



Замечания к выпуску Firebird 3.0.3

2 февраля 2018 - Документ в. 0303-03 — для выпуска Firebird 3.0.3

Helen Borrie (Составитель/Редактор)

Замечания к выпуску Firebird 3.0.4

2 февраля 2018 - Документ в. 0303-03 - для выпуска Firebird 3.0.3

Helen Borrie (Составитель/Редактор)

Перевод на русский язык - Александр Карпейкин — 8 февраля 2018 г.

Содержание

1. Общие сведения	1
Выпуск Firebird 3.0.3	1
Улучшения в выпуске Firebird 3.0.3	1
Выпуск Firebird 3.0.2	5
Что нового	5
Улучшения	5
Выпуск Firebird 3.0.1	7
Отчёты об ошибках	9
Документация	9
2. Новое в Firebird 3.0.2	10
Новые возможности	10
Совместимость со старыми версиями	14
3. Изменения в ядре Firebird	15
Новая архитектура	15
Режимы сервера	16
Провайдеры	17
Строка подключения и протоколы	22
Внешние средства ядра сервера	34
Улучшения оптимизатора	38
Другие оптимизации	38
Удалённый интерфейс/Сетевой протокол	39
Другие улучшения	39
Увеличен лимит подключений	39
Улучшенная диагностика ошибок	40
Обновлена версия ICU	40
Улучшена внутренняя отладочная информация	40
Заменены некорректные сообщения	40
Новая псевдоколонка RDB\$RECORD_VERSION	40
Системная инициализация скриптов	41
4. Изменения в Firebird API и ODS	42
Изменения ODS (структуры данных на диске)	42
Новый номер ODS	42
Идентификатор реализации ID устарел	42
Максимальный размер базы данных	43
Максимальный размер страницы	43
Максимальное количество буферных страниц в кэше	43
Увеличение лимита для количества Transaction ID	43
Увеличены лимиты для подключений и Statement ID	44
Системные таблицы	44
Расширения API (Application Programming Interface)	47
Интерфейсы и новый объектно-ориентированный API	48
Другие новшества в API	52

Улучшения в API	52
5. Зарезервированные слова и изменения	63
Новые ключевые слова в Firebird 3.0	63
6. Изменения и дополнения в параметрах конфигурации	65
Область действия параметров	65
Замена параметров макросами	65
Подключение файлов конфигурации	66
Подстановочные знаки	67
Выражения для значений параметров	68
Конфигурация “Для каждой базы данных” (“Per-database”)	68
Формат записей конфигурации	68
Доступные параметры	69
Новые параметры	70
База данных безопасности - SecurityDatabase	70
AuthServer и AuthClient	70
WireCrypt	71
UserManager	71
TracePlugin	71
WireCryptPlugin	72
KeyHolderPlugin	72
Providers	72
ServerMode	72
RemoteAccess	72
WireCompression	73
IPv6V6Only	73
Изменённые и улучшенные параметры	75
ExternalFileAccess	75
Удалённые или устаревшие параметры	76
RootDirectory	76
LegacyHash	76
OldSetClauseSemantics	76
OldColumnNaming	76
LockGrantOrder	76
Устаревшие настройки приоритетов Windows	77
7. Безопасность	78
Расположение списка пользователей	78
Создание альтернативной базы данных безопасности	79
Шифрование базы данных	79
Ключ шифрования	80
Задачи шифрования	80
Новый метод аутентификации в Firebird 3.0	81
Поддержка SSL/TLS	82
Увеличенная длина пароля	83
Плагин для аутентификации	83
Менеджеры пользователей	84

Шифрование сетевого подключения	85
Ключ шифрования сессии	85
Доступ пользователей к объектам	86
Правила доступа	87
Синтаксис оператора MAPPING для объектов	88
Устаревшие правила доступа	91
Международные наборы символов для учетных записей	
пользователей	92
Условия для подключения Non-ASCII логинов	93
Возможности SQL для управления доступом	95
Управление пользователями с помощью SQL	95
SET ROLE	99
GRANT/REVOKE и GRANTED BY указанному пользователю	
.....	101
REVOKE ALL ON ALL	102
Права пользователей на изменение метаданных	102
Права GRANT EXECUTE для UDF	105
Улучшение для рекурсивных хранимых процедур	105
Права для защиты других объектов метаданных	105
Псевдо-таблицы со списком пользователей	106
Ограничения для DROP ROLE (3.0.1)	107
8. Язык определения данных - Data Definition Language (DDL)	108
Быстрые ссылки для DDL	108
Улучшения DDL	108
Новые типы данных	108
Управление данными NULL в доменах и столбцах	113
Изменения для генераторов (последовательностей)	114
Изменение набора символов по умолчанию	115
Выражения BLOB в вычисляемых столбцах	115
«Задержка» закрытия базы данных для Superserver	115
Возможность сохранения файла тени	117
Новое в SQL для управления пользователями и правами	
доступа	117
9. Язык манипулирования данными (DML)	120
Быстрые ссылки для DML	120
Дополнительные возможности SQL 2008 для MERGE	120
Оконные (аналитические) функции	122
Агрегатные функции, используемые в качестве оконных	
функций	123
Разделы в оконных функциях	124
Сортировка в оконных функциях	125
Эксклюзивные оконные функции	126
Вывод расширенного плана	129
Вывод расширенного плана в isql	129
Внутренние функции	129

SUBSTRING в регулярных выражениях	130
Обратные гиперболические тригонометрические функции	130
Статистические функции	131
Снято ограничение в 32 кБ для аргумента в функции TRIM()	
BLOB	136
Корректировка лимита строковых литералов	136
Улучшения функции DATEADD()	137
Улучшения в языке DML	137
Альтернатива кавычкам в строковых литералах	137
Операторы OFFSET и FETCH совместимы с SQL-2008	138
Запрет одновременного использования явных и неявных соединений	140
Поддержка параметров в левой части предложения WHERE	141
Улучшения для предложения RETURNING	141
Устойчивость курсора	142
Улучшение GTT	142
Улучшение для строк DML	143
Функция COUNT() возвращает тип BIGINT	143
Оптимизация	143
Небольшое улучшение для 1-го диалекта	145
Улучшения встроенного SQL (ESQL)	145
10. Процедурный SQL (PSQL)	147
Быстрые ссылки для PSQL	147
Сохранённые функции PSQL	147
Подпрограммы PSQL	148
Пакеты	150
Подписи (сигнатуры) пакетов	151
Синтаксис пакетов	152
Примеры пакетов	154
Триггер для DDL	156
Права для триггеров DDL	159
Поддержка триггеров DDL в утилитах	160
Пространство имен контекста DDL_TRIGGER	160
Поддержка курсора с возможностью прокрутки (двунаправленного)	166
Синтаксис курсора для PSQL	166
Исключения с параметрами	167
Использование CONTINUE в цикле	169
Стабилизация курсора в PSQL	169
Расширение использования префикса двоеточия	170
Переменная типа курсор в PSQL	170
Новое в использовании двоеточия для переменных	172
Использование SQLSTATE в обработчиках исключений	173

Ошибки времени выполнения EXECUTE BLOCK	174
Увеличен размера тела PSQL	174
11. Мониторинг и утилиты командной строки	175
Мониторинг	175
Трассировка	175
gbak	176
Новая опция бэкапа “Skip Data”	176
Длинные имена для лог файлов	176
Статистика в реальном времени с подробным выводом	176
Новый код возврата	179
gsec	179
isql	179
Расширение SET EXPLAIN для детального просмотра планов	179
Извлечение метаданных	180
Путь для INPUT файлов	180
Увеличение размера буфера команд	180
Детерминированная метка для сохраненных функций	180
fb_lock_print	181
Входные параметры	181
Улучшения для использования	181
gfix	181
Ключ -NoLinger	181
Улучшения для сообщений при проверке	182
Другие тонкие настройки	182
Все утилиты командной строки	182
Замена жёстко закодированных сообщений	182
Очистка синтаксиса произвольного ключа	183
12. Проблемы совместимости	184
Где искать инструменты?	184
Другие изменения для Windows	184
Файл aliases.conf больше не нужен	184
Встроенные соединения	185
Инициализация базы данных безопасности	185
Шаги инициализации	186
Дополнительные сведения о базах данных безопасности ..	187
Устаревшая аутентификация	187
Обновление базы данных безопасности версии 2.x	189
Шаги обновления базы данных	189
безопасности	189
Локальное соединение с SuperServer под Windows	190
Параметры конфигурации	191
Системные таблицы доступны теперь только для чтения	191
Изменения языка SQL	191
Поддержка смешанных соединений больше не	

поддерживается	192
Длина для локальных псевдонимов и имен переменных	
ограничена	192
Изменения для имён пользователей	192
Столбцы вывода курсора должны иметь имя	192
Эффекты «стабильного курсора»	193
Зарезервированные слова	193
Синтаксис конфигурации трассировки	194
Неожиданные ошибки GDSCODE	194
Распознавание удалённого адреса клиента	195
13. Исправление ошибок	196
Выпуск Firebird 3.0.3: Исправление ошибок	196
Ядро сервера выпуска 3.0.3	196
Сбои и зависания сервера выпуска 3.0.3	202
Сборка сервера выпуска 3.0.3	204
API/Удалённый интерфейс сервера выпуска 3.0.3	204
Для POSIX сервера выпуска 3.0.3	205
Утилиты сервера выпуска 3.0.3	205
Выпуск Firebird 3.0.2: Исправление ошибок	206
Ядро сервера выпуска 3.0.2	207
Сбои и зависания сервера выпуска 3.0.2	211
Сборка сервера выпуска 3.0.2	212
API выпуска 3.0.2	212
Для операционных систем POSIX (выпуск 3.0.2)	212
Для операционных систем Windows (выпуск 3.0.2)	213
Выпуск Firebird 3.0.1: Исправление ошибок	213
Ядро сервера выпуска 3.0.1	213
Сбои сервера выпуска 3.0.1	218
API/Удаленный интерфейс выпуска 3.0.1	219
Утилиты выпуска 3.0.1	219
Сборка сервера выпуска 3.0.1	222
Особенности для операционных систем POSIX выпуска 3.0.1	
.....	222
Выпуск Firebird 3.0.0: Исправление ошибок	223
Ядро сервера выпуска 3.0.0	223
Сбои сервера выпуска 3.0.0	225
Утилиты выпуска 3.0.0	225
Сборка сервера выпуска 3.0.0	227
Firebird 3.0 релиз-кандидат 2: Исправление ошибок и	
незначительные улучшения	227
Ядро сервера выпуска 3.0 RC2	229
Сбои сервера выпуска 3.0 RC2	233
Шифрование базы данных выпуска 3.0 RC2	234
API/Удаленный интерфейс выпуска 3.0 RC2	235
Утилиты выпуска 3.0 RC2	236

Сборка сервера и программа установки выпуска 3.0

RC2	237
Firebird 3.0 релиз-кандидат 1: Исправление ошибок и незначительные улучшения	238
Ядро сервера выпуска 3.0 RC1	239
Особенности для операционных систем POSIX 3.0 RC1	246
Особенности для операционных систем Windows 3.0 RC1	247
Сервисы API 3.0 RC1	247
Утилиты 3.0 RC1	247
Firebird 3.0 вторая бета версия: Исправление ошибок	249
Ядро сервера 3.0 Бета 2	249
Сбои сервера 3.0 Бета 2	260
API/Удаленный интерфейс 3.0 Бета 2	261
Утилиты 3.0 Бета 2	262
Улучшения в 3.0 Бета 2	263
Firebird 3.0 первая бета версия	266
Ядро сервера 3.0 Бета 1	266
Сбои сервера 3.0 Бета 1	275
API/Удаленный интерфейс 3.0 Бета 1	277
Утилиты 3.0 Бета 1	277
Firebird 3.0 вторая альфа версия: Исправление ошибок	279
Ядро сервера 3.0 Альфа 2	279
Сбои сервера 3.0 Альфа 2	283
API/Удаленный интерфейс Альфа 2	284
Безопасность/Управление пользователями Альфа 2	284
Процедурный язык Альфа 2	285
Язык определения данных Альфа 2	286
Язык управления данными Альфа 2	287
Утилиты Альфа 2	287
Проблемы с установкой Альфа 2	290
Firebird 3.0 первая альфа версия	290
Ядро сервера Альфа 1	290
API/Удаленный интерфейс Альфа 1	296
Процедурный язык Альфа 1	297
Язык определения данных Альфа 1	298
Язык управления данными и DSQL Альфа 1	300
Утилиты командной строки Альфа 1	301
Поддержка международных языков Альфа 1	303
14. Разработчики Firebird 3.0	304
Приложение А: Licence Notice	306

Таблицы

3.1 Режимы работы сервера	16
4.1 Новые системные таблицы	44
6.1 Параметры, доступные в <code>databases.conf</code>	69
9.1 Описание статистических функций	132
9.2 Описание функций линейной регрессии	134
11.1 Аргументы для вывода статистики <code>gbak</code>	177
14.1 Разработчики Firebird 3.0	304

Глава 1

Общие сведения

Благодарим вас за выбор Firebird 3.0. Мы приглашаем Вас тестировать его и связываться с нами по вопросам нахождения и исправления любых ошибок, которые Вы могли бы обнаружить.

Если Вы используете предыдущую версию Firebird, то Вам необходимо изучить Главу 12, *[Проблемы совместимости](#)*, прежде чем Вы попытаетесь что-либо сделать со своими существующими базами данных!

Выпуск Firebird 3.0.3

Список ошибок, исправленных в Firebird 3.0.3, приведён в Главе 13 - *[Исправление ошибок](#)*.

Предупреждение о безопасности

Если Вы уже используете функцию шифрования базы данных или планируете это сделать, то настоятельно рекомендуется выполнить обновление сервера Firebird до текущей версии — 3.0.3. Более полную информацию об этом читайте (CORE-5735).

[Улучшения в выпуске Firebird 3.0.3](#)

([CORE-5727](#)) Улучшена реакция сервера на сигналы отмены/выключения при сканировании длинного списка страниц указателя.

Реализация - В. Хорсун

([**CORE-5712**](#)) Имя ключа шифрования не является сверхсекретной информацией. Его можно прочитать, например, с помощью утилиты или сервиса `gstat`. Однако для работы с этим именем из программы желательно было получить доступ к имени ключа с помощью вызова `API Attachment::getInfo()`. Теперь имя ключа можно получить с использованием информационного элемента `fb_info_crypt_key`.

Реализация - А. Пешков

([**CORE-5704**](#)) В некоторых предложениях инструкции `ALTER DATABASE` требуется обновить одну строку в `RDB$DATABASE`: `SET DEFAULT CHARACTER SET`, `SET LINGER`, `DROP LINGER`. Другие операторы (например, `BEGIN|END BACKUP`, `ENCRYPT`, `DECRYPT` и т.п.) не должны изменять эту строку.

В предыдущих версиях для предотвращения выполнения одновременных экземпляров `ALTER DATABASE`, работающих в параллельных транзакциях, сервер запускал обновление записи в `RDB$DATABASE` независимо от предложения, указанных пользователем. Следовательно, любая другая транзакция, которая читает запись из `RDB$DATABASE` в режиме `READ COMMITTED NO RECORD VERSION`, будет ненадолго заблокирована, даже с помощью «фиктивного обновления», которое фактически не обновляло запись.

В некоторых случаях, например, с `ALTER DATABASE END BACKUP`, блокировка может длиться 10 минут или более. И пользователь, похоже, не сможет подключиться к базе данных с помощью, например, `isql` до завершения работы команды `ALTER DATABASE END BACKUP`. На самом деле `isql` осуществит успешное подключение, но будет читать информацию из `RDB$DATABASE` сразу после присоединения, используя транзакцию `READ COMMITTED NO RECORD VERSION WAIT`, а затем просто будет ожидать окончания работы оператора `ALTER DATABASE END BACKUP`.

Начиная с этого выпуска Firebird, обновление записи в `RDB$DATABASE` можно избежать, если это возможно, а также помещается неявная блокировка для предотвращения параллельных прогонов инструкции `ALTER DATABASE`.

Реализация - В. Хорсун

([CORE-5676](#)) Все семантически одинаковые запросы должны иметь одинаковый план. Однако до сих пор оптимизатор понимал только явную ссылку внутри предложения **ORDER BY** и игнорировал бы сортировки, производные от эквивалентных выражений. Теперь он рассмотрит классы эквивалентности для навигации по индексу. См. примеры в тикете.

Реализация — Д. Еманов

([CORE-5674](#)) Теперь можно использовать общие табличные выражения.

Реализация - В. Хорсун

([CORE-5660](#)) Очистка большого количества грязных страниц была сделана более быстрой.

Реализация - В. Хорсун

([CORE-5648](#)) Предприняты меры для избежания сериализации команды `isc_attach_database` при выполнении оператора **EXECUTE STATEMENT**.

Реализация - В. Хорсун

([CORE-5629](#)) Выходные данные утилиты **gstat** теперь включают в себя дату и время выполнения анализа.

Реализация - А. Пешков

([CORE-5614](#)) Этап *слияния* при выполнении физического резервного копирования может выполняться слишком долго, особенно с огромным кэшем страниц. Для его сокращения были внесены изменения.

Реализация - В. Хорсун

([**CORE-5610**](#)) Сообщение "Ошибка во время сборки мусора: завершение соединения" ("Error during sweep: connection shutdown") теперь предоставляет информацию о базе данных, в которой проводилась сборка мусора.

Реализация - А. Пешков

([**CORE-5602**](#)) Повышение производительности при выполнении оператора ALTER DOMAIN в случае, если домен имеет много зависимостей.

Реализация - В. Хорсун

([**CORE-5601**](#)) Сведения о сжатии и статус шифрования соединения (fb_info_conn_flags) были добавлены в вызов функции API getInfo(). Для получения дополнительной информации см. примечания в главе API.

Реализация - А. Пешков

([**CORE-5543**](#)) При восстановлении базы данных с ODS 11.1 теперь правильно заполняется поле RDB\$RELATION_TYPE в метаданных.

Реализация - Д. Еманов

([**CORE-4913**](#)) Улучшена скорость резервного копирования с помощью nBackup при перенаправлении на NAS по протоколу SMB.

Реализация - J. Hejda и В. Хорсун

([**CORE-3295**](#)) Оптимизатор теперь может оценить фактический коэффициент сжатия записи.

Реализация - Д. Еманов

Выпуск Firebird 3.0.2

Список ошибок, исправленных в Firebird 3.0.2, приведён в Главе 13 - *Исправление ошибок*.

Одним из важных исправлений ошибок является устранение серьезной уязвимости системы безопасности, присутствующая во всех предыдущих выпусках и суб-релизах Firebird.

Предупреждение

Данная ошибка была доступна только аутентифицируемым пользователям, что несколько ограничивало риски. Однако настоятельно рекомендуется, чтобы любая предыдущая установка Firebird была обновлена незамедлительно.

Что нового

(CORE-4563) Добавлена поддержка для быстрой/низкой задержки “Быстрый путь TCP Loopback (обратная петля)”, введённой в ОС Windows 8 и сервере 2012. Эта функция, как говорят в Microsoft, улучшает производительность стека TCP для локальных петлевых соединений коротким замыканием стека TCP для местных вызовов. Детальное описание функции приведены в блоге [Microsoft Technet](#).

Реализация - В. Хорсун

Улучшения

Улучшения в Firebird 3.0.2:

(CORE-5475) Теперь можно отфильтровать информацию и предупреждения из журнала трассировки.

Реализация - В. Хорсун

(CORE-5442) Расширенные возможности управления при совместном использовании ключа шифрования базы данных между подключениями к

Superserver.

Реализация - А. Пешков

[\(CORE-5441\)](#) Часто используемые страниц данных теперь кэшируется для уменьшения количества выборов из страниц указателей.

Реализация - В. Хорсун

[\(CORE-5434\)](#) Транзакция только для чтения больше не будет принудительно записывать страницу заголовка\TIP на диск сразу после изменения. Это дает значительное увеличение производительности там, где используются многочисленные легкие транзакции только для чтения. На данном этапе это доступно только для сервера в режиме SS. Для CS и SC это более сложно реализовать. Ориентировочно реализация этого будет сделана в Firebird 4.0.

Реализация - В. Хорсун

[\(CORE-5374\)](#) Имя базы данных сделано доступным для плагина шифрования.

Реализация - А. Пешков

[\(CORE-5332\)](#) *libfbclient.so* скомпилирован для Android (x86 / x86-64 / arm64).

Реализация - М. А. Пора

[\(CORE-5257\)](#) Включено вложение ключей в плагине конфигурации.

Реализация - В. Хорсун

[\(CORE-5204\)](#) Код Linux теперь создан с опцией `--enable-binreloc` и включен в сценарий установки, чтобы обеспечить возможность установки Firebird в директорию, отличную от `/opt/firebird`.

Реализация - А. Пешков

[\(CORE-4486\)](#) Сделан фильтр трассировки для ошибок INCLUDE/EXCLUDE по их мнемоническим именам.

Реализация - В. Хорсун

[\(CORE-3885\)](#) Добавлен порт Андроида (arm32).

Реализация - А. Пешков

[\(CORE-3637\)](#) Был выполнен и протестирован порт для Linux на старом процессоре Motorola 680000 для удовлетворения некоторых требований операционной системы Debian.

Реализация - А. Пешков

[\(CORE-1095\)](#) Добавлена поддержка, позволяющая использовать операторы SELECT для предиката BETWEEN.

Реализация — Д. Еманов

Выпуск Firebird 3.0.1

Список ошибок, исправленных в Firebird 3.0.1, приведён в Главе 13 - *Исправление ошибок*.

[\(CORE-5266\)](#) Оператор CREATE ИЛИ ALTER USER SYSDBA PASSWORD <password> теперь может использоваться для инициализации пустой базы данных безопасности SecurityN.fdb.

Реализация - А. Пешков

[\(CORE-5229\)](#) Для URL-подобных строк подключения в операционной системе Windows (см. *Новые форматы подключения для локальных клиентов Superserver в Windows*) включено ограничение поиска для IPv4.

Реализация - Michal Kubecsek

[\(CORE-5216\)](#) Для ошибок времени выполнения, возникающих при выполнении EXECUTE BLOCK, теперь доступны номера строк и столбцов (контекст местоположения).

Реализация — Д. Еманов

[\(CORE-5205\)](#) Был добавлен коммутатор для создания двоичных файлов POSIX со встроенной библиотекой libtommath.

Реализация - А. Пешков

[\(CORE-5201\)](#) gbak теперь возвращает ненулевой код результата при неудачном восстановлении при создании и активации отсроченного индекса пользователя.

Реализация - А. Пешков

[\(CORE-5167\)](#) Неявное преобразование между типами Boolean и string теперь выполняется автоматически, когда используется строка «истина» («True») или «ложь» («False») как значение в выражении. Регистр при этом не учитывается. Не работоспособно при использовании операторов IS, NOT, AND или OR; недоступно для UNKNOWN.

Реализация - A. dos Santos Fernandes

Отчёты об ошибках

Ошибки, исправленные с момента выпуска версии 3..0.0, перечислены и описаны в главе 13 - [*Исправление ошибок*](#).

- Если Вы думаете, что обнаружили ошибку в этой версии, пожалуйста, обратите внимание на инструкцию по сообщению об ошибках, приведенные в статье [Как эффективно сообщать об ошибках](#) 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

Вы найдете все документы (README), упомянутые в этих замечаниях к выпуску, а также многие другие, не упомянутые в подкаталоге doc директории вашей установки Firebird 3.0.

Глава 2

Новое в Firebird 3.0.2

Основными задачами при разработке Firebird 3 были унификация архитектуры сервера, улучшение поддержки SMP (*Symmetric Multiprocessing* - *Симметричная мультипроцессорность*) и многоядерных аппаратных платформ. Дополнительными целями являлись улучшение поточной обработки процессов ядром сервера и совместное использование кэша страниц для всех потоков и соединений с БД.

Наряду с этими целями появились новые стратегии повышения производительности, оптимизации запросов, мониторинга и масштабирования, удовлетворения спроса на дополнительные параметры безопасности. В язык SQL добавлен ряд популярных функций, включая долгожданную поддержку для типа данных Boolean и связанных с ним логических предикатов.

Новые возможности

В нижеприведённом списке приведены сведения о функциях и изменениях со ссылками на главы и разделы, в которых содержится более подробная информация о них.

Унификация Firebird завершена

С завершением разработки полной поддержки SMP Суперсервера ядро Firebird - теперь объединенная библиотека, поддерживающая единую ODS, загружаемую как встроенный механизм или как исполняемая программа “сетевого слушателя”. Выбор режима работы сервера определяется настройками нового параметра конфигурации **ServerMode**, задающего режим блокировки и кэширования. Он указан на глобальном уровне в файле `firebird.conf`.

По умолчанию **ServerMode** = **Super** (псевдоним **ThreadedDedicated** — распараллеленные потоки), т. е. Суперсервер.

Примечание

Файл с алиасами `aliases.conf` теперь заменён файлом `database.conf`, включающего в себя не только псевдонимы для баз данных, но и

(необязательные) параметры конфигурации, позволяющие настраивать базы данных и индивидуальные базы данных безопасности.

Более подробно об этом смотрите в Главе 3 - *[Изменения в ядре Firebird](#)*.

Полная поддержка SMP для режима SuperServer

В режиме SuperServer теперь доступно использование нескольких процессоров и ядер для каждого подключения к базам данных.

[CORE-775](#)

Реализация - В. Хорсун

Новое объектно-ориентированное C++ API

Объектно-ориентированные API интерфейсы C++ позволяют подключать внешние программы и разрешить их безопасную работу внутри движка (ядра) Firebird, включая следующие возможности (но не ограничиваясь только ими):

- Схемы шифрования данных;
- Схемы аутентификации пользователей, включая обмен безопасными ключами;
- Наконец, это поддержка плагинов для хранимых процедур, триггеров и функций, написанных на Java, C++, Object Pascal и др.

Конфигурация «для каждой базы данных» (“Per-Database”)

Пользовательская конфигурация на уровне базы данных теперь может быть осуществлена с помощью записей в файле database.conf (ранее aliases.conf). Подробности читайте в разделе *[Конфигурация “Для каждой базы данных” \(“Per-database”\)](#)*.

Увеличение пределов в нескольких областях

Увеличены максимальные значения для идентификаторов транзакций, подключений к базам данных и операторов. Для 64-разрядных серверов увеличено максимальное количество буферов в кэше. Увеличен максимальный размер базы данных. Подробная информация об этом приведена в Главе 4 - *[Изменения в Firebird API и ODS](#)*.

Несколько баз данных безопасности

Firebird теперь поддерживает управление доступом пользователей

через несколько баз данных безопасности на сервере. Используя параметр SecurityDatabase (см. [База данных безопасности - SecurityDatabase](#)) в файле databases.conf для любой базы данных можно задать собственную базу данных безопасности, отличную от используемой по умолчанию security3.fdb. Пользовательские структуры также могут даже быть определены в этих базах данных самостоятельно.

Примечание

В таблицу мониторинга MON\$DATABASE добавлен флаг MON\$SEC_DATABASE для помощи в определении, какая база данных безопасности используется - по умолчанию, «своя» или «другая».

Новые команды SQL для управления пользователями и доступом

Изменения в архитектуре сервера, ужесточение правил безопасности и требования к целостности данных, наряду с настоятельными просьбами привели к появлению новых команд SQL для управления пользователями и правами их доступа к объектам. Более подробно об этих командах смотрите в разделе [Возможности SQL для управления доступом](#).

Международные символы в учётных данных пользователей

При использовании новых возможностей аутентификации сервер может принимать имена пользователя и их пароли в наборах символов, отличных от ASCII. Более подробную информацию смотрите в разделе [Международные наборы символов для учетных записей пользователей](#) в главе 7 - [Безопасность](#).

Поддержка новых типов данных

Тип данных BOOLEAN (True/False/Unknown), позволяет осуществить поддержку логических предикатов, например:

```
UPDATE ATABLE
SET MYBOOL = (COLUMN_1 IS DISTINCT
FROM COLUMN_2)
```

Подробности читайте в разделе [Тип данных BOOLEAN](#).

Тип данных IDENTITY генерирует уникальные идентификаторы для определенного столбца из внутреннего генератора. Дополнительную

информацию читайте в разделе [Тип столбца IDENTITY](#).

Управление NULL/NOT NULL атрибутами столбца/домена

Атрибутами NULL/NOT NULL в столбце или домене теперь можно управлять с помощью операторов ALTER TABLE или ALTER DOMAIN, соответственно. Дополнительную информацию читайте в разделе [Управление данными NULL в доменах и столбцах](#).

Поддержка пакетов SQL

Подробную информацию читайте в разделе [Пакеты](#).

DDL триггеров

Теперь триггеры могут быть изменены при изменении или удалении объектов базы данных. Для этого блокируются несанкционированные для выполнения этих задач пользователи. Подробную информацию читайте в разделе [Триггер для DDL](#).

«Оконные» функции в DML

Новая серия аналитических функций для работы с несколькими подмножествами в DML. Подробную информацию читайте в разделе [Оконные \(аналитические\) функции](#)

Статистические функции

Набор статистических функций, возвращающих значения для различных вариантов дисперсии, стандартного отклонения и линейных формул регрессии. Подробную информацию читайте в разделе [Статистические функции](#).

Прокручиваемые курсоры

Запросы теперь поддерживают двунаправленные («прокручиваемые») курсоры, позволяющие осуществлять навигацию как вперед, так и назад, в PSQL и DSQL с поддержкой API. Подробную информацию читайте в разделе [Поддержка курсора с возможностью прокрутки \(двунаправленного\)](#).

Совместимые с SQL - 2008 предложения OFFSET и FETCH

Реализована поддержка предложений OFFSET и FETCH, совместимых с SQL - 2008, в качестве альтернативы предложениям {FIRST и SKIP} или {ROWS и TO}. Подробную информацию читайте в разделе [Операторы OFFSET и FETCH совместимы с SQL-2008](#).

Поддержка IPv6

Firebird 3 теперь может использовать соединения по протоколу IPv6 как на стороне клиента, так и на стороне сервера. Смотрите примечания для нового конфигурационного параметра *IPv6V6Only*.

Онлайновая Проверка базы данных

Онлайновая (*On-line*) проверка базы данных, т. е. проверка без эксклюзивного доступа к базе данных и без отключения пользователей, впервые реализованная в Firebird 2.5.4, была портирована в Firebird 3.0. Подробную информацию читайте в разделе. *Выполнение проверки базы данных в режиме онлайн*.

Вывод статистики gbak во время выполнения

Подробный вывод из gbak теперь может включать статистику времени выполнения (см. раздел *Статистика в реальном времени с подробным выводом*), время отклика, прочитанные и записанные страницы. Также поддерживается в сервисах API сервера (см. *Поддержка SPB новой статистики, выводимой gbak*).

Совместимость со старыми версиями

Ряд заметок о совместимости со старыми версиями Firebird приведен в Главе 12 - *Проблемы совместимости*. Приводятся инструкции по инициализации базы данных безопасности (см. раздел *Инициализация базы данных безопасности*), если это не было сделано Вами при установке Firebird 3.x и для настройки устаревшего стиля аутентификации (см. раздел *Устаревшая аутентификация*).

Глава 3

Изменения в ядре Firebird

Изменение архитектуры сервера Firebird 3, начатая в версии 2.5, была завершена с реализацией полной поддержки SMP (*Symmetric Multiprocessing* - *Симметричная мультипроцессорность*) для модели Superserver. В новой схеме можно настроить модель работы по отдельности для каждой базы данных. Метод подключения определяется порядком и содержанием другого параметра, **Providers** и протокола подключения, который выведен из строки подключения, предоставляемой клиентом при запросе соединения с базой данных.

Новая архитектура

Дмитрий Еманов

Новая архитектура сервера Firebird 3.x интегрирует базовые механизмы для режимов Classic, SuperClassic и SuperServer и встроенные модели в общем двоичном файле. Кэш и поведение блокировок, которые отличают режимы работы сервера, теперь определяются внешними параметрами настройки в новом параметре конфигурации **ServerMode**. Метод подключения к серверу определяется параметром **Providers**, и протоколом подключения (Смотрите раздел [Строка подключения и протоколы](#)), который берётся из *строки подключения* клиента, запрашивающего соединение.

Параметры для конфигурирования архитектуры сервера определяются глобально (в файле `firebird.conf`). Провайдеры могут переопределяться отдельно для каждой базы данных (в файле `databases.conf`).

Примечание

Файл `databases.conf` - это замена файла `aliases.conf` из предыдущих версий. В Firebird 3 роль этого файла гораздо больше, чем просто поиск путей к файлам баз данных. Больше подробностей о том, что можно настроить на уровне базы данных, читайте в Главе 6 [Изменения и дополнения в параметрах конфигурации](#).

Режимы сервера

Таблица 3.1. Режимы работы сервера

Режим сервера	Описание	Модель ресурсов	Провайдеры
Super	Выделение потоков	База данных открыта исключительно одним серверным процессом. Подключения пользователей обрабатываются потоками, запущенными из общего пула, и все они совместно используют один кэш страниц базы данных внутри процесса. Это установка по умолчанию.	Удаленный, Engine12 ¹ , обратная петля (Loopback)
SuperClassic	Совместно используемые потоки	Базы данных открываются одним серверным процессом, но доступ не является исключительным: встроенный процесс может одновременно открывать одну и ту же базу данных. Подключения пользователей обрабатываются потоками, запущенными из общего пула, каждый из которых имеет свой собственный кэш страниц базы данных.	Удаленный, Engine12, обратная петля (Loopback)
Classic	Многопроцессорный	Для каждого подключения на сервере запускается отдельный процесс. База данных может быть открыта несколькими классическими процессами, включая локальные процессы для встроенного доступа. Каждый процесс имеет свой собственный кэш страниц	Удаленный, Engine12, обратная петля (Loopback)

		базы данных.	
¹ Только при эксклюзивном доступе			

Провайдеры

Провайдеры (поставщики) определяют, какой метод используется для подключения клиента к серверу, то есть через сеть, локально, через локальный loopback («localhost») или через прямое локальное соединение (старый *libfbembed.so* для POSIX, теперь реализованный как плагин библиотеки *libEngine12.so*; для Window *engine12.dll*; для MacOSX *engine12.dylib*).

- В файле *firebird.conf* все они доступны по умолчанию, следующим образом:

```
#Providers = Remote,Engine12,Loopback
```

- В файле *databases.conf* один или несколько поставщиков могут быть заблокированы вводом этой строки из *firebird.conf*. Затем раскомментируйте её и удалите нежелательного поставщика или поставщиков.

Архитектура провайдеров

Александр Пешков

Несмотря на то, что провайдеры одна из главных особенностей Firebird 3.x, их архитектура не нова. Провайдеры существовали исторически в предшественниках Firebird и, хотя и хорошо скрытые, присутствуют во всех предыдущих версиях Firebird. Они были использовались первоначально для решения задачи, которая выполнялась с тех пор на “интерфейсном уровне” средствами ODBC, ADO, BDE и т.п., чтобы включить доступ к различным механизмам базы данных, используя единый внешний интерфейс.

Впоследствии, эта архитектура Провайдеров (известная тогда как Открытые системы реляционного интерфейса - Open Systems Relational Interface, OSRI) также показала себя как очень эффективный инструмент для поддержки подключения к старым и новым форматам баз данных с различными версиями дисковых структур на одном сервере, используя соединения с локальными и удаленными базами данных.

Провайдеры, реализованные в Firebird 3, позволяют поддерживать все

эти режимы (удаленные соединения, базы данных с различными ODS, внешние механизмы), а также провайдеры объединения в цепочку. Объединение в цепочку - термин для ситуации, когда провайдер использует обратный вызов для стандартного API при выполнении работы с базой данных.

Компоненты

Основным элементом архитектуры провайдеров является **Y-клапан**. При первом подключении или создании базы данных **Y-клапан** проверяет список известных поставщиков и вызывает их по одному, пока один из них не завершит успешно работу. Для уже установленного соединения немедленно вызывается соответствующий провайдер с почти нулевыми накладными расходами.

Ниже приведены некоторые примеры работы **Y-клапана**, когда он выбирает соответствующего провайдера при подключении к базе данных. Они используют конфигурацию по умолчанию, в которую включены три провайдера:

- **Remote** (устанавливает удалённое сетевое подключение);
- **Engine12** (главное ядро сервера баз данных);
- **Loopback** (осуществляет сетевое соединение с локальным сервером по <имени базы данных> без явного предоставления сетевого протокола).

Типичная конфигурация клиента работает следующим образом: при подключении к базе данных RemoteHost:dbname (синтаксис TCP/IP) или \\RemoteHost\dbname (NetBios) удаленный провайдер обнаруживает явный синтаксис сетевого протокола (см. *Строка подключения и протоколы*) и, найдя его в списке поставщиков, перенаправляет вызов на удалённый сервер (RemoteHost).

Если <имя базы данных> содержит только имя базы данных без указания сетевого протокола, то удаленный провайдер отклоняет его, и пытается с помощью Engine12 соединиться с указанным файлом базы данных. Если это удастся, мы получаем встроенное соединение с базой данных.

Примечание

Специальная «встроенная библиотека» (fbembedded) больше не требуется. Для установки встроенного соединения стандартный клиент

загружает соответствующего поставщика и становится встроенным сервером.

Отказоустойчивость

Что происходит, если сервер при попытке подключиться к базе данных возвращает ошибку?

- Не имеет интереса случай, когда файла базы данных, к которому нужно подключиться, не существует;
- Встроенное подключение может завершиться неудачей, если пользователь не имеет достаточных прав для открытия файла базы данных;

Примечание

Таким образом установка прав доступа к базе данных является обязательным требованием для корректной работы Суперсервера.

- При отказе Engine12 в доступе к базе данных провайдер пытается подключиться через **Loopback**. Этот тип подключения не сильно отличается от **Remote**, за исключением того, что он пытается получить доступ к файлу базы данных <dbname> на сервере через локальный шлейф **TCP/IP**.

Под операционной системой **Windows** провайдер попытается подключение через **XNET**, затем обратную петлю (**loopback**) **TCP/IP** (со строкой подключения вида **localhost:<dbname>**), затем обратную петлю именованных каналов (**NetBEUI**) (со строкой подключения вида **\\.\<dbname>**). На сервере может быть отключен протокол **XNET** (или любой другой протокол), поэтому при подключении пробуются все протоколы. Под операционной системой **POSIX** поддерживается только протокол **TCP/IP**, другие протоколы не доступны.

Если подключение успешно выполнено, то оно работает как будто установлено удаленное соединение с базой данных, несмотря на то, что файл базы данных расположен на локальной машине.

Другие провайдеры

Использование провайдеров не ограничено тремя стандартными.

Firebird 3.x не поддерживает структуры баз данных ниже ODS 12. Удаление поддержки старых структур из сервера помогает упростить его код и немного увеличить скорость работы. Учитывая, что этот выигрыш в быстродействии иногда происходит в критических по отношению к производительности случаях, как, например, поиск ключа в индексе, то исключение старого кода и связанных с ним ответвлений действительно позволили быстрее работать серверу Firebird.

Тем не менее архитектура провайдеров позволяет получать доступ к старым базам данных при переходе на более высокую версию Firebird. Подходящий поставщик может быть рассмотрен для включения в более поздний суб выпуск Firebird.

Пользовательские провайдеры

Сильной стороной архитектуры провайдеров является способность добавлять своих поставщиков серверу, клиенту или и то, и другое.

Что еще может понадобиться на клиенте, кроме удаленного соединения? Вспомните цепочку поставщиков, упомянутую ранее. Представьте себе случай, когда доступ к базе данных осуществляется через очень медленное сетевое соединение, например, как 3G или, что еще хуже, GPRS. Первое, что приходит на ум, как способ ускорить работу при таком соединении с базой данных, - это кэширование на клиенте отдельных больших, редко изменяемых таблицы. Такие системы были фактически реализованы, но для этого нужно было переименовать fbclient и загружать его в свою собственную библиотеку, называемую fbclient, что позволяет использовать стандартные инструменты для доступа к базе данных одновременно с кэшированием требуемых таблиц. Такой подход работает, но он явно не идеален.

В архитектуре провайдеров вместо переименования библиотек просто добавляется локальный поставщик кэширования, который может использовать любой метод для обнаружения подключений к нему (что-то вроде префикса **cache@** перед именем базы данных или другой префикс на Ваше усмотрение).

В этом примере, когда используется строка подключения **cache@RemoteHost:dbname**, кэширующий провайдер осуществляет соединение и вызывает Y-клапан еще раз с традиционным именем базы данных **RemoteHost:dbname**. Когда пользователь позже выполняет любой запрос к своей базе данных, кэширующий провайдер получает контроль над ним и для локально закэшированной таблицы может запретить запросы к удаленному серверу.

Использование цепочки позволяет реализовать множество других

полезных вещей. Примером может быть репликация в сервере MySQL на уровне инструкций без необходимости запуска триггеров: просто повторите те же запросы для хоста репликации, после подтверждения транзакции. В этом случае поставщик цепочки будет установлен на сервере, а не на клиенте, и не потребуются никакая модификация строки подключения.

Примечание

Однако репликация на уровне операторов (запросов) является очень сомнительной функцией.

Чтобы избежать циклического переключения при выполнении обратного вызова к Y-клапану в момент подключения к базе данных, такой поставщик может изменить список провайдеров, используя параметр `isc_dpb_config` в DPB. Такой же метод может быть использован и клиентом.

Больше подробную информацию об этом Вы можете прочитать в Главе 6 *Изменения и дополнения в параметрах конфигурации*.

Нельзя игнорировать возможность доступа к внешним механизмам базы данных с использованием провайдеров. Это может показаться странным, учитывая количество инструментов, доступных для такого рода задач. Подумайте о возможности доступа к другим базам данных Firebird с использованием оператора **EXECUTE STATEMENT**, который стал доступен в Firebird начиная с версии 2.5. Провайдер ODBC или другой какой либо другой инструмент для доступа к различным источникам данных, выполняя оператор **EXECUTE STATEMENT**, получает прямой доступ к процедурам и триггерам, к данным из любой базы данных, имеющей драйвер для выбранного инструмента доступа. Возможно даже иметь провайдер, обеспечивающий доступ к некоторому определенному типу внешней базы данных, если есть какие то причины избегать использования ODBC.

Провайдеры: Вопросы (В.) и Ответы (О.)

В. Интерфейсы и провайдеры, вероятно, очень хороши, но у меня есть старая программа, написанная с использованием простых функций API и по многим причинам я не могу переписать её в ближайшее время. Означает ли это, что у меня будут проблемы с миграцией на Firebird 3?

О. Никаких проблем нет. Старый API поддерживается в Firebird 3.x для обратной совместимости и будет поддерживаться и в следующих версиях, если будет потребность в этом..

В. Не пострадает ли производительность при использовании старого API?

О. Функционально API реализован как очень тонкий слой над интерфейсами. Код в большинстве случаев тривиален: преобразование переданных дескрипторов в указатели на интерфейсы, -до сих пор именуемые "проверкой дескрипторов") - и вызов соответствующей функции в интерфейсе.

В функциях, выполняющих операции SQL и выборку данных, кодирования немного больше и оно включает в себя конструкцию SQLDA. Изменения данных, связанные с SQLDA, всегда создавали накладные расходы. Логика между новыми и старыми интерфейсами API не добавляет значительных изменений к этим накладным расходам.

Строка подключения и протоколы

Строка подключения относится к локальному или удаленному пути к базе данных, к которой клиент запрашивает подключение (соединение). Синтаксис строки подключения определяет транспортный протокол, по которому осуществляют общение клиенты и сервер. Устаревший синтаксис для доступных протоколов, поддерживаемых всеми версиями Firebird, следующий (<database file path or alias> - это <путь к базе данных или её алиас>):

Для протокола TCP/IP (также известного как INET протокол)

<host> [/ <port>] : <database file path or alias>

Для именованных каналов (так называемый протокол NetBEUI, или WNET):

\\ <host> [@<port>] \<database file path or alias>

Для локальных подключений просто имя файла:

<database file path or alias>

Локальное соединение подразумевает, имя хоста <host> в строке подключения пропущено. В зависимости от настроек, платформы и версии Firebird, оно может выполняться через встроенный движок (Engine), протокол XNET (с общей памятью) или TCP/IP локальный шлейф (localhost).

Примеры

Подключение через TCP/IP с использованием имени базы данных:

```
192.168.0.11:/db/mydb.fdb  
192.168.0.11:C:\db\mydb.fdb  
myserver:C:\db\mydb.fdb  
localhost:/db/mydb.fdb
```

Подключение через TCP/IP с использованием алиаса базы данных:

```
192.168.0.11:mydb  
myserver:mydb  
localhost:mydb
```

Подключение через TCP/IP с использованием нестандартного порта 3051:

```
192.168.0.11/3051:C:\db\mydb.fdb  
192.168.0.11/3051:mydb  
myserver/3051:/db/mydb.fdb  
localhost/3051:/db/mydb.fdb  
myserver/3051:mydb  
localhost/3051:mydb
```

Подключение через TCP/IP с использованием нестандартного имени сервиса:

```
192.168.0.11/fb_db:C:\db\mydb.fdb  
192.168.0.11/fb_db:mydb  
localhost/fb_db:/db/mydb.fdb  
myserver/fb_db:/db/mydb.fdb  
myserver/fb_db:mydb  
localhost/fb_db:mydb
```

Подключение через именованные каналы (только для ОС Windows):

```
\\myserver\C:\db\mydb.fdb  
\\myserver@fb_db\C:\db\mydb.fdb
```

Локальное подключение:

```
/db/mydb.fdb  
C:\db\mydb.fdb  
mydb
```

Строки соединения URL-стиля

В Firebird 3.0 введён дополнительный, обобщенный синтаксис URL-стиля для строк подключения. Шаблон:

[<protocol> : //[<host> [<port>]] / <database file path or alias>

<protocol> ::= INET | WNET | XNET

INET разрешает протоколы TCP/IP, WNET для именованных каналов, а XNET - старый «локальный протокол Windows» (с общей памятью).

Примеры

Подключение через TCP/IP с использованием имени базы данных:

```
inet://192.168.0.11//db/mydb.fdb  
inet://192.168.0.11/C:\db\mydb.fdb  
inet://myserver/C:\db\mydb.fdb  
inet://localhost//db/mydb.fdb
```

Подключение через TCP/IP с использованием алиаса базы данных:

```
inet://192.168.0.11/mydb  
inet://myserver/mydb  
inet://localhost/mydb
```

Подключение через TCP/IP с использованием нестандартного порта 3051:

```
inet://192.168.0.11:3051/C:\db\mydb.fdb  
inet://192.168.0.11:3051/mydb  
inet://myserver:3051//db/mydb.fdb
```

```
inet://localhost:3051//db/mydb.fdb
inet://myserver:3051/mydb
inet://localhost:3051/mydb
```

Подключение через TCP/IP с использованием нестандартного имени сервиса:

```
inet://192.168.0.11:fb_db/C:\db\mydb.fdb
inet://192.168.0.11:fb_db/mydb
inet://localhost:fb_db//db/mydb.fdb
inet://myserver:fb_db//db/mydb.fdb
inet://myserver:fb_db/mydb
inet://localhost:fb_db/mydb
```

Подключение через именованные каналы (только для ОС Windows):

```
wnet://myserver/C:\db\mydb.fdb
wnet://myserver:fb_db/C:\db\mydb.fdb
```

Локальное подключение через TCP/IP:

```
inet:///db/mydb.fdb
inet://C:\db\mydb.fdb
inet://mydb
```

Локальное подключение через именованные каналы:

```
wnet://C:\db\mydb.fdb
wnet://mydb
```

Локальное подключение с использованием общей памяти:

```
xnet://C:\db\mydb.fdb
xnet://mydb
```

Локальное подключение (по умолчанию встроенное):

```
/db/mydb.fdb  
C:\db\mydb.fdb  
mydb
```

Локальное соединение подразумевает, имя хоста <host> в строке подключения пропущено. В зависимости от настроек, платформы и версии Firebird, оно может выполняться через встроенный движок (Engine), протокол XNET (с общей памятью) или через TCP/IP локальный шлейф (localhost).

На стороне сервера конфигурация провайдера (смотрите раздел [Провайдеры](#)) находится по умолчанию в следующем порядке Remote, Engine12, Loopback. Если удаленный провайдер не соответствует строке подключения, поскольку отсутствуют протоколы или узлы хоста, тогда используется Engine12, обрабатывающий его как соединение без подключения. Чтобы подключиться локально с использованием определенного транспортного протокола, необходимо явно указать этот протокол:

```
inet://<database file path or alias>  
или  
wnet://<database file path or alias>  
или  
xnet://<database file path or alias>
```

Примечание

Протоколы WNET (именованные каналы) и XNET (с общей памятью) доступны только в ОС Windows.

