором EXECUTE PROCEDURE может произойти ошибка «request synchronization error».

CORE-2132 Д. Еманов

При сравнении поля с результатом вызова хранимой процедуры не всегда используется индекс.

CORE-2117 А. дос Сантос Фернандес

Функция ROW_COUNT может вернуть неправильный результат при индексном поиске и подзапросах.

CORE-2115 Д. Еманов

Переполнение буфера плана в запросах с длинным текстом.

CORE-2101 Д. Еманов

При попытке загрузки (fetch) записи после последней записи PSQL-курсора возвращается ошибка «Bugcheck 249 (pointer page vanished)».

CORE-2098 В. Хорсун

Невозможно создать представление (view), основанное на временной таблице.

CORE-2078 Д. Еманов

Неоптимальный план при соединении по неиндексируемому предикату.

CORE-2075 А. дос Сантос Фернандес

Инвертированный порядок байт RDB$DB_KEY при использовании представлений (view) во внешних соединениях (outer join).

CORE-2073 Д. Еманов

Некорректный результат при использовании индексов по boolean-выражению.

CORE-2069 А. дос Сантос Фернандес

Некорректный результат при выборке из представления (view), использующего RDB$DB_KEY.

CORE-2068 А. дос Сантос Фернандес

Неправильные результаты при использовании в предикате IN подзапроса, содержащего RDB$DB_KEY.

CORE-2067 А. дос Сантос Фернандес

Некорректная обработка RDB$DB_KEY в запросах с группировкой

CORE-2066 А. дос Сантос Фернандес

Преобразование SQL_TEXT / SQL_VARCHAR в SQL_TIMESTAMP / SQL_TIME / SQL_DATE

CORE-2053 Д. Еманов

Плохая оптимизация запросов INSERT .. RETURNING, содержащих вычисляемые выражения.

CORE-2045 dos Santos Fernandes

Ссылки на несуществующие поля системных таблиц в метаданных.

CORE-2044 А. дос Сантос Фернандес

Некорректный результат запроса UPDATE OR INSERT ... RETURNING OLD.FIELD, если поле FIELD объявлено как NOT NULL.

CORE-2041 А. дос Сантос Фернандес

При использовании в операторе UPDATE OR INSERT функции GEN_ID() значение генератора увеличивалось с трехкратным шагом.

CORE-2039 Д. Еманов

Некорректная обработка NULL в CHECK-ограничениях доменов.

CORE-2031 А. дос Сантос Фернандес

Оператор UPDATE может записать NULL в первую запись при использовании отбора по RDB$DB_KEY

CORE-2027 А. дос Сантос Фернандес

Неверное вычисление размера буфера при сортировке с участием системных полей.

CORE-2026 В. Хорсун

Проблемы с временными BLOB-ами в read-only базах данных.

CORE-2008 А. дос Сантос Фернандес

В системных таблицах не задан флаг NOT NULL для параметров хранимых процедур.

CORE-2002 К. Вальдеррама

Утечка памяти при ошибке преобразования результата UDF, объявленной с помощью FREE_IT.

CORE-2001 К. Вальдеррама

Иногда вместо ошибки преобразования типов могла возвращаться ошибка «arithmetic exception or string truncation».

CORE-2000 В. Хорсун

При высокой нагрузке менеджер блокировок мог ошибочно сообщить о deadlock-е.

CORE-1986 А. дос Сантос Фернандес

Удаление зависимостей при изменение имени домена.

CORE-1984 В. Хорсун

Менеджер блокировок мог ошибочно сообщить о deadlock-е, если один из ожидающих запросов ожидал окончания таймаута.

CORE-1980 В. Хорсун

Поток выполнения чистки (sweep) может привести к 100% загрузке CPU.

CORE-1970 В. Хорсун

Ошибки «Lock conversion denied (215)» в Classic Server.

CORE-1962 А. дос Сантос Фернандес

Неверный результат функции **EXTRACT (MILLISECONDS FROM ...)**.

CORE-1958 Д. Еманов

При многократном изменении одной записи в одной и той же транзакции могла возникать ошибка «Bugcheck 179 (decompression overran buffer)».

CORE-1957 А. Пешков

Длинные списки контроля доступа (ACL) обрезаются, что приводит к удалению привилегий доступа.

CORE-1943 А. дос Сантос Фернандес

При агрегировании по случайному выражению (например, с участием функции RAND) оператор возвращает бесконечное количество записей.

CORE-1938 Д. Еманов

Ошибка «Bugcheck 243 (missing pointer page)» при компилировании или выполнении запросов к таблицам, удаленным или пересозданным в другом соединении.

CORE-1936 А. дос Сантос Фернандес

Встроенная функция **LOG(base, number)** не проверяет входные параметры на корректность и вместо ошибки при некорректных значениях параметров возвращает NAN.

CORE-1914 А. Пешков

При ошибке во время создания таблицы база данных может остаться в некорректном состоянии.

CORE-1812 Д. Еманов

В базах данных с первым диалектом (dialect 1) в некоторых предикатах с типами date/time не использовались индексы.

CORE-1650 А. дос Сантос Фернандес

Запросы с группировкой по GEN_ID возвращают бесконечное количество записей.

CORE-1607 Д. Еманов

Плохая оптимизация коррелированных подзапросов, содержащих UNION.

CORE-1606 А. Потапченко, В. Хорсун

Добавлена возможность вставки записи в detail-таблицу, даже если соответствующая ей запись master-таблицs заблокирована, но внешний ключ (foreign key) остался неизменным.

CORE-1575 Д. Еманов

Порча памяти при многократных изменениях одной и той же таблицы в одной транзакции.

