Руководство по миграции на Firebird 5.0

Симонов Денис

Версия 1.0 от 23.02.2024

Содержание

Этот материал был создан при поддержке и спонсорстве компании iBase.ru, которая разрабатывает инструменты Firebird SQL для предприятий и предоставляет сервис технической поддержки для Firebird SQL.

Введение

Эта статья предназначена прежде всего для тех, кто планирует в ближайшем будущем обновить СУБД Firebird до версии 5.0. Многие администраторы всё ещё используют Firebird 2.5, но планируют обновиться до версии 5.0. Именно поэтому здесь описан процесс миграции с Firebird версий 2.5, 3.0 и 4.0.

Спонсором написания статьи «Руководство по миграции на Firebird 5.0» является IBSurgeon (iBase.ru) (https://www.ib-aid.com, https://www.ibase.ru): техническая поддержка и инструменты разработчика и администратора для СУБД Firebird.

Глава 1. Ручная установка Firebird 5.0 в Windows

Далее описан процесс установки из zip архива. Даже если вы устанавливаете Firebird из специального инсталляционного пакета, вам всё равно может потребоваться изменить некоторые настройки после установки. Кроме того, ручная установка позволяет установить несколько версий Firebird на одной машине.

Скачайте архив соответствующей разрядности и распакуйте его в директорию, в которой будет размещён сервер Firebird.

Далее необходимо создать пользователя SYSDBA. Для тех администраторов, которые мигрируют с Firebird 3.0 или 4.0, это операция не нова. В Firebird 2.5 и ранее после установки пользователь SYSDBA существовал всегда и имел пароль по умолчанию *masterke*, который надо было сразу же менять.

1.1. Инициализация SYSDBA

Начиная с Firebird 3.0 пользователь SYSDBA не инициализирован по умолчанию (для плагина управления пользователями SRP), поэтому необходимо явно создать пользователя и указать ему пароль. Это можно сделать двумя способами: с использованием консольного инструмента для выполнения интерактивных запросов `isql.exe` и консольного инструмента для управления базой данных безопасности `gsec.exe`.

Замечание

В зависимости от размещения Firebird эти утилиты могут потребовать запуска с привилегиями администратора.

1.1.1. Инициализация SYSDBA с помощью ISQL

Запустите инструмент для выполнения интерактивных запросов `isql.exe`. Соединитесь с базой данных безопасности в режиме встроенного сервера, указав при этом пользователя SYSDBA без пароля. Пользователя SYSDBA ещё не существует в базе данных безопасности, но в embedded режиме пользователь и его пароль не проверяется, и Firebird доверяет любому указанному имени пользователя. Выполните SQL запрос для создания пользователя SYSDBA:

```
CREATE USER SYSDBA PASSWORD '<password>';
```

Пользователь SYSDBA инициализирован, можно выходить из интерактивного режима.

Пример 1. Инициализация SYSDBA через ISQL

```
c:\Firebird\5.0>isql security.db -user SYSDBA

Database: security.db, User: SYSDBA

SQL> CREATE USER SYSDBA PASSWORD 'm8ku234pp';
SQL> exit;
```

1.1.2. Инициализация SYSDBA с помощью GSEC

Запустите `gsec.exe`, указав пользователя SYSDBA и базу данных `security.db`. Выполните команду для добавления пользователя SYSDBA:

```
add SYSDBA -pw <password>
```

Пример 2. Инициализация SYSDBA через GSEC

```
c:\Firebird\5.0>gsec -user SYSDBA -database security.db

* gsec is deprecated, will be removed soon *

GSEC> add SYSDBA -pw m8ku234pp
GSEC> quit
```

Предупреждение

Инструмент `gsec.exe` является устаревшим, многие возможности, доступные через SQL, недоступны в нём.

1.2. Конфигурация

Перед тем как установить и запустить Firebird как службу необходимо выбрать режим работы сервера.

1.2.1. Режим сервера

По-умолчанию Firebird будет стартовать в режиме SuperServer. Если вы хотите чтобы сервер запускался в другой архитектуре, то необходимо изменить значение параметра *ServerMode* в `firebird.conf`. Раскомментируйте его (удалите решётку) и установите нужный режим: Super, SuperClassic или Classic.

```
ServerMode = Classic
```

1.2.2. Авторизация из клиентских библиотек Firebird 2.5

В Firebird 5.0 по умолчанию используется безопасная парольная аутентификация (SRP). Клиенты Firebird 2.5 и более ранние версии использовали традиционную аутентификацию (Legacy_Auth), которая отключена в Firebird 5.0 по-умолчанию, поскольку не является безопасной.

Для поддержки традиционной аутентификации необходимо изменить следующие параметры *AuthServer*, *UserManager* и *WireCrypt*.

Пример 3. Включение авторизации с предыдущими версиями клиента Firebird

```
AuthServer = Srp256, Srp, Legacy_Auth
UserManager = Srp, Legacy_UserManager
WireCrypt = Enabled
```

После вышеперечисленных манипуляций у нас будет активно два менеджера пользователей, по умолчанию активен первый в списке UserManager.

Важно

Одноименные пользователи в разных менеджерах пользователей — это разные пользователи и у них могут быть разные пароли. Это относится и к SYSDBA и владельцу базы данных.

Если вам не нужна поддержка безопасной парольной аутентификации (SRP), то удалите из `AuthServer` плагины Srp256 и Srp; из `UserManager` — Srp, а `WireCrypt` можете изменить на Disabled.

Ранее мы уже создали SYSDBA в менеджере пользователей SRP. В Legacy_UserManager SYSDBA уже существует, причём со стандартным паролем *masterkey*, который необходимо изменить. Сделаем это с использованием инструмента `isql`. В операторе `ALTER USER` необходимо обязательно указать менеджер пользователей Legacy_UserManager.

Пример 4. Изменение пароля SYSDBA в Legacy_UserManager

```
c:\Firebird\5.0>isql security.db -user SYSDBA

Database: security.db, User: SYSDBA

SQL> ALTER USER SYSDBA SET PASSWORD 'er34gfde' USING PLUGIN Legacy_UserManager;
SQL> exit;
```

1.2.3. Установка часового пояса соединения по умолчанию

Начиная с Firebird 4.0 доступны новые типы даты и времени с поддержкой часовых поясов.

Даже если вы не собираетесь в ближайшее время использовать типы с часовыми поясами, то необходимо учитывать, что выражения CURRENT_TIMESTAMP и CURRENT_TIME теперь возвращают типы данных с часовыми поясами. Существует режим совместимости, который позволяет преобразовать типы с часовыми поясами в типы без часовых поясов. Однако такое преобразование может работать неверно, если часовой пояс соединения выставлен неправильно.

Обычно часовой пояс сеанса задаётся на стороне клиента. Если часовой пояс на стороне клиента не выставлен, то по умолчанию используется часовой пояс операционной системы. Вы также можете выставить часовой пояс сеанса по умолчанию с помощью параметра конфигурации `DefaultTimeZone`.

```
DefaultTimeZone = Europe/Moscow
```

1.2.4. Одновременный запуск нескольких экземпляров Firebird

Здесь предполагается, что вы хотите запустить экземпляры разных версий Firebird, каждая из которых установлена в своём каталоге.

Для одновременного запуска нескольких экземпляров Firebird необходимо развести их по разным портам tcp (если, конечно, слушатель запущен в режиме прослушивания TCP/IP). Для этого необходимо изменить в `firebird.conf` параметр *RemoteServicePort*.

Например, если у вас уже есть один сервер, который слушает порт 3050, то необходимо установить любой другой свободный порт, например 3051. В этом случае в строке подключения необходимо будет указывать новый порт (кроме случая когда приложению и клиенту Firebird доступен `firebird.conf` с измененным номером порта по умолчанию).

```
RemoteServicePort = 3051
```

Также необходимо установить уникальные значения параметра *IpcName* для каждого экземпляра сервера СУБД. Это позволит избежать сообщения об ошибках в `firebird.log`. Эти ошибки не являются критическими, если вы не пользуетесь протоколом XNET. Однако, если он используется, то стоит учесть, что этот параметр придётся изменять и на стороне клиента через DPB.