Плагины

Александр Пешков

Начиная с 3-й версии архитектура Firebird поддерживает плагины. Для

ряда predetermined точек в коде Firebird разработчик может написать свой собственный фрагмент кода для выполнения, если это необходимо. Плагин не обязательно должен быть написан третьим лицом: Firebird имеет несколько встроенных плагинов. Даже некоторые основные части Firebird реализованы как плагины.

Что такое плагин?

Термин «плагин» используется для обозначения связанных, но разных вещей:

- динамическая библиотека, содержащая код, который будет загружен как плагин (часто называемый подключаемым модулем) и сохранённая в каталоге `$FIREBIRD/plugins`;
- кода, реализующего подключаемый модуль. Это немного отличается от библиотеки, поскольку одна динамическая библиотека может содержать код для нескольких плагинов;
- фабрика плагинов: объект, созданный с этим кодом (чистый виртуальный класс C++), создающий экземпляры подключаемого модуля (плагины) по запросу Firebird;
- *экземпляр* подключаемого модуля, созданный этой фабрикой.

Типы плагинов

Архитектура Firebird позволяет создавать плагины предварительно определенных типов. Firebird будет иметь фиксированный набор поддерживаемых типов подключаемых модулей. Единственное требование при создании нового плагина — это требование изменить код Firebird. Архитектура плагинов облегчает как добавление новых типов подключаемых модулей, так и упрощает их кодирование.

Для внедрения подключаемого модуля, например, для шифрования базы данных на диске, должен быть подготовлен код Firebird для него: он должен иметь точку, из которой будет вызываться плагин.

Набор типов подключаемых модулей, реализованных в Firebird 3, включает в себя:

Аутентификация пользователя:

- **AuthServer** (проверяет учетные данные пользователя на сервере при использовании логинов);
- **AuthClient** (подготавливает учетные данные, передаваемые по сети);
- **AuthUserManagement** (ведет список пользователей на сервере в формате, известном **AuthServer**).

Внешние механизмы («движки»)

- Контролирует использование различных «движков», смотрите раздел [Внешние средства ядра сервера](#).

Трассировка

- Плагин трассировки (Trace) был введен в Firebird 2.5, но способ взаимодействия с ядром был изменен в Firebird 3, чтобы соответствовать новым общим правилам.

Шифрование

Плагины шифрования используются для

- сети (**WireCrypt**);
- базы данных (**DbCrypt**);
- плагин помощника (**KeyHolder**), используемый поддержки секретного ключа (ключей) для **DbCrypt**.

Провайдеры

Firebird 3 поддерживает [Провайдеры](#) в качестве типа подключаемого модуля.

Технические детали

Плагины используют ряд специальных интерфейсов Firebird. Все определенные для плагина интерфейсы — это считываемая ссылка,

помещающая их под своё управление на время работы. Интерфейсы объявлены в подключаемом файле `include.h`. Простой пример для написания подключаемого модуля можно найти в `DbCrypt_example`.

Примечание

В примере фактически не выполняется никакого шифрования, это всего лишь образец того, как писать плагин. Полные инструкции для написания плагинов в данном документе не рассматриваются.

Особенности написания плагинов

К особенностям написания плагинов можно отнести:

- Вы можете написать плагин на любом языке, который поддерживает чистые виртуальные интерфейсы. Также надо написать запросы к интерфейсу, если они отсутствуют;
- Как и в случае с UDF, вы можете добавить любой разумный код к своему плагину с учётом его целесообразности. Например, запрос ввода пользователя в консоли сервера из плагина вряд ли «разумен»!
- При необходимости Ваш плагин может использовать вызовы API Firebird. Например, сервер аутентификации по умолчанию и менеджер пользователей используют базу данных Firebird для хранения учетных записей;
- Firebird предоставляет набор интерфейсов для настройки ваших плагинов. Их не обязательно использовать, поскольку код подключаемого модуля является общим и может использовать любой полезный метод для сбора информации о конфигурации. Однако, использование стандартных инструментов обеспечивает общность с установленным стилем конфигурации и должно сэкономить время на разработку Вашего плагина и его документирование.

Настройка плагинов

Настройка плагинов состоит из двух частей:

1. Ядру сервера должен быть известен список плагинов, которые он

должен загружать;

2. Иногда сами плагины нуждаются в настройке.

Плагины, которые нужно загрузить для каждого типа подключаемого модуля, определяются в основном файле конфигурации `firebird.conf` (как правило с настройками по умолчанию). Собственные плагины Firebird 3 рассмотрены в главе 6 *Изменения и дополнения в параметрах конфигурации*. Таким образом, набор, обеспечивающий нормальную работу в серверных, клиентских и встроенных случаях, состоит из:

- `AuthServer = Srp, Win_Sspi`
- `AuthClient = Srp, Win_Sspi, Legacy_Auth`
- `UserManager = Srp`
- `TracePlugin = fbtrace`
- `Providers = Remote,Engine12,Loopback`
- `WireCryptPlugin = Arc4`

Примечание

При добавлении других плагинов Вы должны записать его описание в файле `firebird.conf`. Кроме того это служит дополнительной мерой безопасности, позволяющей избежать загрузки неизвестного кода.

В качестве примера рассмотрим запись `TracePlugin = fbtrace`. Что означает в ней значение `fbtrace`? В простейшем случае оно может указывать имя динамической библиотеки, но точный ответ более сложен

Как отмечалось ранее, один подключаемый модуль может реализовать больше, чем один плагин. Кроме того, у единственного плагина может быть больше чем одна конфигурация с отдельной сменной фабрикой, создаваемой для каждой конфигурации. Каждый из этих трех объектных контекстов (модуль | реализация | фабрика) имеет свое собственное имя:

- Имя модуля совпадает с именем файла динамической библиотеки;
- Имя плагина задаёт разработчик. Оно жестко прописано внутри

модуля;

- Название фабрики по умолчанию совпадает с именем имени реализации плагина. Это имя фабрики, которое фактически используется в `firebird.conf`.

Обычно модуль содержит один плагин, который работает только с одной конфигурацией, и все три имени равны, а значит настройка плагина не требуется. Например, плагин `libEngine12.so/Engine12.dll/Engine12.dylib`, содержит реализацию встроенного поставщика `Engine12`. Для его загрузки не требуется ничего, кроме записи `Providers = Engine12`.

Для создания более сложных плагинов необходимо ознакомиться с дополнительными материалами, например, [здесь](#).

plugins.conf

Файл `$ (root) /plugins.conf` содержит два типа записей: **config** и **plugin**.

Записи в секции **plugin** — это ряд правил для загрузки и активации плагина. Формат записей в этой секции следующий:

```
Plugin = PlugName ## имя плагина, на которое нужно ссылаться в
        файле firebird.conf
{
    Module = LibName ## имя динамической библиотеки
    RegisterName = RegName ## имя, данное плагину
разработчиком
    Config = ConfName ## имя используемой записи
конфигурации
    ConfigFile = ConfFile ## имя файла, содержащего
конфигурацию подключаемого модуля
}
```

Когда возникает необходимость подключения плагина **PlugName**, Firebird загружает библиотеку **LibName** и находит подключаемый модуль с именем **RegName**. Настройки конфигурации из записи **ConfName** или файла **ConfFile** передаются в библиотеку.

Примечание

Если указаны как **ConfName**, так и **ConfFile**, будет использоваться запись конфигурации **ConfName**.

Если оба параметра отсутствуют, то будет использоваться **PlugName**. Если параметр **ConfigFile** присутствует и его имя совпадает с именем динамической библиотеки, но имеет расширение .conf, то будет использоваться **ConfigFile**.

Предполагается, что **ConfigFile** будет использовать формат **Key = Value**, как и в других конфигурационных файлах Firebird.

В записи конфигурации плагина используется тот же формат:

```
Config = ConfName
{
    Key1 = Value1
    Key2 = Value2
    ...
}
```

Пример настройки

Предположим, у вас есть сервер и некоторые клиенты доверяют шифрованию сетевого трафика от одного поставщика, а другие предпочитают другой. У них разные лицензии для соответствующих клиентских компонентов, но оба производителя используют имя «BestCrypt» для своих продуктов.

В этом случае требуется переименование библиотек, например, в WC1 и WC2, т. к. в одном каталоге не может быть двух файлов с одним и тем же именем. Теперь модули перестают загружаться автоматически, потому что ни один из них не вызывает теперь «BestCrypt».

Для решения этой проблемы файл *plug-ins.conf* должен содержать что-то вроде этого:

```
Plugin = WC1
{
    RegisterName = BestCrypt
}
```

```
Plugin = WC2
{
    RegisterName = BestCrypt
}
```

Имена модулей будут автоматически установлены соответственно в WC1 и WC2 и они будут найдены. Таким способом Вы можете добавить любую конфигурационную информацию, необходимую для плагинов.

Не забудьте изменить файл `firebird.conf`, чтобы включить оба плагина в параметре **WireCryptPlugin**:

```
WireCryptPlugin = WC1, WC2
```

Теперь сервер при подключении клиента автоматически выберет нужный плагин.

Другой пример распространяется вместе с Firebird. Он находится в директории `$(root)/plugins.conf` и используется для настройки одного из стандартных плагинов, UDR. Поскольку он был написан для использования конфигурации, отличной от настроек по умолчанию, имя модуля и один параметр конфигурации указаны явно.

Плагины: Вопросы (В.) и Ответы (О.)

В. В конфигурации по умолчанию есть плагины с именем *Remote*, *Loopback*, *Arc4*, но нет библиотек с такими именами. Как они работают?

О. Они являются «встроенными» в библиотеку *fbclient* плагинами и, таким образом, всегда присутствуют. Их существование обусловлено старой особенностью распространять клиент Firebird для Windows как единую динамическую библиотеку (dll). Функция сохранена для случаев, когда используется стандартный набор плагинов.

В. Для чего предназначены плагины *Srp* и *Arc4*?

О. *Srp* реализует протокол Secure Remote Passwords (безопасные удалённые пароли), используемый в Firebird 3 по умолчанию для аутентификации пользователей. Эффективная длина пароля составляет 20 байт, делая его устойчивым к большинству атак (включая «человек посередине»). Плагин *Srp* работает без необходимости обмена ключами между клиентом и сервером. Плагин *Arc4* реализует шифрование по методу RC4. Преимуществом такого шифрования является то, что можно генерировать уникальный криптографически сильный ключ как на клиенте,

так и на сервере, что делает невозможным перехват данных, передаваемых по сети во время проверки пароля *SRP*. Ключ используется после квитирования (получения данных) для *SRP* от *Arc4*, что делает безопасным шифрование без необходимости явного обмена ключами между клиентом и сервером.

В. Для чего предназначены плагины *Win_Sspi* и *Legacy_Auth*?

О. Windows SSPI используется начиная с Firebird 2.1 для аутентификации доверенных пользователей Windows. *Legacy_Auth* — это плагин совместимости, обеспечения возможности соединения клиентом Firebird 3 к более старым серверам. По умолчанию плагин включён в клиентскую библиотеку. Обратите внимание, что этот плагин для обеспечения совместимости все ещё передает простые пароли по сети. На сервере он работает с *security3.fdb* так же, как с базой данных безопасности из Firebird 2.5. Следует избегать его использования, за исключением ситуаций, когда Вы хорошо понимаете, чем Вы рискуете. Чтобы использовать *Legacy_Auth* Вам необходимо запретить шифрование сетевого трафика в *firebird.conf*, уменьшив параметр *Default Required* для параметра *WireCrypt*. Либо явно задать значение параметра *WireCrypt*:

WireCrypt = Enabled

или

WireCrypt = Disabled

В. Как я могу узнать, какие стандартные плагины используются для аутентификации и управления пользователями?

О. Они перечислены в файле *firebird.conf*.

Внешние средства ядра сервера

Adriano dos Santos Fernandes

Механизм UDR (User Defined Routines - определяемые пользователем подпрограммы) добавляет слой поверх интерфейса движка FirebirdExternal. Он используется для достижения следующих целей:

- Создание способа подключить внешние модули к серверу и сделать их доступными для использования;

- Создание API для возможности внешних модулей регистрировать свои доступные процедуры;
- Создание экземпляров “присоединённых” подпрограмм, не зависящих от внутренних деталей реализации ядра сервера.

Внешние имена

Внешнее имя для UDR задаётся следующим образом:

```
'<module name>!<routine name>!<misc info>'
'<имя модуля>!<стандартное имя>!<информация>'
```

Параметр *<module name>* используется для поиска библиотеки, *<routine name>* - для поиска подпрограммы, зарегистрированной данным модулем, а *<misc info>* является необязательной пользовательской строкой, которая может быть передана для чтения пользователем.

Доступность модуля

Модули, доступные для UDR должны находиться в каталоге, указанном через в атрибуте path (путь) соответствующего раздела файла plugin_config. По умолчанию модуль UDR должен находиться в директории *<fbroot>/plugins/udr*, т. е. там, как указано в атрибуте пути в файле *<fbroot>/plugins/udr_engine.conf*.

Пользовательская библиотека (модуль) должна включать в себя FirebirdUdr.h (или FirebirdUdrCpp.h) и ссылку на библиотеку udr_engine. Процедуры легко регистрируются с использованием макросов, но ничто не мешает Вам делать это вручную.

Примечание

Примеры написания модулей находятся в директории */examples/udr*. Например, там есть примеры, как написать функции, процедуры выборки данных и триггеры. Также показано, как взаимодействовать с ними через устаревший API.

Возможности

Переменные подпрограммы UDR используются несколькими вызовами одной и той же процедуры до ее выгрузки из кэша метаданных. Однако следует отметить, что экземпляры подпрограмм изолированы «на время сеанса».

Набор символов

По умолчанию UDR использует набор символов, заданный клиентом.

Примечание

Впоследствии подпрограммы смогут изменять набор символов, переопределив метод *getCharSet*. Выбранный набор символов будет совместим со старой клиентской библиотекой Firebird, а также с сообщениями, переданными через API FirebirdExternal.

Подключение UDR к базе данных

Для подключения внешней подпрограммы к базе данных обычно используют команду DDL, но, скорее всего, она уже подключена и хорошо протестирована.

Синтаксис

```
{CREATE [OR ALTER]|RECREATE|ALTER} PROCEDURE <имя>  
    [(<список параметров>)] [RETURNS (<список параметров>)]  
EXTERNAL NAME '<внешнее имя>' ENGINE <внешняя  
подпрограмма>
```

```
{CREATE [OR ALTER]|RECREATE|ALTER } FUNCTION <имя>  
    [ <список параметров> ]RETURNS <тип данных>  
EXTERNAL NAME '<внешнее имя>' ENGINE <внешняя  
подпрограмма>
```

```
{CREATE [OR ALTER]|RECREATE|ALTER} TRIGGER <имя>  
...
```

EXTERNAL NAME '<внешнее имя>' ENGINE <внешняя
подпрограмма>

Примеры

```
Create procedure gen_rows
  (start_n integer not null, end_n integer not null)
  returns (n integer not null)
  external name 'udrcpp_example!gen_rows' engine udr;
```

```
Create function wait_event
  (event_name varchar(31) character set ascii)
  returns integer
  external name 'udrcpp_example!wait_event' engine udr;
```

```
Create trigger persons_replicate after insert on persons
  external name 'udrcpp_example!replicate!ds1' engine udr;
```

Как это работает

Внешние имена неоднозначны для Firebird. Они распознаются как внешние подпрограммы. Внешние подпрограммы объявляются в конфигурационных файлах, возможно в том же файле, как и для плагинов (смотрите демонстрационную библиотеку UDR, которая находится в директории \$(root)/plugins.

```
external_engine = UDR {plugin_module = UDR_engine}
```

```
plugin_module = UDR_engine {filename = $(this)/udr_engine
  plugin_config = UDR_config}
```

```
plugin_config = UDR_config {path = $(this)/udr}
```

Когда сервер Firebird хочет загрузить внешнюю подпрограмму (функцию, процедуру или триггер) в свой кэш метаданных, он получает доступ к ней через подключаемый внешний модуль. Используемый подключаемый модуль указан в атрибуте *plugin_module*.

Примечание

В зависимости от архитектуры сервера (Superserver, Classic, SuperClassic) и особенностей реализации Firebird может получать экземпляры внешнего ядра «для базы данных» или «для соединения». В настоящее время сервер всегда получает экземпляры подпрограмм «для базы данных».

Улучшения оптимизатора

Дмитрий Еманов

CORE-4528. Разрешены объединения хэшей/слияний для псевдо полей *DBKEY* или производных выражений.

CORE-1482. Оптимизатор теперь рассматривает предложение **ORDER BY** при принятии решения о порядке соединения.

Другие оптимизации

Владислав Хорсун

CORE-4556. Страницы данных теперь выделяются как группа последовательных упорядоченных страниц.

CORE-4445. Размер основного файла базы данных увеличивается раньше, чем физическое состояние резервной копии изменяется от состояния ожидания до объединения.

CORE-4443. Операционные системы Linux могут теперь использовать возможность «быстрого увеличения размера файлов» («fast file growth»), если они поддерживают её.

CORE-4432. Подключения к базе больше не блокируют другие коннекты при первичном считывании информации для дельта файла резервного копирования.

CORE-4431. Уменьшена конкуренция для таблицы блокировок во время остановки резервного копирования базы данных.

Удалённый интерфейс/Сетевой протокол

Дмитрий Еманов

[CORE-2530](#)

Дальнейшие улучшения было внесены в сетевой протокол Firebird, обеспечив более плотный поток данных и лучшую логику упреждающей выборки. Были реализованы следующие улучшения:

1. Полная длина поля, имеющего значение NULL, больше не передаётся по сети ([CORE-2897](#)). Флаги NULL (4 байта на поле) заменены битовым массивом и по сети передаётся он. Это усовершенствование доступно только для DSQL API, поэтому gbak не выигрывает от этого улучшения, поскольку использует более низкий уровень BLR в API.
2. Алгоритм упреждающей выборки (пакетный приём) теперь умеет принимать о сообщения переменной длины, т.е. Типы данных VARCHAR и NULL могут уменьшить размер передаваемого сообщения, что позволяет передавать больше строк в каждом пакете данных.

Благодарность

Эта работа спонсировалась пожертвованиями, собранными на девятой конференции разработчиков Firebird в Бразилии.

Другие улучшения

Другие улучшения ядра сервера включают в себя следующее.

Увеличен лимит подключений

Paul Beach

([CORE-4439](#)) Максимальное количество соединений (FD_SETSIZE) в ОС Windows для SuperServer и SuperClassic было увеличено с 1024 до 2048.

Улучшенная диагностика ошибок

Дмитрий Еманов

([**CORE-3881**](#)) Сообщение об ошибках, вызванных повреждениями индекса/ограничения целостности (constraint violations), теперь включает в себя значение проблемного ключа.

Обновлена версия ICU

Adriano dos Santos Fernandes

([**CORE-2224**](#)) Версия ICU (*International Components for Unicode* - *Международные компоненты для юникода*) была обновлена до v.52.1.

Улучшена внутренняя отладочная информация

Владислав Хорсун

Новый фильтр BLOB переводит внутреннюю отладочную информацию в текст.

Заменены некорректные сообщения

Claudio Valderrama C.

Некорректные сообщения, посланные парсером (синтаксическим анализатором), когда встречалась ссылка на неопределенный объект, были заменены на сообщения, показывающие, что произошло на самом деле.

Новая псевдоколонка **RDB\$RECORD_VERSION**

Adriano dos Santos Fernandes

Псевдоколонка с именем **RDB\$RECORD_VERSION** возвращает номер транзакции, создавшей текущую версию. Она извлекается так же, как и **RDB\$DB_KEY**, т. е. **select RDB\$RECORD_VERSION from aTable where...**

Системная инициализация скриптов

Александр Пешков

([*CORE-4085*](#)) Скрипты `systemd` доступны при установке Firebird 3 на операционные системы POSIX.

Глава 4

Изменения в Firebird API и ODS

Изменения ODS (структуры данных на диске)

Новый номер ODS

Firebird 3.0 создает базы данных с 12-й версией ODS. В начальном выпуске Firebird 3.0 база данных со старым ODS (версиями, меньшими 12-й) не может быть открыта. Чтобы начать работать с базой данных со старым ODS, необходимо будет сделать резервную копию с помощью gbak под старым сервером и восстановить её с помощью gbak на Firebird 3.

Примечание

Поддержка устаревшего провайдера для работы с базами данных с версиями ODS 8 - 11.2 запланирована в будущих субрелизах.

Идентификатор реализации ID устарел

Александр Пешков

Идентификатор реализации ID в ODS базы данных устарел из за нового поля в заголовке базы данных, описывающего аппаратные детали, которые должны располагаться в определённом порядке при создании базы данных для совместимости с новой ODS.

Старый ID реализации заменен 4 байтовыми структурами, состоящими из аппаратного ID, ID операционной системы, ID компилятора и флагов совместимости. Три поля ID предназначены только для получения информации: ODS не зависит от них непосредственно, и они не проверяются при открытии базы данных.

Флаги совместимости проверяются на соответствие между базой данных и ядром сервера, открывающим ее. В настоящее время есть только один флаг - для порядка байтов. Как и ранее, Firebird не будет открывать базу данных с прямым порядком байтов (*little-endian*), которая была создана с обратным порядком байтов (*big-endian*), и наоборот.

Пример вывода gstat

```
# ./gstat -h employee
Database "/usr/opt/firebird/examples/empbuild/employee.fdb"
Database header page information:
.....
Implementation
HW=AMD/Intel/x64 little-endian OS=Linux CC=gcc
.....
```

Цель состоит в упрощении работы с портами Firebird для новых платформ.

Максимальный размер базы данных

Максимальный размер базы данных увеличен до 2^{32} страниц (ранее он составлял 2^{31} страниц). Новый предел составляет 16 ТБ | 32 ТБ | 64 ТБ в зависимости от размера страницы.

Максимальный размер страницы

Максимальный размер страницы остается 16 КБ (16384 байта).

Максимальное количество буферных страниц в кэше

Максимальное количество страниц, которые могут быть выделены для кэша базы данных, зависит от того, будет ли база данных от разрядности Firebird - 64 или 32 бита:

- 64 бита: $2^{31} - 1$ (2 147 483 647) страницы;
- 32 бита; 128 000 страниц, т.е. такое же, как и Firebird 2.5.

Увеличение лимита для количества Transaction ID

Дмитрий Еманов

Исторически в Firebird количество транзакций было ограничено

числом 2^{31} транзакцией, отсчитываемым с момента создания базы данных. После этого база данных будет недоступна до тех пор, пока не будет выполнено резервное копирование и восстановление, чтобы сбросить счетчик идентификаторов транзакций до нуля. В Firebird 3.0 первоначально количество транзакций было увеличено до 2^{32} , удвоив время работы базы данных без резервного копирования и восстановления.

Запрос на дальнейшее улучшение связан с тем, что этот предел ещё больше изменится, с введением 48-битных внутренних идентификаторов транзакций, которые публично (через API и таблицы MON\$) представлены в виде 64-разрядных номеров. Это делает новый предел примерно равным $2,8 \cdot 10^{14}$ транзакций. Позже он может быть увеличен до 2^{63} .

Внедрённое решение не имеет дополнительных накладных расходов на хранение, пока счетчики транзакций не превысят границу 2^{32} .

Увеличены лимиты для подключений и Statement ID

Идентификаторы соединения с базой данных (Attachment ID) и идентификаторы операторов (Statement ID) были изменены на 64-разрядные числа как внутри, так и снаружи через API и таблицы MON\$.

Системные таблицы

Новые системные таблицы

Таблица 4.1. Новые системные таблицы

Имя таблицы	Описание
RDB\$AUTH_MAPPING	Хранит проверки подлинности пользователей (<i>аутентификации</i>) и другие сопоставления безопасности
RDB\$PACKAGES	Заголовки пакетов SQL
RDB\$DB_CREATORS	Список пользователей, которым предоставлена право пользоваться оператором CREATE DATABASE при использовании указанной базы данных безопасности
SEC\$USERS	Виртуальная таблица для запроса

	локального списка пользователей
SEC\$USER_ATTRIBUTES	Виртуальная таблица хранения локальных атрибутов пользователей
SEC\$DB_CREATORS	SQL-интерфейс для доступа к списку в RDB\$DB_CREATORS; <i>select * from SEC\$DB_CREATORS</i>
SEC\$GLOBAL_AUTH_MAPPING	SQL-интерфейс для выборки пользователей из таблицы RDB\$AUTH_MAPPING, которые имеют доступ ко всем базам данных для указанной базы данных безопасности; <i>select * from SEC\$GLOBAL_AUTH_MAPPING</i> . Дополнительную информацию смотрите в разделе Доступ пользователей к объектам в главе Безопасность .

Изменения в системных таблицах

RDB\$SYSTEM_FLAG

Claudio Valderrama C.

([CORE-2787](#)) Флаг RDB\$SYSTEM_FLAG была сделан с атрибутом NOT NULL во всех таблицах.

RDB\$TYPES

Дмитрий Еманов

Добавлены пропущенные записи в таблицу RDB\$TYPES. Они описывают числовые значения для этих столбцов:

RDB\$PARAMETER_TYPE	(таблица RDB\$PROCEDURE_PARAMETERS)
RDB\$INDEX_INACTIVE	(таблица RDB\$INDICES)
RDB\$UNIQUE_FLAG	(таблица RDB\$INDICES)
RDB\$TRIGGER_INACTIVE	(таблица RDB\$TRIGGERS)

RDB\$GRANT_OPTION	(таблица RDB\$USER_PRIVILEGES)
RDB\$PAGE_TYPE	(таблица RDB\$PAGES)
RDB\$PRIVATE_FLAG	(таблицы RDB\$PROCEDURES и RDB\$FUNCTIONS)
RDB\$LEGACY_FLAG	(таблица RDB\$FUNCTIONS)
RDB\$DETERMINISTIC_FLAG	(таблица RDB\$FUNCTIONS)

Таблицы мониторинга

Дмитрий Еманов

Изменения в выводе адреса удалённого клиента

До Firebird 3.0 сетевой адрес удаленных клиентов записывался в поле *MON\$ATTACHMENTS.MON\$REMOTE_ADDRESS*, а также мог выводиться с помощью функции *RDB\$GET_CONTEXT('SYSTEM', 'CLIENT_ADDRESS')*. Для протокола TCP/IP (INET) он содержит TCP адрес, разделяемый точками. Для протокола Named Pipes (WNET, NetBeui) он всегда содержал NULL. Для протокола XNET он содержит имя локального узла.

Начиная с Firebird 3.0, сетевой адрес удаленного клиента содержит дополнительно номер порта TCP/IP удаленного клиента, разделенный косой чертой:

<IP address>/<port>

Номер порта подключения также может с помощью новой встроенной функции *RDB\$GET_CONTEXT('SYSTEM', 'CLIENT_PORT')*.

Имя узла также получить их нового столбца *MON\$REMOTE_HOST* таблицы *MON\$ATTACHMENTS*.

Предупреждение

Протокол WNET (Named Pipes/Netbeui) следует считать устаревшим. В следующих версиях Firebird его поддержка, вероятно, будет прекращена.

Для всех таблиц мониторинга были добавлены счётчики производительности (см. [CORE-4564](#)).

MON\$ATTACHMENTS

Стала доступной новая информация:

- Имя операционной системы пользователя (см. [CORE-3779](#));
- Версии протокола подключения и клиентской библиотеки (см. [CORE-2780](#));
- Имя хоста клиента (см. [CORE-2187](#));
- Метод аутентификации, используемый клиентов для подключения - *MON\$AUTH_METHOD* (см. [CORE-4222](#));
- *MON\$REMOTE_ADDRESS* теперь содержит строку *<IP>/<port>* с IP адресом и портом подключения клиента (см. [CORE-5028](#)).

MON\$DATABASE

- Добавлено поле *MON\$OWNER*, содержащее имя владельца базы данных (см. [CORE-4218](#));
- Добавлен флаг *MON\$SEC_DATABASE* типа используемой базы данных безопасности. Значение флага может принимать три значения — *Default/Self/Other*. (см. [CORE-4729](#)).

MON\$STATEMENTS

Предложение PLAN теперь тоже отображается (см. [CORE-2303](#)).

Расширения API (Application Programming Interface)

Новый открытый API (интерфейс программирования приложений) служит для замены устаревшего API в новых приложениях, особенно в объектно-ориентированных. Часть его интерфейса приведена в файле *Interfaces.h* в каталоге */include/firebird* корневого каталога установки Firebird.

Примечание

В ОС POSIX есть символическая ссылка, указывающая на каталог */usr/include/firebird/Interfaces.h* с интерфейсами API.

Новый открытый API также может использоваться внутри пользовательских подпрограмм (UDR — см. раздел [Внешние средства ядра сервера](#)) для обратных вызовов внутри ядра сервера, что позволяет UDR, например, осуществлять выборку данных или изменение данных в БД.

Основное различие между новым и устаревшим API заключается в том, что UDR могут осуществлять выборку данных и изменять их в том же контексте соединения или транзакции, что и запросы пользователя, который вызвал этот UDR. Теперь можно писать внешние триггеры и процедуры, а не только внешние функции (UDF).

Интерфейсы и новый объектно-ориентированный API

Александр Пешков

Firebird нуждался в новом модернизированном API по ряду веских причин:

- Наиболее веская причина - устаревший API основан на использовании 16-разрядных целых чисел, что вносит ряд ограничений — в размер сообщений, в длину операторов SQL, разбивка типа данных BLOB на мелкие части и т.д. И если в своё время 16-разрядный старый API, вероятно, соответствовал предъявляемым требованиям, то в настоящее время он не отвечает современным требованиям и его использование обходится слишком накладным. Тривиальным решением могло бы быть добавление новых функций, поддерживающих 32-разрядные переменные. Большим недостатком такого подхода является очевидная необходимость сохранить поддержку старого API, имея при этом пары функций с одинаковой функциональностью, но отличающихся размерностью целочисленных типов данных. Фактически была введена поддержка 64-разрядных счётчиков производительности, т.к. не имелось более изящного способа реализации.
- Другая менее очевидная, но важная причина, своими истоками происходит из эпохи, когда предшественник Firebird, InterBase, не поддерживал SQL. Для управления базами данных тогда использовался нестандартный язык запросов GDML. Запросы данных транспортировались между клиентом и сервером с использованием сообщений, форматы которых определялись во время компиляции запроса в BLR (*binary language representation - двоичное представление языка*). Операторы в SQL не содержат описания

формата сообщения, поэтому было принято решение объединить каждое сообщение с короткой BLR последовательностью, описывающей его формат.

Интерфейс ISC API также имеет слой XSQLDA над BLR. Решение с XSQLDA заключается в том, что оно инкапсулирует как местоположение данных, так и их формат, что позволяет изменять местоположение или формат (или оба) между вызовами выборки данных. Следовательно, потребность в BLR, передающемся в каждом вызове выборки и, несмотря на это, потенциальная возможность изменить формат данных между выборками, была повреждена на сетевом уровне ещё до появления Firebird.

Но для поддержки уровня XSQLDA, который распространяется поверх API на основе сообщений, API нижнего уровня также должно поддерживать формат BLR для каждого вызова функций.

Эта система, включающая обработку вызовов через несколько слоев, трудно расширять и при этом повысить производительность. К тому же SQLDA не прост в использовании. В силу этих причин и возникла необходимость исправить ситуацию.

- Другие многочисленные причины, возможно, менее востребованные для изменения API, заключались в необходимости улучшения вектора состояния и оптимизации загрузки динамических библиотек.

Интерфейсы также упрощают API и делают более удобными использование сообщений.

Отказ от COM объектов

Новые интерфейсы осознано выполнены несовместимыми с COM объектами, и причины этого связаны с будущим повышением производительности.

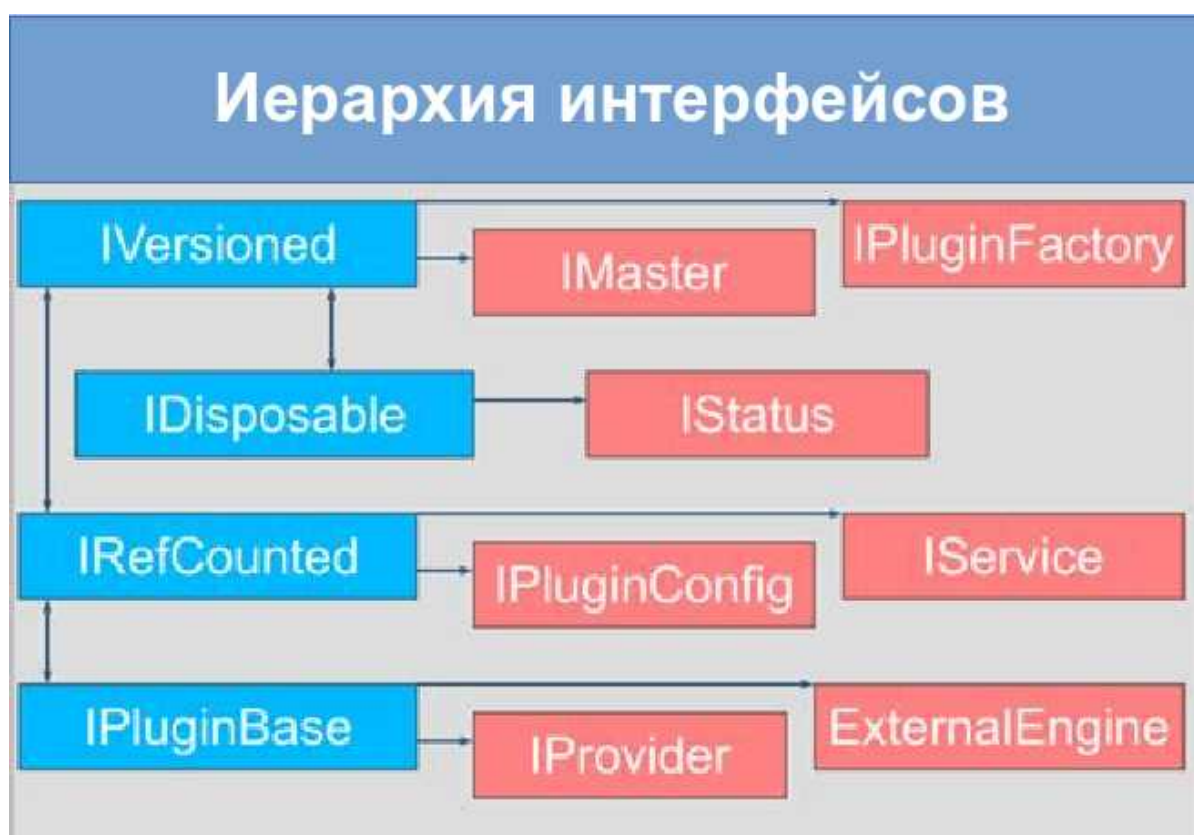
В центре архитектуры Providers (см. раздел [Провайдеры](#)) в Firebird 3.0 находится Y-клапан, который выполняет диспетчеризацию вызовов API и выбор корректного провайдера. Среди потенциальных провайдеров есть и более старые, с более старыми интерфейсами. При использовании COM нам нужно было бы вызывать метод IUnknown для каждого вызова (включая выборку данных), чтобы убедиться, что у провайдера действительно есть некоторый более новый метод API. Наряду с этим возникает большая вероятность разработки в будущем дополнительных функций API для оптимизации производительности. Решение на основе COM не очень хорошо работает в таких условиях.

Интерфейсы Firebird, в отличие от COM, поддерживают несколько версий. Версия интерфейса определяется общим количеством виртуальных

функций, которые она поддерживает, и хранится в виде указателя (как целое число) в начале таблицы виртуальных функций. Это позволяет очень быстро проверить версию интерфейса, так как для этого не требуется дополнительный вызов. То есть проверка указателя не имеет накладных расходов, в отличие от случая использования COM.

Иерархия интерфейсов

Подробное обсуждение всех функций, представленных всеми интерфейсами, выходит за рамки этого документа. Общая схема иерархии интерфейсов выглядит следующим образом:



Основой структуры является интерфейс **IVersioned**. Этот интерфейс позволяет обновлять версии. Множество интерфейсов, не требующих дополнительной настройки во время своей работы, базируются непосредственно на **IVersioned**. Одним из примеров этого является интерфейс **IMaster**. Другие включают в себя несколько интерфейсов обратного вызова, время работы которых должно соответствовать времени жизни объектов, из которых они используются для обратного вызова.

Два интерфейса непосредственно связаны с управлением продолжительностью работы - это **IDisposable** и **IRefCounted**. Последний

особенно активно используется при создании других интерфейсов: **IPlugin** со считываемой ссылкой, как и многие другие интерфейсы, используемые в качестве плагинов. Они включают в себя интерфейсы, которые осуществляют подключение к базе данных, управление транзакциями и операторами SQL.

Не все из интерфейсов нуждаются в дополнительных накладных расходах для вызова по ссылке. Например, **IMaster**, основной интерфейс, через который вызываются функции, доступные для остальной части API, имеет неограниченный срок работы по определению. Для других интерфейсов API время существования строго определено временем жизни родительского интерфейса; интерфейс **IStatus** не многопоточный. К достоинству интерфейсов с ограниченным временем работы относится простой способ их уничтожения (освобождения памяти), то есть функция `dispose ()`.

Каждый подключаемый модуль (плагин) имеет один и только один главный интерфейс - **IPlugin**, который отвечает за базовые функциональные возможности подключаемого модуля. На самом деле у многих плагинов есть только этот интерфейс, хотя это и не является обязательным требованием.

Наконец, есть **IProvider**, своего рода «основной» плагин в API Firebird. Интерфейс **IProvider** является наследником **IPlugin** и должен использоваться каждым провайдером. Для написания своего собственного провайдера Вы должны использовать интерфейс **IProvider**. Он используется также у-клапаном: реализация функции Y-клапана возвращается пользователю при вызове функции `getDispatcher ()` из главного интерфейса.

Интерфейс **IProvider** содержит функции, обеспечивающие соединение с базой данных (подключение и создание) или с диспетчером служб (*Services Manager*).

Интерфейсы: Вопросы (В) и Ответы (О)

В. Мы получаем доступ к новому API, используя **IMaster**. Но как получить доступ к самому **IMaster**?

О. Это делается с помощью только одной новой функции API `fb_get_master_interface()`. Она экспортируется из клиентской библиотеки `fbclient`. **IMaster** также передается как параметр для каждого плагина во время его регистрации в системе.

В. Отказ от использования основанных на COM интерфейсов объясняется использованием метода **IUnknown** и — из-за проблем с производительностью. Вместо этого нужно просто проверить версию интерфейса. Почему это работает быстрее, чем использование **IUnknown**?

О. Как уже говорилось ранее, нам не нужно выполнять виртуальные вызовы при проверке версии интерфейса. Принимая во внимание, что каждый виртуальный вызов означает сброс кэша ЦП, это важная разница, особенно для очень маленьких вызовов, таких как получение определенных свойств метаданных из интерфейса **IMetadata**.

Другие новшества в API

Другие новшества API поддерживают различные плагины, объявляя интерфейсы между ядром сервера и плагином. Помимо подключаемой аутентификации и подключаемого шифрования, Firebird 3 поддерживает «внешние двигатели», мосты между ядром сервера и средами исполнения, которые могут запускать UDR: собственный код, Java и другие (см. раздел [Внешние средства ядра сервера](#)). По большому счету они предназначены для использования решений сторонних поставщиков, а не для разработки клиентских приложений.

Для создания настраиваемых плагинов и мостов в коде плагина должен быть реализован соответствующий интерфейс API.

Улучшения в API

Отметим следующие улучшения API.

С помощью нового API сняты некоторые ограничения размера SQL

Дмитрий Еманов

Только при использовании нового API:

- Размер тела хранимой процедуры или триггера может превышать традиционный предел в 32 КБ. Теоретический предел, предоставляемый с помощью нового API, составляет 4 ГБ. На данный момент, в качестве меры безопасности, вводится жёстко запрограммированный предел в 10 МБ. Этот же предел в 10 МБ также применён ко всем запросам пользователей в DSQL.
- Общий размер всех входных или выходных параметров для хранимой процедуры или пользовательского запроса DSQL больше не ограничивается традиционным размером (64 КБ минус накладные расходы). Теоретический предел, предоставляемый с помощью нового API, составляет 4 ГБ.

Устаревшее API

Доступны следующие улучшения старого API.

Поддержка прокручиваемого курсора

Дмитрий Еманов

В PSQL курсором с возможностью прокрутки (см. [Поддержка курсора с возможностью прокрутки \(двунаправленного\)](#)) можно управлять непосредственно, чтобы гибко переместиться от текущей строки до любой другой строки вперед или назад. Также доступна поддержка API для создания прокручиваемых курсоров в приложениях с DSQL.

Использование прокручиваемого курсора

Результирующий набор данных должен быть открыт с явно указанным флагом **IStatement :: CURSOR_TYPE_SCROLLABLE**.

Метод FETCH

Для интерфейса **IResultSet** доступны следующие методы выборки:

```
int fetchNext(IStatus* status, void* message);  
// эквивалентно оператору FETCH NEXT FROM <cursor name>
```

Перемещает курсор с текущей позиции на следующую строку и возвращает её данные. Если курсор пустой или уже находится на последней строке, то возвращается константа **NO_DATA**.

```
int fetchPrior(IStatus* status, void* message);  
// эквивалентно оператору FETCH PRIOR FROM <cursor name>
```

Перемещает курсор с текущей позиции на предыдущую строку и возвращает её данные. Если курсор пустой или уже находится на первой строке, то возвращается константа **NO_DATA**.

```
int fetchFirst(IStatus* status, void* message);  
// эквивалентно оператору FETCH FIRST FROM <cursor name>
```

Перемещает курсор с текущей позиции на первую строку и возвращает её данные. Если курсор пустой, то возвращается константа **NO_DATA**.

```
int fetchLast(IStatus* status, void* message);  
// эквивалентно оператору FETCH LAST FROM <cursor name>
```

Перемещает курсор с текущей позиции на последнюю строку и возвращает её данные. Если курсор пустой, то возвращается константа **NO_DATA**.

```
int fetchAbsolute(IStatus* status, int position, void* message);  
// эквивалентно оператору FETCH ABSOLUTE <position> FROM  
// <cursor name>
```

Перемещает курсор с текущей позиции на строку, заданную параметром *<position>*, и возвращает данные из найденной строки. Если параметр *<position>* выходит за границы курсора, то возвращается константа **NO_DATA**.

```
int fetchRelative(IStatus* status, int offset, void* message);  
// эквивалентно оператору FETCH RELATIVE <offset> FROM  
// <cursor name>
```

Перемещает курсор с текущей позиции вперёд или назад на значение, заданное параметром *<offset>*, и возвращает данные из найденной строки. Если вычисленная позиция выходит за границы курсора, то возвращается константа **NO_DATA**.

Примечания

1. Когда опция прокрутки опущена, подразумевается NO SCROLL (т.е. курсор открывается для прокручивания только вперёд). Это означает, что может использоваться только вызов API `fetchNext()`. Другие методы выборки вернут ошибку.
2. Прокручиваемые курсоры внутренне реализованы как временный набор записей, т.е. потребляют ресурсы памяти/диска, поэтому эту функцию следует использовать только тогда, когда это действительно необходимо.

Поддержка SPB новой статистики, выводимой gbak

Владислав Хорсун

В подробный вывод утилиты gbak добавлена новая функция: опционный вывод статистики времени выполнения (см. раздел *Статистика в реальном времени с подробным выводом*). Эта функция полностью поддерживается в API-интерфейсах служб с новым элементом в SPB (Service Parameter Block - блок параметров служб)

```
#define isc_spb_bkp_stat 15
```

наряду с его синонимом

```
#define isc_spb_res_stat isc_spb_bkp_stat
```

Использование

```
isc_spb_bkp_stat, <len>, <string>
```

```
isc_spb_res_stat, <len>, <string>
```

где параметр <len> (2 байта) указывает длину следующего строкового параметра, а параметр <string> (1-4 байта) - это строка, состоящая из одного символа, для каждого элемента статистики.

Утилита *fbsvcmgr* также поддерживает новые флаги SPB.

Улучшенные отчеты об ошибках переполнения строк

Александр Пешков

В сообщении об ошибке переполнения строк включена ожидаемая и актуальная длина строки.

Дополнительная информация в отчетах об ошибке «Неверный тип страницы»

Александр Пешков

Более подробная информация в сообщении об ошибке «неправильный тип страницы» («*wrong page type*»), т.е. идентификация ожидаемых и

встречающихся типов страниц по имени вместо численного типа.

Новый параметр для вызова `isc_database_info()`

Владислав Хорсун

В функцию API `isc_database_info()` была добавлена возможность вывода количества свободных страниц в базе данных ([CORE-1538](#)).

Новый флаг служб для переопределения LINGER

Александр Пешков

API-интерфейс служб теперь включает в себя флаг `isc_spb_prp_nolinger`, например (в одной строке):

```
fbsvcmgr host:service_mgr user sysdba password xxx
```

```
action_properties dbname employee prp_nolinger
```

Для получения дополнительной информации о LINGER см. разделе *[«Задержка» закрытия базы данных для Superserver](#)*.

Новые форматы подключения для локальных клиентов Superserver в Windows

В предыдущих версиях Firebird для локальных клиентов, подключенных к Superserver на платформе Windows по протоколу XNET, был доступен серверный протокол, известный как «Локальный Windows». Типичная строка подключения выглядела так:

```
c:\Program Files\Firebird_2_5\examples\empbuild\employee.fdb
```

При такой строке подключения новый унифицированный сервер пытается загрузить встроенный сервер. Для режима SuperServer это теперь не работает - Вы получите сообщение об отказе «Файл используется другим процессом» («File is in use by another process»). Это не баг. Поскольку клиенты Superserver обмениваются ресурсами, другой сервер (в данном случае встроенный сервер) не может подключить клиента к той же базе данных, к которой подключен Superserver.

Но есть выход из этой ситуации. Протокол XNET может выполнять локальные клиентские сеансы для Superserver. Просто теперь Вам нужна более сложная строка подключения. Это можно сделать несколькими

способами:

- Использовать для подключения протокол XNET:

`xnet://alias-or-path-to-database`

т.е. используя путь к базе данных

`xnet://c:\Program Files\Firebird_3_0\examples\empbuild\employee.fdb`

или используя алиас

`xnet://employee`

- С помощью протокола TCP/IP (хост/порт):

`inet://host:port/alias-or-path-to-database`

- С помощью протокола TCP/IP с ограничением поиска по имени IPv4 (хост/порт):

`inet4://host:port/alias-or-path-to-database`

Эта опция добавлена в Firebird 3.0.1. См. также раздел [*IPv6V6Only*](#).

- С помощью протокола Named Pipes (NetBEUI) с использованием хост/порт:

`wnet://host:port/alias-or-path-to-database`

- Соединение с localhost через TCP/IP:

`inet://alias-or-path-to-database`

- Соединение с localhost через Named Pipes (NetBEUI):

`wnet://alias-or-path-to-database`

Выполнение проверки базы данных в режиме онлайн

Владислав Хорсун

Эта функция была перенесена из Firebird 2.5.4.

Валидация базы данных позволяет выполнять низкоуровневые проверки согласованности структур на диске и даже исправлять некоторые незначительные повреждения. Эту процедуру для обеспечения работоспособности любой ценной базы данных администратор базы данных должен периодически проводить.

Для этого требуется эксклюзивный доступ к базе данных: во время проверки запрещается использовать какой-либо параллельный доступ к БД. Иногда блокирование доступа пользователей может привести к серьёзной задержки в работе с базой данных, особенно если она является большой и сложной.

Новая функция онлайн проверки позволяет выполнять некоторые проверки целостности без эксклюзивного доступа.

Что может сделать онлайн проверка?

- Проверку некоторых или всех пользовательских таблиц в БД. *Системные таблицы не проверяются.*
- Проверку некоторых или всех индексов.

Другие проверки ODS, такие как Страницы заголовка (Header)\ PIP\ TIP\ Страницы генераторов (Generators pages) не выполняются.

Защита во время онлайн проверки

Пока таблица (и\или её индекс) проходит проверку, пользователи могут только читать данные из этой таблицы. Любая попытка изменить данные (INSERT\UPDATE\DELETE) будет ждать окончания проверки или, в зависимости от таймаута блокировки пользовательской транзакции, вернет ошибку выполнения операции по таймауту блокировки.

Пока идёт проверка любой вид сборки мусора в таблице или её индексах отключается:

- фоновая и кооперативная сборка мусора просто пропустят эту таблицу;
- принудительная сборка мусора, т.е. работа утилиты *"gfix -sweep db.fdb"* будет завершена с ошибкой.