CORE-1544 Д. Еманов

Если прикладная программа создает «временную» хранимую процедуру, внутренний генератор для поля RDB$PROCEDURES.RDB$PROCEDURE_ID мог переполниться (32K - SMALLINT со знаком), что вызывало ошибку «numeric overflow» при создании новой хранимой процедуры.

Решение проблемы заключается в повторном использовании пропущенных значений. Такое же усовершенствование выполнено для таблиц генераторов (RDB$GENERATORS) и пользовательских исключений (RDB$EXCEPTIONS).

CORE-1343 А. дос Сантос Фернандес

Не допускалось использование подзапроса в качестве выражения функции CASE.

CORE-1246 А. дос Сантос Фернандес

Неверный результат при внешнем соединении (outer join) с производными таблицами (derived table).

CORE-1245 А. дос Сантос Фернандес

Неверный результат при внешнем соединении (outer join) с представлениями (view).

CORE-903 Д. Еманов

Обработка множественного присвоения значения одному и тому же полю в предложении SET оператора UPDATE не соответствует стандарту SQL.

CORE-501 А. дос Сантос Фернандес

Проблемы оптимизации при использовании подзапросов в функции COALESCE.

CORE-216 А. Пешков

Удаление привилегий доступа из-за переполнения буфера при большом количестве привилегий у одного и того же объекта базы данных.

CORE-1421 А. Пешков

SuperServer не останавливался (shutdown) немедленно, если команда на остановку следовала после неудачной попытки соединения.

CORE-1907 А. дос Сантос Фернандес

Удаление и добавление ограничения (check) в домен в одной и той же транзакции оставляет некорректные зависимости между объектами.

CORE-1905 К. Вальдеррама

Неправильная обработка знака решетки (#) в именах файлов в aliases.conf.

CORE-1887 А. Пешков

Новые создаваемые базы данных имеют неправильные права доступа.

CORE-1869 А. Пешков

Механизм предоставления и удаления ролей различается между версиями 2.0 и 2.1.

CORE-1841 В. Хорсун

Переполнение поля RDB$VIEW_RELATIONS.RDB$CONTEXT_NAME при использовании в представлении (view) производных таблиц (derived table) с длинными именами/псевдонимами.

CORE-1840 Д. Еманов

Утечка памяти при выполнении DDL-запросов.

CORE-1838 В. Хорсун

Оператор SET STATISTICS INDEX мог ошибочно изменить номер индекса при работе с индексами временных таблиц (GTT).

CORE-1830 В. Хорсун

Порча индекса при многократном изменении одной и той же записи в одной транзакции с использованием точек сохранения (savepoint).

CORE-1817 В. Хорсун

Параметр RelaxedAliasChecking не применялся к полю RDB$DB_KEY.

CORE-1811 Д. Еманов

Неправильная обработка использования ключевого слова «VALUE» без кавычек.

CORE-1798 А. дос Сантос Фернандес

Оператор INSERT ... RETURNING возвращал NULL в поле RDB$DB_KEY.

CORE-1797 А. дос Сантос Фернандес

Некорректные результаты поля OLD/NEW.RDB$DB_KEY в триггерах.

CORE-1784 А. дос Сантос Фернандес

Ошибка вызова EXECUTE PROCEDURE оператором EXECUTE STATEMENT.

CORE-1777 К. Вальдеррама

В блоке параметров транзакции (TPB) допускались конфликтующие параметры резервирования таблиц.

CORE-1775 В. Хорсун

Ухудшение производительности при проверке привилегий доступа.

CORE-1770 А. Пешков

Ошибка «Bugcheck 291» в DDL-запросах.

CORE-1735 А. дос Сантос Фернандес

Ошибка при выполнении операции SET DEFAULT при наличии ограничений целостности CHECK.

CORE-1731 В. Хорсун

Иногда сервер зависает на несколько минут со 100%-ной загрузкой процессора без какой-либо активности ввода/вывода.

CORE-1730 Д. Еманов

Ошибка при соединении с базой данных, если один из каталогов, перечисленных в параметре TempDirectories конфигурационного файла, не существует.

CORE-1724 В. Хорсун

CTE не могли использоваться в вычисляемых полях и в предикатах IN / ANY / ALL.

CORE-1694 А. дос Сантос Фернандес

Ошибка при создании/изменении триггера уровня базы данных, если он содержит комментарии на кириллице.

CORE-1693 А. дос Сантос Фернандес, Д. Еманов

Ошибка при использовании оператора EXECUTE STATEMENT в триггерах уровня базы данных.

CORE-1689 К. Вальдеррама

Сообщение об ошибке «There are <n> dependencies» сообщает неверное количество зависимых объектов.

CORE-1357 Д. Еманов

Перестал работать механизм DummyPacketInterval.

CORE-1307 А. Пешков

Неправильная обработка параметра –s файла fb_inet_server

CORE-479 А. Пешков

Перезапись существующих привилегий при длинных именах объектов.

Падения сервера/клиента

CORE-3011 Д. Еманов

Зависание или сбой сервера при мониторинге циклически подключающихся и отключающихся соединений.

CORE-2908 В. Хорсун

Сбой сервера или ошибки при работе с базами данных с ODS 8.x.

CORE-2888 А. дос Сантос Фернандес

Порча памяти, приводящая к неверному результату запроса или сбою сервера.

CORE-2576 А. дос Сантос Фернандес

Сбой сервера при парсинге некорректного или неполного BLR.

CORE-2455 А. Пешков

Сбой сервера при выполнении команды DROP DATABASE сразу после ошибки в статистической функции.

CORE-2441 А. Пешков

Сбой сервера при выполнении оператора UPDATE OR INSERT.

CORE-2306 А. Пешков

Немедленное завершение работы Superserver при возникновении ошибки во время запуска потока (thread).

CORE-2368 В. Хорсун

Ошибка при вызове isc_cancel_events(), если событие не найдено.