1.3. Установка и запуск Firebird как службы

Утилита *instsvc.exe* записывает, удаляет или меняет информацию о запуске сервера в базе сервисов операционной системы. Кроме того, она позволяет управлять запуском и остановкой сервиса.

Если запустить её без параметров, то будет выведена справка по командам и параметрам.

```
instsvc
Usage:
  instsvc i[nstall]
                   [ -a[uto]* | -d[emand] ]
                   [ -g[uardian] ]
                   [ -l[ogin] username [password] ]
                   [ -n[ame] instance ]
                   [ -i[nteractive] ]

          sta[rt]   [ -b[oostpriority] ]
                    [ -n[ame] instance ]
          sto[p]    [ -n[ame] instance ]
          q[uery]
          r[emove]  [ -n[ame] instance ]


  '*' denotes the default values
  '-z' can be used with any other option, prints version
  'username' refers by default to a local account on this machine.
  Use the format 'domain\username' or 'server\username' if appropriate.
```

Важно

Утилита instsvc должна запускаться в консоли с административными привилегиями (запуск консоли от имени администратора).

Для установки сервиса необходимо ввести команду

```
instsvc install
```

В этом случае Firebird будет установлен в качестве службы с именем "Firebird Server – DefaultInstance". Эта служба будет запускаться автоматически при старте ОС, под учётной записью LocalSystem, предназначенной для служб.

Если необходимо чтобы было установлено несколько экземпляров Firebird работающих как службы, то необходимо задать им разные имена с помощью опции `-n`

```
instsvc install -n fb50
```

Для запуска службы воспользуйтесь командой

```
instsvc start
```

Если служба была установлена с именем отличным от умолчательного, то необходимо воспользоваться переключателем `-n`

```
instsvc start -n fb50
```

Для остановки службы воспользуйтесь командой

```
instsvc stop
```

Если служба была установлена с именем отличным от умолчательного, то необходимо воспользоваться переключателем `-n`

```
instsvc stop -n fb50
```

Для удаления сервиса необходимо ввести команду

```
instsvc remove
```

Если служба была установлена с именем отличным от умолчательного, то необходимо воспользоваться переключателем `-n`

```
instsvc remove -n fb50
```

Для просмотра всех служб Firebird установленных в системе воспользуйтесь командой

```
instsvc query
```

```
Firebird Server - fb30 IS installed.
  Status  : running
  Path    : C:\Firebird\3.0\firebird.exe -s fb30
  Startup : automatic
  Run as  : LocalSystem

Firebird Server - fb40 IS installed.
  Status  : running
  Path    : C:\Firebird\4.0\firebird.exe -s fb40
  Startup : automatic
  Run as  : LocalSystem

Firebird Server - fb50 IS installed.
  Status  : running
  Path    : C:\Firebird\5.0\firebird.exe -s fb50
  Startup : automatic
  Run as  : LocalSystem
```

1.3.1. Использование install_service.bat и uninstall_service.bat

Для упрощения процедуры установки и удаления служб в ZIP архиве в комплекте с Firebird поставляются два BAT файла: `install_service.bat` и `uninstall_service.bat`.

В этом случае процедура установки Firebird в качестве сервиса выглядит следующим образом

```
install_service.bat
```

В этом случае процедура удаления службы Firebird выглядит следующим образом

```
uninstall_service.bat
```

Если необходимо задать службе имя отличное от умолчательного, то указываем это имя в качестве аргумента

```
install_service.bat fb50
```

Если служба была установлена с именем отличным от умолчательного, то указываем это имя в качестве аргумента

```
uninstall_service.bat fb50
```

1.4. Установка клиента

Если речь идёт об установке только клиентской части, то обязательно требуется файл `fbclient.dll`. Клиент Firebird 5.0 обязательно требует наличия установленного Microsoft Runtime C++ 2015-2022 соответствующей разрядности. Если данная библиотека не установлена, то можно скопировать дополнительные библиотеки, которые поставляются в ZIP архиве под Windows `msvcp140.dll` и `vcruntime140.dll` (для 64-разрядной установки ещё и `vcruntime140_1.dll`).

Желательно, чтобы рядом с `fbclient.dll` был расположен файл сообщений `firebird.msg`. Большинство сообщений об ошибках уже содержатся в `fbclient.dll`, однако если вы собираетесь пользоваться консольными утилитами файл `firebird.msg` обязательно должен присутствовать.

В отличие от Firebird 2.5 и Firebird 3.0, клиентской библиотеки так же требуются файлы ICU (`icudt63.dll`, `icuin63.dll`, `icuuc63.dll` и `icudt63l.dat`). Ранее ICU библиотека требовалась только серверу. Теперь она может потребоваться клиентской части, если вы собираетесь работать с типами данных `TIMESTAMP WITH TIME ZONE` и `TIME WITH TIME ZONE`. ICU библиотека также требуется при вызове функций `UtilInterface::decodeTimeTz()` и `UtilInterface::decodeTimestampTz()`.

Замечание

В Windows 10 может использоваться ICU библиотека поставляемая вместе с операционной системой.

Если необходимо сжатие трафика при работе по TCP/IP, то потребуется библиотека `zlib1.dll`.

Вам может потребоваться библиотека `plugins/chacha.dll`, если вы собираетесь использовать плагин шифрования трафика ChaCha. Этот плагин, используется по умолчанию начиная с Firebird 4.0, поскольку он находится первым списке значений в параметре конфигурации `WireCryptPlugin = ChaCha, Arc4`.

Замечание о загрузке плагинов

`fbclient.dll` версии 3.0 по умолчанию не загружал плагины из динамических библиотек из каталога `plugins`. `fbclient.dll` версии 4.0 и выше использует `plugins/chacha.dll` по умолчанию, если этот плагин присутствует. Отсутствующие плагины игнорируются.

Однако, есть важная особенность. `fbclient.dll` ищет в своём каталоге файл `firebird.conf`, и если он отсутствует, то пытается найти его на каталог выше. Каталог где будет найден `firebird.conf` является корневым каталогом — от которого отсчитываются все остальные известные относительные пути (plugins, intl).

Такое поведение может сыграть с вами злую шутку. Дело в том что 64-разрядный инсталлятор располагает в папке `$(fbroot)/WOW64` 32-разрядную библиотеку `fbclient.dll`. Если захотите использовать библиотеку из данного каталога, то можете получить следующее сообщений об ошибке

```
Error loading plugin ChaCha.
Module C:\Firebird\5.0\plugins/ChaCha exists but can not be loaded.
unknown Win32 error 193.
```

В данном случае 32-разрядный `fbclient.dll` пытался загрузить 64-разрядный плагин ChaCha.

Для исправления данной ошибки достаточно поместить в папку `$(fbroot)/WOW64` файл `firebird.conf`.

Библиотека `fbclient.dll`, а также другие файлы клиентской библиотеки, должны располагаться либо рядом с приложением, либо в одной из директорий в которой производится поиск, например добавленной в `PATH` или системной директории для размещения общедоступных библиотек (`system32` или `SysWOW64`).

Важно

Размещение клиентской библиотеки в `PATH` может помешать другим приложениям, которым требуется клиентская библиотека другой версии или другого сервера. Поэтому, если предполагается, что приложение должно работать независимо от других приложений с конкретной версией клиента, то файлы клиента требуется разместить в папке приложения, и не прописывать этот путь в `PATH`.

1.4.1. Утилита instclient

Для развёртывания клиентской библиотеки Firebird в системном каталоге Windows воспользуйтесь командой

```
instclient install fbclient
```

Важно

Утилита instclient не копирует в системный каталог никаких файлов кроме `fbclient.dll`.

1.5. Установка embedded версии

Начиная с версии Firebird 3.0, embedded версия не распространяется отдельно. Вы можете использовать один и тот же набор файлов и как сетевой сервер и как встраиваемый (embedded) сервер. Но, если требуется встраиваемый комплект минимального размера, то структура файлов и каталогов для Firebird 5.0 embedded следующая:

- intl
  - fbintl.conf
  - fbintl.dll
- plugins
  - engine13.dll
- firebird.conf
- icudt63l.dat
- fbclient.dll
- ib_util.dll
- icudt63.dll
- icuin63.dll
- icuuc63.dll
- msvcp140.dll
- vcruntime140.dll

- vcruntime140_1.dll
- firebird.msg

При необходимости вы также можете скопировать исполняемые файлы утилит `fbsvcmgr.exe`, `fbtracemgr.exe`, `gbak.exe`, `gfix.exe`, `gstat.exe`, `isql.exe`, `nbackup.exe`. Если вы собираетесь использовать `gbak` вместе с переключателем `-zip`, то потребуется также библиотека `zlib1.dll`.

