Ред База Данных

Версия 3.0

Руководство администратора

46.29926343.502120-01 92 1

Содержание

Глава 1

Общие сведения о программе

1.1 Назначение программы

СУБД Ред База Данных (далее СУБД или Ред База Данных) используется для упорядоченного хранения и обработки больших объемов информации. Ред База Данных представляет собой мощную современную СУБД с открытым кодом. Ядро Ред Базы Данных построено на основе одной из самых известных и распространенных в мире СУБД с открытым кодом – Firebird, которая используется в решениях различного масштаба: от встроенных аппаратных систем и решений для небольших компаний до IT- систем крупнейших транснациональных корпораций с размерами баз данных до десятков терабайт и десятками миллионов транзакций в день.

1.2 Функциональное назначение

СУБД Ред База Данных предоставляет пользователям следующие возможности:

- поддержка 64-битных систем;
- поддержка многопроцессорных и многоядерных аппаратных платформ;
- высокое быстродействие;
- возможность хранения базы данных в одном отдельном файле;
- возможность аутентификации и авторизации пользователей с использованием в качестве источников сведений об учетных записях пользователей защищенной БД пользователей или системного каталога;
- возможность «горячего» резервного копирования БД и инкрементного резервного копирования, в т.ч. с применением аппаратных решений для резервного копирования;
- наличие модулей сопряжения практически для всех используемых сред разработки, результатов тестов этих модулей и гарантия стабильной работы;
- возможность работы во «встроенном» в ПО (embedded) локальном режиме в виде библиотеки DLL без отдельной установки и настройки СУБД Ред База Данных, в том числе поддержка встраивания в виртуальную машину Java;
- возможность одновременной модификацию базы данных несколькими пользователями;
- обратная совместимость с базами данных от предыдущих версий Firebird;
- многоверсионная архитектура;
- модульная архитектура;
- соответствие большинству требований стандарта ISO/ANSI SQL;
- низкие требования к аппаратному обеспечению для небольших баз данных;
- большие возможности по расширению функционала самой СУБД посредством модулей;
- ядро, изначально основанное на многоверсионной архитектуре (MGA);
- полное соответствие принципам атомарности, непротиворечивости, изоляции, долговечности (ACID).

1.3 Минимальный состав аппаратных средств

Для установки и нормальной работы СУБД Ред База Данных персональный компьютер должен быть оснащен комплектующими со следующими характеристиками:

- Процессор с поддержкой архитектуры X86_64, 1.6ГГц, 2 ядра.
- Оперативная память от 4Гб.
- Запоминающее устройство объемом не менее 21Гб.
- Клавиатура 101/102-х клавишная рус/лат.
- Сетевая карта с поддержкой Ethernet.

1.4 Минимальный состав программных средств

Для установки и эксплуатации СУБД в ОС семейства Windows не требуется установки дополнительного программного обеспечения.Для ОС семейства Linux необходимы библиотеки: glibc 2.12 и старше, libstdc++ от gcc 5.3 и старше, zlib 1.2.11 и старше, kerberos 1.10 и старше. Также необходима служба xinetd (eXtended InterNET Daemon) — служба с открытым исходным кодом, работающая во многих Unix-подобных системах и управляющая сетевыми соединениями.

1.5 Требования к персоналу, среде эксплуатации и внешним мерам безопасности

Администратор СУБД должен иметь:

- базовые навыки администрирования ОС семейства Windows илиLinux (в зависимости от выбранной архитектуры);
- навыки настройки программных продуктов и ОС;
- опыт работы с командной строкой ОС;
- базовое представление о структуре баз данных и минимальные навыки работы по управлению ими;
- навыки поддержания в работоспособном состоянии технических средств ПК.

Безопасная эксплуатация СУБД предполагает, что:

- должно быть обеспечено отсутствие на компьютере с установленной СУБД нештатных программных средств, позволяющих осуществить несанкционированную модификацию СУБД и администратор не предпринимает попыток модификации СУБД;
- установка, конфигурация и эксплуатация СУБД осуществляется администраторам согласно соответствующей документации;
- администратором предусмотрены мероприятия, направленные на восстановление безопасного состояния СУБД в случае сбоя (отказа);
- персонал, ответственный за администрирование СУБД, должен пройти проверку на благонадежность и компетентность;
- в своей деятельности администратор должен руководствоваться политикой безопасности организации;
- администраторы являются компетентными, хорошо обученными и заслуживающими доверия;
- имеется в наличии одно или более компетентное лицо, которое назначается для управления безопасностью СУБД и информации в нем. Эти лица должны иметь личную ответственность за следующие функции:

  - управление пользователями;
  - создание и сопровождение ролей;
  - установление и сопровождение отношений между ролями;
  - назначение и аннулирование ролей, назначаемых пользователям.

- доступ к СУБД должен осуществляться только из санкционированных точек доступа,размещенных в контролируемой зоне, оборудованной средствами и системами физической защиты и охраны (контроля и наблюдения) и исключающей возможность бесконтрольного пребывания посторонних лиц;
- обеспечено взаимодействие СУБД только с доверенными системами ИТ, правила безопасности которых скоординированы с правилами безопасности СУБД;
- аутентификация субъектов, осуществляющих попытку доступа к СУБД, должна осуществляться с использованием механизмов ОС, под управлением которой функционирует СУБД;
- функционирование СУБД должно осуществляться в среде функционирования (ОС), предоставляющей механизм аутентификации, обеспечивающий адекватную защиту от прямого или умышленного нарушения безопасности нарушителями с высоким потенциалом нападения;
- все сетевые компоненты (такие, как мосты и маршрутизаторы) передают данные правильно, без модификации;
- загрузка СУБД проходит в доверенной среде, предотвращающей несанкционированное прерывание процесса загрузки СУБД и использование инструментальных средств, позволяющих осуществить доступ к защищаемым ресурсам СУБД в обход механизмов защиты;
- права пользователей для получения доступа и выполнения обработки информации основываются на одной или более ролях, которые им назначает администратор. Эти роли точно отражают производственную функцию, обязанности, квалификацию и/или компетентность пользователей в рамках предприятия.

1.6 Режим работы СУБД Ред База Данных

СУБД Ред База Данных поддерживает единственный режим работы — штатный.

Штатный режим работы включает в себя запуск службы сервера, которая обеспечивает доступ к базам данных СУБД. При запуске сервера Ред База Данных создает процесс, который слушает определенный порт (по умолчанию 3050) и ожидает запросов на подключение к базам данных. Когда клиентское приложение отправляет запрос на подключение к базе данных, сервер проверяет права доступа к базе данных и, если они корректны, устанавливает соединение с базой данных и начинает обрабатывать запросы.

В штатном режиме работы СУБД использует файлы баз данных, конфигурационные файлы и файлы журналов для хранения информации о событиях.

Штатный режим работы СУБД обеспечивает все необходимые механизмы безопасности для защиты данных в базе данных.

1.7 Лимиты СУБД Ред База Данных 3.0

Текущие лимиты СУБД Ред База Данных 3.0:

- Размер базы данных - ограничен 2^{32} страницами, то есть от 8ТБ до 64ТБ;
- Максимальное количество таблиц - 32000;
- Максимальный размер одной таблицы - 16КБ - 36ТБ;
- Максимальное число строк в одной таблице - 2^{40};
- Максимальный размер строки - 64КБ;

- Максимальное количество столбцов в таблице - зависит от типа данных;
- Максимальное число индексов на таблицу - около 850, зависит от размера страницы и от числа сегментов;
- Максимальное число индексов на базу данных - около 4 млн, но явного ограничения нет;
- Максимальная длина ключа индекса - 1/4 от размера страницы, то есть 1КБ - 4КБ;
- Максимальная длина запроса - 10МБ;
- Максимально допустимое число транзакций - 2^{48} (то есть 281474976710656).

Ограничения типов данных:

- Char и Varchar - 32767 байт;
- Smallint - 16 бит;
- Integer - 32 бит;
- BigInt - 64 бит;
- Float - 32 бит, от $3.4 * 10^{-38}$ до $3.4 * 10^{38}$;
- Double - 64 бит, $1.7 * 10^{-308}$ до $1.7 * 10^{308}$;
- Timestamp - 64 бит, 1 января 100 до 28 февраля 32768;
- Date - 32 бит, 1 января 100 до 29 февраля 32768;
- Time - 32 бит, 00:00 до 23:59.9999;
- Blob - 32GB;
- Numeric и Decimal - хранятся как smallint, integer или bigint в зависимости от размера, могут достигать 18 знаков.

1.8 Программные интерфейсы

C/C++

Поскольку СУБД написана на `C++`, для работы с ней на этом языке не требуется драйвер, вызовы функций `API` на `C/C++` можно выполнять напрямую.

Желательно использовать для работы с сервером клиентскую библиотеку `fbclient`, которая облегчает взаимодействие с сервером.

СУБД поддерживает 2 версии `API`:

- устаревшую `ISC` с "плоским" стилем вызова функций, характерным для языка C.
- новую объектно-ориентированную `OO API`, построенную на основе интерфейсов.

Описание принципов работы с новой версией `API` и доступных в ней функций можно найти в документе "Система управления базами данных «Ред База Данных». Функциональная спецификация".

Java

`Jaybird` - драйвер построенный на стандарте `JDBC` для подключения к серверам баз данных Ред База Данных и `Firebird`. Драйвер `Jaybird` входит в дистрибутив СУБД и находится в каталоге `jar`, например: `jar/jaybird-jdk18-3.0.33.jar`.

Название библиотеки формируется следующим образом: `jaybird-<Версия JDK>-<Версия драйвера>.jar`. `<Версия JDK>` для `Jaybird 3.x.x` и `4.x.x` может принимать значение `jdk-18`. Для `5.x.x` - значения `jdk-18 и jdk-11`.

`Jaybird 5` требует `Java 8` или выше. `Jaybird 3 и 4` требуют `Java 7` или выше.

Драйверы `Jaybird 3.x.x`, `4.x.x` и `5.x.x` версий доступны для скачивания с Nexus Repository Manager.

Для подключения в maven проект необходимо добавить репозиторий в секцию <repositories> в конфигурацию проекта pom.xml:

```
<repositories>
  <repository>
    <id>nexus-red-soft-ru-jaybird</id>
    <url>http://nexus.red-soft.ru/repository/jaybird</url>
  </repository>
</repositories>
```

а затем в секцию <dependencies> добавить нужную версию Jaybird, например:

```
<dependencies>
  <dependency>
    <groupId>ru.red-soft.jdbc</groupId>
    <artifactId>jaybird-jdk18</artifactId>
    <version>5.0.6</version>
  </dependency>
</dependencies>
```

Для подключения в gradle проект необходимо добавить репозиторий в секцию repositories в конфигурацию проекта build.gradle:

```
repositories {
  maven {
    url 'http://nexus.red-soft.ru/repository/jaybird'
  }
}
```

В секции dependencies указать необходимую версию драйвера:

```
dependencies{
  implementation 'ru.red-soft.jdbc:jaybird-jdk18:5.0.6'
}
```

Python

Пакет firebird-driver является официальным драйвером Python для работы с СУБД Ред База Данных и Firebird.

Соответствует спецификации Python DB API 2.0.

В дополнение к минимальному функционалу Python DB API данный драйвер также предоставляет новый клиентский API, добавленный в Firebird или Ред Базу Данных и ряд дополнительных расширений и улучшений для удобного использования СУБД.

Поддерживает СУБД версии 3.0 и выше. Для работы требуется Python версии 3.8 и выше.

Firebird-driver распространяется в качестве пакета setuptools. Установить можно с помощью утилиты pip из каталога PYPI:

```
pip install firebird-driver
```

Драйвер также можно установить из исходного кода. Для этого необходимо клонировать проект из официального репозитория github:

```
git clone https://github.com/FirebirdSQL/python3-driver.git
```

И выполнить установку с помощью утилиты pip:

```
pip install -e python3-driver
```

Perl

DBD::Firebird (версия 1.34) - драйвер Perl для работы с СУБД Ред База Данных и Firebird. Драйвер работает с СУБД через модуль Perl DBI.

Поддерживает СУБД версии 2.5.1 и выше. Требования:

- Perl версии 5.8.1 и выше;
- Perl DBI версии 1.41 или выше

Установка драйвера может быть выполнена с помощью cpan:

```
- cpanm:

cpanm DBD::Firebird

- CPAN shell:
perl -MCPAN -e shell
install DBD::Firebird
```

Драйвер также можно собрать и установить из исходного кода, скачав его по ссылке.

.Net

.NET data provider написан на C# и предоставляет высокопроизводительную реализацию API-функций СУБД Ред База Данных и Firebird для работы драйвера нет необходимости использовать клиентскую библиотеку fbclient.

Драйвер доступен для скачивания по ссылке https://firebirdsql.org/en/net-provider/. Кроме того, он может быть установлен с помощью NuGet из .NET CLI:

```
dotnet add package FirebirdSql.Data.FirebirdClient
```

Или с помощью Package Manager:

```
NuGet\Install-Package FirebirdSql.Data.FirebirdClient
```

ODBC

ODBC-драйвер для СУБД Ред База Данных и Firebird. Является свободно распространяемым с открытым исходным кодом.

По ссылке https://firebirdsql.org/en/odbc-driver/ доступны 32-х и 64-х битные бинарные пакеты для Windows иLinux, а также архив с исходными кодами для самостоятельной сборки.

Для работы драйвера требуется клиентская библиотека СУБД (fbclient.dll в Windows,fbclient.so в Linux).

TCL

dbi_firebird - это реализация интерфейса dbi (общий интерфейс баз данных SQL для Tcl) для подключения к СУБД Ред База Данных и Firebird.

Для работы драйвера dbi_firebird в системе должен быть установлен TCL версии 8.5 и выше, а также модуль pkgtools. Драйвер и необходимый модуль доступны для загрузки по ссылке.

Установка драйвера производится копированием файлов в каталог TCL. Также можно выполнить сборку драйвера из исходных кодов, которые доступны по ссылке.

Сборка и установка осуществляется следующими командами:

```
./configure
make
make install
```

Для работы с Ред Базой Данных и Firebird из программного интерфейса необходимо выполнить импорт драйвера "package require dbi_firebird", который сделает доступной команду dbi_firebird. С помощью этой команды можно создавать объекты, которые могут подключаться к базе данных.

1.9 Максимальный уровень доступности

Доступность сервера СУБД измеряется в процентах и рассчитывается по формуле:

$$\texttt{Доступность сервера СУБД} = \frac{(\texttt{СВД - ВН})}{\texttt{СВД}} \times 100,$$

где СВД - согласованное время доступности сервера;

ВН - время недоступности сервера.

Время недоступности рассчитывается на основании зарегистрированных обращений пользователей СУБД в поддержку в период ее эксплуатации.

Расчетный показатель доступности составляет 99.91%. Для его достижения нужно руководствоваться аппаратными требованиями и рекомендациями по настройке СУБД, представленными в главе "Настройка производительности Ред Базы Данных".

1.10 Настройка функций безопасности среды функционирования средства

СУБД обеспечивает безопасность доступа пользователей к серверу по умолчанию с помощью идентификатора пользователя и пароля. Как и любой другой сервер базы данных, СУБД Ред База Данных использует соответствующие средства защиты физического, сетевого доступа и файловой системы.

Для обеспечения безопасной работы СУБД в ОС семейства Linux необходимо:

- Установить разрешения на файлы: все файлы и каталоги, используемые СУБД, должны иметь соответствующие разрешения. Владельцем файлов должен быть системный пользователь Firebird. Серверный процесс СУБД должен иметь полный доступ к чтению и записи файлов базы данных. Рекомендуемые разрешения: 600 для файлов базы данных и 700 для каталогов. В процессе эксплуатации разрешения и права доступа рекомендуется устанавливать через делегирование прав пользователям ОС добавлением/исключением в группу Firebird хостовой ОС.
- Настроить брендмауэр: должны быть разрешены входящие соединения с сервером СУБД только из надежных источников. Порт, используемые СУБД по умолчанию (3050), должен быть открыт и доступен.
- Аутентифицировать пользователей: не используйте для подключения к серверу СУБД учетную запись суперадминистратора root хостовой ОС. Каждый пользователь СУБД должен пройти успешную авторизацию в хостовой ОС с использованием встроенных механизмов ОС, а все его действия должны логироваться.

Для реализации функций многофакторнойаутентификации в СУБД «Ред База Данных» необходимо наличие криптопровайдера КриптоПро CSP 5.0 (исполнение 1-Base). Установка и настройка криптопровайдера производится согласно эксплуатационной документации производителя. Специальные настройки, необходимые для безопасной работы СУБД, отсутствуют.

Глава 2

Миграции

Переход с версии 2.6 на 3.0

1. Обновите версию Ред Базы Данных 2.6 до последней https://reddatabase.ru/downloads/rdb26/
2. Удостоверьтесь, что в базе данных нет записей, содержащих значение `NULL` в полях с флагом `NOT NULL`, с помощью, например, следующего скрипта:

```
execute block returns (table_name varchar(100), field_name varchar(100))
as
   declare variable v smallint;
begin
   for select rdb$relation_name from rdb$relations r
       where r.rdb$relation_type=0
   into table_name
   do
     for select rf.rdb$field_name from rdb$relation_fields rf
         where rf.rdb$relation_name=:table_name and rf.rdb$null_flag=1
     into field_name
     do begin
       for execute statement 'select null from rdb$database where exists
           (select null from ' || table_name || ' where '|| field_name ||
           ' is null or ' || field_name || ' || '''' = '''')'
       into v do suspend;
     end
end
```

3. Сделайте бэкап базы данных на версии 2.6:

```
gbak -user sysdba -pas masterkey -b {host/path}<имя базы даных> {host/path}<имя
backup>
```

4. Выполните резервное копирование базы данных безопасности (`security2.fdb`) под версией 2.6:

```
gbak -user sysdba -pas masterkey -b {host/path}security2.fdb security2.fbk
```

5. Скопируйте корневой каталог установки сервера.
6. Деинсталлируйте Ред Базу Данных 2.6.
7. Выполните установку новой Ред Базы Данных 3.0. Стоит отметить, что при установке Ред Базы Данных 3.0 не создается файл `SYSDBA.password`, в котором хранится информация о заданном во время установки СУБД пароле.
8. Настройте файл конфигурации `firebird.conf`.

> Обратите внимание, что некоторые параметры (например, `CompleteBooleanEvaluation`, `OldColumnNaming`, `OldSetClauseSemantics`, `UsePriorityScheduler`, `PrioritySwitchDelay`, `PriorityBoost`, `LegacyHash` и `LockGrantOrder`) больше не поддерживаются.

Вы можете изменить архитектуру сервера, скорректировав значение параметра `ServerMode`. В этом случае необходимо рассчитать значения параметра `DefaultDbCachePages` в соответствии с выбранной архитектурой. Для архитектуры суперсервер значение параметра рекомендуется рассчитать по формуле:

$$\texttt{DefaultDbCachePages} = \frac{\texttt{MemorySize}}{4 * \texttt{PageSize}},$$

где `MemorySize` - общий объём памяти;
`PageSize` - объём страницы.
Для архитектуры классик значение параметра рекомендуется рассчитать по формуле:

$$\texttt{DefaultDbCachePages} = \frac{\texttt{MemorySize}}{4 * \texttt{ConnNum} * \texttt{PageSize}},$$

где `MemorySize` - общий объём памяти;
`PageSize` - объём страницы;
`ConnNum` - предполагаемое максимальное количество соединений.
Далее нужно перезапустить сервер:

```
systemctl restart firebird
```

9. Настройте файл `databases.conf`.
   В версии Ред Базы Данных 3.0 файл `aliases.conf` был переименован в `databases.conf`. Формат записей не изменился, поэтому вы можете смело копировать содержимое файла `aliases.conf` в `databases.conf`. Однако теперь новый файл может содержать гораздо больше параметров конфигурации для настройки на уровне базы.
10. Подключение клиентов от 2.6 по умолчанию выключено в целях безопасности. Чтобы вернуть такую возможность, настройте Legacy-аутентификацию. Для этого в файле конфигурации `firebird.conf` выставить значения следующих параметров:

    ```
    UserManager = Legacy_UserManager
    WireCrypt = Enabled (Disabled, если не собираетесь использовать шифрование сессии)
    AuthServer = Legacy_Auth, Srp, Win_Sspi
    AuthClient = Legacy_Auth, Srp, Win_Sspi
    ```

    Сохраните изменения и перезапустите сервер.
    При смене системы аутентификации администратор `SYSDBA` теперь имеет пароль по умолчанию `masterkey`. Рекомендуем сразу изменить пароль `SYSDBA`.
11. Необходимая версия драйвера `jaybird` на клиенте должна быть не ниже 2.2.8.
12. Если бэкап базы данных содержит внешние функции и процедуры, написанные на Java:
    - Настройте параметры взаимодействия сервера «Ред База Данных» с виртуальной машиной Java с помощью конфигурационного файла `plugins.conf` (расположен в корневом каталоге установки сервера). В нем необходимо раскомментировать секции относящиеся к `fbjava` и указать путь к `JAVA_HOME`:

      ```
      Plugin = JAVA {
         Module = $(dir_plugins)/fbjava
         Config = JAVA_config
      }
      Config = JAVA_config {
         JavaHome = /usr/lib/jvm/java-openjdk
         SecurityDatabase = $(this)/java-security.fdb
         JvmArgsFile = $(this)/jvm.args
      ```

```
    JarDirs = $(this)/jar
}
```

- Перенесите классы, содержащие тела внешних процедур, функций и триггеров, в каталог `/jar` в виде jar-файла.
- Создайте текстовый файл с отображением (он понадобится для дальнейшего рестора). В версии 2.6 при объявлении внешней функции в предложении `EXTERNAL NAME` не указывались аргументы, в отличие от версии 3.0. Поэтому, если функции (процедуры) имеют аргументы, следует создать файл с отображением, где в первом столбце - имя функции(процедуры) в java, а во втором - типы аргументов:

  ```
  package.class.Function1 arg1,arg2,arg3
  package.class.Function2 arg1,arg2,arg3
  ```

  Пример файла с отображением:

  ```
  biz.redsoft.udf.ServerBlobReader.readBlob java.lang.String,long
  ```

13. Выполните восстановление базы данных из сделанного ранее бэкапа под новым сервером.

- Если бэкап содержал внешние функции и процедуры, написанные на Java, то при восстановлении утилитой `gbak` укажите опцию `-rdb_map <файл с отображением>`. В этом случае нужная java функция (процедура) будет подбираться на лету просто по имени.

  ```
  gbak -rdb_map /mvv/temp/db/map26_30 -c -v <path_to_fbk> <path_to_fdb> -user
  sysdba -password masterkey
  ```

  Если опция `-rdb_map` не задана, то при ресторе возникнет ошибка:

  ```
  gbak:restoring function READ_SERVER_BLOB
  gbak:Updating entry point "biz.redsoft.udf.ServerBlobReader.readBlob" ->
                            "biz.redsoft.udf.ServerBlobReader.readBlob()"
  gbak: restoring argument for function READ_SERVER_BLOB
  gbak: restoring argument for function READ_SERVER_BLOB
  gbak: restoring argument for function READ_SERVER_BLOB
  gbak:committing metadata
  gbak:ERROR:java.lang.NoSuchMethodException:biz.redsoft.udf.ServerBlobReader.readBlob()
  ...
  ```

- Если бэкап содержал настроенный полнотекстовый поиск, то, восстанавливая базу данных утилитой `gbak`, укажите флаг `-ig`:

  ```
  gbak -user sysdba -pass masterkey -ig -c <path_to_fbk> <path_to_fdb>
  ```

  Иначе возникнет ошибка:

  ```
  gbak: ERROR:Error while parsing trigger FTS$TRIG_1's BLR
  gbak: ERROR: org.firebirdsql.fbjava.impl.FbException: Unrecognized data
  type: 'void'
  ...
  ```

  Эта необходимость обусловлена проблемой совместимости версий, которая описана в одном из пунктов раздела «Проблемы совместимости 3.0 и 2.6». Как настроить полнотекстовый поиск в версии 3.0 будет описано ниже.

14. Восстановите резервную копию базы данных безопасности на новой версии сервера:

```
gbak -user sysdba -pas masterkey -c security2.fbk {host/path}security2db.fdb
```

15. Из-за новой модели аутентификации в версии 3.0 обновление базы данных безопасности версии 2.6 (`security2.fdb`), непосредственно для использования в новой версии, невозможно. Однако, имеет место процедура обновления, позволяющая сохранить данные учетных записей пользователей – логин, имя, фамилия и т.д, НО не пароли – из `security2.fdb`. Процедура реализована в скрипте `security_database.sql`, который находится в папке `misc/upgrade/security` корневого каталоге установки сервера 3.0.
    Запустите скрипт обновления БД безопасности с помощью isql:

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

    Процедура создаст новые случайные пароли и выведет их на экран. Уведомите пользователей о новых паролях.

16. Запустите скрипт восстановления паролей с помощью isql:

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

    Скрипт `recovery_legacy_passwords.sql` находится также в папке `/misc/upgrade/security`. Если этот скрипт отсутствует, возьмите его из сборки старше 3.0.3.107. Этот скрипт копирует пароли из `security2.fdb`, но только для `legacy`-аутентификации.

17. Для восстановления политик безопасности и паролей многофакторнойаутентификации необходимо (пункт можно пропустить, если в 2.6 не настроена многофакторнаяаутентификация):
    - Изменить `UserManager` в конфигурационном файле `firebird.conf`;
    - Добавить в `AuthServer` и `AuthClient` плагин аутентификации `Multifactor`:

      ```
      UserManager = Multifactor_Manager, Legacy_UserManager
      AuthServer = Multifactor, Legacy_Auth, Srp, Win_Sspi
      AuthClient = Multifactor, Legacy_Auth, Srp, Win_Sspi
      ```

    - Проинициализировать новую систему аутентификации, для этого достаточно выполнить создание системного пользователя, например:

      ```
      echo "CREATE USER SYSDBA PASSWORD \"masterkey\" USING PLUGIN
      Multifactor_Manager;" | /opt/RedDatabase/bin/isql -u sysdba security.db
      ```

      > Подробную информацию о системах аутентификации см. раздел 9.5.

    - Для восстановления учетных данных этой системы аутентификации необходимо повторно выполнить скрипт `security_database.sql` из каталога `/misc/upgrade/security`, а для восстановления паролей и политик безопасности запустить скрипт `recovery_mf_passwords.sql` из того же каталога. Если этот скрипт отсутствует, возьмите его из сборки старше 3.0.3.107.

18. Перекомпилируйте все процедуры, триггеры и view (опционально), чтобы убедиться в корректности миграции. Это можно сделать с помощью `RDB Expert`.

19. Настройте файл конфигурации аудита `fbtrace.conf` в новом формате.
    Старый формат:

    ```
    <database %[\\/](test|azk2|rulez).fdb>
       enabled true
    ```

```
   time_threshold 100
   log_statement_finish true
</database>
```

Новый формат:

```
database = %[\\/](test|azk2|rulez).fdb
{
   enabled = true
   time_threshold = 100
   log_statement_finish = true
}
```

20. Если в базе данных прошлой версии был настроен полнотекстовый поиск, то для его восстановления выполните следующую инструкцию:

- Настройте параметры взаимодействия сервера Ред Базы Данных с виртуальной машиной Java с помощью конфигурационного файла `plugins.conf`, который расположен в корневом каталоге установки сервера. В нем необходимо раскомментировать секции `Plugin=JAVA` и `Config=JAVA_config`. В секции `Config` необходимо задать параметр `JavaHome` равный местоположению движка Java, а также в параметр `JarDirs` добавить через точку с запятой значение `"$(this)/jar/fts"`.

```
Config = JAVA_config {
      JavaHome = /usr/lib/jvm/java-8-openjdk-amd64/jre
      SecurityDatabase = $(this)/java-security.fdb
      JvmArgsFile = $(this)/jvm.args
      JarDirs = $(this)/jar;$(this)/jar/fts
}
```

- Подключитесь к базе данных безопасности `java-security.fdb` и назначьте права доступа пользователям базы данных, использующим код Java.
  В каталоге установки сервера можно найти файл `misc/fts_permissions.sql` со скриптом по назначению прав доступа для работы `fbjava_lucene`. Его можно выполнить на базе данных `java-security`. Там собраны все (или почти все) необходимые права для работы с полнотекстовым поиском. Администратор может по своему усмотрению редактировать скрипт, менять название каталога для хранения индекса lucene или имя роли.

```
echo "input '/opt/RedDatabase/misc/fts_permissions.sql';" |
/opt/RedDatabase/bin/isql -u sysdba -p masterkey
localhost:/opt/RedDatabase/java-security.fdb
```

- В файле конфигурации `fbjava.yaml` укажите пути к Java-библиотекам, которые реализуют функции полнотекстового поиска:

```
classpath:
   - $(root)/jar/fts/*.jar
```

- Если для БД необходим отдельный каталог для хранения индексов, тогда в файле конфигурации `fbjava.yaml` добавьте в секцию `databases`, укажите БД в виде шаблона регулярного выражения, создайте секцию `options` с параметром `ftsDirectory`, в котором будет указан каталог:

```
databases:
    ".*/database.fdb":
```

```
        options:
           ftsDirectory:
              - /path/to/directory
```

- В `jvm.args` установите директорию для хранения индекса (по умолчанию `/tmp/RDBLuceneIndex/`), если в `fbjava.yaml` для БД не указан отдельный каталог.

```
-Dfts.directory=...
```

- По умолчанию в `fts.directory` для БД будет создан каталог (если в `fbjava.yaml` для БД не задан параметр `ftsDirectory`), имя которого эквивалентно GUID БД, в котором будут храниться индексы. Если необходимо отключить создание каталога (хранить все индексы в `fts.directory`), установите параметр `fts.disableGUID`:

```
-Dfts.disableGUID=true
```

- Также в `jvm.args` можно указать параметры для установки максимального размера индексируемого документа `fts.max_document_size` (по умолчанию установлен максимальный размер `2147483647`) и пропуска битых документов `fts.skip_corrupted` (по умолчанию выключен), например:

```
-Dfts.max_document_size=10000
-Dfts.skip_corrupted=true
```

- Перезапустите сервер.
- Подключитесь к БД через `isql` под системным пользователем и выполните скрипт `pre_update_fts.sql` для подготовки обновления FTS. В скрипте создается временная таблица с ключевыми параметрами индексов, и удаляются таблицы/процедуры/триггеры версии 2.6.

```
isql -user sysdba -pass masterkey -i <path_to pre_update_fts.sql>
<path_to_fdb>
```

Этот скрипт Вы не найдете в сборке сервера. Его содержимое представлено в приложении З «Производительность СУБД Ред База Данных 3.0».

- Выполните скрипт инициализации `fts.sql`, который находится в корневом каталоге сервера:

```
isql -u sysdba -p masterkey -i <path_to fts.sql> <path_to_fdb>
```

- Примените второй скрипт `post_update_fts.sql`, который создает индексы из временных таблиц и выполняет полную их переиндексацию:

```
isql -user sysdba -pass masterkey -i <path_to post_update_fts.sql>
<path_to_fdb>
```

Этот скрипт Вы не найдете в сборке сервера. Его содержимое представлено в приложении З «Производительность СУБД Ред База Данных 3.0».

Проблемы совместимости 3.0 и 2.6

1. Изменены методы аутентификации и их настройка:
   - *Версия 2.6:*

Способ аутентификации задается параметром `Authentication` в `firebird.conf`.