(CORE-2222) В. Хорсун

Ошибка при сохранении текстового BLOB-а с использованием фильтра преобразования.

(CORE-2137) Д. Еманов

Сбой сервера при восстановлении (restore) базы данных, если параметр «DummyPacketInterval» задан явным образом.

CORE-2121 А. Пешков

Сбой в клиентской библиотеке при работе с BLOB.
с

CORE-1983 А. Пешков

Ошибка Access Violation при нехватке памяти в операционной системе.

CORE-1965 Д. Еманов

Сбой в менеджере блокировок при высокой нагрузке DDL-запросами.

CORE-1894 А. дос Сантос Фернандес

Сбой сервера при циклических зависимостях между вычисляемыми полями.

CORE-1963 Д. Еманов

Сбой сервера при одновременном предоставлении/удалении привилегий в параллельных соединениях.

CORE-1506 А. Пешков

Сбой сервера при передаче в функцию isc_dsql_execute_immediate() пустой строки.

CORE-210 Д. Еманов

Сбой сервера Classic Server при редактировании одной и той же хранимой процедуры в параллельных соединениях.

CORE-1930 В. Хорсун

Сбой сервера, если у хранимой процедуры удалены выходные параметры, а зависимые от нее хранимые процедуры не перекомпилированы.

CORE-1919 А. дос Сантос Фернандес

Вызов оператора EXECUTE STATEMENT может привести к порче памяти и сбою сервера.

CORE-1884 В. Хорсун

Сбой сервера при использовании выражений в качестве значений по умолчанию для входных параметров хранимых процедур.

CORE-1839 А. Пешков

Сбой сервера при сортировке по полю, вычисляемому с использованием рекурсивного CTE.

CORE-1793 В. Хорсун

Сбой сервера при компилировании запроса с незадействованным параметризованным CTE.

CORE-1512 Д. Еманов

Сбой сервера из-за неправильного парсинга предложения DEFAULT.

Мониторинг/администрирование базы данных

CORE-2209 Д. Еманов

Запросы мониторинга при высокой нагрузке могут привести к падению производительности и даже блокировке остальной активности.

CORE-2171 Д. Еманов

Неправильные значения в поле MON$CALL_STACK.MON$CALLER_ID.

CORE-2017 Д. Еманов

В таблицах мониторинга не отображается статистика ввода/вывода для хранимых процедур.

CORE-1944 А. Пешков

На платформах Big Endian таблицы мониторинга возвращали неправильные данные.

CORE-1890 Д. Еманов

При высокой нагрузке в архитектуре SuperServer запрос мониторинга мог зависнуть.

CORE-1881 Д. Еманов

Запросы мониторинга могли привести к сбою сервера или нарушить работу механизма блокировки страниц.

CORE-1728 А. Пешков

Не работает мониторинг базы данных на платформе Linux из-за отсутствия прав доступа к файлам.

Язык Манипулирования Данными (DML)

CORE-1910 А. дос Сантос Фернандес

В предложении INSERT оператора MERGE допускались неправильные поля.

CORE-1859 А. дос Сантос Фернандес

Функция MAX могла вернуть ошибку переполнения или деления на ноль.

CORE-1828 А. дос Сантос Фернандес

Ошибки при использовании функции ABS в первом диалекте.

Утилиты командной строки

isql

CORE-2831 К. Вальдеррама

Имя пользователя и базы данных не должны быть в выходном скрипте.

CORE-2741 К. Вальдеррама

При извлечении метаданных неправильно интепретировались ограничения целостности CHECK, если ключевое слово «CHECK» написано в смешанном регистре символов.

CORE-915 А. дос Сантос Фернандес

На платформе Windows при извлечении метаданных утилитой ISQL дублировались переносы строк в теле PSQL-модулей.

CORE-2408 А. дос Сантос Фернандес

При извлечении метаданных для параметров хранимых процедур значения по умолчания записывались до флагов NOT NULL и COLLATE.

CORE-2407 А. дос Сантос Фернандес

При извлечении метаданных в операторе CREATE DATABASE не указывалось предложение PAGE_SIZE.

CORE-2370 К. Вальдеррама

План SQL-запроса более 2048 символов не отображался целиком.

CORE-2270 J. Swierczynski, А. Пешков

Расходование всей доступной памяти при запуске «isql» в консоли «zlogin».

CORE-1891 А. дос Сантос Фернандес

Команда SHOW VIEW не отображает информацию для вычисляемых полей.

CORE-1875 В. Хорсун

Ошибки в скриптах, содержащих переменную CURRENT_DATE.

CORE-1862 К. Вальдеррама

В версии 2.1 «isql» формирует некорректный скрипт с метаданными при наличии взаимозависимых селективных хранимых процедур.

CORE-1782 Д. Еманов

Сбой в «isql» при извлечении данных полей с псевдонимом длиннее 30 символов.

CORE-1749 Д. Еманов

Операторы DDL при включенном режиме AUTODDL ON не показывают статистику.

CORE-1507 К. Вальдеррама

В ISQL после команды INPUT сбивается счетчик строк в скриптах.

CORE-1363 К. Вальдеррама

Сбой в «isql» при преобразовании вещественного числа (double) в строку длинной более 23 bytes.

gsec

CORE-2928 А. Пешков

Переполнение буфера во внутренних структурах «gsec»

По неизвестным причинам утилита gsec при отображении копирует хеш пароля в локальную структуру данных. Начиная с версии 2.0 длина хеша была увеличена, и буфер для сохранения мог быть слишком маленьким. Эта ошибка не создает уязвимость и не может быть использована во вред, поскольку хеш никуда не передается и не отображается.

CORE-2528 К. Вальдеррама

Утилита gsec не возвращает коды ошибок операционной системе.