Замечание

Для тех кто мигрирует с Firebird 2.5 следует учитывать 2 момента:

- Вместо единой библиотеки `fbembed.dll` требуется несколько файлов, причём файл `fbclient.dll` переименовывать нельзя. Компоненты доступа должны использовать в качестве точки входа именно библиотеку `fbclient.dll`.
- В файле конфигурации `firebird.conf` следует изменить значение параметра `ServerMode` на `SuperClassic` или `Classic` для того чтобы на одном компьютере можно было подключаться к одной и той же базе данных из разных приложений, использующих embedded (поведения Firebird 2.5 embedded по умолчанию).

Глава 2. Преобразование базы данных к новому формату

Базы данных Firebird 5.0 имеют ODS (On-Disk Structure) 13.1. Для того чтобы Firebird 5.0 мог работать с вашей базой данных её необходимо привести к родной ODS. Обычно это осуществляется с помощью инструмента `gbak`. Однако не торопитесь делать резервное копирование своей БД и её восстановление с новой ODS — сначала необходимо устранить возможные проблемы совместимости.

2.1. Список несовместимостей на уровне языка SQL

Проблемы совместимости языка SQL возможны как для объектов самой базы данных (PSQL процедуры и функции), так и в DSQL запросах, используемых в вашем приложении.

Для обнаружения проблем совместимости языка SQL для объектов базы данных рекомендуется следующий способ. Выполните извлечение метаданных базы данных в скрипт на старой версии Firebird.

```
isql <database> -x -o create_script.sql
```

Раскоментируйте внутри скрипта оператор `CREATE DATABASE`, внесите в него необходимые правки, и попробуйте создать новую базу данных из скрипта в Firebird 5.0:

```
isql -i create_script.sql -o error.log -m
```

где, ключ `-i` – входной файл скрипта; ключ `-o` – выходной файл сообщений; ключ `-m` заставляет `isql` выводить сообщения об ошибках в выходной файл сообщений.

Далее смотрим файл `error.log` на наличие ошибок, и в случае их обнаружения, меняем метаданные в исходной БД. Повторяем описанный выше алгоритм до тех пор, пока все ошибки не будут устранены. После чего можно спокойно делать backup/restore.

Далее перечислим некоторые наиболее часто встречающиеся проблемы совместимости на уровне SQL, которые вы можете исправить ещё до перехода на новую ODS. Полный список несовместимостей вы можете прочитать в Release Notes 5.0 в главе "Compatibility Issues". При миграции с 3.0 необходимо также ознакомится с одноимённой главой в Release Notes 4.0, а при миграции с 2.5 — Release Notes 3.0.

2.1.1. Новые зарезервированные слова

Проверьте вашу базу данных на наличие новых зарезервированных слов в идентификаторах, столбцах и переменных. В первом SQL диалекте такие слова не могут применяться в принципе (надо будет переименовать), в третьем — могут применяться, но должны обрамляться двойными кавычками.

Список новых ключевых и зарезервированных слов вы можете найти в Release Notes 3.0 и 4.0 в главе "Reserved Words and Changes". Ключевые слова могут применяться в качестве идентификаторов, хотя это не рекомендуется.

Начиная с Firebird 5.0 вы можете посмотреть полный список ключевых и зарезервированных слов с помощью запроса:

```
SELECT
  RDB$KEYWORD_NAME,
  RDB$KEYWORD_RESERVED
FROM RDB$KEYWORDS
```

Этот запрос можно выполнить на любой БД с ODS 13.1, например на `employee.db`, входящей в поставку Firebird 5.0.

Столбец `RDB$KEYWORD_NAME` содержит само ключевое слово, а `RDB$KEYWORD_RESERVED` - флаг является ли ключевое слово зарезервированным.

2.1.2. Имена столбцов в PSQL курсорах

Актуально: при миграции с Firebird 2.5.

Все выходные столбцы в PSQL курсорах объявленных как `DECLARE CURSOR` должны иметь явное имя или псевдоним. То же самое касается PSQL курсоров используемых как `FOR SELECT ⋯ AS CURSOR <cursor name> DO ⋯`.

Пример 5. Проблема с неименованными столбцами в курсорах

```
create procedure sp_test
returns (n int)
as
  declare c cursor for (select 1 /* as a */ from rdb$database);
begin
  open c;
  fetch c into n;
  close c;
  suspend;
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
```

2.1.3. Новые типы данных

Актуально: при миграции с Firebird версий 2.5, 3.0.

В Firebird 4.0 введены новые типы данных:

- `TIMESTAMP WITH TIME ZONE`
- `TIME WITH TIME ZONE`
- `INT128`
- `NUMERIC(38, x)` и `DECIMAL(38, x)`
- `DECFLOAT(16)` и `DECFLOAT(34)`

  Последние два типа не вызывают особых проблем, поскольку раньше вы их не использовали, и обычно выражения их не возвращают.

Некоторые выражения теперь могу возвращать типы `NUMERIC(38, x)`, `DECIMAL(38, x)` и `INT128`. О решении этой проблемы мы поговорим позже, поскольку на этапе изменения ODS они обычно не проявляются.

Выражения `CURRENT_TIMESTAMP` и `CURRENT_TIME` теперь возвращают типы `TIMESTAMP WITH TIME ZONE` и `TIME WITH TIME ZONE`.

Для старых клиентских библиотек и приложений вы можете установить режим совместимости типов, однако это не поможет внутри хранимых процедур, функций и триггеров. Вам необходимо использовать выражения `LOCALTIMESTAMP` и `LOCALTIME` вместо `CURRENT_TIMESTAMP` и `CURRENT_TIME` там где вы не хотите получить типы данных с часовыми поясами. Данные выражения специально были введены в корректирующих релизах Firebird 2.5.9 и Firebird 3.0.4, чтобы вы заранее могли подготовить свои базы данных для миграции на Firebird 4.0 и выше.

При присваивании переменной (столбца) типа `TIMESTAMP` значения выражения `CURRENT_TIMESTAMP` будет произведено преобразование типа, то есть неявный `CAST(CURRENT_TIMESTAMP AS TIMESTAMP)`, поэтому даже без замены `CURRENT_TIMESTAMP` и `CURRENT_TIME` на `LOCALTIMESTAMP` и `LOCALTIME` всё будет продолжать работать, но производительность в некоторых случаях может упасть. Например:

```
create global temporary table gtt_test (
    id  integer not null,
    t   timestamp default current_timestamp
) on commit preserve rows;

alter table gtt_test add constraint pk_gtt_test primary key (id);
```

Здесь поле `t` имеет тип `TIMESTAMP`, а `CURRENT_TIMESTAMP` возвращает `TIMESTAMP WITH TIME ZONE` из-за чего производительность `INSERT` в такую таблицу снижается.

Этот случай подробно описан в баг трекере, тикет 7854.

Первоначально падение производительности составляло 30%, что довольно существенно, но после ряда оптимизаций оверхед удалось снизить до 3-5%. Если вы не хотите лишних затрат, то лучше использовать `LOCALTIMESTAMP` там, где не предполагается оперировать временем с часовым поясом.

2.1.4. Литералы дат и времени

Актуально: при миграции с Firebird версий 2.5, 3.0.

В Firebird 4.0 ужесточён синтаксис литералов дат и времени.

Литералы 'NOW', 'TODAY', 'TOMORROW', 'YESTERDAY' с префиксами TIMESTAMP, DATE, TIME теперь запрещены. Дело в том, что значение таких литералов вычислялось во время подготовки DSQL запроса или компиляции PSQL модулей, что приводило к неожиданным результатам.

Если что-то вроде TIMESTAMP 'NOW' использовалось в запросах DSQL в коде приложения или в перенесенном PSQL, возникнет проблема совместимости с Firebird 4 и выше.