1.1 `native` - традиционная парольная;

1.2 `trusted` - доверенная (только в Windows);

1.3 `multifactor` - многофакторная(по ГОСТ паролю, сертификату, доверенному сертификату - задается политиками безопасности);

1.4 `gss` - доверенная в разных ОС, через Kerberos;

1.5 `krb5` - доверенная через Kerberos;

1.6 `mixed` - всё вышеперечисленное.

- *Версия 3.0:* Способ аутентификации задается параметрами `AuthClient` и `AuthServer` в `firebird.conf`.

1.1 `Legacy_Auth` - традиционная парольная (аналог `native` из 2.6);

1.2 `Srp` - новая парольная, более безопасная;

1.3 `Win_Sspi` - доверенная в Windows (аналог `trusted` из 2.6);

1.4 `Multifactor` - многофакторная;

1.5 `Gss` - доверенная в разных ОС, через Kerberos.

Аутентификация по протоколу LDAP и все настройки сохранились в версии 3.0. Изменился лишь порядок поиска пользователей:

- В 2.6 сервер сначала искал пользователя в `security2.fdb`, а если не получалось - в LDAP.
- В 3.0 наоборот: если LDAP настроен, то пользователь ищется сначала там, потом в `security3.fdb`. Кроме `sysdba`, он ищется сначала в `security3.fdb`.

2. Добавление нового пользователя посредством SQL в версии 3.0:

- Укажите плагин управления пользователями в параметре `UserManager: Srp, Legacy_UserManager, Multifactor_Manager`.

> В зависимости от выбранного плагина аутентификации, данные о пользователях хранятся в разных таблицах базы данных безопасности. Поэтому одноименные пользователи, созданные с помощью разных плагинов управления пользователями — это разные пользователи. Таким образом пользователя, созданного с помощью одного плагина управления пользователями, можно удалить или изменить, указав только тот же самый плагин.

- SQL синтаксис (неполный):

```
CREATE USER username PASSWORD 'password' [USING PLUGIN 'pluginname'];
```

Необязательное предложение `USING PLUGIN` позволяет явно указывать какой плагин управления пользователями будет использован. По умолчанию используется тот плагин, который был указан первым в списке параметра `UserManager` в файле конфигурации `firebird.conf`. Допустимыми являются только значения, перечисленные в параметре `UserManager`.

> Если предложение `USING PLUGIN` не указано, то при добавлении пользователя он сам добавляется во все плагины из списка параметра `DefaultUserManagers` (в том числе его атрибуты).

Пример настройки многофакторнойаутентификации, создания нового пользователя и подключения им к базе данных:

2.1 В конфигурационном файле `firebird.conf` выставите значения следующих параметров:

```
AuthServer = Multifactor, Srp
AuthClient = Multifactor, Srp
UserManager = Multifactor_Manager
CryptoPlugin = Crypto_API
```

2.2 Установите соединение с базой данных пользователем `sysdba`. Создайте нового пользователя `testuser`:

```
.\isql -u sysdba -p masterkey localhost:e:\database.fdb
create user testuser password 'pass';
```

При этом пользователь `sysdba` подключился к БД через `Srp` (не Multifactor) аутентификацию. Чтобы пользователь `sysdba` смог подключиться посредством многофакторнойаутентификации, создайте многофакторного`sysdba`:

```
create user sysdba password 'masterkey';
```

2.3 Подключитесь к БД новым пользователем:

```
.\isql -u testuser -p pass localhost:e:\database.fdb
```

Пользователь `testuser` прошел многофакторнуюпроверку.

3. Изменены правила кумулятивного действия ролей:
   - Если пользователь не указывает роль при подключении к серверу:
     - *в 2.6:* он получает права всех ролей, которые ему назначены;
     - *в 3.0:* он получает права только тех ролей, которые были грантованы пользователю с ключевым словом `DEFAULT` , т.е.

```
GRANT DEFAULT ADMIN_ROLE TO ADMIN_USER;
```

   - Если пользователь при подключении указал конкретную роль:
     - *в 2.6:* он получает только её привилегии;
     - *в 3.0:* он получает её привилегии и привилегии ролей, которые ему назначены с `DEFAULT`;
   - Изменение текущей роли:
     - *в 2.6:* невозможно без переподключения;
     - *в 3.0:* пользователь может с помощью оператора `SET ROLE` сменить роль, указанную при подключении. В этом случае привилегии пользователя `CURRENT_USER` будут складываться из привилегий роли, назначенной оператором `SET ROLE` и привилегии ролей, которые ему назначены с `DEFAULT`.
4. Системные таблицы `RDB$` теперь только для чтения. Операторы вставки, обновления или удаления для них будут отклонены.
5. В версии 3.0 вводится ряд новых ключевых и зарезервированных слов. Новые зарезервированные слова не должны использоваться в качестве идентификаторов.
6. Изменения в SQL синтаксисе:
   - Смешивание явных и неявных соединений не рекомендуется, но позволяется. Некоторые виды смешивания запрещены. Например, такой запрос вызовет ошибку `"Column does not belong to referenced table"`

```
SELECT *
FROM TA, TB
```

```
JOIN TC ON TA.COL1 = TC.COL1
WHERE TA.COL2 = TB.COL2
```

Это происходит потому, что явный JOIN не может видеть таблицу TA. Однако следующий запрос будет выполнен без ошибок, поскольку изоляция не нарушена.

```
SELECT *
FROM TA, TB
JOIN TC ON TB.COL1 = TC.COL1
WHERE TA.COL2 = TB.COL2
```

- Теперь все идентификаторы SQL могут иметь максимальную длину - 31 байт. В PSQL алиасы столбцов и таблиц, а также имена локальных переменных ограничены длиной в 31 байт. Имена пользователей рассматриваются как идентификаторы SQL и, соответственно, ограничены длиной в 31 байт.
- В Ред Базе Данных 3.0 имена пользователей подчиняются общему правилу наименования идентификаторов объектов метаданных. Таким образом, пользователь с именем "Alex" и с именем "ALEX" будут разными пользователями.
- Оператор DECLARE CURSOR теперь требует, чтобы все столбцы были именованы или имели алиас. Это же требование применяется к оператору FOR SELECT ... AS CURSOR <имя курсора> DO .... Например, такая процедура

```
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
```

вызовет ошибку:

```
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
```

- В Ред Базе Данных версии 3.0 изменился синтаксис объявления внешний функций и процедур, написанных на Java.
- В некоторые DDL операторы добавлено новое необязательное предложение SQL SECURITY, а именно:

```
CREATE TABLE <имя таблицы> (...) [SQL SECURITY {DEFINER | INVOKER}]
ALTER TABLE <имя таблицы> ... [{ALTER SQL SECURITY {DEFINER | INVOKER} |
                                                    DROP SQL SECURITY}]
CREATE [OR ALTER] FUNCTION <функция>...[SQL SECURITY {DEFINER|INVOKER}] AS...
CREATE [OR ALTER] PROCEDURE <процедура>..[SQL SECURITY{DEFINER|INVOKER}] AS..
CREATE [OR ALTER] TRIGGER <триггер>...[SQL SECURITY {DEFINER | INVOKER} |
                                                  DROP SQL SECURITY] [AS...]
CREATE [OR ALTER] PACKAGE <пакет> [SQL SECURITY {DEFINER | INVOKER}] AS ...
```

Оно определяет, в контексте какого пользователя будет проходить работа с объектом. Ключевое слово `INVOKER` (значение по умолчанию) указывает, что объект вызывается с правами текущего пользователя. Задание ключевого слова `DEFINER` означает, что объект вызывается с правами его владельца (создателя).
Значение по умолчанию на уровне всей базы данных можно изменить оператором `ALTER DATABASE SET DEFAULT SQL SECURITY`.
В версии 2.6 в операторах по созданию и изменению хранимых процедур ту же функциональность выполняло предложение `AUTHID {OWNER | CALLER}`. В версии 3.0 данный синтаксис допустим, но является устаревшим и не будет поддерживаться в следующих версиях СУБД.
Совместное использование предложений `SQL SECURITY` и `AUTHID` в DDL операторах над процедурами вызовет ошибку.
Более подробно о новом синтаксисе можно прочитать в Руководстве по SQL.

7. При добавлении ограничения `SET NOT NULL` полю таблицы с существующими данными, придется вручную исправлять базу данных, если имеются записи с `NULL`. При этом значение по умолчанию не применимо. Оно сработает только при добавлении нового поля с `NOT NULL` для старых записей.
8. Файл `firebird-<название_архитектуры>.socket` из папки `/opt/RedDatabase/misc`, в котором можно задать максимально число соединений, отсутствует в версии 3.0. По умолчанию, Ред База Данных 3.0, как и Firebird 3.0, слушает порты сама в любой архитектуре.
9. Протокол аутентификации Legacy начиная с версии 3.0 совместим с Firebird любых версий только частично. Возможно подключение клиентов Firebird к серверу Ред Базы Данных. Но подключение клиентов Ред Базы Данных 3.0 к серверам Firebird по протоколу Legacy невозможно.
10. В версии 2.6 объявить триггер, написанный на языке Java, нельзя. Поэтому полнотекстовый поиск использовал PSQL триггеры, которые вызывали Java функции. Эти функции были обязаны возвращать результат. В 3.0 после перехода на fbjava они были переписаны в настоящие триггеры на Java и потеряли свое возвращаемое значение. При восстановлении базы данных из 2.6 BLR пытается скомпилироваться, так, как будто это функция внешнего движка, но это уже невозможно, потому что она теперь возвращает `void`. Чтобы избежать конфликта восстановления необходимо использовать ключ `-ig` (утилиты `gbak`)- тогда восстановление будет продолжаться, несмотря на ошибки, но неисправный BLR-код будет представлять `NULL`.
11. Изменена настройка полнотекстового поиска. Инструкцию можно найти в Руководстве по SQL.
12. В связи с архитектурой плагинов некоторые таблицы в `security` создаются только при создании пользователя через соответствующий плагин:
    - `PLG$SRP`, `PLG$SRP_VIEW` создаются при создании пользователя плагином `SRP`.
    - `PLG$MF`, `PLG$MF_VIEW`, `PLG$POLICIES`, `PLG$PASSWD_HISTORY` создаются при создании пользователя плагином `Multifactor_Manager`.

Переход с версии 3.0 на 2.6

1. Прекратите работы с базой данных и сделайте её копию.
2. Выгрузите данные о пользователях:

```
isql -user SYSDBA -ch <кодировка> security.db -input
/opt/RedDatabase/misc/downgrade/security_to_26.sql >
/home/firebird/security_migration.sql
```

3. В конфигурационном файле `firebird.conf` установите параметр `UdfAccess = Full`.
4. Запустите миграцию:

```
rdbsvcmgr service_mgr -user SYSDBA -action_migrate - mig_version ods11.2
-dbname <база данных>
```

5. Выгрузите скрипты с несовместимыми объектами (`/home/firebird/prefix_permanent.sql`) и объектами, требующими перекомпиляции (`/home/firebird/prefix_restore.sql`):

```
isql -user SYSDBA -password masterkey -ch <кодировка> <база данных>
-extract -incompatible /home/firebird/prefix
```

   Каталог `/home/firebird` должен существовать. Если его нет, то создайте его предварительно.

6. Необходимо исправить `SQL`, содержащийся в `prefix_permanent.sql`, чтобы он был совместим с версией 2.6. После исправления выполните скрипт:

```
isql -user SYSDBA -password masterkey -ch <кодировка> -i
/home/firebird/prefix_permantnt.sql <база данных> -nodbtrig
```

7. Выполните бэкап на версии СУБД Ред База Данных 3.0, используя `gbak` от версии 2.6. Для этого в каталог `/opt/rdb_2.6` скопируйте файлы СУБД от 2.6. После этого выполните:

```
export FIREBIRD=/opt/rdb_2.6
export LD_LIBRARY_PATH=/opt/rdb_2.6/lib
gbak -user sysdba -password masterkey -b -v -g 127.0.0.1:<база данных>
<файл бэкапа> -y <файл лога бэкапа>
```

8. Сделайте копии конфигурационных файлов `cp /opt/RedDatabase/*.conf /home/firebird`.
9. Удалите Ред Базу данных 3.0.
10. Установите Ред Базу данных 2.6.
11. Восстановите базу даннх из резервной копии:

```
gbak -user sysdba -password masterkey -c -v <файл бэкапа> <база данных
2.6> -y <путь к логу рестора>
```

   При этом могут возникнуть сообщения `-bad debug info`. Эти предупреждения нужно игнорировать.

12. Перекомпилируйте объекты:

```
isql -user SYSDBA -password masterkey -ch <кодировка> -i
/home/firebird/prefix_restore.sql <база данных 2.6> -nodbtrig
```

13. Восстановите пользователей из базы данных безопасности:

```
isql -user SYSDBA -password masterkey -ch <кодировка>
/opt/RedDatabase/security2.fdb -input
/home/firebird/security_migration.sql
```

14. Уберите права на объект базы данных, потому что на 2.6 они работают иначе:

```
isql -user SYSDBA -password masterkey -ch <кодировка> <база данных 2.6>
update rdb$database set rdb$security_class = null;
commit;
quit;
```

15. Восстановите параметры конфигурации от 3.0. Необходимо для каждого файла конфигурации, сохранённого на шаге 8, перенести активные опции в соответствующий файл

конфигурации в СУБД версии 2.6. Если в 2.6 нет параметра, который присутствовал в 3.0, необходимо либо найти подходящий по смыслу (т.е. случай, когда изменилось имя параметра), либо не переносить его (такого параметра в версии 2.6 просто нет).
Помните, что файлы `directories.conf` и `fbtrace.conf` имеют другой синтаксис. Например, если в 3.0 `directories.conf` выглядит так:

```
database = /mnt/db/test.fdb
{
  test_dir = /tmp/blobs
  blobs = /var/blobs
}
```

а в 2.6 используется XML-подобный синтаксис:

```
<database /mnt/db/test.fdb>
  test_dir /tmp/blobs
  blobs /var/blobs
</database>
```

16. Перезапустите сервер.

Переход с Firebird 4.0 на Ред Базу Данных 3.0

1. Остановите сервер `firebird` и переименуйте файл службы:

```
mv /usr/lib/systemd/system/firebird.service
/usr/lib/systemd/system/fb.service
systemctl daemon-reload
```

2. Установите Ред Базу Данных и остановите службу `firebird`:

```
systemctl stop firebird.service
```

Убедитесь, что служба остановлена:

```
systemctl status firebird.service
```

3. Запустите сервер `Firebird 4`:

```
systemctl start fb.service
```

Сервер Ред Базы Данных при этом не должен быть запущен.

4. Убедитесь, что у сервера Ред Базы Данных 3.0 есть нужные плагины, `UDF` и `UDR` (например, `cluster`, `crypto`).
5. Выполните бэкап с помощью `gbak` от Ред Базы Данных 3:

```
gbak -user sysdba -pas masterkey -b -v -g -y <путь_к_логу_бекапа>
localhost:<путь_к_базе_fb4> <путь_к_бэкапу>
```

6. Остановите сервер `Firebird 4` и запустите сервер Ред База Данных 3:

```
systemctl stop fb.service
systemctl start firebird.service
```

7. Восстановите базу данных из резервной копии, используя gbak от Ред Базы Данных 3.0:

```
gbak -user sysdba -pas masterkey -c -v -y <путь_к_логу_рестора> -se
localhost:service_mgr <путь_к_бекапу> <путь_к_базе_rdb3>
```

8. При ресторе некоторые объекты могут восстановиться некорректно, тогда в логе будет запись:

```
gbak: WARNING:Database is not online due to failure to restore one or more
objects. gbak: WARNING:Run gfix -online to bring database online.
```

   Список объектов, которые нужно перекомпилировать, указан в логе в формате: `<процедура/функция/триггер> <имя объекта> was invalidated due to BLR parsing error`

9. Перекомпилируйте проблемные объекты с помощью RDB Expert:
   - Подключитесь к восстановленной базе данных от имени SYSDBA;
   - Нажмите правой кнопкой мыши на узел с процедурами (триггерами, функциями и пакетами). В контекстном меню выберите пункт Перекомпилировать все невалидные Процедуры;
   - Если при перекомпиляции невалидных процедур возникли ошибки, запустите перекомпиляцию ещё раз. Если после этого ошибки остались, обратитесь в поддержку СУБД;
   - Отключитесь от базы данных в RDB Expert.
10. Переведите базу данных в режим online:

```
gfix -user sysdba -pas masterkey -online <путь_к_базе_rdb3>
```

Глава 3

Редакции СУБД Ред База Данных 3.0

СУБД Ред База Данных выпускается в нескольких редакциях. Это открытая редакция, стандартная и промышленная. От выбора редакции зависит содержимое дистрибутива.

Базовый состав дистрибутива присутствует во всех редакциях - в том числе и в *Открытой*, которую можно скачать на сайте. В него входят: сервер СУБД (исполняемые и исходные коды), дополнительные модули расширения СУБД, средства инсталляции, настройки и администрирования СУБД. В базовый функционал включены:

- Поддержка стандарта SQL-2008
- Масштабируемая SMP архитектура SuperServer
- Мониторинг производительности и аудит событий в реальном времени
- Аудит изменения прав доступа
- Автономные транзакции
- 64-битный счетчик транзакций
- Инкрементальный бэкап
- Возможность восстановления инкремента в существующую БД
- Онлайн валидация БД
- Глобальные временные таблицы
- Внешние хранимые процедуры, триггеры и функции
- Оконные функции
- Расширенный набор системных функций

Стандартная редакция предназначена для использования в бизнес-приложениях на предприятиях среднего и малого бизнеса. Она входит в реестр российского ПО и имеет сертификат ФСТЭК. Функционал расширен по сравнению с Открытой редакцией и в нем дополнительно присутствует:

- Аутентификация по ГОСТ алгоритмам
- Поддержка создания внешних процедур, функций и триггеров, написанных на Java
- Контроль целостности файлов сервера и метаданных
- Аутентификация через LDAP/AD

Промышленную редакцию предлагается использовать в приложениях на предприятиях крупного бизнеса, на участках, где ценность данных и стоимость отказа системы чрезвычайно велики. В дополнение к содержимому стандартной редакции в поставку включены:

- Синхронная и асинхронная репликация
- Отказоустойчивый кластер
- Аутентификация через GSSAPI
- Полнотекстовый поиск
- Утилита репликации/зеркалирования БД (`RedReplicator`)

Глава 4

Установка, обновление и запуск сервера СУБД Ред База Данных

4.1 Поддерживаемые ОС

СУБД Ред База Данных может функционировать на следующих ОС:

- Microsoft Windows (x64);
- Linux x86, Linux x86_64 дистрибутивы, использующие:
  - glibc 2.12 и старше;
  - libstdc++ от gcc 5.3 и старше;
- Linux дистрибутивы, поддерживающие Linux Standard Base ISO/IEC 23360, начиная с версии 3.0.

Рекомендуемые ОС семейства Linux:

- РЕД ОС 7.3 и старше;
- Альт 8 СП и старше;

4.2 Приемка поставленного сервера СУБД Ред База Данных

Правила приемки описаны в разделе 4 технических условий "Система управления базами данных «Ред База Данных»" ТУ 502120-001-29926343-2015.

4.3 Установка в ОС Windows

4.3.1 Установка из исполняемого файла

Скачать дистрибутив Ред Базы Данных можно с официального сайта СУБД — reddatabase.ru. Загрузка доступна только авторизованному пользователю.

Запустите установку СУБД Ред База Данных с помощью файла `RedDatabase-OE-3.0.X.X-windows-X.exe`, определив разрядность используемой операционной системы.

Инсталляция СУБД Ред База Данных осуществляется с помощью стандартного мастера установки программ. В ходе установки мастер собирает всю необходимую для установки сервера информацию, производит копирование файлов и регистрацию программных модулей в реестре Windows.

Для установки Ред База Данных 3.0 необходимы права администратора.

Выберите язык установки. Предусмотрена установка на русском и английском языках.

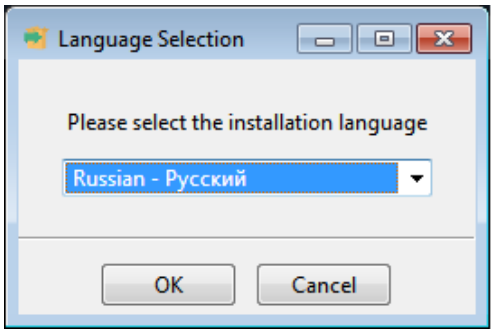


Рисунок 4.1 — Выбор языка

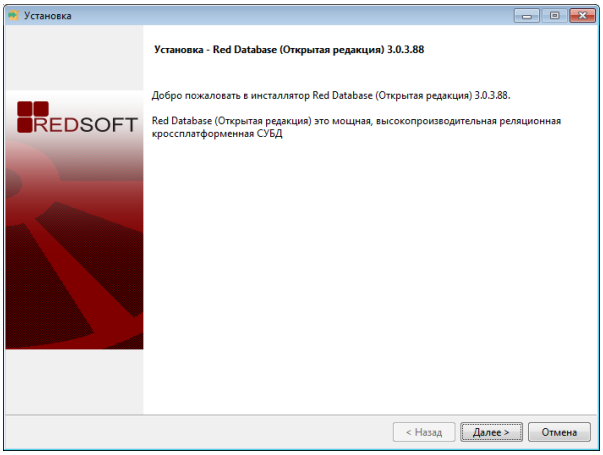


Рисунок 4.2 — Приветственное окно установки

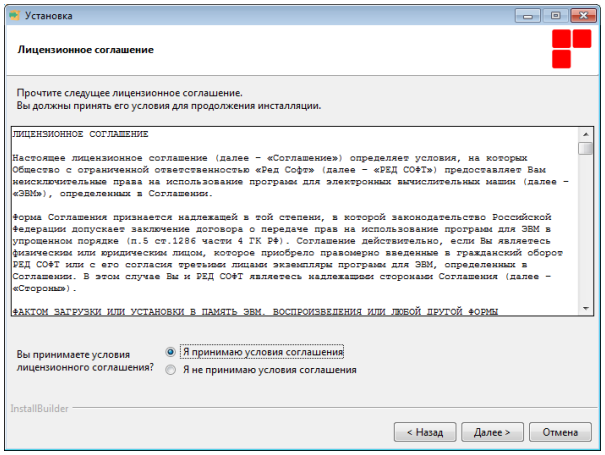


Рисунок 4.3 — Лицензионное соглашение

Во время инсталляции Вам будет предложено выбрать архитектуру сервера: Classic, SuperClassic или SuperServer:

- **Classic**:
  - использует отдельный процесс на каждое пользовательское соединение;
  - каждый процесс содержит в себе все что нужно для работы с базой данных: область памяти для метаданных, кэш данных для минимизации повторных чтений из файла БД; память для сортировок;
  - если происходит сбой, другие соединения остаются работоспособными
  - поддержка мультипроцессорности: в многопроцессорных системах ОС автоматически распределяет процессы по процессорам/ядрам
- **Superserver**:
  - один процесс с общей областью памяти для всех пользовательских соединений;
  - поддержка мультипроцессорности: параллельные запросы пользователей выполняются на разных ядрах;
  - возможный сбой в одном процессе разорвет все подключения;
- **SuperClassic**:
  - единый процесс на всех пользователей с общей памятью под сортировки;
  - используется пул потоков ОС для обработки запросов от соединений, таким образом каждое соединение работает в отдельном потоке управляемом ОС, а неактивные соединения не отъедают ресурсы потоков;

– каждый поток со своим кэшем данных и областью метаданных;
– поддержка мультипроцессорности: потоки ОС легко распараллеливаются;
– возможный сбой в одном процессе разорвет все подключения.

Каждый из режимов стабилен, и нет причин полностью отдавать предпочтение какому то одному. Конечно, у вас могут быть свои собственные конкретные соображения. Если Вы сомневаетесь, просто следуйте за установкой по умолчанию. Позже вы можете изменить архитектуру через файл конфигурации `firebird.conf` (параметр `ServerMode`), что потребует перезагрузки, но не переустановки.

Режим сервера может быть настроен для каждой базы данных отдельно.

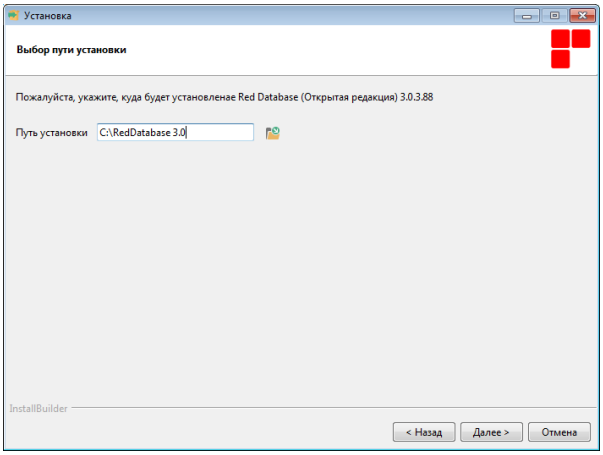


Рисунок 4.4 — Выбор каталога установки

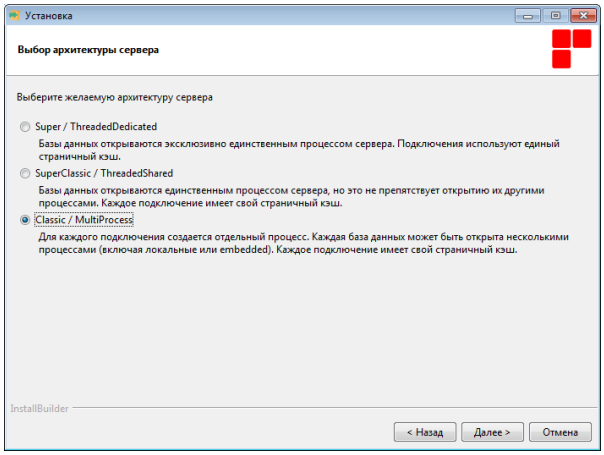


Рисунок 4.5 — Выбор архитектуры сервера

Изначально в системе существует только один пользователь – администратор сервера `SYSDBA` (пароль по умолчанию – `masterkey`). Этот пользователь обладает полными правами на выполнение всех функций по управлению работой сервера и работе с базами данных. В процессе инсталляции Вас попросят изменить пароль данного пользователя в целях безопасности.

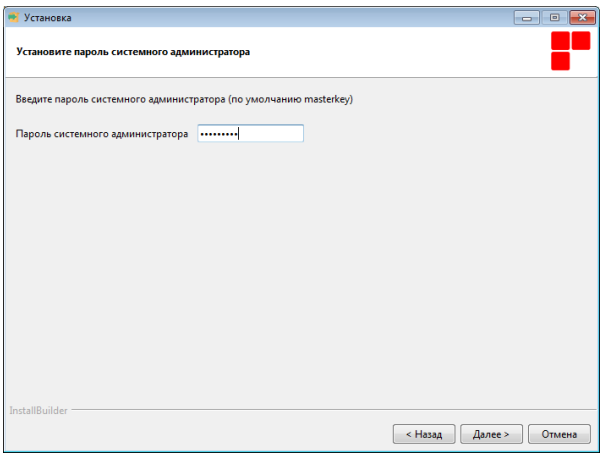


Рисунок 4.6 — Установка пароля

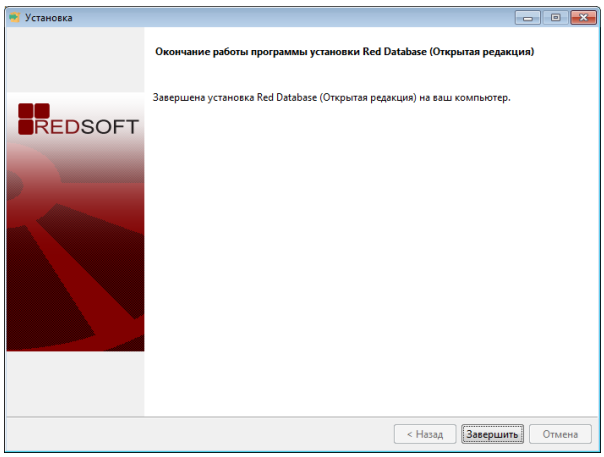


Рисунок 4.7 — Завершение установки

По окончанию процесса установки будет запущен серверный процесс `rdbserver`, который будет запускаться автоматически при перезагрузке сервера. Сервер будет работать как системная служба.

Сервер может работать и в качестве приложения (это менее предпочтительный вариант). Для запуска используйте следующую команду:

```
rdbserver -a
```

Исполняемый файл `rdbserver.exe` расположен в корневом каталоге установки Ред Базы Данных.

Остановка приложения выполняется через иконку в системном трэе (*Shutdown*).

Проверьте версию СУБД:

```
gfix.exe -z
```

Проверьте сетевое подключение к БД:

```
isql.exe -u sysdba -p masterkey localhost:<путь_к_БД>
```

4.3.2 Установка из архива

1. Скачайте архив с корневым каталогом сервера Ред Базы Данных с официального сайта СУБД — reddatabase.ru. Загрузка доступна только авторизованному пользователю.
2. Распакуйте содержимое архива в пустую папку.
3. В `firebird.conf` раскомментируйте параметр `ServerMode` и укажите желаемую архитектуру сервера.
4. Запустите `install_service.bat`. По окончании работы bat-файла служба СУБД автоматически запустится.
5. Проверьте версию СУБД:

   ```
   gfix.exe -z
   ```

6. По умолчанию существует только пользователь `SYSDBA` плагина `Legacy`. В `firebird.conf` дополните параметр `AuthServer` значением `Legacy_Auth`. Создайте пользователя `SYSDBA` плагином `Srp`.
7. Проверьте сетевое подключение к БД:

   ```
   isql.exe -u sysdba -p masterkey localhost:<путь_к_БД>
   ```

4.4 Установка в Unix-системах

4.4.1 Установка в графическом режиме

Файлы Ред База Данных 3.0 поставляются в виде бинарного пакета. При запуске его из любой графической системы (например, KDE) будет вызван мастер установки, который произведет сбор всей необходимой информации и установит СУБД Ред База Данных 3.0 на Ваш компьютер.

Для установки СУБД Ред База Данных необходимо скопировать дистрибутивный файл `RedDatabase-3.0.X.X-X-linux-X.bin` на жесткий диск, а в операционной системе назначить в правах этого файла разрешение на исполнение:

```
# chmod +x RedDatabase-3.0.X.X-X-linux-X.bin
```

После этого запустить установку СУБД Ред База Данных:

```
# ./RedDatabase-3.0.X.X-X-linux-X.bin
```

Для установки сервера Ред База Данных 3.0 необходимы права суперпользователя (root).

Инсталляция СУБД Ред База Данных осуществляется с помощью стандартного мастера установки программ. Прежде всего предлагается выбрать язык установки. Предусмотрена инсталляция на русском и английском языках.

Рисунок 4.8 — Выбор языка установки

Во время инсталляции Вам будет предложено выбрать архитектуру сервера: Classic, SuperClassic или SuperServer. Подробнее о каждой архитектуре было описано выше.