После начала онлайн проверки таблицы для предотвращения одновременных модификаций её данных на неё накладываются два вида блокировок:

- блокировка таблицы в режиме PR (protected read - защищённое чтение);
- **Новая** блокировка сбора мусора в режиме PW (protected write - защищённая запись).

Обе блокировки используют определенный пользователем тайм-аут блокировки. Любому заблокированному запросу выдаётся сообщение об ошибке и что таблица пропущена.

Как только блокировки вступают в действие, таблица и её индексы будут проверены так же, как и при полной проверке. После завершения проверки блокировки снимаются и вся процедура повторяется для следующей таблицы.

Новая команда API служб: `isc_action_svc_validate`

Онлайновая проверка реализована как служба Firebird и доступна через интерфейс API служб. Таким образом, онлайновая проверка не может быть запущена из утилиты *gfix*.

Вызов включает следующие элементы:

Команда (Action):

`isc_action_svc_validate`

Параметры:

`isc_spb_dbname:`
имя базы данных, string - обязательный параметр

`isc_spb_val_tab_incl, isc_spb_val_tab_excl,`
`isc_spb_val_idx_incl, isc_spb_val_idx_excl :`
шаблоны для имен таблиц\индексов, string -
необязательный параметр

`isc_spb_val_lock_timeout:`
таймаут блокировки, integer - необязательный

параметр

Результат:

текстовые сообщения с ходом процесса онлайн проверки

Интерактивное использование `isc_action_svc_validate`

Утилита *fbsvcmgr* полностью поддерживает новый сервис. Синтаксис:

```
fbsvcmgr [host:]service_mgr [user <...>] [password <...>]  
    action_validate dbname <имя файла>  
    [val_tab_incl <шаблон>]  
    [val_tab_excl <шаблон>]  
    [val_idx_incl <шаблон>]  
    [val_idx_excl <шаблон>]  
    [val_lock_timeout <number>]
```

где

`val_tab_incl` - шаблон для имён таблиц, включенных в проверку

`val_tab_excl` - шаблон для имён таблиц, исключённых из
проверки

`val_idx_incl` - шаблон для имён индексов, включенных в проверку,
по умолчанию%, т. е. все индексы

`val_idx_excl` - шаблон для имён индексов, исключённых из
проверки

`val_lock_timeout` - тайм-аут блокировки, используется для
блокировки таблицы при проверке, в секундах
по умолчанию - 10 секунд; 0 — не ждать;
-1 - бесконечное ожидание

Замечания по использованию

- Шаблоны - это регулярные выражения, обрабатываемые по тем же правилами, что и выражения типа «SIMILAR TO»;
- Независимо от диалекта базы данных все шаблоны чувствительны к регистру;
- Если пропущен шаблон для таблиц, то будут проверены все пользовательские таблицы;
- Если пропущен шаблон для индексов, то будут проверены все индексы назначенных таблиц;
- Системные таблицы не проверяются;
- Для указания списка таблиц или индексов:
 1. Разделяйте имена с символом '|';
 2. Не добавляйте пробелы: строка TAB1 | TAB2 является ошибочной;
 3. Заключите весь список в двойные кавычки, чтобы интерпретатор команд его правильно обработал.

Примеры

1. Проверка всех таблиц базы данных 'c:\db.fdb', имена которых начинаются с символа 'A'. Индексы не проверяются. Блокировка ожидания не выполняется.

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

2. Проверка таблиц TAB1 и TAB2 и всех индексов в них. Таймаут блокировка составляет 10 секунд (значение по умолчанию).

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

3. Поведение по умолчанию параметров val_XXX: проверка всех пользовательских таблиц и всех их индексов в базе данных 'c:\db.fdb', Блокировка ожидания имеет значение по умолчанию 10 секунд.

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

Улучшение кода

Александр Пешков

([**CORE-4387**](#)) Теперь функции *IStatement::execute()* и *IAttachment::execute()* возвращают указатель ошибки на старый интерфейс транзакции.

Глава 5

Зарезервированные слова и изменения

Новые ключевые слова в Firebird 3.0

Зарезервированные

Ключевые слова, отмеченные звездочками (*), ранее не были зарезервированы.

BOOLEAN	REGR_SLOPE
CORR	REGR_SXX
COVAR_POP	REGR_SXY
COVAR_SAMP	REGR_SYY
DELETING *	RETURN
DETERMINISTIC	ROW
FALSE	SCROLL
INSERTING *	SQLSTATE
OFFSET	STDDEV_POP
OVER	STDDEV_SAMP
RDB\$RECORD_VERSION	TRUE
REGR_AVGX	UNKNOWN
REGR_AVGY	UPDATING *
REGR_COUNT	VAR_POP
REGR_INTERCEPT	VAR_SAMP
REGR_R2	

Не зарезервированные

ABSOLUTE	LEAD
ACOSH	LINGER
ASINH	NAME
ATANH	NTH_VALUE
BODY	PACKAGE
CONTINUE	PARTITION
DDL	PLUGIN
DECRYPT	PRIOR
DENSE_RANK	RANK
ENCRYPT	RELATIVE
ENGINE	ROW_NUMBER
FIRST_VALUE	SERVERWIDE
IDENTITY	TAGS
INCREMENT	TRUSTED
LAST_VALUE	USAGE
LAG	

Глава 6

Изменения и дополнения в параметрах конфигурации

Файл *aliases.conf* переименован в *databases.conf*. Можно просто переименовать старый *aliases.conf* из предыдущей версии и новое ядро сервера будет продолжать использовать его по-прежнему. Однако *databases.conf* может теперь включать некоторую дополнительную конфигурационную информацию отдельно для каждой из баз данных.

Область действия параметров

Некоторые параметры отмечены как настраиваемые для каждой базы данных (*per-database*) или для каждого соединения (*per-connection*).

- Конфигурация для баз данных находится в файле *databases.conf*;
- Конфигурация подключения в первую очередь предназначена для использования в клиентских приложениях и выполняется с использованием параметра DPB *isc_dpb_config* или, для служб, параметра SPB *isc_spb_config*;
- В случае встроенного сервера (*embedded*) DPB может использоваться для настройки поведения клиентов при первом подключении к в каждой базе данных по отдельности.

Замена параметров макросами

Ряд предопределенных макросов (синтаксис *\$(имя)*) доступны для использования в файлах конфигурации для замены имени каталога:

\$(root)

Корневой каталог установки Firebird

\$(install)

Директория установки Firebird. Изначально *\$(root)* и *\$(install)* одинаковы. Можно переопределить *\$(root)*, установив или изменив переменную среды *FIREBIRD*, в этом случае она будет отличаться от *\$(install)*.

`$(this)`

Директория, в которой находится текущий конфигурационный файл

`$(dir_conf)`

Директория, в которой находятся файлы `firebird.conf` и `databases.conf`

`$(dir_secdb)`

Директория, в которой находится файл базы данных безопасности

`$(dir_plugins)`

Директория, в которой находятся плагины

`$(dir_udf)`

Директория, в которой по умолчанию находятся UDF

`$(dir_sample)`

Директория, в которой находятся примеры

`$(dir_sampledb)`

Директория, в которой находится пример базы данных (`employee.fdb`)

`$(dir_intl)`

Директория, в которой находятся международные модули

`$(dir_msg)`

Директория, в которой находится файл с сообщениями (`firebird.msg`). Обычно `$(dir_msg)` должен быть таким же, как и `$(root)`, но его можно переопределить изменив переменную среды `FIREBIRD_MSG`

Подсказка

Вы можете посмотреть использование некоторых макросов в файле `database.conf`.

Примечания

При установке Firebird по умолчанию макросы `$(dir_conf)` и `$(dir_secdb)` обычно будут совпадать с `$(root)` и `$(install)`.

`$(dir_plugins)`, `$(dir_udf)`, `$(dir_sample)`, `$(dir_sampledb)` и `$(dir_intl)` являются предопределенными подкаталогами внутри `$(root)`.

Соглашения о сборке не являются «правилами», которые применяются при каждом распространении Firebird. Например, специфичные дистрибутивы пакета для Linux, помещают компоненты Firebird в директориях, которые соответствуют их собственным стандартам. Как иллюстрация: пользовательские двоичные файлы, такие как *isql* могли бы быть расположены в каталоге `/usr/bin`, двоичные файлы сервера в `/usr/sbin`, конфигурационные файлы в `/etc/firebird.d` и т.д. Очевидно, что `$(root)` в этом случае не имеет смысла, даже если макросы `$(dir_something)` по-прежнему указывают на фактические каталоги.

Подключение файлов конфигурации

Один файл конфигурации может быть включён в другой с помощью директивы “include”, т. е.:

```
include some_file.conf
```

Относительный путь расположения главного файла конфигурации рассматривается как относительный путь и для подключаемого файла конфигурации. Так, если в нашем примере главный файл находится по относительному пути `/opt/config/master.conf`, то наш подключаемый файл находится в каталоге `/opt/config/some_file.conf`.

Подстановочные знаки

Стандартные подстановочные знаки `*` и `?` могут использоваться в директиве `include`, чтобы подключить все соответствующие файлы в произвольном порядке.

Выражения для значений параметров

Ранее значения параметров с типом данных байт задавались по умолчанию как `integer` и определяли количество байтов. Теперь вы можете задавать их значение, если это необходимо, в килобайтах, мегабайтах или гигабайтах, добавив К, М или G (не чувствительны к регистру). Например, 24М читается как 25165824 (24 * 1024 * 1024).

Булевские значения задавались как ненулевые (`true`) | нулевые (`false`) по умолчанию, но теперь вы можете использовать заключенные в двойные кавычки строки «у», «yes» или «true» вместо ненулевой цифры.

Конфигурация “Для каждой базы данных” (“Per-database”)

Пользовательская конфигурация на уровне базы данных задаётся с помощью записей в файле `databases.conf`.

Формат записей конфигурации

Формат для указания псевдонима базы данных в файле `aliases.conf`:

`aliasname = /absolute/path/to/database_file`

Если вы не добавляете какие-либо дополнительные параметры конфигурации для псевдонима конкретной базы данных, то формат записей остаётся старым, например:

`emp = c:\Program Files\examples\empbuild\employee.fdb`

**** или ****

`emp = /opt/firebird/examples/empbuild/employee.fdb`

**** или ****

`emp = $(dir_sampleDb)/employee.fdb`

Немного более сложный формат используется для случаев, когда определенные не глобальные параметры должны заданы для конкретной базы данных. Запись для базы данных начинается с объявления псевдонима, как и ранее. Специфические директивы для базы данных перечисляются ниже, в фигурных скобках.

```
#
# директивы для MYBIGDB
MYBIGDB = opt/databases/mybigdb.fdb
{
    # Задаём размер памяти для блокировок
    LockMemSize = 32M

    # Задаём размер хеш-таблицы
    LockHashSlots = 19927
}
```

Доступные параметры

Следующие параметры могут быть скопированы и затем вставлены в `database.conf` и использованы для их переопределения для конкретных баз данных.

Таблица 6.1. Параметры, доступные в `databases.conf`

Для ядра сервера		
DatabaseGrowthIncrement	DeadlockTimeout	DefaultDbCachePages
EventMemSize	FileSystemCacheThreshold	ExternalFileAccess
GCPolicy	LockAcquireSpins	LockHashSlots
LockMemSize	MaxUnflushedWrites	MaxUnflushedWriteTime
SecurityDatabase (см. <i>База безопасности SecurityDatabase</i>)	<i>UserManager</i>	
<i>WireCompression</i>	<i>WireCrypt</i>	<i>WireCryptPlugin</i>
Для клиента	Некоторые параметры можно настроить при подключении клиента через DPB/SPB, в качестве альтернативы их настройке в <code>database.conf</code> . Чтобы понять эти различия, пожалуйста, прочитайте раздел <i>Область действия параметров</i> в начале этой главы.	
AuthClient (см. <i>AuthServer и AuthClient</i>)	Providers (см. <i>Провайдеры</i>)	
Следующие параметры могут быть настроены ТОЛЬКО через DPB / SPB		
ConnectionTimeout	DummyPacketInterval	IpcName

RemoteAuxPort	RemotePipeName	RemoteServiceName
RemoteServicePort	TCPNoNagle	

Новые параметры

Добавлены новые параметры в `firebird.conf`:

База данных безопасности - SecurityDatabase

Определяет имя и расположение базы данных безопасности, в которой хранятся имена пользователей и пароли, используемые сервером для проверки удаленных подключений. Путь к ней определяется в файле `firebird.conf`, по умолчанию `$(root)/security3.fdb`. Он может быть переопределен для конкретной базы данных в файле конфигурации `database.conf`.

AuthServer и AuthClient

Два параметра определяют, какие методы проверки подлинности могут использоваться сервером и клиентом при подключении по сети. Разрешенные методы перечисляются в виде строковых символов (слов), разделенных запятыми, точками с запятой или пробелами.

- Безопасные удаленные пароли (*Srp*), используя соответствующий операционной системе плагин (`libSrp.so` | `Srp.dll` | `Srp.dylib`);
- Только для ОС Windows: когда учетные данные для подключения не указаны используется интерфейс поставщика поддержки безопасности (*Sspi*);
- Для общения со старыми версиями сервера клиентские приложения могут использовать устаревшую аутентификацию (*Legacy_Auth*).

Для **AuthServer** перечисляются методы `Srp` и `Win_Sspi`; Для **AuthClient** - `Srp`, `Win_Sspi` и `Legacy_Auth`.

Для отключения ненужного метода удалите маркер комментария (`#`) и удалите метод из списка.

Оба параметра могут использоваться в файле `databases.conf`. Оба параметра могут использоваться как в DPB, так и в SPB для специфичной конфигурации подключения.

WireCrypt

Параметр определяет, следует ли шифровать сетевое соединение. Он имеет три возможных значения: **Обязательно** | **Включено** | **Отключено** (**Required** | **Enabled** | **Disabled**). Значение по умолчанию установлено следующим образом: для подключений, поступающих на сервер, требуется шифрование (**Required**), и оно включается для подключений, отправляемых на сервер (**Enabled**).

Для получения доступа к серверу с использованием старой клиентской библиотеки (т.е. требуется отключение шифрования), параметр **WireCrypt** в файле конфигурации сервера должен иметь значение «**Enabled**» или «**Disabled**», чтобы переопределить значение по умолчанию «**Required**».

Правила просты: если одна сторона имеет параметр **WireCrypt** = **Required**, а на другой установлено имеет значение **Disabled**, то сторона с **WireCrypt** = **Required** отклоняет соединение.

В случае отсутствия подключаемого модуля **WireCrypt** или *ключа шифрования*, когда канал должен быть зашифрован, соединения также будут отклоняться.

Во всех остальных случаях соединение устанавливается без шифрования, если хотя бы одна сторона имеет **WireCrypt** = **Disabled**. В других случаях устанавливается зашифрованное соединение.

UserManager

Параметр задаёт тип подключаемого модуля, который будет работать с базой данных безопасности. Это может быть список с пробелами, запятыми или точкой с запятой в качестве разделителей: используется первый плагин из списка.

Плагин по умолчанию **Srp** (**libSrp.s0** | **Srp.dll** | **Srp.dylib**).

Параметр **UserManager** может задаваться в файле **database.conf** для конкретной базы данных с целью его переопределения.

TracePlugin

Задаёт подключаемый модуль, используемый средством трассировки Firebird для отправки данных трассировки в клиентское приложение или данные аудита в файл логирования.

Плагин по умолчанию **fbtrace** (**libfbtrace.s0** | **fbtrace.dll** | **fbtrace.dylib**).

WireCryptPlugin

Определяет плагин, используемый для шифрования и дешифрования данных, передаваемых по сети.

По умолчанию Arc4, подразумевает использование подключаемого модуля Alleged RC4. Сконфигурированный плагин, который требует ключа, сгенерированного настроенным подключаемым модулем проверки подлинности, может быть переопределен в API для конкретного соединения через DPB или SPB.

Совет

Информацию о настройке подключаемых модулей см. в разделе *Настройка плагинов* в главе 3.

KeyHolderPlugin

Этот параметр должен предоставлять некоторую форму временного хранения для ключей шифрования базы данных. Пока плагин по умолчанию не реализован, но демонстрационный плагин для ОС Linux, названный libCryptKeyHolder_example.so, может быть найден в поддиректории /plugins/.

Providers

Список разрешенных провайдеров для доступа к базам данных, которые обсуждаются в главе 3 *Изменения в ядре Firebird*.

ServerMode

Определяет режим работы сервера («модель сервера»). Подробности приведены в главе 3 *Изменения в ядре Firebird*.

RemoteAccess

Параметр в firebird.conf и databases.conf, обеспечивающий эффективную и настраиваемую замену жестко закодированных правил, ограничивающих доступ к security3.fdb. Его также можно использовать для настройки ограниченного удаленного доступа к любой другой базе данных,

включая базы данных безопасности (отличных от умолчательной security3.fdb).

По умолчанию RemoteAccess включен для всех баз данных, кроме базы данных безопасности. Если вы намерены использовать более одной базы данных безопасности, то настоятельно рекомендуется отключить удаленный доступ к ней (к ним) в файле database.conf.

Для ужесточения безопасности, для всех баз данных сервера, Вы можете установить **RemoteAccess** в **false** в файле firebird.conf. В этом случае для разрешения доступа к определенным базам данных использовать записи о них в файле database.conf.

Параметр RemoteAccess является логическим. Он может задаваться как true/false, 1/0, либо Yes/No.

WireCompression

Александр Пешков

Параметр в firebird.conf или databases.conf, включающий или отключающий сжатие данных при передаче их по сети на глобальном уровне или для отдельных баз данных.

По умолчанию отключено (**False**). Для использования сжатия данных настройки должны быть корректными как на сервере, так и на клиенте:

- Для включения сжатия данных на стороне сервера, в firebird.conf и/или databases.conf измените значение Wirecompression на True;
- Чтобы активировать Wirecompression от стороны клиента, передайте надлежащий тег в элементе конфигурации вызова SPB или DPB:

```
isc_dbp_config/isc_sbp_config <string-length> "WireCompression=true"
```

- Версии сервера и клиента должны быть Firebird 3 или выше (протокол >= 13).

См. в Трекере [**CORE-733**](#).

IPv6V6Only

Michael Kubecek

Параметр используется только в firebird.conf. (TCP порты должны быть созданы до того, как будет установлено какое-либо соединение).

Firebird 3 поддерживает соединения IPv6, как на стороне клиента, так и на стороне сервера.

Сервер

По умолчанию сервер Firebird прослушивает нулевой адрес IPv6 (: :) и принимает все входящие соединения, будь то IPv4 или IPv6, а параметр `IPv6V6Only` имеет значение **false** (= 0). Если для него установить значение **true**, то сервер, все еще прослушивающий неявно или явно на нулевом IPv6-адресе, будет принимать только соединения IPv6.

Примечания

Другой адрес прослушивания, будь то IPv4 или IPv6, можно установить с помощью параметра `RemoteBindAddress`. Если используется IPv4-адрес или ненулевой адрес IPv6, директива `IPv6V6Only` не действует.

На платформах POSIX в классическом режиме параметры `RemoteBindAddress`, `RemoteServicePort` и `RemoteServiceName` игнорируются `fbserver`, так как сокет прослушивания настроен на (x) `inetd`. Адрес прослушивания и/или порт необходимо установить в (x) `inetd`.

Параметр `IPv6V6Only` является логическим. Он может задаваться как `true/false`, `1/0`, либо `Yes/No`.

Клиент

Стандартная текстовая форма адреса IPv6 использует символ двоеточия для разделения четырех групп цифр. В строке подключения адрес IPv6 должен быть заключен в квадратные скобки, чтобы устранить двусмысленность с использованием двоеточия в качестве разделителя между IP адресом хоста и пути к базе данных. Например:

```
connect '[2014:1234::5]:test';
```

```
connect '[2014:1234::5]/3049:/srv/firebird/test.fdb';
```

Примечание

Для непротиворечивости квадратные скобки могут дополнительно использоваться вокруг адреса IPv4 или доменного имени.

Если в строке подключения используется доменное имя, все адреса (IPv4 и IPv6) проверяются в порядке, возвращаемом преобразователем, до тех пор, пока соединение не будет установлено. Если все попытки не удались, то клиент не сможет подключиться.

Изменённые и улучшенные параметры

Были изменены или улучшены следующие параметры:

ExternalFileAccess

Записи в списке «Restrict» параметра **ExternalFileAccess** могут использоваться для управления именами файлов с относительными путями.

Записи в списке «Restrict» уже использовались для манипулирования именами файлов без использования пути. Например, используя

```
ExternalFileAccess = /opt/extern
```

и следующую последовательность команд

```
SQL> create table qq external file 'zz' (x int);  
SQL> insert into qq values(1);  
SQL> commit;
```

будет создан файл `/opt/extern/zz`.

Но если выполнить что-то подобное этому

```
create table qq external file 'dir/zz' (x int);
```

результатом будет ошибка отказа в доступе к файлу `/opt/firebird/bin/dir/zz`.

Улучшение устраняет этот пробел, изменяя имя файла в соответствии со значением параметра и, при необходимости, создавая отсутствующие компоненты пути, такие как «dir» в приведенном выше примере.

Удалённые или устаревшие параметры

Следующие параметры были удалены или устарели:

RootDirectory

В более старых версиях этот параметр предоставлял избыточный вариант для записи пути файловой системы к «корневым» файлам Firebird (firebird.conf, базе данных безопасности и т.д.).

LegacyHash

Этот параметр предназначался для использования старой базы данных безопасности security.fdb из установок Firebird 1.x после того, как он подвергся сценарию обновления и чтобы включить или отключить использование устаревшего алгоритма шифрования хэша DES. Он больше не поддерживается.

OldSetClauseSemantics

Этот параметр включал временную поддержку ошибки реализации в определенных последовательностях предложений SET в версиях Firebird до v.2.5. Он больше не доступен.

OldColumnNaming

Этот параметр временно активировал поддержку устаревших кодов для старой ошибки InterBase/Firebird 1.0, которая генерировала неименованные столбцы для вычисляемых выходных данных, которые не были явно заданы псевдонимами в спецификации SELECT. Он больше не доступен.

LockGrantOrder

Этот параметр используется, чтобы позволить менеджеру блокировок Firebird эмулировать поведение выделения блокировок InterBase v3.3, посредством чего блокировки предоставлялись без определенного порядка и так быстро, как только они стали доступны, а не согласно нормальному порядку (сначала прибывший, сначала подаваемый). Эта устаревшая опция больше не поддерживается.

Устаревшие настройки приоритетов Windows

Параметры UsePriorityScheduler, PrioritySwitchDelay и PriorityBoost, которые изначально относились к устаревшим процессорам на устаревших версиях Windows, больше не поддерживаются.

Глава 7

Безопасность

Улучшения безопасности в Firebird 3 включают:

Расположение списка пользователей

Александр Пешков

CORE-685

Firebird теперь поддерживает неограниченное количество баз данных безопасности. Любая база данных может выступать в качестве базы данных безопасности и может представлять собой базу данных безопасности для себя.

Используйте файл `database.conf` для настройки базы данных безопасности, отличной от задаваемой по умолчанию. Пример, приведённый ниже, показывает настройку файла `/mnt/storage/private.security.fdb` как базу данных безопасности для первой и второй баз данных:

```
first = /mnt/storage/first.fdb
{
    SecurityDatabase = /mnt/storage/private.security.fdb
}

second = /mnt/storage/second.fdb
{
    SecurityDatabase = /mnt/storage/private.security.fdb
}
```

Здесь мы используем третью базу данных в качестве собственной базы данных безопасности:

```
third = /mnt/storage/third.fdb
{
    SecurityDatabase = third
}
```


Примечание

Значение параметра SecurityDatabase (см. [База данных безопасности - SecurityDatabase](#)) может быть псевдонимом базы данных или фактическим путем к базе данных.

Создание альтернативной базы данных безопасности

Чтобы начать использовать отдельную базу данных безопасности, отличную от задаваемой умолчанию, надо её создать, если она ещё не существует. Используется встроенное соединение isql:

```
> isql -user sysdba
```

```
SQL> create database '/mnt/storage/private.security.fdb';
```

Теперь подключитесь к любой базе данных, которая будет обслуживаться базой данных безопасности, которую вы сейчас готовите, чтобы создать своего пользователя SYSDBA:

```
SQL>connect first;
```

```
SQL>create user sysdba  
        password 'sysdba-in-private-security-password';
```

```
SQL>commit;
```

```
SQL>exit;
```

Шифрование базы данных

Александр Пешков

CORE-657

Начиная с Firebird 3 появилась возможность шифрования данных, хранящихся в базе данных. Но шифруется не весь файл базы данных, а только данные, индексы и страницы BLOB.

Чтобы обеспечить шифрование базы данных, Вам необходимо

получить или написать подключаемый модуль (плагин) `crypt`.

Примечание

В примере для плагина `crypt` в поддиректории `/examples/dbcrypt` фактически не выполняется никакого шифрования, это всего лишь образец того, как писать плагин.

Ключ шифрования

Основной проблемой шифрования базы данных является хранение секретного ключа. Firebird предоставляет помощника для передачи этого ключа от клиента, но это не означает, что хранение ключа на клиенте является наилучшим способом: это не более чем возможная альтернатива. Очень плохим вариантом является хранение ключа на том же диске, что и база данных.

Задачи шифрования

Чтобы эффективно разделить шифрование и доступ к ключам, подключаемый модуль `crypt` для базы данных разбит на две части: само шифрование и хранение секретного ключа. Это может быть эффективным подходом для сторонних плагинов, когда вы хотите использовать хороший алгоритм шифрования, но у вас есть свой секретный способ хранения ключа.

Шифрование базы данных

После того, как вы определились с плагином `crypt` и ключом, Вы можете зашифровать свою базу данных с помощью следующей команды:

```
ALTER DATABASE ENCRYPT WITH <PLUGIN_NAME>  
[KEY <key-name>]
```

Необязательный аргумент `KEY` позволяет передать имя ключа в команде, если этого требует плагин.

Шифрование начнется сразу после того, как этот оператор завершится и будет выполнено в фоновом режиме. При шифровании нормальная активность базы данных не нарушается.

Мониторинг шифрования

Во время шифрования прогресс можно отслеживать, используя поле `MON$CRYPT_PAGE` в псевдотаблице `MON$DATABASE`. Например:

```
select MON$CRYPT_PAGE * 100/MON$PAGES  
from MON$DATABASE
```

В данном примере будет выведен процент выполненного шифрования.

Вы также можете просмотреть страницу заголовка базы данных, используя вызов утилиты ***gstat -e*** (и отслеживать динамику процесса шифрования периодическими вызовами ***gstat -e***). Если база данных была зашифрована, то ***gstat -h*** также может предоставлять ограниченную информацию о состоянии шифрования.

Расшифровка базы данных

Для дешифрования базы данных выполните команду:

```
ALTER DATABASE DECRYPT
```

Для ОС Linux пример плагина с именем `libDbCrypt_example.so` можно найти в подкаталоге `/plugins/`.

Новый метод аутентификации в Firebird 3.0

Александр Пешков

Весь код, связанный с аутентификацией (проверкой подлинности), поддерживается плагином. Несмотря на то, что Firebird выполняет универсальную работу, например извлекает данные проверки подлинности из сетевого сообщения или помещает их в такие сообщения, все действия, связанные с вычислением хэшей, хранением данных в базах данных или в других местах, используя определенные простые числа и так далее, выполняются плагинами.

Firebird 3 имеет новый метод аутентификации пользователей, реализованный в качестве плагина по умолчанию: безопасный протокол удаленного доступа (SRP). Цитата из [Википедии](#):

“Протокол SRP создает большой закрытый ключ, разделяемый между

двумя сторонами способом, аналогичным обмену ключами Diffie-Hellman, а затем подтверждает обеим сторонам, что эти два ключа идентичны, и что обе стороны имеют пароль пользователя. В случаях, когда требуются шифрованные сообщения, а также аутентификация, протокол SRP более безопасен, чем альтернативный протокол SSH, и быстрее, чем использование обмена ключами Diffie-Hellman с подписанными сообщениями. Он также не зависит от третьих сторон, в отличие от Kerberos.”

SSH для работы требует предварительного обмена ключами между сервером и клиентом при размещении открытого ключа на сервере. SRP не нуждается в этом. Все потребности клиента - это логин и пароль. Все обмены происходят после установления соединения.

Более того, SRP устойчив к атакам «человек-в-середине» («man-in-the-middle»).

Важно

Использование нового метода проверки подлинности несовместимо со старыми базами данных безопасности и паролями. Однако, доступна процедура обновления для миграции пользователей из базы данных security2.fdb Firebird 2.x. Инструкции приведены в разделе [Обновление базы данных безопасности версии 2.x](#) в главе 12 [Проблемы совместимости](#).

Использование старой базы данных безопасности может поддерживаться с помощью плагина проверки подлинности Legacy_Auth (см. [Устаревшая аутентификация](#)), но это уберет все преимущества безопасности Firebird 3.

Клиент Firebird 3 с настройкой по умолчанию имеет возможность общаться со старыми серверами.

[Поддержка SSL/TLS](#)

[CORE-3251](#)

Итак, ответ на вопрос «Использует ли Firebird SSL/TLS для проверки пароля?» это и «да и нет». Ответ «Нет» возникает, потому что по умолчанию SSL не используется. Это связано с незначительной несовместимостью лицензирования между Firebird и OpenSSL, самой популярной реализацией SSL.

«Да», потому что любой свободно может написать подключаемый модуль аутентификации, который использует SSL и TLS.

Увеличенная длина пароля

CORE-1898

Реализация SRP в нашем плагине увеличила длину пароля с 8 до 20 символов или более. Из-за использования SHA1 для хэшей максимальная безопасность предоставляется для паролей длиной до 20 символов. Более длительные пароли могут использоваться без ограничений, но есть удаленная возможность хэш-коллизий между паролями, которые отличаются за пределами 20-го байта. Просто имейте в виду, что любой пароль длиной более 20 символов может использовать один и тот же хэш с более коротким паролем, поэтому теоретически они могут быть атакованы с использованием грубой силы.

Вы можете довольно легко написать свой пользовательский плагин SRP, используя алгоритм хеширования, который гарантировал бы уникальные хэши для более длинных паролей.

Замечание

Увеличение длины означает, что пароль SYSDBA по умолчанию в ОС Windows и MacOS - это теперь полная строка «masterkey» (9 символов), а не «masterke» (8 символов), как в старых версиях!

Плагин для аутентификации

Плагин Authentication состоит из трех частей:

- **Клиент** - готовит данные на клиенте для отправки на сервер;
- **Сервер** - проверяет правильность пароля;
- **Менеджер пользователей** - добавляет, изменяет и удаляет пользователей на сервере. Он не требуется, если используется какой-либо внешний метод аутентификации, например, доверенная аутентификация Windows.

На самом деле все три части являются отдельными плагинами, которые следует настраивать отдельно в файле `firebird.conf`. Рассмотрим пример

настройки сервера для приема соединений от старых клиентов. Значение по умолчанию:

```
AuthServer = Srp, Win_Sspi
```

```
UserManager = Srp
```

Чтобы разрешить доступ к старым клиентам, необходимо изменить значение параметра **AuthServer**:

```
AuthServer = Srp, Win_Sspi, Legacy_Auth
```

Если Вы также хотите управлять списком пользователей в старом формате, то должны добавить:

```
UserManager = Legacy_UserManager
```

Менеджеры пользователей

В файле **firebird.conf** перечислены несколько пользовательских менеджеров. По умолчанию используется первый из списка менеджеров пользователей. Выборка данных из таблицы из **sec\$users** могла показать бы следующий результат:

```
SQL> select SEC$USER_NAME, SEC$PLUGIN from sec$users;
```

SEC\$USER_NAME	SEC\$PLUGIN
=====	=====
SYSDBA	Legacy_UserManager
SYSDBA	Srp
QA_USER1	Srp
QA_USER2	Srp
QA_USER3	Srp
QA_USER4	Srp
QA_USER5	Srp
GUEST	Srp
SHUT1	Srp
SHUT2	Srp
QATEST	Srp

В таком списке могут быть два пользователя с именем SYSDBA, т.к.

каждый менеджер пользователей имеет свой собственный логин SYSDBA.

Примечания

- Все команды управления пользователями могут содержать предложение **USING PLUGIN**, целью которого является включение выбора конкретного плагина UserManager из списка в файле **firebird.conf**;
- Менеджером пользователей по умолчанию в **firebird.conf** является **Srp**. Если Вам нужно управлять устаревшими логинами, установите его в **Legacy_UserManager**, например **UserManager = Legacy_UserManager, Srp**;
- **UserManager** можно настроить на уровне базы данных, в файле **database.conf**.

Шифрование сетевого подключения

Александр Пешков

CORE-672

Весь сетевой трафик в Firebird 3 может быть дополнительно зашифрован. Как и при аутентификации, для шифрования и дешифрования сетевого трафика используются плагины.

Плагин по умолчанию - **arc4** (предположительно **RC4**). Вы также можете написать собственный плагин для сетевого шифрования данных, но при этом Вы должны обязательно использовать актуальную версию библиотеки **fbclient** Firebird 3.

Ключ шифрования сессии

Проблема при использовании симметричного шифра состоит в том, где получить ключ для него. Firebird предполагает, что такой ключ, также называемый секретным ключом сеанса, создается плагином аутентификации на этапе установления соединения. Плагин **SRP** полностью удовлетворяет этому требованию, создавая криптографически сильный сеансовый ключ.

Совет

Если Вы хотите использовать шифрование с плагином

аутентификации, который не обеспечивает сеансовый ключ и соглашается использовать некоторый предопределенный ключ, скажем, сохранённый на стороне клиента как файл и на сервере в базе данных безопасности для этого определенного клиента, то Вы затем должны заставить этот плагин сообщить Firebird о том, что у него действительно есть сеансовый ключ.

Спецификации для ключа

Спецификации для размера ключа, его формата, способа его расчета и проверки и т.д. не обобщаются. Формат ключа и другие данные определяются плагином сетевого шифрования/дешифрования.

В частности, RC4 использует симметричный ключ, который может иметь любую длину, тогда как ключ, созданный SRP, имеет длину 20 байтов. Этот ключ представляет собой хэш SHA-1 в сеансе SRP и некоторых других связанных с SRP вещах, таких как имя пользователя.

Экспорт ключа из плагина аутентификации

Для экспорта ключа из плагина аутентификации используется интерфейс **ServerBlock** или **ClientBlock**. Один из них всегда передается в серверную/клиентскую часть плагина аутентификации. Оба имеют метод «**newKey**», который возвращает указатель на интерфейс **CryptKey**. Этот интерфейс, в свою очередь, имеет методы **setSymmetric** и **setAsymmetric** для хранения симметричного или асимметричного ключа в интерфейсе, то есть для экспорта этого ключа.

Доступ пользователей к объектам

Александр Пешков

Firebird 3 вводит новые привилегии SQL для сопоставления доступа между пользователями и группами и объектами безопасности и между базами данных ([**CORE-1900**](#)).

Теперь, когда Firebird поддерживает несколько баз данных безопасности, возникают некоторые новые проблемы, которые не возникали ранее при единой глобальной базе данных безопасности. Группы баз данных с использованием той же базы данных безопасности были эффективно

разделены. Сопоставления пользователей обеспечивают средства для достижения такой же эффективности, когда несколько баз данных используют свои собственные базы данных безопасности. В некоторых случаях требуется контроль за ограниченным взаимодействием между такими группами. Например:

- При обмене данными между группами с помощью оператора **EXECUTE STATEMENT ON EXTERNAL DATA SOURCE**;
- Когда требуется серверный доступ SYSDBA к базам данных из других групп с использованием служб;
- Похожие проблемы существовали в Firebird 2.1 и 2.5 для Windows, благодаря поддержке аутентификации доверенных пользователей - существовали два отдельных списка пользователей: один в базе данных безопасности и другой в Windows. Проблема возникала в случаях, когда необходимо было связать эти два списка пользователей.

Единственным решением для всех этих случаев является сопоставление регистрационной информации пользователя при подключается к серверу Firebird с внутренними объектами безопасности в базе данных **CURRENT_USER** и **CURRENT_ROLE**.

Правила доступа

Правила сопоставления состоят из четырех частей информации:

1. Область сопоставления — локальная, с текущей базой данных или глобальная, влияющая на все базы данных группы, включая базы данных безопасности;
2. Имя сопоставления - идентификатор SQL для сопоставления объектов в базе данных, как и любых других;
3. Объект из которого (*FROM*) производится сопоставления. Он состоит из четырех элементов:

- Источник аутентификации

имя подключаемого модуля *или*

результат сопоставления в другой базе данных *или*

использование аутентификации на сервере *или*

любой метод

- Имя базы данных, в которой была выполнена аутентификация
- Имя объекта, из которого производится сопоставление
- Тип этого имени - пользователь|роль|группа ОС - в зависимости от плагина, добавившего это имя во время аутентификации

Любой элемент принимается, требуется только его **тип**.

4. Объект, с которым производится сопоставление. Он состоит из двух элементов:

- Имя объекта, с которым была производится сопоставление
- Тип, для которого допустимы только пользователь (USER) или роль (ROLE)

*Синтаксис оператора **MAPPING** для объектов*

Сопоставления определяются с использованием следующего набора операторов DDL:

```
CREATE | ALTER | CREATE OR ALTER} [GLOBAL] MAPPING name
  USING {
    PLUGIN name [IN database] | ANY PLUGIN [IN database |
      SERVERWIDE] |
    MAPPING [IN database] | '*' [IN database]}
  FROM {ANY type | type name}
  TO {USER | ROLE} [name]
--
DROP [GLOBAL] MAPPING name
```

Описание

- Любое отображение может быть помечено как глобальное (GLOBAL).

Глобальное сопоставление лучше всего работает, если в качестве базы данных безопасности используется база данных Firebird 3 или более поздней версии. Если Вы планируете для этой цели использовать

другую базу данных, например, используя своего собственного поставщика, то Вам следует создать в ней таблицу с именем **RDB\$MAP** с той же структурой, что и **RDB\$MAP**, в базе данных Firebird 3 и с доступом для записи только для пользователя **SYSDBA**;

Опасность!

Если существуют глобальные и локальные сопоставления с одним и тем же именем, то сделайте так, чтобы все знали, что они разные объекты!

- Операторы **CREATE**, **ALTER** и **CREATE OR ALTER** используют один и тот же набор параметров. Имя (идентификатор) сопоставления используется для его идентификации, как и в других наборах команд DDL;
- Предложение **USING** имеет очень сложный набор опций:
 - явное указание имени плагина означает, что он будет работать только для этого плагина;
 - может использоваться любой доступный плагин, если только источник не является наследником предыдущего сопоставления;
 - он должен работать только с серверными плагинами;
 - он можно работать только с предыдущими результатами сопоставления;
 - он можно использовать любой метод, используя в качестве аргумента звездочку (*);
 - ему может быть предоставлено имя базы данных, которая создала сопоставление для объекта FROM.

Примечание

Этот аргумент недействителен для сопоставления аутентификации на сервере.

- Предложение **FROM** принимает обязательный аргумент типа именованного объекта:
 - При сопоставлении имен из подключаемых модулей тип определяется подключаемым модулем;
 - При сопоставлении результата предыдущего сопоставления тип может быть только **USER** или **ROLE**

→ Указание явного имени учитывается при сопоставлении;
→ Используйте ключевое слово **ANY** для работы с любым именем данного типа.

- В предложении **TO** необходимо указать **USER** или **ROLE**, с которыми выполняется сопоставление. Параметр **NAME** не является обязательным: если он не указан, используется имя из исходного сопоставления.

Примеры

В примерах используется синтаксис оператора **CREATE**. Использование оператора **ALTER** точно такое же, а использование **DROP** вполне очевидно.

1. Включение использования доверенной аутентификации Windows во всех базах данных, использующих текущую базу данных безопасности:

```
CREATE GLOBAL MAPPING TRUSTED_AUTH  
  USING PLUGIN WIN_SSPI  
  FROM ANY USER TO USER;
```

2. Включить доступ к SYSDBA для администраторов Windows в текущей базе данных:

```
CREATE MAPPING WIN_ADMINS  
  USING PLUGIN WIN_SSPI  
  FROM Predefined_Group  
  DOMAIN_ANY_RID_ADMINS  
  TO ROLE RDB$ADMIN;
```

Примечание

Группа **DOMAIN_ANY_RID_ADMINS** не существует в Windows, но такое имя будет добавлено подключаемым модулем **WIN_SSPI** для обеспечения точной обратной совместимости.

3. Включить конкретного пользователя из другой базы данных для доступа к текущей базе данных с другим именем:

```
CREATE MAPPING FROM_RT
```

```
USING PLUGIN SRP IN "rt"  
FROM USER U1 TO USER U2;
```

Важно

Имена баз данных или псевдонимы должны быть заключены в двойные кавычки в операционных системах с именами файлов, чувствительными к регистру.

4. Включение SYSDBA сервера (из основной базы данных безопасности) для доступа к текущей базе данных - в предположении, что база данных использует базу данных безопасности, отличную от умолчательной:

```
CREATE MAPPING DEF_SYSDBA  
  USING PLUGIN SRP IN "security.db"  
  FROM USER SYSDBA  
  TO USER;
```

5. Проверка, что пользователи, которые вошли в систему с использованием устаревшего модуля проверки подлинности, не имеют слишком большого количества привилегий:

```
CREATE MAPPING LEGACY_2_GUEST  
  USING PLUGIN LEGACY_AUTH  
  FROM ANY USER  
  TO USER GUEST;
```

Устаревшие правила доступа

Предыдущие версии Firebird имели одно жёстко закодированное глобальное правило по умолчанию: пользователи, успешно прошедшие аутентификацию в базе данных безопасности, имеют доступ ко всем базам данных. Суть этого "безопасного" правила: для базы данных безопасности нет смысла не доверять самой себе!

Но для обратной совместимости это правило сохранено в Firebird 3.

Универсальное сопоставление используется для задания правила для пользователя, подключённого к базе данных с использованием БД безопасности **security_1**, и выполняющего запрос к базе данных, использующей БД безопасности **security_2**, или когда аутентификация используется для всего сервера, например, такая как **WIN_SSPI**. Правило

вступает в действие всякий раз, когда ядро Firebird обрабатывает блок аутентификации, связанный с запросом на подключение к базе данных.

Сопоставление доверенных пользователей с CURRENT_USER и CURRENT_ROLE

Каждая запись в блоке аутентификации содержит имя подключаемого модуля, которое добавило его, тип записи (имя пользователя, группа ОС, роль и т.д.), имя объекта указанного типа (пользователь SYSDBA, роль PUBLIC, группа DOMAIN_ADMINS) и имя базы данных безопасности, в которой прошла аутентификация. При аутентификации на уровне всего сервера имя базы данных безопасности может иметь значение NULL (например, при аутентификации с помощью WIN_SSPI). Правило сопоставления оценивает все эти параметры, чтобы назначить пользователю значения контекстных переменных CURRENT_USER и CURRENT_ROLE в текущем подключении к серверу. В Firebird 3 явное сопоставление существует в системах с включенной проверкой подлинности “доверенного пользователя”, включая проверку подлинности WIN_SSPI в Windows, чтобы имя пользователя системы было присвоено контекстным переменным CURRENT_USER и CURRENT_ROLE.

Международные наборы символов для учетных записей пользователей

Александр Пешков

Новые модули проверки подлинности обеспечивают определенную гибкость при создании имен пользователей и паролей в отношении наборов символов. Чтобы использовать международную поддержку учетных данных, необходимо использовать SRP или любой другой набор плагинов аутентификации, который поддерживает новую модель.

Предупреждение

Настоятельно не рекомендуется использовать устаревший метод проверки подлинности, за исключением случаев, когда это необходимо для подключения клиента к серверам со старыми версиями Firebird (меньше, чем 3.x).

При создании плагинов для аутентификации нет необходимости

беспокоиться о наборах символов, поскольку весь обмен данными с новыми плагинами выполняется с использованием UTF8. Просто надо быть готовым обрабатывать имена пользователей и пароли, содержащие символы, выходящие за пределы 7-разрядного подмножества символов ASCII.

Условия для подключения Non-ASCII логинов

Простое правило для использования имён пользователей и паролей, отличных от ASCII - набор символов и кодовая страница, выбранная клиентом для данных, должны соответствовать тем, в которых учётные данные хранятся на сервере, и, очевидно, что учётные данные должны быть указаны в указанном наборе символов. Терминалы POSIX и большинство клиентов GUI этим условиям удовлетворяет. Однако есть три довольно распространенные ситуации, когда могут возникнуть проблемы:

1. Подключение через текстовую консоль Windows с выбранной кодовой страницей OEM;
2. Пропуск команды SET NAMES в *isql*;
3. Выполнение скриптов.

Рассмотрим эти ситуации более подробно:

1. Все операционные системы, кроме Windows, используют одинаковый набор символов (кодовая страница, язык) по умолчанию для всех программ. В Windows по историческим причинам приложения GUI используют кодовую страницу ANSI, но приложения запускаются в эмуляторе консоли командной строки с использованием соответствующей кодовой страницы OEM. В качестве примера, в русской версии Windows для GUI приложений используется кодовая страница 1251 (ANSI на русском языке), а для приложений командной строки - кодовая страница 866 (OEM на русском языке). Клиент Firebird всегда использует кодовую страницу ANSI. Для обеспечения полнофункционального поведения консольного приложения, консоль должна быть переключена на кодовую страницу ANSI (chcp 1251 для нашего русского примера). Странности Windows на этом не прекращаются. Параметры, передаваемые в приложение Windows (но не в 16-разрядную DOS) всегда передаются в кодировке ANSI, даже если приложение было

запущено из консоли командной строки, работающей по умолчанию в режиме OEM. Это может создать иллюзию, что утилиты Firebird корректно запускаются из консоли OEM. Однако они корректно работают только, и только в том случае, когда международная информация явно приведена в командной строке, например

```
C:> isql server:database -user <intl-login> -pas <intl-password>
```

Однако, как только вы попытаетесь использовать команду **CONNECT** или **CREATE DATABASE** в утилите **isql** или ввести пароль в терминале с кодировкой OEM Вы столкнетесь с проблемами. Это очень неудобно. Проблемы с консолью OEM планируется исправить несколько позднее.

2. В настоящее время параметры командной строки и команд **CONNECT** и **CREATE DATABASE** в **isql** не зависят от **SET NAMES** или параметра **Charset**. Все остальные команды, в частности **CREATE/ALTER/DROP USER**, зависят от них, и поэтому для них необходимо задавать набор символов. Это очень важно, потому что в будущих версиях планируется, чтобы использование международных учетных данных зависело только от явно указанного набора символов. Обратите внимание на один очень запутанный случай использования - это попытка установить пароль без ASCII для текущего пользователя:

```
SQL> alter current user set password '<intl-password>';
```

Эта команда успешно выполнится, даже если набор символов установлен неправильно. Однако последующая попытка входа в систему с изменённым паролем завершится ошибкой — неправильный пароль.

3. В скриптах основное требование - установить правильный набор символов для SQL сервера, который должен запустить сценарий, используя оператор **SET NAMES**. Для Firebird 3 важно правильно указать настройки языка и кодовой страницы для среды, в которой должен выполняться сценарий. Они должны соответствовать набору символов, установленному в скрипте. Правильная настройка языка влияет, в частности, на учётные данные, переданные на сервер при подключении к нему. Остальная часть скрипта должна продолжаться успешно даже без правильной настройки среды (Это не относится к операторам **INSERT** и **UPDATE**,

в которых имеются строки с кодировкой, отличной от указанной в операторе SET NAMES).

Таким образом, если Вы планируете использовать международные символы в скрипте, то Вам необходимо следить за международными настройками в вашей среде.

Напоминание

Имена пользователей - это идентификаторы SQL, а значит к ним должны применяться такие же правила, то есть они заключаются в двойные кавычки, если содержат международные символы или когда чувствительны к регистру.

Возможности SQL для управления доступом

Изменения в архитектуре сервера, ужесточение правил к безопасности и целостности данных, наряду с появлением новых требований, привели к появлению в этом выпуске ряда новых команд SQL для управления пользователями и доступа к объектам.

Управление пользователями с помощью SQL

Александр Пешков

В Firebird 3 был расширен набор SQL команд DDL для управления учётными записями пользователей, что улучшает возможности управления (добавление, изменение или удаление) пользователей в базе данных безопасности при обычном подключении к базе данных.