Пример 6. Следующий код не будет скомпилирован

```
..
DECLARE VARIABLE moment TIMESTAMP;
..
SELECT TIMESTAMP 'NOW' FROM RDB$DATABASE INTO :moment;

/* здесь переменная: moment будет "заморожена" как отметка времени
в момент последней компиляции процедуры или функции  */
..
```

Необходимо вычистить такие литералы, например заменить их на явное преобразование `CAST('NOW' AS TIMESTAMP)`, в коде ваших процедур и функций до преобразования вашей базы данных в новую ODS.

Кроме того, необходимо проверить другие литералы дат и времени с явным заданием известной даты (времени). Ранее в таких литералах позволялись разделители частей даты и времени не соответствующие стандарту. Теперь такие разделители запрещены. Подробнее о разрешённых форматах литералов даты и времени вы можете прочитать в "Руководство по языку SQL СУБД Firebird 5.0" в главе "Литералы даты и времени".

2.1.5. INSERT ... RETURNING требует привилегию SELECT

Актуально: при миграции с Firebird версий 2.5, 3.0.

Начиная с Firebird 4.0, если какой-либо оператор `INSERT` содержит предложение `RETURNING`, которое ссылается на столбцы базовой таблицы, то вызывающей стороне должна быть

предоставлена соответствующая привилегия SELECT.

2.1.6. Эффект стабильности курсора

Актуально: при миграции с Firebird 2.5.

В Firebird 3.0 было сделано важное улучшение, которое называется "стабильность курсоров". В следствии этого улучшения некоторые запросы могут работать по-другому. Это прежде всего касается запросов, которые изменяют таблицу и читают её в том же курсоре. Стабильность курсора позволяет устранить множество ошибок, присутствующих в предыдущих версиях Firebird, самой известной из которых, является бесконечный цикл в запросе:

```
INSERT INTO some_table
SELECT * FROM some_table
```

Маловероятно, что ваши приложения содержат именно такие запросы, тем не менее стабильность курсора может проявляться не совсем в очевидных случаях:

- некий DML триггер модифицирует таблицу, а затем в том же триггере происходит чтение этой таблицы через оператор SELECT. Если данные были модифицированы не в текущем контексте выполнения триггера, то вы можете не увидеть изменения в SELECT запросе;
- селективная хранимая процедура SP_SOME изменяет записи в некоторой таблицы SOME_TABLE, а затем вы выполняете JOIN с той же таблицей:

```
FOR
  SELECT ...
  FROM SP_SOME(...) S
  JOIN SOME_TABLE ...
```

Если в вашем коде присутствуют подобные случаи, то рекомендуем переписать данные части с учётом эффекта "стабильности курсора".

2.2. Поддержка внешних функций (UDF) объявлена устаревшей

Поддержка внешних функций (UDF) начиная с Firebird 4 объявлена устаревшей.

Эффект от этого заключается в том, что UDF нельзя использовать с конфигурацией по умолчанию, поскольку для параметра UdfAccess в firebird.conf значение по умолчанию теперь None. Библиотеки UDF ib_udf и fbudf изъяты из дистрибутива.

Большинство функций в этих библиотеках уже устарели в предыдущих версиях Firebird и были заменены встроенными аналогами. Теперь доступны безопасные замены для некоторых из оставшихся функций либо в новой библиотеке определяемых пользователем

подпрограмм (UDR) с именем `[lib]udf_compat.[dll/so/dylib]` (это делается после смены ODS), либо в виде преобразований по сценарию в сохраненные функции PSQL.

Рекомендуем заранее (до перехода на новую ODS) заменить UDF функции на их встроенные аналоги. Если вы делаете миграцию с Firebird 3.0, вы также можете переписать часть функций на PSQL.

Если после этих шагов у вас остались UDF функции, то необходимо изменить параметр конфигурации

```
UdfAccess = Restrict UDF
```

2.3. Преобразование базы данных к новой ODS

После предварительной подготовки, вы можете попробовать преобразовать базу данных к новой ODS с помощью инструмента *gbak*.

Рекомендовать всегда начинать с backup/resore метаданных:

```
old_version\gbak -b -g -m old_db stdout | new_version\gbak -c -m stdin
new_db
```

Иначе можно получить ошибку метаданных после того, как будет записан весь терабайт данных, что будет очень обидно. Кроме того, на восстановленных в новой версии метаданных удобно проверять работу скриптов перекомпиляции объектов базы.

В данном примере предполагается, что на одной машине стоят Firebird 3.0 и Firebird 5.0. Firebird 3.0 работает используя TCP порт 3053, а Firebird 5.0 — 3055.

Прежде всего необходимо создать резервную копию вашей базы данных на текущей версии Firebird с помощью следующей команды.

```
gbak -b -g -V -user <username> -pas <password> -se <service> <database> <backup_file>
-Y <log_file>
```

Пример 7. Создание резервной копии на текущей версии Firebird

```
gbak -b -g -V -user SYSDBA -pas 8kej712 -se server/3053:service_mgr my_db
d:\fb30_backup\my_db.fbk -Y d:\fb30_backup\backup.log
```

Далее необходимо восстановить вашу копию на Firebird 5.0.

```
gbak -c -v -user <username> -pas <password> -se <service> <backup_file>
<database_file> -Y <log_file>
```

Начиная с Firebird 5.0 утилита gbak может создавать резервную копию и восстанавливать базу данных используя параллелизм. Количество параллельных потоков, используемых при backup или restore, указывается с помощью опции -parallel или сокращённо -par. Использвание параллельных потоков может ускорить процесс восстановления в 2-3 раза, в зависимости от вашего аппартного обеспечения и базы данных.

По умолчанию, паралелизм отключен в Firebird 5.0. Для того, чтобы была возможность его использовать необходимо установить параметр MaxParallelWorkers в firebird.conf. Этот параметр ограничивает максимальное количество паралеллельных потоков, которое может быть использовано ядром Firebird или его утилитами. По умолчанию он равен 1. Рекомендуется установить MaxParallelWorkers в значение равное максимальному количеству физических или логических ядер вашего процессора (или процессоров).

Теперь для восстановления вы можете использовать следующую команду.

```
gbak -c -par <N>  -v -user <username> -pas <password> -se <service> <backup_file>
<database_file> -Y <log_file>
```

Здесь N - количество паралельных потоков которое будет использовать gbak, оно должно быть меньшим или равным значению установленном в MaxParallelWorkers.

Пример 8. Восстановление резервной копии на Firebird 5.0 с использованием 8 паралельных потоков

```
gbak -c -par 8 -v -user SYSDBA -pas 8kej712 -se server/3055:service_mgr
d:\fb30_backup\my_db.fbk d:\fb50_data\my_db.fdb -Y d:\fb50_data\restore.log
```

Важно

Обратите внимание, на переключатели -V и -Y, они обязательно должны использоваться, чтобы вы могли просмотреть в лог файле, что в процессе восстановления пошло не так.

После восстановления внимательно изучите restore.log на предмет ошибок. Однако, в этом логе не будет ошибок несовместимости уровня SQL, поскольку объекты БД при restore не перекомпилируются. Если какая-то процедура или триггер содержат несовместимые конструкции, то впоследствии при ALTER такого объекта будет выдана ошибка.

Полностью очистить БД от таких ошибок можно только если извлечь скрипт из БД операцией

```
isql -x <database> > script.sql
```

в предыдущей версии Firebird, и создать пустую БД в Firebird 5.0 из этого скрипта, исправляя возникающие ошибки создания метаданных по очереди.

2.3.1. Предупреждения об отсутствии UDF

После восстановления в файле `restore.log` вы можете увидеть следующие предупреждения

```
gbak: WARNING:function UDF_FRAC is not defined
gbak: WARNING: module name or entrypoint could not be found
```

Это означает, что у вас есть UDF, которые объявлены в базе данных, но их библиотека отсутствует. Выше уже было описано, что надо делать в этом случае. Но это в основном касалось ваших UDF библиотек. Однако если вы использовали UDF из комплекта, поставляемого с Firebird, а именно ib_udf и fbudf, то вы можете заменить их на встроенные функции или на безопасные аналоги UDR расположенные в библиотеке `udf_compat.dll`. Для этого необходимо запустить SQL скрипт миграции, поставляемый в комплекте с Firebird 5.0, который расположен в `misc/upgrade/v4.0/udf_replace.sql`. Это делается следующей командой

```
isql -user sysdba -pas masterkey -i udf_replace.sql {your-database}
```

Пример 9. Предупреждение

Этот сценарий не повлияет на объявления UDF из сторонних библиотек!