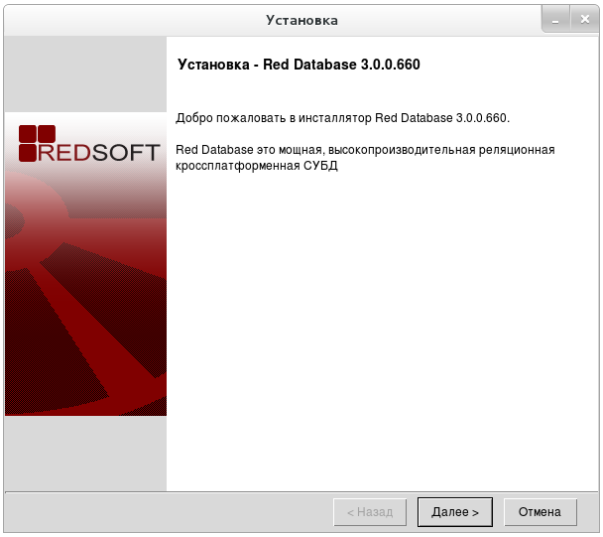


Рисунок 4.9 — Приветственное окно установки

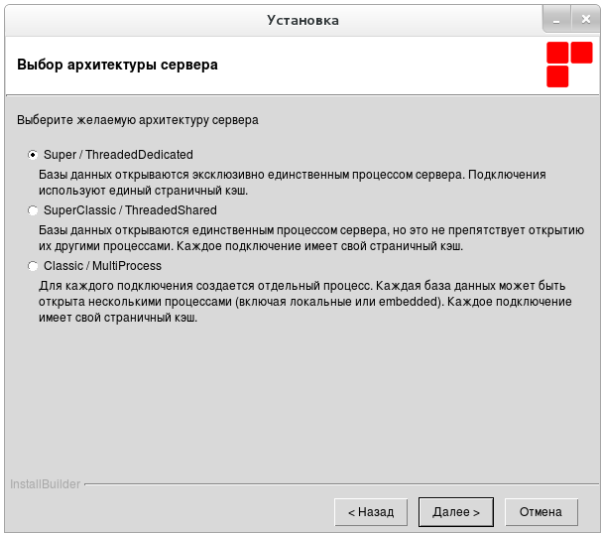


Рисунок 4.10 — Выбор архитектуры сервера

Изначально в системе существует только один пользователь – администратор сервера `SYSDBA` (пароль по умолчанию – `masterkey`). Этот пользователь обладает полными правами на выполнение всех функций по управлению работой сервера и работе с базами данных. В процессе инсталляции Вас попросят изменить пароль данного пользователя в целях безопасности.

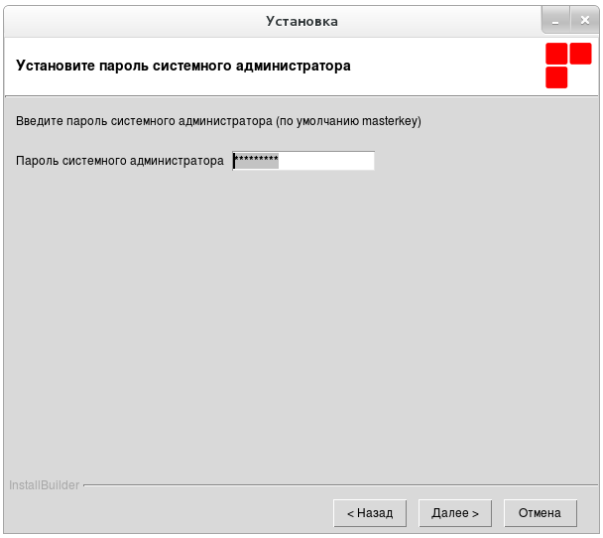


Рисунок 4.11 — Установка пароля

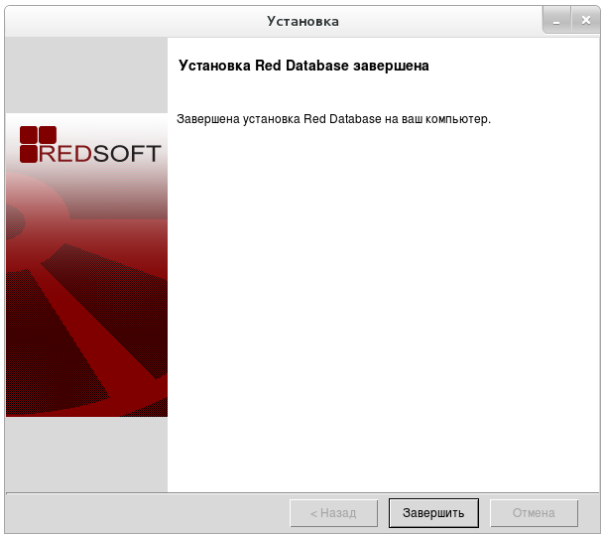


Рисунок 4.12 — Завершение установки

После установки сервер автоматически не запускается. Вам понадобится сделать это вручную в зависимости от типа системы Linux и вашего установочного пакета (см. Запуск и остановка сервера).

Проверьте версию СУБД:

```
gfix -z
```

Проверьте сетевое подключение к БД:

```
isql -u sysdba -p masterkey localhost:<путь_к_БД>
```

4.4.2 Установка в консольном режиме

Если программа инсталляции запущена в консольном режиме (с ключом `--mode text`), то она последовательно будет выводить запросы на подтверждение тех или иных параметров установки, таких как выбор компонентов или пароль пользователя `SYSDBA`. В случае, если на тот или иной запрос мастера установки предусмотрен ответ по умолчанию, то такой вариант обозначен заглавной буквой, и в этом случае этот вариант можно подтвердить нажатием клавиши «Ввод», в случае же, если выбор по умолчанию не предусмотрен, то необходимо обязательно ответить на вопрос «Да» (Y) или «Нет» (N).

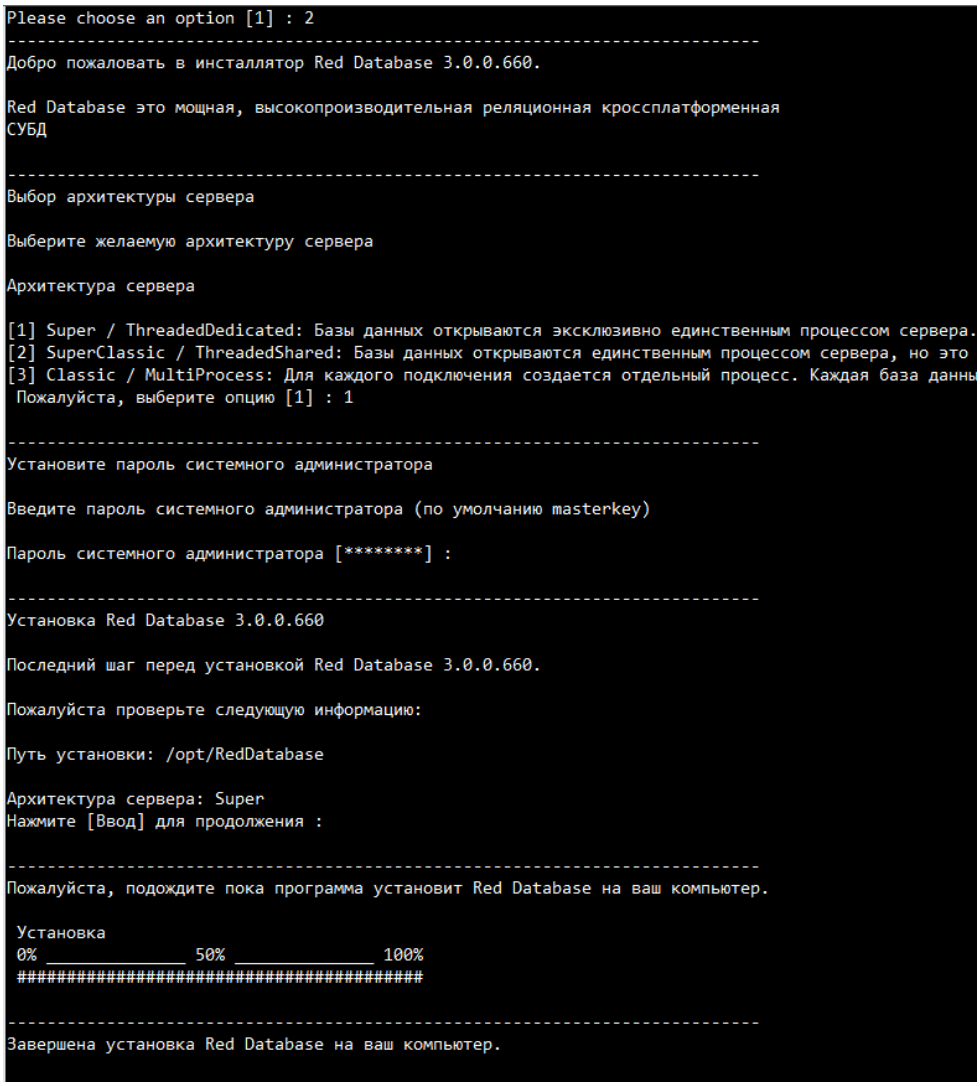


Рисунок 4.13 — Пример установки Ред База Данных 3.0 в текстовом режиме

4.4.3 Установка пакетов

Существует два вида пакетов для установки сервера СУБД Ред База Данных — `reddatabase-server-3.0` и `reddatabase-common-3.0`. Пакет `reddatabase-common` устанавливает каталоги, стартовые скрипты, юниты `systemd`, создает пользователей. Пакет `reddatabase-server` устанавливает бинарные файлы. Установка происходит в `/usr/lib/reddatabase/3.0`.

Возможна установка пакетов параллельно с инсталлятором любой версии и параллельно с пакетами версии 4.0.

Версия пакета `reddatabase-common` должна быть не ниже `reddatabase-server`.

После установки сервер автоматически не запускается. Для запуска нужно запустить сервис `reddatabase@3.0`. У инсталляций с `systemd` дополнительно есть сервис `reddatabase`, который запускает все установленные в пакетах сервера, то есть например `reddatabase@3.0` и `reddatabase@4.0`.

Пример установки `rpm` и `deb` пакетов:

```
$ sudo rpm -ivh rdb30-reddatabase-common-3.0.5.95.rpm
$ sudo rpm -ivh rdb30-reddatabase-server-3.0-x86_64-3.0.5.95.rpm
```

```
-----------------------------------------------------------------------
$ sudo apt-get install rdb30-reddatabase-common-3.0.5.95.deb
$ sudo apt-get install rdb30-reddatabase-server-3.0-x86_64-3.0.5.95.deb
```

4.4.4 Установка отладочных символов

1. Скачайте архив с отладочными символами с официального сайта СУБД — reddatabase.ru. Версия отладочных символов должна совпадать с версией установленной Ред Базы Данных. Загрузка доступна только авторизованному пользователю.
2. Распакуйте содержимое архива в папку `/opt/RedDatabase`.

```
tar -xvf RedDatabase-3.0.X.X-X-dbg-linux-X.tar.gz -strip-components 1
```

4.5 Процедуры после установки

Используйте двухфакторную аутентификацию. Для использования этого в файле конфигурации `firebird.conf` укажите следующие настройки:

```
AuthServer = Multifactor, Srp
AuthClient = Multifactor, Srp
UserManager = Multifactor_Manager
CryptoPlugin = Crypto_API
```

Для того, чтобы пользователь Ред База Данных смог пройти многофакторнуюаутентификацию, он должен быть предварительно создан с помощью плагина управления пользователями `Multifactor_Manager`:

```
CREATE USER test PASSWORD 'test'
USING PLUGIN Multifactor_Manager;
```

Для того, чтобы сделать `SYSDBA` многофакторным, необходимо создать нового пользователя с именем `SYSDBA`, используя плагин `Multifactor_Manager`.

Настройте политику безопасности по умолчанию, выполнив следующую команду:

```
ALTER POLICY "DEFAULT" AS
  PSWD_MIN_LEN = 8,
  PSWD_NEED_DIFF_CASE = true,
  PSWD_NEED_CHAR = 5,
  PSWD_NEED_DIGIT = 2,
  MAX_FAILED_COUNT = 4;
```

Политика безопасности по умолчанию:

- Длина пароля не менее 8 символов;
- Пароль должен содержать символы в разных регистрах;
- Пароль должен содержать минимум 5 букв;
- Пароль должен содержать минимум 2 цифры;
- Максимальное количество неуспешных попыток аутентификации до блокировки равно 4.

Для обеспечения контроля целостности выполните следующее:

1. Добавьте в секцию `Service` файла `/lib/systemd/system/firebird.service` проверку целостности процедур:

```
ExecStartPost=sh -c'LD_PRELOAD=""/opt/RedDatabase/bin/mint -M check -d <путь к
файлу базы данных> -o procedures -u sysdba -p masterkey -C <алиас>-R 80 -i <путь
к файлу сохранения> -I AT_SIGNATURE|| true'
```

   Загрузите изменения службы:

```
systemctl daemon-reload
```

2. Создайте задачу `cron`, которая будет выполнять проверку целостности каждые 12 часов:

```
crontab -e

0 */12 * * * /opt/RedDatabase/bin/mint -M check -d <путь к файлу базы данных> -o
procedures -u sysdba -p masterkey -C "CN_cert,CN_issuer,CN_serial" -R 80 -I
AT_SIGNATURE -i <путь к файлу сохранения>
```

4.6 Обновление сервера

1 способ

Начиная с версии 3.0, обновление сервера между минорными- и патч-версиями Ред Базы Данных происходит с помощью стандартного мастера установки программ. Перед установкой новой версии больше не требуется вручную удалять старую.

Чтобы обновить сервер, просто скачайте нужную версию в зависимости от ОС и разрядности и запустите установку, как было описано в подразделах Установка в ОС Windows и Установка в Unix-системах.

Перед обновлением убедитесь, что все пользователи отключены от баз данных. С помощью команды:

```
gfix -shut -force <n>
```

можно закрыть все подключения и запретить последующие.

В процессе инсталляции (обновления) Вам будет предложено выбрать только язык установки и принять лицензионное соглашение. Новая версия будет установлена по пути прежней с заменой всех файлов, кроме:

- `databases.conf`
- `directories.conf`
- `fbtrace.conf`
- `firebird.conf`
- `plugins.conf`
- `java-security.fdb`
- `security3.fdb`
- `fbjava.yaml`
- `fbtrace_dba.conf`
- `fbtrace_sec.conf`

- `jvm.args`
- `firebird.log` (только на Linux)

После успешной инсталляции эти оригинальные файлы сохраняются с суффиксом `.dist`.

2 способ (для ОС Window)

1. Скачать архив с новой версией Ред Базы Данных.
2. Остановить службу RedDatabase Server
3. Убедиться в отсутствии процессов `rdbserver.exe`.
4. Сделать копию каталога с установленной Ред Базой Данных (желательно указывать номер версии в названии каталога).
5. Выполнить установку новой версии СУБД с помощью распаковки файлов из скачанного архива с новой версией Ред Базы Данных (с заменой файлов).
6. Скопировать из каталога со старой версией Ред Базы Данных следующие файлы (в папку с установленной СУБД):
    - `databases.conf`
    - `directories.conf`
    - `fbtrace.conf`
    - `firebird.conf`
    - `plugins.conf`
    - `java-security.fdb`
    - `security3.fdb`
    - `fbjava.yaml`
    - `fbtrace_dba.conf`
    - `fbtrace_sec.conf`
    - `jvm.args`
    - `firebird.log`
7. Запустить службу СУБД — RedDatabase Server
8. Проверить версию СУБД:

```
gfix.exe -z
```

9. Проверьте сетевое подключение к БД:

```
isql.exe -u sysdba -p masterkey localhost:<путь_к_БД>
```

2 способ

1. Скачать дистрибутив новой версии Ред Базы Данных.
2. Остановить службу `firebird`

```
systemctl stop firebird
-----------------------
service firebird stop
```

3. Закомментировать алиас в `databases.conf`.
4. Проверить с помощью `lsof`, что файл базы данных не держит ни один процесс (не должны выводиться pid-ы процессов).

```
sudo lsof <путь_до_базы_данных>
```

Если таковые остались — завершить их.

```
killall rdbserver
```

5. Переименовать каталог `opt/RedDatabase/` в `opt/RedDatabase_<версия_СУБД>`
6. Установить скачанный дистрибутив.
7. Из каталога старой версии скопировать следующие файлы:
   - `databases.conf`
   - `directories.conf`
   - `fbtrace.conf`
   - `firebird.conf`
   - `plugins.conf`
   - `java-security.fdb`
   - `security3.fdb`
   - `fbjava.yaml`
   - `fbtrace_dba.conf`
   - `fbtrace_sec.conf`
   - `jvm.args`
   - `firebird.log`
8. Запустить службу СУБД

```
systemctl start firebird
------------------------
service firebird start
```

9. Раскомментировать алиас в `databases.conf`.
10. Проверит сетевое подключение

```
isql -u sysdba -p masterkey localhost:<путь_к_БД>
```

4.7 Запуск и остановка сервера

После окончания процесса установки на Windows сервер Ред Базы Данных должен быть запущен как сервис.На Linux сервис Вы должны запустить вручную.

4.7.1 На Linux

Используйте команду `top` командной строки для проверки запущенных процессов в интерактивном режиме. Если сервер Ред База Данных 3.0 запущен, вы должны увидеть процесс с именем `rdbserver` и, возможно, также `rdbguard` (процесс Guardian).

На картинке показан вывод `top`, ограниченный `grep`, чтобы отображать только записи, содержащие подстроку `rdb`:

```
[user@centos6  ]$ top -b -n1 | grep rdb
  3835 firebird 20 0 31288  920  516 S 0.0 0.0 0:00.00 rdbguard
  3836 firebird 20 0  127m 2468 1980 S 0.0 0.1 0:00.01 rdbserver
```

Как альтернатива, вместо команды `top`, Вы можете использовать `ps -ax` или `ps -aux`, при необходимости перенаправив вывод `grep`.

Другой способ проверить сервер после установки - запустить клиент (например, `isql`) и подключиться к базе данных или создать ее. Эти операции описаны в Руководстве по SQL.

Если окажется, что среди запущенных процессов не окажется сервиса `rdbserver`, запустите сервер вручную. Управление запуском и остановкой сервера осуществляет демон инициализации `systemd`

```
systemctl start firebird
systemctl stop firebird
```

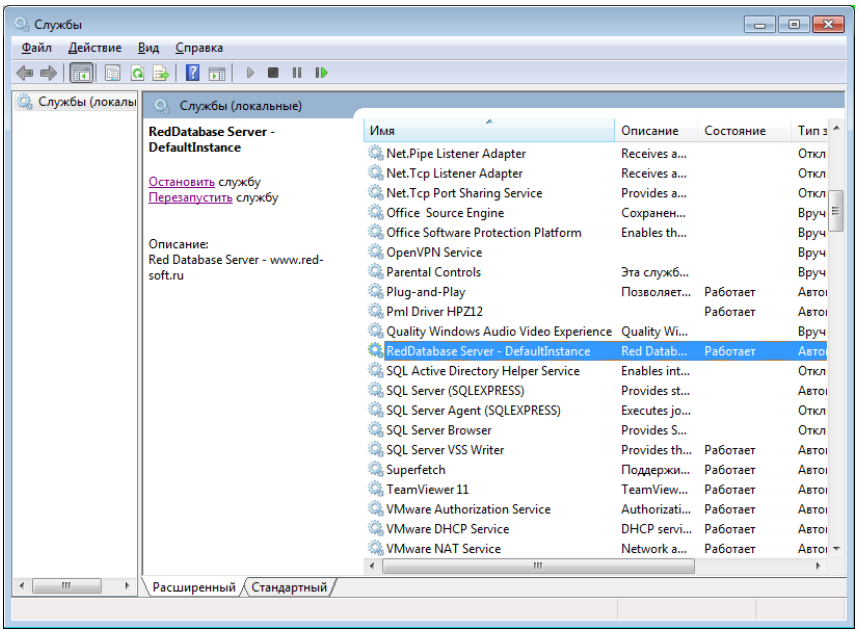
или `init` скрипт:

```
service firebird start
service firebird stop
```

4.7.2 На Windows

Откройте *Панель управления→Администрирование→Службы.*

На картинке представлен вид апплета Services (Службы) на Windows 7. Внешний вид может изменяться в зависимости от версии Windows.



В списке сервисов вы должны найти сервер RedDatabase. Если сервер по каким то причинам не запущен, вы можете сделать это сейчас, щелкнув правой кнопкой мыши запись RedDatabase Server-DefaultInstance и нажав «Запустить».

4.8 Сборка из исходных файлов

Дистрибутив СУБД Ред База Данных, самостоятельно собранный из предоставляемых исходных файлов, не считается сертифицированным.

4.8.1 На ОС Windows

На официальном сайте СУБД Ред Базы Данных — reddatabase.ru — можно загрузить исходные коды сервера.

Для сборки сервера СУБД Ред База Данных необходимы:

- ОС Windows 7 x86, x86_64 и выше.
- Visual Studio 2010 SP1. Для поддержки 64-разрядной сборки необходимо при установке среды отметить компонент «The x64 compilers and tools»
- Sed 4.1.5 и выше. После установки необходимо добавить путь к `sed.exe` в переменные окружения, например: `C:\Program Files (x86)\sed\bin`.
- Файловый архиватор 7-Zip.

Все исходные файлы необходимо поместить в одну папку, например, `C:\RedDatabase\rdb_3_0` (путь должен быть без пробелов).

Для сборки сервера необходимо запустить BAT файл:

```
builds\win32\run_all.bat
```

Если данный BAT файл запускается из 32-разрядного приложения, то будет собран 32-разрядный сервер, если из 64-разрядного – 64-разрядный. Также 64-разрядный сервер из 32-разрядного приложения можно собрать, если перед запуском установить переменную среды:

```
set FB_PROCESSOR_ARCHITECTURE=x86
```

или

```
set FB_PROCESSOR_ARCHITECTURE=AMD64.
```

В результате удачной сборки в каталоге `firebird` появятся соответственно директории `output_Win32` и `output_x64`.

4.8.2 На ОС Linux

В комплект поставки СУБД Ред База Данных входят исходные файлы сервера.

Для сборки сервера СУБД Ред База Данных под ОС Linux необходимы следующие компоненты:

- gcc-5 или выше;
- gcc-c++-5 или выше;
- libstdc++-5 или выше;
- autotools;
- rh-python36;
- devtoolset-4-toolchain;
- devtoolset-4-libatomic-devel;
- make (gmake);
- zlib-devel;
- libicu-devel;
- openldap-devel;
- ncurses-devel;
- openmotif;
- wget;

- krb5-devel;
- binutils-devel;
- libatomic;
- CryptoPRO;
- redhat-lsb.

Сначала необходимо запустить процедуру конфигурации сборки. Для этого запускается файл `autogen.sh`:

```
./autogen.sh --enable-bindeps --enable-binreloc --with-builtin-tommath
--enable-wincrypt --prefix=/opt/RedDatabase
```

Это традиционный способ сборки, результатом которого будут исполнимые файлы сервера, не совместимые со стандартом LSB. Как собрать LSB-совместимый сервер в данном руководстве не описывается.

Скрипту `autogen.sh` можно передать ряд опций, описание которых выводится при запуске `autogen.sh --help`. Наиболее важными являются следующие опции:

- `--enable-bindeps` – использовать бинарные компоненты fbjava, fbjava-lucene и jaybird.
- `--with-builtin-tommath` — использовать библиотеку libtommath поставляемый с исходным кодом (по умолчанию используется установленная в системе).
- `--enable-wincrypt` – сборка криптоплагина, используемого для работы с некоторыми функциями безопасности сервера. Для сборки криптоплагина требуется установить пакеты CryptoPro (lsb-cprocsp-base; lsb-cprocsp-rdr; lsb-cprocsp-kc1; lsb-cprocsp-capilite).
- `--prefix=/path` – путь установки сервера СУБД Ред База Данных. По умолчанию собранный сервер устанавливается в каталог `/opt/RedDatabase`.

По завершении процедуры конфигурирования должна быть запущена сборка командой `make`. Она соберет сервер и подготовит его к установке. Установка осуществляется командой `make install`, запущенной от имени суперпользователя (`root`).

4.9 Деинсталляция

Деинсталляция сервера осуществляется запуском программы `uninstall`, расположенной в корневой папке установки Ред Базы Данных. После запуска скрипта пользователь должен подтвердить, что действительно хочет удалить Ред Базу Данных, после чего будет произведена деинсталляция сервера.

Программа деинсталляции `uninstall` также доступна с параметром `--mode text`.

Глава 5

Состав файлов сервера

5.1 Файлы конфигурации

databases.conf

В версии Ред Базы Данных 3.0 файл `aliases.conf` был переименован в `databases.conf`. В этом текстовом файле можно сопоставить конкретный путь к БД и псевдоним, чтобы затем в прикладных кодах использовать более короткий и удобный псевдоним для обращения к нужной базе данных. Также здесь указываются индивидуальные настройки для каждой конкретной базы данных.

directories.conf

В данном файле хранятся псевдонимы каталогов и соответствующие им реальные пути к ним, где хранятся файлы, заполненные `BLOB` – данными.

fbtrace.conf

Файл с шаблоном настроек `fbtrace.conf` находится в корневом каталоге и содержит список отслеживаемых событий и указывает размещение логов трассировки для каждого события. Это позволяет достаточно гибко настроить параметры аудита различных событий для любой базы данных, при этом логирование будет осуществляться в отдельные файлы.

fbtrace_dba.conf

Файл конфигурирования для аудита действий пользователя `SYSDBA`. Имеет такую же структуру, что и `fbtrace.conf`, и на его основе создается дополнительная системная сессия аудита.

fbtrace_sec.conf

Файл конфигурирования для аудита событий безопасности. Имеет такую же структуру, что и `fbtrace.conf`, и на его основе создается дополнительная системная сессия аудита.

firebird.conf

Файл содержит параметры настройки сервера.

plugins.conf

Файл используется для настройки различных плагинов. Если в файле не указана конфигурация для плагина, то для него будут действовать настройки по умолчанию.

replication.conf

Используется для настройки системы репликации.

security3.fdb

База данных безопасности. В этой базе хранятся параметры пользователей системы, политики доступа, глобальные роли.

firebird.msg

Файл с сообщениями сервера (в основном об ошибках).

firebird.log

Лог-файл сервера.

5.2 Инструменты администрирования и сервисы Ред Базы Данных

rdblogmgr <.exe>

Утилита настройки журнала репликации. Данная утилита предназначена для вывода детализации текущего состояния журнала для заданной базы (общее состояние журнала, настройки журнала конфигурации, список использованных сегментов). Дополнительно, утилита `rdblogmgr` позволяет выполнить ручное архивирование заданного сегмента журнала или всех сегментов, а также принудительно помечает используемый сегмент как полный для возможности его архивирования.

rdbrepldiff <.exe>

Утилита сравнения мастер-базы и реплики.

rdbreplmgr <.exe>

Утилита управления асинхронной репликой. Выводит информацию о состоянии асинхронной репликации, а также позволяет применять вручную журналы к реплике и создавать реплику как копию мастер-базы.

gbak <.exe>

Эта утилита предназначена для резервного копирования и восстановления баз данных. Она также обнаруживает разрушения базы данных, освобождает дисковое пространство, появившееся в результате удалений, разрешает незавершенные транзакции, позволяет разделять базы данных на несколько файлов. Она также используется для создания переносимой копии с целью восстановления вашей базы данных на другой аппаратной платформе.

gfix <.exe>

Это набор общих вспомогательных утилит для изменения свойств баз данных, устранения небольших повреждений базы данных, выполнения различных задач чистки и т. д. Утилита также предоставляет средство администратора для отключения конкретных баз данных до завершения работы сервера. Она может быть использована вместе с утилитой `gbak` для восстановления некоторых типов нарушений в базе данных.

gpre <.exe>

Это препроцессор, который конвертирует исходный код, написанный на некоторых языках и содержащий встроенный псевдокод SQL, в корректный отформатированный вызов функций Firebird API.

gsec <.exe>

Этот инструмент поддержки списка пользователей и их паролей является интерфейсом командной строки для базы данных security3.fdb; он управляет записями пользователей на сервере.

gstat <.exe>

Этот инструмент получения статистики собирает и отображает статистические сведения по индексам и данным базы данных.

hashgen <.exe>

Используется для проверки целостности компонентов СУБД Ред База Данных на внешних накопителях и в оперативной памяти во время загрузки и динамически в процессе работы сервера.

instclient.exe

Назначение утилиты `instclient` состоит в том, что она:

- позволяет установить клиентскую часть Ред Базы Данных одной командой;
- позволяет установить клиентскую часть как `fbclient.dll`, либо как `gds32.dll`;
- позволяет проверить наличие установленной библиотеки `fbclient` или `gds32`;
- позволяет удалить уже установленный в системе `fbclient` или `gds32`.

instreg.exe

Эта утилита прописывает необходимую информацию в реестр Windows, указывая стандартное расположение остальных файлов сервера.

instsvc.exe

Утилита записывает, удаляет или меняет информацию о запуске сервера в базе сервисов операционной системы Windows.

isql <.exe>

Интерактивный инструмент, который позволяет выполнять запросы к базе данных.

mint <.exe>

Утилита осуществляет контроль за целостностью метаданных в БД. Эта утилита предназначена для извлечения и хэширования метаданных из баз данных, а также для проверки ранее полученного хэша метаданных.

nbackup <.exe>

Утилита позволяет создавать резервные копии и восстанавливать из резервных копий также, как `gbak`, и дополнительно позволяет создавать инкрементные копии и восстанавливать из них БД.

rdbguard <.exe>

Исполняемый файл приложения Guardian. Он контролирует состояние сервера. Если сервер был остановлен по какой-либо причине, Guardian автоматически перезапускает его.

rdbserver <.exe>

Исполняемый файл в случае архитектуры Classic, SuperClassic или SuperServer.

rdb_lock_print <.exe>

Эта утилита формирует статистические данные файла блокировок, который поддерживается в Ред Базе Данных для управления последовательностью изменений базы данных несколькими транзакциями. Она может быть полезным инструментом анализа проблем взаимной блокировки.

rdbsvcmgr <.exe>

Утилита предоставляет интерфейс командной строки для Services API, обеспечивая доступ к любой службе, которая реализуется в СУБД.

rdbtracemgr <.exe>

Утилита для работы в интерактивном режиме с трассировкой.

Глава 6

Настройка сервера «Ред База Данных»

Для настройки сервера «Ред База Данных» используется файл `firebird.conf`. Настройки считываются из файла один раз при старте сервера, если архитектура Супер или Суперклассик, и при каждом соединении с базой, если архитектура сервера Классик.

По умолчанию все параметры в файле конфигурации закомментированы. Для обозначения комментариев используется символ «#». Текст, следующий после символа «#», до конца строки является комментарием, например:

```
#комментарий
DefaultDbCachePages = 32768 #комментарий
```

Максимальная длина строки в файле конфигурации сервера равна 80 символов.

Первое слово в строке, начинающейся не с символа комментария, считается названием параметра. Справа от имени параметра, после символа «=», указывается значение параметра.

В файле конфигурации присутствуют параметры трех типов:

- Целочисленный;
- Строковый;
- Логический (булев).

Значения параметров, определяющих объем памяти, указываются в байтах. В конце таких значений можно ставить сокращения k, m и g, соответствующие килобайтам, мегабайтам и гигабайтам.

Некоторые параметры могут задаваться только для конкретных баз данных или будут действовать только для конкретных настроек соединения. Настройки для баз данных задаются в файле `databases.conf`. Настройки для соединения - прежде всего клиентский инструмент и выполняется с помощью параметра `isc_dpb_config` в DPB (или, для Services, `isc_spb_config`). Обратите внимание на то, что настройки для баз данных могут выполняться при помощи DPB в случае Embedded сервера, при подключении к базе данных в первый раз.