CORE-1680 А. Пешков

Команда DISPLAY отображает только несколько первых записей, если база данных пользователей содержит более 50 пользователей.

gbak

CORE-2914 Д. Еманов

Сбой сервера при восстановлении (restore) базы данных с индексом по выражению, ссылающемуся на несуществующую библиотеку UDF.

CORE-2793 Д. Еманов

Бинарное содержание файла резервной копии (backup) не совпадало после повторного цикла backup/restore одной и той же базы данных. Это наблюдается во всех версиях gbak, начиная с 1.5.4 и исправлено в версии 2.1.4 и 3.0-alpha.

CORE-2634 А. Пешков

Ухудшение производительности при восстановлении базы данных с большим объемом метаданных (большим количеством полей таблиц).

CORE-2291 В. Хорсун

При выполнении PSQL-модуля, содержащего singleton-запрос, возвращающий несколько записей, возвращалась ошибка «Bugcheck 284 (cannot restore singleton select data)» вместо ошибки с правильным кодом и текстом.

CORE-2285 А. Пешков

Порча базы данных с большим количеством привилегий после backup/restore.

CORE-2245 К. Вальдеррама

Ошибки при восстановлении базы данных с длинными сообщениями пользовательских исключений.

CORE-2223 А. Пешков

Многочисленные ошибки в gbak при работе со списками управления доступом (ACL).

CORE-2214 А. дос Сантос Фернандес

При restore не восстанавливаются привилегии доступа.

CORE-1911 К. Вальдеррама

Процессы backup и restore при работе через Services API не потокобезопасны.

CORE-1843 А. Пешков

При использованиии Service Manager утилита gbak не поддерживает пробелы в путях к файлам.

CORE-1703 Д. Еманов

Задержки и зависания при перенаправлении вывода gbak на вход другого процесса.

nBackup

CORE-2750 К. Вальдеррама

nBackup не работает после удаления дельта-файла.

CORE-2648 В. Хорсун

При записи в дельта-файл NBackup игнорируется параметр Forced Writes.

CORE-2266 В. Хорсун

nBackup не полностью блокирует файл базы данных от записи.

CORE-1696 Р. Симаков

Deadlock в менеджере блокировок при работе утилиты nBackup.

CORE-1876 Н. Самофатов

Механизм инкрементального копирования перестал работать в версии 2.1.

gfix

CORE-2846 Д. Еманов

Если вызов **gfix -shut <mode> -attach <timeout>** после истечения заданного таймаута завершился неудачей из-за наличия активных соединений, то последующие соединения с базой данных становятся невозможны.

CORE-97 Д. Еманов

Вызов **gfix -shut -force** не снимает блокировку с файла базы данных, предотвращая restore в существующий файл.

CORE-2268 В. Хорсун

Некорректные номера транзакций приводят к ошибкам BUGCHECK в утилите gfix.

CORE-2271 А. Пешков

При выполнении утилитой gfix операций проверки/починки большой базы данных привилегии пользователя проверялись с большой задержкой.

CORE-1961 Д. Еманов, R. Симаков

Ошибка «Bugcheck 210 (page in use during flush)» при выполнении операции проверки базы данных утилитой gfix.

gstat

CORE-2519 В. Хорсун

Утилита gstat вовзращала неправильную информацию о таблицах, содержащих более 2 миллиардов записей.

CORE-1412 К. Вальдеррама

Исправлены некоторые старые ошибки gstat при обработке параметров.

fb_lock_print

CORE-2598 Д. Еманов

Параметр **fb_lock_print -c[onsistency]** не работает на платформе Windows.

CORE-2354 А. Пешков

Вывод команды «fb_lock_print -ia» не записывался в файл между итерациями.

qli Query Utility for GDML

CORE-2247 А. Пешков

В утилите QLI неправильно выровнены буферы для сообщения и дескриптора.

Services Manager

CORE-1982 А. дос Сантос Фернандес

Одновременные операции резервного копирования или восстановления с помощью Services API могут мешать друг другу.

Сетевой интерфейс/API

CORE-2563 Д. Еманов

Существует возможность добиться отказа работы сетевого интерфейса Superserver посылкой сетевых пакетов специального формата, что приводило к невозможности установления новых соединений. Об уязвимости сообщили Core Security Technologies.

CORE-2437 А. Пешков

Переполнение буфера на клиенте при получении событий (event).

CORE-2307 Д. Еманов

Неправильные результаты информационных API-вызовов.

CORE-2234 В. Хорсун

На платформе Windows в архитектуре Classic Server после закрытия клиентского соединения соответствующий ему процесс мог быть не завершен из-за неправильной проверки на стороне сервера. Это же самое могло произойти при длительных блокировках (deadlock) при высокой нагрузке.

CORE-2151 В. Хорсун

Неправильная обработка пути к каталогу временных файлов, содержащего пробелы.

CORE-2033 А. Пешков

Не обнаруживается функция _Unwind_GetIP в клиентской библиотеке при ее статической линковке с библиотекой libstdc++.

CORE-2018 В. Хорсун

Невозможность подключения нескольких соединений к базе данных в режиме read-only.

CORE-1972 А. Пешков

Не привилегированный пользователь мог поменять у любой базы данных параметры «Forced Writes, Read Only, SQL Dialect» и др. из-за неправильной обработке DPB-параметров. Изменения могут отразиться на многих существующих приложениях, драйверах и инструментах. Исправления также внесены в версии 2.1.2 и 2.0.5.

CORE-1868 А. Пешков

Сбой в клиентской библиотеке при вызове isc_dsql_free_statement().

CORE-1763 В. Хорсун

Клиентская библиотека не устанавливала параметры соединения SO_KEEPALIVE и TCP_NODELAY.

CORE-1755 и CORE-1756 Д. Коваленко, А. Пешков

Ошибки при вызове isc_start_transaction с некорректными параметрами.