2.3.2. Быстрое обновление ODS при миграции с Firebird 4.0

Если вы производите миграцию с Firebird 4.0, то существует более быстрый способ обновления ODS, чем backup/restore.

Традиционным способом обновления ODS (On-Disk Structure) является выполнение backup на старой версии Firebird и restore на новой. Это довольно длительный процесс, особенно на больших базах данных.

Однако, в случае обновления минорной версии ODS (номер после точки) backup/restore является избыточным (необходимо лишь добавить недостающие системные таблицы и поля, а также некоторые пакеты). Примером такого обновления является обновление ODS 13.0 (Firebird 4.0) до ODS 13.1 (Firebird 5.0), поскольку мажорная версия ODS 13 осталось той же.

Начиная с Firebird 5.0 появилась возможность обновления минорной версии ODS без длительный операция backup и restore. Для этого используется утилита `gfix` с переключателем `-upgrade`.

Ключевые моменты:

- Обновление необходимо производить вручную с помощью команды `gfix -upgrade`
- Требуется монопольный доступ к базе данных, в противном случае выдается ошибка.
- Требуется системная привилегия `USE_GFIX_UTILITY`.
- Обновление является транзакционным, все изменения отменяются в случае возникновения ошибки.
- После обновления Firebird 4.0 больше не может открывать базу данных.

- Это односторонняя модификация, возврат назад невозможен. Поэтому перед обновлением сделайте копию базы данных (с помощью `nbackup -b 0`), чтобы иметь точку восстановления, если что-то пойдет не так во время процесса.
- Обновление ODS с помощью `gfix -upgrade` не изменяет страницы данных пользовательских таблиц, таким образом записи не будут перепакованы с помощью нового алгоритма сжатия RLE. Но вновь вставляемые записи будут сжаты с помощью усовершенствованного RLE.

Таким образом, для быстрого обновления ODS вам необходимо проделать следующие шаги:

- Сделать резервную копию базы данных, например с помощью `nbackup -b 0`, чтобы иметь точку восстановления, если что-то пойдет не так.
- Выполнить команду:

```
gfix -upgrade <dbname> -user <username> -pass <password>
```

Данный способ обновления ODS в отличие от backup/restore, занимает секунды (речь о `gfix -upgrade`), а не минуты или часы.

Глава 3. Перенос псевдонимов баз данных

Этот раздел актуален для тех кто мигрирует с Firebird 2.5.

Файл `aliases.conf` в котором настраивались псевдонимы баз данных переименован в `databases.conf`. Он полностью обратно совместим по синтаксису, однако его назначение значительно расширено. Теперь в нём можно задавать некоторые индивидуальные параметры для каждой базы данных. Настоятельно рекомендуем воспользоваться этой возможностью, если ваш сервер обслуживает более одной базы данных.

Параметры, которые можно задавать на уровне базы данных, помечены в файле `firebird.conf` надписью 'Per-database configurable'.

Глава 4. Перенос списка пользователей

Перенос списка пользователей из Firebird версий 2.5, 3.0 и 4.0 осуществляется по-разному.

4.1. Перенос списка пользователей из Firebird 4.0

Самым простым будет перенос списка пользователей из Firebird 4.0.

Чтобы перенести базу данных безопасности с Firebird 4.0 на 5.0, создайте резервную копию файла `security4.fdb` с помощью `gbak` Firebird 4.0 и восстановите его как `security5.fdb` с помощью `gbak` Firebird 5.0. Используйте `gbak` локально (используя встроенное соединение), пока Firebird Server не запущен.

Копирование файла `security4.fdb` и переименование его в `security5.fdb` и обновление ODS с помощью опции `gfix -UPGRADE` также будет работать, но мы рекомендуем выполнить резервное копирование и восстановление.

4.2. Перенос списка пользователей из Firebird 3.0

Чтобы перенести пользователей из базы безопасности Firebird 3.0 в базу данных безопасности Firebird 4.0 необходимо выполнить резервную копию `security3.fdb` с помощью *gbak* и восстановите его как `security5.fdb` с помощью `gbak` Firebird 5.0.

Однако учтите, что в этом случае вы потеряете некоторые новые возможности. Мы пойдём более сложным способом:

1. Сделайте резервную копию базы данных безопасности на Firebird 3.0

```
c:\Firebird\3.0>gbak -b -g -user SYSDBA security.db d:\fb30_backup\security.fbk
```

2. Восстановите резервную копию на Firebird 5.0 под новым именем

```
c:\Firebird\5.0>gbak -c -user SYSDBA -pas 8kej712 -se localhost/3054:service_mgr
d:\fb30_backup\security.fbk d:\fb50_data\security_30.fdb
```

3. Сохраните следующий скрипт для переноса пользователей в файл `copy_user.sql`

```
set term ^;

EXECUTE BLOCK
AS
  -- замените на параметры вашей копии БД безопасности
  DECLARE SRC_SEC_DB     VARCHAR(255) = 'd:\fb50_data\security_30.fdb';
  DECLARE SRC_SEC_USER   VARCHAR(63) = 'SYSDBA';
  ------------------------------------------------
```

```
  DECLARE PLG$USER_NAME  SEC$USER_NAME;
  DECLARE PLG$VERIFIER   VARCHAR(128) CHARACTER SET OCTETS;
  DECLARE PLG$SALT       VARCHAR(32) CHARACTER SET OCTETS;
  DECLARE PLG$COMMENT    BLOB SUB_TYPE TEXT CHARACTER SET UTF8;
  DECLARE PLG$FIRST      SEC$NAME_PART;
  DECLARE PLG$MIDDLE     SEC$NAME_PART;
  DECLARE PLG$LAST       SEC$NAME_PART;
  DECLARE PLG$ATTRIBUTES BLOB SUB_TYPE TEXT CHARACTER SET UTF8;
  DECLARE PLG$ACTIVE     BOOLEAN;
  DECLARE PLG$GROUP_NAME SEC$USER_NAME;
  DECLARE PLG$UID        PLG$ID;
  DECLARE PLG$GID        PLG$ID;
  DECLARE PLG$PASSWD     PLG$PASSWD;
BEGIN
  -- перемещаем пользователей из плагина SRP
  FOR EXECUTE STATEMENT Q'!
      SELECT
          PLG$USER_NAME,
          PLG$VERIFIER,
          PLG$SALT,
          PLG$COMMENT,
          PLG$FIRST,
          PLG$MIDDLE,
          PLG$LAST,
          PLG$ATTRIBUTES,
          PLG$ACTIVE
      FROM PLG$SRP
      WHERE PLG$USER_NAME <> 'SYSDBA'
!'
          ON EXTERNAL :SRC_SEC_DB
          AS USER :SRC_SEC_USER
          INTO :PLG$USER_NAME,
               :PLG$VERIFIER,
               :PLG$SALT,
               :PLG$COMMENT,
               :PLG$FIRST,
               :PLG$MIDDLE,
               :PLG$LAST,
               :PLG$ATTRIBUTES,
               :PLG$ACTIVE
  DO
  BEGIN
    INSERT INTO PLG$SRP (
        PLG$USER_NAME,
        PLG$VERIFIER,
        PLG$SALT,
        PLG$COMMENT,
        PLG$FIRST,
        PLG$MIDDLE,
        PLG$LAST,
        PLG$ATTRIBUTES,
```

```
        PLG$ACTIVE)
    VALUES (
        :PLG$USER_NAME,
        :PLG$VERIFIER,
        :PLG$SALT,
        :PLG$COMMENT,
        :PLG$FIRST,
        :PLG$MIDDLE,
        :PLG$LAST,
        :PLG$ATTRIBUTES,
        :PLG$ACTIVE);
  END
  -- перемещаем пользователей из плагина Legacy_UserManager
  FOR EXECUTE STATEMENT Q'!
      SELECT
          PLG$USER_NAME,
          PLG$GROUP_NAME,
          PLG$UID,
          PLG$GID,
          PLG$PASSWD,
          PLG$COMMENT,
          PLG$FIRST_NAME,
          PLG$MIDDLE_NAME,
          PLG$LAST_NAME
      FROM PLG$USERS
      WHERE PLG$USER_NAME <> 'SYSDBA'
!'
          ON EXTERNAL :SRC_SEC_DB
          AS USER :SRC_SEC_USER
          INTO :PLG$USER_NAME,
               :PLG$GROUP_NAME,
               :PLG$UID,
               :PLG$GID,
               :PLG$PASSWD,
               :PLG$COMMENT,
               :PLG$FIRST,
               :PLG$MIDDLE,
               :PLG$LAST
  DO
  BEGIN
    INSERT INTO PLG$USERS (
        PLG$USER_NAME,
        PLG$GROUP_NAME,
        PLG$UID,
        PLG$GID,
        PLG$PASSWD,
        PLG$COMMENT,
        PLG$FIRST_NAME,
        PLG$MIDDLE_NAME,
        PLG$LAST_NAME)
    VALUES (
```

```
        :PLG$USER_NAME,
        :PLG$GROUP_NAME,
        :PLG$UID,
        :PLG$GID,
        :PLG$PASSWD,
        :PLG$COMMENT,
        :PLG$FIRST,
        :PLG$MIDDLE,
        :PLG$LAST);
  END
END^

set term ;^

commit;

exit;
```