Существует ряд предопределенных переменных, которые могут быть использованы в файлах конфигурации, где требуется имя каталога. Полный их список выглядит следующим образом:

- `$(root)` – корневой каталог
- `$(install)` – директория, куда установлена СУБД. Изначально `$(root)` и `$(install)` одинаковые. `$(root)` может быть переопределена установкой или изменением переменной окружения `FIREBIRD`, в таком случае эта переменная отлична от `$(install)`.
- `$(this)` – каталог, в котором находится текущий файл конфигурации.
- `$(dir_conf)` – директория, где располагается `firebird.conf` и `databases.conf`.
- `$(dir_secdb)` – директория, где располагается база данных безопасности по-умолчанию.
- `$(dir_plugins)` – каталог расположения плагинов (`plugins`).
- `$(dir_udf)` – директория, где располагаются функции UDF по-умолчанию (`udf`).
- `$(dir_sample)` – каталог с примерами (`examples`).
- `$(dir_sampledb)` – директория, где лежит пример базы данных (`examples/empbuild`).

- `$(dir_intl)` – директория, в которой расположены международные модули (`intl`).
- `$(dir_msg)` – каталог, где находится файл с сообщениями сервера `firebird.msg`. Обычно он совпадает с `$(root)`, но может быть переопределен переменной окружения `FIREBIRD_MSG`.

Один конфигурационный файл может включать другой с помощью директивы `include`:

```
include some_file.conf
```

Относительный путь представляет собой путь по отношению к текущему файлу конфигурации. Так, в примере выше файл `/opt/config/master.conf` ссылается на файл по пути `/opt/config/some_file.conf`.

Директива `include` поддерживает групповые символы `*` и `?`.

6.1 Общие настройки

DatabaseAccess

Параметр `DatabaseAccess` позволяет обеспечить управление безопасностью при доступе к файлам баз данных. Доступ к базам данных на сервере может быть полным (`Full`), ограниченным (`Restrict`) или запрещенным (`None`).

Параметр `DatabaseAccess` имеет строковый тип; по умолчанию значение параметра равно `Full` - полный доступ. Для того, чтобы запретить доступ, следует выставить значение параметра, равное `None`. Для ограничения доступа используется значение `Restrict`. В этом случае после слова `Restrict` указываются директории, в которых могут быть сохранены файлы баз данных.

При указании каталогов могут быть использованы как абсолютные, так и относительные пути. Относительные пути берутся от корневого каталога инсталляции сервера «Ред База Данных». В качестве разделителя директорий используется символ «;».

```
DatabaseAccess = None
DatabaseAccess = Restrict C:\DataBase
DatabaseAccess = Restrict C:\DataBase;D:\Mirror
DatabaseAccess = Restrict /db;/mnt/mirrordb
DatabaseAccess = Full
```

Неконтролируемый доступ к базам данных может поставить под угрозу безопасность вашей системы. Поэтому настоятельно рекомендуется ограничивать директории для размещения баз данных.

RemoteAccess

Параметр предоставляет или отменяет удаленный доступ к базам данных.

```
RemoteAccess = true
```

По-умолчанию `RemoteAccess` включен для всех баз данных, за исключением базы данных безопасности. Если вы намереваетесь использовать больше одной специализированной базы данных безопасности, то рекомендуем отключить удаленный доступ к ним в файле `databases.conf`.

Для повышенной безопасности следует отключить `RemoteAccess` в `firebird.conf` и включить его в `databases.conf` для некоторых отдельных баз.

Параметр имеет тип `Boolean` и может принимать значения `true/false, 1/0` или `Yes/No`.

ExternalFileAccess

Параметр `ExternalFileAccess` позволяет обеспечить управление правами на создание таблиц во внешних файлах. Разрешение на доступ к внешним файлам может быть полным (`Full`), ограниченным (`Restrict`) или запрещенным (`None`).

Параметр `ExternalFileAccess` имеет строковый тип; значение по умолчанию равно «`None`» - запрет на создание внешних таблиц. Для того, чтобы разрешить создание и доступ к внешним файлам, следует выставить значение параметра равным «`Full`». Для ограничения доступа используется значение «`Restrict`». В этом случае после слова `Restrict` указываются директории, в которых могут быть сохранены файлы внешних таблиц. При указании каталогов могут быть использованы как абсолютные, так и относительные пути. Относительные пути берутся от корневого каталога Ред Базы Данных. В качестве разделителя директорий используется символ «;».

```
ExternalFileAccess = None
ExternalFileAccess = Restrict C:\DataBase
ExternalFileAccess = Restrict C:\DataBase;D:\Mirror
ExternalFileAccess = Restrict /db;/mnt/mirrordb
ExternalFileAccess = Full
```

Неконтролируемая возможность использования внешних таблиц может поставить под угрозу безопасность вашей системы. Поэтому настоятельно рекомендуется использовать этот параметр для ограничения директорий размещения внешних таблиц.

BackupAccess

Параметр `BackupAccess` задаёт список каталогов, в которых разрешено создание резервных копий БД. При создании резервных копий первым проверяется параметр `BackupAccess`. Если создание копий по указанному в параметре пути разрешено - создается заданный каталог (при необходимости - со всеми родительскими).

Параметр `BackupAccess` имеет строковый тип. Параметр может принимать значения: полный (`Full`), ограниченный (`Restrict`) или запрещенный (`None`). По умолчанию значение параметра равно `Full` - полный доступ. Для того, чтобы запретить доступ, следует выставить значение параметра, равное `None`. Для ограничения доступа используется значение `Restrict`. В этом случае после слова `Restrict` указываются директории, в которых могут быть сохранены файлы резервных копий БД.

При указании каталогов могут быть использованы как абсолютные, так и относительные пути. Относительные пути берутся от корневого каталога инсталляции сервера «Ред База Данных». В качестве разделителя директорий используется символ «;».

```
BackupAccess = None
BackupAccess = Restrict C:\DataBase
BackupAccess = Restrict C:\DataBase;D:\Mirror
BackupAccess = Restrict /db;/mnt/mirrordb
BackupAccess = Full
```

UdfAccess

Параметр `UdfAccess` предназначен для определения директорий, в которых могут быть сохранены библиотеки UDF. Разрешение на доступ к библиотекам внешних функций может быть полным (`Full`), ограниченным (`Restrict`) или запрещенным (`None`).

Параметр `UdfAccess` имеет строковый тип; значение по умолчанию равно `Restrict UDF` - `udf`-библиотеки ищутся только в корневом каталоге сервера в папке udf. Для того, чтобы запретить использование udf, нужно выставить значение параметра равным «`None`».

При указании каталогов могут быть использованы как абсолютные, так и относительные пути. Относительные пути берутся от корневого каталога инсталляции сервера «Ред База Данных». В качестве разделителя директорий используется символ «;».

```
UdfAccess = Restrict UDF
```

Неконтролируемая возможность использования внешних функций может быть использована для того, чтобы поставить под угрозу безопасность как баз данных, так и всей системы (например, CVE-2017-11509). Поэтому настоятельно рекомендуется использовать данный параметр для ограничения директорий размещения udf-библиотек или отключить его совсем.

TempDirectories

С помощью параметра `TempDirectories` можно задать каталог, в котором сервер «Ред База Данных» будет хранить временные данные, допускающие разбиение на части. Это кэш курсоров, буферов записей, undo-логов, а также временные BLOB, блоки сортировок, данные мониторинга.

Параметр `TempDirectories` имеет строковый тип; значение по умолчанию равно пустой строке. Если параметр `TempDirectories` не активен, то путь к временному каталогу определяется исходя из значения переменных окружения `FIREBIRD_TMP, TEMP, TMP`.

В качестве значения параметра может быть задан путь к одному или нескольким каталогам. Для папок допускаются как абсолютные, так и относительные пути. Относительные пути берутся от корневого каталога инсталляции сервера «Ред База Данных». Если требуется определить несколько временных каталогов, то в качестве разделителя используется символ «;».

Если указана одна или несколько директорий, то выгрузка временных данных при сортировке будет осуществляться в указанные каталоги по очереди (если в текущей временной директории не осталось места, то временные файлы будут сохраняться в следующую по списку)

```
TempDirectories = c:\temp
TempDirectories = c:\temp;d:\temp
```

TempTableDirectory

Параметр `TempTableDirectory` задает каталог, в котором сервер «Ред База Данных» будет хранить данные временных таблиц и блобов. Если параметр не задан или указанный каталог недоступен, будет использован каталог из переменных окружения `FIREBIRD_TMP, TEMP, TMP`.

BlobTempSpace

Параметр `BlobTempSpace` устанавливает место, в котором будут сохраняться данные `BLOB`. Параметр имеет логический тип. По умолчанию его значение равно `false`. Если значении равно `true`, то `BLOB`, для которых явно не задано хранилище, будут создаваться во временном пространстве (таком же, как для данных `GTT`). Если значение `false`, то `BLOB`, для которых явно не задано хранилище, будут создаются в файле базы данных. `BLOB`, созданные пользовательскими приложениями, не затрагиваются, даже если хранилище не указано в `BPB`.

```
BlobTempSpace = false
```

SeparateTempTableFiles

Параметр `SeparateTempTableFiles` в файле конфигурации firebird.conf позволяет использовать для разных соединений разные файлы временных таблиц, чтобы уменьшить их разрастание. Оказывает влияние только при использовании архитектуры `Super`. Параметр имеет логический тип. По умолчанию его значение равно 1. В этом случае данные глобальных временных таблиц и временных `BLOB` будут размещаться в отдельном файле для каждого соединения. Если выставить значение параметра равным 0, то для всех соединений будет использоваться один файл

```
SeparateTempTableFiles = 1
```

AuditTraceConfigFiles

Параметр `AuditTraceConfigFile` в файле конфигурации `firebird.conf` задает имя и расположение файла с настройками системного аудита. Этот параметр имеет строковый тип и по умолчанию указывает на `fbtrace.conf`. Пустое значение параметра означает, что системный аудит выключен.

Есть возможностью указания ссылок на другие конфигурационные файлы. Каждый из них имеет такую же структуру, что и `fbtrace.conf`, и на их основе создаются дополнительные системные сессии аудита. Имеются три таких набора конфигураций для раздельного аудита:

- Производительности (`fbtrace.conf`)
- Событий безопасности (`fbtrace_sec.conf`)
- Действий пользователя `SYSDBA` (`fbtrace_dba.conf`)

```
AuditTraceConfigFiles = fbtrace.conf; fbtrace_sec.conf;
```

MaxUserTraceLogSize

Задает максимальный суммарный размер (в мегабайтах) временных файлов, создаваемых сессией пользовательской трассировки Services API. После прочтения временного файла приложением он автоматически удаляется. Параметр имеет целочисленный тип. Единица измерения - мегабайты. По умолчанию максимальный размер файла вывода ограничен 10 МБ. Если значения ограничения `MaxUserTraceLogSize` достигнуто, то сервер автоматически приостанавливает сессию слежения.

```
MaxUserTraceLogSize = 10
```

DefaultDbCachePages

Параметр `DefaultDbCachePages` используется для настройки количества страниц, которые могут быть удержаны кэшем, из расчета на каждую базу данных. Суперсервер использует единый кэш (32768 страниц) для всех подключений. Классик создает отдельный кэш (по умолчанию 1024 страниц) для каждого соединения.

Для архитектуры суперсервер значение параметра рекомендуется рассчитать по формуле:

$$\texttt{DefaultDbCachePages} = \frac{\texttt{MemorySize}}{4 * \texttt{PageSize}},$$

где `MemorySize` - общий объём памяти;
`PageSize` - объём страницы.

Для архитектуры классик значение параметра рекомендуется рассчитать по формуле:

$$\texttt{DefaultDbCachePages} = \frac{\texttt{MemorySize}}{4 * \texttt{ConnNum} * \texttt{PageSize}},$$

где `MemorySize` - общий объём памяти;
`PageSize` - объём страницы;
`ConnNum` - предполагаемое максимальное количество соединений.

При изменении данного параметра следует учитывать особенности аппаратно-программной платформы, других настроек сервера (`TempBlockSize`). Также, определение оптимальных для конкретной задачи настроек кэширования данных сервером «Ред База Данных» может быть произведено экспериментальным путем.

Параметр имеет целочисленный тип. Единица измерения – страница базы данных. По умолчанию параметр имеет значение 32768. Максимальное значение 2147483647 страниц. Минимальное значение параметра – 0. Если значение параметра равно нулю, то сервер не будет выполнять кэширование страниц данных.

```
DefaultDbCachePages = 32768
```

DatabaseGrowthIncrement

Параметр позволяет указать объем дискового пространства, которое может быть выделено под базу данных. Дисковое пространство резервируется в системе, что позволяет в дальнейшем снизить физическую фрагментацию файла (-ов) базы данных и дает возможность продолжить работу в условиях недостатка места на диске. Если режим резервирования включен, то сервер резервирует 1/16 часть от уже используемого дискового пространства для одного соединения, но не меньше 128 KB и не больше, чем значение, заданное параметром `DatabaseGrowthIncrement` (по умолчанию 128 MB).

Для отключения резервирования дискового пространства необходимо выставить значение `DatabaseGrowthIncrement` равным 0.

```
DatabaseGrowthIncrement = 134217728
```

Пространство под теневые копии баз данных не резервируется.

FileSystemCacheThreshold

Параметр `FileSystemCacheThreshold` устанавливает порог использования кэша файловой системы сервером «Ред База Данных». Значение параметра `FileSystemCacheThreshold` определяет максимально допустимое количество страниц, которые могут находиться в кэш-памяти одновременно. Кэш файловой системы будет использоваться, если размер выделенного для базы данных страничного кэша меньше, чем значение параметра `FileSystemCacheThreshold`.

Параметр имеет целочисленный тип. Единица измерения – страница базы данных (определяется при создании БД, может иметь размер от 4 до 16 Кб). По умолчанию параметр имеет значение - 65536 страниц. Максимально допустимое значение параметра – 2147483647. Минимальное значение параметра – 0. Если значение параметра `FileSystemCacheThreshold` меньше `DefaultDbCachePages`, то сервер не будет использовать кэш файловой системы. Значение `DefaultDbCachePages` будет взято из заголовка базы данных, если оно равно 0, то из databases.conf. Если в `databases.conf` значение параметра `DefaultDbCachePages` не установлено, то оно будет взято из firebird.conf.

```
FileSystemCacheThreshold = 65536
```

FileSystemCacheSize

Параметр `FileSystemCacheSize` устанавливает максимальный размер оперативной памяти, используемый системным файловым кэшем 64-битными Windows XP или Windows Server 2003 с Service Pack 1 или выше.

Параметр содержит целое число, представляющее собой количество (в процентах) оперативной памяти, которое может быть использовано под файловый кэш. Значение может быть от 10 до 95%. Если задать значение 0, операционная система сама будет определять размер файлового кэша. Это и есть значение по умолчанию.

```
FileSystemCacheSize = 0
```

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

Если Firebird запущен как приложение или в режиме `Embedded` или установлен не из официального дистрибутива, учетная запись может не иметь данной привилегии. Процесс не

выдаст ошибку при запуске, а просто запишет соответствующее сообщение в файл firebird.log и будет работать с настройками операционной системы.

RemoteFileOpenAbility

Параметр имеет логический тип. По умолчанию его значение равно 1. В этом случае сервер «Ред База Данных» может открыть базу данных, только если она сохранена на физическом диске компьютера, на котором запущен сервер. Запросы на подключениях с базами данных, сохраненными на сетевых дисках, переадресовываются на сервер «Ред База Данных», работающий на компьютере, которому принадлежит диск.

Это ограничение предотвращает возможность того, чтобы две различных копии сервера открыли одновременно одну и ту же базу данных. Нескоординированный доступ нескольких копий сервера к одной базе данных может привести к ее повреждению. Блокировка файла на системном уровне предотвращает нескоординированный доступ к файлу базы данных. Для отключения этой опции следует выставить значение параметра `RemoteFileOpenAbility` равным 0 (ложь).

Этот параметр может вызвать неисправимое повреждение базы данных. Не используйте эту опцию, если вы не понимаете рисков потери данных.

Сетевая файловая система не обеспечивает надежного способа координации доступа к файлам. Если вторая копия сервера соединится с базой данных сохраненной на сетевом диске, то это может повредить базу данных

```
RemoteFileOpenAbility = 0
```

TempBlockSize

Параметр `TempBlockSize` используется для управления пространством временных каталогов. Временные каталоги используются для выгрузки результатов обработки больших объемов данных (например, при сортировке данных). Параметр `TempBlockSize` определяет минимальный размер блока, выделяемого на один запрос с сортировкой.

Параметр имеет целочисленный тип. Единица измерения - байты. По умолчанию параметр имеет значение 1048576 байт. Максимально допустимое значение 2147483647 байт. Минимальное значение параметра – 0.

```
TempBlockSize = 1048576
```

TempCacheLimit

Параметр `TempCacheLimit` используется для ограничения объёма временного пространства, которое может быть кэшировано в оперативной памяти. В этом пространстве будут сохраняться данные сортировок, буферов записи, небольших временных блобов перед материализацией и т.д.

Параметр имеет целочисленный тип. В архитектуре Классик это ограничение на одно подключение и по умолчанию значение равно 8 Мб. В архитектуре Супер это ограничение на все подключения и по умолчанию значение равно 64 Мб. Максимально допустимое значение $(2^{64} - 1)$ байт. Минимальное значение параметра равно 0.

В целях увеличения производительности желательно установить это значение больше суммарного размера временных файлов в каталогах `TempDirectories`, если это позволяет объем доступной оперативной памяти.

```
TempCacheLimit = 67108864
```

AuthServer и AuthClient

Параметр `AuthServer` - набор методов аутентификации, разрешенных на сервере (определяется в файле конфигурации сервера).

Параметр `AuthClient` - набор методов аутентификации, поддерживаемых клиентом (определяется в файле конфигурации на клиенте).

Включенные методы перечисляются в виде строковых символов, разделенных запятыми, точками с запятой или пробелами. Если проверить подлинность с помощью первого метода не удалось, то сервер переходит к следующему и т.д. Если ни один метод не подтвердил подлинность, то пользователь получает сообщение об ошибке.

Ред База Данных 3.0 поддерживает следующие методы аутентификации:

- Безопасная парольная аутентификация использующая алгоритм хэширования SHA для передачи данных: `Srp, Srp224, Srp256, Srp384, Srp512`. По умолчанию используется Srp256;
- Традиционная (`Legacy_Auth`) аутентификация;
- Доверительная (`Win_Sspi`) аутентификация для ОС Windows;
- Многофакторная(`Multifactor`) аутентификация с применением политик безопасности;
- Доверенная аутентификация через механизм GSSAPI (`Gss`);
- Доверенная аутентификации для выполнения Execute Statement On External без указания логина и пароля (`ExtAuth`).

По умолчанию используется метод Secure remote passwords (`Srp`), оперирующий соответствующими плагинами ОС (`libSrp.s0 | Srp.dll | Srp.dylib`).

```
AuthServer = Srp
AuthClient = Srp, Srp256, Win_Sspi, Legacy_Auth, Multifactor, ExtAuth
```

Если вы хотите использовать плагины аутентификации, которые не предоставляют ключа шифрования (`Win_Sspi`, `Legacy_Auth`, `Gss`), то следует отключить обязательное (`Required`) шифрование каналов передачи данных (параметр `WireCrypt`), кроме случаев, когда вы работаете с протоколом XNET.

Чтобы отключить какой-нибудь из методов, раскомментируйте строку и удалите нежелательный метод из списка.

Оба параметра могут быть использованы в `databases.conf`. Они могут использоваться как в DPB, так и в SPB для конкретных настроек соединения.

UserManager

Устанавливает плагин, который будет работать в базе данных безопасности. Это может быть список с пробелами, запятыми или точками с запятой в качестве разделителей: используется первый подключаемый модуль из списка. Всего существует четыре возможных плагина:

- Srp;
- Legacy_UserManager;
- Multifactor_Manager;
- Ldap.

Для поддержки старой базы данных безопасности и управления пользователями в ней, следует установить значение параметра `Legacy_UserManager`.

Для создания многофакторныхпользователей и управления ими следует установить значение параметра `Multifactor_Manager`.

Для получения списка пользователей каталога `LDAP` с помощью псевдотаблицы `SEC$USERS` следует установить значение параметра `Ldap`.

В SQL операторах управления пользователями можно явно указать какой плагин будет использоваться.

```
UserManager = Srp
```

Одноименные пользователи, созданные с помощью разных плагинов управления пользователями — это разные пользователи.

Параметр `UserManager` можно использовать в `database.conf` для переопределения в конкретной базе данных.

DefaultUserManagers

Позволяет указать набор стандартных плагинов.

Если предложение `USING PLUGIN` не указано, то при создании пользователя он сам добавляется во все стандартные плагины (со всеми указанными атрибутами).

При изменении пароля пользователя он меняется во всех стандартных плагинах. Если в каком-либо стандартном плагине нет пользователя, то он добавляется.

Если меняется какой-либо другой атрибут, то он также меняется и в других стандартных плагинах, но если пользователь отсутствует, то он не создаётся.

При удалении пользователя он также удалится из всех стандартных плагинов.

Пример использования:

```
UserManager = Srp Legacy_UserManager Multifactor_Manager
DefaultUserManagers = Legacy_UserManager Multifactor_Manager
```

Значение по умолчанию:

```
DefaultUserManagers =
```

Можно использовать только те плагины, которые были указаны в параметре UserManager

TrustedUser

Логин доверенного пользователя, который при подключении может указать имя любого другого пользователя и подключиться к БД с этим именем. Работает для всех плагинов аутентификации. По умолчанию параметр не задан, т.е. подменять логин при подключении запрещено.

```
TrustedUser = SomeTrust
```

TracePlugin

Задает плагин, используемый функцией трассировки Firebird для отправки данных трассировки в приложение клиента или данных аудита в лог файл.

```
TracePlugin = fbtrace
```

WireCryptPlugin

Плагин поточного шифра используется для шифрования и дешифрования данных, передаваемых по сети.

```
WireCryptPlugin = Arc4
```

По-умолчанию устанавливается значение параметра `Arc4`, что означает использование плагина потокового шифра `Alleged RC4`. Возможно установить значение параметра `Wire_WinCrypt`. Сконфигурированный плагин, который требует ключ, сгенерированный настроенным подключае-

мым модулем аутентификации, может быть переопределен в API для конкретного соединения через DPB или SPB.

KeyHolderPlugin

Этот параметр представляет собой некоторую форму временного хранилища для ключей шифрования базы данных.

Реализованного плагина по-умолчанию нет, но образец для Linux под названием `libCryptKeyHolder_example.so` можно найти в папке `/plugins/`.

LdapPlugin

Используется для получения информации об учётной записи с сервера LDAP вместо базы данных безопасности.

AllowEncryptedSecurityDatabase

Этот параметр позволяет использовать зашифрованную базу данных безопасности.

Если полагаться на шифрование сетевого канала посредством ключа, сгенерированного плагином аутентификации (например, SRP), чтобы передавать ключи шифрования базы данных по этому каналу, то использование зашифрованных баз данных безопасности является своего рода порочным кругом. Для того, чтобы отправить ключ шифрования базы данных по сетевому каналу безопасным путем, канал должен быть уже зашифрован, но для этого требуется сетевой ключ шифрования от плагина аутентификации, которому необходимо открыть базу данных безопасности для проверки хэша, которая, в свою очередь, требует ключа шифрования БД. К счастью, в большинстве случаев нет необходимости шифровать базу данных безопасности - она неплохо защищена сама по себе, если вы используете криптостойкие пароли. Но в некоторых случаях желательно иметь зашифрованную базу данных безопасности, например, если кто-то хочет использовать в качестве собственной БД безопасности зашифрованную базу. В этом случае следует зашифровать ключ, прежде чем передавать его на сервер с помощью функции обратного вызова. Перед включением этого параметра убедитесь, что ваши ключи хорошо зашифрованы. Учтите, что при включении этой опции незашифрованная передача ключа может случиться даже с незашифрованной БД безопасности.

Эта функция не поддерживается традиционным плагином аутентификации - если вы заботитесь о безопасности, никогда не используйте традиционную аутентификацию.

```
AllowEncryptedSecurityDatabase = false
```

Providers

Провайдеры - это практически то, что мы подразумеваем под способами, используемыми для соединения клиента с сервером, т.е. через интернет; на том же компьютере через 'localhost'; или через прямое соединение в локальной сети (старый `libfbembed.so` для POSIX сейчас улучшен как библиотека `libEngine12.so`;для Windows – `engine12.dll`;для MacOSX – `engine12.dylib`).

В `firebird.conf` доступны по-умолчанию следующие провайдеры:

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

В `databases.conf` один или несколько провайдеров могут быть заблокированы, если вставить и раскомментировать строку из `firebird.conf` и удалить нежелательные провайдеры.

Хотя это ключевая особенность версии 3.0, архитектура провайдеров не нова. Провайдеры присутствуют во всех предыдущих версиях Ред Базы данных, хотя и хорошо скрыты. Они были введены изначально для решения задачи, которая с тех пор выполнялась программными интерфейсами ODBC, ADO, BDE и т.д., чтобы обеспечить доступ к различным базам данных с помощью одного внешнего интерфейса.

Впоследствии эта архитектура провайдеров (известная как Open Systems Relational Interface, OSRI) также показала себя очень эффективной для поддержки сочетания старых и новых форматов базы данных на одном сервере, имеющем смешанные подключения к локальным и удаленным базам данных.

Провайдеры, реализованные в Ред Базе Данных 3.0, позволяют поддерживать все эти режимы (удаленные соединения, базы данных с разными ODS), а также замыкание (chaining) провайдеров. Замыкание - это термин для ситуации, когда провайдер использует обратный вызов стандартного API при выполнении операции над базой данных.

Главным элементом архитектуры провайдеров является `y-valve`. На начальном этапе вызова `attach` или `create database y-valve` просматривает список известных провайдеров и вызывает их по одному, пока один из них не завершит запрошенную операцию успешно. Для соединения, которое уже установлено, соответствующий провайдер вызывается сразу с почти нулевыми накладными расходами.

Рассмотрим пример работы `y-valve`, когда он выбирает подходящего провайдера при подключении к базе данных. По-умолчанию имеется три провайдера: `Remote, Engine12, Loopback`.

Типичная конфигурация клиента работает таким образом: при подключении к базе данных с именем `RemoteHost: dbname` (синтаксис TCP/IP) или `\\RemoteHost\dbname` (NetBios), провайдер `Remote` обнаруживает явный синтаксис сетевого протокола и перенаправляет вызов `RemoteHost`.

Когда `<имя базы данных>` не содержит сетевого протокола, а только имя базы данных, провайдер `Remote` отклоняет его, а провайдер `Engine12` выходит на первый план и пытается открыть файл с именованной базой данных. Если это проходит успешно, создается подключение к базе данных.

Но что происходит, если СУБД возвращает ошибку при попытке подключения к базе данных?

- Если файл базы данных, к которому нужно подключиться, не существует, то в этом нет интереса.
- Встроенное соединение может не работать, если пользователь, подключившийся к нему, не имеет достаточных прав для открытия файла базы данных. Это было бы обычной ситуацией, если бы база данных не была создана этим пользователем во встроенном режиме или если ему явно не были предоставлены права ОС на встроенный доступ к базам данных.
- После отказа провайдера `Engine12` в получении доступа к базе данных, пытается подключиться провайдер `Loopback`. Он не очень отличается от `Remote`, за исключением того, что он пытается получить доступ к именованной базе данных `<dbname>` на сервере с сетевым интерфейсом «внутренней петли» (loopback) в сетевом протоколе TCP/IP.

Провайдеры не ограничены тремя вышеперечисленными. Версия 3.0 не поддерживает pre-ODS 12 провайдер. Удаление поддержки старых форматов из движка помогает упростить его код и выиграть немного в скорости. Принимая во внимание, что это увеличение скорости иногда имеет место в критических для производительности местах, таких как поиск ключа в индексном блоке действительно делает работу СУБД быстрее.

Тем не менее, архитектура провайдеров делает возможным доступ к старым базам данных при переходе на более высокую версию Ред Базы Данных.

DeadlockTimeout

Значение параметра `DeadlockTimeout` определяет, сколько секунд будет ждать менеджер блокировок после возникновения конфликта до его разрешения.

Параметр имеет целочисленный тип. Единица измерения - секунды. Значение по умолчанию равно 10 секунд. Минимально допустимое значение параметра равно 0. Максимально допустимое значение равно 2147483647.

Слишком большое значение параметра может ухудшить производительность системы.

```
DeadlockTimeout = 10
```

MaxUnflushedWrites

Параметр `MaxUnflushedWrites` определяет, как часто страницы из кэш памяти будут выгружаться на жесткий диск (активен только при значении параметра `ForcedWrites=Off`).

Значение параметра `MaxUnflushedWrites` определяет максимальное количество не выгруженных на диск страниц, накопившихся в кэш-памяти до подтверждения транзакции.

Параметр имеет целочисленный тип и измеряется в страницах. Значение по умолчанию равно 100 страниц. Для не Win32 систем значение по умолчанию является -1(Отключено). Максимально допустимое значение равно 2147483647.

```
MaxUnflushedWrites = 100
```

Чем больше значение параметра, тем выше вероятность потери данных при возникновении аппаратного сбоя в системе.

MaxUnflushedWriteTime

Параметр `MaxUnflushedWriteTime` определяет, как часто страницы из кэш памяти будут выгружаться на жесткий диск (активен только при значении параметра `ForcedWrites=Off`).

Значение параметра `MaxUnflushedWriteTime` определяет время, по истечении которого страницы данных, ожидающие подтверждения транзакции в кэш-памяти, будут выгружены на диск.

Параметр имеет целочисленный тип и измеряется в секундах. Значение по умолчанию равно 5 секунд. Для не Win32 систем значение по умолчанию является -1 (Отключено). Максимально допустимое значение равно 2147483647.

```
MaxUnflushedWriteTime = 5
```

Чем больше значение параметра, тем выше вероятность потери данных при возникновении аппаратной ошибки в системе.

BugcheckAbort

Опция `BugcheckAbort` определяет, прерывать ли работу сервера при возникновении внутренней ошибки или снимать дамп ядра для последующего анализа.

Параметр имеет логический тип. Возможные значения 0 и 1. Значение по умолчанию равно 0, в этом случае механизм снятия дампов отключен.

```
BugcheckAbort = 0
```

RelaxedAliasChecking

Параметр `RelaxedAliasChecking` позволяет снять ограничение на использование псевдонимов имен таблиц в запросах. Использование псевдонимов имен таблиц позволяет выполнять подобные запросы:

```
SELECT TABLE.X FROM TABLE A
```

Параметр имеет логический тип. Значение по умолчанию равно 0. Если значение параметра равно 1, то ограничение на использование псевдонимов таблиц в запросах снимается.

```
RelaxedAliasChecking = 0
```

ConnectionTimeout

С помощью параметра `ConnectionTimeout` устанавливается ограничение на время ожидания соединения. После того как порог, установленный значением параметра, будет превышен, попытка соединения будет признана неудачной.