CORE-1726 А. Пешков

Ошибки в функции isc_service_start в архитектуре Super Server.

CORE-1079 А. Пешков

При каждом обращении к библиотеке fbclient/fbembed происходит утечка 64Kb памяти.

Безопасность

CORE-2657 А. Пешков

С помощью недокументированного SPB-тега можно получить привилегии SYSDBA с произвольным именем пользователя.

CORE-2087 А. Пешков

Если параметр «RemoteBindAddress» содержит имя компьютера вместо IP-адреса или содержит несуществующий IP-адрес, сервер будет неявно привязан ко всем сетевым интерфейсам без сообщений в файле firebird.log или системном логе. Теперь в таких случаях будет осуществляться привязка к локальному интерфейсу (127.0.0.1).

CORE-2055 А. Пешков

Переполнение буфера в клиентской библиотеке Firebird.

CORE-1845 А. Пешков

Обычный пользователь может узнать каталог установки сервера с помощью стандартных вызовов.

CORE-1810 А. Пешков

Неправильная обработка символа точки в именах пользователей.

Интернациональная языковая поддержка

CORE-2642 А. дос Сантос Фернандес

Проблемы с инициализацеий ICU в многопоточных приложениях на платформе Windows.

CORE-2607 А. дос Сантос Фернандес

Проблемы при использовании маркера набора символов в запросах к таблицам мониторинга в PSQL-модулях. Более подробная информация приведена в разделе Маркер набора символов.

CORE-2361 А. дос Сантос Фернандес

Обрезание строки при чтении данных из поля с набором символов 8859_1 Spanish с помощью функции isc_dsql_fetch() в соединении с набором символов UTF8.

CORE-2227 А. дос Сантос Фернандес

Проблемы при создании триггеров, ссылающихся на поля, имена которых содержат символы с диакритическим знаком (ударением).

CORE-2123 А. дос Сантос Фернандес, Д. Коваленко

Проблемы с получением данных с набором символов UNICODE_FSS в соединении с набором символов CP943C.

CORE-2122 А. дос Сантос Фернандес, Д. Коваленко

Проблемы при транслитерации больших текстовых BLOB между UNICODE_FSS / UTF8 и другими наборами символов.

CORE-2095 А. дос Сантос Фернандес, Д. Коваленко

Ошибки в функции CVJIS_eucj_to_unicode().

CORE-2019 А. дос Сантос Фернандес

Обрезание строки при преобразовании UTF-8-строк.

CORE-1989 А. дос Сантос Фернандес

Поле с набором символов UTF8 и collation UNICODE_CI не может быть использовано в качестве внешнего ключа (foreign key).

CORE-1927 А. дос Сантос Фернандес

Процедура **sp_register_character_set** может вернуть отрицательные значения для поля RDB $CHARACTER_SETS.RDB$CHARACTER_SET_ID.

CORE-1690 А. дос Сантос Фернандес

Ошибки при работе с таблицами, содержащими строки с набором символов UTF8.

CORE-1596 А. дос Сантос Фернандес

Ошибка в функции CsConvert::convert

CORE-1432 А. дос Сантос Фернандес

При изменении у поля атрибута collation он не меняется у старых форматов (у существующих записей).

CORE-316 А. дос Сантос Фернандес

Невозможно соединиться с базой данных с мультибайтовыми символами в имени файла.

CORE-1802 А. дос Сантос Фернандес

Неправильный подсчет максимального размера ключа для PXW_CSY.

CORE-1774 А. дос Сантос Фернандес

Ошибки при поиске по некоторым символам с collation ES_ES_CI_AI.

CORE-1254 А. дос Сантос Фернандес

Ошибки при выполнении DISTINCT с последовательностями сортировки (collation), нечувствительными к регистру/ударению.

Ошибки на платформе POSIX

CORE-3067 А. Пешков

На 64-битных платформах POSIX (кроме HP-UX) не удалялись временные файлы.

CORE-3019 А. Пешков

На последних версиях Gentoo Linux при запуске сервера с архитектурой SuperServer и SuperClassic при обращении к библиотеке «/etc/init.d/functions.sh» возникала ошибка «ERROR: firebird does not have a start function».

CORE-3001 А. Пешков

Ошибка при создании пользователя и группы «firebird» во время инсталляции.

CORE-2919 А. Пешков

Скрипт инсталляции на платформе Linux не учитывал нестандартные дистрибутивы.

CORE-2845 П. Бич

При компиляции на платформе Solaris возвращается ошибка «need to use SFIO», хотя начиная с версии Solaris 10 SFIO не требуется. Для решения проблемы в файл `common.h` добавлена проверка и предупреждение о необходимости поддержки SFIO.

CORE-2814 В. Хорсун

Сбой сервера при вызове функции `map_sort_data` на платформе SPARC.

CORE-2601 А. Пешков

Игнорировались многие стандартные ключи утилиты **configure**, предназначенные для настройки каталогов установки на платформах POSIX. Почти невозможно заставить работать стандартные ключи GNU без изменения их значений по умолчанию. Вместо этого были добавлены новые ключи, позволяющие задать расположение различных файлов Firebird. Ключи перечислены в разделе Ключи для утилиты configure.

CORE-2572 А. Пешков

Некорректная обработка блокировок LCK_page_space на платформах big-endian.

CORE-2221 П. Бич, А. Пешков

На платформах POSIX после изменения прав для файла security2.fdb с 0660 на 0666 становились невозможны любые соединения к любым базам данных.

CORE-2093 А. Пешков

Ошибка при запуске SuperServer на Solaris 64-bit.

CORE-1909 А. Пешков

Неправильная информация об ошибках в файле firebird.log в linux/amd64

CORE-1885 А. дос Сантос Фернандес, А. Пешков

Обрыв соединения при выполнении CREATE COLLATION на платформе Posix.