Важно

Не забудьте заменить значение переменной `SRC_SEC_DB` на путь к копии вашей БД безопасности.

Замечание

Мы исключили копию пользователя SYSDBA, поскольку инициализировали его при установке.

4. Выполните скрипт на Firebird 5.0 подключившись к БД безопасности в embedded режиме

```
c:\Firebird\5.0>isql -i "d:\fb50_data\copy_users.sql" -u SYSDBA -ch UTF8
security.db
```

Поздравляем! Ваши пользователи перенесены с сохранением всех атрибутов и паролей.

4.3. Перенос списка пользователей из Firebird 2.5

Перенос пользователей из Firebird 2.5 более сложен. В Firebird 3.0 ввели новый способ аутентификации SRP - Secure Remote Password Protocol. Старый способ аутентификации также доступен, но выключен по умолчанию поскольку считается недостаточно безопасным. В Release Notes 3.0 описан способ переноса пользователей из Legacy_UserManager в SRP, однако в этом случае вы не сможете подключаться через fbclient версии 2.5. Кроме того, перенести пароли из Legacy_UserManager в SRP невозможно. Предлагаемый скрипт перенесёт список пользователей, но будут сгенерированы случайные пароли. Если вы хотите восстановить прежние пароли, то это придётся делать вручную. Я написал альтернативный скрипт, который позволяет перенести пользователей из `security2.fdb` в `security5.fdb` в плагин Legacy_UserManager. Здесь я опишу оба варианта.

4.3.1. Копирование списка пользователей в плагин SRP

Из-за новой модели аутентификации в Firebird 3 обновление базы данных безопасности версии 2.5 (security2.fdb) напрямую для использования в Firebird 5 невозможно. Однако существует процедура обновления, позволяющая сохранить данные учетной записи пользователя — имя пользователя, имя и другие аттрибуты, но не пароли — из базы данных security2.fdb, которая использовалась на серверах версии 2.x.

Процедура требует запуска сценария `security_database.sql`, который находится в каталоге `misc/upgrade` вашей установки Firebird 3. Эти инструкции предполагают, что у вас есть временная копия этого сценария в том же каталоге, что и исполняемый файл isql.

Замечание

- В Firebird 5.0 файл сценария обновления БД безопасности `security_database.sql` отсутствует в каталоге `misc/upgrade`, поэтому вам необходимо скачать zip архив с дистрибутивом Firebird 3.0.
- В приведенных ниже командах замените *masterkey* фактическим паролем SYSDBA для вашего сервера, если это необходимо.

1. Сделайте резервную копию БД безопасности `security2.fdb` на Firebird 2.5

```
c:\Firebird\2.5>bin\gbak -b -g -user SYSDBA -password masterkey -se service_mgr
c:\Firebird\2.5\security2.fdb d:
\fb25_backup\security2.fbk
```

2. Разверните резервную копию на Firebird 5.0

```
c:\Firebird\5.0>gbak -c -user SYSDBA -password masterkey -se
localhost/3054:service_mgr d:\fbdata\5.0\security2.fbk d:\f
bdata\5.0\security2db.fdb -v
```

3. На сервере Firebird 5.0 перейдите в каталог, в котором находится утилита isql, и запустите сценарий обновления:

```
isql -user sysdba -pas masterkey -i security_database.sql
{host/path}security2db.fdb
```

   `security2db.fdb` - это просто пример имени базы данных: это может быть любое предпочтительное имя.

4. Процедура генерирует новые случайные пароли и затем выводит их на экран. Скопируйте вывод и уведомите пользователей об их новых паролях.

4.3.2. Копирование списка пользователей в плагин Legacy_UserManager

В отличие от предыдущего варианта, данный скрипт сохранит ваши исходные пароли. Однако, мы советуем вам в будущем всё равно перейти на плагин Srp.

1. Сделайте резервную копию БД безопасности `security2.fdb` на Firebird 2.5

```
c:\Firebird\2.5>bin\gbak -b -g -user SYSDBA -password masterkey -se service_mgr
c:\Firebird\2.5\security2.fdb d:
\fb25_backup\security2.fbk
```

2. Разверните резервную копию на Firebird 5.0

```
c:\Firebird\5.0>gbak -c -user SYSDBA -password masterkey -se
localhost/3054:service_mgr d:\fbdata\5.0\security2.fbk d:\f
bdata\5.0\security2db.fdb -v
```

3. Сохраните следующий скрипт для переноса пользователей в файл `copy_security2.sql`

```
set term ^;

EXECUTE BLOCK
AS
  -- замените на параметры вашей копии БД безопасности
  DECLARE SRC_SEC_DB     VARCHAR(255) = 'd:\fbdata\5.0\security2.fdb';
  DECLARE SRC_SEC_USER   VARCHAR(63) = 'SYSDBA';
  ------------------------------------------------
  DECLARE PLG$USER_NAME  SEC$USER_NAME;
  DECLARE PLG$COMMENT    BLOB SUB_TYPE TEXT CHARACTER SET UTF8;
  DECLARE PLG$FIRST      SEC$NAME_PART;
  DECLARE PLG$MIDDLE     SEC$NAME_PART;
  DECLARE PLG$LAST       SEC$NAME_PART;
  DECLARE PLG$GROUP_NAME SEC$USER_NAME;
  DECLARE PLG$UID        INT;
  DECLARE PLG$GID        INT;
  DECLARE PLG$PASSWD     VARBINARY(64);
BEGIN
  FOR EXECUTE STATEMENT q'!
      SELECT
          RDB$USER_NAME,
          RDB$GROUP_NAME,
          RDB$UID,
          RDB$GID,
          RDB$PASSWD,
          RDB$COMMENT,
          RDB$FIRST_NAME,
          RDB$MIDDLE_NAME,
          RDB$LAST_NAME
```

```
      FROM RDB$USERS
      WHERE RDB$USER_NAME <> 'SYSDBA'
!'
      ON EXTERNAL :SRC_SEC_DB
      AS USER :SRC_SEC_USER
      INTO
          :PLG$USER_NAME,
          :PLG$GROUP_NAME,
          :PLG$UID,
          :PLG$GID,
          :PLG$PASSWD,
          :PLG$COMMENT,
          :PLG$FIRST,
          :PLG$MIDDLE,
          :PLG$LAST
  DO
  BEGIN
    INSERT INTO PLG$USERS (
        PLG$USER_NAME,
        PLG$GROUP_NAME,
        PLG$UID,
        PLG$GID,
        PLG$PASSWD,
        PLG$COMMENT,
        PLG$FIRST_NAME,
        PLG$MIDDLE_NAME,
        PLG$LAST_NAME)
    VALUES (
        :PLG$USER_NAME,
        :PLG$GROUP_NAME,
        :PLG$UID,
        :PLG$GID,
        :PLG$PASSWD,
        :PLG$COMMENT,
        :PLG$FIRST,
        :PLG$MIDDLE,
        :PLG$LAST);
  END
END^

set term ;^

commit;

exit;
```

Важно

Не забудьте заменить значение переменной `SRC_SEC_DB` на путь к копии вашей БД безопасности.