Параметр `ConnectionTimeout` имеет целочисленный тип и измеряется в секундах. Значение

по умолчанию равно 180 секунд. Минимальное значение равно 0. Максимально допустимое значение равно 2147483647.

```
ConnectionTimeout = 180
```

WireCrypt

Параметр устанавливает, следует ли шифровать сетевое соединение. Он может принимать три возможных значения: `Required, Enabled, Disabled`. По-умолчанию установлено, что шифрование является обязательным (`Required`) для подключений, поступающих на сервер и включенным (`Enabled`) для подключений, исходящих с сервера.

```
WireCrypt = Enabled (for client) / Required (for server)
```

Чтобы получить доступ к серверу с использованием более старой клиентской библиотеки, параметр `WireCrypt` в файле конфигурации сервера должен быть включен (`Enabled`) или выключен (`Disabled`).

Правила очень просты: если на одной стороне стоит значение `WireCrypt = Required`, а на другой установлено значение `Disabled`, то первая сторона отклоняет соединение и оно не устанавливается. Если на одной стороне стоит значение `WireCrypt = Enabled`, то на другой шифрования может и не быть вовсе.

Отсутствующий подключаемый модуль `WireCrypt` или ключ шифрования в случаях, когда канал должен быть зашифрован, также препятствует соединению.

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

Таблица 6.1 — Совместимость параметров `WireCrypt` на клиенте и на сервере

	DISABLED	ENABLED	REQUIRED
DISABLED	Шифрование отключено	Шифрование отключено	Ошибка соединения
ENABLED	Шифрование отключено	Шифрование включено, если плагин аутентификации предоставляет ключ шифрования. Иначе шифрования нет.	Шифрование включено, если плагин аутентификации предоставляет ключ шифрования. Иначе ошибка подключения.
REQUIRED	Ошибка соединения	Шифрование включено, если плагин аутентификации предоставляет ключ шифрования. Иначе ошибка подключения.	Шифрование включено, если плагин аутентификации предоставляет ключ шифрования. Иначе ошибка подключения.

WireCompression

Параметр может быть задействован и в `firebird.conf` и в `databases.conf`; он включает или отключает сжатие данных, передающихся по проводам.

По-умолчанию параметр отключен.

```
WireCompression = false
```

Для правильной работы параметра требуется корректная настройка как на сервере, так и на клиенте:

- Чтобы включить `WireCompression` на стороне сервера, поставьте параметр в значение `true` в файле `firebird.conf` или `database.conf`.
- Для того, чтобы активировать `WireCompression` на стороне клиента, передайте соответствующий тег в вызов DPB и SPB:

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

- Версии сервера и клиента должны быть не ниже Ред Базы Данных 3.0

DummyPacketInterval

Параметр `DummyPacketInterval` используется для того, чтобы установить число секунд ожидания в «тихом» режиме, прежде чем сервер начнет посылать пустые пакеты для подтверждения соединения.

Параметр имеет целочисленный тип и измеряется в секундах. Значение по умолчанию равно 0 секунд. Максимально допустимое значение равно 2147483647 секунд.

```
DummyPacketInterval = 0
```

Не используйте данный параметр в windows-системах, если в ней запущены TCP/IP клиенты. Это может привести к постоянному увеличению объема использования не страничной памяти ядра, что может привести к зависанию или аварийному отказу системы.

В Windows это - единственный способ обнаружить и разъединить неактивных клиентов при использовании NetBEUI, XNET или IPC протоколов.

Сервер «Ред База Данных» использует опцию разъема SO_KEEPALIVE, чтобы следить за активными подключениями по TCP/IP протоколу. Если вас не устраивает заданное по умолчанию 2-часовое время ожидания (`keepalive`), то следует изменить параметры настройки своей операционной системы соответственно:

В операционных системах семейства Posix отредактируйте файл:
`/proc/sys/net/ipv4/tcp_keepalive_*`.

RemoteServiceName, RemoteServicePort

Параметры `RemoteServiceName` и `RemoteServicePort` используются для установки номера порта или имени сервиса, которые будут использоваться для клиентских баз данных.

Параметр `RemoteServiceName` имеет строковый тип. Значение по умолчанию равно «`gds_db`».

Параметр `RemoteServicePort` имеет целочисленный тип. Значение по умолчанию равно 3050.

```
RemoteServiceName = gds_db
RemoteServicePort = 3050
```

Изменять следует только один из этих параметров, не оба сразу. Сервер ищет номер порта для клиентских соединений в следующем порядке – сначала `RemoteServiceName` (соответствующая значению параметра запись ищется в файле «`cervices`»), затем `RemoteServicePort`.

RemoteAuxPort

Параметр `RemoteAuxPort` определяет номер TCP-порта, который будет использоваться для передачи уведомлений о событиях сервера.

Параметр `RemoteAuxPort` имеет целочисленный тип. Значение по умолчанию равно 0. В этом случае номер порта будет выбираться случайно.

```
RemoteAuxPort = 0
```

Для сервера «Ред База Данных» архитектура Классик номер порта в любом случае будет выбираться случайно, независимо от значения параметра `RemoteAuxPort`.

TcpRemoteBufferSize

Параметр `TcpRemoteBufferSize` определяет размер TCP/IP пакета для обмена сообщениями между сервером и клиентом. Чем больше размер пакета, тем больше данных будет передаваться за одну передачу.

Параметр имеет целочисленный тип и измеряется в байтах. Значение по умолчанию равно 8192. Минимально допустимое значение равно 1448. Максимальное значение равно 32767.

```
TcpRemoteBufferSize = 8192
```

TcpNoNagle

Параметр `TcpNoNagle` используется для настройки пакетов в TCP/IP сетях. В Linux по умолчанию библиотека сокетов минимизирует количество физических записей путем буферизации записей перед фактической передачей данных. Для этого используется встроенный алгоритм, известный как Nagle's Algorithm. Он был разработан, для того, чтобы избежать проблем с маленькими пакетами в медленных сетях.

Параметр имеет логический тип. По умолчанию значение параметра равно 1 (истина). В этом случае буферизация не используется. На медленных сетях в Linux это позволяет увеличить скорость передачи.

```
TcpNoNagle = 1
```

IPv6V6Only

Этот параметр можно устанавливать только в `firebird.conf`. Ред База Данных поддерживает IPv6 подключение на стороне сервера и клиента. Параметр может принимать значения `true/false, 1/0` или `Yes/No`.

Сервер

По-умолчанию, сервер прослушивает пустой IPv6 адрес (::) и принимает все входящие подключения, будь то IPv4 или IPv6 (`IPv6V6Only = false`). Если параметр установлен в true, сервер, прослушивая явно или неявно пустой IPv6 адрес, принимает только IPv6 подключения.

Клиент

Адреса IPv6 отображаются как восемь четырёхзначных шестнадцатеричных чисел (то есть групп по четыре символа), разделённых двоеточием. В строке подключения необходимо заключать IPv6 адрес в квадратные скобки, чтобы разрешить неоднозначность с использованием двоеточия в качестве разделителя между IP адресом хоста и путем к базе данных. К примеру:

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

RemoteBindAddress

Параметр `RemoteBindAddress` указывает адрес сетевого интерфейса, с которого будут разрешены входящие подключения. При этом все входящие соединения через другие сетевые интерфейсы будут запрещены.

Параметр имеет строковый тип. По умолчанию его значение равно пустой строке (разрешены соединения с любого сетевого интерфейса).

```
RemoteBindAddress =
```

LockMemSize

Значение параметра `LockMemSize` определяет объем памяти, которая будет выделена менеджеру блокировок. В архитектуре Классик данный параметр используется для начального распределения, далее таблица расширяется динамически до предела памяти. В архитектуре Супер значение

параметра определяет начальное распределение и предел выделяемой памяти.

Параметр имеет целочисленный тип. Единица измерения – байты. Значение по умолчанию равно 1048576 байт. Минимальное значение равно 0.

Максимально допустимое значение равно 2147483647.

```
LockMemSize = 1048576
```

Размер таблицы блокировок влияет на:

1. Размер кэша страниц базы данных. Страница, помещенная в кэш, блокируется, как минимум, один раз, страницы, которые читаются несколькими клиентами, могут блокироваться несколько раз (архитектура Классик).
2. Число одновременных транзакций. Каждая транзакция имеет блокировку, которая ее идентифицирует. Блокировка используется для синхронизации транзакций, а также для того, чтобы распознать случаи, когда транзакция завершилась без подтверждения или отката.
3. События. Механизм оповещения о событиях основывается на блокировках. Число событий и число клиентов, ожидающих эти события, влияют на размер таблицы блокировок.

LockAcquireSpins

В архитектуре сервера классик только одно клиентское соединение может обратиться к таблице блокировки в одно и то же время. Доступ к таблице блокировки управляется с помощью `mutex(a)`. `Mutex` может быть затребован в условном, либо безусловном режиме. Если `mutex` затребован в условном режиме, то ожидание является отказом, и запрос должен повториться. В безусловном режиме `mutex` будет ожидаться до тех пор, пока не будет получен.

Параметр `LockAcquireSpins` имеет целочисленный тип. Его значение устанавливает количество попыток, которые будут сделаны в условном режиме. По умолчанию значение параметра равно 0, в этом случае будет использоваться безусловный режим.

Параметр имеет эффект только на SMP (симметричных мультипроцессорных) системах.

```
LockAcquireSpins = 0
```

LockHashSlots

Параметр `LockHashSlots` используется для настройки числа слотов хэширования блокировок. Чем больше слотов используется, тем короче получаются цепочки хэширования, что увеличивает производительность при повышенной нагрузке.

Параметр имеет целочисленный тип. По умолчанию значение параметра равно 65521. В качестве значения рекомендуется указывать простое число, чтобы хэш-алгоритм производил хорошее распределение.

```
LockHashSlots = 65521
```

Увеличение значения данного параметра необходимо только при высокой загрузке (одновременно с ним следует увеличить и параметр `LockMemSize` на тот же процент). Он вычисляется с использованием утилиты `Lock Print` по следующему принципу.

Запускаем утилиту

```
rdb_lock_print -d <database> | <alias>
```

В группе заголовка блока (`LOCK_HEADER BLOCK`), которая описывает основную конфигурацию и состояние таблицы блокировок, смотрим значение элемента `Hash lengths` (длина цепочки хэширования). Этот элемент сообщает минимальную, среднюю и максимальную длину цепочки слотов.

Чем длиннее будут цепочки, тем медленнее будет работать менеджер блокировок. Если среднее значение больше 3 или максимальное больше 10, то это означает, что слотов недостаточно. Поэтому следует увеличить параметр `LockHashSlots` в 2-3 раза (при этом взять простое число).

> Для применения параметра необходимо, чтобы сервер пересоздал таблицу блокировок (при этом в системе не должно остаться подключений и старой таблицы блокировок).

EventMemSize

Значение параметра `EventMemSize` определяет объем разделяемой памяти, которая будет выделена менеджеру событий.

Параметр `EventMemSize` имеет целочисленный тип. Единица измерения – байты. Значение по умолчанию равно 65356. Минимально допустимое значение равно 0. Максимальное значение равно 2147483647.

```
EventMemSize = 65536
```

OptimizationStrategy

Значение параметра `OptimizationStrategy` определяет стратегию оптимизации запросов. Параметр имеет строковый тип. Может принимать следующие значения:

- `default` - используется стратегия по умолчанию (является значением по умолчанию);
- `first` - для запросов выбирается такой план доступа, который позволяет максимально быстро получить первые записи в выборке;
- `all` - для запросов выбирается такой план доступа, который предполагает, что в запросе будут выбраны все записи.

```
OptimizationStrategy = default
```

PlanRecursionLimit

Параметр `PlanRecursionLimit` устанавливает максимальный уровень вложенности при выводе планов. Значение по умолчанию равно 0, оно означает, что вложенные планы выводиться не будут.

```
PlanRecursionLimit = 0
```

ReadConsistency

Параметр обеспечивает согласованность чтения данных для запросов в режиме `READ COMMITTED`. При запуске такого запроса для него делается снимок версий, который будет использоваться запросом для чтения данных.

```
ReadConsistency = 1
```

DefaultIsolationLevel

Параметр `DefaultIsolationLevel` определяет уровень изоляции транзакций по умолчанию. Может принимать одно их трёх значений: `SNAPSHOT`, `SNAPSHOT TABLE STABILITY`, `READ COMMITTED`. По умолчанию установлено значение `SNAPSHOT`.

```
DefaultIsolationLevel = SNAPSHOT
```

Таблица 6.2 — Совместимость параметров DefaultIsolationLevel и ReadConsistency

DefaultIsolationLevel	ReadConsistency	Уровень изоляции
SNAPSHOT	1	SNAPSHOT
SNAPSHOT	0	SNAPSHOT
SNAPSHOT TABLE STABILITY	1	SNAPSHOT TABLE STABILITY
SNAPSHOT TABLE STABILITY	0	SNAPSHOT TABLE STABILITY
READ COMMITTED	1	READ COMMITTED RECORD_VERSION
READ COMMITTED	0	READ COMMITTED NO RECORD_VERSION

SecurityLog

Параметр SecurityLog определяет, где регистрировать события безопасности. Возможные значения параметра:

- local - регистрировать события безопасности в локальных логах сервера (firebird.log, журналы аудита);
- syslog - регистрировать события безопасности в syslog;
- both - регистрировать события безопасности в локальных логах сервера (firebird.log, журналы аудита) и в syslog.

```
SecurityLog = local
```

6.2 Настройки ядра

CpuAffinityMask

Параметр CpuAffinityMask позволяет указать, какие процессоры будут использоваться сервером (для для ОС Windows).

Параметр имеет эффект только в SMP (симметричных мультипроцессорных) системах.

Параметр имеет целочисленный тип. Значение параметра соответствует элементам битового массива, в котором каждый бит представляет центральный процессор. Таким образом, чтобы использовать только первый процессор, значение параметра должно быть равно 1. Чтобы использовать и центральный процессор 1, и центральный процессор 2 - 3. Чтобы использовать центральный процессор 2, и центральный процессор 3 - 6. Значение по умолчанию равно 0.

```
CpuAffinityMask = 0
```

GCPolicy

Параметр GCPolicy используется для управления работой «сборщика мусора». Параметр имеет строковый тип. Возможные значения параметра:

- background - сборщик мусора работает как фоновый, собирая мусор в отдельном потоке;
- cooperative - сборщик мусора работает в оперативном режиме, собирая мусор немедленно при чтении "мусорных" версий;

- `combined` - сборщик мусора работает в оперативном режиме, но если мусор собрать не удается, то о "замусоренных" страницах сигнализируется фоновому сборщику мусора.

По умолчанию в архитектуре Супер сервера «сборщик мусора» работает в комбинированном режиме. В архитектуре Классик сервера этот параметр игнорируется, а «сборщик мусора» всегда работает в оперативном режиме.

```
GCPolicy = combined
```

SecurityDatabase

Определяет имя и расположение базы данных безопасности, в которой хранятся имена пользователей и пароли, используемые сервером для проверки удаленных подключений.

По-умолчанию в `firebird.conf`:

```
SecurityDatabase = $(dir_secDb)/security3.fdb
```

Параметр может быть переопределен для определенной базы данных в файле `databases.conf`.

6.3 Настройки для многопоточной работы

MaxParallelWorkers

Максимальное количество параллельных потоков, которое может создать ядро. Этот параметр работает только для рестора (не для бэкапа). Для рестора параллельная обработка реализована только при построении индексов. По умолчанию `MaxParallelWorkers = 1`, т.е. параллельное создание индексов отключено, даже если в ресторе задан ключ `-PAR`.

Допустимые значения: от 1 до 64. Другие значения будут проигнорированы и по умолчанию использоваться 1.

```
MaxParallelWorkers = 16
```

ParallelWorkers

Число потоков, используемых рестором по умолчанию, если не задана опция `-PAR <n>` в `gbak`.

Допустимые значения: от 1 до `MaxParallelWorkers`. Другие значения будут проигнорированы и по умолчанию использоваться 1.

```
ParallelWorkers = 4
```

6.4 Настройки для Windows-систем

GuardianOption

Параметр определяет должен ли сторож (`Guardian`) запускать сервер после того, как его работа была завершена неправильно.

Параметр имеет логический тип. Значение по умолчанию равно 1 (истина). Для того, чтобы отключить "сторож" сервера следует выставить значение параметра равным 0 (ложь).

```
GuardianOption = 1
```

ProcessPriorityLevel

Параметр определяет уровень приоритетов процессов сервера «Ред База Данных». Параметр имеет целочисленный тип и может принимать значения:

- 0 – нормальный приоритет;
- положительное значение – повышенный приоритет;
- отрицательное значение – пониженный приоритет.

Все изменения данного параметра должны быть тщательно проверены, чтобы гарантировать, что сервер продолжает обрабатывать запросы.

```
ProcessPriorityLevel = 0
```

IpcName

Параметр `IpcName` определяет имя области разделяемой памяти используемой в качестве транспортного канала в локальном протоколе. Параметр имеет строковый тип. Значение по умолчанию равно `FIREBIRD`.

Локальный протокол в версиях 2.6, 2.5, 2.1 и 2.0 не совместим с любой предыдущей версией Firebird или InterBase.

```
IpcName = FIREBIRD
```

Сервер может регистрировать объекты в пространстве имен `Global`, только если он выполняется под учетной записью с привилегией `SE_CREATE_GLOBAL_NAME`. Это означает, что, если вы работаете под ограниченной учетной записью в Vista, XP SP2 или 2000 SP4, возможность использования локального протокола для других сеансов будет недоступна.

RemotePipeName

Параметр `RemotePipeName` определяет название канала (`Pipe`), используемого как транспортный канал в протоколе NetBEUI. Название канала в протоколе NetBEUI имеет то же самое значение, что и номер порта для протокола TCP/IP.

Параметр имеет строковый тип. Значение по умолчанию равно `interbas` и совместимо с InterBase /Firebird 1.

```
RemotePipeName = interbas
```

6.5 Настройки для Unix/Linux систем

Redirection

Параметр `Redirection` используется для отключения защиты от переадресации запросов на другие сервера. Возможность переадресации запросов на другие серверы изначально присутствовала в InterBase. Но она была исключена корпорацией Borland в InterBase 6.0 после доработки добавившей SQL-диалекты. Возможность перенаправления запросов была восстановлена в Firebird 2.0. Но на сегодняшний день использование этой возможности (прокси сервер) представляет угрозу безопасности. Например, вы используете защищенный сервер «Ред База Данных», доступ к которому осуществляется из глобальной сети. В этом случае, если у сервера есть доступ к локальной сети, то он будет исполнять роль шлюза для входящих запросов типа:

```
firebird.your.domain.com:internal_server:/private/database.fdb
```

При этом злоумышленнику достаточно знать имя или IP-адрес хоста вашей локальной сети, потому что для соединения не требуется знать логин и пароль на внешнем сервере. Такой шлюз позволяет обойти систему сетевой защиты, установленную в вашей локальной сети.

Параметр имеет логический тип. Значение по умолчанию равно 0 (ложь). В этом случае возможность перенаправления запросов отключена. Для включения этой опции следует значение параметра выставить равным 1 (истина).

```
Redirection = 0
```

6.6 Настройки архитектуры

ServerMode

Определяет архитектуру сервера.

- **Super:** исключительно один серверный процесс обслуживает все подключения, используя потоки для обработки запросов; общий пул для всех соединений; общий кэш страниц на уровне базы.
- **Superclassic:** базы данных открываются одним серверным процессом, но доступ не исключительный - embedded процесс может открыть одновременно ту же базу; подключения обрабатываются потоками, запущенными из общего пула, каждый из которых имеет свой собственный страничный кэш базы данных.
- **Classic:** отдельный процесс на каждое соединение с БД; каждая база данных может быть открыта несколькими процессами (включая локальные для embedded доступа); отдельный кэш страниц на каждое соединение

MemoryWipePasses

Параметр `MemoryWipePasses` используется для настройки необходимости и метода обезличивания освобождаемой сервером оперативной памяти и дискового пространства.

> При необходимости обезличивания памяти требуется указывать полный сетевой путь при соединении с БД.

Параметр имеет целочисленный тип. Значение по умолчанию равно 0, это означает, что обезличивание памяти отключено. Возможные значения:

- 0 — обезличивание не происходит;
- 1 — происходит обнуление памяти;
- >1 — происходит чередование записи 0xFF и 0x00 в освобождаемую память, последний проход при этом в любом случае заполняет блок нулями.

```
MemoryWipePasses = 0
```

MaxOpenFileBlobs

Параметр `MaxOpenFileBlobs` ограничивает число одновременно открытых файлов BLOB. Когда количество открытых файлов достигнет этого предела, некоторые из них будут закрыты. Нулевое значение означает, что предела нет.

```
MaxOpenFileBlobs = 0
```

6.7 Настройки LDAP

LDAPServer

Задаёт адрес сервера LDAP, используемый для хранения учётной информации пользователей. По умолчанию задано пустое значение, т.е. сервер LDAP не используется, и учётная информация ищется только в БД безопасности. Есть возможность задать резервные сервера, прописав их через ";". Если недоступен основной сервер, то будет выполнена попытка подключения к резервным серверам.

```
LDAPServer = 192.168.1.1
```

LDAPConnectionRetries

Параметр LDAPConnectionRetries задаёт максимальное количество попыток подключения к серверам, указанным в LDAPServer. Попытка считается неуспешной, если не удалось подключиться ни к одному серверу из списка. Минимальное значение равно 1. По умолчанию задано LDAPConnectionRetries = 3.

```
LDAPConnectionRetries = 3
```

LDAPDomain

Задаёт имя домена, в котором происходит аутентификация. Если аутентификация выполняется методом bind, то СУБД сначала попытается подключиться с именем пользователя в том виде, в котором его передал клиент (например, Ivan.Petrov). Если таким способом подключиться не получилось, при заданном параметре LDAPDomain, СУБД сделает bind ещё раз с именем пользователя в формате <имя пользователя>@<значение LDAPDomain> (например, Ivan.Petrov@example.com).

Также ветка, определяемая доменом в параметре LDAPDomain, будет использована как база для поиска пользователей, если не задан параметр LDAPUserBase.

```
LDAPDomain = example.com
```

LDAPEncryption

Тип шифрования, используемый при подключении к серверу LDAP. Может принимать три значения: None, SSL, TLS.

- None – незащищённое подключение к порту 389.
- SSL – подключение к LDAP через SSL к порту 636.
- TLS – подключение с TLS-шифрованием к порту 389.

По умолчанию используется незащищённое подключение.

Сертификат сервера LDAP будет проверяться, если параметр VerifyLdapServer не отключен.

```
LDAPEncryption = None
```

LDAPEncryptionMinVersion

Минимальная версия протокола шифрования, которая должна поддерживаться сервером LDAP. Может принимать следующие значения:

- 1.0, 1.1 и 1.2 - для шифрования с помощью TLS;

- 2.0 и 3.0 - для шифрования с помощью SSL.

Если указанная версия не поддерживается сервером LDAP, то соединение будет разорвано.

VerifyLdapServer

Включает или отключает проверку сертификата сервера LDAP при включении LDAPEncryption.

```
VerifyLdapServer = 1
```

LDAPUserDN

DN пользователя, от имени которого сервер будет подключаться к LDAP для аутентификации других пользователей. Этот пользователь должен иметь права на чтение атрибутов LDAP, используемых сервером «Ред База Данных». Если параметр не задан, сервер будет делать bind к LDAP с именем и паролем реального пользователя, чтобы пройти аутентификацию.

```
LDAPUserDN = uid=rdb,ou=people,dc=example,dc=com
LDAPUserDN = cn=rdb,cn=users,dc=example,dc=com
```

LDAPPassword

Пароль пользователя, определенного в атрибуте LDAPUserDN, от имени которого сервер будет подключаться к LDAP для аутентификации других пользователей.

```
LDAPPassword = 123qwe
```

LDAPUserBase

Ветка в LDAP, которая будет использована как стартовая для поиска пользователей. При аутентификации имена пользователей будут искаться в этой ветке и рекурсивно во всех ветках, находящихся в ней до первого совпадения. Также она будет использована для поиска пользователей при получении их групп или атрибутов.

Если этот параметр не задан, то в качестве базы для поиска пользователей будет использована ветка, определяемая доменом в параметре LDAPDomain.

```
LDAPUserBase = ou=people,dc=example,dc=com
LDAPUserBase = cn=users,dc=example,dc=com
```

LDAPUserPrefix

Название атрибута, в котором хранится имя пользователя в его DN в LDAP. В основном используется, когда не задан LDAPUserDN. Если параметр не задан в конфигурации клиента (по умолчанию), то при Legacy-аутентификации клиентская библиотека fbclient будет передавать пароль пользователя серверу в зашифрованном виде и для аутентификации будут использоваться пароли в атрибутах rdbPassword и rdbSecurePassword.

Если параметр задан, то пароль будет передаваться на сервер в открытом виде, чтобы он смог аутентифицировать пользователя в LDAP методом bind (работает только для Legacy-аутентификации).

```
LDAPUserPrefix = uid
LDAPUserBase = cn
```

LDAPUserFilter

Фильтр для поиска учетных записей пользователей. Здесь шаблон %u будет заменен на имя пользователя, указанное в процессе аутентификации.

```
LDAPUserFilter = &(objectClass=user)(cn=%u)
LDAPUserFilter = uid=%u
```

LDAPGroupBase

Ветка в LDAP, которая будет использована как стартовая для поиска групп пользователей. Поиск групп будет выполняться рекурсивно. Группы пользователя будут преобразованы в его роли на сервере.

```
LDAPGroupBase = ou=group,dc=example,dc=com
LDAPGroupBase = cn=users,dc=example,dc=com
```

LDAPMembershipFilter

Фильтр, который будет использован при поиске в LDAP групп, к которым принадлежит пользователь.

Существует три основные схемы, используемые в LDAP для указания членства в группах. В соответствии с используемой схемой нужно формировать фильтр. В нём в качестве имени предполагаемого пользователя указывается шаблон %u, а в качестве DN пользователя указывается %d. Если указано пустое значение, принадлежность пользователя к группам не определяется.

```
LDAPMembershipFilter = memberUid=%u
LDAPMembershipFilter = member=cn=%u,cn=users,dc=example,dc=com
LDAPMembershipFilter = member=%d
```

LDAPUserCertificate

Атрибут LDAP, в котором будет храниться сертификат пользователя. Если данный параметр задан, то при многофакторномподключении сертификат, предъявленный пользователем, должен соответствовать его сертификату в LDAP (если настроены параметры подключения к LDAP-серверу). Сертификат должен храниться в двоичном формате (DER).

```
LDAPUserCertificate = userCertificate;
```

LDAPPasswordSync

Данный параметр задаёт синхронизацию пароля пользователя в БД безопасности и в LDAP. Здесь перечисляются атрибуты, которые должны меняться в LDAP при смене пароля пользователя в БД безопасности. Допускается указание через «;» следующих атрибутов:

- rdbPassword — пароль пользователя для метода Legacy_Auth;
- rdbSrpVerifier — пароль пользователя для метода Srp;
- rdbSecurePassword — пароль пользователя для метода Multifactor;
- userPassword — пароль пользователя, используемый обычно в UNIX-системах;
- sambaLMPassword — пароль пользователя, используемый SAMBA-протоколом;
- sambaNTPassword — пароль пользователя, используемый SAMBA-протоколом.

По умолчанию выполняется синхронизация всех паролей.

```
LDAPPasswordSync = rdbPassword;userPassword
```

LDAPReadOnly

Заданный параметр может перевести LDAP-сервер в режим только для чтения (значение 1). Тогда параметр LDAPPasswordSync будет игнорироваться. По умолчанию используется значение 0.

```
LDAPReadOnly = 0
```

6.8 Настройки безопасности

LoginFailureDelay

Задаёт время в секундах, на которое задерживается подключение к БД безопасности. Задержка добавляется, когда количество неудачных попыток для конкретного пользователя превышает максимальное число неудачных попыток (на данный момент 4) или когда общее количество неудачных попыток входа для всех пользователей превышает максимальное число одновременных сбоев (на данный момент 16). Счетчик неудачных попыток сбрасывается, если в указанном интервале нет сбоев входа в систему.

```
LoginFailureDelay = 8
```

ServerCertificate

Задаёт алиас сертификата, которым сервер будет удостоверять свою подлинность клиенту. Этот сертификат должен быть связан с соответствующим закрытым ключем. Алиас — это строка, в которой через запятую перечислены имя владельца сертификата (`SubjectCN`), издатель сертификата (`IssuerCN`) и серийный номер сертификата в шестнадцатеричном виде. Сервер будет искать такой сертификат в хранилище пользователя, от имени которого он запущен и в хранилище компьютера.

```
ServerCertificate = test,Test Center CRYPTO-PRO,143dd54900020002b231
```

ServerPrivatePin

В этом параметре задаётся пароль закрытого ключа из ключевого контейнера, используемого сервером. Если закрытый ключ не защищён паролем, указывается пустая строка.

```
ServerPrivatePin = mypass
```

TrustedCertificate

Указывает алиас доверенного сертификата. Если пользователь предъявляет сертификат, совпадающий с доверенным, для этого сертификата не выполняется верификация, а пользователь может указывать своё имя без пароля. По умолчанию доверенный сертификат не указан.

```
TrustedCertificate = test,Test Center CRYPTO-PRO,143dd54900020002b231
```

CertUsernameDN

Задаёт название атрибута в секции `Subject` у сертификата, который содержит имя владельца. По умолчанию используется атрибут `CN`.

```
CertUsernameDN = CN
```

CertUsernamePattern

Здесь указывается регулярное выражение в синтаксисе SQL, которое используется для извлечения имени пользователя из атрибута владельца в сертификате. По умолчанию указывается пустая строка, что означает использование значения атрибута целиком.

```
CertUsernamePattern = [a-zA-Z0-9.]+
```

CertVerifyChain

Указывается нужно ли выполнять проверку цепочки сертификации предъявленного пользователем сертификата. Чтобы проверка могла быть выполнена, сервер должен иметь доступ к центру, выдавшему сертификат пользователя или доступ к локальному списку отзыва сертификатов. По умолчанию используется значение 1, т.е. проверка включена.

```
CertVerifyChain = 1
```

HashesFile

Для того, чтобы включить контроль целостности файлов сервера, необходимо указать имя файла с хэш-суммами подконтрольных файлов сервера «Ред База Данных». Относительный путь берется от корневой директории сервера.

Параметр имеет строковый тип. Значение по умолчанию равно `hashes`.

Алгоритмы хэширования зависят от используемого криптоплагина.

```
HashesFile = hashes
```

IntegrityCheckInterval

Задает интервал времени регулярной проверки целостности файлов. Нулевое значение параметра отключает регулярную проверку. Параметр имеет целочисленный тип и измеряется в секундах. Значение по умолчанию равно 0.

Этот параметр не будет работать без параметра, указанного в `"HashesFile"`.

```
IntegrityCheckInterval = 10
```

IntegrityShutdownAttempts

Задает количество попыток прекратить работу СУБД безопасными средствами в случае несовпадения хэшей. Параметр имеет целочисленный тип. Значение по умолчанию равно 5.

Этот параметр не будет работать без параметра, указанного в `"IntegrityCheckInterval"`.

```
IntegrityShutdownAttempts = 2
```

KrbServerKeyfile

Устанавливает путь к файлу ключа, которым сервис СУБД аутентифицируется в Kerberos. По умолчанию путь не задан.

```
KrbServerKeyfile = /etc/krb5.keytab
```

GssServiceName

Устанавливает название сервиса СУБД на сервере Kerberos. По умолчанию установлено следующее значение:

```
GssServiceName = rdb_server
```

GssHostName

Устанавливает имя хоста сервера СУБД для аутентификации через GSS. По умолчанию используется localhost.

```
GssHostName =
```

GSSLibrary

Динамическая библиотека GSSAPI (`libgssapi_krb5.so`).