CORE-1854 А. Пешков

При использовании Trusted Authentication на платформе Unix значение переменной CURRENT_USER могло возвращаться не в верхнем регистре (как должно).

CORE-1826 А. Пешков

Скрипты changeRunUser.sh и restoreRootRunUser.sh не меняли пользователя в скриптах init.d.

CORE-1818 А. Пешков

На платформе Posix не удалялись временные файлы для временных таблиц.

CORE-1807 А. Пешков

После аварийного завершения сервер мог быть привязан к неправильному TCP-порту.

CORE-1766 А. Пешков

На платформе Linux инсталлятор Classic Server указывал у файла `isc_monitor1` некорректных владельца и группу.

CORE-1671 А. Пешков

Ошибки при вызове функции atexit() в клиентских библиотеках, загруженных с помощью dlopen.

Ошибки на платформе Windows

CORE-2769 Д. Еманов

В высоконагруженных системах на платформе Windows может возникнуть ошибка при соединении по локальному протоколу (XNET) из-за таймаута при ожидании установки события `xnet_response_event`. Для решения этой проблемы теперь можно использовать параметр ConnectionTimeout в файле конфигурации `firebird.conf`.

CORE-2108 В. Хорсун

При использовании локального протокола (XNET) в условиях высокой нагрузки Windows неверно вычисляется номер свободного блока памяти, что могло привести к ошибке при установлении соединения.

CORE-2107 В. Хорсун

Ошибка при соединении по протоколу TCP\IP к Classic Server на платформе Windows в высоконагруженных системах.

CORE-1923 Д. Еманов

При успешном выполнении команда **instsvc.exe remove** возвращала 1 вместо 0.

CORE-1820 П. Ривз, Д. Еманов

Инсталлятор не обнаруживал запущенную копию сервера.

CORE-1105, CORE-1390, CORE-1566, CORE-1639 Д. Еманов

Псевдонимы баз данных некорректно работают при использовании протокола XNET.

Ошибки на платформе MacOSX

CORE-2065 П. Бич

Инсталлятор для платформы MacOSX не включал путь к клиентской библиотеке в список каталогов для поиска библиотек.

Другие исправленные ошибки

CORE-2282 К. Вальдеррама

Неправильная обработка чисел меньше -1 в UDF-функциях округления.

CORE-2281 К. Вальдеррама

Неправильная обработка отрицательных чисел в UDF-функциях округления.

Глава 18

Команда разработчиков проекта Firebird 2.5

Таблица 18.1. Команда разработчиков Firebird

Разработчик	Страна	Основные задачи
Дмитрий Еманов (Dmitry Yemanov)	Российская Федерация	Штатный разработчик, лидер команды разработчиков ядра сервера
Александр Пешков (Alex Peshkov)	Российская Федерация	Штатный разработчик, координатор направления безопасности; сборка релиз-пакетов; портирование сервера на различные платформы
Клаудио Вальдерама (Claudio Valderrama)	Чили	Проверка кода; поиск и исправление ошибок; развитие утилиты ISQL; разработка и поддержка UDF-функций
Владислав Хорсун (Vladyslav Khorsun)	Украина	Штатный разработчик, проектирование и разработка SQL-расширений
Арно Бринкман (Arno Brinkman)	Нидерланды	Усовершенствование оптимизатора и индексов; развитие возможностей DSQL
Адриано дос Сантос Фернандес (Adriano dos Santos Fernandes)	Бразилия	Поддержка и внедрение новых наборов символов; расширение обработки текстовых и BLOB типов; развитие возможностей DSQL; проверка кода
Николай Самофатов (Nickolay Samofatov)	Российская Федерация	Разработка ядра сервера, nBackup
Роман Симаков (Roman Simakov)	Российская Федерация	Разработка ядра сервера, nBackup
Билл Оливер (Bill Oliver)	США	Развитие ветки Vulcan, разработка ядра сервера
Пол Бич (Paul Beach)	Франция	Менеджер релизов; сборка пакетов для платформ HP-UX, MacOS и Solaris
Павел Цизар (Pavel Cisar)	Чехия	Проектирование и разработка инструментов QA, драйвера для Python
Филипп Маковски (Philippe Makowski)	Франция	Тестирование
Пол Ривз (Paul Reeves)	Франция	Сборка пакетов и разработка инсталлятора для платформы Win32
Роман Рокицкий (Roman Rokytskyy)	Германия	Разработчик и координатор Jaybird

Разработчик	Страна	Основные задачи
Евгений Путилин (Evgeny Putilin)	Российская Федерация	Разработка поддержки хранимых процедур на языке Java
Иржи Чинчура (Jiri Cincura)	Чехия	Разработчик и координатор .NET-провайдера
Владимир Цвигун (Vladimir Tsvigun)	Украина	Разработчик и координатор драйвера ODBC/JDBC
Стефен Бойд (Stephen Boyd)	Канада	Разработка GPRE
Пол Винкенуг (Paul Vinkenoog)	Нидерланды	Координатор и разработчик документации; проектировщик и разработчик инструментов документирования
Норман Данбар (Norman Dunbar)	Великобритания	Разработчик документации
Павел Меньшиков (Pavel Menshchikov)	Российская Федерация	Переводчик документации
Александр Карпейкин (Alex Karpeykin)	Российская Федерация	Переводчик документации
Томнеко Хаяши (Tomneko Hayashi)	Япония	Переводчик документации
Умберто Мазотти (Umberto Masotti)	Италия	Переводчик документации
Хелен Борри (Helen Borrie)	Австралия	Редактор и составитель Release Notes
а также		
Université du Littoral Côte d'Opale Masters students	Франция	Тестирование и разработка тестов

Приложение A: SQLSTATE

Коды и сообщения SQLSTATE

Ниже перечислены все поддерживаемые коды и сообщения SQLSTATE:

1. 5-символьный код SQLSTATE состоит из SQL CLASS (2 символа) и SQL SUBCLASS (3 символа).
2. Там где возможно, установлено однозначное соответствие с устаревшими кодами SQLCODE.
3. Во многих случаях, коды SQLCODE и SQLSTATE не имеют однозначного соответствия - но это не является ошибкой, а соответствует SQL-стандарту и направлено на упразднение устаревших кодов SQLCODE.