Замечание

Мы исключили копию пользователя SYSDBA, поскольку инициализировали его при установке.

4. Выполните скрипт на Firebird 5.0 подключившись к БД безопасности в embedded режиме

```
c:\Firebird\5.0>isql -i "d:\fb40_data\copy_security2.sql" -u SYSDBA -ch UTF8
security.db
```

Поздравляем! Ваши пользователи перенесены с сохранением всех атрибутов и паролей.

Глава 5. Настройка доверительной аутентификации

Настройка доверительной аутентификации (trusted authentification) в Firebird 5.0 делается точно так же как она делалась в Firebird 3.0 или 4.0. Для тех производит миграцию с Firebird 2.5 опишем этот процесс подробнее.

1. Первым делом необходимо подключить плагин доверительной аутентификации в файле конфигурации `firebird.conf` или `databases.conf` в параметре *AuthServer* (по умолчанию он отключен). Для этого необходимо добавить плагин с именем Win_Sspi, и будем использовать его совместно с Srp256.

   ```
   AuthServer = Srp256, Win_Sspi
   ```

2. Следующим шагом необходимо включить отображение пользователей из Win_Sspi на `CURRENT_USER`. Для этого необходимо создать отображение в целевой базе данных с помощью следующего запроса

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

   Данный SQL запрос создаёт отображение только на уровне текущей базе данных. Отображение не будет применяться к другим базам данных расположенных на том же сервере. Если вы хотите создать общее отображение для всех баз данных, то добавьте ключевое слово GLOBAL.

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

3. Включение SYSDBA-подобного доступа для администраторов Windows (если он нужен).

   Для включения такого доступа необходимо создать следующее отображение

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

   Вместо включения SYSDBA-подобного доступа для всех администраторов Windows, вы

можете дать административные привилегии конкретному пользователю с помощью следующего отображения

```
create global mapping cto_sysdba
using plugin win_sspi
from user "STATION9\DEVELOPER"
to user SYSDBA;
```

Глава 6. Несовместимости на уровне приложения

На уровне API клиентская библиотека fbclient 5.0 совместима с предыдущими версиями. Однако могут возникнуть проблемы совместимости на уровне некоторых SQL запросов. Большинство из них мы уже описывали ранее в разделе Список несовместимостей на уровне языка SQL. Далее опишем некоторые другие проблемы, которые могут возникнуть в приложении.

6.1. Удалён сетевой протокол WNET

Сетевой протокол WNET (он же Named Pipes, он же NetBEUI), ранее поддерживаемый на платформе Windows, удален в Firebird 5.0.

Те пользователи Windows, которые работали с любой строкой подключения WNET (`\\server\dbname` или `wnet://server/dbname`), должны вместо этого переключиться на протокол INET (TCP) (строка подключения `server:dbname`, `server/port:dbname`, `inet://server/dbname` или `inet://server:port/dbname`).

6.2. Новые типы данных

Актуально: при миграции с Firebird версий 2.5, 3.0.

Как уже говорилось ранее, некоторые выражения могут возвращать новые типы данных, которые не могут быть интерпретированы вашим приложением без его доработки. Такая доработка может занять существенное время или оказаться вам не по силам. Для упрощения миграции на новые версии вы можете установить параметр *DataTypeCompatibility* в режим совместимости с необходимой версией в `firebird.conf` или `databases.conf`.

```
DataTypeCompatibility = 3.0
```

или

```
DataTypeCompatibility = 2.5
```

Это самый быстрый путь добиться совместимости с новыми типами данных. Однако со временем вы можете начать внедрять поддержку новых типов в своё приложение. Естественно, это будет происходить постепенно - сначала один тип, потом другой и так далее. В этом случае вам надо настроить отображение тех типов, поддержку которых вы ещё не доделали, на другие типы данных. Для этого используется оператор `SET BIND OF`.

Синтаксис

```
SET BIND OF { <type-from> | TIME ZONE } TO { <type-to> | LEGACY | NATIVE | EXTENDED }
```

Ключевое слово `LEGACY` в части `TO` используется, когда тип данных, отсутствующий в предыдущей версии Firebird, должен быть представлен способом понятным для старого клиентского программного обеспечения (возможна некоторая потеря данных). Существуют следующие преобразования в `LEGACY` типы:

Таблица 1. Преобразования в legacy типы

DataTypeCompatibility	Native тип	Legacy тип
2.5	BOOLEAN	CHAR(5)
2.5 или 3.0	DECFLOAT	DOUBLE PRECISION
2.5 или 3.0	INT128	BIGINT
2.5 или 3.0	TIME WITH TIME ZONE	TIME WITHOUT TIME ZONE
2.5 или 3.0	TIMESTAMP WITH TIME ZONE	TIMESTAMP WITHOUT TIME ZONE

При установке параметра `DataTypeCompatibility` выполняется преобразование новых типов данных в legacy типы согласно таблице описанной выше.

Подробное описание этого оператора есть в "Firebird 4.0 Release Notes" и "Руководство по языку SQL СУБД Firebird 5.0". С помощью него вы можете управлять отображением новых типов в вашем приложении выполнив соответствующий запрос сразу после подключения, и даже написать `AFTER CONNECT` триггер в котором использовать несколько таких операторов.

Например, предположим, что вы добавили в ваше приложение поддержку даты и времени с часовыми поясами, но у вас до сих пор не поддерживаются типы `INT128` и `DECFLOAT`. В этом случае вы можете написать следующий триггер.

```
create or alter trigger tr_ac_set_bind
on connect
as
begin
  set bind of int128 to legacy;
  set bind of decfloat to legacy;
end
```

6.3. Согласованное чтение в транзакциях READ COMMITTED

Актуально: при миграции с Firebird версий 2.5, 3.0.

Firebird 4 не только вводит согласованность чтения (`READ CONSISTENCY`) для запросов в

транзакциях READ COMMITTED, но также делает его режимом по умолчанию для всех транзакций READ COMMITTED, независимо от их свойств RECORD VERSION или NO RECORD VERSION.

Это сделано для того, чтобы обеспечить пользователям лучшее поведение — как соответствующее спецификации SQL, так и менее подверженное конфликтам. Однако это новое поведение может также иметь неожиданные побочные эффекты.

Пожалуй самый важный из них это так называемые рестарты при обработке конфликтов обновления. Это может привести к тому, что некоторый код, не подверженный транзакционному контролю, может выполняться многократно в рамках PSQL. Примерами такого кода может быть:

- использование внешних таблиц, последовательностей или контекстных переменных;
- отправка электронных писем с использованием UDF;
- использование автономных транзакций или внешних запросов.

В режиме изолированности READ COMMITTED READ CONSISTENCY конфликт обновлений обрабатывается иначе. Если при выполнении UPDATE или DELETE обнаруживается запись, которая уже изменена или удалена другой транзакцией (транзакция подтверждена), то все изменения выполненные в текущем запросе откатываются и он выполняется заново. Это называется рестартом запроса.

Подробнее о согласованном чтении в транзакциях READ COMMITTED вы можете прочитать "Firebird 4.0 Release Notes".

Другим важным эффектом является то, что недофетченные курсоры в транзакциях READ COMMITTED READ CONSISTENCY в Read Only режиме теперь удерживают сборку мусора. Рекомендуем вам отказаться от использования в приложении единой длинной READ COMMITTED READ ONLY транзакции, и заменить её на несколько таких транзакций, каждая из которых активна ровно столько времени сколько это необходимо.

Если особенности режима READ CONSISTENCY по каким-либо причинам нежелательны, то чтобы вернуть устаревшее поведение, необходимо установить параметр конфигурации ReadConsistency равным 0.