Поддерживается также библиотека `libvas-gssapi.so` от One Identity Authentication Services. При её использовании СУБД после аутентификации определяет группы, назначенные пользователю в домене, и назначает ему одноимённые роли, существующие в базе данных.

```
GSSLibrary = libgssapi_krb5.so
```

6.9 Настройка с учетом оборудования и нагрузки на СУБД

На производительность СУБД могут повлиять следующие параметры:

`FileSystemCacheThreshold`

Для архитектуры классик рекомендуется установить это значение больше страничного кэша, чтобы включить кэширование на уровне файловой системы. Для архитектуры суперсервер при наличии достаточного объема оперативной памяти также рекомендуется установить это значение больше страничного кэша. Если памяти недостаточно, то лучше выключить кэширование на уровне файловой системы, установив значение `FileSystemCacheThreshold` в 0.

`FileSystemCacheSize`

В `Windows` при избыточном использовании памяти под файловый кэш рекомендуется ограничить это использование, установив значение данного параметра, например в 60%.

`LockHashSlots`

Большая длина хэш-таблицы блокировок, позволяющая за счет небольшого увеличения объема памяти под эту таблицу, ускорить работу с ней. Для архитектуры классик рекомендуется значение 65521. Для архитектуры суперсервер рекомендуется значение 30011.

`TempBlockSize`

Минимальный размер блока сортировки и шаг его расширения при необходимости. Позволяет несколько ускорить работу алгоритма сортировки за счёт выделения памяти бóльшими блоками. При наличии больших сортировок рекомендуется увеличить значение параметра до 2097152.

`TempCacheLimit = 4194304000`

Максимальный размер временного пространства, который может быть закэширован. При наличии достаточного объема памяти позволяет хранить в ней (вместо выгрузки на диск) результаты сортировок, выгрузок `blob`-ов и т.д. Т.к. у каждого процесса сервера собственный кэш, может потреблять большие объемы памяти на классической архитектуре.

Значение параметра `TempCacheLimit` рекомендуеся сделать больше суммарного размера временных файлов в каталогах `TempDirectories`. При наличии свободной памяти можно увеличить значение параметра и понаблюдать за эффектом с помощью аудита.

`DeadlockTimeout = 100`

Если известно, что архитектура приложения не допускает дедлоков, можно увеличить этот параметр чтобы сервер не сканировал таблицу блокировок каждые 10 секунд при подозрении на

потенциальный дедлок. Количество сканирований и найденных дедлоков отображается в заголовке `fb_lock_print`. Если первый параметр значительно больше нуля, а второй - очень близок к нему, значит характер работы прикладной части предполагает ситуации блокировок, которые дедлоками не являются, но вынуждают СУБД выполнять их поиск. В этом случае можно увеличить значение `DeadlockTimeout` чтобы сервер не выполнял лишнюю работу.

`DefaultDbCachePages = 2048`

Количество страниц, используемых процессом в качестве кэша. Для классика по умолчанию - 75. Увеличение позволяет до определенной степени сократить обмен данными с диском. Т.к. у каждого процесса классика будет собственный кэш, нужно учитывать общее потребление памяти. Значительное увеличение (>1000) обычно не даёт эффекта.

Для архитектуры суперсервер значение параметра рекомендуется рассчитать по формуле:

$$\texttt{DefaultDbCachePages} = \frac{\texttt{MemorySize}}{4 * \texttt{PageSize}},$$

где `MemorySize` - общий объём памяти;
`PageSize` - объём страницы.

Для архитектуры классик значение параметра рекомендуется рассчитать по формуле:

$$\texttt{DefaultDbCachePages} = \frac{\texttt{MemorySize}}{4 * \texttt{ConnNum} * \texttt{PageSize}},$$

где `MemorySize` - общий объём памяти;
`PageSize` - объём страницы;
`ConnNum` - предполагаемое максимальное количество соединений.

`LockAcquireSpins = 100`

Позволяет при недоступности мьютекса на таблицу блокировок не усыплять запрашивающий его процесс (дорогостоящая операция), а проверять этот мьютекс на доступность указанное количество раз перед усыплением. При большом количестве процессов с короткими блокировками (OLTP-нагрузка) позволяет несколько увеличить производительность сервера за счет более активного использования CPU.

Для архитектуры классик при большом значении `Mutex wait` в выводе `rdb_lock_print` (>20 %) рекомендуется включить эту настройку и после перезагрузки сервера посмотреть уменьшится ли значение `Mutex wait`.

`WireCrypt = Disabled`

Отключает шифрование сетевого трафика.

`LockMemSize`

Значение параметра определяет объем памяти, которая будет выделена менеджеру блокировок. Для архитектуры классик рекомендуется значение 20971520. Для суперсервера изменять не требуется.

6.10 Настройка Linux

Для увеличения допустимого числа процессов СУБД и количества открытых ими файлов:

- в `/lib/systemd/system/firebird.service` лимиты `LimitNPROC` и `LimitNOFILE` необходимо установить значение 10000;
- параметр ядра vm.max_map_count необходимо увеличить до 256000, написав в `/etc/sysctl.conf` строку:

```
vm.max_map_count=256000
```

- для `systemd`-систем в `/usr/lib/systemd/firebird.service`:

```
Environment = FIREBIRD_TMP=/path/to/tmp
```

При наличии достаточного объема RAM можно смонтировать в память временный каталог, куда сервер будет выгружать файлы:

- дописать в `/etc/fstab` строку

```
tmpfs /tmp tmpfs size=32G 0 0
```

где 32G - максимальный размер, до которого может расшириться временный каталог в памяти;

- отключить `Transparent Huge Pages`, например, записав в `/etc/rc.local`:

```
echo never > /sys/kernel/mm/redhat_transparent_hugepage/enabled
```

6.11 Настройка работы с Java методами

В «Ред База Данных» реализована возможность создания внешних процедур, функций и триггеров с использованием языка программирования Java. Они могут располагаться в jar-файлах. В «Ред База Данных» для работы с этими файлами используется движок `FBJAVA`, который позволяет запускать функции, процедуры и триггеры на платформе Java.

Для их использования необходимо установить JRE не ниже 8. А также настроить параметры взаимодействия сервера «Ред База Данных» с виртуальной машиной Java с помощью конфигурационного файла `plugins.conf`, который расположен в корневом каталоге установки сервера. В нем необходимо раскомментировать секции относящиеся к `fbjava`, указать путь к `JAVA_HOME` и путь до каталога с jar-файлами (переменная `JarDirs`):

```
Plugin = JAVA {
    Module = $(dir_plugins)/fbjava
    Config = JAVA_config
}
Config = JAVA_config {
    JavaHome = /usr/lib/jvm/java-openjdk
    SecurityDatabase = $(this)/java-security.fdb
    JvmArgsFile = $(this)/jvm.args
    JarDirs = $(this)/jar
}
```

Внутренние классы, необходимые для FBJava, находятся в папке `/jar` установки сервера.

Классы, содержащие методы, которые будут использоваться в качестве тела внешних процедур, функций и триггеров, необходимо скопировать в каталог, указанный в переменной `JarDirs` в виде jar-файла.

Классы в файловой системе доступны всем базам данных, обрабатываемыми процессом RDB. По аналогии с сервером приложений они являются системными классами.

Более подробно о взаимодействиях с Java методами из базы данных описано в Руководстве разработчика.

Глава 7

Рекомендации по безопасной настройке СУБД

7.1 Общие рекомендации

- **Проверяйте актуальность версии СУБД**

  Компания "РЕД СОФТ" постоянно работает над улучшением качества своих продуктов, поэтому регулярно выпускаются новые сборки, которые исправляют имеющиеся ошибки, улучшают прикладной функционал или добавляют новые возможности по использованию. Рекомендуемые версии официально публикуются на сайте СУБД: reddatabase.ru.

- **Выбирайте правильную архитектуру СУБД**

  Существует четыре различных взаимозаменяемых архитектуры сервера:
  - **ClassicServer** — один процесс на одно соединение; поддержка многопроцессорных машин. Подходит для мощных систем с несколькими ЦПУ и большим количеством ОЗУ. Данная архитектура не вызывает отказ в обслуживании всех клиентов при сбое одного серверного процесса, и как следствие более надежна.
  - **SuperServer** — все соединения используют один процесс, меньшие требования к памяти при большем быстродействии; для многопроцессорных машин (до 3.0 для однопроцессорных). Это компактная и высокопроизводительная версия для встраивания в распространяемое ПО.
  - **SuperClassic Server** — один процесс, но свой поток на каждое соединение. Комбинация лучшего от SuperServer и Classic. Идеально подходит для виртуализации.
  - **Embedded** (встраиваемая) версия — весь движок содержится в одной библиотеке с именем клиентской библиотеки сервера, идеально подходит для однопользовательских систем, не требует инсталляции в Windows.

  Если вы не знаете какая архитектура подходит именно вам, то используйте Super Server. Позднее вы сможете изменить архитектуру сервера.

- **Защитите каталоги**

  Правильно настройте права доступа к каталогам, где размещается сама СУБД, базы данных, конфигурационные файлы и журналы аудита.
  Любой, кто имеет доступ к файловой системе на уровне чтения может скопировать БД и извлечь все данные из неё. Любой, кто имеет права на запись может уничтожить информацию, подменить ее или сделать БД нечитаемой. Как правило, только процесс сервера должен иметь доступ к файлам и каталогам. Пользователи не должны иметь доступа даже на уровне только для чтения. Они будут выполнять запросы к БД через сервер, а сервер гарантирует, что пользователи получают только разрешенный тип доступа к каким-либо объектам в базе данных.
  Права на каталог с СУБД и конфигурационные файлы настраиваются автоматически инсталлятором. Их владельцем становится `root`. Но файлы `firebird.log`, `*.fdb`, `rdb_guard` доступны для записи пользователю `firebird:firebird`.
  Владельцем каталога с базой данных и с журналами аудита должен быть пользователь, от имени которого запускается сервер (в Linux по умолчанию это пользователь `firebird:firebird`, в Windows — `System`). Только владельцу каталога должны быть предоставлены права на запись и чтение.

- **Делайте регулярные копии**

  Резервное копирование необходимо для возможности быстрого и недорогого восстановления информации (пользовательских баз данных, базы данных безопасности СУБД, конфигурационных файлов настроек СУБД, журналов аудита СУБД) в случае утери рабочей копии информации по какой-либо причине.
  Периодичность резервного копирования определяется исходя из важности хранимых данных. Для информации имеющей особенную ценность периодичность должна быть не реже чем 1 раз в сутки.
  Утилита `Nbackup`, входящая в комплект поставки СУБД, позволяет создавать резервные копии, восстанавливать из резервных копий и дополнительно позволяет создавать инкрементные копии и восстанавливать из них БД. Инкрементная резервная копия содержит только изменения со времени создания определенной,ранее созданной резервной копии. `Nbackup` позволяет блокировать файл базы данных и может работать с активной базой данных, не мешая подключенным к ней пользователям. Созданная резервная копия базы данных всегда будет отображать состояние базы данных на момент начала создания резервной копии. Если вы используете инкрементное копирование не забывайте периодически делать полные копии БД.
  Типичный порядок создания резервной копии: Блокировать базу данных с помощью параметра `-L (Lock)`:

  ```
  nbackup [-U <пользователь> -P <пароль>] -L <база_данных>
  ```

  Создать резервную копию, сжать файл базы данных, используя любые другие программы. Простое копирование файла также допустимо.
  Разблокировать базу данных с помощью параметра `-UN (UNlock)`:

  ```
  nbackup [-U <пользователь> -P <пароль>] -UN <база_данных>
  ```

  Более подробно об использовании утилит резервного копирования можно прочитать в главе 8 «Утилиты командной строки».

- **Настройте репликацию**

  Репликация предназначена для обеспечения повышенной отказоустойчивости в случае повреждения физической структуры файла базы данных, вызванной техническим сбоем оборудования, программным сбоем операционной системы или самой СУБД. Репликация — одна из техник масштабирования баз данных. Она подразумевает перенос любых изменений данных в реальном времени с основного рабочего сервера на один или несколько резервных серверов, гарантируя, таким образом, их идентичность с точки зрения хранящихся на них данных. В процессе подключения к основному серверу проверяется наличие и доступность резервных серверов, после чего устанавливается с ними постоянное соединение. Любые сетевые проблемы с данным соединением, обнаруженные в процессе репликации, считаются сбоем резервного сервера. Настройка системы репликации приведена в главе 11 «Репликация».

- **Запускайте СУБД от имени несистемного пользователя**

  На Unix-подобных системах, СУБД обычно уже запущена от имени пользователя `firebird` по умолчанию, а не `root`.На Windows платформах также необходимо запустить сервис СУБД под заданной учетной записью (например, `firebird`).На практике запуск службы от имени пользователя `localsystem` представляет угрозу безопасности, тем более если ваша система подключена к сети Интернет.

- **Не создавайте пользовательские БД от пользователя sysdba**

  Пользователь `sysdba` - это аккаунт с полными правами доступа для всех ваших баз данных. Его пароль должен быть известен только нескольким доверенным администраторам. Не используйте этого супер-пользователя для регулярной работы с БД. Вместо этого, со-

здайте учетные записи пользователей и наделите их необходимыми правами.

7.2 Рекомендации по настройке СУБД

- **Смените пароль SYSDBA**

  Если вы еще не сделали этого на этапе установки, смените пароль администратора `sysdba`. Для этого используйте оператор SQL:

```
ALTER USER SYSDBA PASSWORD '<пароль>';
```

- **Используйте псевдонимы для БД**

  В конфигурационном файле `databases.conf` можно сопоставить реальный путь к БД и специальный псевдоним, чтобы затем использовать более короткий и удобный псевдоним для обращения к нужной базе данных.
  Использование псевдонимов БД позволяет скрывать физическое расположение баз данных от конечных пользователей. Например, добавив строку в `databases.conf`:

```
personal = C:\DB\WORK\personal.fdb
```

  пользователям для подключения к БД будет достаточно знать псевдоним БД - `personal`. Псевдонимы также позволят вам без проблем перемещать базы данных, ведь клиенты продолжат использовать свои существующие строки подключения.
  Псевдонимы начинают работать немедленно как только были добавлены и сохранены - нет необходимости в перезагрузке сервера.

- **Измените метод аутентификации**

  СУБД Ред База Данных поддерживает следующие режимы аутентификации:
  - Безопасная парольная аутентификация использующая алгоритм хэширования SHA для передачи данных: `Srp, Srp224, Srp256, Srp384, Srp512`. По умолчанию используется Srp256;
  - Традиционная (`Legacy_Auth`) аутентификация, унаследованная от предыдущих версий;
  - Доверительная (`Win_Sspi`) аутентификация для ОС Windows;
  - Многофакторнаяаутентификация (`Multifactor`) с применением политик безопасности;
  - Доверенная аутентификация через механизм GSSAPI (`Gss`);
  - Аутентификация по протоколу LDAP.

  Изменение режима аутентификации производится с помощью параметров `AuthServer, AuthClient` в файле конфигурации `firebird.conf`.

- **Измените стандартный порт подключения**

  Измените имя сервиса (`RemoteServiceName`) или стандартный порт подключения (`RemoteServicePort`), которые будут использоваться для клиентских баз данных. Изменять следует только один из этих параметров, не оба сразу.

```
RemoteServicePort = 49156
```

  Эти параметры позволяют усложнить идентификацию злоумышленником вашего сервера в локальной сети.

- **Защитите сервер от атаки перебором паролей**

  Пользователям можно назначить политику, которая будет отвечать за максимальное количество неудачных попыток входа, после которых учетная запись будет заблокирована.

```
CREATE POLICY BanPolicy AS MAX_FAILED_COUNT=5;
```

При заданной длине и наборе возможных символов труднее всего взломать пароли в виде строки случайных символов. Они достаточно долго выдерживают атаку полным перебором (из-за высокой энтропии), но их и труднее всего запомнить. Создайте политику безопасности, которая не позволит пользователям использовать простые пароли.

- `PSWD_NEED_CHAR` — минимальное количество букв в пароле;
- `PSWD_NEED_DIGIT` — минимальное количество цифр в пароле;
- `PSWD_NEED_DIFF_CASE` — требование использования различных регистров букв в пароле;
- `PSWD_MIN_LEN` — минимальная длина пароля;
- `PSWD_VALID_DAYS` — срок действия пароля;
- `PSWD_UNIQUE_COUNT` — количество последних не повторяющихся паролей;

```
CREATE POLICY PasswordPolicy AS PSWD_NEED_CHAR=7, PSWD_NEED_DIGIT=5,
PSWD_MIN_LEN=12, PSWD_NEED_DIFF_CASE=true, PSWD_VALID_DAYS=30,
PSWD_UNIQUE_COUNT=3;
```

Будет создана политика которая, запретит использование паролей длиною менее 12 символов, где должно быть не менее 7 букв и 5 цифр. Должны использоваться буквы в разном регистре. Срок действия пароля 30 дней. 3 последних пароля не должны повторяться. Назначьте политику пользователю и смените ему пароль. Политика должна примениться.

- **Контролируйте активность пользователей**

Для простых учетных записей пользователей вполне достаточно одной открытой сессии. Для учетный записи администратора `sysdba` это значение может быть увеличено до 2.
Компьютер, оставленный без присмотра пользователем, может быть использован злоумышленником. Задайте время бездействия пользователя, после которого сессия будет блокирована.
Администраторы могут владеть неактуальной информацией об используемых учетных записях. Полезно блокировать те учетные записи пользователей, которые не используются продолжительное время.

```
CREATE POLICY AccPolicy AS MAX_UNUSED_DAYS=45;
```

Будет создана политика которая, установит максимум одну сессию для пользователя, с длительностью бездействия в 10 минут, и 45 днями неиспользования учетной записью.

- **Контролируйте целостность ваших файлов**

Контроль за файлами сервера означает, что для всех критически важных файлов сервера (бинарные файлы, файлы конфигурации, база данных безопасности `security3.fdb` и т. д.) может быть вычислен и проверен хэш.
Для генерации хэшей используется утилита `hashgen`, входящая в комплект поставки СУБД. Пример использования утилиты `hashgen`:

```
hashgen generate -S 32798 ./security3.fdb > hash.sign
```

Этой командой сгенерирована и сохранена в файл `hash.sign` контрольная сумма для БД безопасности `security3.fdb`.
Для того, чтобы включить контроль целостности файлов сервера, необходимо указать имя файла, содержащего контрольные суммы (хэши) защищаемых файлов сервера в конфигурационном файле `firebird.conf`. Имя этого файла задается параметром `HashesFile`:

```
HashesFile = hash.sign
```

Если задано значение этого параметра, то каждый раз при запуске сервера и регулярно в процессе работы СУБД в соответствии со значением параметра `IntegrityCheckInterval` файла конфигурации `firebird.conf`, происходит проверка целостности файлов сервера.

```
IntegrityCheckInterval = 10
```

Раз в 10 секунд сервер будет проверять БД `security3.fdb`. Если хэши при проверке не совпадут, то СУБД попытается прекратить работу безопасными средствами. Количество попыток безопасного прекращения работы задается параметром `IntegrityShutdownAttempts`:

```
IntegrityShutdownAttempts = 2
```

- **Обезличивайте память**

  Параметр `MemoryWipePasses` файла конфигурации `firebird.conf` используется для настройки необходимости и метода обезличивания освобождаемой сервером оперативной памяти и дискового пространства. Значение по умолчанию равно 0, это означает, что обезличивание памяти отключено. Возможные значения:

  – 0 — обезличивание не происходит;
  – 1 — происходит обнуление памяти;
  – >1 — происходит чередование записи `0xFF` и `0x00` в освобождаемую память, последний проход при этом в любом случае заполняет блок нулями.

- **Включите аудит событий**

  Аудит событий реализован на основе утилиты `FBTrace`. Средства аудита позволяют серверу отслеживать и записывать в лог-файлы такие события: соединения и отсоединения от БД (создания и удаления БД), операции DML и DDL, выполнение хранимых процедур и т.д. Запись в лог для каждой конкретной БД начинает вестись с момента ее создания или присоединения к ней и до момента отсоединения от нее или ее удаления. Регистрируются события, завершившиеся как удачно, так и неудачно (с ошибкой).
  Сессию системного аудита запускает сам сервер. Это означает, что нет необходимости взаимодействия с пользователем. События, которые будут отслеживаться в этой сессии, задаются в конфигурационном файле и читаются при старте сессии.
  Файл с шаблоном настроек `fbtrace.conf` находится в корневом каталоге и содержит список отслеживаемых событий и указывает размещение логов трассировки для каждого события. Это позволяет достаточно гибко настроить параметры аудита различных событий для любой базы данных, при этом логирование будет осуществляться в отдельные файлы. По умолчанию аудит выключен. Для включения аудита необходимо изменить в конфигурационном файле `fbtrace.conf` параметр `enabled`.

- **Используйте шифрование**

  В СУБД Ред База Данных существует возможность зашифровать данные хранимые в базе данных. Не весь файл базы данных шифруется: только страницы данных, индексов и blob.
  Для того чтобы сделать шифрование базы данных возможным необходим плагин шифрования базы данных. Ред База Данных предоставляет плагины `Crypto_Api` и `RdbCrypt`. Также плагин можно написать самостоятельно.

  > Пример плагина шифрования в `examples/dbcrypt` не производит реального шифрования, а просто демонстрирует, как можно написать этот плагин.

  Затем Вы можете зашифровать свою базу данных с помощью следующей команды:

```
ALTER DATABASE ENCRYPT WITH <имя плагина> [KEY <имя ключа шифрования>]
```

Также в СУБД Ред База Данных есть защита канала передачи данных. Параметр `WireCrypt` устанавливает, следует ли шифровать сетевое соединение. Он может принимать три возможных значения: `Required, Enabled, Disabled`. По-умолчанию установлено, что шифрование является обязательным (`Required`) для подключений, поступающих на сервер и включенным (`Enabled`) для подключений, исходящих с сервера.

- **Изучите документацию**

  Мы создали подробную документацию (https://reddatabase.ru/documentation/). В ней описаны все необходимые моменты по правильному использованию СУБД Ред База Данных. Прочитайте её, задайте нам вопросы (rdb@red-soft.ru) и начните использовать СУБД прямо сейчас.

Глава 8

Утилиты командной строки

8.1 Утилита ISQL

ISQL - это утилита командной строки для работы с базами данных «Ред Базы Данных» при помощи языка структурированных запросов (Structured Query Language — SQL, подробнее о языке SQL см. «Руководство по SQL»). Утилита может быть использована для создания БД, создания и изменения метаданных и для выполнения различных запросов к БД. Утилита может работать в двух режимах: пакетном и интерактивном.

В пакетном режиме утилита получает на вход файл со скриптом SQL, который содержит одну или несколько команд. По завершении выполнения всех переданных на вход команд утилита завершает свою работу.

В интерактивном режиме пользователь последовательно вводит команды для работы с базами данных и тут же получает результат их выполнения. При этом одна команда может быть разбита на несколько строк. После завершения обработки каждой команды и вывода всех результатов ее работы пользователь получает приглашение ввести следующую команду до тех пор, пока не будет введена команда выхода из интерактивного режима (`exit` или `quit`).

В интерактивном режиме история команд пишется в файл. По умолчанию максимальный размер истории ограничен 1000 строк. Изменить это ограничение можно с помощью переменной среды `ISQL_HISTSIZE`. В пакетном режиме история не пишется.

Запуск ISQL

Запуск утилиты производится следующим образом:

```
isql [-u <пользователь>] [-p <пароль>] [-r <роль>] [[<спецификация сервера>] <БД>]

<спецификация сервера> ::= host[\port | service]:
                         | \\host[@port | service]\
                         | <protocol>://[ host[: port | service]/]

<protocol> = inet | wnet | xnet
```

Соединение с базой данных

После запуска утилиты необходимо либо присоединиться уже к существующей БД, либо создать новую. Для создания базы используется оператор `CREATE DATABASE`, для соединения с уже существующей базой данных — оператор `CONNECT` (подробнее см. «Руководство по SQL», глава 3). Присоединиться к уже существующей БД можно непосредственно при запуске утилиты, указав имя базы, и, если необходимо, имя пользователя и пароль[1]. Необязательный переключатель `-role` задаёт имя роли, права которой будут учитываться при работе с базой данных.

```
isql 127.0.0.1:c:\temp\base.fdb -user testuser -password pass -role rdb$admin
```

[1]Имя пользователя и пароль можно не указывать, если в системе установлены контекстные переменные `ISC_USER` и `ISC_PASSWORD`. А также если используется доверительная аутентификация Windows (`Win_Sspi`) илидоверенная через механизм GSSAPI (`gss`), и пользователь системы, от имени которого запускается утилита ISQL, является членом группы администраторов. Эти опции также действуют длявсех описываемых ниже утилит.

Символ терминатора

Символом конца строки (завершения команды) в ISQL по умолчанию является точка с запятой. Этот символ можно изменить командой:

```
SET TERM <строка>;
```

где `<строка>` может быть как одним символом, так и группой символов.

Транзакции в ISQL

ISQL может использоваться для выполнения операций трех типов:

- изменение структуры БД (DDL-операции);
- изменение и выборка данных из БД (DML-операции);
- просмотр структуры БД (извлечение метаданных).

В каждом из этих случаев, если не задано отдельно оператором `SET TRANSACTION`, ISQL автоматически запускает транзакцию по умолчанию.

При выполнении DDL-операции эта транзакция автоматически подтверждается после ввода каждого оператора DDL, и стартует новая транзакция. Отключить режим автоматического подтверждения DDL-операций можно командой `SET AUTO[DDL] {OFF|ON}`.

При выполнении запроса выборки/изменения данных стартует транзакция с уровнем изоляции `SNAPSHOT`. Такая транзакция будет активной до тех пор, пока не будет вручную подтверждена оператором `COMMIT` или отменена оператором `ROLLBACK`.

Извлечение метаданных производится с помощью команды `SHOW`. При этом ISQL запускает транзакцию с уровнем изоляции `READ COMMITED`, что дает возможность видеть все изменения метаданных, подтвержденные другими пользователями.

Переключатели командной строки

Переключатели командной строки начинаются в символа «-». Достаточно указать только начальные символы переключателей.

Таблица 8.1 — Опции ISQL

Опция	Описание
`-a(ll)`	Извлечение всех метаданных, включая не-SQL объекты (например внешние функции). Используется совместно с командой `extract`
`-b(ail)`	Этот переключатель указывает утилите поручить ошибку ОС, но только в пакетном режиме (`set bail on`). Переключатель возвращает код ошибки операционной системе. Он был добавлен для предотвращения выполнения скриптов после обнаружения ошибки.
`-c(ache) <число>`	Задать число страниц, которые будут кэшироваться при соединении с БД
`-ce(rtificate) <алиас>`	Использовать сертификат для проверки подлинности пользователя при многофакторнойаутентификации
`-ch(arset) <кодировка>`	Задать кодировку для текущего соединения (`set names`)
`-d(atabase) <база данных>`	Задать имя и путь к БД, которое будет записано в выходной поток

Опция	Описание
-e(cho)	Включает или подавляет дублирование команд на указанное устройство вывода (монитор, в файл, и т. д.) (set echo on)
-ex(tract)	Извлечь метаданные
-f(etch_password)	Извлечь пароль из файла
-i(nput) <имя файла>	Задать файл с SQL-запросами для выполнения (set input).
-l(ogin)	Эффективный логин доверенного пользователя. Для аутентификации по паролю с именем другого пользователя. См. параметр конфигурации TrustedUser.
-m(erge)	Перенаправление ошибок на поток стандартного вывода
-m2	Отправлять информацию о статистике и планах в выходной файл, который указан в переключателе -o(utput)
-n(oautocommit)	Отключить автоматическое подтверждение DDL-операций (set autoddl off)
-nod(btriggers)	Не запускать триггеры базы данных
-now(arnings)	Не показывать предупреждения
-o(utput) <имя файла>	Задать файл для вывода результата выполнения запросов. Без аргументов перенаправляет вывод на стандартное устройство вывода (монитор)(set output)
-pag(elength) <размер>	Размер страницы
-p(assword) <пароль>	Пароль пользователя
-pi(n) <PIN>	Задать PIN (пароль), если он необходим для получения закрытого ключа сертификата пользователя
-q(uiet)	Не показывать сообщение "Use CONNECT..."
-r(ole) <роль>	Имя роли
-r2 <роль>	Имя роли с учетом регистра
-s(qldialect) <диалект>	SQL диалект (set sql dialect)
-t(erminator) <строка>	Команда терминатора (set term)
-tr(usted)	Использовать доверительную аутентификацию
-u(ser) <пользователь>	Имя пользователя
-x	Извлечение метаданных
-z	Показать версии утилиты и сервера

Общие команды

Общие команды isql выполняют множество полезных задач, включая чтение, запись и выполнение скриптов схемы, а также выполнение команд командной строки.

Таблица 8.2 — Общие команды

Команда	Описание
`BLOBDUMP <ID blob> <имя файла>`	Сохраняет данные `BLOB` в указанном файле. `<ID blob>` – идентификатор, содержащий два шестнадцатеричных числа, разделенных двоеточием (:). Первое число является идентификатором таблицы, содержащей столбец `BLOB`. Второе – последовательный номер объекта. Для получения этого идентификатора выдайте любой оператор `SELECT`, который выбирает столбец `BLOB`. Выход покажет шестнадцатеричный идентификатор `BLOB` выше или на месте столбца `BLOB` в зависимости от того, установлен ли `SET [DISPLAY]` в `ON` или `OFF`.
`BLOBVIEW <ID blob>`	Отображает данные `BLOB` в текстовом редакторе по умолчанию. `<ID blob>` – идентификатор, содержащий два шестнадцатеричных числа, разделенных двоеточием (:). См. описание `BLOBDUMP` для определения идентификатора. *Замечание:* `BLOBVIEW` может вернуть ошибку "Invalid transaction handle" после закрытия редактора. Для исправления ситуации запустите транзакцию вручную с помощью команды `SET TRANSACTION`.
`EDIT [<имя файла>]`	Позволяет отредактировать и заново выполнить предыдущую команду isql или пакет команд в исходном файле. `<имя файла>` (необязательно) – полностью заданное имя файла для редактирования в файловой системе. Команда `EDIT` также может быть использована для открытия предыдущих операторов в редакторе и последующего их выполнения.
`HELP`	Отображает список команд isql с их описанием.
`INput <имя файла>`	Читает и выполняет блок команд из указанного текстового файла (скрипта SQL). Входные файлы могут содержать другие команды `INPUT`, предоставляя таким образом возможность проектирования цепочного или структурированного набора скриптов DDL.
`OUTput [<имя файла>]`	Перенаправляет выходные данные в файл на диске или на стандартное устройство вывода (монитор). Если имя файла не указано, результаты появятся на стандартном выводе, на мониторе (т. е. вывод в файл отключен).
`SHELL [<команда>]`	Предоставляет временный доступ к окну командной строки без подтверждения или отката любой транзакции. Аргумент `<команда>` является необязательным, это команда или вызов, допустимый в командной строке, из которой была запущена isql. Команда будет выполнена, а управление возвращено isql. Если команда не указана, isql открывает интерактивную сессию в командной строке, ввод `EXIT` возвращает управление isql.
`EXIT`	Подтверждает текущую транзакцию без подсказки, закрывает базу данных и завершает сессию isql.
`QUIT`	Отменяет текущую транзакцию и закрывает окно isql.