Формы синтаксиса

```
CREATE USER username [options_list] [TAGS (tag [, tag [, tag ...]])]
```

```
ALTER USER username [SET] [options_list] [TAGS (tag [, tag [, tag ...]])]
```

```
ALTER CURRENT USER [SET] [options_list]  
[TAGS (tag [, tag [, tag ...]])]
```

```
CREATE OR ALTER USER username [SET] [options]  
[TAGS tag [, tag [, tag ...]])]
```

DROP USER username [USING PLUGIN plugin_name]

Опции (OPTIONS) — список (возможно, пустой) со следующими параметрами:

PASSWORD 'password' // пароль

FIRSTNAME 'string value' // Имя

MIDDLENAME 'string value' // Отчество

LASTNAME 'string value' // Фамилия

ACTIVE // установить активным

INACTIVE // установить неактивным

USING PLUGIN plugin_name // использовать плагин имя плагина

Каждый параметр TAG может иметь одну из двух форм:

TAGNAME = 'string value'

или DROP TAGNAME для полного удаления пользовательского атрибута

DROP TAGNAME

Примечание

Значением параметра TAGNAME вида имя/значение может быть любой действительный идентификатор SQL.

Устаревшие методы

Выше уже говорилось, что Firebird 3.0 может поддерживать несколько баз данных безопасности. Но эта возможность не поддерживается утилитой gsec или API интерфейсом служб. Использование этих методов устарело.

Подробности использования

Предложения CREATE USER, CREATE или ALTER USER и DROP USER доступны для использования только для SYSDBA или другого

пользователя, которому предоставлена роль **RDB\$ADMIN** в базе данных безопасности (и, конечно, подключившемся к базе данных с этой ролью).

Для создания нового пользователя требуется предложение **PASSWORD**.

Обычный пользователь может использовать свой собственный пароль, атрибуты реального имени и теги.

Если вы хотите изменить свои данные, то Вы можете использовать упрощенную форму – **ALTER CURRENT USER**.

В командах **ALTER USER** или **CREATE OR ALTER USER** должен присутствовать, по крайней мере, один из параметров **PASSWORD**, **FIRSTNAME**, **MIDDLENAME**, **LASTNAME**, **ACTIVE**, **INACTIVE** или **TAGS**.

Опция **INACTIVE** используется для отключения возможности подключения пользователя, не удаляя его. Предложение **ACTIVE** восстанавливает возможность подключения.

Быстрый Совет

Начиная с версии Firebird 3.0.1 оператор **CREATE** или **ALTER USER SYSDBA PASSWORD <пароль>** может использоваться для инициализации пустой базы данных безопасности **Security.fdb**.

TAGS - это список атрибутов, определенных для конечного пользователя. Длина строкового значения опции **tag** не должна превышать 255 байт.

Настройка списка тегов для пользователя сохраняет ранее установленные теги, если они не упоминаются в текущем списке.

Примечание

UID или **GID**, введенные в устаревшей утилите **gsec**, рассматриваются теперь как тег в интерфейсе SQL.

Примеры

Общие:

```
CREATE USER superhero PASSWORD 'test';  
ALTER USER superhero SET FIRSTNAME 'Clark' LASTNAME
```

'Kent';

CREATE OR ALTER USER superhero SET PASSWORD 'ldQfA';

DROP USER superhero;

ALTER CURRENT USER SET PASSWORD

'SomethingLongEnough';

Работа с тегами:

ALTER USER superhero SET TAGS (a='a', b='b');

NAME	VALUE
=====	=====
A	a
B	b

ALTER USER superhero SET TAGS (b='x', c='d');

NAME	VALUE
=====	=====
A	a
B	x
C	d

ALTER USER superhero SET TAGS (drop a, c='sample');

NAME	VALUE
=====	=====
B	x
C	sample

Отображение списка пользователей:

```
SELECT CAST(U.SEC$USER_NAME AS CHAR(20)) LOGIN,  
       CAST(A.SEC$KEY AS CHAR(10)) TAG,  
       CAST(A.SEC$VALUE AS CHAR(20)) "VALUE",  
       SEC$PLUGIN "PLUGIN"
```

```
FROM SEC$USERS U LEFT JOIN SEC$USER_ATTRIBUTES A ON  
       U.SEC$USER_NAME = A.SEC$USER_NAME;
```

LOGIN	TAG	VALUE	PLUGIN
=====	=====	=====	=====
SYSDBA	<null>	<null>	Srp
SUPERHERO	B	x	Srp
SUPERHERO	C	sample	Srp
SYSDBA	<null>	<null>	Legacy_UserManager

Примечание

Вывод информации зависит от используемого плагина управления пользователями. При использовании устаревшего подключаемого модуля некоторые параметры не поддерживаются и будут просто игнорироваться.

SET ROLE

Александр Пешков

CORE-1377

Совместимый с SQL-2008 оператор SET ROLE позволяет задать контекстную переменную CURRENT_ROLE, которая была предоставлена для CURRENT_USER, или пользователю, подключённому к базе данных в качестве доверенного лица (SET TRUSTED ROLE).

Синтаксис SET ROLE

Предоставление роли <rolename> пользователю CURRENT_USER:

```
SET ROLE <rolename>
```

Пример использования SET ROLE

```
SET ROLE manager;
select current_role from rdb$database;
```

Результат:

ROLE

=====

MANAGER

SET TRUSTED ROLE

Идея отдельной команды **SET TRUSTED ROLE** заключается в следующем: когда доверенный пользователь подключается к базе данных без предоставления информации о роли оператор **SET TRUSTED ROLE** делает доверенную роль (если таковая существует) текущей (**CURRENT_ROLE**) без какой-либо дополнительной активности, такой как установка её в **DPB**.

Доверенная роль не является определенным типом роли, она может быть любой ролью, созданной с помощью оператора **CREATE ROLE**, или предопределенной системной ролью, такой как **RDB\$ADMIN**. Роль становится доверенной ролью для подключения, когда подсистема сопоставления объектов безопасности (см. [Доступ пользователей к объектам](#)) находит совпадение между результатом проверки подлинности, переданным из плагина, и локальным или глобальным сопоставлением для текущей базы данных. Роль может быть даже такой, которая явно не предоставлена явно доверенному пользователю.

Примечания

- По умолчанию доверенная роль не назначается. Это можно изменить с помощью соответствующего плагина проверки подлинности и команд **CREATE/ALTER MAPPING** (см. [Синтаксис оператора MAPPING для объектов](#)).
- Хотя **CURRENT_ROLE** можно изменить с помощью **SET ROLE**, не всегда можно восстановить роль с помощью этой же команды, потому что она выполняет проверку прав доступа.

Синтаксис

Включение доступа к доверенной роли, если **CURRENT_USER** вошел в систему как доверенный пользователь при условии, что роль доступна:

SET TRUSTED ROLE

Примером использования доверенной роли является назначение системной роли RDB\$ADMIN администратору Windows при использовании доверенной аутентификации Windows.

GRANT/REVOKE и GRANTED BY указанному пользователю

Александр Пешков

Ранее правом предоставлять или отзывать SQL привилегии всегда обладал текущий пользователь. Теперь в командах GRANT и REVOKE можно предоставить права от имени другого пользователя, отличного от CURRENT_USER.

Синтаксис

```
grant <right> to <object> [{granted by|as} [user] <username>]
revoke <right> from <object> [{granted b|as} [user] <username>]
```

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

Пример (пользователь SYSDBA)

```
create role r1;
grant r1 to user1 with admin option;
grant r1 to public granted by user1;

-- (in isql)
show grant;

/* Grant permissions for this database */
GRANT R1 TO PUBLIC GRANTED BY USER1
GRANT R1 TO USER1 WITH ADMIN OPTION
```

REVOKE ALL ON ALL

Эта новая команда полезна для удаления прав доступа ко всем объектам в базе данных пользователя после удаления его из базы данных безопасности или другого источника аутентификации.

Синтаксис

REVOKE ALL ON ALL FROM [USER] username

REVOKE ALL ON ALL FROM [ROLE] rolename

Пример

```
# gsec -del guest
```

```
# isql employee
```

```
fbs bin # .isql employee
```

```
Database: employee
```

```
SQL> REVOKE ALL ON ALL FROM USER guest;
```

```
SQL>
```

Права пользователей на изменение метаданных

Дмитрий Еманов и Роман Симаков

В Firebird 3 системные таблицы доступны только для чтения. Поэтому введён синтаксис SQL, предоставляющий возможность для назначения привилегий изменения метаданных указанным пользователям или ролям для указанных объектов (см. [CORE-735](#)).

Примечание

Некоторые люди применяют псевдоним «DDL привилегии» к этой функции. Не путайте его с «DDL триггеров»! Более правильным названием является “привилегии метаданных”

Синтаксис

Назначение привилегий для метаданных:

```
GRANT CREATE <object-type>
  TO [USER | ROLE] <user-name> | <role-name>
  [WITH GRANT OPTION];
```

```
GRANT ALTER ANY <object-type>
  TO [USER | ROLE] <user-name> | <role-name>
  [WITH GRANT OPTION];
```

```
GRANT DROP ANY <object-type>
  TO [USER | ROLE] <user-name> | <role-name>
  [WITH GRANT OPTION];
```

Удаление привилегий для метаданных:

```
REVOKE [GRANT OPTION FOR] CREATE <object-type>
  FROM [USER|ROLE] <user-name>|<role-name>;
```

```
REVOKE [GRANT OPTION FOR] ALTER ANY <object-type>
  FROM [USER|ROLE] <user-name>|<role-name>;
```

```
REVOKE [GRANT OPTION FOR] DROP ANY <object-type>
  FROM [USER|ROLE] <user-name>|<role-name>;
```

Специальная форма для доступа к базе данных:

```
GRANT CREATE DATABASE
  TO [USER|ROLE] <user-name> | <role-name>;
```

```
GRANT ALTER DATABASE
  TO [USER|ROLE] <user-name> | <role-name>
  [WITH GRANT OPTION];
```

```
GRANT DROP DATABASE
  TO [USER|ROLE] <user-name> | <role-name>
  [WITH GRANT OPTION];
```

```
REVOKE CREATE DATABASE
  FROM [USER | ROLE] <user-name> | <role-name>;
```

```
REVOKE [GRANT OPTION FOR] ALTER DATABASE
  FROM [USER | ROLE] <user-name> | <role-name>;
```

```
REVOKE [GRANT OPTION FOR] DROP DATABASE
FROM [USER | ROLE] <user-name> | <role-name>;
```

Замечания по использованию

- <object-type> может быть любым из следующих:

CHARACTER SET	COLLATION	DOMAIN
EXCEPTION	FILTER	FUNCTION
GENERATOR	PACKAGE	PROCEDURE
ROLE	SEQUENCE	TABLE
VIEW		

Примечание

Доступ к метаданным для триггеров и индексов осуществляется через привилегии для таблицы, для которых они созданы.

- При использовании параметра **ANY** пользователь сможет выполнять любую операцию для любого объекта;
- Если параметр **ANY** отсутствует, то пользователь сможет выполнять операции над объектом только в том случае, если он является его владельцем;
- Если опция **ANY** была получена с помощью операции **GRANT**, то для ее отмены используется команда **REVOKE** с такими же параметрами, как и у операции **GRANT**.

Примеры

```
GRANT CREATE TABLE TO Joe;
GRANT ALTER ANY TABLE TO Joe;
REVOKE CREATE TABLE FROM Joe;
```

Права GRANT EXECUTE для UDF

Дмитрий Еманов

CORE-2554: Разрешение на выполнение оператора EXECUTE теперь поддерживается для UDF (как для устаревших, так и для PSQL).

Синтаксис

```
GRANT EXECUTE ON FUNCTION <name> TO <grantee list>
    [<grant option> <granted by clause>]
--
REVOKE EXECUTE ON FUNCTION <name> FROM <grantee list>
    [<granted by clause>]
```

Примечание

Изначально разрешение для выполнения оператора EXECUTE предоставляется владельцу функции (пользователю, создавшему или объявившему функцию).

Улучшение для рекурсивных хранимых процедур

Александр Пешков

Рекурсивная хранимая процедура больше не требует привилегии на оператор EXECUTE при вызове самой себя (см. ***CORE-3242***).

Права для защиты других объектов метаданных

Для защиты объектов метаданных, отличных от таблиц, представлений, процедур и функций, введено новое разрешение USAGE, соответствующее требованиям SQL-2008.

Синтаксис

```
GRANT USAGE ON <object type> <name>
    TO <grantee list> [<grant option> <granted by clause>]
--
REVOKE USAGE ON <object type> <name>
```

FROM <grantee list> [<granted by clause>]

--

<object type> ::= {DOMAIN | EXCEPTION | GENERATOR |
SEQUENCE | CHARACTER SET | COLLATION}

Примечания

Изначально разрешение USAGE предоставляется владельцу объекта (пользователю, создавшему объект).

В Firebird 3.0.0 разрешено использовать USAGE только для исключений (exception) - см. [CORE-2884](#) и генераторов (generator)/последовательностей (sequence) — gen_id, next value for см. [CORE-2553](#). Предоставление привилегий для наборов символов, сортировок и доменов отключено, что делает эти типы объектов недоступными для любых команд GRANT или REVOKE. Доступ к ним не подлежит какой-либо форме принудительного исполнения, хотя это может измениться в будущих выпусках, если это будет сочтено целесообразным.

Псевдо-таблицы со списком пользователей

[CORE-2639](#)

Для получения доступ к спискам пользователей и атрибутов надо выполнить запрос к виртуальным таблицы **SEC\$USERS** и **SEC\$USER_ATTRIBUTES**.

Важно

Эта функция сильно зависит от плагина. Учтите, что некоторые параметры игнорируются при использовании устаревшего модуля управления пользователями. управления пользователями.

Псевдо-таблицы очень похожи на семейство таблиц MON\$, используемых для мониторинга сервера. Таблица создается по запросу при запуске оператора

```
SELECT * FROM SEC$USERS
```

или

```
SELECT * FROM SEC$USER_ATTRIBUTES
```

В результате перечисляются пользователи (или их атрибуты) в базе данных безопасности, которая настроена для текущей базы данных и доступна для управления текущему пользователю. Таблица SEC\$USERS включает поле, указывающее, имеет ли пользователь роль RDB\$ADMIN в базе данных безопасности.

Ограничения для DROP ROLE (3.0.1)

CORE-5248

До Firebird 3.0.1 любой пользователь мог удалить роль. Это ошибка была исправлена в субрелизе 3.0.1. Теперь только пользователь, создавший роль, и тот, кто получил роль WITH ADMIN OPTION, может удалить её.

Важно

Чтобы получить правильное поведение роли в старых базах данных, её нужно пересоздавать!

Глава 8

Язык определения данных - Data Definition Language (DDL)

Быстрые ссылки для DDL

- Тип данных BOOLEAN;
- Тип столбца IDENTITY;
- Управление данными NULL в доменах и столбцах;
- Изменения для генераторов (последовательностей);
- Изменение набора символов по умолчанию;
- Выражения BLOB в вычисляемых столбцах;
- «Задержка» закрытия базы данных для Superserver;
- Возможность сохранения файла тени;
- Новое в SQL для управления пользователями и правами доступа.

Улучшения DDL

В языка определения данных SQL были добавлены следующие улучшения.

Новые типы данных

В Firebird 3 введён полноценный булевский тип, а также эмуляция столбца «identity» в стиле Microsoft.

Тип данных BOOLEAN

Adriano dos Santos Fernandes

Логический тип данных **Boolean**, совместимый с SQL-2008 (8 бит), содержит значения **TRUE** и **FALSE**. Если это не запрещено ограничением **NOT NULL**, тип данных **BOOLEAN** также может иметь значение **UNKNOWN** (т.е. **NULL**). Спецификация не делает различия между значением **NULL** и **UNKNOWN** для этого типа данных в предикатах SQL, условиях поиска или значении логического выражения: они могут

использоваться взаимозаменяемо, т.к. эквивалентны.

Как и во многих языках программирования, тип данных **BOOLEAN** может использоваться в SQL с неявными значениями. Например, допустимыми выражениями являются `field2 OR field2` и `NOT field1`.

Оператор IS

Оператор **IS** [**NOT**] используется в качестве предикатов для сопоставления. Например, `field1 IS FALSE`, или `field1 IS NOT TRUE`.

Примечание

Операторы эквивалентности ("**=**", "**! =**", "**<>**" и т.д.) действительны во всех сравнениях.

Примеры

```
CREATE TABLE TBOOL (ID INT, BVAL BOOLEAN);
COMMIT;
```

```
INSERT INTO TBOOL VALUES (1, TRUE);
INSERT INTO TBOOL VALUES (2, 2 = 4);
INSERT INTO TBOOL VALUES (3, NULL = 1);
COMMIT;
```

```
SELECT * FROM TBOOL
ID          BVAL
=====
1           <true>
2           <false>
3           <null>
```

-- Проверка значения TRUE

```
SELECT * FROM TBOOL
      WHERE BVAL
ID          BVAL
=====
1           <true>
```

-- Проверка значения FALSE

```
SELECT * FROM TBOOL
WHERE BVAL IS FALSE
```

ID	BVAL
=====	=====
2	<false>

```
-- Проверка значения UNKNOWN
SELECT * FROM TBOOL
WHERE BVAL IS UNKNOWN
```

ID	BVAL
=====	=====
3	<null>

```
-- Логическое выражение в SELECT
SELECT ID, BVAL, BVAL AND ID < 2
FROM TBOOL
```

ID	BVAL	
=====	=====	=====
1	<true>	<true>
2	<false>	<false>
3	<null>	<false>

```
-- PSQL - декларация с начальным значением
DECLARE VARIABLE VAR1 BOOLEAN = TRUE;
```

```
-- Действительный синтаксис, но, как и при сравнении с NULL,
-- никогда не вернет никакой записи
SELECT * FROM TBOOL WHERE BVAL = UNKNOWN
SELECT * FROM TBOOL WHERE BVAL <> UNKNOWN
```

Использование Boolean для других типов данных

Хотя **BOOLEAN** по сути не конвертируется в любой другой тип данных, в версии 3.0.1 строки 'true' и 'false' (без учета регистра) будет неявно приводиться к **BOOLEAN** в выражениях. Например, в выражении

```
if (true > 'false') then ...
```

'false' конвертируется в тип данных **BOOLEAN**. Попытка использовать логические операторы **AND**, **NOT**, **OR** и **IS** не удастся. Например, выражение

NOT 'False' недопустимо.

Логическое значение может быть явно преобразовано в строку и из строки с использованием оператора CAST. Значение UNKNOWN недоступно для любого преобразования.

Другие примечания

- Представлен в API типом FB_BOOLEAN и константами FB_TRUE и FB_FALSE;
- Значение TRUE больше значения FALSE.

Ключевые слова INSERTING, UPDATING и DELETING

Для избежания двусмысленности при использовании логических выражений ранее не зарезервированные ключевые слова INSERTING, UPDATING и DELETING, возвращающие True|False при использовании в PSQL, были сделаны зарезервированными словами во всех контекстах. Если вы ранее использовали любое из этих слов в качестве идентификаторов для объектов базы данных, столбцов, переменных или параметров, то их необходимо будет переопределить либо с новыми именами, либо путем заключением этих идентификаторов в двойные кавычки.

Тип столбца IDENTITY

Adriano dos Santos Fernandes

Столбец типа identity - это столбец, связанный с внутренним генератором последовательности. Его значение устанавливается автоматически, если столбец пропущен в инструкции INSERT.

Синтаксис

```
<column definition> ::=  
    <name> <type> GENERATED BY DEFAULT AS IDENTITY  
    [(START WITH <value>)] <constraints>
```

При определении столбца необязательное предложение START WITH позволяет инициализировать генератор любым значением, отличным от нуля (см. [CORE-4199](#)).

```
<alter column definition> ::=  
    <name> RESTART [WITH <value>]
```

Определение столбца может быть изменено только для изменения начального значения генератора. Оператор **RESTART** сбрасывает генератор до нуля; Необязательное предложение **WITH <value>** позволяет изменить начальное значения генератора на любое, отличное от нуля, значение (см. [CORE-4206](#)).

Правила

- Тип данных столбца **identity** должен быть числом с нулевым масштабом (т.е. с отсутствием цифр после запятой). Разрешено использование следующих типов: **SMALLINT**, **INTEGER**, **BIGINT**, **NUMERIC(x,0)** и **DECIMAL(x,0)**;
- Для столбца типа данных **identity** нельзя использовать предложения **DEFAULT** или **COMPUTED**.

Примечания

- Столбец типа **identity** нельзя изменить на обычный, также как и обратное преобразование;
- Столбец типа **identity** неявно имеет атрибут **NOT NULL**;
- Уникальность для столбца типа **identity** автоматически не применяется. Для обеспечения уникальности требуется ограничение **UNIQUE** или **PRIMARY KEY**;
- Не рекомендуется использование других методов генерации ключевых значений для столбцов типа **identity**, например, с помощью триггера или путем предоставления пользователям возможности изменять или добавлять их, во избежание непредвиденных нарушений целостности данных.

Пример

```
create table objects (  
    id integer generated by default as identity primary key,  
    name varchar(15));  
  
insert into objects (name) values ('Table');
```

```
insert into objects (name) values ('Book');
insert into objects (id, name) values (10, 'Computer');

select * from objects;
```

ID	NAME
=====	=====
1	Table
2	Book
10	Computer

Детали реализации

Для поддержки **identity** в таблицу RDB\$RELATION_FIELDS добавлены два новых столбца: RDB\$GENERATOR_NAME и RDB\$IDENTITY_TYPE.

- Поле RDB\$GENERATOR_NAME хранит имя автоматически созданного генератора для столбца. Значение RDB\$SYSTEM_FLAG в таблице RDB\$GENERATORS для этого генератора будет равным 6;
- В настоящее время RDB\$IDENTITY_TYPE всегда хранит значение 1 (по умолчанию) для столбцов с типом **identity** и NULL для остальных столбцов.

Управление данными NULL в доменах и столбцах

Adriano dos Santos Fernandes

Синтаксис оператора ALTER доступен для изменения значения NULL столбца таблицы или домена.

Синтаксис

```
ALTER TABLE <table name>
    ALTER <field name> {DROP|SET} [NOT] NULL
ALTER DOMAIN <domain name> {DROP|SET } [NOT] NULL
```

Примечания

Успешность изменения флага в столбце таблицы из NULL в NOT

NULL зависит от результата полной проверки данных в таблице. Перед изменением флага убедитесь, что столбец не имеет значений NULL.

При изменении домена подлежат проверке все таблицы, использующие домен.

Задание NOT NULL для столбца, зависящего от домена, преобладает над заданием NOT NULL для домена. В этом случае задание для домена атрибута NULL не распространяется на столбец.

Изменения для генераторов (последовательностей)

Добавлены дополнительные параметры для изменения генераторов (последовательностей). Если ранее в SQL единственным вариантом был ALTER SEQUENCE <имя последовательности> RESTART WITH <значение>, то теперь предоставляется полный синтаксис, а объекты GENERATOR и SEQUENCE являются синонимами для всего диапазона команд.

RESTART теперь можно использовать самостоятельно для пересоздания последовательности или изменения начального значения. В системную таблицу RDB\$GENERATORS для хранения этого значения добавлен новый столбец RDB\$INITIAL_VALUE.

Генератор (последовательность) также может быть создан, пересоздан или изменен для включения необязательного предложения “step” (шаг, приращение), чтобы генератор увеличивал своё значение на два или большее число вместо значения по умолчанию для шага 1. Предложение реализуется как INCREMENT BY <number> и хранится в поле RDB\$GENERATOR_INCREMENT таблицы RDB\$GENERATORS.

Синтаксис

```
{CREATE|RECREATE} {SEQUENCE|GENERATOR } <sequence name>
```

```
[START WITH <value>] [INCREMENT BY <number>]
```

```
CREATE OR ALTER {SEQUENCE|GENERATOR} <sequence name>
```

```
{RESTART|START WITH <value>} [INCREMENT BY <number>]
```

```
ALTER {SEQUENCE|GENERATOR} <sequence name>
```

RESTART [WITH <value>] [INCREMENT BY <number>]

Функция GEN_ID()

Устаревшая функция GEN_ID (имя_генератора, шаг) все ещё работает с установкой шага (приращения) во втором аргументе. При её использовании сохраненное в поле RDB\$GENERATOR_INCREMENT значение переопределяется.

Изменение набора символов по умолчанию

Adriano dos Santos Fernandes

ALTER DATABASE

...

SET DEFAULT CHARACTER SET <new_charset>

Изменение набора символов не изменяет существующие данные. Новый набор символов по умолчанию используется только в последующих командах DML и предполагает параметры сортировки по умолчанию нового набора символов.

Выражения BLOB в вычисляемых столбцах

Adriano dos Santos Fernandes

Подстрока из столбца BLOB теперь может использоваться для определения вычисляемого столбца.

Пример

ALTER TABLE ATABLE

ADD ABLOB

COMPUTED BY

(SUBSTRING(BLOB_FIELD FROM 1 FOR 20))

«Задержка» закрытия базы данных для Superserver

Иногда желательно, чтобы Firebird в режиме Суперсервера (Superserver) сохранял базу данных открытой в течение какого-то периода

времени после закрытия последнего подключения к ней, т. е. заставить его «задержаться» некоторое время. Это может помочь улучшать производительность при низкой стоимости затрат в условиях частого открытия и закрытия базы данных (подключений и отключений), сохраняя ресурсы (в частности кэш) «теплыми» для следующего подключения.

Для управления этой дополнительной задержкой закрытия базы данных, работающих под Firebird в режиме Суперсервера (Superserver), в оператор ALTER DATABASE добавлен новый параметр LINGER.

Синтаксис

```
ALTER DATABASE SET LINGER TO {секунды};
```

```
ALTER DATABASE DROP LINGER;
```

Использование

Для установки или очистки задержки для базы данных, выполните следующие действия:

```
-- установка задержки 30 секунд  
ALTER DATABASE SET LINGER TO 30;
```

Любая из следующих форм очистит настройку задержки и вернет базу данных в нормальное состояние (без задержек):

```
ALTER DATABASE DROP LINGER;  
ALTER DATABASE SET LINGER TO 0;
```

Примечание

Удаление (DROP) LINGER не является идеальным решением при необходимости отключить задержку, за исключением единственного случая - когда требуется принудительное завершение работы сервера. Утилита *gfix* теперь имеет ключ *-NoLinger*, который закроет указанную базу данных сразу же после отключения последнего пользователя независимо от настройки параметра LINGER в базе данных. Установленное значение параметра LINGER при этом сохраняется и работает нормально при последующих подключениях к базе данных.

Такое же однократное переопределение теперь доступно также через API интерфейс служб с использованием тега `isc_spb_prp_nolinger`, например (в одной строке):

```
fbsvcmgr host:service_mgr user sysdba password xxx  
action_properties dbname employee prp_nolinger
```

Обсуждения разработки этой функции смотрите также в тикете трекера [*CORE-4263*](#).

Возможность сохранения файла тени

Александр Пешков

Команда `DROP SHADOW` имеет новую возможность сохранения теневого файла в файловой системе:

```
DROP SHADOW shadow_num  
[{DELETE | PRESERVE} FILE]
```

См. [*CORE-4955*](#).

Новое в SQL для управления пользователями и правами доступа

Много новых возможностей и улучшений были добавлены к словарю DDL для управления пользователями и правами доступа к объектам в базах данных. Они подробно описаны в Главе 7 [*Безопасность*](#) в разделе [*Возможности SQL для управления доступом*](#).

Управление пользователями с помощью SQL

Набор команд SQL DDL для управления учётными записями пользователей был расширен в Firebird 3, что улучшает возможности управления (добавление, изменение или удаление) пользователями в базе данных безопасности при обычном подключении к базе данных.

gsec устарел!

Начиная с Firebird 3 утилита командной строки **gsec** считается устаревшей. Она может работать с **security3.fdb**, но не будет работать с альтернативными базами данных безопасности.

SET ROLE и SET TRUSTED ROLE

Совместимый с SQL-2008 оператор **SET ROLE** позволяет задать контекстную переменную **CURRENT_ROLE**, которая была предоставлена для **CURRENT_USER**, или пользователю, подключённому к базе данных в качестве доверенного лица (**SET TRUSTED ROLE**).

GRANT/REVOKE и GRANTED BY указанному пользователю

Ранее правом предоставлять или отзывать SQL привилегии всегда обладал текущий пользователь. Теперь в командах **GRANT** и **REVOKE** можно предоставить права от имени другого пользователя, отличного от **CURRENT_USER**.

REVOKE ALL ON ALL

Эта новая команда полезна для удаления прав доступа ко всем объектам в базе данных пользователя после удаления его из базы данных безопасности или другого источника аутентификации.

GRANT/REVOKE для привилегий метаданных

В Firebird 3 системные таблицы доступны только для чтения. Поэтому введён синтаксис SQL, предоставляющий возможность для назначения привилегий изменения метаданных указанным пользователям или ролям для указанных объектов. Более полную информацию об этом читайте в разделе *Права пользователей на изменение метаданных*.

Права GRANT EXECUTE для UDF

Разрешение на выполнение оператора **EXECUTE** теперь поддерживается для **UDF** (как для устаревших, так и для **PSQL**).

Права для защиты других объектов метаданных

Для защиты объектов метаданных, отличных от таблиц,

представлений, процедур и функций, введено новое разрешение **USAGE**, соответствующее требованиям SQL-2008.

Глава 9

Язык манипулирования данными (DML)

В этой главе представлены дополнения и улучшения, которые были добавлены в язык манипулирования данными (*data manipulation language - DML*) SQL в Firebird 3.0.

Быстрые ссылки для DML

- Дополнительные возможности SQL 2008 для MERGE;
- Оконные (аналитические) функции;
- Вывод расширенного плана;
- SUBSTRING в регулярных выражениях;
- Обратные гиперболические тригонометрические функции;
- Статистические функции;
- Снято ограничение в 32 кБ для аргумента в функции TRIM() BLOB;
- Улучшения функции DATEADD();
- Альтернатива кавычкам в строковых литералах;
- Операторы OFFSET и FETCH совместимы с SQL-2008;
- Запрет одновременного использования явных и неявных соединений;
- Поддержка параметров в левой части предложения WHERE;
- Улучшения для предложения RETURNING;
- Предложение RETURNING может использовать псевдонимы;
- Устойчивость курсора;
- Улучшение GTT;
- Улучшение для строк DML;
- Функция COUNT() возвращает тип BIGINT;
- Улучшения для SIMILAR TO;
- OR'ed параметр в предложении WHERE;
- Небольшое улучшение для 1-го диалекта;
- Улучшения встроенного SQL (ESQL).

Дополнительные возможности SQL 2008 для MERGE

Adriano dos Santos Fernandes

Поддержка MERGE была дополнена внедрением следующих функций:

- Добавление расширения DELETE для оператора MERGE ([CORE-2005](#));
- Включение использования нескольких операторов WHEN MATCHED|NOT MATCHED ([CORE-3639](#)) и способность применять условия для WHEN MATCHED|NOT MATCHED;
- Добавление в оператор MERGE пункта RETURNING ... INTO ... ([CORE-3020](#)).

Цель оператора MERGE - считывать данные из источника и выполнять операции INSERT или UPDATE в целевой таблице в соответствии с выбранным условием. Доступно в DSQL и PSQL.

Синтаксис

```
<merge statement> ::=
MERGE
    INTO <table or view> [[AS] <correlation name>]
    USING <table or view or derived table>
        [[AS] <correlation name>]
    ON <condition>
    <merge when>...
    <returning clause>
```

```
<merge when> ::=
<merge when matched> |
<merge when not matched>
```

```
<merge when matched> ::=
WHEN MATCHED [AND <condition>] THEN
    {UPDATE SET <assignment list>|DELETE }
```

```
<merge when not matched> ::=
WHEN NOT MATCHED [ AND <condition> ] THEN
    INSERT [(<column list>)]
    VALUES (<value list>)
```

Правила

Должно быть указано по крайней мере одно из предложений `<merge when matched>` или `<merge when not matched>`.

Пример

```
MERGE INTO customers c
  USING
    (SELECT * FROM customers_delta WHERE id > 10) cd
  ON (c.id = cd.id)
  WHEN MATCHED THEN
    UPDATE SET name = cd.name
  WHEN NOT MATCHED THEN
    INSERT (id, name) VALUES (cd.id, cd.name)
```

Примечания

Правое соединение (`right join`) выполняется между таблицами `INTO` (слева) и `USING` с использованием условия. Если запись существует в левой таблице (`INTO`), то выполняется операция `UPDATE`, иначе выполняется операция `INSERT`.

Как только будет определено, совпадает ли источник с записью в целевом объекте, набор, сформированный из соответствующих (при совпадении - `WHEN MATCHED`/при не совпадении - `WHEN NOT MATCHED`) предложений, оценивается в указанном порядке для проверки их необязательных условий. Первое предложение, для которого условие выполняется, является тем, будет выполнено, а последующие будут проигнорированы.

Если соединение не возвращает ни одной записи, то оператор `INSERT` не выполняется.

Оконные (аналитические) функции

Adriano dos Santos Fernandes

Согласно спецификации SQL, оконные функции (также известные как аналитические функции) являются своего рода агрегацией, а не “фильтром” результирующего набора данных запроса. Строки агрегированных данных смешиваются с результирующим набором данных.

Оконные функции используются с предложением **OVER**. Они могут отображаться только в списке **SELECT** или в предложении **ORDER BY** запроса.

Кроме того, с использованием предложения **OVER** оконные функции Firebird могут быть *разделёнными* и *упорядоченными*.

Синтаксис

```
<window function> ::= <window function name>([<expr> [,<expr> ...]])
OVER (
    [PARTITION BY <expr> [, <expr> ...]]
    [ORDER BY <expr>
        [<direction>]
        [<nulls placement>]
        [, <expr> [<direction>] [<nulls placement>] ...]
)
<direction> ::= {ASC | DESC}
<nulls placement> ::= NULLS {FIRST | LAST}
```

Агрегатные функции, используемые в качестве оконных функций

Все агрегатные функции могут использоваться как оконные функции, с использованием предложения **OVER**.

Представьте себе таблицу **EMPLOYEE** с идентификаторами колонок **ID**, **NAME** и **SALARY**, а также необходимость показать каждому сотруднику его зарплату и процентную ставку его зарплаты в общей сумме заработной платы.

С помощью обычного запроса это достигается так:

```
select
    id, department, salary,
    salary/(select sum(salary) from employee) portion
from employee
order by id;
```

Результат

id	department	salary	portion
----	-----	-----	-----
1	R & D	10.00	0.2041

2	SALES	12.00	0.2449
3	SALES	8.00	0.1632
4	R & D	9.00	0.1837
5	R & D	10.00	0.2041

Запрос повторяющийся и длительный для выполнения, особенно если EMPLOYEE является сложным представлением.

Этот же запрос может быть задан более элегантным способом и выполняться значительно быстрее с помощью оконной функции:

```
select
    id, department, salary,
    salary / sum(salary) OVER () portion
from employee
order by id;
```

Здесь `sum(salary) OVER ()` вычисляется один раз как сумма всех зарплат из таблицы EMPLOYEE.

Разделы в оконных функциях

Подобно агрегатным функциям, которые могут работать отдельно или по отношению к группе, оконные функции также могут работать в группе, которая называется «разделом».

Синтаксис

`<window function>(...) OVER (PARTITION BY <expr> [, <expr> ...])`

Агрегация над группой может производиться более, чем с одной строкой данных, поэтому набор результатов, созданный разделом, объединяется с основным запросом, используя тот же список выражений, что и раздел.

Продолжая пример с таблицей EMPLOYEE, вместо того, чтобы получать процент зарплаты каждого сотрудника по сравнению с общей суммой всех сотрудников, мы хотели бы получить часть, основанную только на сотрудниках одного отдела:

```
select
    id, department, salary,
    salary / sum(salary) OVER (PARTITION BY department)
```

```
portion
  from employee
  order by id;
```

Результат

id	department	salary	portion
----	-----	-----	-----
1	R & D	10.00	0.3448
2	SALES	12.00	0.6000
3	SALES	8.00	0.4000
4	R & D	9.00	0.3103
5	R & D	10.00	0.3448

Сортировка в оконных функциях

Предложение ORDER BY может использоваться в оконных функциях с разделами или без разделов и со стандартными агрегатными функциями позволяет возвращать частичные агрегации по мере обработки записей.

Пример

```
select
  id, department, salary,
  salary / sum(salary) OVER (ORDER BY salary) cumul_salary
from employee
order by id;
```

Результат

id	salary	cumul_salary
----	-----	-----
3	8.00	8.00
4	9.00	17.00
1	10.00	37.00
5	10.00	37.00
2	12.00	49.00

Затем cumul_salary возвращает частичную/накопленную (или выполненную) агрегацию (функция SUM). Может показаться странным, что 37.00 повторяется для идентификаторов 1 и 5, но так оно и должно работать.

Предложение ORDER BY сгруппировало ключи id вместе, и агрегация вычисляется один раз (но суммируя два раза 10.00). Чтобы избежать этого, Вы можете добавить поле ID в конце предложения ORDER BY.

Можно использовать несколько окон с разными сортировками, а также части ORDER BY, такие как ASC/DESC и NULLS FIRST/LAST.

С разделами ORDER BY работает одинаково, но на каждой границе раздела агрегация сбрасывается.

С оператором ORDER BY могут использоваться все агрегатные функции за исключением LIST().

Эксклюзивные оконные функции

Помимо агрегатных функций есть ещё эксклюзивные оконные функции, которые в настоящее время разделены на категории *ранжирования* и *навигации*.

Функции ранжирования

Функции ранжирования вычисляют порядковый ранг строки в разделе окна. В эту категорию входят функции DENSE_RANK, RANK и ROW_NUMBER.

Синтаксис

```
<ranking window function> ::=  
    DENSE_RANK() |  
    RANK() |  
    ROW_NUMBER()
```

Функции ранжирования могут использоваться для создания разных типов инкрементных счетчиков. Рассмотрим SUM (1) OVER (ORDER BY SALARY) в качестве примера того, что могут делать функции ранжирования, причем каждая из них по-разному. Ниже приведен пример запроса, а также его сравнение с поведением функции SUM.

```
select  
    id,  
    salary,  
    dense_rank() over (order by salary),  
    rank() over (order by salary),  
    row_number() over (order by salary),
```



```
sum(1) over (order by salary)
from employee
order by salary;
```

Результат

r	id	salary	dense_rank	rank	row_numbe
	sum				
----	-----	-----	-----	-----	-----
	3	8.00	1	1	1
	4	9.00	2	2	2
	1	10.00	3	3	4
	5	10.00	3	4	4
	2	12.00	4	5	5

Разница между `DENSE_RANK` и `RANK` заключается в том, что существует разрыв, связанный с повторяющимися только в `RANK` строками (`order by salary`). `DENSE_RANK` продолжает присваивать порядковые номера после дублирования поля `salary`. С другой стороны, функция ранжирования `ROW_NUMBER` всегда назначает порядковые номера, даже если есть повторяющиеся значения.

Навигационные функции

Навигационные функции получают простое (не агрегированное) значение выражения из другой строки запроса в том же разделе.

Синтаксис

```
<navigational window function> ::=
    FIRST_VALUE(<expr>) |
    LAST_VALUE(<expr>) |
    NTH_VALUE(<expr>, <offset>) [FROM FIRST | FROM LAST] |
    LAG(<expr> [ [, <offset> [, <default> ] ] ) |
    LEAD(<expr> [ [, <offset> [, <default> ] ] )
```

Важно

`FIRST_VALUE`, `LAST_VALUE` и `NTH_VALUE` также работают в рамках оконной функции. В настоящее время Firebird всегда

обращается от первой к текущей строке раздела, а не к последней. Вероятно, это приводит к появлению странных результатов для NTH_VALUE и особенно LAST_VALUE.

Пример

```
select
  id, salary,
  first_value(salary) over (order by salary),
  last_value(salary) over (order by salary),
  nth_value(salary, 2) over (order by salary),
  lag(salary) over (order by salary),
  lead(salary) over (order by salary)
from employee
order by salary;
```

Результат

id	salary	first_value	last_value	nth_value	lag	lead
—	-----	-----	-----	-----	----	-----
3	8.00	8.00	8.00	<null>	<null>	9.00
4	9.00	8.00	9.00	9.00	8.00	10.00
1	10.00	8.00	10.00	9.00	9.00	10.00
5	10.00	8.00	10.00	9.00	10.00	12.00
2	12.00	8.00	12.00	9.00	10.00	<null>

FIRST_VALUE и LAST_VALUE получают, соответственно, первое и последнее значение упорядоченного раздела.

NTH_VALUE получает n-ое значение, начиная с первой (по умолчанию) или последней записи, из упорядоченного раздела.

Смещение на 1 от первого значения было бы эквивалентно FIRST_VALUE, а смещение на 1 от последнего значения было бы эквивалентно LAST_VALUE.

LAG ищет предыдущую строку, а LEAD - следующую строку. LAG и LEAD получают их значения на расстоянии от текущей строки с переданным смещением (значение которого по умолчанию равно 1). В случае, когда смещение указывает за пределы раздела, возвращается параметр по умолчанию, т.е. NULL.

Вывод расширенного плана

Дмитрий Еманов

Оператор **PLAN** теперь может выводить данные в более структурированной и понятной форме, например:

```
SELECT statement
  -> First [10]
    -> Sort [SUM, O_ORDERDATE]
      -> Aggregate
        -> Sort [L_ORDERKEY, O_ORDERDATE,
                  O_SHIPPRIORITY]
          -> Inner Loop Join
            -> Filter
              -> Table #ORDERS# Access By ID
                -> Bitmap
                  -> Index #ORDERS_ORDERDATE# Range
                      Scan
            -> Filter
              -> Table #CUSTOMER# Access By ID
                -> Bitmap
                  -> Index #CUSTOMER_PK# Unique Scan
            -> Filter
              -> Table #LINEITEM# Access By ID
                -> Bitmap
                  -> Index #LINEITEM_PK# Unique Scan
```

Вывод расширенного плана в isql

Для отображения этой опции в утилиту *isql* был добавлен новый синтаксис **SET EXPLAIN [ON | OFF]**. Более детально об этом об этом изложено в разделе «*Расширение SET EXPLAIN для детального просмотра планов*» главы 11 «*Мониторинг и утилиты командной строки*».

Внутренние функции

Дополнениями и усовершенствованиями набора внутренних функций являются:

SUBSTRING в регулярных выражениях

Adriano dos Santos Fernandes

Теперь поиск подстроки может использовать регулярное выражение.

Шаблон поиска

SUBSTRING(<string>
[NOT] SIMILAR TO <pattern> ESCAPE <char>)

Обсуждение смотрите здесь — [***CORE-2006***](#).

Дополнительные сведения об использовании выражений SIMILAR TO приведены в файле *README.similar_to.txt* в подкаталоге */doc/* вашей установки Firebird.

Подсказка

Используемое регулярное выражение - это SQL. Руководство доступно в главе DML в замечаниях к выпуску Firebird 2.5, а [также на веб-сайте Firebird](#) на английском и в разделе «SIMILAR TO: поиск по регулярным выражениям» в [русской версии замечаний к выпуску Firebird 2.5](#).

Обратные гиперболические тригонометрические функции

Claudio Valderrama C.

Реализованы шесть обратных гиперболических тригонометрических функций:

ACOSH

Возвращает гиперболический косинус числа (выраженного в радианах). Формат: ACOSH (<число>).

ASINH

Возвращает гиперболический синус числа (выраженного в радианах). Формат: ASINH (<число>).

ATANH

Возвращает гиперболический тангенс числа (выраженного в радианах).
Формат: ATANH (<число>).

COSH

Возвращает гиперболический косинус угла (выраженного в числовом виде). Формат: COSH (<число>)

SINH

Возвращает гиперболический синус угла (выраженного в числовом виде). Формат: SINH (<число>)

TANH

Возвращает гиперболический тангенс угла (выраженного в числовом виде). Формат: TANH (<число>)

Статистические функции

Hajime Nakagami
Adriano dos Santos Fernandes

Реализован набор статистических функций, совместимых со стандартами SQL.

Агрегатные статистические функции

Функции включают в себя несмещённую выборочную дисперсию (*Sample Variance*), выборочную дисперсию для группы (*Population Variance*), выборочное стандартное отклонение (*Sample Standard Deviation*), среднеквадратичное отклонение для группы (*Population Standard Deviation*), выборочную ковариацию (*Sample Covariance*), ковариацию совокупности (*Population Covariance*), и коэффициент корреляции (*Coefficient of Correlation*). См. [**CORE-4717**](#).

Синтаксис

<single param statistical function> ::=
 <single param statistical function name>(<expr>)

<single param statistical function name> :=
 { VAR_POP | VAR_SAMP | STDDEV_POP | STDDEV_SAMP }

<dual param statistical function> :=
 <dual param statistical function name>(<expr1>, <expr>)

<dual param statistical function name> :=
 { COVAR_POP | COVAR_SAMP | CORR }

Семантика

- Функции VAR_SAMP, STDDEV_SAMP и COVAR_SAMP возвращают значение NULL, если выборка данных пустая или содержит только одну запись;
- Функции VAR_POP, STDDEV_POP, COVAR_POP и CORR возвращают значение NULL, если выборка данных пустая.

Примеры

SELECT STDDEV_SAMP(salary) FROM employees;

SELECT CORR(salary, record_of_service) FROM employees;

Таблица 9.1. Описание статистических функций

Функция	Формат	Описание
VAR_SAMP	VAR_SAMP(<expr>)	Возвращает несмещённую выборочную дисперсию. Эквивалентно:
$\frac{(\text{SUM}(\langle \text{expr} \rangle^2) - \text{SUM}(\langle \text{expr} \rangle)^2 / \text{COUNT}(\langle \text{expr} \rangle))}{(\text{COUNT}(\langle \text{expr} \rangle) - 1)}$		
VAR_POP	VAR_POP(<expr>)	Возвращает выборочную дисперсию. Эквивалентно:
$\frac{(\text{SUM}(\langle \text{expr} \rangle^2) - \text{SUM}(\langle \text{expr} \rangle)^2 / \text{COUNT}(\langle \text{expr} \rangle))}{\text{COUNT}(\langle \text{expr} \rangle)}$		
STDDEV_SAMP	STDDEV_SAMP(<expr>)	Возвращает выборочное стандартное отклонение.

		Эквивалентно:
$\text{SQRT}(\text{VAR_SAMP}(\langle \text{expr} \rangle))$		
STDDEV_POP	STDDEV_POP($\langle \text{expr} \rangle$)	Возвращает среднеквадратичное отклонение. Эквивалентно:
$\text{SQRT}(\text{VAR_POP}(\langle \text{expr} \rangle))$		
COVAR_SAMP	COVAR_SAMP($\langle \text{expr1} \rangle$, $\langle \text{expr2} \rangle$)	Возвращает выборочную ковариацию. Эквивалентно:
$(\text{SUM}(\langle \text{expr1} \rangle * \langle \text{expr2} \rangle) - \text{SUM}(\langle \text{expr1} \rangle) * \text{SUM}(\langle \text{expr2} \rangle) / \text{COUNT}(*)) / (\text{COUNT}(*)) - 1)$		
COVAR_POP	COVAR_POP($\langle \text{expr1} \rangle$, $\langle \text{expr2} \rangle$)	Возвращает ковариацию совокупности. Эквивалентно:
$(\text{SUM}(\langle \text{expr1} \rangle * \langle \text{expr2} \rangle) - \text{SUM}(\langle \text{expr1} \rangle) * \text{SUM}(\langle \text{expr2} \rangle) / \text{COUNT}(*)) / \text{COUNT}(*))$		
CORR	CORR($\langle \text{expr1} \rangle$, $\langle \text{expr2} \rangle$)	Возвращает коэффициент корреляции. Эквивалентно:
$\text{COVAR_POP}(\langle \text{expr1} \rangle, \langle \text{expr2} \rangle) / (\text{STDDEV_POP}(\langle \text{expr2} \rangle) * \text{STDDEV_POP}(\langle \text{expr1} \rangle))$		

Функции линейной регрессии

Набор функций REGR_* анализирует отношения между двумя наборами числовых данных, учитывая только значения, отличные от NULL, в любом выражении ([CORE-4722](#)).

Синтаксис

$\langle \text{regr function} \rangle ::= \langle \text{function name} \rangle(\langle \text{expr1} \rangle, \langle \text{expr2} \rangle)$

$\langle \text{function name} \rangle := \{ \text{REGR_AVGX} \mid \text{REGR_AVGY} \mid \text{REGR_COUNT} \mid \text{REGR_INTERCEPT} \mid \text{REGR_R2} \mid \text{REGR_SLOPE} \mid \text{REGR_SXX} \mid \text{REGR_SXY} \mid \text{REGR_SYY} \}$

Формулы

Y: <expr1> (<expr1> IS NOT NULL AND <expr2> IS NOT NULL)

X: <expr2> (<expr1> IS NOT NULL AND <expr2> IS NOT NULL)

N =SUM(CASE WHEN x IS NOT NULL AND
y IS NOT NULL THEN 1 END)

Важно

В синтаксисе функций **y** интерпретируется в качестве переменной, зависящей от **x**.