6.4. Изменения в оптимизаторе

Оптимизизатор меняется в каждой версии Firebird. В основном эти изменения положительные, то есть ваши запросы должны работать быстрее, но часть запросов может замедлиться, поэтому необходимо тестировать производительность вашего приложения, и если где-то произошло замедление, то необходимо вмешательство со стороны программиста.

Для большинства изменений оптимизатора вы не можете повлиять на план запроса, изменяя конфигурацию сервера. В этом случае, вы можете сделать следующее:

- переписать SQL запрос так, чтобы он работал быстрее на новой версии сервера;

- создать или удалить индексы;
- если ничего из выше перечисленного не помогло, то создайте тикет о регрессии по адресу https://github.com/FirebirdSQL/firebird/issues.

Есть пару моментов в работе оптимизатора, на которые можно повлиять с помощью изменения конфигурации:

6.4.1. Использование Refetch при сортировке широких наборов данных

Актуально: при миграции с Firebird версий 2.5, 3.0.

Начиная с Firebird 4.0 появился новый метод доступа `Refetch`, который позволяет оптимизировать сортировку широких наборов данных. Под широким набором данных понимается набор данных в котором суммарная длина полей записи велика.

Исторически сложилось так, что при выполнении внешней сортировки Firebird записывает как ключевые поля (то есть, которые указаны в предложении `ORDER BY` или `GROUP BY`), так и неключевые поля (все остальные поля, на которые имеются ссылки внутри запроса) в блоки сортировки, которые либо сохраняются в памяти, либо во временные файлы. После завершения сортировки эти поля считываются обратно из блоков сортировки. Обычно этот подход считается более быстрым, поскольку записи считываются из временных файлов в порядке соответстветствующему отсортированным записям, а не выбираются случайным образом со страницы данных. Однако если неключевые поля большие (например, используются длинные VARCHAR), то это увеличивает размер блоков сортировки и, таким образом, приводит к большему количеству операций ввода-вывода для временных файлов. Firebird 4 предлагает альтернативный подход (метод доступа `Refetch`), когда внутри блоков сортировки хранятся только ключевые поля и записи `DBKEY`, а неключевые поля извлекаются из страниц данных после сортировки. Это повышает производительность сортировки в случае длинных неключевых полей.

Таким образом планы ваших запросов, использующих сортировку могут поменятся. Для управления данным методом доступа введён новый параметр конфигурации `InlineSortThreshold`. Значение, указанное для `InlineSortThreshold`, определяет максимальный размер записи сортировки (в байтах), которая может храниться встроенно, то есть внутри блока сортировки. Ноль означает, что записи всегда перезагружаются. Оптимальное значение данного параметра необходимо подбирать экспериментальным путём. Значение по умолчанию равно 1000 байт.

Рассмотрим следующий пример:

```
SELECT
  field_1, field_2, field_3, field_4
FROM SomeTable
ORDER BY field_1
```

До Firebird 4.0 в блоки сортировки всегда были включены все 4 поля. Начиная с Firebird 4.0, если суммарная длина полей `field_1 .. field_4` превышает значение `InlineSortThreshold`, то в блоки сортировки попадёт только `field_1`, а затем будет выполнен `Refetch`.

6.4.2. Преобразование OUTER JOINs в INNER JOINs

Существует ряд проблем с оптимизацией OUTER JOINs в Firebird.

Во-первых, в настоящее время OUTER JOIN может быть выполнен только одним алгоритмом соединения NESTED LOOP JOIN, что может быть изменено в следующих версиях.

Во-вторых, при соединении потоков внешними соединениями порядок соединения строго фиксирован, то есть, оптимизатор не может изменить его, чтобы результат оставался правильным.

Однако, если в условии `WHERE` существует предикат для поля "правой" (присоединяемой) таблицы, который явно не обрабатывает значение `NULL`, то во внешнем соединении нет смысла. В этом случае начиная с Firebird 5.0 такое соединение будет преобразовано во внутреннее, что позволяет оптимизатору применять весь спектр доступных алгоритмов соединения.

Допустим у вас есть следующий запрос:

```
SELECT
  COUNT(*)
FROM
  HORSE
  LEFT JOIN FARM ON FARM.CODE_FARM = HORSE.CODE_FARM
WHERE FARM.CODE_COUNTRY = 1
```

В Firebird 5.0 такой запрос неявно будет преобразован в эквивалентную форму:

```
SELECT
  COUNT(*)
FROM
  HORSE
  JOIN FARM ON FARM.CODE_FARM = HORSE.CODE_FARM
WHERE FARM.CODE_COUNTRY = 1
```

Если `LEFT JOIN` использовался в качестве подсказки для указания порядка соединения очень активно, то переписать множество запросов на новый лад может быть проблематично. Для таких разработчиков существует параметр конфигурации `OuterJoinConversion` в `firebird.conf` или `database.conf`. Установка параметра `OuterJoinConversion` в `false` отключает трансформацию Outer Join во внутренние соединение. Отметим, что этот параметр является временным решением для облегчения миграции и, в будущих версиях Firebird он может быть удалён.

6.5. RETURNING, возвращающий множество записей

Начиная с Firebird 5.0 клиентские модифицирующие операторы `INSERT .. SELECT`, `UPDATE`, `DELETE`, `UPDATE OR INSERT` и `MERGE`, содержащие предложение `RETURNING` возвращают курсор, то

есть они способны вернуть множество записей вместо выдачи ошибки "multiple rows in singleton select", как это происходило ранее.

Теперь эти запросы во время подготовки описываются как `isc_info_sql_stmt_select`, тогда как в предыдущих версии они были описаны как `isc_info_sql_stmt_exec_procedure`.

Singleton-операторы `INSERT .. VALUES`, а также позиционированные операторы `UPDATE` и `DELETE` (то есть, которые содержат предложение `WHERE CURRENT OF`) сохраняют существующее поведение и описываются как `isc_info_sql_stmt_exec_procedure`.

Однако все эти запросы, если они используются в PSQL и применяется предложение `RETURNING`, по-прежнему рассматриваются как сингелтоны.

Если ваше приложение использует модифицирующие операторы `INSERT .. SELECT`, `UPDATE`, `DELETE`, `UPDATE OR INSERT` и `MERGE`, содержащие предложение `RETURNING`, то это может быть причиной возникновения ошибок. Убедитесь, что ваш драйвер или компонент доступа правильно обрабатывает подобные запросы, и если это не так, то либо модифицируйте код (приложения или компонента), либо дождитесь пока выйдет обновление соотвествующего драйвера/компонента правильно обрабатывающего данные запросы.

Примеры модифицирующих операторов содержащих RETURNING, и возвращающих набор данных:

```
INSERT INTO dest(name, val)
SELECT desc, num + 1 FROM src WHERE id_parent = 5
RETURNING id, name, val;

UPDATE dest
SET a = a + 1
WHERE id = ?
RETURNING id, a;

DELETE FROM dest
WHERE price < 0.52
RETURNING id;

MERGE INTO PRODUCT_INVENTORY AS TARGET
USING (
  SELECT
    SL.ID_PRODUCT,
    SUM(SL.QUANTITY)
  FROM
    SALES_ORDER_LINE SL
    JOIN SALES_ORDER S ON S.ID = SL.ID_SALES_ORDER
  WHERE S.BYDATE = CURRENT_DATE
    AND SL.ID_PRODUCT = :ID_PRODUCT
  GROUP BY 1
) AS SRC(ID_PRODUCT, QUANTITY)
ON TARGET.ID_PRODUCT = SRC.ID_PRODUCT
WHEN MATCHED AND TARGET.QUANTITY - SRC.QUANTITY <= 0 THEN
  DELETE
WHEN MATCHED THEN
  UPDATE SET
    TARGET.QUANTITY = TARGET.QUANTITY - SRC.QUANTITY,
    TARGET.BYDATE = CURRENT_DATE
RETURNING OLD.QUANTITY, NEW.QUANTITY, SRC.QUANTITY;
```

Глава 7. Заключение

В этой статье я постарался описать наиболее часто встречающиеся проблемы и их решения при миграции на Firebird 5.0 с Firebird 2.5, 3.0 и 4.0. Надеюсь, что данная статья поможет вам перевести ваши базы данных и приложения на Firebird 5.0 и воспользоваться всеми преимуществами новой версии.