Команды SHOW

Команды `SHOW` используются для отображения метаданных, включая таблицы, индексы, процедуры, триггеры и привилегии. Они могут отображать список имен всех объектов указанного типа или предоставлять детальную информацию о конкретном объекте, заданном в команде.

```
SHOW <объект> [<имя объекта>]
```

Задание имени объекта по маске производится с помощью группового символа «%», который будет задавать маску имен объектов подобно LIKE в SQL-запросах, то есть обозначает любое количество любых символов в именах объектов.

Следующий пример покажет все таблицы, начинающиеся с tab.

```
show tables tab%
```

Таблица 8.3 — Команды SHOW

Команда	Описание
SHOW ALL	Покажет все метаданные.
SHOW CHECKs <имя таблицы>	Отображает имена и тексты всех определенных пользователем ограничений CHECK, заданных для указанной таблицы.
SHOW COMMENTs	Отображает все комментарии, которые были созданы для разных объектов текущей БД.
SHOW {COLLATIONs | COLLATION <имя>} SHOW {COLLATEs | COLLATE <имя>}	Выводит список всех определенных пользователем параметров сортировки текущей БД.
SHOW {DOMAINs|DOMAIN <имя>}	Отображает определение одного указанного домена <имя> или отображает список имен всех доменов, объявленных в базе данных (DOMAINs) .
SHOW {DATABASE|DB}	Отображает информацию о подключенной базе данных (имя файла, размер и количество выделенных страниц, интервал очистки, номера транзакций, статус Forced Writes, ODS, набор символов по умолчанию).
SHOW DEPENdencies <имя объекта> SHOW DEPENdency <имя объекта>	Отображает все зависимости для указанного имени объекта. На выходе – разделенный запятыми список других объектов БД, с которыми указанный объект зависим.
SHOW {EXCEPtions|EXCEPtion <имя>}	Отображает текст одного указанного исключения <имя> или отображает список имен и текстов всех исключений, объявленных в базе данных (EXCEPtions).
SHOW {FILTERs | FILTER <имя>}	Выдает список всех BLOB–фильтров, объявленных оператором declare filter или показывает подробную информацию о конкретном фильтре.
SHOW {FUNCtions|FUNCtion <имя>}	Отображает объявление указанной внешней функции <имя> или отображает список имен всех внешних функций, объявленных в базе данных (FUNCtions).
SHOW SYStem {GENERATORs|GENERATOR <имя>} SHOW SYStem {SEQuences|SEQuence <имя>}	Эти две команды идентичны. Отображают объявление указанного системного генератора <имя> вместе с его текущим значением или отображает список имен всех системных генераторов, объявленных в базе данных вместе с их текущими значениями (GENERATORs|SEQuences).

Команда	Описание
SHOW {GENERATORs|GENERATOR <имя>} SHOW {SEQuences|SEQuence <имя>}	Эти две команды идентичны. Отображают объявление указанного генератора <имя> вместе с его текущим значением или отображает список имен всех несистемных генераторов, объявленных в базе данных вместе с их текущими значениями (GENERATORs|SEQuences).
SHOW {GRANTs| GRANT {<имя объекта>|<имя роли>}}	Если команда содержит GRANTs, то она отображает список всех привилегий в текущей БД. Иначе отображает информацию привилегий и ролей по отношению к указанному объекту в подключенной базе данных или отображает членство пользователей в роли.
SHOW {INDexes|INDICES} SHOW {INDICES|INDexes} <имя таблицы> SHOW INDex <имя индекса>	Первая форма отображает информацию обо всех индексах для всех таблиц в подключенной базе данных. Вторая форма отображает информацию об индексах для указанной таблицы <имя таблицы>. Наконец третья форма отображает информацию об указанном индексе.
SHOW {PROCedures | PROCedure <имя>}	Отображает все процедуры в подключенной базе данных с их зависимостями или отображает текст указанной процедуры с объявлениями и типами (входной/выходной) каждого аргумента.
SHOW {ROLEs | ROLE <имя>}	Отображает имена ролей SQL в подключенной базе данных или отображает всех пользователь, получивших данную роль.
SHOW SECCLAsses <имя объекта>	Отображает информацию о классах безопасности для данного объекта.
SHOW SQL DIALECT	Отображает диалекты SQL клиента и подключенной базы данных.
SHOW SYStem [<имя таблицы>]	Команда без параметров отображает имена системных таблиц, функций и сортировок для конкретных наборов данных. Для уточнения деталей о конкретной таблице укажите имя таблицы.
SHOW {TABLEs | TABLE <имя>}	Отображает список имен всех таблиц в алфавитном порядке или показывает подробности указанной таблицы.
SHOW {TRIGgers | TRIGger <имя>}	Отображает список имен всех таблиц вместе с именами их триггеров в алфавитном порядке или для заданного триггера указывает таблицу, к которой он принадлежит, отображает параметры заголовка, статус активности и исходный код PSQL тела триггера.
SHOW USERS	Выводит список пользователей, соединенных с БД в настоящее время.
SHOW VERsion	Отображает информацию о программной версии isql и серверной программы Ред База Данных, а также номер структуры на диске (ODS) подключенной базы данных.
SHOW {VIEWs | VIEW <имя>}	Отображает все представления или информацию об указанном представлении.

Команды SET

Команды SET позволяют просматривать и изменять некоторые вещи, связанные со средой ISQL. Отдельные из них доступны в скриптах.

Таблица 8.4 — Команды SET

Команда	Описание
SET	Выводит на экран текущие установленные опции SET
SET AUTO[DDL] [on|off]	Задает, будут ли операторы DLL подтверждаться автоматически после их выполнения или будут подтверждаться после явного выполнения COMMIT. Оператор доступен в скриптах. Без аргументов – просто переключает AUTODDL между включено и выключено.
SET BAIL [on|off]	Задание данной опции определяет поручить или нет ошибку ОС, но только в пакетном режиме. Переключатель возвращает код ошибки операционной системе. Команда была добавлена для предотвращения выполнения скриптов после обнаружения ошибки.
SET BLOB [n|all|off]	Задает необходимость отображения подтипа BLOB и отображения данных BLOB. n – отображать BLOB заданного подтипа. Значение по умолчанию n=1 (текст). Положительные числа определены в системе; отрицательные числа определяются пользователем. all – отображать данные BLOB любого подтипа. off – отключает отображение данных BLOB. Вывод показывает только идентификатор BLOB (два шестнадцатеричных числа, разделенных двоеточием).
SET BULK_INSERT <подготовленный insert запрос>	Примитивная обработка подготовленных запросов на вставку. После ввода этой команды пользователь входит в интерактивный режим (BULK>), где требуется ввести параметры в виде кортежа (val1, ..., valN). Для выхода из интерактивного режима введите пустую строку или любой символ, отличный от '(' и '--' (можно явно выйти, введя слово stop). Кортежи должны располагаться на одной строке, за исключением параметров в кавычках. Однако, если добавить '+++' после запятой, параметры кортежа можно переносить на следующую строку. Одиночные комментарии ('--') можно добавлять только между кортежами. Команды commit или commit work вводятся с первого символа строки и располагаются на одной строке. Любые символы, идущие за кортежем, кроме ')' и '+++', будут игнорироваться.
SET COUNT [on|off]	Включает/выключает отображение количества строк, найденных по запросам.
SET ECHO [on|off]	Включает/выключает отображение команд до их выполнения.
SET EXPLAIN [on|off]	Включает/выключает расширенного подробного плана запроса. Этот план выводит более подробную информацию о методах доступа используемых оптимизатором, однако его нельзя включить в запрос.
SET HEADING [on|off]	Включает/выключает отображение заголовков столбцов.

Команда	Описание
SET LIST [on|off]	Задает формат отображения результатов запросов SELECT. По умолчанию данные на экране выводятся в табличном варианте, где сверху названия столбцов, а под ними все строки, удовлетворяющие запросу SELECT. Значение ON позволяет выводить результат в другом виде, где для каждой строки (результата запроса) выводится своя отдельная таблица (с заголовками столбцов слева).
SET NAMES <csname>	Задает набор символов, который будет активным в транзакциях базы данных.
SET PLAN [on|off]	Задает, нужно ли отображать план запроса оптимизатора.
SET PLANONLY	Задает только подготовку запросов SELECT и отображение плана без выполнения самого запроса.
SET [TRUSTED] ROLE	Изменяет текущую роль. Позволяет установить контекстной переменной CURRENT_ROLE одну из назначенных ролей для пользователя CURRENT_USER или роль, полученную в результате доверительной аутентификации (SET TRUSTED ROLE).
SET {ROWCOUNT|MAXROWS} [<n>]	Указывает какое максимальное количество строк будет выводится при выполнении оператора SELECT. Значение 0 (стоит по умолчанию) говорит о том, что ограничений нет.
SET SQLDA_Display [on|off]	Отображает или скрывает внутренние подробности об SQL-операторах, которые были выполнены isql.
SET SQL DIALECT <n>	Устанавливает SQL-диалект в то значение, которое было задано для сессии клиента. $n \in \{1, 2, 3\}$.
SET STATs [on|off]	Определяет, отображать ли статистику выполнения, которая будет следовать за выходными данными запроса.
SET TIME [on|off]	Задает, отображать ли время в значении DATE (только диалект 1).
SET TERM <строка>	Задает символ, который будет использоваться в качестве терминатора команды или оператора. Он доступен в скриптах.
SET TRANSACTION	Запускает на выполнение транзакцию с заданными характеристиками.
SET {WARNINGS|WNG} [on|off]	Задает, выводить ли предупреждающие сообщения.
SET WIDTH <столбец> [<n>]	Используя команду set width, пользователь может регулировать ширину столбца на экране. Но только для character – столбцов. <n> — количество символов.

8.2 Утилита GBAK

Утилита gbak, входящая в поставку «Ред Базы Данных», является средством создания резервных копий баз данных и восстановления баз данных из резервных копий. Она есть во всех дистрибутивах для всех платформ. И в отличие от интерактивных средств, использующих Services API (например, RDB Expert), позволяет автоматизировать процесс резервного копирования.

Термин "резервное копирование"имеет достаточно общий смысл, целью которого является получение копии базы данных, пригодной для архивирования. Например, "резервную копию"бызы данных можно изготовить простым способом – скопировать базу данных, предварительно остановив сервер Ред Базы Данных (иначе, при работающем сервере, копия будет являться "поврежденным файлом т. к. копирование производится последовательно, а база данных – файл произвольного доступа). Если же нужно сделать резервную копию базы данных "на ходу во время работы СУБД, то

для этого и предназначен gbak.

> В gbak действует принцип «обратной совместимости». Это значит, что созданные резервные копии в более ранних версиях «Ред База Данных» могут быть восстановлены в более поздних, но не наоборот.

8.2.1 Права на запуск gbak

Для того, чтобы выполнить любую операцию над базой данных с помощью gbak, необходимо пройти процедуру идентификации и аутентификации — указать имя и пароль пользователя, который имеет право на запуск сервиса gbak. Это могут сделать SYSDBA, владелец базы или администратор с ролью rdb$admin. Для указания имени и пароля пользователя используются переключатели -user и -pa[ssword], соответственно, например:

```
gbak -b employee.fdb emp.fbk -user testuser -pas pass -role rdb$admin ...
```

Необязательный переключатель -role задаёт имя роли, права которой будут учитываться при выполнении каких-либо действий утилиты. Поэтому если пользователь не указывает роль, то он получает права только тех ролей, которые ему назначены с DEFAULT.

Имя пользователя и пароль можно не указывать, если в системе установлены контекстные переменные ISC_USER и ISC_PASSWORD. А также если используется доверительная аутентификация Windows (Win_Sspi) или доверенная через механизм GSSAPI (gss), и пользователь системы, от имени которого запускается утилита gbak, является членом группы администраторов.

Далее в примерах опции -user, -pas и -role будем опускать для наглядности команд.

8.2.2 Имена файлов

Имя базы

Поскольку gbak - это программа, которая читает данные из базы данных, то сама база данных и сервер могут находиться где угодно.

В следующем примере использован локальный коннект:

```
gbak -b employee.fdb employee.fbk
```

подразумевающий, что gbak выполняется на сервере.

Если запускать gbak с клиентской машины, и обращаться к серверу, то тогда вместо employee.fdb надо было бы писать

```
server:c:\RedDatabase\bin\employee.fdb
```

то есть штатную строку коннекта с указанием сервера и пути к БД (или алиаса).

Разумеется, gbak создавая резервную копию, будет создавать ее локально, т. е. на компьютере где находится gbak. И если сервер – это другой компьютер, то фактически вся база данных будет считана по сети.

> Если требуется осуществить резервное копирование в файл на стороне сервера, не стоит делать резервные копии баз размером гигабайт и выше например через 100Мбит сеть. Быстрее будет сделать резервную копию на сервере, а затем просто скопировать ее файл на другой компьютер (если это нужно). Или вместо бэкапа по сети лучше использовать сервисы.

Имя резервной копии

Имя резервной копии может быть абсолютно любое, включая любое расширение. Например, bak, gbk, fbk, и так далее. Но лучше имя резервной копии формировать как: имя базы и дата, причем дату лучше всего указывать в формате YYYYMMDD – так имена файлов будут корректно сортироваться при просмотре папки.

8.2.3 Опции утилиты

В таблицах приведены все доступные опции утилиты gbak и краткое описание к ним. Они сгруппированны по назначению: для резервного копирования, для восстановления и опции, доступные в обеих операциях.

Более подробно о большинстве опций будет рассказано в подразделах, посвященных созданию резервных копий и восстановлению из них.

Таблица 8.5 — Общие опции GBAK (для backup и restore)

Опция	Описание
-fe[tch_password]	Получить пароль из файла.
-m[eta_data]	Производит резервное копирование или восстановление только метаданных.
-pas[sword]	Пароль пользователя, выполняющего операцию gbak (см.).
-ro[le]	Подсоединиться с использованием роли (см.).
-se[rvice]	Создаёт резервную копию на том же компьютере, где находится база данных-источник. Для этого вызывается Service Manager на компьютере-сервере.
-skip_d[ata] <шаблон>	Не сохранять в бэкап (или рестор) данные таблиц, имена которых соответствуют указанному здесь регулярному выражению в SQL-синтаксисе (см. оператор SIMILAR TO) [2].
-keep_d[ata] <шаблон>	Включать только данные таблиц, имена которых соответствуют указанному здесь регулярному выражению в SQL-синтаксисе (см. оператор SIMILAR TO). Причем, если <шаблон> для -KEEP_DATA и -SKIP_DATA соответствует одной и той же таблице, то таблица пропускается.
-st[atistics] TDRW	Вывести статистику в процессе работы (работает вместе с -v(erbose) и -verbi(nt)). T—время от начала работы gbak; D—время прошедшее с последнего вывода на экран; R—число страниц прочитанных с последнего вывода на экран; W—число страниц записанных с последнего вывода на экран.
-user	Имя пользователя, выполняющего операцию gbak (см.).
-v[erify]	Отчет о каждом выполненом действии
-verbi[nt] <n>	Подробный вывод лога действии с заданным интервалом обработанных записей. Минимальное значение - 100.
-y <путь к файлу> | SUPPRESS	Вывод лога в файл. Указывается совместно с -v(erify) или -verbi(nt). Если указан параметр suppress, не будет выведено никакой информации даже с опцией -v(erify).
-z	Показать версию gbak и версию сервера «Ред База Данных».

[2] Внешние ключи, ссылающиеся на эти таблицы, будут деактивированы

Таблица 8.6 — Backup опции GBAK

Опция	Описание
-B[ACKUP_DATABASE] [O[VERWRITE]]	Основная опция при создании резервной копии. Если файл резервной копии уже существует, то резервное копирование не будет выполнено. Но если указать опцию O[VERWRITE], файл бэкапа будет заменен.
-co[nvert]	Преобразование внешних таблиц (external tables) во внутренние таблицы.
-e[xpand]	Не производить сжатие резервной копии. По умолчанию резервная копия записывается в "сжатом"виде.
-fa[ctor] n	Использовать блокирующий фактор n для ленточного накопителя. Устарел.
-g[arbage_collect]	Не собирать мусор во время резервного копирования [3].
-ig[nore]	Игнорировать контрольные суммы, а также поврежденные BLOB при бэкапе и отключать поврежденные BLR при ресторе[4] Если указана эта опция, то в суперсервере другие подключения к базе данных будут запрещены на время выполнения бэкапа.
-l[imbo]	Игнорировать зависшие двухфазные транзакции(limbo).
-nod[btriggers]	Предотвращает срабатывание триггеров базы данных во время резервного копирования.
-nt	Создаёт резервную копию в непереносимой формате между аппаратными платформами.
-ol[d_descriptions]	Производит резервное копирование метаданных в формате старого стиля, т.е. в режиме совместимости со старыми базами данных. Устарел.
-par[allel] <n>	Количество параллельных рабочих потоков для бэкапа. Включает многопоточное выполнение резервного копирования.
-t[ransportable]	Создаёт транспортабельную (переносимую) резервную копию. Этот параметр включен по умолчанию, поэтому указывать его нет никакой необходимости.

Таблица 8.7 — Restore опции GBAK

Опция	Описание
-C[REATE_DATABASE]	Восстанавливает (создаёт) базу данных из бэкапа в новый файл.
-R[ECREATE_DATABASE] [O[VERWRITE]]	Создаёт новую базу данных или заменяет (если указана опция O) существующую базу данных из бэкапа.
-REP[LACE_DATABASE]	Восстанавливает файл базы данных, заменяя имеющуюся базу данных.
-bu[ffers]	Задать размер страничного кэша для БД.
-fix_fss_d[ata] <кодировка>	Исправить кодировку данных

[3] «Ред База Данных» - это версионная СУБД. Версии записей создаются при update и delete живут определенное время (пока они нужны активным транзакциям), и убираются как мусор, в определенные моменты. **Мусорные версии записей - это те, которые уже не нужны ни одной активной транзакции.**

[4] При чтении записи сервер перебирает не более 10 млн. версий записи, чтобы исключить возможность бесконечного цикла.

Опция	Описание
-fix_fss_m[etadata] <кодировка>	Исправить кодировку метаданных
-i[nactive]	Деактивировать индексы во время восстановления. При использовании опции в логе восстановления будет сообщение "gbak: WARNING:Database is not online due to inactive indexes (option -I)". Перед возвращением базы данных в режим "online" не забудьте активировать индексы.
-ig	Игнорировать ошибки в BLR процедур, функций и триггеров.
-k[ill]	Восстановить без создания теневых копий
-mo[de] <режим доступа>	Режим доступа "read_only"или "read_write"
-n[o_validity]	Отключение всех ограничений проверки данных (check) и ограничений NOT NULL. При использовании опции в логе восстановления будет сообщение "gbak: WARNING:Database is not online due to skipped restoring of validity conditions (option -N)". Перед возвращением базы данных в режим "online" не забудьте пересоздать отключенные ограничения.
-o[ne_at_a_time]	Подтверждать транзакцию после восстановления каждой таблицы.
-p[age_size]	Установить размер страницы
-par[allel] <n>	Количество параллельных рабочих потоков для рестора. Включает многопоточное выполнение восстановления БД, если MaxParallelWorkers > 1 (см. firebird.conf). Вместо этого параметра утилиты можно указать параметр конфигурации ParallelWorkers.
-use_[all_space]	Не резервировать место под версии записей. Максимально заполнять страницы данных (для read-only баз).
-rdb_map[ping_file] <файл с отображением>	Для корректного восстановления БД с версии 2.X, если ее бэкап содержит внешние Java функции или процедуры. Необходимость в этой опции возникла по причине изменения синтаксиса: в версии 2.X при объявлении внешней функции(процедуры) не указывались ее аргументы, в отличии от версии 3.0 и выше.Поэтому, если функции(процедуры) имеют аргументы, следует создать текстовый файл <файл с отображением>,где первый столбец - имя функции(процедуры) в java, а во втором - типы аргументов: package.class.Function1 arg1,arg2,arg3 package.class.Function2 arg1,arg2,arg3 В этом случае нужная java функция (процедура) будет подбираться на лету просто по имени.

8.2.4 Создание резервной копии

Утилита gbak является программой, которая подсоединяется к базе данных, стартует транзакцию snapshot, и затем сохраняет в специальный файл метаданные (описания таблиц, процедур, триггеров и т. д.) и данные (запросами select * from tablename).

Благодаря тому, что считывание данных происходит в транзакции snapshot, gbak на протяжении процесса чтения данных "видит"неизменные данные благодаря версионности. То есть, во время работы gbak другие приложения могут работать с базой данных. Однако, те изменения, которые были произведены приложениями во время работы gbak, разумеется не будут сохранены в резервную копию БД.

Формат командной строки для создания резервной копии:

```
gbak -B[ACKUP_DATABASE] [O[VERWRITE]] [backup опции] [общие опции] <база данных>
<файл резервной копии>
```

Основная опция при создании резервной копии — `-b` (другие опции подробно описаны дальше). Причем опции могут быть указаны как в начале, так и в конце.

Ключ `-b` означает, что необходимо выполнить резервное копирование базы данных, путь к которой указан как `<база данных>`, а результаты резервного копирования сохранить в файл, указанный как `<файл резервной копии>`. При этом, если путь последнего содержит несуществующие каталоги, они будут созданы автоматически.

Если `<файл резервной копии>` уже существует, то резервное копирование не будет выполнено. Но если указать опцию `O[VERWRITE]`, файл бэкапа будет заменен [5].

```
gbak -b o employee.fdb employee.fbk
```

Теперь поподробнее остановимся на других опциях бэкапа.

-v[erify], -verbi[nt] <n>

Полный вывод лога действий.

> Эти опции является взаимоисключающими. Использование `-verify` аналогично указанию `-verbint 10000`.

Без указаний этих опций `gbak` не выводит никаких сообщений при выполнении резервного копирования (или восстановления), кроме сообщений об ошибках, если такие возникли.

С указанием опций выводится большой объем информации о выполненных действиях. По умолчанию выходные данные отображаются на экране, но вы можете перенаправить лог в файл с помощью ключа `-y`.

Впрочем, для систем с регулярным резервным копированием, а также когда backup выполняется долго, лучше сразу указать опцию полного вывода лога действий. Даже в случае появления ошибки контекст этой ошибки будет виден четче, и также будет видно что именно сервер успел поместить в резервную копию до ошибки.

Так выглядит команда с ключом `-v`:

```
gbak -b employee.fdb employee.fbk -v
```

> Параметр `-v` в справке, выдаваемой `gbak`, описан как `-v[erify]`. Вопреки своему названию, эта опция ничего не проверяет. Скорее, этот параметр должен расшифровываться как "verbose".
>
> Единственный способ проверить резервную копию — восстановить ее, проверить, не содержит ли она ошибок, и, возможно, выполнить несколько запросов для проверки работоспособности.

Если `-v[erify]` выводит информацию о каждом обработанном действии, то `-verbi[nt] <n>` - с заданным интервалом обработанных записей. Интервал управляет тем, сколько записей будет выведено `gbak`; другими словами, он контролирует частоту вывода сообщений `"...records written"`. Минимальное значение - 100.

Так выглядит команда с ключом `-verbi`:

[5] Для ознакомления с утилитой и самостоятельного выполнения операций из примеров, рекомендуем взять тестовую базу данных `employee.fdb` из папки установки RedDatabase и скопировать эту базу в каталог `bin`, рядом с `gbak`.
В примерах опции `-user`, `-pas` и `-role` опускаем намеренно для наглядности команд. См. :ref:‘subsec:gbakuser‘.

```
gbak -b employee.fdb employee.fbk -verbi 200
```

-y <имя файла> | SUPPRESS

Перенаправление лога в файл или полное подавление записи лога.

Указывается совместно с опцией `-v(erify)` или `-verbi(nt)`.

Если задан параметр `-y <имя файла>`, то вывод лога будет записан в файл `<имя файла>`, без какого-либо вывода на экран.

```
gbak -b employee.fdb employee.fbk -v -y e_bak.txt
```

Если указан параметр `-y suppress`, то независимо от указания опций `-v` или `-verbi`, не будет выведено никаких сообщений ни в файл, ни на экран:

```
gbak -b employee.fdb employee.fbk -v -y suppress
```

Лог-файл не должен существовать. Если он есть, операция резервного копирования или восстановления завершится неудачей:

```
gbak -backup employee.fdb employee.fbk -y e_bak.txt -v
gbak:cannot open status and error output file e_bak.txt
gbak:Exiting before completion due to errors
```

Имя файла лога может быть любое, включая расширение. Расширение лога имеет смысл выбрать таким, чтобы по умолчанию файл открывался Блокнотом или программой, которой вы привыкли просматривать текстовые файлы.

-g[arbage_collect]

Отключает сборку мусора

Если подробнее, то этот параметр запрещает серверу проверять читаемые записи на наличие мусорных, что ускоряет процесс бэкапа. Фактически `gbak -b` без ключа `-g` будет вызывать срабатывание кооперативной сборки мусора, причем для всех данных, т. к. будут прочитаны все данные всех таблиц. Создание резервной копии должно выполняться максимально быстро, а соответственно лучше не загружать сервер в это время сборкой мусора. Поэтому рекомендуется всегда использовать такое начало командной строки бэкапа:

```
gbak -backup -garbage_collect employee.fdb employee employee.fbk
```

При резервном копировании никакой "мусор" никогда не попадает в файл backup, ни при каких условиях.

-t[ransportable]

Создание транспортабельной (переносимой) резервной копии.

Этот параметр действует по умолчанию, поэтому указывать его нет никакой необходимости.

Он означает, что полученный файл резервной копии можно восстановить на альтернативной аппаратной платформе, где порядок байт в целых числах отличается. То есть, например, между

Intel и Sparc, или HP-UX и Intel, и так далее. Но между Windows и Linux (или другой ОС) на Intel файл резервной копии будет и так переносимым, даже при указании ключа `-nt` (non-transportable). Так что, про `-t`, как и про `-nt` можно забыть, и никогда не указывать их в командной строке `gbak`.

-e[xpand]

Сжатие (низкой степени) файла резервной копии

Впервые `gbak` был создан примерно в 1985-86 годах, а тогда жесткие диски были очень малого объема. Поэтому по умолчанию `gbak -b` производит некую легкую компрессию данных. Отключить ее можно параметром `-expand`.

Были проведы тесты резервного копирования с параметром `-expand`, которые показали небольшое ускорение процесса бэкап (на 5-7.5%), но сильное увеличение размера резервной копии (до 30%). Так что практическую полезность параметра `-expand`, учитывая сильное увеличение размера резервной копии, можно считать равной нулю.

-co[nvert]

Преобразование внешних таблиц в обычные таблицы.

Если в базе данных созданы внешние таблицы (external tables), то при создании резервной копии они будут помещены внутрь бэкапа как обычные таблицы. Без параметра `-co` внешние таблицы в резервную копию не попадают.

Можно сказать, что `-co` нужен тогда, когда вам требуется "взять с собой"не только базу, но и внешние файлы. Правда, в зависимости от назначения эти файлы могут иметь разный размер, и может оказаться, что сами они будут больше, чем база данных. Для обычного, регулярного backup, параметр `-co` не требуется.

-m[eta_data]

Выполняется резервное копирование или восстановление только метаданных.

Сохраняет только метаданные (описания таблиц, процедуры, триггеры и т. д.). Данные в резервную копию не попадают. При ресторе восстанавливаются только метаданные базы данных, а любые данные из файла резервной копии не восстанавливаются. Используется когда вам нужно сделать копию пустой БД.

```
gbak -backup -meta_data employee employee.meta.fbk
```

```
gbak -create employee.fbk mytest.fdb -meta_data
```

-l[imbo]

Игнорирование limbo-транзакций

Не сохраняет в резервной копии БД версии записей, которые созданы транзакциями, находящимися в состоянии in limbo. Такое состояние может быть только у не завершившихся транзакций двухфазного коммита (2PC).

Например: если двухфазная транзакция (например, между двумя разными базами данных), завершилась неудачно из-за того, что сервер упал до commit или rollback, но после того, как изменения были подготовлены, то это limbo-транзакция. Этот переключатель заставляет процесс бэкапа игнорировать данные таких транзакций.

Её не следует использовать для обычного резервного копирования, а использовать, как ключ `-IG[NORE]`, только для попытки восстановления после сбоя.

-fe[tch_password]

Считать пароль из файла

```
-FE[TCH_PASSWORD]  <имя файла> | stdin | /dev/tty
```

С этой опцией пароль соответствующего пользователя (указанного в `-user`) считываться из файла, а не указываться в командной строке. Указанное имя файла не заключено в кавычки и должно быть доступно для чтения пользователю, запустившему `gbak`.

Если имя файла указано как `stdin`, пользователю будет предложено ввести пароль. В системах POSIX имя файла `/dev/tty` также приведет к запросу пароля.

-par[allel] <n>

Количество потоков, используемых для создания резервной копии.

Ред База Данных (начиная с версии 3.0) поддерживает многопоточный бэкап. Он показал себя до 5 раз быстрее обычного бэкапа в один поток. Реальное ускорение зависит от процессора, дисковой подсистемы и структуры базы данных.

По умолчанию используется один поток.

При создании резервной копии эта опция контролирует количество соединений, используемых для чтения пользовательских данных. Каждый дополнительный рабочий поток создает отдельное соединение для чтения данных одновременно с другими потоками. Все соединения используют один и тот же снимок базы данных для обеспечения консистентности резервной копии. Потоки создаются и управляются самим `gbak`. Метаданные базы данных читаются одним потоком.

Для включения распараллеливания в `gbak` выполните команду:

```
gbak -b employee.fdb employee.fbk -par 8 ...
```

Параметры конфигурации `MaxParallelWorkers` и `ParallelWorkers` не оказывают влияния на включение или отключение многопоточного бэкапа в `gbak` за исключением операций создания индекса. `MaxParallelWorkers` может ограничивать количество параллельных рабочих процессов во время создания индекса, а `ParallelWorkers` используется при создания индекса, если `-par[allel]` не указан.

Так как каждый рабочий поток создаёт собственное подключение к БД, невозможен параллельный бэкап базы данных в режиме `single shutdown`. В случае задания параметра `-par 2` или более, возникнет ошибка `"ERROR: database ... shutdown"`.

-skip_d[ata] <шаблон>

Исключает данные указанных таблиц из резервной копии или восстановленой базы данных.

В бэкап (или рестор) не попадают данные таблиц, имена которых соответствуют указанному здесь регулярному выражению в SQL-синтаксисе. Метаданные таблицы включаются.

Противоположная опция `-KEEP_DATA`. Чтобы пропустить все данные, используйте `-META_DATA`.

```
gbak -b employee.fdb employee.fbk -skip_d table1|table2| ...
```

Исключение данных таблиц из резервной копии или восстановленной базы данных может вызвать ошибки во время восстановления, если существует внешний ключ в таблице, которая не была исключена, связанный с таблицей, которая была исключена.

Можно использовать опцию восстановления `-INACTIVE`, чтобы отключить все индексы, первичные, уникальные и внешние ключи. Если есть ограничения `CHECK`, зависящие от исключенных данных, возможно, потребуется также указать опцию `-NO_VALIDITY`.

`-keep_d[ata] <шаблон>`

В резервную копию или восстановленую БД включаются только данные указанных таблиц.

В бэкап (или рестор) попадают данные только тех таблиц, имена которых соответствуют указанному здесь регулярному выражению в SQL-синтаксисе. Метаданные остальных таблицы включаются, а данные пропускаются.