Код SQLSTATE	Сообщение	Соответствующий SQLCODE
SQLCLASS 00 (Success)		
00000	Success	
SQLCLASS 01 (Warning)		
01000	General Warning	
01001	Cursor operation conflict	
01002	Disconnect error	
01003	NULL value eliminated in set function	
01004	String data, right-truncated	
01005	Insufficient item descriptor areas	
01006	Privilege not revoked	
01007	Privilege not granted	
01008	Implicit zero-bit padding	
01100	Statement reset to unprepared	
01101	Ongoing transaction has been committed	
01102	Ongoing transaction has been rolled back	
SQLCLASS 02 (No Data)		
02000	No data found or no rows affected	
SQLCLASS 07 (Dynamic SQL error)		

Код SQLSTATE	Сообщение	Соответствующий SQLCODE
07000	Dynamic SQL error	
07001	Wrong number of input parameters	
07002	Wrong number of output parameters	
07003	Cursor specification cannot be executed	
07004	USING clause required for dynamic parameters	
07005	Prepared statement not a cursor-specification	
07006	Restricted data type attribute violation	
07007	USING clause required for result fields	
07008	Invalid descriptor count	
07009	Invalid descriptor index	
SQLCLASS 08 (Connection Exception)		
08001	Client unable to establish connection	
08002	Connection name in use	
08003	Connection does not exist	
08004	Server rejected the connection	
08006	Connection failure	
08007	Transaction resolution unknown	
SQLCLASS 0A (Feature Not Supported)		
0A000	Feature Not Supported	
SQLCLASS 0B (Invalid Transaction Initiation)		
0B000	Invalid transaction initiation	
SQLCLASS 0L (Invalid Grantor)		
0L000	Invalid grantor	
SQLCLASS 0P (Invalid Role Specification)		
0P000	Invalid role specification	
SQLCLASS 0U (Attempt to Assign to Non-Updatable Column)		
0U000	Attempt to assign to non-updatable column	
SQLCLASS 0V (Attempt to Assign to Ordering Column)		
0V000	Attempt to assign to Ordering column	
SQLCLASS 20 (Case Not Found For Case Statement)		

Код SQLSTATE	Сообщение	Соответствующий SQLCODE
20000	Case not found for case statement	
SQLCLASS 21 (Cardinality Violation)		
21000	Cardinality violation	
21S01	Insert value list does not match column list	
21S02	Degree of derived table does not match column list	
SQLCLASS 22 (Data Exception)		
22000	Data exception	
22001	String data, right truncation	
22002	Null value, no indicator parameter	
22003	Numeric value out of range	
22004	Null value not allowed	
2205	Error in assignment	
2206	Null value in field reference	
2207	Invalid datetime format	
22008	Datetime field overflow	
22009	Invalid time zone displacement value	
2200A	Null value in reference target	
2200B	Escape character conflict	
2200C	Invalid use of escape character	
2200D	Invalid escape octet	
2200E	Null value in array target	
2200F	Zero-length character string	
2200G	Most specific type mismatch	
22010	Invalid indicator parameter value	
22011	Substring error	
22012	Division by zero	
22014	Invalid update value	
22015	Interval field overflow	
22018	Invalid character value for cast	
22019	Invalid escape character	

Код SQLSTATE	Сообщение	Соответствующий SQLCODE
2201B	Invalid regular expression	
2201C	Null row not permitted in table	
22020	Invalid limit value	
22021	Character not in repertoire	
22022	Indicator overflow	
22023	Invalid parameter value	
22024	Character string not properly terminated	
22025	Invalid escape sequence	
22026	String data, length mismatch	
22027	Trim error	
22028	Row already exists	
2202D	Null instance used in mutator function	
2202E	Array element error	
2202F	Array data, right truncation	
SQLCLASS 23 (Integrity Constraint Violation)		
23000	Integrity constraint violation	
SQLCLASS 24 (Invalid Cursor State)		
24000	Invalid cursor state	
24504	The cursor identified in the UPDATE, DELETE, SET, or GET statement is not positioned on a row	
SQLCLASS 25 (Invalid Transaction State)		
25000	Invalid transaction state	
25	xxxx	
25S01	Transaction state	
25S02	Transaction is still active	
25S03	Transaction is rolled back	
SQLCLASS 26 (Invalid SQL Statement Name)		
26000	Invalid SQL statement name	
SQLCLASS 27 (Triggered Data Change Violation)		
27000	Triggered data change violation	

Код SQLSTATE	Сообщение	Соответствующий SQLCODE
SQLCLASS 28 (Invalid Authorization Specification)		
28000	Invalid authorization specification	
SQLCLASS 2B (Dependent Privilege Descriptors Still Exist)		
2B000	Dependent privilege descriptors still exist	
SQLCLASS 2C (Invalid Character Set Name)		
2C000	Invalid character set name	
SQLCLASS 2D (Invalid Transaction Termination)		
2D000	Invalid transaction termination	
SQLCLASS 2E (Invalid Connection Name)		
2E000	Invalid connection name	
SQLCLASS 2F (SQL Routine Exception)		
2F000	SQL routine exception	
2F002	Modifying SQL-data not permitted	
2F003	Prohibited SQL-statement attempted	
2F004	Reading SQL-data not permitted	
2F005	Function executed no return statement	
SQLCLASS 33 (Invalid SQL Descriptor Name)		
33000	Invalid SQL descriptor name	
SQLCLASS 34 (Invalid Cursor Name)		
34000	Invalid cursor name	
SQLCLASS 35 (Invalid Condition Number)		
35000	Invalid condition number	
SQLCLASS 36 (Cursor Sensitivity Exception)		
36001	Request rejected	
36002	Request failed	
SQLCLASS 37 (Invalid Identifier)		
37000	Invalid identifier	
37001	Identifier too long	
SQLCLASS 38 (External Routine Exception)		
38000	External routine exception	