Любую из функций линейной регрессии можно использовать в качестве оконной.

Y и X имеют тип DOUBLE PRECISION. N имеет тип SMALLINT, INTEGER или BIGINT.

Все функции исключают пары выражений, где одно из выражений в паре имеет значение NULL. Если в выборке нет строк, то функции (за исключением REGR_COUNT()) возвращают значение NULL.

Таблица 9.2. Описание функций линейной регрессии

Функция	Формат	Описание
REGR_AVGX	REGR_AVGX(Y, X)	Вычисляет среднее независимой переменной линии регрессии. Возвращаемое значение имеет тип DOUBLE PRECISION
Формула: REGR_AVGX(Y, X) = SUM(X) / N		
REGR_AVGY	REGR_AVGY(Y, X)	Вычисляет среднее зависимой переменной линии регрессии.

		Возвращаемое значение имеет тип DOUBLE PRECISION
Формула: $\text{REGR_AVGY}(Y, X) = \text{SUM}(Y) / N$		
REGR_COUNT	REGR_COUNT(Y, X)	Возвращает количество непустых пар, используемых для создания линии регрессии. В случае когда после исключения пар, в которых любое выражение имеет значение NULL, строк не остаётся, функция возвращает 0. Возвращаемое значение имеет тип SMALLINT, INTEGER или BIGINT
Формула: $\text{REGR_COUNT}(Y, X) = N$		
REGR_INTERCEPT	REGR_INTERCEPT(Y, X)	Вычисляет точку пересечения линии регрессии с осью Y
Формула: $\text{REGR_INTERCEPT}(Y, X) = \text{REGR_AVGY}(Y, X) - \text{REGR_SLOPE}(Y, X) * \text{REGR_AVGX}(Y, X)$		
REGR_R2	REGR_R2(Y, X)	Вычисляет коэффициент детерминации, или R-квадрат, линии регрессии
Формула: $\text{REGR_R2}(Y, X) = \text{POWER}(\text{CORR}(Y, X), 2)$		
REGR_SLOPE	REGR_SLOPE(Y, X)	Вычисляет угол наклона линии регрессии

Формула: $\text{REGR_SLOPE}(Y, X) = \text{COVAR_POP}(Y, X) / \text{VAR_POP}(X)$		
REGR_SXX	REGR_SXX(Y, X)	Возвращает сумму квадратов независимого выражения (y) в паре выражений (y и X)
Формула: $\text{REGR_SXX}(Y, X) = N * \text{VAR_POP}(X)$		
REGR_SXY	REGR_SXY(Y, X)	Возвращает сумму произведений независимого выражения, умноженное на зависимое выражение в паре выражений (Y и X)
Формула: $\text{REGR_SXY}(Y, X) = N * \text{COVAR_POP}(Y, X)$		
REGR_SYY	REGR_SYY(Y, X)	Возвращает сумму квадратов зависимого выражения в паре выражений (Y и X)
Формула: $\text{REGR_SYY}(Y, X) = N * \text{VAR_POP}(Y)$		

Снято ограничение в 32 кБ для аргумента в функции TRIM() BLOB

Adriano dos Santos Fernandes

В предыдущих версиях TRIM (подстрока из строки) допускала BLOB для обоих аргументов, но первый аргумент должен был быть меньше 32 КБ. Теперь оба аргумента могут принимать BLOB размером до 4 ГБ.

Корректировка лимита строковых литералов

Adriano dos Santos Fernandes

Внутренняя длина строки на некоторых уровнях может составлять почти 64 КБ. Проверки записи строк в текстовый BLOB показали, что можно безопасно принимать строковый литерал именно такого размера.

Соответственно:

1. “Предел безопасности” на длину строки литерала (32 КБ - 3) для записи в текстовые BLOB был повышен до 65 533 байт (64 КБ-3);
2. Для недопущения превышения лимита для строк в многобайтовых наборах символов ограничение в символах вычисляется во время выполнения. Например, для строки UTF8 (максимум в 4 байта/символ) предел, вероятно, будет около (слово $(65533/4) = 16383$ символа).

См. [*CORE-4881*](#).

Улучшения функции DATEADD()

Adriano dos Santos Fernandes

Улучшения внутренней функции DATEADD():

- Теперь функция поддерживает дробное значение в миллисекундах ([*CORE-4457*](#));
- Тип входных данных аргументов <сумма> изменен с целого на bigint ([*CORE-4310*](#)).

Улучшения в языке DML

В выпуске Firebird 3 добавлено много полезных улучшений DML.

Альтернатива кавычкам в строковых литералах

Adriano dos Santos Fernandes

Теперь можно использовать символ или пару символов для замены используемых ранее двойных кавычек (удвоенного экранированного апострофа) для строки с кавычкой внутри. Ключевое слово **q** или **Q** в начале строки сообщает парсеру, что строка содержит внутри себя кавычки, являющиеся разделителями строкового литерала.

Синтаксис

```
<alternate string literal> ::=  
    { q | Q } <quote> <alternate start char>
```

[{ <char> }...] <alternate end char> <quote>

Правила

Когда <alternate start char> содержит символы '(', '{', '[' или '<', <alternate end char> находится в паре со своим соответствующим «партнёром», а именно: ')', '}', ']' и '>'. В других случаях символ <alternate end char> совпадает с <alternate start char>.

Внутри строки, т.е. <char>, могут использоваться одиночные (не экранированные) кавычки. При этом каждая кавычка будет частью результирующей строки.

Примеры

```
select q'{abc{def}ghi}' from rdb$database;  
-- результат: abc{def}ghi  
  
select q'!That's a string!' from rdb$database;  
-- результат: That's a string
```

Операторы OFFSET и FETCH совместимы с SQL-2008

Mark Rotteveel

Совместимые с SQL-2008, предложения OFFSET и FETCH обеспечивают стандартный эквивалент для операторов FIRST и SKIP, а также альтернативу для оператора ROWS ... TO для выборки набора данных.

- Предложение OFFSET указывает число строк, которые нужно пропустить;
- Предложение FETCH указывает количество строк для выборки.

Как и в случае с операторами SKIP и FIRST, предложения OFFSET и FETCH могут применяться независимо, как в выражениях верхнего уровня, так и во вложенных запросах. Они доступны в PSQL и DSQL.

Синтаксис

```
SELECT ... [ ORDER BY <expr_list> ]  
          [ OFFSET <simple_value_expr> { ROW | ROWS } ]
```

[FETCH { FIRST | NEXT } [<simple_value_expr>]
{ ROW | ROWS } ONLY]

<simple_value_expr> - (числовой) литерал, параметр DSQL (?) или именованный параметр в PSQL (:name_parameter), который сопоставляется с типом данных INTEGER.

Примеры

```
-- 1:
SELECT *
FROM T1 ORDER BY COL1
      OFFSET 10 ROWS;

-- 2:
SELECT *
FROM T1 ORDER BY COL1
      FETCH FIRST 10 ROWS ONLY;

-- 3:
SELECT * FROM (
      SELECT * FROM T1 ORDER BY COL1 DESC
              OFFSET 1 ROW
              FETCH NEXT 10 ROWS ONLY
    ) a
ORDER BY a.COL1
      FETCH FIRST ROW ONLY;
```

Примечания

- Предложения FIRST/SKIP и ROWS являются нестандартными альтернативами;
- Предложения OFFSET и/или FETCH нельзя смешивать с альтернативными предложениями FIRST/SKIP или ROWS в одном выражении для запроса;
- Не допускается использование выражений и ссылок на столбцы ни в OFFSET, ни в FETCH;
- В отличие от оператора ROWS, предложения OFFSET и FETCH доступны только в операторах SELECT;
- Не поддерживается «Процент FETCH», определенный в стандарте SQL;

- Не поддерживается «FETCH ... WITH TIES», определенный в стандарте SQL.

Запрет одновременного использования явных и неявных соединений

Дмитрий Еманов

Хотя смешивание синтаксиса неявных и явных соединений не рекомендуется, анализатор, тем не менее, позволял это сделать. Некоторые "смешивания" на самом деле вызывают выдачу оптимизатором непредвиденных результатов, в том числе ошибку - "Нет записей для выборки". Такие смешивания запрещены в других серверах SQL, теперь они запрещены в Firebird.

Дискуссию на эту тему см. [*CORE-2812*](#).

Поддержка параметров в левой части предложения WHERE

Adriano dos Santos Fernandes

Подзапрос с параметром в левой части условия WHERE ... IN (SELECT ...) завершался раньше с ошибкой «Тип данных параметра неизвестен».

Теперь такой стиль разрешён. Например:

```
SELECT <columns> FROM table_1 t1
      WHERE <conditions on table_1> AND
             (? IN (SELECT some_col
                    FROM table_2 t2
                    WHERE t1.id = t2.ref_id))
```

Важно

В этих случаях лучшей практикой кодирования SQL будет заключаться в использовании EXISTS; Однако разработчики сталкивались с этой проблемой при использовании сгенерированного в Hibernate SQL, который использовал этот нежелательный стиль.

Улучшения для предложения RETURNING

Adriano dos Santos Fernandes

В предложение RETURNING добавлены два улучшения:

Предложение RETURNING может использовать псевдонимы

При использовании предложения RETURNING для возврата значения клиенту значение теперь может быть передано с помощью псевдонима.

Пример без алиаса и с алиасом

```
-- без алиаса
UPDATE T1 SET F2 = F2 * 10
  RETURNING OLD.F2, NEW.F2;

-- с алиасами
UPDATE T1 SET F2 = F2 * 10
  RETURNING OLD.F2 OLD_F2, NEW.F2 AS NEW_F2;
```

Примечание

Ключевое слово AS является необязательным.

Предложение RETURNING для позиционных обновлений и удалений

Добавлена поддержка для предложения RETURNING в позиционных (WHERE CURRENT OF) операторах UPDATE и DELETE.

Пример

```
UPDATE T1 SET F2 = F2 * 10
  WHERE CURRENT OF C
  RETURNING NEW.F2 NEW_F2;
```

Устойчивость курсора

Владислав Хорсун

До этого выпуска Firebird страдал от печально известной ошибки, при которой операция модификации данных могла заикливаться и, в зависимости от операции, удалить все строки в таблице, продолжить обновление одних и тех же строк до бесконечности или вставлять строки до тех пор, пока на сервере не закончатся ресурсы. Затронуты были все операторы DML - INSERT, UPDATE, DELETE, MERGE. Это происходило по причине использования ядром сервера для этих операций неявного курсора.

Для обеспечения стабильности строки, которые необходимо вставить, обновить или удалить, должны быть каким-либо образом отмечены, чтобы избежать многократного повторения операции. Другой обходной путь заключается в том, чтобы заставить запрос иметь сортировку в плане для материализации курсора.

Начиная с Firebird 3 ядро сервера использует журнал отмены для проверки, была ли строка уже вставлена, изменена или удалена текущим курсором.

Важно

Такая стабилизация *НЕ* работает с циклами в PSQL, использующими оператор SUSPEND.

Улучшение GTT

Владислав Хорсун

Глобальные временные таблицы (GTT) теперь доступны для записи даже для транзакций только для чтения. Эффект этого следующий.

Транзакция только для чтения в базе данных для чтения и записи

Доступна для записи данных как в ON COMMIT PRESERVE ROWS, так и в ON COMMIT DELETE ROWS.

Транзакция только для чтения в базе данных только для чтения

Доступна для записи данных только в ON COMMIT DELETE ROWS.

Также

- Откат (Rollback) для GTT ON COMMIT DELETE ROWS работает быстрее;
- Строки не нуждаются в резервном копировании при откате (Rollback);
- Сборка мусора в GTT не задерживается активными транзакциями из других соединений с базой данных.

Примечание

Эти изменения были также портированы в Firebird 2.5.1.

Улучшение для строк DML

Adriano dos Santos Fernandes

Строки в запросах DML теперь преобразуются или проверяются для избежания передачи неверных строк внутренним компонентам ядра сервера, например, в столбец MON\$STATEMENTS.MON\$SQL_TEXT.

Принятие решение зависит от набора символов, используемого в подключении к базе данных.

- Для кодировкой NONE - non-ASCII символы преобразуются в вопросительные знаки;
- В других кодировках строки проверяются на наличие неправильных символов.

Функция COUNT() возвращает тип BIGINT

Функция COUNT() теперь возвращает результат типа BIGINT вместо типа INTEGER.

Оптимизация

Оптимизация для этого выпуска включает в себя:

Улучшения для SIMILAR TO

Adriano dos Santos Fernandes

Улучшена производительность оператора SIMILAR TO.

OR'ed параметр в предложении WHERE

Дмитрий Еманов

Увеличена производительность для (table.field =: param or : param = -1) в предложении WHERE.

Лучший выбор для навигации

Дмитрий Еманов

Ранее при наличии в плане предложения ORDER в операторе SELECT оптимизатор выбирал первый индекс из предложения ORDER BY или GROUP BY. Подход "*первый пришёл*" является не самым лучшим в случаях доступности нескольких вариантов индекса. Теперь Firebird 3 просматривает все доступные варианты и выбирает наиболее подходящий индекс.

См. [CORE-4285](#).

Более простое выполнение запросов, использующих предложение UNION

Дмитрий Еманов

Ранее выполнение запросов с использованием UNION было иерархическим, что часто приводило к избыточному чтению. Проведённая оптимизация заменяет иерархический путь выполнения более простым решением, повышающим производительность.

См. [CORE-4165](#).

Пропуск индекса для использования составного индекса

Дмитрий Еманов

Оптимизатор теперь позволяет пропустить индекс (при плане с использованием ORDER), когда доступен подходящий индекс соединения (A, B) с условием запроса вида WHERE A =? ORDER B.

См. [*CORE-1846*](#).

Улучшение производительности для SET STATISTICS INDEX

Владислав Хорсун

Для вычисления селективности индекса BTR_selectivity() теперь будет проходить весь уровень листа заданного индекса b-дерева. На протяжении всего процесса операция дискового ввода-вывода происходит только после пересчёта статистики. Эффект заключается в том, чтобы исключить долгого ожидания окончания запросов AST из параллельных приложений, таких как запросы блокировки страниц, мониторинг, отмена и т.п. Улучшение в Firebird 3 возможно решило эту проблему.

См. [*CORE-1846*](#).

Небольшое улучшение для 1-го диалекта

Adriano dos Santos Fernandes

Для 1-го диалекта включена поддержка SQL_INT64, SQL_DATE и SQL_TIME.

См. [*CORE-3972*](#).

Улучшения встроенного SQL (ESQL)

Дмитрий Еманов

В этом выпуске в подмножество Embedded SQL были включены два улучшения:

Поддержка оператора UPDATE OR INSERT

См. [*CORE-4438*](#).

Поддержка предложения RETURNING

См. [*CORE-4437*](#).

Глава 10

Процедурный SQL (PSQL)

Улучшения в процедурном SQL (PSQL) включают в себя:

Быстрые ссылки для PSQL

- Сохранённые функции PSQL;
- Подпрограммы PSQL;
- Пакеты;
- Триггер для DDL;
- Исключения с параметрами;
- Использование CONTINUE в цикле;
- Стабилизация курсора в PSQL;
- Переменная типа курсор в PSQL;
- Использование SQLSTATE в обработчиках исключений;
- Увеличен размера тела PSQL.

Сохранённые функции PSQL

Дмитрий Еманов

Теперь в PSQL можно написать скалярную функцию и вызвать ее так же, как и внутреннюю функцию.

Синтаксис для DDL

```
{CREATE [OR ALTER] | ALTER | RECREATE} FUNCTION <name>
    [(param1 [, ...])]
RETURNS <type>
AS
BEGIN
...
END
```

Подсказка

Синтаксис оператора **CREATE** представляет собой объявление функции PSQL, аналогичный устаревшему **DECLARE** для UDF.

Пример

```
CREATE FUNCTION F(X INT) RETURNS INT
AS
BEGIN
    RETURN X+1;
END;

SELECT F(5) FROM RDB$DATABASE;
```

Подпрограммы PSQL

Adriano dos Santos Fernandes

Заголовок модуля PSQL (хранимая процедура, хранимая функция, триггер, исполняемый блок) теперь может принимать подпрограммы и суб-функции в объявлениях для использования в теле модуля.

Синтаксис объявления подпрограмм

```
DECLARE PROCEDURE <name> [(param1 [, ...])]
[RETURNS (param1 [, ...])]
AS
...
```

Синтаксис объявления суб-функций

```
DECLARE FUNCTION <name> [(param1 [, ...])]
RETURNS <type>
AS
...
```

Примеры

```
SET TERM ^;
--
-- суб-функция в EXECUTE BLOCK
--
EXECUTE BLOCK RETURNS (N INT)
AS
    DECLARE FUNCTION F(X INT) RETURNS INT
    AS
    BEGIN
        RETURN X+1;
    END
BEGIN
    N = F(5);
    SUSPEND;
END ^
--
-- суб-функция внутри а хранимой функции
--
CREATE OR ALTER FUNCTION FUNC1 (n1 INTEGER, n2
    INTEGER)
RETURNS INTEGER
AS
    DECLARE FUNCTION SUBFUNC (n1 INTEGER, n2
        INTEGER)
    RETURNS INTEGER
    AS
    BEGIN
        RETURN n1 + n2;
    END
BEGIN
    RETURN SUBFUNC(n1, n2);
END ^
--
select func1(5, 6) from rdb$database ^
```

Пакеты

Adriano dos Santos Fernandes

Благодарность

Эта работа спонсировалась пожертвованиями, собранными на пятой конференции разработчиков Firebird в Бразилии в 2008 году.

Пакет - это группа процедур и функций, управляемых как единое целое. Понятие "*пакетов*" компонентов кода операций в базе данных решает несколько задач:

Модульное построение

Идея пакетов заключается в разделении блоков взаимозависимого кода на логические модули, как это делается в языках программирования.

В программировании давно признано что, например, группирование кода различными способами - в пространствах имен, модулях или классах, является хорошим стилем. При использовании стандартных процедур и функций в базе данных это невозможно. Хотя они могут быть сгруппированы в различные сценарии, остаются две проблемы:

1. Группировка не представлена в метаданных базы данных;
2. Все стандартные процедуры участвуют в плоском пространстве имен и вызываются всеми (здесь не имеется в виду разрешения безопасности).

Упрощение отслеживания зависимостей

Создание механизма отслеживания зависимостей между коллекцией связанных внутренних процедур, а также между этой коллекцией и другими процедурами (и в пакетах, и вне пакетов).

Пакеты Firebird состоит из двух частей: заголовок (ключевое слово **PACKAGE**) и тело (ключевое слово **PACKAGE BODY**). Это разделение очень похоже на блок Delphi - заголовок, определяющую интерфейсную часть, и тело, определяющее его реализацию.

Сначала создаётся заголовок пакета (**CREATE PACKAGE**), а затем его тело (**CREATE PACKAGE BODY**).

Каждый раз, когда пакет определяет, что используется некоторый объект базы данных, информации о зависимости от этого объекта регистрируется в системных таблицах Firebird. Это означает, что для удаления или изменения этого объекта Вы сначала должны удалить все

объекты, зависящие от него. Поскольку зависимости от других объектов существуют только для тела пакета, то тело пакета может быть легко удалено (на самом деле тело пакета обычно делают комментарием для упрощения последующего восстановления), даже если какой-нибудь другой объект зависит от этого пакета. Тело пакета удаляется (закомментировано), а заголовок остаётся, что позволяет пересоздать пакет после того, как сделаны изменения, связанные с удалёнными объектами, от которых зависел.

Упрощение управления правами

Хорошей практикой является создание процедур/подпрограмм с предоставлением прав на их выполнение роли/ролям и/или пользователям. Firebird выполняет процедуры/подпрограммы с привилегиями вызывающего абонента, а это означает необходимость предоставлять использование ресурсов каждой процедуре, когда эти ресурсы не будут напрямую доступны для вызывающего их пользователя. Таким образом для каждой использованной в пакете процедуры/подпрограммы должны быть предоставлены права пользователям и/или ролям.

Подпрограммы пакета не имеют индивидуальных прав на выполнение. Привилегии действуют на весь пакет в целом. Привилегии, предоставленные пакетам, действительны для всех подпрограмм тела пакета, в том числе и частных, и сохраняются для заголовка пакета.

Примеры

```
GRANT SELECT ON TABLE secret TO PACKAGE pk_secret;  
GRANT EXECUTE ON PACKAGE pk_secret TO ROLE role_secret;
```

«Частные области» видимости

Это частная (*private*) область в процедуре, доступная для использования только в текущем пакете. Пакеты Firebird аналогичны модулям Delphi. Если подпрограмма не объявлена в заголовке пакета (*interface*), но реализована в теле (*implementation*) пакета, то она становится частной (*private*). Частную подпрограмму можно вызвать только из пакета, в котором она объявлена.

Подписи (сигнатуры) пакетов

Все подпрограммы, принадлежащие пакету, хранят элементы цифровой подписи (в виде набора [имя подпрограммы, параметры и возвращаемый тип данных]) в системных таблицах

Подпись процедуры/подпрограммы может быть запрошена следующим образом:

```
SELECT...  
    -- пример запроса
```

Синтаксис пакетов

```
<package_header> ::=  
{ CREATE [OR ALTER] | ALTER | RECREATE } PACKAGE <name>  
AS  
BEGIN  
    [ <package_item> ... ]  
END
```

```
<package_item> ::=  
    <function_decl> ; |  
    <procedure_decl> ;
```

```
<function_decl> ::=  
    FUNCTION <name> [( <parameters> )] RETURNS <type>
```

```
<procedure_decl> ::=  
    PROCEDURE <name> [( <parameters> )  
        [RETURNS ( <parameters> )]]
```

```
<package_body> ::=  
{ CREATE | RECREATE } PACKAGE BODY <name>  
AS  
BEGIN  
    [ <package_item> ... ]  
    [ <package_body_item> ... ]  
END
```

```
<package_body_item> ::=  
    <function_impl> |  
    <procedure_impl>
```

```
<function_impl> ::=  
FUNCTION <name> [( <parameters> )] RETURNS <type>  
AS
```

```
BEGIN
...
END
|
FUNCTION <name> [( <parameters> )]
    RETURNS <type>
    EXTERNAL NAME '<name>' ENGINE <engine>

<procedure_impl> ::=
PROCEDURE <name> [( <parameters> )
    [RETURNS ( <parameters> )]]
AS
BEGIN
...
END
|
PROCEDURE <name> [( <parameters> )
    [RETURNS ( <parameters> )]]
    EXTERNAL NAME '<name>' ENGINE <engine>

<drop_package_header> ::=
    DROP PACKAGE <name>

<drop_package_body> ::=
    DROP PACKAGE BODY <name>
```

Синтаксические правила

- Все процедуры, объявленные в заголовке и в начале тела, должны быть реализованы в теле пакета с одной и той же подписью (см. [Подписи \(сигнатуры\) пакетов](#)); это означает, что нельзя объявлять подпрограмму по-разному в заголовке и в теле;
- Значения по умолчанию для параметров процедуры не могут быть переопределены в разделах <package_item> и <package_body_item>. Они могут находиться в <package_body_item> только для частных (*private*) процедур, которые не доступны вне пакета.

Совет

- Оператор DROP PACKAGE удаляет сначала тело пакета и только после этого его заголовок;

- Тело пакета сохраняется после выполнения команды ALTER/RECREATE PACKAGE. Флаг в столбце RDB\$PACKAGES.RDB\$VALID_BODY_FLAG отображает состояние тела пакета (см. [CORE-4487](#));
- В настоящее время объявления UDF (DECLARE EXTERNAL FUNCTION) внутри пакетов не поддерживаются;
- Внутри пакетов доступен синтаксис предложения COMMENT ON для описания процедур, функций и их параметров ([CORE-4484](#)).

Примеры пакетов

Простой пример пакета

```
SET TERM ^;
-- заголовок пакета, только декларация
CREATE OR ALTER PACKAGE TEST
AS
BEGIN
    -- публичная процедура
    PROCEDURE P1(I INT) RETURNS (O INT);
END

-- тело пакета, реализация
RECREATE PACKAGE BODY TEST
AS
BEGIN
    -- частная (private) функция
    FUNCTION F1(I INT) RETURNS INT;
    PROCEDURE P1(I INT) RETURNS (O INT)
    AS
        BEGIN
        END
    FUNCTION F1(I INT) RETURNS INT
    AS
        BEGIN
            RETURN 0;
        END
END ^
```

Пример пакета для работы с сессионными переменными

```
SET TERM ^;
CREATE PACKAGE APP_VAR
AS
BEGIN
    -- Возвращает дату начала периода
    -- Функция помечена как детерминированная, что позволяет
    -- рассматривать её как инвариант в запросах
    FUNCTION      GetDateBegin()      RETURNS      DATE
DETERMINISTIC;

    -- Возвращает дату окончания периода
    -- Функция помечена как детерминированная, что позволяет
    -- рассматривать её как инвариант в запросах
    FUNCTION GetDateEnd() RETURNS DATE DETERMINISTIC;

    -- Устанавливает диапазон дат рабочего периода
    PROCEDURE SetDateRange(ADateBegin DATE, ADateEnd
DATE);
END^

CREATE PACKAGE BODY APP_VAR
AS
BEGIN
    -- Возвращает дату начала периода
    FUNCTION GetDateBegin() RETURNS DATE
    AS
    BEGIN
        RETURN
        RDB$GET_CONTEXT('USER_SESSION',
'DATEBEGIN');
    END

    -- Возвращает дату окончания периода
    FUNCTION GetDateEnd() RETURNS DATE
    AS
    BEGIN
        RETURN
        RDB$GET_CONTEXT('USER_SESSION', 'DATEEND');
```

```
END

-- Устанавливает диапазон дат рабочего периода
PROCEDURE SetDateRange(ADateBegin DATE, ADateEnd
DATE)
AS
BEGIN
    RDB$SET_CONTEXT('USER_SESSION',
        'DATEBEGIN', ADateBegin);
    RDB$SET_CONTEXT('USER_SESSION',
        'DATEEND', ADateEnd);
END
END^
SET TERM ;^
```

Использование:

```
-- Установка рабочего периода
EXECUTE PROCEDURE APP_VAR.SetDateRange(
    date '01.01.2017', date '31.12.2017');

-- Использование в запросах
SELECT *
    FROM SALES_ORDER
WHERE bydate
    BETWEEN APP_VAR.GetDateBegin() AND
        APP_VAR.GetDateEnd();
```

Примечание

Дополнительные примеры можно найти в директории установки
Firebird ../examples/package/

Триггер для DDL

Adriano dos Santos Fernandes

Благодарность

Эта работа спонсировалась пожертвованиями, собранными на пятой конференции разработчиков Firebird в Бразилии в 2008 году.

Основной целью создания «Триггера для DDL» являлось включение ограничений для пользователей, которые пытаются создать, изменить или удалить объект DDL, а также ведение журнала изменений метаданных.

Синтаксис

```
<database-trigger> ::=  
{CREATE | RECREATE | CREATE OR ALTER}  
TRIGGER <name>  
[ACTIVE | INACTIVE]  
{BEFORE | AFTER} <ddl event>  
    [POSITION <n>]  
AS  
BEGIN  
    ...  
END
```

```
<ddl event> ::=  
ANY DDL STATEMENT  
| <ddl event item> [{OR <ddl event item>}...]  
<ddl event item> ::=  
CREATE TABLE  
    | ALTER TABLE  
    | DROP TABLE  
    | CREATE PROCEDURE  
    | ALTER PROCEDURE  
    | DROP PROCEDURE  
    | CREATE FUNCTION  
    | ALTER FUNCTION  
    | DROP FUNCTION  
    | CREATE TRIGGER  
    | ALTER TRIGGER  
    | DROP TRIGGER  
    | CREATE EXCEPTION  
    | ALTER EXCEPTION  
    | DROP EXCEPTION  
    | CREATE VIEW
```

| ALTER VIEW
| DROP VIEW
| CREATE DOMAIN
| ALTER DOMAIN
| DROP DOMAIN
| CREATE ROLE
| ALTER ROLE
| DROP ROLE
| CREATE SEQUENCE
| ALTER SEQUENCE
| DROP SEQUENCE
| CREATE USER
| ALTER USER
| DROP USER
| CREATE INDEX
| ALTER INDEX
| DROP INDEX
| CREATE COLLATION
| DROP COLLATION
| ALTER CHARACTER SET
| CREATE PACKAGE
| ALTER PACKAGE
| DROP PACKAGE
| CREATE PACKAGE BODY
| ALTER PACKAGE BODY
| DROP PACKAGE BODY

Семантика

1. Триггеры BEFORE запускаются до изменений в системных таблицах. Триггеры AFTER запускаются после изменений в системных таблицах.

Важное правило

Тип события [BEFORE|AFTER] DDL не может быть изменён.

2. Когда оператор DDL запускает триггер, который вызывает исключение (BEFORE или AFTER, умышленно или непреднамеренно), оператор не будет подтверждён. Т.е. исключения могут использоваться для

гарантированной отмены оператора DDL при несоблюдении определённых условий.

3. Действия DDL триггера выполняются только после успешного подтверждения транзакции, в которой выполнялась затронутая DDL команда. Не забывайте также о том, что в AFTER триггере, можно выполнить только те команды, которые возможно сделать после DDL команды без автоматической фиксации транзакций. Вы не можете, например, создать таблицу в триггере и затем использовать её.
4. Для операторов «CREATE OR ALTER» триггер срабатывает один раз для события CREATE или события ALTER - в зависимости от того, существовал ли ранее объект. Для операторов RECREATE триггер вызывается для события DROP, если объект существовал, и после этого для события CREATE.
5. Если объект не существует, то триггеры на события ALTER и DROP обычно не запускаются. Исключения для этого случая описаны в пункте 6.
6. Исключение из пункта 5 заключается в том, что DDL триггер срабатывает перед использованием команды ALTER/DROP USER даже в том случае, когда пользователь не существует. Это происходит потому, что эти команды выполняют DML в базе данных безопасности, для которой не выполняется проверка существования пользователя перед выполнением этих команд. Данное поведение, вероятно, будет отличаться для встроенных пользователей, поэтому не пишите код, зависящий от этого.
7. Если некоторое исключение возбуждено после того, как начала выполняться команда DDL, но до момента запуска AFTER триггера, то AFTER триггер не выполняется.
8. Для упакованных процедур и функций (т.е. входящих в состав пакетов) индивидуальные триггеры {CREATE | ALTER | DROP}{PROCEDURE | FUNCTION} не запускаются.

Права для триггеров DDL

Следующие пользователи могут создавать, изменять или удалять триггеры DDL и использовать ключи, связанные с триггерами, в утилитах Firebird (см. *Поддержка триггеров DDL в утилитах*):

- Владелец базы данных;
- SYSDBA;
- пользователь, вошедший в систему с ролью администратора

RDB\$ADMIN;

- Пользователь, имеющий права на изменение метаданных и выполнение оператора ALTER DATABASE (см. [Права пользователей на изменение метаданных](#)).

Поддержка триггеров DDL в утилитах

Триггеры DDL определяются на уровне базы данных, поэтому к ним могут применяться ключи **-nodbtriggers** (в утилитах командной строки gbak и isql) и **-T** (nbackup). Эти параметры могут использовать только пользователи с соответствующими правами на изменение метаданных (см. выше).

Пространство имен контекста DDL_TRIGGER

Вместе с триггерами DDL было создано новое пространство имен DDL_TRIGGER для использования с функцией RDB\$GET_CONTEXT. Его использование допустимо только при срабатывании триггера DDL. Его использование допустимо в хранимых процедурах и функциях, вызываемых триггерами DDL.

Контекст DDL_TRIGGER работает как стек. Перед срабатыванием триггером DDL триггера, значения, относящиеся к выполняемой команде, помещаются в стек. После завершения триггера значения вытесняются из стека. Поэтому в случае каскадных операторов DDL, когда пользовательская команда DDL запускает триггер DDL, и этот триггер выполняет другую команду DDL с помощью оператора EXECUTE STATEMENT, значения переменных в пространстве имен DDL_TRIGGER будут соответствовать команде, которая активировала последний триггер DDL в стеке вызовов.

Переменные контекста DDL_TRIGGER

- EVENT_TYPE: тип события (CREATE, ALTER, DROP);
- OBJECT_TYPE: тип объекта (TABLE, VIEW, др.);
- DDL_EVENT: имя события (<ddl event item>), где <ddl_event_item> = EVENT_TYPE || ' ' || OBJECT_TYPE;
- OBJECT_NAME: имя объекта метаданных;
- OLD_OBJECT_NAME: для отслеживания переименование домена (см. примечание ниже);
- NEW_OBJECT_NAME: для отслеживания переименование домена (см. примечание ниже);

- SQL_TEXT: текст инструкции SQL.

Примечание

ALTER DOMAIN <старое имя> TO <новое имя> устанавливает значения для OLD_OBJECT_NAME и NEW_OBJECT_NAME в обоих триггерах (BEFORE и AFTER). Для этой команды OBJECT_NAME будет иметь старое имя объекта в BEFORE триггере и новое имя объекта в триггере AFTER.

Примеры использования DDL триггеров

Использование триггера DDL для применения согласованной схемы именования, в которой имена хранимых процедур должны начинаться с префикса "SP_":

```
set auto on;
create exception e_invalid_sp_name 'Неверное имя хранимой процедуры
(должно начинаться с префикса SP_)';
```

```
set term !;
create trigger trig_ddl_sp before CREATE PROCEDURE
as
begin
    if (rdb$get_context('DDL_TRIGGER', 'OBJECT_NAME')
        not starting 'SP_') then
        exception e_invalid_sp_name;
end!
```

```
-- Проверка работы триггера trig_ddl_sp
```

```
create procedure sp_test
as
begin
end!
```

```
create procedure test
as
begin
end!
```

```
-- The last command raises this exception and procedure TEST is not
created
-- Statement failed, SQLSTATE = 42000-- exception 1
-- -E_INVALID_SP_NAME
-- -Неверное имя хранимой процедуры (должно начинаться с
префикса SP_)
-- -At trigger 'TRIG_DDL_SP' line: 4, col: 5
```

set term ;!

Реализация пользовательской безопасности для DDL, в этом случае выполнение команд DDL разрешено только для определенных пользователей:

```
create exception e_access_denied 'Access denied';
```

```
set term !;
create trigger trig_ddl before any ddl statement
as
begin
    if (current_user <> 'SUPER_USER') then
        exception e_access_denied;
end!
```

```
-- Проверка работы триггера trig_ddl
```

```
create procedure sp_test
as
begin
end!
```

```
-- The last command raises this exception and procedure SP_TEST is not
created
-- Statement failed, SQLSTATE = 42000
-- exception 1
-- -E_ACCESS_DENIED
-- -Access denied
-- -At trigger 'TRIG_DDL' line: 4, col: 5
```

set term ;!

Триггер для регистрации действий и попыток DDL:

```
create sequence ddl_seq;

create table ddl_log (
    id bigint not null primary key,
    moment timestamp not null,
    user_name varchar(31) not null,
    event_type varchar(25) not null,
    object_type varchar(25) not null,
    ddl_event varchar(25) not null,
    object_name varchar(31) not null,
    sql_text blob sub_type text not null,
    ok char(1) not null
);

set term !;

create trigger trig_ddl_log_before before any ddl statement
as
    declare id type of column ddl_log.id;
begin
-- Мы должны производить изменения
-- в AUTONOMOUS TRANSACTION,
-- таким образом, если произойдёт исключение и команда
-- не будет запущена, она всё равно будет зарегистрирована
begin
    insert into ddl_log (id, moment,
        user_name,
        event_type,
        object_type,
        ddl_event,
        object_name,
        sql_text, ok)
    values (next value for ddl_seq, current_timestamp,
        current_user,
        rdb$get_context('DDL_TRIGGER', 'EVENT_TYPE'),
        rdb$get_context('DDL_TRIGGER', 'OBJECT_TYPE'),
        rdb$get_context('DDL_TRIGGER', 'DDL_EVENT'),
        rdb$get_context('DDL_TRIGGER', 'OBJECT_NAME'),
```

```
        rdb$get_context('DDL_TRIGGER', 'SQL_TEXT'), 'N')
    returning id into id;
    rdb$set_context('USER_SESSION', 'trig_ddl_log_id', id);
end
end!
```

-- Примечание: созданный выше триггер будет запущен для этой DDL.
-- Хорошей идеей является использование `--nodbtriggers`
-- при работе с ним

```
create trigger trig_ddl_log_after after any ddl statement
as
begin
-- Здесь нам требуется автономная транзакция, потому что
-- в оригинальной транзакции мы не увидим запись, вставленную
-- в BEFORE триггере в автономной транзакции, если
-- пользовательская транзакции не запущена
-- с режимом изоляции READ COMMITTED
    in autonomous transaction do
        update ddl_log set ok = 'Y'
        where id =
            rdb$get_context('USER_SESSION', 'trig_ddl_log_id');
end!
```

```
commit!
```

```
set term ;!
```

-- Удаляем запись о создании trig_ddl_log_after

```
commit;
```

-- Проверка
-- Эта команда будет зарегистрирована единожды
-- (т.к. не существует, то RECREATE вызовет событие CREATE)
-- с OK = Y

```
recreate table t1 (
    n1 integer,
    n2 integer
);
```

-- Оператор не выполнится, т.к. T1 уже существует
 -- таким образом ОК будет иметь значение N

```
create table t1 (
    n1 integer,
    n2 integer
);
```

-- T2 не существует. Записи об удалении не будет.
 drop table t2;

-- Это будет регистрироваться дважды
 -- (т.к. T1 существует, RECREATE
 -- воспринимается DROP и CREATE) with OK = Y.

```
recreate table t1 (
    n integer
);
```

commit;

```
select id, ddl_event, object_name, sql_text, ok
       from ddl_log order by id;
```

ID	DDL_EVENT	OBJECT_NAME	SQL_TEXT	OK
==	=====	=====	=====	===
2	CREATE TABLE	T1	80:3	Y

=====

SQL_TEXT:

```
recreate table t1 (
n1 integer,
n2 integer
)
```

3	CREATE TABLE	T1	80:2	N
---	--------------	----	------	---

=====

SQL_TEXT:

```
create table t1 (
n1 integer,
n2 integer
)
```

=====

```

4      DROP TABLE      T1                                80:6          Y
=====
SQL_TEXT:
recreate table t1 (
n integer
)
=====
5      CREATE TABLE    T1                                80:9          Y
=====
SQL_TEXT:
recreate table t1 (
n integer
)
=====

```

Поддержка курсора с возможностью прокрутки (двунаправленного)

Дмитрий Еманов

Вместо последовательного извлечения строк в прямом направлении (например, с использованием оператора *FOR SELECT...*), “прокручиваемость” допускает гибкую навигацию по набору данных с помощью открытого курсора как назад, так и вперед. Относительно текущей позиции курсора можно осуществлять выборку данных как из предыдущих, так и последующих строк. Прокручиваемый курсор может работать непосредственно в PSQL. Поддержка API доступна также для выборки строк с использованием курсора в DSQL.

Синтаксис курсора для PSQL

Объявление курсора:

```
DECLARE <name> SCROLL CURSOR FOR (<select expression>)
```

Для продвижения вперед:

```
FETCH <cursor name> [INTO <var name> [, <var name> ...]];
```


Для продвижения в любом направлении:

```
FETCH {NEXT | PRIOR | FIRST | LAST | ABSOLUTE <n> |  
      RELATIVE <n>}  
FROM <cursor name> [INTO <var name> [, <var name> ...]];
```

Смотрите также главу 4 *Изменения в Firebird API и ODS*. Раздел *Поддержка курсора с возможностью прокрутки (двунаправленного)* объясняет немного больше о его использовании для различных вариантов выборки.

Примечания

1. Когда опция прокрутки опущена, подразумевается NO SCROLL (т.е. курсор открывается только для чтения вперёд). Это означает, что могут использоваться только команды FETCH [NEXT FROM]. Другие команды вернут ошибку.
2. Прокручиваемые курсоры внутренне материализуются как временный набор записей, таким образом потребляя ресурсы памяти/диска, поэтому эту функцию следует использовать только тогда, когда это действительно необходимо.

Исключения с параметрами

Adriano dos Santos Fernandes

Теперь исключение может быть определено с сообщением, содержащим слоты для параметров, которые заполняются и передаются при вызове исключения, используя следующий синтаксис

```
EXCEPTION <name> USING ( <value list> )
```

Примеры

```
create exception e_invalid_val 'Invalid value @1 for the field @2';  
...  
    if (val < 1000) then  
        thing = val;  
    else  
        exception e_invalid_val using (val, 'thing');
```

end

```
CREATE EXCEPTION EX_BAD_SP_NAME
    'Name of procedures must start with "@1" : "@2";
```

```
CREATE TRIGGER TRG_SP_CREATE
    BEFORE CREATE PROCEDURE
AS
    DECLARE SP_NAME VARCHAR(255);
BEGIN
    SP_NAME =
    RDB$GET_CONTEXT('DDL_TRIGGER', 'OBJECT_NAME');
    IF (SP_NAME NOT STARTING 'SP_') THEN
        EXCEPTION EX_BAD_SP_NAME
            USING ('SP_', SP_NAME);
END;
```

Примечания

Вектор состояния создается с помощью следующей комбинации кода: `isc_except`, `<номер исключения>`, `isc_formatted_exception`, `<форматированное сообщение об исключении>`, `<параметры исключения>`.

Поскольку используется новый код ошибки (`isc_formatted_exception`), клиент должен быть версии 3.0 или, по крайней мере, использовать файл `firebird.msg` из версии 3.0, чтобы перевести вектор состояния в строку.

Учитывая каждый параметр (слева направо), передаваемый в случаях вызова исключения, начиная с 1 параметра по N-й:

- Если N-й параметр не передается, то текст не подставляется;
- Если передается значение NULL, то оно заменяется на строку '***
нуль ***';
- Если передается больше параметров, чем определено для исключения при его создании, то лишние параметры игнорируются;
- Общая длина сообщения, включая значения параметров, по-прежнему ограничена 1053 байтами.

Использование CONTINUE в цикле

Adriano dos Santos Fernandes

CONTINUE - это команда, дополняющая операторы BREAK/LEAVE, и позволяющая управлять потоком данных - остановить и начать следующую итерацию цикла FOR/WHILE.

Синтаксис

CONTINUE [<label>];

Пример

```
FOR SELECT A, D
  FROM ATABLE
  INTO :achar, :ddate DO
  BEGIN
    IF (:ddate < current_data - 30) THEN
      CONTINUE;
    ELSE
      /* что-то выполняется */
  ...
END
```

Стабилизация курсора в PSQL

Владислав Хорсун

Теперь курсоры без SUSPEND стабильны внутри PSQL:

```
FOR SELECT ID
  FROM T
  WHERE VAL IS NULL
  INTO :ID DO
  BEGIN
    UPDATE T
      SET VAL = 1
    WHERE ID = :ID;
  END
```

Ранее этот блок работал в непрерывном цикле. Теперь цикл не будет дополнительно выбирать значение, если оно было установлено в цикле.

Примечание

Это может изменить поведение устаревшего кода.

Если внутри блока есть приостановка, старая нестабильность остается. Например, этот запрос по-прежнему создает бесконечный цикл:

```
FOR SELECT ID
  FROM T
INTO :ID DO
  BEGIN
    INSERT INTO T (ID) VALUES (:ID);
    SUSPEND;
  END
```

Расширение использования префикса двоеточия

Adriano dos Santos Fernandes

До сих пор префикс двоеточия (:) использовался в PSQL для обозначения ссылки на переменную в операторах DML. Его использование было расширено в Firebird 3 для двух несвязанных целей:

1. Позволить OLD/NEW полям в курсорах быть прочитанными и присваивать их значение переменным;
2. Сделать присвоение переменных в операторах DML и PSQL в модулях и блоках более гибким и, при необходимости, устранить неоднозначность между именами полей и именами переменных.

Переменная типа курсор в PSQL

В PSQL теперь поддерживаются ссылки на курсор, как на переменные. Текущая запись доступна в PSQL теперь через имя курсора - явного (DECLARE AS CURSOR) или неявного (FOR SELECT), тем самым делая предложение INTO необязательным.

В циклах FOR SELECT для этого требуется указание предложения AS

CURSOR. Например:

```
execute block as
begin
    for select id, x
        from t1 as cursor c1 do
        begin
            for select id, x
                from t2 where x = :c1.x as cursor c2 do
            begin
                /* . . . */
            end
        end
    end
end
```

Примечание

Обратите внимание на расширение использования двоеточия (:) в качестве префикса к указателю на курсор.

Другие примеры

```
for select
    rdb$relation_id as id, rdb$relation_name as name
from rdb$relations
where rdb$view_blr is null
as cursor tables do
begin
    out_id = tables.id;
    out_name = tables.name;
    suspend;
end
```

Для избежания неоднозначности можно использовать префикс двоеточия:

```
    out_id = :tables.id;
    out_name = :tables.name;
-- или
    :out_id = :tables.id;
    :out_name = :tables.name;
```

«tables» здесь - это имя курсора и действует аналогично OLD/NEW в триггерах.

Новое в использовании двоеточия для переменных

Теперь допустимо применять префикс двоеточия к переменным по обе стороны инструкции присваивания, но только в конструкциях PSQL.

Ранее это были действительны такие присвоения значений:

```
var1 = :var2;  
/* или */  
new.fld = :var;
```

в то время как были недопустимы такие выражения:

```
:var1 = :var2;  
-- или  
:new.fld = :var;
```

См. [CORE-4434](#).

Теперь это допустимый синтаксис:

```
create trigger t1 before insert on t1  
as  
    declare v integer;  
begin  
    :v = :old.n;  
    :new.n = :v;  
end
```

Фактически в этом примере нет разницы между

```
:v = :old.n;  
-- и  
v = :old.n;
```

Вы думаете, что это просто “различие в синтаксе”, но в других случаях такое присваивание значений предоставляет средства для устранения неоднозначности между ссылками на имена полей и именами переменных.

Пример возможной неоднозначности

```
for select
    rdb$relation_id as id, rdb$relation_name as name
from rdb$relations
where rdb$view_blr is null as cursor tables do
begin
    out_id = :tables.id;
    select tables.name
        from tables
        where tables.id = :tables.id
    into :out_name;
    suspend;
end
```

Здесь внутри вложенного оператора **SELECT** «tables» - это одновременно имя таблицы и имя курсора и поэтому двоеточие используется для устранения неоднозначности.

Использование SQLSTATE в обработчиках исключений

Дмитрий Еманов

Код **SQLSTATE** стал доступен для перехвата исключений с помощью оператора **WHEN**. Код **SQLSTATE** используется также, как ранее использовались коды **SQLCODE** и **GDSCODE**:

```
...
WHEN SQLSTATE '22006' DO
    BEGIN
        -- ЧТО-ТО ВЫПОЛНЯЕТСЯ
    END
...
```

Примечание

Коды **SQLSTATE** являются строковыми литералами.

Ошибки времени выполнения EXECUTE BLOCK

Начиная с версии 3.0.1 отображаются номера строк и столбцов (местоположение) для ошибок времени выполнения, вызванных при выполнении EXECUTE BLOCK.

Увеличен размера тела PSQL

При использовании нового API размер тела хранимой процедуры или триггера может превышать традиционное ограничение в 32 кБ. В настоящее время в качестве меры безопасности жёстко задан предел 10 МБ, хотя теоретический предел составляет 4 ГБ. Традиционный размер всех входных или выходных параметров больше не ограничивается традиционным размером (64 КБ минус накладные расходы).

Более детально см. в разделе «*С помощью нового API сняты некоторые ограничения размера SQL*».

Глава 11

Мониторинг и утилиты командной строки

В Firebird 3.0 не выпущены новые функции мониторинга и другие утилиты. Просто были улучшены уже существующие функции.

Мониторинг

Дмитрий Еманов

В набор виртуальных таблиц, хранящих данные мониторинга, внесен ряд изменений. Они перечислены в разделе *Таблицы мониторинга* главы 4 *Изменения в Firebird API и ODS*.

Распознавание удаленных клиентов

Начиная с этой версии в столбец `MON$ATTACHMENTS.MON$REMOTE_ADDRESS` для протоколов TCPv4 и TCPv6 записывается также номер порта удаленного адреса клиента. Новый столбец в этой таблице также сообщает имя узла удаленного клиента. Более подробная информация об этом приведена в разделе *Изменения в выводе адреса удалённого клиента*.