Код SQLSTATE	Сообщение	Соответствующий SQLCODE
SQLCLASS 39 (External Routine Invocation Exception)		
39000	External routine invocation exception	
SQLCLASS 3B (Invalid Save Point)		
3B000	Invalid save point	
SQLCLASS 3C (Ambiguous Cursor Name)		
3C000	Ambiguous cursor name	
SQLCLASS 3D (Invalid Catalog Name)		
3D000	Invalid catalog name	
3D001	Catalog name not found	
SQLCLASS 3F (Invalid Schema Name)		
3F000	Invalid schema name	
SQLCLASS 40 (Transaction Rollback)		
40000	Ongoing transaction has been rolled back	
40001	Serialization failure	
40002	Transaction integrity constraint violation	
40003	Statement completion unknown	
SQLCLASS 42 (Syntax Error or Access Violation)		
42000	Syntax error or access violation	
42702	Ambiguous column reference	
42725	Ambiguous function reference	
42818	The operands of an operator or function are not compatible	
42S01	Base table or view already exists	
42S02	Base table or view not found	
42S11	Index already exists	
42S12	Index not found	
42S21	Column already exists	
42S22	Column not found	
SQLCLASS 44 (With Check Option Violation)		
44000	WITH CHECK OPTION Violation	
SQLCLASS 45 (Unhandled User-defined Exception)		

Код SQLSTATE	Сообщение	Соответствующий SQLCODE
45000	Unhandled user-defined exception	
SQLCLASS 54 (Program Limit Exceeded)		
54000	Program limit exceeded	
54001	Statement too complex	
54011	Too many columns	
54023	Too many arguments	
SQLCLASS HY (CLI-specific Condition)		
HY000	CLI-specific condition	
HY001	Memory allocation error	
HY003	Invalid data type in application descriptor	
HY004	Invalid data type	
HY007	Associated statement is not prepared	
HY008	Operation canceled	
HY009	Invalid use of null pointer	
HY010	Function sequence error	
HY011	Attribute cannot be set now	
HY012	Invalid transaction operation code	
HY013	Memory management error	
HY014	Limit on the number of handles exceeded	
HY015	No cursor name available	
HY016	Cannot modify an implementation row descriptor	
HY017	Invalid use of an automatically allocated descriptor handle	
HY018	Server declined the cancellation request	
HY019	Non-string data cannot be sent in pieces	
HY020	Attempt to concatenate a null value	
HY021	Inconsistent descriptor information	
HY024	Invalid attribute value	
HY055	Non-string data cannot be used with string routine	
HY090	Invalid string length or buffer length	
HY091	Invalid descriptor field identifier	

Код SQLSTATE	Сообщение	Соответствующий SQLCODE
HY092	Invalid attribute identifier	
HY095	Invalid FunctionId specified	
HY096	Invalid information type	
HY097	Column type out of range	
HY098	Scope out of range	
HY099	Nullable type out of range	
HY100	Uniqueness option type out of range	
HY101	Accuracy option type out of range	
HY103	Invalid retrieval code	
HY104	Invalid LengthPrecision value	
HY105	Invalid parameter type	
HY106	Invalid fetch orientation	
HY107	Row value out of range	
HY109	Invalid cursor position	
HY110	Invalid driver completion	
HY111	Invalid bookmark value	
HYC00	Optional feature not implemented	
HYT00	Timeout expired	
HYT01	Connection timeout expired	
SQLCLASS XX (Internal Error)		
XX000	Internal error	
XX001	Data corrupted	
XX002	Index corrupted	

Приложение B: Licence Notice

Содержание данной Документации распространяется на условиях лицензии «Public Documentation License Version 1.0» (далее «Лицензия»); Вы можете использовать эту Документацию, только если согласны с условиями Лицензии. Копии текста Лицензии доступны по адресам http://www.firebirdsql.org/pdfmanual/pdl.pdf (PDF) и http://www.firebirdsql.org/manual/pdl.html (HTML).

Оригинальная Документация называется *Firebird 2.5 Release Notes*.

Первоначальный Автор Оригинальной Документации: Хелен Борри.

Copyright (C) 2004–2009. Все права защищены. Адрес электронной почты для контакта: helebor at users dot sourceforge dot net.

Перевод на русский язык: Александр Карпейкин. Редактор: GR (C) 2008-2010. Все права защищены. Адрес электронной почты для контакта: firebird_sql at mail dot ru.

Примечание переводчика: далее представлен оригинальный текст раздела, так как его перевод не имеет равноценной юридической силы.

The contents of this Documentation are subject to the Public Documentation License Version 1.0 (the «License»); you may only use this Documentation if you comply with the terms of this Licence. Copies of the Licence are available at http://www.firebirdsql.org/pdfmanual/pdl.pdf (PDF) and http://www.firebirdsql.org/manual/pdl.html (HTML).

The Original Documentation is entitled *Firebird 2.5 Release Notes*.

The Initial Writer of the Original Documentation is: Helen Borrie. Persons named in attributions are Contributors.

Copyright (C) 2004-2009. All Rights Reserved. Initial Writer contact: helebor at users dot sourceforge dot net.