Трассировка

Последние улучшения функций трассировки включают в себя:

- Вывод трассировки теперь поддерживает отображение объясненного плана. См. [*CORE-4451*](#) - Владислав Хорсун;
- Трассировка выполнения хранимых функций. См. [*CORE-4345*](#) - Владислав Хорсун;
- Трассировка больше не игнорирует тот факт, что пользователи из разных баз данных безопасности фактически являются разными пользователями. Теперь она "знает", в какой базе данных безопасности была выполнена проверка подлинности определенного SYSDBA или где были установлены права привилегированного пользователя. Теперь

невозможно, например, для пользователя с именем SYSDBA получить доступ к security3.fdb, если он в настоящее время не прошел проверку подлинности для этого доступа. См. [CORE-4851](#) - Александр Пешков.

gbak

Новая опция бэкапа “Skip Data”

Александр Пешков

В утилиту командной строки **gbak** добавлен новый ключ **-skip_d(ata)** для обеспечения возможности игнорировать данные из определенных таблиц во время резервного копирования.

Ключ **-skip_d(ata)** принимает в качестве аргумента регулярное выражение. Например, для пропуска данных двух таблиц (**sales** и **customer**) в базе данных с псевдонимом **'employee'** при выполнении резервного копирования выполните следующую команду:

```
gbak -skip_d '(sales|customer)' employee e1.fbk
```

Совет

Регулярное выражение в утилите **gbak** выполняется по тем же правилам, что и при поиске в SQL с помощью оператора **SIMILAR TO**. Руководство доступно в главе DML в замечаниях к выпуску Firebird 2.5 и на сайте по [следующему адресу](#).

Длинные имена для лог файлов

Александр Пешков

Это улучшение позволяет журналу **gbak** принимать длинное имя для файла, не вызывая сообщения об ошибке “*Attempt to store 256 bytes in a clumplet*”.

Статистика в реальном времени с подробным выводом

Владислав Хорсун

Теперь утилита **gbak** может показывать некоторую статистику времени

выполнения в своем подробном выводе. Добавлен новый ключ командной строки **-STATISTICS** для указания, какие статистические данные должны быть выведены. В настоящее время реализованы четыре элемента:

Таблица 11.1. Аргументы для вывода статистики gbak

Элемент	Аргумент	Выводимые данные
Общее время	T	Время, прошедшее с момента запуска процесса gbak , в секундах и миллисекундах
Временная дельта	D	Время, прошедшее с момента вывода предыдущей строки, в секундах и миллисекундах
Прочитано страниц	R	Число прочитанных страниц с момента вывода предыдущей строки, integer
Записано страниц	W	Число записанных страниц с момента вывода предыдущей строки, integer

В ключе **-STATISTICS** должен быть указан хотя бы один из аргументов. Аргументы не учитывают регистр и могут указываться в любом порядке. Это означает эквивалентность “TDRW” и “WdrT”.

Ключ **-STATISTICS** не будет иметь никакого эффекта, если не указан параметр **-v[er]**.

Подробный вывод со статистикой включает две специальные строки:

- строка с заголовками для указанной статистики, выводимая перед другими строками статистики:

gbak: time delta reads writes

- строка с итоговыми статистическими сводками по указанным позициям, выводимая после окончания основного процесса:

gbak: 46.684 0.002 171 82442 total statistics

Примечание

Эта функция полностью поддерживается в API служб с новым элементом (см. [Поддержка SPB новой статистики, выводимой gbak](#)) в SPB (Service Parameter Block). Утилита **fbsvcmgr** также поддерживает реализацию SPB.

Примеры

1. Показать общее время с момента запуска gbak:

```
gbak -v -STATISTICS T -b employee emp_bkp.fbk -y log21.log
```

2. Показать время дельты и количество прочитанных страниц:

```
... -STAT DR ...
```

3. Показать все элементы статистики:

```
... -sta TDRW ...
```

4. Пример подробного вывода gbak:

```
firebird>gbak -v -stat tdrw -r a.fbk a.fdb
gbak:opened file a.fbk
gbak: time delta reads writes
gbak: 0.173 0.173 0 0 transportable backup -- data in XDR format
gbak: 0.175 0.002 0 0 backup file is compressed
gbak: 0.177 0.001 0 0 backup version is 10
gbak: 0.270 0.092 0 650 created database a.fdb, page_size 8192
```

bytes

```
gbak: 0.273 0.002 0 2 started transaction
...
gbak: 18.661 0.002 0 0 restoring data for table TEST1
gbak: 18.698 0.036 0 0 10000 records restored
...
gbak: 25.177 0.036 0 0 1770000 records restored
gbak: 25.220 0.042 0 1633 1780000 records restored
...
gbak: 38.702 0.002 0 0 restoring privilege for user SYSDBA
gbak: 38.707 0.004 22 0 creating indexes
gbak: 45.015 6.308 82 38394 activating and creating deferred
```

index T2_VAL

```
...
gbak: 46.682 0.008 4 13 finishing, closing, and going home
gbak: 46.684 0.002 171 82442 total statistics
gbak:adjusting the ONLINE and FORCED WRITES flags
```

См. Также [*CORE-1999*](#).

Новый код возврата

Александр Пешков

Начиная с версии 3.0.1 **gbak** возвращает ненулевой код результата при сбое восстановления во время создания и активации отложенного индекса пользователя.

gsec

Начиная с версии Firebird 3 утилита **gsec** считается устаревшей. Это означает, что вам предлагается использовать новые функции SQL для управления доступом (см. [Управление пользователями с помощью SQL](#)), описанные в главе 7 [Безопасность](#), вместо существующих в **gsec** эквивалентов.

Важно

Утилита **gsec** может работать с **security3.fdb**, но не будет работать с альтернативными базами данных безопасности.

isql

Расширение SET EXPLAIN для детального просмотра планов

Дмитрий Еманов

Добавлена новая опция SET: SET EXPLAIN [ON | OFF]. Она расширяет опцию SET PLAN, чтобы сообщить об объясненном плане вместо стандартного (см. [Вывод расширенного плана в isql](#)).

Если SET PLAN не используется (пропущен), то SET EXPLAIN включает вывод плана. SET PLANONLY работает также, как и в предыдущих версиях.

Параметры использования

SET PLAN = простой план + выполнение запроса

SET PLANONLY = простой план, без выполнения запроса

SET PLAN + SET EXPLAIN = объяснённый план + выполнение запроса

SET PLAN + SET EXPLAIN + SET PLANONLY = объяснённый план, без выполнения запроса

SET EXPLAIN = объяснённый план + выполнение запроса

SET EXPLAIN + SET PLANONLY = объяснённый план, без выполнения запроса

Извлечение метаданных

Claudio Valderrama C.

Работа утилиты `isql` с ключом для извлечения метаданных (`-[e]x[tract]`) была улучшена для создания скрипта, который учитывает порядок зависимостей объектов.

Также добавлена метка, отражающая *детерминированный* (*deterministic*) флаг для хранимых функций - Adriano dos Santos Fernandes.

Путь для INPUT файлов

Adriano dos Santos Fernandes

Команда `INPUT` будет теперь использовать относительный путь на основе открытого в последний раз каталога и последнего открытого файла в цепочке для определения местоположения следующего файла.

Увеличение размера буфера команд

Adriano dos Santos Fernandes

Размер буфера команды `isql` увеличен с 64 КБ до 10 МБ в соответствии с новыми предельными значениями для ядра сервера ([*CORE-4148*](#)).

Детерминированная метка для сохраненных функций

Adriano dos Santos Fernandes

В команду **SHOW FUNCTION** была добавлена метка, отображающая *детерминированный* флаг для сохраненных функций ([CORE-4940](#)).

fb_lock_print

Входные параметры

Дмитрий Еманов

Теперь утилита **fb_lock_print** может принимать 32-разрядные целые числа в качестве входных параметров для секунд и интервалов. Раньше они были ограничены размерностью типа данных **SMALLINT**.

Улучшения для использования

Владислав Хорсун

Несколько других небольших улучшений:

1. Из командной строки (**-help**) доступна более подробная справка по использованию;
2. Журнал событий и список владельцев больше не выводятся по умолчанию: если требуется, то они могут быть запрошены явно. По умолчанию теперь используется только заголовок;
3. Новый ключ **-o[wners]** позволяет выводить только владельцев с ожидающими запросами (блокировки).

gfix

Ключ -NoLinger

Александр Пешков

В утилиту **gfix** добавлен новый ключ **-NoLinger** для обеспечения одноразового переопределения параметра **LINGER** базы данных.

Больше информации об этом Вы найдёте в разделе [«Задержка» закрытия базы данных для Superserver](#).

Улучшения для сообщений при проверке

Владислав Хорсун

- Критические сообщения проверки теперь разделены на второстепенные;
- Теперь имя таблицы возвращается в тексте сообщений об ошибках проверки правильности, чтобы помочь определить контекст ошибки.

Другие тонкие настройки

Некоторые неудобных реализаций были исправлены в нескольких утилитах.

Все утилиты командной строки

Разрешение проблемы с путями к базе данных

Александр Пешков

Все утилиты считывают пути к базе данных из файла `databases.conf`, когда они хотят получить доступ к файлу базы данных напрямую. Но не все они выполняют одни и те же правилам для получения пути к базе данных. Теперь все они используют одни и те же правила для этого.

Справка и информация о версии

Claudio Valderrama C.

Все утилиты командной строки, кроме `gpre` и `qli`, теперь представляют справочную информацию и сведения о версии унифицированным и согласованным образом ([*CORE-2540*](#)).

Замена жёстко закодированных сообщений

Claudio Valderrama C.

В утилитах `tracemanager` и `nbackup` жёстко закодированные сообщения были заменены регулярными параметризованными типами.

Очистка синтаксиса произвольного ключа

Claudio Valderrama C.

В утилитах `qli` и `nbackup` были сделаны опции ключей для проверки правильности (или неправильности) при вводе сокращенных ключей.

Глава 12

Проблемы совместимости

В этом разделе представлены функции и модификации, которые могут повлиять на способ установки и использования Firebird в предыдущих версиях.

Где искать инструменты?

В Windows Вы найдете все исполняемые программы, включая утилиты командной строки, в папке установки Firebird (root). В предыдущих версиях они находились в подкаталоге `..bin` в папке установки Firebird.

На платформах POSIX Вы найдете инструменты и другие исполняемые программы в местах, аналогичных используемым для более ранних версий. Где именно - зависит от используемого Вами дистрибутива (в зависимости от платформы POSIX).

Другие изменения для Windows

Обратите внимание, что установщик больше не предоставляет возможности для переименования (или копирования) клиентской библиотеки `fbclient.dll` в `gds32.dll` или для её поиска в системной папке. Вам придется делать это вручную, если Ваши приложения всё ещё требуют этого.

Файл `aliases.conf` больше не нужен

В корневом каталоге Firebird файл `aliases.conf` заменён на `database.conf`. Формат псевдонимов базы данных не изменился, поэтому вы можете скопировать их и вставить содержимое вашего существующего файла `aliases.conf` в файл `databases.conf`. Новый файл `databases.conf` может содержать гораздо больше деталей конфигурации, чтобы обеспечить возможность настройки на уровне базы данных многих функций, которые ранее были доступны только на глобальном уровне сервера.

Встроенные соединения

Для встроенного подключения на платформах POSIX больше не требуется аутентификация. При вводе имени пользователя и пароля пароль игнорируется. Приложения по-прежнему могут требовать имя пользователя и, возможно, имя роли для использования привилегий SQL в базах данных.

Эта функция является новой для встраиваемого Firebird на POSIX, но этот способ встраиваемых всегда работал на Windows, когда встроенное ядро сервера было отдельным исполняемым файлом.

Инициализация базы данных безопасности

По умолчанию Firebird 3 использует новую модель аутентификации, которая использует плагин **SRP** для работы с пользовательскими паролями и для генерации уникальных идентификаторов сеанса для шифрования трафика. База данных безопасности (**security3.fdb**) не имеет предопределенных пользователей. Это сделано преднамеренно.

Тем не менее, при использовании стандартных установщиков для Windows, Linux и MacOSX, пользователь **SYSDBA** создается на последнем шаге установки с паролем, который является либо случайным, либо определяется с помощью ввода его пользователем.

В ситуации, когда этап инициализации **SYSDBA** завершается неудачей или отсутствует в процессе установки конкретной ОС, такой как установка из архива **zip** для Windows или выделенного порта платформы POSIX, может потребоваться ручная инициализация базы данных безопасности для использования с плагином **SRP**. В этом случае Вам необходимо создать пользователя **SYSDBA** и настроить пароль для него с помощью SQL команды создания пользователя во встроенном режиме в качестве первого шага для получения доступа к базам данных и утилитам.

Можно использовать и утилиту **gsec**, но, будучи устаревшей, она не обсуждается здесь.

Важно

Эта инициализация не требуется и не должна выполняться, если Вы настроили сервер для использования устаревшей (до Firebird 3) аутентификации и управления пользователями. Устаревшие плагины безопасности полностью сохраняют прежнее поведение и, таким

образом, содержат устаревшую запись для SYSDBA с «masterke» в качестве начального пароля (в POSIX начальный пароль может быть другим).

Инструкции по настройке firebird.conf для устаревшего поведения аутентификации приведены в разделе «Устаревшая аутентификация».

Шаги инициализации

Инициализация выполняется во встроенном режиме с помощью утилиты isql. Для встроенного соединения пароль не требуется и будет проигнорирован, если Вы предоставите его. Если Вы опускаете имя пользователя, то встроенное подключение будет работать нормально и без учетных данных входа (“входа в систему”), используя учётные данные хоста. Однако, хотя имя пользователя не подлежит аутентификации, создание или изменение чего-либо в существующей базе данных безопасности требует, чтобы пользователь был SYSDBA; в противном случае isql выдаст ошибку привилегии для запроса CREATE USER.

Команды управления пользователями SQL могут работать с любой открытой базой данных. Поскольку пример базы данных employee.fdb присутствует в Вашей установке и уже имеет алиас в файле databases.conf, то удобно использовать её для задачи управления пользователями.

1. **Остановите сервер Firebird.** Firebird 3 активно кэширует подключения к базе данных безопасности. Наличие подключений к серверу может помешать isql установить встроенное подключение.

2. Запустите интерактивный сеанс isql и откройте базу данных сотрудников с использованием псевдонима:

```
> isql -user sysdba employee
```

3. Создайте пользователя SYSDBA:

```
SQL> create user SYSDBA password 'SomethingCryptic';  
SQL> commit;  
SQL> quit;
```

Примечание

Пользователь SYSDBA автоматически будет иметь полные права

администратора. Не назначайте роль ADMIN (RDB\$ADMIN) для SYSDBA.

Быстрый совет

Начиная с версии Firebird 3.0.1 оператор CREATE ИЛИ ALTER USER SYSDBA PASSWORD <пароль> может использоваться для инициализации пустой базы данных безопасности securityN.fdb.

4. Для завершения инициализации запустите сервер Firebird. Теперь Вы сможете выполнять сетевое подключение к базам данных, включая базы данных безопасности, используя пароль пользователя sysdba.

О паролях

- Эффективный пароль, используемый диспетчером пользователей по умолчанию SRP может иметь длину до 20 символов, хотя будет действительным и пароль до 255 символов;
- Если Вы планируете в будущем настроить сервер на использование устаревшей проверки подлинности (настоятельно не рекомендуется!), то будут прочитаны только первые 8 символов любого пароля, включая пароль SYSDBA;
- Настоятельно не рекомендуется использовать пароль masterkey для sysdba!

Дополнительные сведения о базах данных безопасности

Начиная с версии Firebird 2, пользователи - в том числе и sysdba - не могут подключаться непосредственно к базе данных безопасности. Начиная с Firebird 3 можно установить встроенное подключение непосредственно к базе данных безопасности. Рекомендуется, чтобы администраторы баз данных отказались от использования gsec и ознакомились с новыми функциями управления пользователями SQL (см. [Управление пользователями с помощью SQL](#)).

Устаревшая аутентификация

Если Вы не собираетесь сразу использовать зашифрованные SRP логины и хотите использовать базу данных безопасности security3.fdb, как Вы это делали в предыдущих версиях Firebird, выполните следующие

действия:

1. Откройте файл `firebird.conf` для редактирования и найдите запись для параметра **UserManager**:

```
#UserManager = Srp
```

Удалите символ “#” и измените значение параметра на следующее

```
UserManager = Legacy_UserManager
```

2. Найдите параметр **WireCrypt**:

```
#WireCrypt = Enabled (for client) / Required (for server)
```

Удалите символ “#” и измените значение параметра на следующее

```
WireCrypt = Enabled
```

-- или, если Вы не планируете использовать шифрование

SRP

```
WireCrypt = Disabled
```

3. Найдите параметр **AuthServer**:

```
#AuthServer = Srp, WinSspi, Legacy_Auth
```

Удалите символ “#” и измените порядок следования аргументов

```
AuthServer = Legacy_Auth, Srp, WinSspi
```

4. Найдите параметр **AuthClient**:

```
#AuthClient = Srp, WinSspi, Legacy_Auth
```

Удалите символ “#” и измените порядок следования аргументов

```
AuthClient = Legacy_Auth, Srp, WinSspi
```

5. Сохраните изменения.

6. Остановите и рестартуйте сервер Firebird для вступления изменений в силу.

Устаревшие пароли

Старый пароль `masterke` теперь доступен для Вашего первого входа

пользователя **sysdba**. Так как этот пароль известен всем, то его необходимо изменить как можно скорее.

Напоминание: устаревшая процедура проверки подлинности пользователя читает только первые 8 символов пароля.

Обновление базы данных безопасности версии 2.x

По причине использования новой модели аутентификации в Firebird 3 непосредственное использование базы данных безопасности версии 2.5 (**security2.fdb**) в Firebird 3 невозможно. Тем не менее, существует процедура обновления, позволяющая сохранить данные учетной записи пользователя (его имя и др.), но не пароли, из базы данных **security2.fdb**, которая использовалась в версии 2.x. сервера.

Пользователь SYSDBA при этом не затрагивается: после процедуры обновления он останется прежним.

Шаги обновления базы данных безопасности

Примечание

- При необходимости замените **masterkey** актуальным паролем SYSDBA для текущей версии сервера, как это показано в приведённых ниже командах;
- Для этой процедуры необходимо запустить скрипт **security_database.sql**, который находится в подкаталоге **misc/upgrade** вашей установки Firebird 3.

1. Под сервером Firebird 2.5 создайте резервную копию базы данных безопасности, используя пароль **sysdba** от версии 2.5:

```
gbak -user sysdba -pas masterke -b {host/path}security2.fdb  
sec2.fbk
```

2. Под сервером Firebird 3 восстановите резервную копию от версии 2.5, используя пароль **sysdba** от версии 3:

```
gbak -user sysdba -pas masterkey -c sec2.fbk  
{host/path}security2db.fdb
```

3. Под сервером Firebird 3 с помощью утилиты **isql** выполните скрипт

обновления:

```
isql -user sysdba -pas masterkey -i security_database.sql  
{host/path}security2db.fdb
```

"security2db.fdb" - здесь это только пример для имени базы данных безопасности: она может иметь любое имя, например, "sec2db.fdb".

4. Эта процедура генерирует новые случайные пароли и выводит их затем на экран. Запишите эти данные и оповестите пользователей о новых паролях.

5. После этого остановите сервер Firebird. Переименуйте файл security3.fdb или переместите его в другое место, а затем переименуйте обновлённую базы данных в security3.fdb.

Локальное соединение с SuperServer под Windows

В предыдущих версиях Firebird для локальных клиентов, подключенных к Superserver на платформе Windows по протоколу XNET, был доступен серверный протокол, известный как «Локальный Windows». Типичная строка подключения выглядела так:

```
c:\Program Files\Firebird_2_5\examples\empbuild\employee.fdb
```

При такой строке подключения новый унифицированный сервер пытается загрузить встроенный сервер. Для режима SuperServer это теперь не работает - Вы получите сообщение об отказе «Файл используется другим процессом» («File is in use by another process»). Это не баг. Поскольку клиенты Superserver обмениваются ресурсами, другой сервер (в данном случае встроенный сервер) не может подключить клиента к той же базе данных, к которой подключен Superserver.

Но есть выход из этой ситуации. Протокол XNET может выполнять локальные клиентские сеансы для Superserver. Просто теперь Вам нужна более сложная строка подключения:

```
xnet://alias-or-path-to-database
```

т.е. используя путь к базе данных

```
xnet://c:\Program Files\Firebird_3_0\examples\empbuild\employee.fdb
```

или используя алиас

xnet://employee

Примечание

Новые строки подключения также доступны в качестве альтернативы другим локальным протоколам подключения - см. *Новые форматы подключения для локальных клиентов Superserver в Windows*.

Параметры конфигурации

Ранее устаревшие параметры **CompleteBooleanEvaluation**, **OldColumnNaming** и **OldSetClauseSemantics** больше не поддерживаются и были удалены из firebird.conf. Отсутствие одного или нескольких из этих параметров может привести к неправильной работе Ваших приложений, поэтому, пожалуйста, проверьте эти параметры в firebird.conf на старой версии сервера.

Параметры **UsePriorityScheduler**, **PrioritySwitchDelay**, **PriorityBoost**, **LegacyHash** и **LockGrantOrder** больше не используются и также удалены.

Важно

Убедитесь, что Вы изучили раздел *Изменения и дополнения в параметрах конфигурации* при подготовке к обновлению программного обеспечения пользователя Firebird 3.

Системные таблицы доступны теперь только для чтения

Системные таблицы (RDB\$*) теперь доступны только для чтения. Попытки выполнения любого оператора DDL или операций insert/update/delete для них будут отклонены.

Изменения языка SQL

Обратите внимание на некоторые изменения в реализации языка SQL.

Поддержка смешанных соединений больше не поддерживается

Некорректное смешивание явных и неявных соединений больше не поддерживаются в соответствии со спецификацией SQL. Это также означает, что в явном соединении `A JOIN B ON <условие>` не разрешено ссылаться на любой поток, кроме A и B. См. [CORE-2812](#).

Длина для локальных псевдонимов и имен переменных ограничена

Имена псевдонимов столбцов и таблиц, а также имена локальных переменных в PSQL теперь ограничены длиной в 31 байт. Это специально было сделано для соответствия спецификации SQL, требующей, чтобы они были обычными идентификаторами SQL с зависящими от реализации ограничениями на длину. В Firebird максимальная длина идентификаторов составляет 31 байт. См. также [CORE-2350](#).

Изменения для имён пользователей

Имена пользователей рассматриваются как идентификаторы SQL и, соответственно, теперь ограничены максимальной длиной в 31 байт.

Теперь также поддерживаются имена пользователей с учётом регистра. Это позволяет указывать имена пользователей в двойных кавычках в `isql` при выполнении операторов `CONNECT` и `CREATE DATABASE`.

Столбцы вывода курсора должны иметь имя

Оператор `DECLARE CURSOR` в PSQL теперь требует, чтобы все выходные столбцы были явно названы или имели псевдонимы (алиасы). Это же требование применяется и к оператору `FOR SELECT ... AS CURSOR <имя курсора> DO ...` в PSQL. Данное требование обусловлено новой возможностью считывания элементов курсора непосредственно в виде псевдо столбцов, например, `MY_CURSOR.COLUMN_A`.

Пояснение:

```
create procedure sp_test
as
  declare c cursor for (select 1 /* as a */ from rdb$database);
  declare n int;
begin
  open c;
```

```
fetch c into n;  
close c;  
end
```

Statement failed, SQLSTATE = 42000
unsuccessful metadata update
-ALTER PROCEDURE SP_TEST failed
-Dynamic SQL Error
-SQL error code = -104
-Invalid command
-no column name specified for column number 1 in derived table

C

Эффекты «стабильного курсора»

Некоторые операторы теперь могут работать по-разному из-за улучшения «стабильности курсора» (см. [Стабилизация курсора в PSQL](#)). Были затронуты следующие заявления:

- те, которые изменяют таблицу, которая явно или неявно выбрана из одного и того же оператора;
- (как побочный эффект) некоторые операторы **MERGE** могут работать по-разному, если возможны многочисленные совпадения.

Примечание

Стандарт SQL предусматривает, что инструкция слияния **MERGE** должна вызывать ошибку при обнаружении нескольких совпадений. Firebird не так строг в этом отношении, но его поведение в этих случаях следует считать неопределенным.

См. также [CORE-3362](#), [CORE-3317](#), [CORE-4796](#).

Зарезервированные слова

Введен ряд новых зарезервированных ключевых слов. Пожалуйста, ознакомьтесь с ними в разделе «[Зарезервированные слова и изменения](#)» и убедитесь, что Ваши запросы в DSQL и тела процедур/триггеров не содержат ни одного из этих ключевых слов в качестве идентификаторов. В противном случае необходимо будет либо заключать их в двойные кавычки (только для

диалекта 3), либо переименовать их.

Особое внимание обратите на ключевые слова INSERTING, UPDATING и DELETING в Ваших модулях PSQL. Теперь они являются зарезервированными словами и поэтому не должны использоваться в качестве идентификаторов.

Также внимательно проверьте, что ключевое слово BOOLEAN не используется в качестве имени домена в базах данных.

Синтаксис конфигурации трассировки

Изменен формат секций и пар ключ - значение в файле конфигурации трассировки:

Старый формат:

```
<database %[\V](test|azk2|rulez).fdb>
    enabled true
    time_threshold 100
    log_statement_finish true
</database>
```

Новый формат:

```
database = %[\V](test|azk2|rulez).fdb
{
    enabled = true
    time_threshold = 100
    log_statement_finish = true
}
```

Важно

Обратите внимание на заголовок раздела базы данных и знаки равенства между именами параметров и значениями.

Неожиданные ошибки GDSCODE

Неожиданный GDSCODE может быть возвращен для операторов UPDATE/DELETE, а именно: isc_deadlock вместо isc_lock_conflict.

Фактически, полный стек ошибок ранее содержал {isc_lock_conflict, isc_deadlock, isc_update_conflict}, тогда как теперь он содержит более правильные значения {isc_deadlock, isc_update_conflict}.

Однако системная переменная GDSCODE возвращает первый элемент ошибки, что, возможно, вызывает проблемы совместимости. Убедитесь, что помимо isc_lock_conflict обработчики ошибок для обновления и удаления также проверяют другие коды ошибок isc_deadlock и isc_update_conflict.

Распознавание удалённого адреса клиента

Начиная с этой версии адрес удаленного клиента для протоколов TCPv4 и TCPv6 в MON\$ATTACHMENTS.MON\$REMOTE_ADDRESS включает в себя номер порта, отделенный от IP адреса косой чертой (слэшем):

<IP address>/<port>

Новый столбец в таблице MON\$ATTACHMENTS также сообщает имя узла удаленного клиента.

Дополнительную информацию см. в разделе «*Изменения в выводе адреса удалённого клиента*» главы 4 «*Изменения в Firebird API и ODS*».

Глава 13

Исправление ошибок

Выпуск Firebird 3.0.3: Исправление ошибок

Следующие ошибки были исправлены до выпуска официального Firebird 3.0.3:

Ядро сервера выпуска 3.0.3

([CORE-5735](#)) Дополнительный владелец ключа может открыть несанкционированные подключения к зашифрованной базе данных, создавая таким образом уязвимость для базы данных.

Исправил А. Пешков

([CORE-5720](#)) При большой нагрузке на сервер в классическом режиме сборка мусора может работать слишком медленно.

Исправил В. Хорсун

([CORE-5713](#)) Псевдоним поля может исчезнуть в сложном запросе.

Исправил А. dos Santos Fernandes

([CORE-5695](#)) Функция POSITION не учитывала сопоставление для BLOB.

Исправил В. Хорсун

([CORE-5694](#)) В столбцах с ограничением UNIQUE могут встречаться повторяющиеся значения.

Исправил В. Хорсун

([CORE-5684](#)) При удалении записи из MON\$ATTACHMENTS с использованием предложения ORDER BY возникает ошибка “нет текущей записи для операции выборки” (“no current record for fetch operation”).

Исправил Д. Еманов

([CORE-5681](#)) При выполнении внешней инструкции и откате локальной транзакции может произойти нарушение прав доступа.

Исправил В. Хорсун

([CORE-5675](#)) Функции isc_vax_integer() и isc_portable_integer() неправильно работали с короткими отрицательными числами.

Исправил Д. Сибиряков

([CORE-5673](#)) Ограничение UNIQUE не будет работать в зашифрованной базе данных при выполнении первой команды вставки не уникального значения.

Исправил А. Пешков

([CORE-5667](#)) Сообщение “СТЕ ‘X’ имеет циклические зависимости” (“СТЕ ‘X’ has cyclic dependencies”) появлялось в случае, когда ‘X’ был псевдонимом для набора результатов, и в запросе с прежним именем ‘X’ была предыдущая часть СТЕ.

Исправил В. Хорсун

([CORE-5659](#)) При определенных условиях оптимизатор создает плохой план.

Исправил Д. Еманов

([CORE-5645](#)) Неправильная транзакция может быть передана внешнему серверу.

Исправил A. dos Santos Fernandes

([CORE-5643](#)) Сообщение “Ошибка вызова munmap операционной системы. Код ошибки 12” (“Operating system call munmap failed. Error code 12”) может появиться в firebird.log при большой нагрузке в классическом или суперклассическом сервере.

Исправил А. Пешков

([CORE-5638](#)) Неправильный результат с индексом при нечувствительных к регистру параметрах сортировки NUMERIC-SORT.

Исправил A. dos Santos Fernandes

([CORE-5618](#)) Некоторые части страниц BLOB второго уровня не освобождались при удалении.

Исправил Д. Логашов

([CORE-5605](#)) При запуске ПЛАГИНА для доступа к базе данных версии 2.5, ядро неправильно проверило бы существующую карту, если использовалась устаревшая база данных безопасности (до FB3).

Исправил А. Пешков

([CORE-5600](#)) При добавлении нового столбца BLOB типа TEXT и обновлении другого поля в этой же операции будет сообщаться недопустимый идентификатор BLOB объекта.

Исправил В. Хорсун

([CORE-5598](#)) Ошибка "Размер блока превышает ограничение реализации" ("Block size exceeds implementation restriction") может возникнуть при внутреннем соединении больших наборов данных с длинным ключом с помощью HASH JOIN плана.

Исправил Д. Еманов

([CORE-5588](#)) Интерфейс DbInfo не передавался вспомогательным экземплярам подключаемого модуля.

Исправил А. Пешков

([CORE-5580](#)) Подписи упакованных функций не проверялись на несоответствие с атрибутом [NOT] DETERMINISTIC.

Исправил A. dos Santos Fernandes

([CORE-5576](#)) Запросы, содержащие предложение WITH LOCK, могут вызывать ошибки.

Исправил Р. Симаков

([CORE-5567](#)) Прямые модификации системных таблиц не были полностью запрещены.

Исправил A. dos Santos Fernandes

([CORE-5555](#)) Обработка ошибок для SELECT WITH LOCK нарушила совместимость с Firebird 2.5.

Исправил В. Хорсун

([CORE-5553](#)) База данных не может быть зашифрована, если для параметра DatabaseAccess установлено значение None.

Исправил А. Пешков

([CORE-5550](#)) Вычисляемое поле DECIMAL в представлении могут быть сохранены в таблицу RDB\$FIELD_PRECISION с неверным значением.

Исправил А. dos Santos Fernandes

([CORE-5549](#)) При создании или запуске Firebird на оборудовании, на котором не было SSE, могут возникать ошибки.

Исправил А. Пешков

([CORE-5542](#)) Триггеры уровня базы данных, связанные с событиями TRANSACTION (START/COMMIT/ROLLBACK), не учитывали их индекс POSITION.

Исправил А. dos Santos Fernandes

([CORE-5535](#)) Поле RDB\$FIELD_SUB_TYPE в таблице RDB\$FUNCTION_ARGUMENTS содержит неверное значение после изменения функции.

Исправил А. dos Santos Fernandes

([CORE-5528](#)) Внутренняя проверка согласованности Firebird (limbo impossible (184), файл: vio.линия.cpp: 2379)

Исправил В. Хорсун

([CORE-5527](#)) Внешние подпрограммы не получали значения по умолчанию для своих выходных параметров.

Исправил A. dos Santos Fernandes

([CORE-5526](#)) Внешние подпрограммы могут получать параметры с неправильной длиной или типом данных.

Исправил A. dos Santos Fernandes

([CORE-5517](#)) Классический сервер может периодически вызывать ошибку "Переполнение памяти глобального сопоставления" ("Global mapping memory overflow") и требовать перезапуска для принятия новых соединений.

Исправил А. Пешков

([CORE-5416](#)) Утечка памяти: Firebird не освобождал память после нагрузочных тестов.

Исправил В. Хорсун

([CORE-5415](#)) Обнаружена и исправлена трудно уловимая ошибка, из-за которой неожиданная перезапись памяти могла привести к повреждению памяти случайной страницы базы данных.

Исправил В. Хорсун, с помощью тестов и дампов памяти Д. Коваленко.

([CORE-4492](#)) Наличие операторов OR или IN в предикатах для RDB\$DBKEY оптимизатору не удавалось использовать план индекса, хотя он должен был использоваться.

Исправил Д. Еманов

([CORE-2853](#)) Ошибка: page N, page type 5 lock conversion denied (215).

Исправил В. Хорсун

([CORE-2731](#)) Рекурсивная инструкция EXECUTE STATEMENT не работала должным образом.

Исправил В. Хорсун

([CORE-2502](#)) Ошибки циклических зависимостей в CTE, включающих алиасы.

Исправил В. Хорсун

([CORE-2284](#)) После отката (rollback) инструкции CREATE TABLE в таблице RDB\$PAGES оставались записи.

Исправили Р. Симаков и Д. Еманов

Сбои и зависания сервера выпуска 3.0.3

([CORE-5730](#)) Подключение к серверу может зависнуть при работе с зашифрованными базами данных по сетевому протоколу, отличному от TCP.

Исправил А. Пешков

([CORE-5719](#)) Сервер Firebird 3 может аварийно завершить работу при восстановлении из резервной копии Firebird 2.5.

Исправил A. dos Santos Fernandes

([CORE-5707](#)) Начало и конец физического резервного копирования в одной и той же транзакции может привести к падению сервера.

Исправил В. Хорсун

([**CORE-5706**](#)) Конфигурация трассировки с неуместным символом “{” может привести к падению сервера.

Исправил A. dos Santos Fernandes

([**CORE-5591**](#)) После определенных ошибок сервер может зависнуть во время запуска транзакции,.

Исправил A. Пешков

([**CORE-5562**](#)) Firebird может аварийно завершить работу, если часто происходила загрузка и выгрузка UDF.

Исправил A. Пешков

([**CORE-5547**](#)) Сервер может аварийно завершить работу при компиляции хранимой процедуры с вложенными ссылками на вычисляемые на основе запроса поля.

Исправил Д. Еманов

([**CORE-5533**](#)) Сервер может аварийно завершить работу, если база данных содержит триггер уровня базы данных.

Исправил A. dos Santos Fernandes

([**CORE-5516**](#)) При остановке службы Firebird может произойти сбой.

Исправил В. Хорсун

Сборка сервера выпуска 3.0.3

([*CORE-5654*](#)) Промежуточные файлы сборки для примеров плагинов были перепутаны.

Исправил А. Пешков

API/Удалённый интерфейс сервера выпуска 3.0.3

([*CORE-5721*](#)) Информационные элементы `isc_info_length` и `fb_info_crypt_state` имели одинаковый код.

Важно

Программное обеспечение, использующее константу `fb_info_crypt_state`, должно быть перекомпилировано, чтобы оно работало с выпуском 3.0.3 и более поздними версиями

Исправил А. Пешков

([*CORE-5686*](#)) Firebird 3.0.2 позволяет использовать любой протокол версии ≥ 10 для подключения, в том числе и с помощью протокол 14 InterBase.

Исправил М. Rotteveel

([*CORE-5649*](#)) Команды `isc_add_user/isc_modify_user` могут создать непригодную для использования запись пользователя.

Исправил А. Пешков

([*CORE-5521*](#)) Состояние состязания могло возникнуть между регистрацией события и уведомлением о событии.

Исправил В. Хорсун

([CORE-5510](#)) Сервер мог отправить вектор состояния с более, чем с 20 элементами старым клиентам и вызвать разрыв соединения.

Исправил Р. Симаков

Для POSIX сервера выпуска 3.0.3

([CORE-5650](#)) Удаление процедуры на классическом сервере может привести к сбою сегментации.

Исправил А. Пешков

([CORE-5624](#)) Исправлена старая ошибка, при которой обратные косые черты в строках пути для использования в файловых системах POSIX не всегда преобразовывались в прямые косые черты.

Исправил А. Пешков

Утилиты сервера выпуска 3.0.3

gbak

([CORE-5653](#)) gbak очень медленно восстанавливал данные с большим количеством мелких BLOB в классическом сервере Linux.

Исправил Р. Симаков

([CORE-5579](#)) Во время восстановления в gbak может возникнуть ошибка синхронизации запросов.

Исправил В. Хорсун

([CORE-5566](#)) Сервер может аварийно завершить работу при восстановлении из резервной копии, если теневой (shadow) файл уже существует.

Исправил Д. Еманов

isql

([CORE-5569](#)) isql неправильно вставлял столбцы UNICODE_FSS/UTF8, когда они использовали сортировку.

Исправил A. dos Santos Fernandes

([CORE-5570](#)) Отрицательная бесконечность (дважды) неправильно показывалась без знака в isql.

Исправил A. dos Santos Fernandes

([CORE-5540](#)) Альтернативное выполнение isql из командной строки в режиме embedded работает в 5-10 раз медленнее, т. е. быстрее-медленнее-быстрее-медленнее...

Исправил В. Хорсун

nBackup

([CORE-5613](#)) Superserver может зависнуть при изменении физического состояния резервного копирования при высокой нагрузке.

Исправил В. Хорсун

Управление пользователями

([CORE-5651](#)) Проблема миграции пользователей в 3.0.

Исправил А. Пешков

Выпуск Firebird 3.0.2: Исправление ошибок

Следующие ошибки были исправлены до выпуска официального Firebird 3.0.2:

Ядро сервера выпуска 3.0.2

([**CORE-5501**](#)) Диагностика утилиты gstat была непонятной, когда поврежденная страница в файле БД была зашифрованной.

Исправил А. Пешков

([**CORE-5498**](#)) Не удаётся создать соединение для обработки событий.

Исправил В. Хорсун

([**CORE-5496**](#)) Создание SRP SYSDBA с явным указанием прав администратора (-admin yes в gsec или роли администратора при создании пользователя) создавало две учетные записи SYSDBA.

Исправил А. Пешков

([**CORE-5489**](#)) Плохая производительность фильтрации для NULL внутри сканирования навигационного индекса.

Исправил Д. Еманов

([**CORE-5474**](#)) Параметр «Restrict UDF» для параметра конфигурации UdfAccess оказался неэффективным из-за того, что fbudf.so динамически связывался с libc.

Исправили А. Пешков и A. dos Santos Fernandes

([**CORE-5472**](#)) Проблемы с передачей ключа dbcrypt.

Исправил А. Пешков

([**CORE-5470**](#)) Трассировка INCLUDE_FILTER с помощью [[: WHITESPACE:]]+ не будет работать, когда была выпущена инструкция,

содержащая символ новой строки.

Исправил В. Хорсун

([CORE-5464](#)) При чтении BLOB объекта, хранящегося в несовместимой кодировке, в fbclient возникает ошибка прав доступа (*access violation*).

Исправил В. Хорсун

([CORE-5457](#)) Bugcheck 167 (неверный запрос SEND).

Исправил Д. Еманов

([CORE-5456](#)) При попытке перенести базу данных из Firebird 2.5.2 в 3.0.1 восстановление gbak всегда завершается сбоем.

Исправил Д. Еманов

([CORE-5452](#)) При выгрузке динамической библиотеки ядра сразу после закрытия рабочих потоков (GC и/или записи кэша) возникает ошибка сегментации.

Исправил А. Пешков

([CORE-5448](#)) Ошибка при создании представления таблицы, в которой есть столбец с набором символов NONE, а в базе данных набор символов по умолчанию UTF8 с сортировкой по умолчанию UNICODE.

Исправил В. Хорсун

([CORE-5447](#)) Когда оператор EXECUTE STATEMENT начинается с комментария (символов '--') возникает ошибка “неожиданный конец команды” (*Unexpected end of command*) с неправильным номером столбца.

Исправил A. dos Santos Fernandes

([CORE-5432](#)) Classic Server подвергнулся регрессии производительности: слишком много времени тратится на переключение контекста.

Исправил Д. Еманов

([CORE-5422](#)) После отката может произойти ошибка «Overth buffer decompression». Это повлияло на все платформы, и, в частности, на Windows Superserver, когда GCPolicy установлен в «Combined» или «Cooperative», а CpuAffinityMask настроен на использование нескольких ядер.

Исправил В. Хорсун

([CORE-5421](#)) Снижение производительности в FB 3.0.2 по сравнению с FB 2.5.7.

Исправил Д. Еманов

([CORE-5414](#)) Ошибка восстановления на FB 3.0 из резервной копии FB 2.5: ошибка 221 (*cannot remap*).

Исправил Д. Еманов

([CORE-5410](#)) Зависимости не сохранялись при использовании определенных типов конструкций в подпрограммах.

Исправил A. dos Santos Fernandes

([CORE-5393](#)) Плохая оптимизация для некоторых операций с представлениями, содержащими подзапросы.

Исправил Д. Еманов

([CORE-5392](#)) BUGCHECK 179 (*decompression overran buffer*) или неожиданный конфликт блокировки могли произойти во время возврата версий записи.

Исправил Д. Еманов

([CORE-5390](#)) isql завершает работу с повреждением памяти, если длина инструкции превышает 10 МБ.

Исправил A. dos Santos Fernandes

([CORE-5384](#)) Менеджер подключаемого модуля не может предоставить соответствующую конфигурационную информацию, когда имя файла конфигурации установлено в файле `plugins.conf`.

Исправил А. Пешков

([CORE-5383](#)) Не распознаются зависимости в пакетах PSQL.

Исправил A. dos Santos Fernandes

([CORE-5382](#)) Инструкция SQL длиной более 10 МБ может быть обработана (усечена) неправильно.

Исправил Д. Еманов

([CORE-5381](#)) Регрессия: Выборка данных из представления, содержащего вложенное представление, не может быть выполнена.

Исправил Д. Еманов

([CORE-5368](#)) Регрессия: клиентское приложение может зависнуть или вызвать нарушение прав доступа при выборке из нескольких потоков с помощью одного встроенного соединения.

Исправил А. Пешков

([CORE-5367](#)) Регрессия: Утверждение **BOOLEAN** как параметр условия поиска неверно отклоняется как недопустимое.

Исправил А. dos Santos Fernandes

([CORE-5366](#)) Регрессия: Выражение **CASE** с более чем 255 условиями отклоняется.

Исправил А. dos Santos Fernandes

([CORE-5350](#)) Ошибка при вызове под Windows функции `CachedMasterInterface::set`.

Исправил В. Хорсун

([CORE-5314](#)) UDF, объявленные с большими размерностями `varchar`, требуют слишком много времени для выполнения.

Исправил А. dos Santos Fernandes

([CORE-3530](#)) Оператор **BETWEEN** не поддерживался для столбцов **COMPUTED BY** - «feature is not supported».

Исправил Д. Еманов

Сбои и зависания сервера выпуска 3.0.2

([CORE-5428](#)) Старая ошибка иногда может вызвать ошибку сегментации в архитектуре **Classic**, когда структура таблицы с триггерами была изменена (DDL), в то время как другое подключение к базе данных готовило запрос на работу с этой таблицей. Иногда указатели на старые триггеры кэшировались компилятором запроса и ошибка возникала при следующем использовании.

Исправил А. Пешков

([CORE-5425](#)) Нагрузочный тест зависал на сервере.

Исправил А. Пешков

([CORE-5417](#)) Зависание ядра сервера, если при остановке (*shutdown*) базы данных запускается автоматическая сборка мусора.

Исправил В. Хорсун

([CORE-1894](#)) (Старая ошибка) Циклические зависимости между вычисляемыми полями приводят к падению сервера.

Исправил А. dos Santos Fernandes

Сборка сервера выпуска 3.0.2

([CORE-5120](#)) Проблемы с функцией `snprintf` (в `autoconfing_msvc.h`) и определяемых пользователем литералов в VS 2015 (MSVC 14).

Исправил В. Хорсун

API выпуска 3.0.2

([CORE-5359](#)) Новый API - `getScale (MessageMetadata)` возвращает `INTEGER`, в то время как `setScale()` для `MetadataBuilder` - ожидается беззнаковым.

Исправил А. Пешков

Для операционных систем POSIX (выпуск 3.0.2)

([CORE-5503](#)) Сервер архитектуры Classic был ограничен 64 соединениями при использовании сценариев `initd` `systemd`.

Исправил А. Пешков

([CORE-5484](#)) База данных на общем ресурсе NFS недоступна при доступе с использованием пути точки монтирования.

Исправил А. Пешков

([CORE-5458](#)) Не удавалось подключиться к базе данных из-за ошибочных точек подключения NFS.

Исправил А. Пешков

Для операционных систем Windows (выпуск 3.0.2)

([CORE-5477](#)) Трассировка дублировала `ascii_char(13)` в выходных данных в Windows.

Исправил В. Хорсун

Выпуск Firebird 3.0.1: Исправление ошибок

Следующие ошибки были исправлены до выпуска официального Firebird 3.0.1:

Ядро сервера выпуска 3.0.1

([CORE-5355](#)) XrbBuilder мог создать новый TPB.

Исправил А. Пешков

([CORE-5351](#)) LEFT JOIN неправильно вводит UDF во внутренний поток, что приводит к неправильным результатам.

Исправил Д. Еманов

([CORE-5341](#)) Пользовательская сортировка не работала с BLOB

Исправил А. dos Santos Fernandes

([CORE-5339](#)) При выполнении проверки перезаписи может выполняться оператор createDatabase().

Исправил А. Пешков

([CORE-5337](#)) Подзапрос в выражении списка вставки игнорировал изменения, внесенные ранее в той же самой операции

Исправил В. Хорсун

([CORE-5329](#)) База данных может пострадать от частичного повреждения в режиме "использовать все пространство" (*no-reserve*).

Исправили В. Хорсун и Д. Еманов

([CORE-5327](#)) BUGCHECK 165 (не удастся найти страницу указателей - *cannot find tip page*)

Исправил В. Хорсун

([CORE-5322](#)) Каскадное удаление в таблице, ссылающейся сама на себя, может вызвать ошибку "нет текущей записи для операции выборки" ("no current record for fetch operation").

Исправил В. Хорсун

([CORE-5313](#)) При выполнении операции LIST происходит ошибка «тип данных неизвестен» (*"Data type unknown"*).

Исправил А. dos Santos Fernandes

([CORE-5307](#)) Случайные ошибки "недопустимый дескриптор транзакции" (*"invalid transaction handle"*) возвращались из ядра.

Исправил Д. Еманов

([CORE-5305](#)) ON UPDATE CASCADE переставал работать для внешнего ключа, ссылающегося на себя.

Исправил В. Хорсун

([CORE-5302](#)) Регрессия: производительность снижалась при массовой вставке в таблицу с индексами.

Исправил В. Хорсун

([CORE-5292](#)) Попытка зашифровать базу данных при отсутствии соответствующего ключа может привести к ее повреждению.

Исправил А. Пешков

([CORE-5289](#)) Небольшая утечка памяти может произойти при возврате пустого результирующего набора.

Исправил В. Хорсун

([CORE-5279](#)) Было нарушено предоставление привилегий представлениям.

Исправил А. Пешков

([CORE-5277](#)) Параметрам с многобайтовыми наборами символов разрешено превышать ограничение длины для полей VARCHAR.

Исправил A. dos Santos Fernandes

([CORE-5275](#)) Индекс выражения мог стать несгласованным, если оператор `CREATE INDEX` был прерван после создания В-дерева, но до подтверждения создания.

Исправил Д. Еманов

([CORE-5268](#)) Вложенные условия `OR` могут привести к неправильным результатам.

Исправил Д. Еманов

([CORE-5248](#)) Согласованность ролей и привилегий в синтаксисе оператора `GRANT` необходимо привести в соответствие со стандартом SQL.

Исправил А. Пешков

([CORE-5244](#)) Попытка выполнить инструкцию `SELECT`, используя `isc_dsqli_execute_immediate()` выдает неверное сообщение об ошибке.

Исправил А. Пешков

([CORE-5241](#)) Затронутые строки не учитывались для некоторых операций обновления с представлениями.

Исправил Д. Еманов

([CORE-5236](#)) Предикаты `IN/ANY/ALL` могут вызывать не оптимальное выполнение соединений (поздняя фильтрация).

Исправил Д. Еманов

([CORE-5233](#)) Не выровненный доступ к памяти в `MET_format ()` вызвал бы ошибку сегментации на машинах с обратным порядком байтов.

Исправил James Clarke

([CORE-5232](#)) 32-разрядные блокировки в системах с обратным порядком байтов приводят к взаимной блокировке (*deadlock*).

Исправил James Clarke

([CORE-5231](#)) Оператор EXECUTE STATEMENT выдает ошибку BLR, если количество выходных параметров превышает 256.

Исправил Д. Еманов

([CORE-5226](#)) Запрос с планом ORDER может вернуть неправильный результирующий набор (с отсутствующими записями) при перемещении по убывающему индексу.

Исправил Д. Еманов

([CORE-5225](#)) Если аутентификация с использованием первого плагина, заданного для пользователя, не заканчивалась успешно, то она переставала работать, вместо того, чтобы продолжиться со следующим плагином.

Исправил А. Пешков

([CORE-5222](#)) SELECT WITH LOCK может вызвать непредвиденные ошибки при конфликте обновления при параллельной работе.

Исправил Д. Еманов

([CORE-5210](#)) POST_EVENT не работал с сервером и клиентом Firebird 3.0.

Исправил В. Хорсун

([CORE-5213](#)) База данных может быть расшифрована после изменения нескольких байтов в её заголовке без '*согласия*' из плагина шифрования.

Исправил А. Пешков

([CORE-5193](#)) Проблема с приоритетом при использовании оператора IS.

Исправил А. dos Santos Fernandes

([CORE-5183](#)) Регрессия: Искажалась нумерация колонка/линия в сообщениях об ошибках при использовании в проблемном операторе синтаксиса `alias.name`.

Исправил Д. Еманов

([CORE-5142](#)) Вызывалась ошибка «нет текущей записи для извлечения» (*“no current record to fetch”*), если какая-то запись должна была быть удалена как самим оператором, так и каким-то триггером, запущенным во время выполнения инструкции.

Исправил В. Хорсун

([CORE-1746](#)) При вставке данных в таблицу можно было создать индекс по выражению для неё.

Исправил В. Хорсун

Сбои сервера выпуска 3.0.1

([CORE-5234](#)) Нарушение прав доступа из определенных UDF приводит к падению сервера.

Исправил В. Хорсун

API/Удаленный интерфейс выпуска 3.0.1

([CORE-5347](#)) Сервер может отклонить соединение при получении первого сжатого пакета данных.

Исправил А. Пешков

([CORE-5296](#)) Ошибка сетевого протокола при выполнении обратного вызова клиента для получения ключа шифрования базы данных.

Исправил А. Пешков

([CORE-5278](#)) Некоторые параметры SPB приводят к ошибкам в Firebird 3.0.

Исправил А. Пешков

([CORE-5224](#)) Дескрипторы идентификаторов транзакций для API служб не поддерживали новые 48-разрядные идентификаторы транзакций.

Исправил Д. Еманов

([CORE-5154](#)) Интерфейс API служб не мог работать с именами баз данных, использующих отличный от ASCII набор символов.

Исправил Д. Сибиряков

Утилиты выпуска 3.0.1

fbsvcmgr

([CORE-5270](#)) fbsvcmgr не вызывал ошибок при попытке закрыть базу данных без указанного таймаута (prp_force_shutdown N).

Исправил А. Пешков

gbak

([CORE-5228](#)) Восстановление может зависнуть, если база данных содержит более 4 миллиардов записей.

Исправил Д. Еманов

gfix

([CORE-5295](#)) Проверка может продолжаться после окончания файла при работе с многофайловой базой данных.

Исправил В. Хорсун

isql

([CORE-5308](#)) Команда BLOBVIEW вызывала зависание isql.

Исправил А. Пешков

([CORE-5294](#)) При использовании оператора SHOW GRANTS в isql для новой пустой базы данных возникала утечка памяти.

Исправил А. Пешков

([CORE-5220](#)) При извлечении метаданных утилита isql -[e]X[tract_metadata] пропускала двойные кавычки в предложении COLLATE <collation> оператора CREATE DOMAIN, если <collation> использовал набор символов, отличный от ASCII.

Исправил A. dos Santos Fernandes

([CORE-5218](#)) При извлечении метаданных утилита isql -[e]X[tract_metadata] не экспортировала явно определенные имена NOT NULL ограничений для баз данных в 3-ем диалекте.

Исправил Д. Еманов

([**CORE-5217**](#)) При извлечении метаданных в утилите isql -[e]X[tract_metadata] может произойти сбой при экспорте исключения с длиной текста сообщения длиной более 127 байт.

Исправил Д. Еманов

([**CORE-5207**](#)) При извлечении метаданных утилита isql -[e]X[tract_metadata] может генерировать неверные заявления GRANT USAGE для доменов

Исправил Д. Еманов

Мониторинг и трассировка

([**CORE-5246**](#)) При выборе из таблиц MON\$ возникает ошибка усечения строки, если определённая пользователем контекстная переменная превышает размер в 255 байт.

Исправил Д. Еманов

([**CORE-5203**](#)) Сеанс трассировки может полностью использовать одиночный процессор (CPU).

Исправил А. Пешков

nBackup

([**CORE-5264**](#)) База данных, расположенная на RAW-устройстве (например, ZFS), не может быть разблокирована из состояния nBackup-locked.

Исправил А. Пешков

Сборка сервера выпуска 3.0.1

([CORE-5348](#)) Базы данных, созданные под управлением Windows, принадлежат администратору Windows, а не SYSDBA.

Исправил В. Хорсун

([CORE-5284](#)) Firebird не удавалось создать с ключом USE_VALGRIND.

Исправил А. Пешков

Особенности для операционных систем POSIX выпуска 3.0.1

([CORE-5335](#)) При использовании `op_que_events` порта, который не был подготовлен для обработки событий, может возникнуть ошибка сегментации (*segfault*).

Исправил А. Пешков

([CORE-5285](#)) Могла произойти ошибка сегментации (*segfault*), если соединение было закрыто, в то время как запросы оставались все ещё открытыми.

Исправил А. Пешков

([CORE-5249](#)) Неправильное использование системного вызова `readlink()`.

Исправил А. Пешков

Выпуск Firebird 3.0.0: Исправление ошибок

Следующие ошибки были исправлены до выпуска официального Firebird 3.0.0:

Ядро сервера выпуска 3.0.0

([**CORE-5184**](#)) При попытке сохранить информацию об исключении в интерфейсе состояния в диспетчере cloop будет выполняться утверждение (*assertion*).

Исправил А. Пешков

([**CORE-5182**](#)) Неспособность найти запись при повторной выборке (*re-fetch*) данных приводило к непоследовательным последствиям при последующем выполнении инструкции.

Исправил Д. Еманов

([**CORE-5166**](#)) Нарушение уникального ключа для столбца типа BOOLEAN возвращало ошибочное сообщение.

Исправил A. dos Santos Fernandes

([**CORE-5165**](#)) HOWING COUNT (*) NOT IN (<Q>) ошибочно исключает запись из внешнего результирующего набора. (<Q> представляет собой набор результатов без значений NULL)

Исправил A. dos Santos Fernandes

([**CORE-5161**](#)) Можно было создать уникальный индекс по не уникальным данным.

Исправил В. Хорсун

([CORE-5159](#)) Регрессия: ядро сервера может выдавать ошибки транслитерации при работе с не ASCII символами и строками.

Исправил A. dos Santos Fernandes

([CORE-5157](#)) При отправке данных с клиента на сервер небольшими пакетами сервер может переходить в бесконечный цикл.

Исправил А. Пешков

([CORE-5149](#)) Регрессия: LEFT JOIN неправильно обрабатывал COALESCE во внутреннем потоке, выдавая неправильные результаты.

Исправил Д. Еманов

([CORE-5147](#)) Выполнение оператора CREATE TRIGGER вызывает ошибку «Неоднозначность имён полей между таблицами В и А» при их (имён) совпадении.

Исправил A. dos Santos Fernandes

([CORE-5146](#)) Порядок соединения был не оптимальным, если одна таблица имеет селективный предикат, а для другой был рассчитан минимальный (MIN).

Исправил Д. Еманов

([CORE-5144](#)) Когда база данных шифруется или расшифровывается при высокой параллельной нагрузке, может возникнуть взаимная блокировка (deadlock).

Исправил А. Пешков

([CORE-5130](#)) Незначительная регрессия: При попытке скомпилировать представление как с подзапросом, так и с предложением WITH CHECK

OPTION в его определении, возвращалось неверное сообщение об ошибке - недопустимое сочетание.

Исправил A. dos Santos Fernandes

([CORE-5133](#)) Оператор ALTER SEQUENCE RESTART WITH не мог изменить начальное значение.

Исправил Д. Еманов

([CORE-5122](#)) Оптимизатор не использует индекса выражения по строке, если запрос выполняется в соединении с использованием набора символов, отличным от набора символов, использовавшегося при создании индекса.

Исправил Д. Еманов

([CORE-4645](#)) Внутренняя проверка целостности Firebird (не найдена страница указателей (165), файл: tra.cpp line: 2375), если слишком много транзакций было создано для одного соединения с базой данных только для чтения. Это была очень старая ошибка, унаследованная от InterBase 6.

Исправил В. Хорсун

Сбои сервера выпуска 3.0.0

([CORE-5153](#)) Регрессия: сервер даёт сбой, когда агрегатные функции используются вместе с предикатом NOT IN.

Исправил A. dos Santos Fernandes

Утилиты выпуска 3.0.0

gbak

([CORE-5143](#)) Использование утилиты gbak для восстановления базы данных с ключом -O (NE_AT_A_TIME) может завершиться неудачей, когда

какая-либо функция SQL обращается к таблице,.

Исправил Д. Еманов

gfix

([CORE-5140](#)) Попытка установить количество буферов страниц в не поддерживаемое значение возвратит неправильное сообщение об ошибке.

Исправил В. Хорсун

isql

([CORE-5194](#)) При выполнении операции извлечения метаданных генерировалось недействительное предложение COMPUTED BY.

Исправил A. dos Santos Fernandes

Мониторинг

([CORE-5179](#)) Статистика уровня базы данных (MON\$STAT_GROUP = 0) при одновременной нагрузке может содержать несколько строк.

Исправил Д. Еманов

Управление пользователями

([CORE-5189](#)) Коды операций для плагинов управления пользователями отсутствовали в общедоступном API.

Исправил А. Пешков

([CORE-5162](#)) Пользователи с одним и тем же именем могут сосуществовать в базе данных безопасности, если они были созданы разными плагинами. Однако, теги или атрибуты для пользователя, созданного с использованием одного плагина, отображались в виртуальных таблицах SEC\$ для

пользователя с тем же именем, созданного другим плагином.

Исправил А. Пешков

([CORE-5155](#)) Оператор [CREATE OR] ALTER USER не работает, если параметр PASSWORD присутствует и не был указан сразу после параметра USER.

Исправил А. Пешков

Сборка сервера выпуска 3.0.0

([CORE-5181](#)) Сборка завершается неудачей, если не был определен параметр HAVE_MMAP.

Исправил А. Пешков

([CORE-5180](#)) Для сборки сервера для Linux излишне требовались права root.

Исправил А. Пешков

([CORE-5134](#)) Примеры dbcrypt плагинов содержали ссылки на внутренние файлы.

Исправил А. Пешков

Firebird 3.0 релиз-кандидат 2: Исправление ошибок и незначительные улучшения

Следующие ошибки были исправлены до выпуска релиз-кандидата 2 (Release Candidate 2, в дальнейшем RC2) Firebird 3.0 :

Незначительные улучшения

([CORE-4996](#)) **Новое:** функция обновления SECURITY2.FDB →

SECURITY3.FDB

Реализация - А. Пешков

([CORE-1538](#)) **Новое:** В функцию `isc_database_info()` API была добавлена опция для возврата количества свободных страниц в базе данных.

Реализация - В. Хорсун

([CORE-5112](#)) **Улучшение:** Улучшены сообщения об ошибках для некорректной SQLDA.

Реализация - Д. Сибиряков

([CORE-5100](#)) **Улучшение:** Улучшен контроль над ключами шифрования базы данных. См. [Задачи шифрования](#).

Реализация - А. Пешков

([CORE-5083](#)) **Улучшение:** Был предложен способ установки правильных смещений в пользовательской реализации интерфейса `ImessageMetadata`.

Реализация - А. Пешков

([CORE-5077](#)) **Улучшение:** В `isql` команда `SHOW DATABASE` показывает состояние шифрования базы данных, равно как и `gstat -h[eaders]`.

Реализация - А. Пешков

([CORE-5063](#)) **Улучшение:** В `isql` команда `SHOW DATABASE` показывает количество используемых и свободных страниц в базе данных.

Реализация - В. Хорсун

([CORE-5028](#)) **Улучшение:** В таблицу мониторинга `MON$ATTACHMENTS`

был добавлен номер удаленного порта.

Реализация - В. Хорсун

([**CORE-4978**](#)) **Улучшение:** Улучшена проверка и восстановление потерянных страниц данных.

Реализация - Р. Симаков и В. Хорсун

([**CORE-2493**](#)) **Улучшение:** В сообщения об ошибках в файле firebird.log для TCP соединений теперь добавляется IP адрес удаленного узла.

Реализация - В. Хорсун

Ошибки

Ядро сервера выпуска 3.0 RC2

([**CORE-5110**](#)) Сообщалось о ложном нарушении первичного или внешнего ключа при использовании флага `isc_dpb_no_garbage_collect` для соединения.

Исправил В. Хорсун

([**CORE-5104**](#)) `SELECT` из внешнего источника данных выполняется медленно.

Исправили В. Хорсун и А. Пешков

([**CORE-5097**](#)) Выражения `COMPUTED BY` не преобразовывались в свой тип данных внутри ядра сервера.

Исправил А. dos Santos Fernandes

([CORE-5093](#)) ALTER COLUMN не изменял тип вычисляемого столбца.

Исправил A. dos Santos Fernandes

([CORE-5082](#)) Сервер не проверял правильность пары пользователь/пароль, используемой в операторе EXECUTE STATEMENT.

Исправил A. Пешков

([CORE-5078](#)) При изменении типа столбца с длинного VARCHAR на текстовый BLOB неожиданно выдавалась ошибка "Недопустимый идентификатор BLOB" ("Invalid BLOB ID").

Исправил B. Хорсун

([CORE-5075](#)) Триггер ON DISCONNECT с EXECUTE STATEMENT не работал и приводил к сбою сервера при его пересоздании.

Исправил B. Хорсун

([CORE-5062](#)) Функция CHAR_TO_UUID в индексированном столбце вызывает ошибку «оценка выражения не поддерживается» ("*expression evaluation not supported*").

Исправил A. dos Santos Fernandes

([CORE-5060](#)) Не удаётся создать представление (оператор CREATE VIEW) выборки из системной таблицы, несмотря на наличия всех доступные прав.

Исправил A. Пешков

([CORE-5056](#)) Блокировка записи в файл базы данных очищалась при получении статистики базы данных через API интерфейс служб.

Исправил A. Пешков

([CORE-5049](#)) Некорректно вычислялась размерность столбцов для представлений.

Исправил A. dos Santos Fernandes

([CORE-5047](#)) `mapUser()` устанавливало дополнительное соединение с использованием неверного поставщика.

Исправил А. Пешков

([CORE-5039](#)) Подключение к службе с недопустимым именем службы приводило к неправильному сообщению об ошибке.

Исправил А. Пешков

([CORE-5034](#)) При отключении могла возникнуть задержка не менее 5 секунд, если отключение последовало сразу за инициализацией диспетчера событий.

Исправил В. Хорсун

([CORE-5033](#)) Файл `fb_user_mapping` можно было оставить в каталоге блокировки Firebird после правильного завершения всех рабочих процессов Firebird.

Исправил В. Хорсун

([CORE-5032](#)) Невозможно осуществить подключение к базе данных со своей собственной базой данных безопасности при отсутствии `security3.fdb`.

Исправил А. Пешков

([CORE-5026](#)) Проверка подлинности СУБД Firebird будет прекращена после неудачной попытки установить авторизацию через первый общий плагин.

Исправил А. Пешков

([CORE-5020](#)) Предложение ORDER BY для ключевых столбцах составного индекса может отключить использование других индексов.

Исправил Д. Еманов

([CORE-5018](#)) Не индексируемые предикаты, возможно, не применялись сразу после извлечения данных при соединении таблиц.

Исправил Д. Еманов

([CORE-5017](#)) Прерывание аих соединения во время фазы настройки TCP приводило к сбою сервера с необработанным исключением (без записи в файл firebird.log).

Исправил А. Пешков

([CORE-4997](#)) При попытке двух потоков одновременно первым установить встроенное подключение к серверу между ними происходил конфликт (или, выражаясь образно, «соревнование»).

Исправил А. Пешков

([CORE-4991](#)) Плагин Keyholder загружался при подключении к базе данных, но не при создании базы данных.

Исправил А. Пешков

([CORE-4980](#)) Оператор REVOKE может изменять привилегии, предоставленные системным таблицам во время создания базы данных.

Исправил А. Пешков

([CORE-4840](#)) Транзакции с isc_tpb_autocommit могут зависнуть на сервере.

Исправил Д. Еманов

([CORE-4969](#)), При возникновении ошибки в любом настроенном подключаемом модуле UserManager таблица SEC\$USERS становится недоступной.

Исправил А. Пешков

([CORE-4964](#)) При подключении к базе данных безопасности реальные ошибки скрывались менеджером пользователя Srp.

Исправил А. Пешков

([CORE-4885](#)) Получение текущего имени пользователя при использовании доверенной аутентификации Windows или имени пользователя операционной системы, содержащем символы, отличные от ASCII, может вызвать ошибку.

Исправили А. Пешков и Д. Еманов

([CORE-2923](#)) Проблема с зависимостями представлений (VIEW).

Исправил А. dos Santos Fernandes

Сбои сервера выпуска 3.0 RC2

([CORE-5107](#)) При использовании SET AUTODDL OFF и последовательности команд CREATE + ALTER + DROP VIEW приводило к сбою сервера.

Исправил А. dos Santos Fernandes

([CORE-5087](#)) Выключение базы данных может привести к сбою сервера при запуске нескольких приложений EXECUTE STATEMENT.

Исправил В. Хорсун

([CORE-5016](#)) При выполнении удаления после добавления нового ссылочного столбца во время сборки мусора может произойти сбой сервера.

Исправил Д. Еманов

([CORE-5015](#)) При отправке события из триггера ON DISCONNECT может произойти сбой сервера.

Исправил В. Хорсун

([CORE-4982](#)) Сервер может аварийно завершить работу во время выборки данных из процедуры, если срабатывало исключение при резервном копировании текущей точки сохранения.

Исправил Д. Еманов

Шифрование базы данных выпуска 3.0 RC2

([CORE-5105](#)) При первоначальном шифровании базы данных могут возникать блокировки.

Исправил А. Пешков

([CORE-5044](#)) Для каждой тени страницы шифруются по отдельности.

Исправил А. Пешков

([CORE-4999](#)) Попытка получить доступ к базе данных с недопустимым ключом шифрования приводит к взаимоблокировке (deadlock).

Исправил А. Пешков

([CORE-4990](#)) Базы данных полностью не расшифровываются.

Исправил А. Пешков

([CORE-5041](#)) База данных может быть повреждена после нескольких запусков операторов ALTER DATABASE ENCRYPT/DECRYPT на открытой базе данных.

API/Удаленный интерфейс выпуска 3.0 RC2

([CORE-5102](#)) Вызов API fbclient.dll (неверно) изменял управляющее слово FPU.

Исправил В. Хорсун

([CORE-5067](#)) Из-за слишком длительной продолжительности сборки мусора зачинок базы данных безопасности новые подключения блокировались.

Исправил В. Хорсун

([CORE-5042](#)) Клиент FB3 не может создать базу данных под сервером FB2.5.

Исправил А. Пешков

([CORE-5029](#)) Когда многопоточное приложение использует протокол XNET может произойти сбой в библиотеке fbclient.

Исправил В. Хорсун

([CORE-4998](#)) Ни клиенту, ни серверу не удавалось закрыть подключение после неудачной проверки подлинности.

Исправил В. Хорсун

Утилиты выпуска 3.0 RC2

gbak

([**CORE-5101**](#)) Восстановление базы данных происходит медленно, если она содержала много небольших таблиц с индексами.

Исправил В. Хорсун

gstat

([**CORE-5088**](#)) При закрытии **gstat -h** может возникнуть ошибка сегментации.

Исправил А. Пешков

isql

([**CORE-5092**](#)) Команда **EXTRACT** теряет типы данных для столбцов **COMPUTED BY**.

Исправил А. dos Santos Fernandes

([**CORE-5089**](#)) Извлечение метаданных (**isql -x**): оператор «**CREATE PROCEDURE/FUNCTION**» сохранит ссылку **TYPE OF** для столбца таблицы, хотя столбец больше не существует, если в процедуре или функции был указан параметр этого типа, объявленный в момент его создания.

Исправил А. dos Santos Fernandes

([**CORE-5061**](#)) Выходной план неожиданно усекается после того, как запрос был упрощён путем его сокращения.

Исправил Д. Еманов

Мониторинг

([CORE-4708](#)) Содержание MON\$EXPLAINED_PLAN в MON\$STATEMENTS усекалось, если оно превысило предел в 32KB.

Исправил Д. Еманов

([CORE-4179](#)) Таблицы MON\$ становились недоступны при интенсивной загрузке дисковой системы.

Исправил Д. Еманов

Сборка сервера и программа установки выпуска 3.0 RC2

([CORE-5053](#)) (POSIX): скрипт changeServerMode.sh может изменить конфигурацию сервера.

Исправил А. Пешков

([CORE-5002](#)) (WINDOWS): В установщике Windows выявлена путаница кодировок (стандарт ANSI и UTF8) в его GUI для не англоязычной среды.

Исправил Р. Reeves

([CORE-4910](#)) (WINDOWS): Сборка для Windows не будет выполнена, если в путях к исходным файлам содержатся пробелы.

Исправил Д. Сибиряков

([CORE-4862](#)) (WINDOWS): ibase.h из пакета x64 включал в себя несуществующий файл.

Исправил Д. Еманов

([CORE-4842](#)) (POSIX): В некоторых средах Linux сценарий установщика завершался ошибкой.

Исправили А. Пешков и М. Masyk

Firebird 3.0 релиз-кандидат 1: Исправление ошибок и незначительные улучшения

Следующие ошибки были исправлены до выпуска релиз-кандидата (Release Candidate 1, в дальнейшем RC1) Firebird 3.0 :

Незначительные улучшения

([**CORE-4911**](#)) **Улучшение:** Большинство ошибок, происходящих в провайдерах, являются фатальными, и бессмысленно пытаться использовать других поставщиков в надежде, что они будут работать, если не работает этот. Например, если удаленный перенаправитель обнаруживает, что формат строки подключения к базе данных соответствует определенному протоколу, и соединение по этому протоколу не удалось, то попытка использовать других поставщиков будет бесполезной. Или, предположим, что сервер сообщает о некоторой внутренней ошибке базы данных (кроме недействительной версии ODS), тогда будет бесполезно пытаться открыть одну эту базу данных через провайдер loopback.

Лучшим вариантом оказалось перечисление кодов тех ошибок, после которых имело бы смысл попробовать другие провайдеры - это прежде всего `isc_unavailable` и `isc_no_priv`. С учетом желательности сохранения содержимого вектора состояния "как есть", было принято решение добавить еще несколько кодов ошибок.

Реализация - А. Пешков

([**CORE-4898**](#)) **Улучшение:** Если в базе данных имеется много функций, то их создание и загрузка функций было медленными. Теперь этот процесс получил ускорился.

Реализация - А. dos Santos Fernandes

([**CORE-4880**](#)) **Улучшение:** Ускорилось создание пакетов, содержащих много функций.

Реализация - A. dos Santos Fernandes

([CORE-4939](#)) **Улучшение:** Операции ввода-вывода в файле резервной копии `pbbackup` теперь выровнены по границе размера страницы.

Реализация - В. Хорсун

([CORE-4936](#)) **Улучшение:** Уменьшена внутренняя конкуренция в фоновом сборщике мусора.

Реализация - В. Хорсун

([CORE-4935](#)) **Улучшение:** Улучшена производительность при отказе от версии несуществующей записи или удалении недавно созданной версии первичной записи, когда цепочка версий слишком длинная.

Реализация - В. Хорсун

Ошибки

Ядро сервера выпуска 3.0 RC1

([CORE-4984](#)) Сортировка по составному индексу вместе с условием диапазона давало неправильные результаты.

Исправил Д. Еманов

([CORE-4954](#)) Упакованная процедура с входным значением по умолчанию не вызывается, если не указан параметр по умолчанию.

Исправил A. dos Santos Fernandes

([CORE-4952](#)) SuperServer мог зависнуть при закрытии (*shutdown*) работы с базой данных.

Исправил В. Хорсун

([CORE-4951](#)) При попытке фиксации транзакции двухфазной фиксации выдаётся сообщение: "Incorrect parameters provided to internal function INF_transaction_info Invalid token" ("неправильные параметры, предоставленные внутренней функции Inf_transaction_info недопустимый маркер").

Исправил А. Пешков

([CORE-4950](#)) Для любого системного индекса статистика не может быть пересчитана или обновлена.

Исправил А. dos Santos Fernandes

([CORE-4947](#)) Составной оператор ALTER TABLE при попытке выполнить операции ADD и DROP для одного и того же ограничения терпит неудачу.

Исправил А. dos Santos Fernandes

([CORE-4943](#)) Кастинг даты в строку в 1-м диалекте работает неправильно, если в хранимой процедуре используется домен с проверкой ограничений.

Исправил А. dos Santos Fernandes

([CORE-4938](#)) Не удаётся выполнить операции (получение статистики, изменение режима принудительной записи, резервное копирование) в базе данных при работе со встроенным сервером (embedded mode).

Исправил В. Хорсун

([CORE-4929](#)) Не компилируется код PSQL с блоком "ELSE IF (<expr>) THEN", содержащим команды управления явным курсором.

Исправил Д. Еманов

([CORE-4921](#)) Предикат IS [NOT] DISTINCT FROM внутри объединений/агрегатов приводил к использованию суб-оптимальных планов.

Исправил Д. Еманов

([CORE-4917](#)) Команда ALTER DOMAIN ... TO <new_name> позволяет указать <new_name> в таком же шаблоне, как у системных доменов, т.е. совпадающий с 'RDB \$ [[: DIGIT:]] * '.

Исправил A. dos Santos Fernandes

([CORE-4914](#)) Перевод базы данных в состояние "full shutdown" может привести к повреждению индексов для длинных текстовых столбцов.

Исправил В. Хорсун

([CORE-4909](#)) Операции MERGE/HASH JOIN приводят к неправильным результатам, если ключи соединения VARCHAR отличаются только конечными пробелами.

Исправил Д. Еманов

([CORE-4906](#)) Остановка сервера с активным сеансом трассировки происходит с ошибкой - нарушение прав доступа.

Исправил В. Хорсун

([CORE-4904](#)) При вставке данных в поле с длинными ключами может произойти повреждение индекса.

Исправил В. Хорсун

([CORE-4902](#)) Ошибка утверждения (m_format -> fmt_length == m_data.getCount())

Исправил Д. Еманов и В. Хорсун

([CORE-4897](#)) При восстановлении базы данных служба `fbsvcmgr` может создать поврежденный журнал.

Исправил А. Пешков

([CORE-4890](#)) При размещении комментария (однострочного или многострочного) между конечным оператором и символом терминатора (^) в коде хранимой процедуры происходит ошибка компиляции, если программы используют старый API.

Исправил А. Пешков

([CORE-4889](#)) Использование службы `fbsvcmgr` с ключом `action_trace_start` под `SuperServer` вызовет блокировку в реальном времени и приведёт к зависанию соединений, использующих локальный протокол.

Исправил В. Хорсун

([CORE-4887](#)) Триггер `AFTER CREATE/ALTER PACKAGE DDL` запускается до вставки и обновления системных таблиц `RDB$PROCEDURES` и `RDB$FUNCTIONS`, тогда как он должен запускаться после всех изменений в системных таблицах.

Исправил А. dos Santos Fernandes

([CORE-4884](#)) Скрипт, содержащий оператор `EXECUTE BLOCK` с несколькими вложенными предложениями `BEGIN..END`, вызывает сбой во время синтаксического анализа.

Исправил А. dos Santos Fernandes

([CORE-4872](#)) Библиотека `fbclient` не может работать через удалённый протокол с серверами более ранней версии, чем Firebird 2.5.

Исправил Д. Еманов

([CORE-4861](#)) Ошибка сегментации при работе с сохранённым исключением в запросе.

Исправил А. Пешков

([CORE-4854](#)) Клиентская библиотека некорректно обрабатывала не-UTF8 представление международных символов в SPB.

Исправил А. Пешков

([CORE-4848](#)) `MERGE ... WHEN NOT MATCHED ... RETURNING` возвращал неправильные (не нулевые) значения, когда вставка данных не была выполнена.

Исправил А. dos Santos Fernandes

([CORE-4846](#)) Попытка изменить триггер, срабатывающий на события в таблице T1, для срабатывание на события в таблице T2 заканчивается неудачей. При этом выдаётся странное сообщение: «Trigger table can not be modified».

Исправил А. dos Santos Fernandes

([CORE-4836](#)) `GRANT UPDATE(c) ON T TO U01 WITH GRANT OPTION`: Пользователь U01 не сможет отозвать эту привилегию, если до этого был выполнен любой оператор DML для этой таблицы.

Исправил А. Пешков

([CORE-4710](#)) Неправильное сообщение об ошибке “Invalid request BLR at

offset 361 context already in use (BLR error)”, возникающее только тогда, когда счетчик контекста находится рядом с пределом 256.

Исправил Д. Еманов

([CORE-4680](#)) После использования событий при подсоединении к базе данных может возникнуть взаимная блокировка (deadlock).

Исправил В. Хорсун

([CORE-4585](#)) Если столбец основан на домене, то не удаётся создать ограничение (CHECK) его проверки.

Исправил А. dos Santos Fernandes

([CORE-4497](#)) Ухудшение: неправильная обработка курсора в операторе FOR, если выражение NOT EXISTS (select * from <view>) использовалось для проверки результатов, полученных из хранимой процедуры.

Исправил Д. Еманов

([CORE-4292](#)) Сервер игнорирует асинхронные запросы (мониторинг или отмена) при подготовке запроса с большим количеством оконных функций.

Исправил А. dos Santos Fernandes

([CORE-4279](#)) При создании базы данных с указанием isc_dpb_lc_ctype = OCTETS выдаётся неверное сообщение об ошибке: “CHARACTER SET OCTETS is not defined”.

Исправил А. dos Santos Fernandes

([CORE-4277](#)) После создания базы данных с определённым набором символов по умолчанию нельзя создать столбец типа VARCHAR без явного указания набора символов.

Исправил A. dos Santos Fernandes

([CORE-4276](#)) Создание таблицы приведет к ошибке, если столбец был определен с набором символов DOS775.

Исправил A. dos Santos Fernandes

([CORE-4107](#)) Выполнение операции (вложенный запрос + производная таблица + объединение) приводит к неправильному набору результатов при использовании оператора GROUP BY.

Исправил Д. Еманов

([CORE-3717](#)) Существует нелогичность между тремя методами управления пользователями в отношении их допустимых имен и паролей.

Исправил А. Пешков

([CORE-3545](#)) Противоречивая проверка ограничений домена в PSQL.

Исправил A. dos Santos Fernandes

Особенности для операционных систем POSIX 3.0 RC1

([CORE-4919](#)) На операционных системах AIX и Solaris все процессы на классическом сервере зависали после принудительного завершения одного из процессов.

Исправил А. Пешков

Особенности для операционных систем Windows 3.0 RC1

([**CORE-4859**](#)) После удаления папки C:\ProgramData\firebird и установки сервера (из архива) в любую другую папку встроенный сервер не работает.

Исправил В. Хорсун

([**CORE-4860**](#)) Проверка базы данных в Windows завершается неудачно, если аргумент **dbname** содержит косую черту (/) и одновременно существует соединение, которое также использовало '/' при подключении.

Исправил В. Хорсун

Сервисы API 3.0 RC1

([**CORE-4879**](#)) При использовании сервисов API для управления пользователями (с помощью **fbsvcmgr** или путем построения SPB в какой-либо программе, например) требовалось (но не документировалось явно) поместить параметр имени пользователя сразу после нужного действия (добавить/изменить). Если перед именем пользователя был помещен какой-либо другой параметр, то диспетчер служб на сервере воспринимает это неправильно и построит ошибочную командную строку для утилиты **gsec**.

Исправил А. Пешков

Утилиты 3.0 RC1

fbsvcmgr

([**CORE-4855**](#)) Проверка базы данных во время действия DML в другом соединении приводит к ошибкам: "Error while trying to read from file" и "Page in use during flush (210)" ("Ошибка при попытке чтения из файла" и "Страница используется во время сброса (210)").

Исправил В. Хорсун

([CORE-4876](#)) Именованный трассированный сеанс, запущенный fbsvcmgr с не пустым значением для параметра `trc_name`, не может быть остановлен с использованием его имени.

Исправил В. Хорсун

gbak

([CORE-4928](#)) Информация о соединении в триггере on CONNECT не может быть сохранена, если соединение было установлено **gbak**.

Исправил A. dos Santos Fernandes

gfix

([CORE-4899](#)) При использовании **gfix -online** в Classic сервере выдаётся сообщение: "IProvider::attachDatabase failed when loading mapping cache", если доступ к базе данных осуществлялся с использованием удаленного протокола.

Исправил А. Пешков

([CORE-4865](#)) Онлайн-проверка может возвращать ложный отчёт о двойных выделенных страницах.

Исправил В. Хорсун

([CORE-3548](#)) Утилита **gfix** возвращает ошибку после корректного завершения работы с базой данных.

Исправил А. Пешков

isql

([CORE-4882](#)) Команда `isql INPUT` (или опция `-i` в командной строке) неправильно читала большие (>64K) строки.

Исправил А. dos Santos Fernandes

([CORE-4864](#)) Ошибка создания базы данных в `isql`.

Исправил А. Пешков

([CORE-4870](#)) В `isql SET COUNT ON` сообщает неправильное количество затронутых строк при обновлении представления, созданного с опцией `WITH CHECK OPTION`.

Исправил А. dos Santos Fernandes

Firebird 3.0 вторая бета версия: Исправление ошибок

Следующие улучшения и исправления ошибок были сделаны во второй бете версии Firebird 3.0.0 (в дальнейшем 3.0 Бета 2):

Ядро сервера 3.0 Бета 2

([CORE-4819](#)) Сервер неправильно проверял предложения `RETURNING_VALUES` для `EXECUTE PROCEDURE` и `INTO` для `EXECUTE STATEMENT`, что могло привести к ошибкам.

Исправил А. dos Santos Fernandes

([CORE-4809](#)) Операторы `HASH/MERGE JOIN` не использовались для более чем двух потоков, если они были объединены с помощью предложений `USING/NATURAL` и соединение было основано на конкатенации `DBKEY`.

Исправил Д. Еманов

([CORE-4806](#)) Ухудшение: генераторы могли быть модифицированы

непривилегированными пользователями.

Исправил Д. Еманов

[**\(CORE-4798\)**](#) Ухудшение: функции MIN() и MAX() при использовании в запросе соединений (join) игнорируют возможную навигацию по индексу.

Исправил Д. Еманов

[**\(CORE-4789\)**](#) Данные типа timestamp, которые были вне допустимого диапазона, не были запрещены к приведению к типам VARCHAR или CHAR.

Исправил A. dos Santos Fernandes

[**\(CORE-4786\)**](#) При попытке добавить в таблицу строку с дубликатом первичного/уникального ключа, имеющего длину более 127 символов, в сообщении об ошибке не указывалось имя проблемного ключа.

Исправил Д. Еманов

[**\(CORE-4781\)**](#) Максимальная длина строки (32765 байт) не проверялась синтаксическим анализатором.

Исправил A. dos Santos Fernandes

[**\(CORE-4774\)**](#) Требовалось указать необязательный алиас для таблицы при её обновлении с возвратом (RETURNING) псевдостолбцов (RDB\$).

Исправил A. dos Santos Fernandes

[**\(CORE-4768\)**](#) Выполнение операции CREATE USER ... TAGS (argument_1

= 'value1', ..., argument_N = 'valueN') возвращает неправильные результаты при указании большого количества аргументов.

Исправил А. Пешков

[\(CORE-4767\)](#) При выполнении операции CREATE USER ... TAGS (attr = 'prefix #suffix') наличие символа "#" в значении атрибута приводит к удалению следующих за ним символов.

Исправил А. Пешков

[\(CORE-4760\)](#) Нельзя создать пользователя с многобайтовыми не ASCII символами в имени.

Исправил А. Пешков

[\(CORE-4759\)](#) При построении индекса по выражению может возникнуть ошибка при восстановлении базы данных из резервной копии (см. также [CORE-4671](#)).

Исправил В. Хорсун

[\(CORE-4755\)](#) Если число аргументов в параметризованном исключении больше семи, то при его создании выдаётся неверное сообщение.

Исправил А. dos Santos Fernandes

[\(CORE-4754\)](#) Работа с GTT одновременно из нескольких подключений к базе данных (с использованием оператора EXECUTE STATEMENT с предложением ON EXTERNAL и разными ролями) могут привести к ошибке "Internal Firebird consistency check (invalid SEND request (167), file: JrdStatement.cpp line: 325)".

Исправил В. Хорсун

[\(CORE-4753\)](#) Сервер мог зависнуть во встроенном режиме.

Исправил В. Хорсун

[\(CORE-4752\)](#) Инструкция EXECUTE STATEMENT с использованием параметров типа BLOB вызывает ошибку "Invalid BLOB ID".

Исправил A. dos Santos Fernandes

[\(CORE-4747\)](#) При получении текста из MON\$STATEMENTS.MON\$SQL_TEXT с использованием оператора EXECUTE STATEMENT с предложением ON EXTERNAL с неуказанным аргументом db_connect может возникнуть ошибка "Invalid BLOB ID".

Исправил В. Хорсун

[\(CORE-4744\)](#) Выполнение оператора ALTER DATABASE SET DEFAULT CHARACTER SET приводило к двум ошибкам: 1) оператор изменял набор символов по умолчанию только для текущего присоединения; 2) не проводилась проверка существования нового набора символов, пока он не станет использоваться.

Исправил A. dos Santos Fernandes

[\(CORE-4743\)](#) Предоставленная роль не работает, если имя пользователя определено символами, отличным от ASCII.

Исправил А. Пешков

[\(CORE-4740\)](#) Оператор SIMILAR TO с шаблоном {n,} вызывал ошибку "Invalid pattern" в версии Firebird 2.5 и приводил к странным результатам в версии Firebird 3.0.

Исправил A. dos Santos Fernandes

[\(CORE-4733\)](#) Команда ALTER TABLE <T> ALTER TYPE <C>

DOMAIN_WITH_NOT_NULL не проверяла данные в столбце <C> и устанавливала неправильные значения в <C> - ZERO / JULIAN_DATE / ASCII (0) для типов INT, TIMESTAMP и VARCHAR.

Исправил A. dos Santos Fernandes

(CORE-4725) При выполнении команд ALTER DOMAIN и ALTER TABLE были выявлены несоответствия - выполнение команду удаления DROP NOT NULL и DROP PRIMARY KEY.

Исправил A. dos Santos Fernandes

(CORE-4719) При выполнении команды REVOKE ALL ON ALL FROM <existing_user> выдаётся сообщение "Statement failed, SQLSTATE = 00000 + unknown ISC error 0".

Исправил А. Пешков

(CORE-4713) При откате после вставки в таблицу с индексом по выражению выдаётся ошибка "BLOB not found".

Исправил В. Хорсун

(CORE-4712) Сообщения "Error in isc_release_request() ... when working with legacy security database" (ошибка ... при работе с устаревшей базой данных безопасности) появлялись в файле firebird.log для классического сервера при подключении с устаревшей проверкой подлинности пользователя.

Исправил А. Пешков

(CORE-4702) Ухудшение: Порядок соединения (JOIN) был менее оптимальным, чем в v2.x.

Исправил Д. Еманов

(CORE-4684) При подготовке сложного запроса возникает ошибка - "Too

many Contexts of Relation/Procedure/Views. Maximum allowed is 256” (“слишком много контекстов связи/процедуры/представлений. Максимально допустимое значение 256”).

Исправил Д. Еманов

[\(CORE-4675\)](#) Условие WHERE <field> = <cursor>. <Field> не использует существующий индекс.

Исправил Д. Еманов

[\(CORE-4673\)](#) Вычисляемый индекс для вычисляемого столбца хранит значение NULL для всех его ключей.

Исправил Д. Еманов

[\(CORE-4670\)](#) В некоторых случаях могут быть пропущены ошибки нарушения ограничений.

Исправил Д. Еманов

[\(CORE-4668\)](#) Запрос SELECT FROM MON \$ TABLE_STATS не работал в серверах в режиме суперклассик и классик.

Исправил Д. Еманов

[\(CORE-4665\)](#) Критерии поиска WHERE <field_C> STARTING WITH <:value> ORDER BY <field_N> приводят к неправильным результатам в случае, когда field_C является ведущей частью составного индекса, состоящего из {field_C, field_N}.

Исправил Д. Еманов

[\(CORE-4659\)](#) При внедрении поддержки нескольких менеджеров

пользователей, в результате выполнения некоторых операций по настройке защиты каждой базы данных, может появиться ошибка «Отсутствует контекст безопасности для\\SECURITY3.FDB» (“Missing security context for\\SECURITY3.FDB”).

Исправил А. Пешков

[\(CORE-4656\)](#) Сервер может зависнуть при поиске версий удалённых записей.

Исправил Д. Стародубов

[\(CORE-4648\)](#) Пользователю с ролью RDB\$ADMIN было отказано в праве на создание базы данных.

Исправил А. Пешков

[\(CORE-4644\)](#) При большой параллельной нагрузке могут возникать ошибки контекста безопасности и открытия базы данных.

Исправил А. Пешков

[\(CORE-4634\)](#) Ухудшение: При использовании одного и того же индексированного столбца одновременно в предложениях WHERE и ORDER BY может произойти ошибка «Нет текущей записи для операции выборки» (“No current record for fetch operation”).

Исправил Д. Еманов

[\(CORE-4631\)](#) Когда область общей памяти, поддерживающая таблицу блокировки, не может быть переназначена, вектор состояния возвращает сообщение об ошибке «Менеджер блокировки не доступен» (“Lock manager out of room”) без указания каких-либо деталей низкого уровня (например, ошибка уровня ОС), тем самым скрывая реальную причину проблемы.

Исправил Д. Еманов

[\(CORE-4624\)](#) Использование символа ':' в пути для подключения таблицы

приводит к игнорированию всех символов, следующих за ним (в т.ч. и его самого).

Исправил А. Пешков

[\(CORE-4623\)](#) Ухудшение: Использование в PSQL переменных с типом данных "DOMAIN" и "TYPE OF", ссылающихся на BLOB с SUB_TYPE < 0, приводит к неработоспособности кода.

Исправил А. dos Santos Fernandes

[\(CORE-4622\)](#) Триггеры с инструкцией UPDATE OR INSERT и оператором IIF() работают не так, как ожидалось.

Исправил Д. Еманов

[\(CORE-4618\)](#) Оператор ROLLBACK (откат) отказывается отменить изменения, если оператор MERGE несколько раз обновлял одни и те же целевые строки, а оптимизатор использовал PLAN MERGE.

Исправил Д. Еманов

[\(CORE-4604\)](#) Данные с типом VARCHAR имеют разную длину (CHAR_LENGTH()) до и после выполнения оператора EXECUTE STATEMENT.

Исправил А. dos Santos Fernandes

[\(CORE-4599\)](#) Функция REPLACE() не корректно работает с многобайтовыми наборами символов.

Исправил А. dos Santos Fernandes

[\(CORE-4583\)](#) Встроенный сервер пытается загрузить ICU DLL из папок,

прописанных в переменной PATH.

Исправил А. Пешков

[**\(CORE-4582\)**](#) Во время периода "задержки закрытия базы данных" (параметр "LINGER"), возникают проблемы при попытке изменить некоторые свойства базы данных.

Исправил А. Пешков

[**\(CORE-4581\)**](#) Встроенный сервер пытается загрузить пользовательские функции (UDF) из неверной директории.

Исправил В. Хорсун

[**\(CORE-4572\)**](#) При выполнении функций PSQL, когда число фактических аргументов не соответствовало числу формальных аргументов, возвращается неверное сообщение об ошибке.

Исправил Д. Еманов

[**\(CORE-4571\)**](#) Выборка данных из таблицы со столбцами ICU приводит к сбою, когда база данных была создана на ОС Linux с ICU 4.2.1, а затем скопирована в ОС Windows.

Исправил А. Пешков

[**\(CORE-4561\)**](#) При использовании курсора с предложением ORDER BY ID + 0 и оператора FOR UPDATE WITH LOCK происходит вызов ошибки BUGCHECK(183).

Исправил Д. Еманов

[**\(CORE-4539\)**](#) Сервер не принимает правильный план запроса.

Исправил Д. Еманов

[\(CORE-4507\)](#) Не удаётся удалить исходный код процедуры.

Исправил Д. Еманов

[\(CORE-4472\)](#) При вызове функции <F> из внутренней функции, объявленной в другом блоке, выдаётся сообщение «Модифицируемая функция <F> в настоящее время используется» (“Modifying function <F> which is currently in use”).

Исправил Д. Еманов

[\(CORE-4384\)](#) Возникают проблемы при работе с таблицей, которая имеет более 65535 страниц указателей.

Исправил Д. Еманов

[\(CORE-4383\)](#) Сборка "мусора" индекса и полей BLOB не работает при обновлениях.

Исправили Д. Сибиряков и Д. Еманов

[\(CORE-4382\)](#) Пользовательские точки сохранения не освобождались после подтверждения (COMMIT) транзакции.

Исправил Д. Сибиряков

[\(CORE-4369\)](#) При слиянии (MERGE) таблиц с участием нескольких совпадений вызывалась ошибка BUGCHECK(177).

Исправил Д. Еманов

[\(CORE-4280\)](#) Сохраненные функции PSQL могут принимать

повторяющиеся входные аргументы.

Исправил Д. Еманов

([CORE-4161](#)) Невозможно вставить записи в таблицу, содержащую столбец, определённый как GENERATED DEFAULT AS IDENTITY.

Исправил Д. Еманов

([CORE-3373](#)) В столбец VARCHAR(25) можно было поместить строку длиной в 31 символ.

Исправил A. dos Santos Fernandes

([CORE-2932](#)) После выполнения команды ALTER POSITION положение поля будет неправильным.

Исправил A. dos Santos Fernandes

([CORE-2848](#)) При высокой нагрузке могут возникать ошибки "Блокировка преобразования запрещена" ("lock conversion denied") на уровне страницы или "блокировка запрещена" ("lock denied"), при этом взаимоблокировки не были обнаружены.

Исправил В. Хорсун

([CORE-214](#)) Слишком медленно выполняется оператор COUNT (DISTINCT ...).

Исправил Д. Еманов

Сбой сервера 3.0 Бета 2

([**CORE-4766**](#)) При попытке управлять списком пользователей с помощью инструкции EXECUTE STATEMENT от имени пользователя, не являющегося пользователем SYSDBA, но с ролью администратора RDB\$ADMIN, происходит нарушение прав доступа.

Исправил В. Хорсун

([**CORE-4705**](#)) Сбой сервера Superclassic при отключении после выполнения инструкции EXECUTE STATEMENT не от имени текущего пользователя.

Исправил Д. Еманов

([**CORE-4653**](#)) Бесконечные взаимные вызовы хранимых процедур вызывали сбой сервера вместо возврата соответствующей ошибки - "слишком много одновременных выполнений одного и того же запроса" ("Too many concurrent executions of the same request").

Исправил В. Хорсун

([**CORE-4075**](#)) При выдаче исключения в вычисляемом индексе сервер выдаст ошибку или аварийно завершит свою работу.

Исправил Д. Еманов

([**CORE-3632**](#)) Приложение аварийно завершит свою работу после вызова fork() с помощью libfbembed.so.

Исправил А. Пешков

API/Удаленный интерфейс 3.0 Бета 2

([CORE-4795](#)) Библиотека `fbclient` дважды выполняла функцию обратного вызова события.

Исправил А. Пешков

([CORE-4794](#)) Команда `isc_cancel_events()` возвращает ошибку “недопустимый идентификатор события (дескриптор) (код: 335545021)” (“invalid events id (handle) (code: 335545021)”), если события уже не было в очереди.

Исправил А. Пешков

([CORE-4788](#)) Сервер `Superclassic` мог зависать при получении сетевого пакета.

Исправил А. Пешков

([CORE-4785](#)) Плохой пакет `or_execute` приводит к аварийному завершению работы сервера.

Исправил А. Пешков

([CORE-4756](#)) В Firebird 3 механизм использования событий (`events`) был повреждён.

Исправил А. Пешков

([CORE-4646](#)) При использовании метода `LegacyAuth` на сервере и конфигурации по умолчанию на клиенте подключение к базе данных отклоняется.

Исправил А. Пешков

([CORE-4639](#)) Серверы на ОС Linux могут отклонять соединения с ошибкой

транслитерации.

Исправил Д. Еманов

([CORE-4471](#)) Плагин Legacy_Auth не соединится с сервера FB3 к серверу FB2.5, после попытки использовать Win_Sspi.

Исправил А. Пешков

Утилиты 3.0 Бета 2

isql

([CORE-4782](#)) Команда SHOW TABLE в утилите isql потерпит неудачу, если в таблице содержится поле с сортировкой Unicode в DDL.

Исправил А. dos Santos Fernandes

([CORE-4706](#)) Утилита isql неправильно заполняла столбцы BLOB, если псевдоним столбца имел длину более 17 символов.

Исправил А. dos Santos Fernandes

([CORE-4578](#)) Входные файлы не закрывались должным образом утилитой isql.

Исправил А. Пешков

([CORE-4503](#)) Команда SHOW USERS в isql показывает только текущего пользователя.

Исправил А. Пешков

([CORE-4259](#)) Вызов setlocale(LC_CTYPE, "") не должен выполняться в isql

для Windows, т. к. при этом нарушается обычная работа команды `chcp`.

Исправил F. Schlottmann-Goedde и Д. Еманов

gbak

([**CORE-4715**](#)) Восстановление теневой базы данных завершается неудачей при использовании ключа `-k ("restore without shadow")`.

Исправил А. Пешков

([**CORE-4704**](#)) Разрешения для генераторов и исключений повреждались после операции резервного копирования/восстановления базы данных.

Исправил Д. Еманов

gsec

([**CORE-4698**](#)) Выполнение команды `add -user SYSDBA -pw masterkey -admin yes` в утилите `gsec` приводит к сбою сервера.

Исправили В. Хорсун и Д. Еманов

Улучшения в 3.0 Бета 2

([**CORE-4791**](#)) Для устранения неоднозначности с булевыми выражениями предложения `INSERTING`, `UPDATING` и `DELETING` стали зарезервированными словами.

Исправил A. dos Santos Fernandes

([**CORE-4735**](#)) Выражение `'where bool_field IS true | false'` теперь может использовать один и тот же индекс (в запросах с условием `'where bool_field = true | false'`), если такой индекс существует.

Исправил A. dos Santos Fernandes

([CORE-4731](#)) В Firebird 3 запрещено использование операторов DML и DDL в системных таблицах (начинающихся с префикса RDB\$).

Исправил А. Пешков

([CORE-4729](#)) В таблицу MON\$DATABASE был добавлен флаг для определения какой тип базы данных безопасности используется - по умолчанию, собственная или другая.

Исправил A. dos Santos Fernandes

([CORE-4696](#)) Теперь нет необходимости в очистке "грязных" страниц на диске после создания индекса для временной таблицы.

Исправил В. Хорсун

([CORE-4685](#)) Устранены некоторые опасные проблемы в POSIX ОС при использовании псевдонимов (алиасов) из файла `databases.conf`, указывающих на символические или жесткие ссылки, разрешающие недопустимый доступ к одной и той же базе данных.

Исправил А. Пешков

([CORE-4671](#)) Автоматически создаваемые сервером временные большие двоичные объекты удалялись до подтверждения или отката транзакции.

Исправил В. Хорсун

([CORE-4610](#)) Диагностика сообщит теперь имя дескриптора при возникновении ошибок транслитерации в значениях блока параметров.

Исправил А. Пешков

([**CORE-4607**](#)) Добавлена поддержка нескольких плагинов в параметре `UserManager` (настраивается в файле `firebird.conf`) для управления аутентификацией пользователей и возможность их использования в SQL.

Исправил А. Пешков

([**CORE-4605**](#)) Параметр конфигурации `CryptPlugin` был переименован в `WireCryptPlugin` (для понимания, что это сетевое шифрование).

Исправил А. Пешков

([**CORE-4590**](#)) Тип данных результата, возвращаемого функциями `CHAR_LENGTH()`, `BIT_LENGTH()` и `OCTET_LENGTH()` для `BLOB` изменён на `BIGINT`.

Исправил А. dos Santos Fernandes

([**CORE-4386**](#)) Более детальный отчёт об ошибке “Объект используется” (“Object in use”).

Исправил Д. Еманов

([**CORE-3526**](#)) Добавлена поддержка перехвата ошибок `WHEN SQLSTATE` в `PSQL`.

Исправил Д. Еманов

([**CORE-3234**](#)) Добавлена поддержка текстовых `BLOB` $\geq 32K$ в качестве первого аргумента для функции `TRIM()`.

Исправил А. dos Santos Fernandes

([**CORE-3226**](#)) Добавлена поддержка протокола IPv6 для удалённого подключения.

Исправил М. Kubecsek

([**CORE-733**](#)) Добавлена поддержка сжатия данных при передаче по сети.

Исправил А. Пешков

Firebird 3.0 первая бета версия

Следующие улучшения и исправления ошибок были сделаны в первой бете версии Firebird 3.0.0 (в дальнейшем 3.0 Бета 1):

Ядро сервера 3.0 Бета 1

([**CORE-4576**](#)) Не удавалось запустить поток записи кэша данных подключения пользователя. После этого не удавалось подключиться к базе данных безопасности.

Исправил В. Хорсун

([**CORE-4574**](#)) Ухудшение: Выдавался неправильный результат в подзапросе с использованием агрегирования.

Исправил А. dos Santos Fernandes

([**CORE-4570**](#)) Команда ALTER PACKAGE возвращает неверную ошибку.

Исправил А. dos Santos Fernandes

([**CORE-4566**](#)) При использовании системного поля (например, TYPE OF COLUMN RDB\$TYPES.RDB\$FIELD_NAME) в наборе символов метаданных выполнение оператора EXECUTE BLOCK приводило к ошибке - неправильный размер выходного параметра или аргумента.

Исправил А. Пешков

([CORE-4565](#)) Номер ошибки GDSCODE может иметь значение, равное нулю в предложении WHEN при конкурентных подключениях в случае возникновения взаимной блокировки (lock conflict). Эта ошибка проявлялась для серверов в режиме классик и суперклассик.

Исправил В. Хорсун

([CORE-4555](#)) Триггер оставался активным (продолжал работать) после удаления.

Исправил А. dos Santos Fernandes

([CORE-4530](#)) Соединение в запросе двух таблиц на основе DB_KEY может оказаться неэффективным.

Исправил Д. Еманов

([CORE-4522](#)) Привилегии DDL не защищали от удаления BLOB фильтров непривилегированным пользователям.

Исправил Р. Симаков

([CORE-4515](#)) Ухудшение: Трассировка показывала выполнение операции UPDATE при выполнении вставки (INSERT) в <some_table>.

Исправил Д. Еманов

([CORE-4505](#)) Попытка использования именованного курсора (cursor_1) другим именованным курсором завершится ошибкой, если предварительно не открыть cursor_1.

Исправил А. Пешков

([CORE-4488](#)) Оператор FOR SELECT <L> FROM <T> AS CURSOR <C> возвращает неправильный результат, если таблица <T> была изменена внутри блока BEGIN ... END курсора. Ссылки курсора, которые не являются переменными, должны представлять текущее состояние записи. Ссылки на курсоры должны возвращать новые значения, если курсор был с использованием предложения «where current of». Первый пример, описанный в [CORE-4488](#), должен возвращать значение NULL.

Исправил A. dos Santos Fernandes

([CORE-4478](#)) Не сообщалась информация о сбое загрузки поставщика или плагина.

Исправил A. Пешков

([CORE-4477](#)) Поле RDB\$MAP_TO_TYPE отсутствовало в системной таблице RDB\$TYPES.

Исправил A. Пешков

([CORE-4468](#)) Не работал оператор CREATE USER GRANT ADMIN ROLE.

Исправил A. Пешков

([CORE-4464](#)) Дублирующиеся дескрипторы в операторе CREATE/ALTER USER обрабатывались неправильно.

Исправил A. Пешков

([CORE-4453](#)) Ухудшение: не работает ограничение NOT NULL, если оно объявлено в домене.

Исправил A. dos Santos Fernandes

([CORE-4447](#)) Позиционированный оператор UPDATE препятствует использованию индекса при последующих ссылок на обновлённое поле с использованием курсора.

Исправил Д. Еманов

([CORE-4444](#)) Сервер мог зависать и блокировать все подключения в случае нехватки места на диске во время физического резервного копирования.

Исправил В. Хорсун

([CORE-4433](#)) GlobalRWLock не мог понизить уровень блокировки с EX до SH, если происходило чтение блокированной записи.

Исправил В. Хорсун

([CORE-4435](#)) При вызове метода `release()` вместо `detach()` для подключения к базе данных во встроенном сервере приложение остаётся бесконечно активным.

Исправил А. Пешков

([CORE-4430](#)) Свойства пользователя, созданные в плагине Legacy_UserManager, заполнялись пробелами длиной до 10 символов.

Исправил А. Пешков

([CORE-4415](#)) Команда SHOW для пустого (create trigger tr before any ddl statement as begin end) триггера бессмысленно извлекает его полный DDL (TR, Sequence: 0, Type: BEFORE CREATE TABLE OR ALTER TABLE OR DROP TABLE OR CREATE PROCEDURE OR и т.д.).

Исправил A. dos Santos Fernandes

([CORE-4396](#)) Запрос, выполняемый с помощью оператора EXECUTE STATEMENT, возвращал неверный результат.

Исправил A. dos Santos Fernandes

([CORE-4395](#)) Оператор EXECUTE STATEMENT ON EXTERNAL не находил базу данных Firebird 2.5.

Исправил А. Пешков

([CORE-4394](#)) При использовании устаревшего API выдаётся ошибка: "Курсор не найден, ошибка" ("Cursor not found error").

Исправил А. Пешков

([CORE-4388](#)) Выполнение оператора SELECT WITH LOCK может привести к бесконечному циклу для одной записи.

Исправил Д. Еманов

([CORE-4381](#)) Ошибки времени выполнения (Run-time errors) возвращали неверную информацию о строке/столбце.

Исправил A. dos Santos Fernandes

([CORE-4379](#)) Явные курсоры, содержащие коррелированные вложенные запросы в списке выбора, работали медленно.

Исправил Д. Еманов

([CORE-4376](#)) Подготовка ошибочной инструкции DDL не вызывала ошибку и не показывала её.

Исправил A. dos Santos Fernandes

([CORE-4375](#)) Процедура будет выполняться бесконечно, если она содержит более 32767 операторов внутри любого блока BEGIN/END.

Исправил Д. Еманов

([CORE-4374](#)) При использовании EXECUTE STATEMENT с BLOB возникает ошибка: превышен лимит длины.

Исправил A. dos Santos Fernandes

([CORE-4373](#)) При компиляции пакетов не было проверки наличия дубликатов имен.

Исправил A. dos Santos Fernandes

([CORE-4372](#)) Может произойти взаимная блокировка (deadlock), когда две страницы данных содержат фрагменты записей, указывающих друг на друга.

Исправил В. Хорсун

([CORE-4371](#)) Операторы CREATE FUNCTION и CREATE PROCEDURE, ссылающиеся на несуществующее исключение будет возвращает сообщение об ошибке “ошибка при синтаксическом анализе BLR” (“Error while parsing function's BLR”), а не “исключение не определено” (“exception not defined”).

Исправил Д. Еманов

([CORE-4366](#)) Предикат WHERE, содержащий NULL IS NOT DISTINCT FROM (select min(NULL) from ...), возвращал неверный результат.

Исправил Д. Еманов

([CORE-4365](#)) Распределение предикатов равенства не работает для некоторых сложных запросов (строился не оптимальный план запроса).

Исправил Д. Еманов

([CORE-4360](#)) Оператор SELECT из производной таблицы, содержащей предложение GROUP BY по столбцу со строковым значением, возвращает неверные результаты.

Исправил Д. Еманов

([CORE-4354](#)) Синтаксический анализ рекурсивного запроса возвращает ошибку "столбец не принадлежит ссылочной таблице" ("Column does not belong to referenced table"), не смотря на то, что в исходной таблице такой столбец существует.

Исправил A. dos Santos Fernandes

([CORE-4353](#)) Сортировка записей производилась значительно дольше, чем это действительно было необходимо.

Исправил Д. Еманов

([CORE-4344](#)) Ошибка "нет текущей записи для операции выборки" ("no current record for fetch operation") выдавалась при явном внутреннем соединении таблицы, процедуры с другой таблицей.

Исправил Д. Еманов

([CORE-4334](#)) Ресурсы (например, файлы сортировки), принадлежащие триггеру, могут остаться не освобождёнными, если работа триггера была прервана асинхронно.

Исправил Д. Еманов

([CORE-4331](#)) Функции LAG, LEAD и NTH_VALUE вызывают ошибку, если второй аргумент имеет значение NULL.

Исправил A. dos Santos Fernandes

([CORE-4330](#)) Функция LAG возвращает неверный результат, если значение смещения было назначено из таблицы.

Исправил A. dos Santos Fernandes

([CORE-4326](#)) Требовалось использование ключевого слова SET в инструкции ALTER USER, хотя оно является необязательным.

Исправил А. Пешков

([CORE-4318](#)) Ухудшение: Предикаты, включающие переменные PSQL или параметры, не использовались в агрегатах, что приводило к большому использованию памяти.

Исправил Д. Еманов

([CORE-4313](#)) Ошибка "попытка повторного открытия открытого курсора" ("Attempt to reopen an open cursor") может возникнуть после попытки открытия его после закрытия.

Исправил Д. Еманов

([CORE-4309](#)) Запись «Cache Writer» в таблице MON\$ATTACHMENTS исчезает при удалении записи в MON\$ATTACHMENTS из другого соединения, которое выполняет обновление в большой таблице.

Исправил В. Хорсун

([CORE-4307](#)) Любые поля, присутствующие только в предложении WHERE в представлении с опцией WITH CHECK OPTION, вызывают недопустимое нарушение ограничений CHECK CONSTRAINT.

Исправил А. dos Santos Fernandes

([CORE-4303](#)) Служба продолжала работать после её принудительной (release) остановки.

Исправил А. Пешков

([CORE-4301](#)) Если данные в таблице SEC\$USERS содержали не ASCII символы, то они читались некорректно.

Исправил А. dos Santos Fernandes

([CORE-4286](#)) При попытке выполнить другой оператор SQL с использованием другого имени курсора возникает ошибка “Оператор уже имеет назначенный курсор” (“Statement already has a cursor assigned”).

Исправил А. Пешков

([CORE-4118](#)) Индекс по выражению может не учитываться для производных полей или полей представления.

Исправил Д. Еманов

([CORE-3305](#)) После создания или изменения недопустимого триггера возвращается ошибка "BLOB не найден".

Исправил Д. Еманов

([CORE-2350](#)) Слишком длинное имя столбца для псевдонима в операторе SELECT не отклонялось.

Исправил A. dos Santos Fernandes

([CORE-1475](#)) База данных с активными подключениями не может быть заменена при восстановлении из файла резервной копии даже после завершения (shutdown) работы базы данных.

Исправил Д. Еманов

Сбои сервера 3.0 Бета 1

([CORE-4575](#)) При отключении последнего подключения во время сборки мусора сервер аварийно завершает работу.

Исправил В. Хорсун

([CORE-4568](#)) Сервер может аварийно завершить работу при отключении от базы данных, находящейся под нагрузкой.

Исправил Д. Еманов

([CORE-4510](#)) Ошибка проверки базы данных может привести к сбою сервера.

Исправил В. Хорсун

([CORE-4506](#)) Сервер аварийно завершает работу при выполнении практически любой команды "show ..." после переподключения.

Исправил А. Пешков

([CORE-4500](#)) Firebird аварийно завершает работу после не успешного перераспределения общей памяти таблицы блокировки.

Исправил Д. Еманов

([CORE-4498](#)) Сервер аварийно завершает работу при получении явного плана для поиска на основе DBKEY.

Исправил Д. Еманов

([CORE-4422](#)) Сервер аварийно завершает работу при использовании функции ROW_NUMBER () OVER (PARTITION BY x) в предложении ORDER BY.

Исправил Д. Еманов

([CORE-4419](#)) Сервер может аварийно завершить работу при сортировке записей размером более 128 кб.

Исправил Д. Еманов

([CORE-4322](#)) Сервер аварийно завершает работу при использовании агрегатных или оконных функций в рекурсивном запросе.

Исправил А. dos Santos Fernandes

([CORE-4321](#)) Ухудшение: Утилита командной строки isql не уничтожала инструкцию SQL.

Исправил А. Пешков

([**CORE-4319**](#)) Сервер аварийно завершает работу, когда в конфигурации трассировки содержится строка “connection_id=NN”, и будет предпринята попытка подключения к несуществующей базе данных/псевдониму.

Исправил В. Хорсун

([**CORE-4304**](#)) Сервер аварийно завершает работу, если попытка пересоздания (RECREATE) таблицы с внешним ключом была сделана после синтаксической ошибки, предшествующей попытке выполнения оператора RECREATE.

Исправил A. dos Santos Fernandes

API/Удаленный интерфейс 3.0 Бета 1

([**CORE-4275**](#)) Команда CREATE DATABASE завершается ошибкой ("Could not find acceptable ICU library"), если библиотека fbclient.dll была загружена из другого каталога (Providers = Engine12).

Исправил В. Хорсун

Утилиты 3.0 Бета 1

isql

([**CORE-4480**](#)) isql выдаёт предупреждение “Неверный формат отладочной информации” (“Bad debug info format”) при подключении к базе данных с сохранёнными функциями после восстановления.

Исправил Д. Еманов

([**CORE-4440**](#)) isql аварийно завершает работу при выполнении команды SHOW VERSION без активного подключения к базе данных.

Исправил А. Пешков

([CORE-4380](#)) isql усекает BLOB при чтении пустого сегмента.

Исправил A. dos Santos Fernandes

([CORE-4320](#)) Ухудшение: Выполнение SQL запроса вместе со статистикой (SET STAT ON) в isql приводит к аварийному завершению работы сервера.

Исправил В. Хорсун

gbak

([CORE-4470](#)) Восстановление в gbak базы данных, содержащей зависимости между представлениями и упакованными функциями, завершается ошибкой.

Исправил A. dos Santos Fernandes

([CORE-4425](#)) Пользовательские параметры сортировки, основанные на UNICODE, не обновлялись до новой версии ICU при восстановлении базы данных.

Исправил A. dos Santos Fernandes

([CORE-4417](#)) gbak не может подтвердить создание индекса для первичного ключа с акцентированными символами умляют.

Исправил A. dos Santos Fernandes

([CORE-4346](#)) gbak Firebird 3 не мог восстановить резервные копии, созданные в более ранних версиях серверов.

Исправил Д. Еманов

nBackup

([CORE-4461](#)) nBackup выводил сообщения об ошибках в `stdout` вместо `stderr`.

Исправил А. Пешков

qli

([CORE-4327](#)) qli выдаёт ошибку при копировании NULL BLOB между базами данных.

Исправил А. Пешков

Firebird 3.0 вторая альфа версия: Исправление ошибок

Следующие улучшения и исправления ошибок были сделаны во второй альфе версии Firebird 3.0.0 (в дальнейшем 3.0 Альфа 2):

Ядро сервера 3.0 Альфа 2

([CORE-4302](#)) Для некоторых ключей убывающий индекс может быть очень неэффективным.

Исправил В. Хорсун

([CORE-4289](#)) Поле NOT NULL производной таблицы могло иметь значение NULL при ссылке на него извне.

Исправил Д. Еманов

([CORE-4281](#)) Параметры TYPE OF сохраненных функций могли приводить к зависанию сервера при изменении зависимого домена или столбца.

Исправил А. dos Santos Fernandes

([CORE-4270](#)) Подзапрос с использованием оконной функции и условия условия <field> IN (select ...) может привести к ошибке.

Исправил Д. Еманов

([CORE-4265](#)) При подключении к сильно загруженной базе данных может возникнуть непредвиденная ошибка блокировки.

Исправил Д. Еманов

([CORE-4262](#)) Могли происходить ошибки парсинга контекста при работе с производными таблицами и оператором CASE.

Исправил Д. Еманов

([CORE-4261](#)) Результат соединения (JOIN) может быть неправильным, если объединенные поля были созданы с помощью функции row_number().

Исправил Д. Еманов

([CORE-4258](#)) Граница минимального значения для типов BIGINT/DECIMAL(18) была неправильной.

Исправил A. dos Santos Fernandes

([CORE-4251](#)) Служба Guardian может записать мусор в конце сообщения в журнале событий (Event Log).

Исправил В. Хорсун

([CORE-4250](#)) В службе Guardian могла происходить ошибка доступа при завершении работы с базой данных.

Исправил В. Хорсун

([CORE-4237](#)) Метаданные, возвращаемые из запросов к системным таблицам для возвращаемых аргументов UDF, отличаются от возвращаемых в Firebird 2.5.

Исправил А. dos Santos Fernandes

([CORE-4234](#)) Оператор IF (subfunc()) выполняется с ошибкой, когда subfunc() возвращает логическое значение.

Исправил А. dos Santos Fernandes

([CORE-4229](#)) Двухнаправленный курсор не позиционировался при первом вызове FETCH LAST.

Исправил Д. Еманов

([CORE-4227](#)) Конфликт синтаксического анализатора вызывал неправильную оценку в операторе BETWEEN и в логических выражениях.

Исправил А. dos Santos Fernandes

([CORE-4216](#)) Утечка памяти в триггере TRIGGER ON TRANSACTION COMMIT.

Исправил В. Хорсун

([CORE-4211](#)) Встроенный сервер зависал на 5 секунд при закрытии, при этом ошибки о тайм-ауте в процессе завершения работы и недопустимый мьютекс записывались в firebird.log.

Исправил А. Пешков

([CORE-4201](#)) Вычисляемое поле возвращает значение NULL внутри триггера "перед вставкой" (BEFORE INSERT).

Исправил Д. Еманов

([CORE-4198](#)) Возникает неправильная ошибка "символ неизвестен", если строка SQL завершился шестнадцатеричным числом.

Исправил A. dos Santos Fernandes

([CORE-4177](#)) Некоторые логические выражения не разрешались.

Исправил A. dos Santos Fernandes

([CORE-4159](#)) Выводилась некорректная статистика памяти.

Исправил Д. Еманов

([CORE-4156](#)) Параметры функций RDB\$GET_CONTEXT/RDB\$SET_CONTEXT были неправильно описаны как CHAR NOT NULL, а не как VARCHAR NULLABLE.

Исправил A. dos Santos Fernandes

([CORE-3989](#)) Плохая производительность и медленный отклик при выполнении множества параллельных сортировок.

Исправил Д. Еманов

([CORE-3921](#)) При значении параметра bugcheckabort=1 и старте сборки мусора, начиная с зазора приблизительно 21000, выдаётся ошибка "Bugcheck 186 (record disappeared — запись исчезла)" и происходит 100% загрузка процессора.

Исправил В. Хорсун

([CORE-2165](#)) При использовании условия строгого неравенства могут возникать ненужные чтения индекса.

Исправил В. Хорсун

Сбои сервера 3.0 Альфа 2

([CORE-4293](#)) Сервер может аварийно завершить работу оператора SELECT с длинным или сложным списком составных предикатов OR/AND.

Исправил Д. Еманов

([CORE-4271](#)) Пересоздание ошибочного тела пакета могло привести к падению сервера.

Исправил A. dos Santos Fernandes

([CORE-4268](#)) Отключение от базы данных может привести к падению сервера.

Исправил Д. Еманов

([CORE-4267](#)) Сборка мусора в базе данных может привести к падению сервера.

Исправил Д. Еманов

([CORE-4225](#)) Сервер может аварийно завершить работу при попытке трассировки базы данных, имеющей триггеры уровня базы данных.

Исправил В. Хорсун

([CORE-4185](#)) Возникали сбои сервера с сообщением “invalid lock id (NNNNN)” - “неверный идентификатор блокировки (NNNNN)”.

Исправил А. Пешков

API/Удаленный интерфейс Альфа 2

([CORE-4283](#)) При одновременной регистрации событий могут возникать ошибки “Resource temporarily unavailable” (“ресурс временно недоступен”).

Исправил А. Пешков

([CORE-4236](#)) Сообщение об успешном завершении работы с базой данных выдавалось до окончания работы всех активных подключений.

Исправил Д. Еманов

([CORE-4178](#)) Новый API всё ещё возвращал неявные артефакты исторических определений полей данных вместо правильных свойств метаданных, которые сделали бы интерфейс фактически пригодным для использования.

Исправил А. Пешков

([CORE-4162](#)) При вызове `attachDatabase()` не выдавались предупреждения.

Исправил А. Пешков

Безопасность/Управление пользователями Альфа 2

([CORE-4241](#)) Можно успешно осуществить подключение к базе данных с пустым паролем.

Исправил А. Пешков

([CORE-4200](#)) Не подтверждённая (с помощью оператора COMMIT) выборка данных предложения из псевдо таблицы SEC\$USERS блокирует новые подключения к базе данных.

Исправил А. Пешков

Процедурный язык Альфа 2

([CORE-4247](#)) Позиционированный оператор DELETE (WHERE CURRENT OF <CURSOR>) может не работать для таблиц с новыми полями.

Исправил А. dos Santos Fernandes

([CORE-4244](#)) Создание процедуры может быть проблемным, если она включает добавление текста с набором символов DOS864.

Исправил А. dos Santos Fernandes

([CORE-4184](#)) Выполнение пустого EXECUTE BLOCK с выходными параметрами NOT NULL приводит к ошибке.

Исправил А. dos Santos Fernandes

([CORE-4160](#)) Параметризованное исключение игнорирует символы, переданные в качестве параметра, если они отличны от ASCII.

Исправил А. dos Santos Fernandes

([CORE-4145](#)) Подготовка к выполнению (prepare) оператора EXECUTE BLOCK, в котором используются домены, вызывала утечку памяти.

Исправил А. dos Santos Fernandes

Язык определения данных Альфа 2

([**CORE-4214**](#)) Глобальные временные таблицы могли ссылаться на постоянные отношения, хотя они не должны уметь это делать.

Исправил В. Хорсун

([**CORE-4212**](#)) Удаление внешнего ключа в глобальной временной таблице приводит к падению сервера.

Исправил В. Хорсун

([**CORE-4203**](#)) Не удавалось создать упакованные подпрограммы с параметрами CHAR или VARCHAR.

Исправил A. dos Santos Fernandes

([**CORE-4180**](#)) Оператор CREATE COLLATION не проверял набор базовых параметров сортировки для набора символов по умолчанию.

Исправил A. dos Santos Fernandes

([**CORE-4173**](#)) Повторная установка значения генератора в одной транзакции устанавливает его равным нулю.

Исправил Д. Еманов

([**CORE-4155**](#)) Внешние процедуры DDL в пакетах неверно сообщали о завершении с помощью двоеточия в качестве ошибки.

Исправил A. dos Santos Fernandes

Язык управления данными Альфа 2

([**CORE-4269**](#)) Выводились неправильные данные при использовании поля с результатом оконной функции в запросе с бесполезным предложением WHERE 0=0 и GROUP BY.

Исправил Д. Еманов

([**CORE-4255**](#)) Не работал параметризованный запрос с использованием RDB\$DB_KEY.

Исправил Д. Еманов

([**CORE-4240**](#)) Рекурсивный запрос возвращает неверные результаты, если требуется прохождение более, чем через одну ветвь.

Исправил Д. Еманов

([**CORE-4158**](#)) При использовании экранирующих кавычек оператор LIKE не работает.

Исправил A. dos Santos Fernandes

Утилиты Альфа 2

gfix

([**CORE-4297**](#)) Утилита gfix завершается аварийно, если размер описания транзакции limbo будет больше 1 КБ.

Исправил В. Хорсун

fbsvcmgr

([**CORE-4298**](#)) Утилита fbsvcmgr не распознала sts_record_versions и другие ключи sts.

Исправил А. Пешков

isql

([CORE-4259](#)) Ошибка при выполнении команды `isql setlocale(LC_CTYPE, "")` в ОС Windows из-за ссылки на `editline`, которая недоступна на этой платформе.

Исправил F. Schlottmann-Goedde

([CORE-4205](#)) ISQL -х не мог выводить предложение `START WITH` для генераторов / последовательностей.

Исправил A. dos Santos Fernandes

([CORE-4149](#)) Новые типы прав (разрешений) не отображаются `isql`.

Исправил Д. Еманов

([CORE-362](#)) Определённые символы невозможно было ввести в `isql`.

Исправил F. Schlottmann-Goedde

gbak

([CORE-4202](#)) Не производится резервное копирование/восстановление с более старой версии до версии 3.0 - выдаётся ошибка BLR.

Исправил Д. Еманов

([CORE-4168](#)) Резервная копия, содержащая процедуры или триггеры, выбранные из внешних таблиц, не может быть восстановлена, если `ExternalFileAccess = None`.

Исправил Д. Еманов

([**CORE-4164**](#)) Для генераторов и исключений, восстановленных из резервной копии, отсутствовало имя владельца.

Исправил Д. Еманов

nbackup

([**CORE-2648**](#)) Дельта-файл nBackup игнорировал параметр Forced Writes базы данных.

Исправил В. Хорсун

Мониторинг базы данных (MON\$)

([**CORE-4235**](#)) Могла произойти блокировка (*deadlock*) при доступе к таблицам мониторинга при параллельной загрузке.

Исправили Д. Еманов и В. Хорсун

([**CORE-4176**](#)) Таблицы мониторинга возвращали неполную информацию, если сервер работал в режиме Classic или SuperClassic.

Исправил Д. Еманов

Трассировка

([**CORE-4219**](#)) Регулярные выражения с двойной косой чертой не выполняются при трассировке.

Исправил А. Пешков

([**CORE-4163**](#)) Конфигурационный файл fbtrace.conf содержал синтаксические ошибки.

Исправил А. Пешков

Проблемы с установкой Альфа 2

([**CORE-4153**](#)) Попытка использовать Legacy_Auth непосредственно после установки не будет работать без перезапуска службы.

Исправил P. Reeves

Firebird 3.0 первая альфа версия

Следующие улучшения и исправления ошибок были сделаны в первой альфе версии Firebird 3.0.0 (в дальнейшем 3.0 Альфа 1):

Ядро сервера Альфа 1

([**CORE-4135**](#)) Сборка мусора блокировала создание параллельных подключений в Суперсервере.

Исправил В. Хорсун

([**CORE-4134**](#)) При запуске автоматической сборки мусора могло возникнуть состояние состязания для подключений к базе данных.

Исправил В. Хорсун

([**CORE-4074**](#)) Извлечение метаданных из таблицы, содержащей столбец COMPUTED BY с функцией POSITION, могло выводить искаженный результат.

Исправил A. dos Santos Fernandes

([**CORE-4027**](#)) Создание таблицы с вычисляемыми полями, содержащими оператор SELECT FIRST, может привести к искажённому результату.

Исправил A. dos Santos Fernandes

([**CORE-3973**](#)) В сгруппированном запросе для столбца с псевдонимами в SQLDA отсутствовало исходное имя таблицы, имя столбца и владелец.

Исправил A. dos Santos Fernandes

([**CORE-3947**](#)) Неправильные результаты запроса, когда столбец в предложении WHERE использовал опцию сопоставления (NUMERIC-SORT=1).

Исправил A. dos Santos Fernandes

([**CORE-3941**](#)) Уникальный индекс по выражению имеет проблему выравнивания памяти.

Исправил A. dos Santos Fernandes

([**CORE-3929**](#)) Недопустимая ошибка “попытка обновления столбца только для чтения” появляется при выборе MINVALUE из списка более 255 элементов.

Исправил A. dos Santos Fernandes

([**CORE-3894**](#)) Номера, выводимые в сообщении об ошибке при попытке уменьшить размер столбца CHAR или VARCHAR были неправильными.

Исправил A. dos Santos Fernandes

([**CORE-3874**](#)) Значение вычисляемого столбца отображается в несуществующих строках, выводимых из левого соединения (LEFT JOIN) таблиц.

Исправил A. dos Santos Fernandes

([**CORE-3820**](#)) Некоторые наборы символов дублируются в системной таблице типов RDB\$TYPES.

Исправил A. dos Santos Fernandes

([**CORE-3754**](#)) Оператор SIMILAR TO работает некорректно.

Исправил A. dos Santos Fernandes

([**CORE-3735**](#)) Непривилегированный пользователь может удалять данные из системных таблиц RDB\$DATABASE, RDB\$COLLATIONS и RDB\$CHARACTER_SETS.

Исправил Д. Еманов

([**CORE-3694**](#)) В запросе с группировкой по подзапросу + хранимая процедура + агрегат происходит "внутренняя проверка согласованности" ("Internal consistency check").

Исправил A. dos Santos Fernandes

([**CORE-3672**](#)) При создании вычисляемого индекса для больших столбцов невозможно было использовать функцию SUBSTRING.

Исправил A. dos Santos Fernandes

([**CORE-3638**](#)) Введена некоторая корректировка для сортировок: сопоставление FR_CA_CI_AI; FR_FR был изменен так, чтобы он был идентичен FR_CA, а FR_FR_CI_AI был изменен так, чтобы он был идентичен новому FR_CA_CI_AI.

Исправил A. dos Santos Fernandes

([CORE-3476](#)) В функцию LIST добавлена поддержка конкатенации двоичных блобов, как будто они являются текстом.

Исправил A. dos Santos Fernandes

([CORE-3401](#)) При использовании **[type of] <domain>** и **type of <column>** могли возникать ошибки сортировки.

Исправил A. dos Santos Fernandes

([CORE-3338](#)) Регрессия: изменения в коде отключили поддержку индексов выражений, использующих функции COALESCE, CASE и DECODE.

Исправил A. dos Santos Fernandes

([CORE-3317](#)) Успех удаления строк зависел от порядка их вставки.

Исправил В. Хорсун

([CORE-3310](#)) Выборка данных в DSQL из представления, использующего сложное выражение, включающее в себя функцию RDB\$GET_CONTEXT и BETWEEN, вызывала ошибку преобразования.

Исправил A. dos Santos Fernandes

([CORE-3291](#)) Новый псевдостолбец (RDB\$RECORD_VERSION) для записи номера транзакции, которая создала новую версию записи.

Исправил A. dos Santos Fernandes

([CORE-3260](#)) Interlock.h не был переносимым.

Исправил А. Пешков

([CORE-3250](#)) Сервер Firebird не может запускаться под любым именем пользователя, кроме «root», «firebird», «interbas» или «interbase».

Исправил А. Пешков

([CORE-3239](#)) Порядок сортировки UTF8 UNICODE_CI не мог использоваться в составном индексе.

Исправил А. dos Santos Fernandes

([CORE-3204](#)) Ошибка нарушения ограничений, связанная с использованием функции CAST приведением, не возникала внутри представлений.

Исправил А. dos Santos Fernandes

([CORE-3052](#)) Сравнения с участием нескольких сегментов индекса могут привести к неправильным наборам результатов.

Исправил А. dos Santos Fernandes

([CORE-2988](#)) Не сообщался номер параллельной транзакции при возникновении тайм-аута блокировки.

Исправил Н. Самофатов

([CORE-2957](#)) COUNT (*) из большой таблицы мог вернуть отрицательный результат.

Исправил Д. Еманов

([CORE-2952](#)) Имена классов классов в выражениях SIMILAR TO могут быть чувствительными к регистру или нечувствительными к регистру, в зависимости от параметров сортировки левой части, тогда как они должны быть недвусмысленно нечувствительными к регистру.

Исправил Д. Сибиряков

([CORE-2932](#)) Операция ALTER TABLE..ALTER COLUMN..ALTER POSITION может привести к неправильной позиции столбца.

Исправил A. dos Santos Fernandes

([CORE-2922](#)) Набор символов, используемый в константе, не регистрируется как зависимость.

Исправил A. dos Santos Fernandes

([CORE-2913](#)) Выражения COLLATE применялись неправильно.

Исправил A. dos Santos Fernandes

([CORE-2798](#)) При выборке данных из представлений, содержащих вызовы процедур, в выходных данных плана отсутствовали имена представлений.

Исправил Д. Еманов

([CORE-2796](#)) DB_KEY всегда был нулем для строк во внешних таблицах.

Исправил Д. Еманов

([CORE-2678](#)) Полное внешнее соединение (full outer join) не могло использовать доступные индексы, что иногда приводило к очень медленному выполнению запросов.

Исправил Д. Еманов

([CORE-2508](#)) Использование некоторых символов в двойных кавычках в именах индексов, например, скобочного символа, может привести к неудачной логике синтаксического анализа при создании удобочитаемого плана.

Исправил Д. Еманов

([**CORE-2155**](#)) Объединение хранимой процедуры с представлением или таблицей могло завершиться ошибкой "нет текущей записи для операции выборки" ("No current record for fetch operation").

Исправил Д. Еманов

([**CORE-1712**](#)) Ошибочно выводилась ошибка переполнения буфера (buffer overrun error) при преобразовании типа DOUBLE PRECISION в тип VARCHAR в базе данных с 1-м диалектом.

Исправил С. Valderrama С.

([**CORE-1605**](#)) Агрегированный запрос вызывал ошибку "Bugcheck 232 (invalid operation недопустимая операция)".

Исправил А. dos Santos Fernandes

([**CORE-1550**](#)) Выполнялось ненужное сканирование индекса, когда один и тот же индекс используется в предложениях WHERE и ORDER BY.

Исправил Д. Еманов

API/Удаленный интерфейс Альфа 1

([**CORE-3718**](#)) Клиентская библиотека может зависнуть после неудачной попытки подключения к удаленному вспомогательному (для событий) порту.

Исправил А. Пешков

([**CORE-3475**](#)) Параметры внутри функции CAST были ошибочно описаны в SQLDA как не допускающие значения NULL.

Исправил A. dos Santos Fernandes

([**CORE-3269**](#)) Клиент не может корректно выполнять отсоединение от базы данных, если сервер стал недоступен.

Исправил A. Пешков

([**CORE-2484**](#)) Только для ОС Windows: в вектор состояния ошибки будет возвращено ошибочное сообщение об успешном выполнении команды при неудачной попытке подключения к файлу базы данных.

Исправил C. Valderrama C.

([**CORE-2431**](#)) Строковые значения в сообщениях об ошибках не преобразовывались в набор символов соединения.

Исправил A. dos Santos Fernandes

Процедурный язык Альфа 1

([**CORE-4018**](#)) Использование системного домена в объявлениях аргументов или возвращаемых значений в хранимой процедуре может помешать модификации процедуры.

Исправил A. dos Santos Fernandes

([**CORE-3737**](#)) Определения параметров блока EXECUTE не соблюдаются и могут привести к неправильному поведению в отношении наборов символов.

Исправил A. dos Santos Fernandes

([**CORE-3545**](#)) Проверка ограничений домена CHECK при использовании в объявлениях PSQL была непоследовательной: она использовала тип выражения вместо типа переменной.

Исправил A. dos Santos Fernandes

([**CORE-3055**](#)) Имена переменных или аргументов могут быть неправильными или отсутствовать в сообщениях об ошибках при использовании более 256 переменных.

([**CORE-3047**](#)) Параметры сортировки в EXECUTE BLOCK использовали неправильную логику.

Исправил A. dos Santos Fernandes

([**CORE-2204**](#)) Ограничения на выходные параметры хранимой процедуры проверялись, даже если процедура не возвращала строк.

Исправил A. dos Santos Fernandes

([**CORE-1620**](#)) Неверное сообщение об ошибке — возвращался абсурдный номер столбца, если для инструкции EXECUTE STATEMENT была подготовлена пустая строка SQL.

Исправил Д. Еманов

Язык определения данных Альфа 1

([**CORE-3114**](#)) Попытка удалить несуществующий генератор (последовательность) приводит к серьезному исключению.

Исправил A. dos Santos Fernandes

([**CORE-3056**](#)) Если в одной транзакции после выполнения команды CREATE COLLATION продолжать выполнять другие команды DDL, то при их выполнении могут возникнуть проблемы.

Исправил A. dos Santos Fernandes

([CORE-2696](#)) Команда ALTER TABLE разрешала добавление столбца с определением NOT NULL, позволяя неподготовленному администратору БД повредить таблицу.

Исправил A. dos Santos Fernandes

([CORE-1748](#)) Невосстановимая резервная копия: проблема, которая возникла, если бы команда ALTER TABLE ... ADD COLUMN добавила столбец с ограничением NOT NULL. Исправление для CORE-2696 (см. выше) решило эту проблему.

Исправил A. dos Santos Fernandes

([CORE-1518](#)) Добавление столбца с ненулевым (NOT NULL) значением в существующую таблицу с данными сделает её противоречивой. Исправление для CORE-2696 (см. выше) решило эту проблему.

Исправил A. dos Santos Fernandes

([CORE-1355](#)) Клиентские инструменты, как правило, по разному интерпретируют NULL, который возвращается из столбца, не допускающего значения NULL. Исправление для CORE-2696 (см. выше) теперь не позволяет добавить столбец с ограничением NOT NULL в заполненную таблицу.

Однако неясно, делает ли эта часть исправления обязательной указывать значение по умолчанию для столбца, не допускающего значения NULL.

Исправил A. dos Santos Fernandes

([CORE-634](#)) Неправильное поведение оператора DELETE: когда в предложение WHERE входит подзапрос, включающий операторы FIRST/SKIP операция затрагивает каждую строку в таблице.

Исправил В. Хорсун

([**CORE-304**](#)) Проблема безопасности метаданных: любой пользователь может изменить или удалить генераторы и исключения.

Исправил Д. Еманов

Язык управления данными и DSQL Альфа 1

([**CORE-4144**](#)) При подготовке запроса с UNION неправильно выдавалась ошибка “контекст уже используется (ошибка BLR)” (*“context already in use (BLR error)”*).

Исправил В. Хорсун

([**CORE-4005**](#)) Рекурсивные CTE возвращали неверное сообщение об ошибке.

Исправил В. Хорсун

([**CORE-3416**](#)) Вставка слова, содержащего 8-разрядный символ 'ä' в столбец CHARACTER SET ASCII, будет успешной вместо того, чтобы вызвать ошибку транслитерации.

Исправил A. dos Santos Fernandes

([**CORE-3201**](#)) Внутренняя функция ATAN2 возвращала неверное значение с аргументами (0, 0).

Исправил A. dos Santos Fernandes

([**CORE-3174**](#)) Индекс по выражению с использованием оператора TRIM может привести к некорректному индексированному поиску.

Исправил A. dos Santos Fernandes

([CORE-2699](#)) Общий табличные выражения (СТЕ) выражения теперь может использоваться с параметрами.

Исправил A. dos Santos Fernandes

([CORE-2606](#)) Многобайтовое значение CHAR, запрашиваемое как VARCHAR, возвращалось с заполненными пробелами.

Исправил A. dos Santos Fernandes

([CORE-2238](#)) Использование предложения IS DISTINCT FROM с полями UTF8 и большими полями VARCHAR приводит к ошибке «Превышен предел реализации» (“Implementation limit exceeded”).

Исправил Д. Еманов

([CORE-1188](#)) Предложение STARTING WITH ? (где предоставленное значение параметра - пустая строка) завершится ошибкой, если в плане используется составной индекс.

Исправил A. dos Santos Fernandes

([CORE-92](#)) Бесконечный цикл вставки: INSERT INTO THIS_TABLE SELECT ... FROM THIS_TABLE будет выполняться до тех пор, пока не будут исчерпаны ресурсы.

Исправил В. Хорсун

Утилиты командной строки Альфа 1

([CORE-2547](#)) Утилиты не всегда учитывали минимальное количество символов, необходимое для распознавания опции.

Исправил С. Valderrama С.

Другие старые ошибки, исправленные в утилитах командной строки:

FbGuard

([**CORE-2784**](#)) После падения FBServer Guardian будет продолжать создавать все больше и больше потоков.

Исправил С. Valderrama С.

([**CORE-1595**](#)) Значок Firebird Guardian в трее исчезает после сбоя Windows Explorer.

Исправил С. Valderrama С.

isql

([**CORE-4137**](#)) Утилита isql генерировала выходные данные скрипта метаданных с синтаксическими ошибками в предложении набора символов (CHARACTER SET), например, CHARACTER SETISO8859_1.

Исправил А. dos Santos Fernandes

([**CORE-3431**](#)) Утилита isql неправильно заполняла данные UTF-8.

Исправил А. dos Santos Fernandes

([**CORE-2788**](#)) Утилита isql извлекает размер массива после имени набора символов.

Исправил С. Valderrama С.

gbak

([**CORE-3575**](#)) gbak не поддерживал объемы резервного копирования размером более 4 ГБ.

Исправил А. Пешков

([**CORE-2740**](#)) gbak восстанавливал поврежденные представления без каких-либо предупреждений для пользователя.

Исправил С. Valderrama С.

([**CORE-2545**](#)) В утилите gbak отсутствовало несколько проверок.

Исправил С. Valderrama С.

nbackup

([**CORE-2543**](#)) Утилита nbackup может скрыть реальную причину неудачи инкрементального копирования.

Исправил С. Valderrama С.

Поддержка международных языков Альфа 1

([**CORE-4136**](#)) Символы «Sharp-S» некорректно обрабатываются в сортировке UNICODE_CI_AI.

Исправил А. dos Santos Fernandes

Глава 14

Разработчики Firebird 3.0

Таблица 14.1. Разработчики Firebird 3.0

Разработчик	Страна	Основные задачи
Дмитрий Еманов	Российская Федерация	Штатный разработчик; руководитель основной команды; Разработчик базы данных; Лидер команды разработчиков ядра сервера
Александр Пешков	Российская Федерация	Штатный разработчик; Координатор направления безопасности; Сборка выпусков сервера; Портирование сервера на различные платформы
Claudio Valderrama	Чили	Проверка кода; Поиск и исправление ошибок; Развитие утилиты ISQL; Разработка и поддержка UDF; Разработчик и исполнитель
Владислав Хорсун	Украина	Штатный разработчик; Разработчик базы данных; Разработчик и исполнитель функциональных возможностей SQL
Adriano dos Santos Fernandes	Бразилия	Управление международными наборами символов; Улучшение текстовых типов и текстовых BLOB; Новые функциональные возможности DSQL; Проверка кода
Роман Симаков	Российская Федерация	Спонсор разработки ядра сервера
Paul Beach	Франция	Менеджер релизов; Сборка выпусков для HP-UX, MacOS и Solaris
Pavel Cisar	Чехия	Координатор и разработчик

Разработчик	Страна	Основные задачи
		инструментов контроля качества (QA tools)
Philippe Makowski	Франция	Тестирование качества (QA)
Paul Reeves	Франция	Сборка и установщик Win32
Mark Rotteveel	Нидерланды	Координатор и исполнитель Jaybird
Jiri Cincura	Франция	Координатор и разработчик .NET провайдеров
Александр Потапченко	Российская Федерация	Координатор и разработчик ODBC/JDBC драйверов для Firebird
Stephen Boyd	Канада	Спонсор разработки GPRE
Алексей Ковязин	Российская Федерация	Координатор сайта Firebird
Paul Vinkenoog	Нидерланды	Координатор проекта документации Firebird; Автор документации; Разработчик и исполнитель утилит
Norman Dunbar	Великобритания	Автор документации
Павел Меньшиков	Российская Федерация	Переводчик документации
Tomneko Hayashi	Япония	Переводчик документации
Umberto (Mimmo) Masotti	Италия	Переводчик документации
Helen Borrie	Австралия	Редактор и составитель замечаний к выпускам (Release Notes)

Приложение А: 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 3.0 Release Notes. Первоначальный Автор Оригинальной Документации: Хелен Борри.

Copyright (C) 2004–2015. Все права защищены. Адрес электронной почты для контакта: `helebor at users dot sourceforge dot net`.

Перевод на русский язык: Александр Карпейкин. Copyright © 2008-2017. Все права защищены.

Адрес электронной почты для контакта: `karp dot fb at gmail dot com`.

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 3.0 Release Notes.

The Initial Writer of the Original Documentation is: Helen Borrie. Persons named in attributions are Contributors.

Copyright (C) 2004-2015. All Rights Reserved. Initial Writer contact: `helebor at users dot sourceforge dot net`.