Противоположная опция `-SKIP_D[ATA]`.

Если `<шаблон>` для `-KEEP_DATA` и `-SKIP_DATA` соответствует одной и той же таблице, то таблица пропускается.

Выборочное включение данных таблиц в резервную копию или восстановленную базу данных может вызвать ошибки во время восстановления, если существует внешний ключ в таблице, которая была включена, связанный с таблицей, которая не была включена.

Можно использовать опцию восстановления `-INACTIVE`, чтобы отключить все индексы, первичные, уникальные и внешние ключи. Если есть ограничения `CHECK`, зависящие от не включенных данных, возможно, потребуется также указать опцию `-NO_VALIDITY`.

Замечания

Резервное копирование может быть выполнено в любой момент, во время работы пользователей с базой данных. Эту операцию можно и нужно автоматизировать, сделав резервное копирование регулярным. Без резервных копий вы рискуете остаться ни с чем, если база данных окажется повреждена по какой-либо причине.

Отсюда же следует, что категорически нельзя делать резервные копии на тот же самый логический диск, где находится база данных. Еще лучше делать резервные копии на другой физический диск, поскольку чтение и запись будут разделены, и это даст как минимум 30% ускорение процесса резервного копирования.

8.2.5 Восстановление базы данных из резервной копии

Для восстановления базы из резервной копии действует следующая команда:

```
gbak -C[REATE_DATABASE] [restore опции] [общие опции] <файл резервной копии> <база данных>
```

Ключ `-C` означает, что необходимо восстановить базу данных из резервной копии, путь к которой указан как `<файл резервной копии>`, а результаты восстановления сохранить в файл, указанный как `<база_данных>`. Недостающие каталоги, если такие имеются, в пути к файлу будут автоматически созданы. Файл `<база_данных>` не должен существовать, иначе произойдёт ошибка.

Вместо данной команды можно использовать `-R[ECREATE_DATABASE] [O[VERWRITE]]` (восстанавливает БД в новый файл или восстанавливает ее вместо старой, если используется параметр `o`) и `-REP[LACE_DATABASE]` (восстанавливает БД, заменяя имеющуюся базу данных).

```
gbak -c emp.fbk emp.fdb
```

База `emp.fdb` будет восстановлена из резервной копии `emp.fbk`. Процесс восстановления базы данных из резервной копии происходит следующим образом:

1. Сервер создает пустую базу данных `emp.fdb`. ODS базы данных будет определяться версией сервера, а не версией `gbak`, т. к. только сервер создает базу данных, а `gbak` при restore всего лишь ее наполняет метаданными и данными.
   Пустая база данных будет содержать все таблицы `rdb$` (пока пустые), и будет иметь ряд параметров, например, такие как размер страницы, forced write и т. д., которые или взяты из резервной копии, или установлены в командной строке `gbak`.
2. Сервер считывает метаданные (описания таблиц и индексов, процедур) из резервной копии и переносит их в базу данных.
3. Сервер считывает данные из резервной копии и переносит их в базу данных.
4. Сервер считывает остальные метаданные (триггеры, гранты, check constraints и т. п.) из резервной копии и переносит их в базу данных.
5. Сервер создает (активирует) все индексы таблиц (которые были активны в момент создания резервной копии).

То есть, восстановленная из резервной копии база данных будет на самом деле не копией старой базы, а совершенно новой базой данных, наполненной старыми данными.

Помните, что если вы делаете восстановление на новой версии сервера, например, резервную копию делали на Ред Базе Данных 2.6 (формат БД ODS 11.2), а восстанавливаете на Ред Базе Данных 3.0(формат БД ODS 12.3), то база будет создана в формате, поддерживаемом по умолчанию Ред Базой Данных 3.0, и Ред База Данных 2.6 с этой базой работать не сможет.

Узнать версию ODS поможет команда:

```
gstat -h <база_данных>
```

Резервные копии тоже имеют свой формат, и резервная копия базы данных, сделанная например в Ред Базе Данных 3.0 утилитой `gbak` этой же версии, не может быть восстановлена утилитой `gbak` от Ред Базы Данных 2.6.

Теперь поподробнее остановимся на опциях бэкапа. О некоторых общих опциях (для бэкапа и рестора) : -fe[tch_password], -m[eta_data], -skip_d[ata], -keep_d[ata], -v[erify], -verbi[nt], -y уже шла речь в предыдущем разделе. О дополнительных параметрах restore рассказано ниже.

-R[ECREATE_DATABASE] [O[VERWRITE]]

Восстановление в новую БД, при необходимости позволяя перезаписать существующую БД.

Эта команда аналогична команде `-C[REATE_DATABASE]`, которая выполняет восстановление базы данных в новый файл. Но если указанный файл уже существует, будет выдано сообщение об ошибке.

Избежать такую ошибку поможет опция `o[verwrite]`. В таком случае файл с существующей базой данных будет молча удален и вместо него создан новый с тем же именем.Этого допускать нельзя, потому что по разным причинам восстановление из резервной копии может не состояться, и тогда вы останетесь без оригинальной базы данных и с невосстановимой резервной копией.

Если вы замените открытую и работающую базу данных, есть большая вероятность, что вы ее испортите. Для достижения наилучших результатов и минимальной вероятности повреждения базы данных следует закрыть ее перед заменой. Чтобы закрыть базу данных, используйте `gfix` следующим образом:

```
gfix -shut -tran 60 employee.fdb
```

В приведенном выше примере предотвращается запуск любой новой транзакции, что предотвращает выполнение новых запросов или новых сеансов подключения к базе данных. Прежде чем завершить работу базы данных, `gfix` будет ждать до 60 секунд, пока все выйдут из системы и завершится все текущие транзакции. Если какие-либо длительные транзакции не завершатся по истечении 60 секунд, время завершения работы истечет, а база данных останется открытой.

> Мы настоятельно не рекомендуем использовать команду `-r[ecreate_database] o[verwrite]` с целью избежания потери данных.

Этот ключ намеренно сокращен до `-r`, чтобы не допустить, чтобы ничего не подозревающие администраторы баз данных перезаписали существующую базу данных, думая, что `-r` был сокращен от `-restore`. В более старых версиях `-r` фактически был аббревиатурой `-REP[LACE_DATABASE]` и делал восстановление путем сначала удаления существующей базы данных, а затем ее воссоздания из резервной копии.

Использование `-r[ecreate_database] o[verwrite]` фактически аналогично использованию `-rep[lace_database]`.

-REP[LACE_DATABASE]

Восстановление в базу данных, позволяя перезаписать существующую базу данных.

Если база данных уже существует, то она будет удалена перед восстановлением. Этого допускать нельзя, потому что по разным причинам восстановление из резервной копии может не состояться, и тогда вы останетесь без оригинальной базы данных и с невосстановимой резервной копией.

Команда `-rep` полностью аналогична `-r o`.

> Мы настоятельно не рекомендуем использовать команду `-rep` с целью избежания потери данных.

-bu[ffers] <n>

Изменяет размер кэша базы данных.

По умолчанию кэш базы данных задан в файле конфигурации сервера (параметр `DefaultDbCachePages` в `firebird.conf`), и равен 2048 страниц. Это значение действует для всех баз данных, у которых размер кэша задан неявно. Если вы используете на сервере несколько баз данных, то может потребоваться указать для них разный размер кэша, в зависимости от назначения этих баз.

Узнать текущий размер кэша поможет команда `gstat -h <база данных>`.

Параметр `-bu <n>` позволяет при восстановлении базы данных задать или изменить этот размер. К сожалению, у `gbak` можно указать значение `-bu` только больше 0. Если вы обнаружили, что в бэкапе уже "зашито"значение кэша, то "сбросить"его при restore не получится. Для убирания размера кэша в БД придется использовать `gfix`.

Если размер кэша вообще задан в БД (а в большинстве случаев это не делают), то этот параметр используется обычно при переносе базы данных, например, со старого сервера на новый, или при переносе БД между архитектурами Classic и SuperServer.

Эквивалентно команде `gfix -bu <n>`.

```
gstat -h employee.fdb | grep -i buffer # Проверить текущий размер кэша
Page buffers      0
```

```
gbak -c -buffer 200 /backups/employee.fbk employee.fdb
gstat -h employee.fdb | grep -i buffer
Page buffers      200
```

-fix_fss_d[ata], -fix_fss_m[etadata]

Изменение набора символов метаданных (и данных)

Опции `-fix_fss_metadata` и `-fix_fss_data` предназначены для исправления набора символов (charset)метаданных (и данных), которые были записаны в некорректной кодировке. Например, при редактировании процедур или триггеров в код могли попасть комментарии или константы в кодировке `none`, в то время когда они на самом деле являются символами `Windows-1251`.

В штатной ситуации указывать эти опции не требуется, но если при восстановлении `gbak` выводит сообщение об ошибке:

```
Malformed string
```

то нужно принудительно указать нужную кодировку при помощи указанных опций. После этого в столбцах Unicode данные будут записаны корректно.

`gbak` также может выдавать сообщение:

```
gbak:Invalid metadata detected. Use -FIX_FSS_METADATA option.
```

что сигнализирует об аналогичной ситуации, и требует явного указания данной опции с нужной кодировкой.

> Указывать опции `-fix...` можно **только один раз**. Если вы сделаете еще раз backup/restore этой же базы данных с этими опциями, то исходные тексты процедур и триггеров будут испорчены.
>
> Указание неправильного имени набора символов может привести к логическому повреждению ваших данных.

> Не используйте эту опцию без четкого понимания того, что она делает. Неправильное использование может повредить ваши данные вместо того, чтобы что-то исправить. Всегда сохраняйте копию исходной базы данных и ее резервную копию.

-i[nactive]

Не выполнять создание (активацию) индексов (последняя фаза восстановления).

После восстановления базы данных все индексы в ней останутся отключены (неактивны). Фактически с такой базой данных работать нельзя, т. к. при отключенных индексах `Primary key`, `Foreign key` и `Unique` возможны нарушения целостности данных в таблицах (дублирование первичных ключей и т. д.).

Данный параметр имеет смысл использовать разве что в специфических целях – например, восстановить только данные из бэкапа за максимально быстрое время, или определить длительность создания всех индексов (замерить время `gbak -c`, затем замерить время `gbak -c -i`, после чего вычесть одно время из другого – получите длительность активации всех индексов), чтобы определить или текущую производительность каталога `temp`, или сравнить ее с явно заданным в конфигурации расположении `temp` на другом физическом диске.

Если вы восстановили бэкап с опцией `-i`, индексы можно активировать командой `alter index <имя индекса> active` (для каждого индекса).

-ig

Игнорировать ошибки в BLR процедур, функций и триггеров.

Если эта опция не указана и в процессе восстановления в BLR находится ошибка, восстановление останавливается. Если использовать ключ `-ig` восстановление будет продолжено, но неисправный BLR-код хранимых процедур, функций и триггеров будет представлять `NULL`. В конце лога восстановления будет выдано предупреждение:

```
gbak: WARNING:Database is not online due to failure to restore one or more objects.
```

> Если в логе восстановления появилось такое предупреждение обязательно перекомпилируйте исходный текст объектов.

Исходный код для перекомпиляции объектов можно получить с помощью следующих запросов:

```
select RDB$PROCEDURE_SOURCE from RDB$PROCEDURES where RDB$PROCEDURE_BLR is NULL;
select RDB$FUNCTION_SOURCE from RDB$FUNCTIONS where RDB$FUNCTION_BLR is NULL;
select RDB$TRIGGER_SOURCE from RDB$TRIGGERS where RDB$TRIGGER_BLR is NULL;
```

-k[ill]

Восстановить базу данных без создания теневый копий

Эта опция восстанавливает базу данных, но не воссоздает ранее существовавшие теневые копии. На самом деле этот параметр не только "не создает"shadow, но и еще удаляет существующие, например в варианте с backup/restore с промежуточным переименованием базы данных.

```
gbak -b -kill /backups/employee.fbk employee.fdb
```

Впрочем, поскольку использовать `shadow` сейчас нет смысла, то и параметр `-k` можно считать устаревшим.

Эквивалентно команде `gfix -kill`.

-mo[de] <режим доступа>

Восстанавливает базу данных в режиме read_ write или read_ only

```
-MO[DE]  READ_ONLY | READ_WRITE
```

По умолчанию режим берется от базы данных, для которой была создана резервная копия.

```
gbak -b -mode read_only employee.fbk employee.fdb
```

Эквивалентно команде `gfix -mode read_write/read_only`.

-n[o_validity]

Отключает ограничения CHECK

Эта опция аналогична `-i[nactive]`, за исключением того, что она отключает ограничения `CHECK` в востанавливаемой базе данных.

-o[ne_at_a_time]

Подтверждать транзакцию после восстановления каждой таблицы.

Эта опция восстанавливает данные потаблично, используя транзакцию для каждой таблицы. Она может быть полезна, если предыдущее восстановление не удалось из-за ошибок данных. Обычно восстановление выполняется за одну транзакцию с одним коммитом в конце восстановления. Если восстановление по какой-либо причине прерывается, конечным результатом является пустая база данных. С опцией `-o[ne_at_a_time]` запускается транзакция для каждой таблицы и после восстановления каждой таблицы выполняется коммит.

-p[age_size] <размер страниц>

Позволяет задать новый размер страницы базы данных.

По умолчанию база данных восстанавливается с тем же размером страницы базы данных, что и исходная база данных (как записано в файле резервной копии).

Посмотреть размер страницы базы данных можно с помощью команды `gstat -h`.

```
gstat -h employee | grep -i "page size"Page size 4096
gbak -b -page_size 8192 /backups/employee.fbk employee.fdb
gstat -h employee | grep -i "page size"Page size 8192
```

Сделать рестор с параметром `-p` - это единственный способ, которым можно изменить размер страницы базы данных. В зависимости от версии допустимые размеры страниц: 1024, 2048, 4096, 8192, 16384 и 32768 байт. Если у базы данных размер страницы 1024 или 2048 байт – рекомендуется сделать backup и restore с указанием размера страницы 4096, 8192 или 16384 байт (если ваша версия сервера поддерживает страницы в 16к). Потому что даже небольшие базы данных с таким небольшим размером страницы имеют явно худшую производительность, чем базы со страницей 4 килобайта и выше.

-par[allel] <n>

Количество потоков, используемых для восстановления из резервной копии.

Ред База Данных (начиная с версии 3.0) поддерживает многопоточный restore. Он показал себя до 2-х раз быстрее обычного рестора в один поток. Реальное ускорение зависит от процессора, дисковой подсистемы и структуры базы данных.

По умолчанию используется один поток. Но для рестора это не идентично явному указанию `-PARALLEL 1`.

Для включения распараллеливания в `gbak` появилась опция `-par <n>`:

```
gbak -c backup database -par 8 ...
```

а также два параметра конфигурации `ParallelWorkers` и `MaxParallelWorkers`.

Для рестора параллельная обработка реализована только при построении индексов. При этом внутри ядра создаются `N` потоков, где `N` - либо значение, переданное из `gbak`, либо значение параметра конфига `ParallelWorkers` (если при ресторе не задан ключ `-PAR`). `N` должен быть меньше значения параметра конфигурации `MaxParallelWorkers` (максимальное допустимое значение - 64), иначе операция завершится с ошибкой. По умолчанию `MaxParallelWorkers = 1`, т.е. параллельное создание индексов отключено даже если в ресторе задан ключ `-PAR`.

Параллельные потоки одновременно читают таблицу и сортируют её записи. Затем один основной поток строит дерево сортировки (B+tree) и создаёт индекс.

Т.к. при параллельном создании индексов рабочие потоки внутри ядра создают собственные подключения к базе, она не может быть восстановлена в режиме `single shutdown` как это было раньше. Поэтому при использовании опции `-par` при ресторе база создаётся в режиме `multi shutdown`, т.е. в процессе рестора к ней может подключиться `SYSDBA` или владелец базы и сделать с ней что хочет.

-use_[all_space]

Максимально заполнять страницы данных (для read-only баз).

По умолчанию базы данных Ред Базы Данных резервируют примерно 30% пространства на страницах данных, для размещения версий при будущих вставках, удалениях или обновлениях записей. Если предполагается запись базы данных на диск, то лучше базу данных несколько "сжать указав параметр `-use_` при restore (одновременно указав `-mode read_only`).

Эквивалентно команде `gfix -use full`. Обратная команда (снятие режима максимального заполнения страниц) – `gfix -use reserve`.

Замечания

Восстановление (например, проверочное) на тот же самый диск, где находится резервная копия, не так опасно, как в случае backup (когда свободное место может кончиться, и резервная копия будет неполной, да еще и базаданных может быть повреждена). Но разумеется имеет те же самые проблемы с производительностью, когда восстановление проводится на тот же самый физический диск (поочередные чтение и запись с разных участков диска). Поэтому при восстановлении базы данных из резевной копии рекомендуется использовать разные физические диски.

8.2.6 Работа с GBAK через Services API

Как уже было сказано, `gbak` - это программа, которая сама выполняет команду резервного копирования, "прокачивая"данные через себя. Но утилита может работать и в другом режиме, используя Services API для выполнения сервером по команде действий, аналогичных `gbak`. Утилита `gbak`, использующая Services API, отправляет серверу команду, а сервер ее выполняет сам (то есть не `gbak`, а сам сервер будет выполнять резервное копирование). В этом случае программа, отдавшая команду, может получать лог выполнения от сервера и выдавать все, что сообщит о процессе сервер. Чтобы включить работу `gbak` в этом режиме, используйте опцию `-se`:

```
gbak -b -g -se server:service_mgr c:\db\e.fdb d:\bak\e.fbk ...
```

```
gbak -c -se server:service_mgr d:\e.fbk c:\db\e.fdb
```

`server` – имя компьютера, где находится сервер Ред Базы Данных (если на этом же, то можно указать `localhost`). `server` можно не указывать, если сервер "локальный и работает локальный протокол:

```
gbak -b -g -se service_mgr c:\db\e.fdb d:\bak\e.fbk ...
```

`service_mgr` – имя интерфейса Services API, оно обязательно, неизменно, и пока только одно (может быть в дальнейшем появится что-то еще).

Поскольку не `gbak`, а сам сервер теперь выполняет резервное копирование, то есть несколько требований:

- пути к базам (или алиасы) должны быть указаны только серверные, как для базы так и для файла резервной копии.

- имя сервера к имени базы данных добавлять не нужно, т. к. оно уже должно быть указано как опция команды `-se`.
- у сервера должны быть права на запись туда, куда сохраняется резервная копия. На Windows сервер по умолчанию стартует под учетной записью LocalSystem, которая не имеет и не может иметь прав на внешние ресурсы (например, шаренные папки). Поэтому, если вы хотите сохранять резервные копии БД на другой компьютер сразу, а не путем копирования получившегося на сервере файла backup – нужно создать пользователя (например, `reddatabase`), дать этому пользователю права на папки установки сервера, папки с базами данных и папки с резервными копиями, и затем остановить сервер и запустить его указав в параметрах сервиса новое имя пользователя.

По результатам тестирования резервное копирование через Services API является самым быстрым, по сравнению с локальным протоколом или tcp/ip (результаты тестирования). Однако, если потребуется прекратить процесс backup или restore, то:

- для `gbak -b/-c` достаточно принудительно снять (завершить) процесс `gbak.exe`, или если он запущен интерактивно, нажать в окне cmd Ctrl-C. Поскольку backup или restore выполняется не сервером, а утилитой `gbak`, этот процесс будет прекращен.
- для backup и restore, выполняемых через Services API, это возможно только остановкой сервера, т. к. именно он выполняет этот процесс.

8.3 Утилита NBACKUP

`Nbackup` позволяет создавать резервные копии и восстанавливать из резервных копий так же, как `gbak`, и дополнительно позволяет создавать инкрементные копии и восстанавливать из них БД. Инкрементная резервная копия содержит только изменения со времени создания определенной, ранее созданной резервной копии.

Второе назначение `Nbackup` – блокировать файл базы данных. Таким образом, Вы после этого сможете сами создавать обычные копии или резервные копии с помощью утилит по Вашему выбору. В этом режиме `Nbackup` ничего не резервирует, а лишь создает подходящие условия, чтобы Вы могли без каких бы то ни было проблем создавать резервные копии.

Таблица 8.8 — Опции основных задач утилиты `NBACKUP`

Опция	Описание
`-L(OCK) <база данных>`	Блокировка базы данных для резервирования. Это означает, что основной файл базы данных временно замораживается с возможностью внесения изменений. Как и в режиме резервирования, изменения фиксируются во временном файле дельты.
`-UN(LOCK) <база данных>`	Разблокировка ранее заблокированной базы данных. Сначала определяется наличие любых изменений с момента блокирования базы данных и производится объединение временного файла дельты и основного файла базы данных. После этого база данных переводится в нормальный режимчтения/записи, а временный файл удаляется.

Опция	Описание
-F(IXUP) <база данных>	Разблокировка базы данных после самостоятельного восстановления из блокированной резервной копии. Так как копия блокированной базы данных является так же блокированной, поэтому не получится использовать копию как рабочую базу данных. Поэтому, если исходная база данных повреждена или утеряна, то нужно восстановить/разархивировать/скопировать базу данных из копии и разблокировать ее с параметром -F (но не -UN).
-B(ACKUP) <уровень>|<GUID> <база данных> [<резервный файл>]	Создание резервной копии всей базы данных или инкрементных резервных копий[6]
-R(ESTORE) <база данных> [<резервный файл 0> [<резервный файл 1>...]]	Восстановление из резервной копии всего файла базы данных или из инкрементных резервных копий

Таблица 8.9 — Специальные опции утилиты NBACKUP

Опция	Описание
-D(IRECT) [ON | OFF]	Включает/выключает небуферизованный ввод/вывод при чтении БД
-I	Восстанавливает инкрементную копию в существующую базу данных[7]
-S(IZE)	Вывести количество страниц в базе после блокировки БД [8].
-DE(COMPRESS) <команда>	Команда для разархивирования бэкапа во время восстановления. Символ @ в команде соответствует файлу бэкапа.

Таблица 8.10 — Общие опции утилиты NBACKUP

Опция	Описание
-NOD(BTRIGGERS)	Не запускать триггеры базы данных
-U(SER) <имя пользователя>	Имя пользователя
-P(ASSWORD) <пароль пользователя>	Пароль пользователя
-RO(LE) <роль>	Роль пользователя
-FETCH_PASSWORD <файл>	Считывает пароль из файла
-Z	Выдает версию СУБД

Опция -I может повредить базу данных, если она была изменена со времени предыдущего восстановления. Кроме того, эта опция после восстановления инкрементного бэкапа переводит БД в режим "только для чтения".

Создание резервной копии всей базы данных

Nbackup может работать с активной базой данных, не мешая подключенным к ней пользователям. Созданная резервная копия базы данных всегда будет отображать состояние базы данных на момент начала создания резервной копии.

Синтаксис nbackup для создания резервной копии всей базы данных:

[6] GUID базы данных можно узнать из вывода утилиты gstat с параметром -h
[7] Работает только с опцией -R
[8] Работает только с опцией -L(OCK)

```
nbackup [-U <пользователь> -P <пароль> [-RO <роль>]] -B 0 <база_данных>
[<резервный_файл>]
```

Например:

```
nbackup -user testuser -password pass -role rdb$admin -b 0 base.fdb
base_10_04_17.nbk
```

Имя пользователя и пароль можно не указывать, если в системе установлены контекстные переменные `ISC_USER` и `ISC_PASSWORD`. А также если используетсядоверительная аутентификация Windows (`Win_Sspi`) илидоверенная через механизм GSSAPI (`gss`), и пользователь системы, от имени которого запускается утилита ISQL, является членом группы администраторов.

Необязательный переключатель `-role` задаёт имя роли, права которой будут учитываться при выполнении каких-либо действий утилиты. Поэтому если пользователь не указывает роль, то он получает права только тех ролей, которые ему назначены с `DEFAULT`. Например, если в команде выше не указать роль пользователя (предположим роль `rdb$admin` не была назначена ему с `DEFAULT`), то выдастся сообщение об ошибке:

```
-ALTER DATABASE failed
-no permission for ALTER access to DATABASE
```

Уровень резервной копии 0 означает создание резервной копии всей базы данных. Уровни резервных копий больше 0 используются для создания инкрементных резервных копий.

Вместо имени файла базы данных можно указать псевдоним (alias, из файла `databases.conf`).

Вместо имени файла резервной копии можно указать `stdout`. Это перенаправит резервную копию в стандартный поток вывода, откуда Вы сможете перенаправить ее, например, на ленточный накопитель или на вход утилиты для сжатия получаемой резервной копии.

Восстановление из резервной копии всего файла базы данных

Резервная копия всей базы данных восстанавливается следующим образом:

```
nbackup [-U <пользователь> -P <пароль> [-RO <роль>]] -R
<база_данных>[<резервный_файл>]
```

Например:

```
nbackup -user sysdba -password masterkey -r base.fdb base_10_04_17.nbk
```

Уровень не указывается при восстановлении. Параметр `-R` должен быть указан последним.

> Если указанная база данных уже существует и нет активных соединений, она будет перезаписана без предупреждения! Если есть активные соединения, восстановление не состоится, и Вы получите сообщение об ошибке.

Создание инкрементных резервных копий

Инкрементная резервная копия базы данных содержит только изменения с момента создания последней резервной копии. Существует два подхода к создания инкрементной резервной копии:

- По уровню резервной копии (больше 0);
- С помощью `GUID` последней созданной резервной копии базы данных.

Создание инкрементных резервных копий для многофайловых баз данных еще не поддерживается.

Создание инкрементных резервных копий разных уровней

Инкрементная резервная копия уровня N содержит изменения базы данных с момента создания последней резервной копии уровня $N-1$.

Примеры:

Через день после создания резервной копии всей базы данных (уровня 0) создается резервная копия уровня 1:

```
nbackup -B 1 base.fdb base_11_04_17.nbk
```

Эта резервная копия будет содержать только изменения базы данных за последний день. Если через день вновь создать резервную копию уровня 1:

```
nbackup -B 1 base.fdb base_12_04_17.nbk
```

Эта копия будет содержать изменения за последние два дня, то есть с момента создания резервной копии всей базы данных, а не только с момента создания предыдущей инкрементной копии уровня 1.

Далее, при создании резервной копии уровня 2 (допустим, в тот же день):

```
nbackup -B 2 base.fdb base_12_04_17_2.nbk
```

Эта резервная копия будет содержать изменения только с момента создания последней резервной копии уровня 1 (то есть за несколько часов).

Создание инкрементных резервных копий с помощью GUID

Создать инкрементную резервную копию базы данных можно используя `GUID`. Такой способ не требует знания уровня резервной копии для создания инкремента. Резервная копия будет содержать только изменения с момента создания последней резервной копии.

После создания резервной копии всей базы данных (уровня 0) можно создать инкрементную резервную копию. Для этого сначала необходимо узнать `GUID` последней резервной копии (`Database backup GUID`). Это можно сделать с помощью утилиты `gstat` с параметром `-h`.

Синтаксис для создания инкрементной резервной копии базы данных:

```
nbackup [-U <пользователь> -P <пароль> [-RO <роль>]] -B <GUID> <база_данных>
[<резервный_файл>]
```

Например:

После создания резервной копии всей базы данных (уровня 0) узнаём её `GUID`:

```
gstat -h base.fdb
```

В выводе будет `GUID` последней резервной копии (`Database backup GUID`):

```
Variable header data:
   Database backup GUID: {DEE5ECBE-3731-4FB4-4693-2D96439FD746}
   Database GUID: {6A03998A-33B5-2BCC-B8CB12CAA7ED}
```

Далее создаём инкрементную резервную копию:

```
nbackup -B {DEE5ECBE-3731-4FB4-4693-} base.fdb base_11_04_17.nbk
```

После этого GUID в выводе утилиты gstat изменится и можно будет создать инкрементную резервную копию, которая будет содержать изменения, внесённые с момента создания последней резервной копии.

Восстановление из инкрементных резервных копий

При восстановлении базы данных из инкрементных резервных копий необходимо обеспечить наличие полной цепочки инкрементных резервных копий, начиная с уровня 0 и до уровня, которым необходимо завершить восстановление.

Синтаксис:

```
nbackup [-U <пользователь> -P <пароль> -RO <роль>] -R <база_данных>
[<резервная_копия0> [<резервная_копия1> [...] ] ]
```

Таким образом, восстановление для предыдущего примера до уровня 2 будет выглядеть так:

```
nbackup -R base.fdbbase_10_04_17.nbk base_12_04_17.nbk base_11_04_17_2.nbk
```

Nbackup считает все аргументы после параметра -R именами файлов с резервными копиями. По этой причине никакие другие параметры (-U или -P) не могут следовать за списком файлов параметра -R.

Не существует формального ограничения на уровень резервной копии, однако на практике редко имеет смысл создавать копии уровней больше 3 или 4.

Блокировка базы данных и самостоятельное резервное копирование

Если Вы предпочитаете использовать какие-то другие утилиты для создания резервных копий базы данных или просто делать обычную копию базы данных как резервную, то для этого существует режим блокировки/разблокировки программы nbackup. «Блокировка» в данном случае означает, что основной файл базы данных временно замораживается, а не невозможность внесения изменений в базу данных. Как и в режиме резервирования, изменения фиксируются во временном файле дельты; при разблокировании файл дельты объединяется с основным файлом базы данных.

Самостоятельное резервное копирование обычно происходит по следующему сценарию:

1. Блокировка базы данных с помощью параметра -L(OCK):

   ```
   nbackup [-U <пользователь> -P <пароль> [-RO <роль>]] -L<база_данных>
   ```

2. Создание резервной копии, сжатие файла базы данных, используя любые другие программы или простое копирование файла с базой.
3. Разблокировка базы данных с помощью параметра -UN(LOCK):

   ```
   nbackup [-U <пользователь> -P <пароль> [-RO <роль>]] -UN <база_данных>
   ```

Восстановление из резервной копии, сделанной после блокировки

Копия блокированной базы данных является так же блокированной, поэтому Вы не сможете просто использовать копию как рабочую базу данных. В случае, если Ваша исходная база данных повреждена или утеряна, и нужно восстановить базу данных из копии, сделанной Вами самостоятельно, действуйте следующим образом:

1. Разархивируйте/скопируйте/восстановите файл базы данных с помощью используемых Вами утилит.
2. Теперь разблокируйте базу данных, но не с параметром `-UN(LOCK)`, а с параметром `-F(IXUP)`:

```
nbackup -F <база_данных>
```

При использовании параметра `-UN` сначала определяется наличие любых изменений с момента блокирования базы данных (после использования параметра `-L`) и производится объединение временного файла дельты и основного файла базы данных. После этого база данных переводится в нормальный режим чтения/записи, а временный файл удаляется. При использовании параметра `-F` только изменяется в «нормальное» значение флага состояния самостоятельно восстановленной базы данных.

8.4 Утилита GFIX

Эта утилита является одним из основных инструментов администратора БД. Утилита `GFIX` позволяет:

- Выполнять принудительную чистку (`sweep`) базы данных;
- Изменять интервал автоматической чистки;
- Закрывать базу данных для получения монопольного доступа, и затем снова открывать ее;
- Переводить базу в режимы «чтение/запись» или «только чтение»;
- Переключаться между синхронным и асинхронным вводом (Forced Writes);
- Изменять диалект БД;
- Устанавливать размер кэша;
- Изменять GUID базы данных;
- Отыскивать повисшие транзакции и отменять или подтверждать их;
- Активировать или удалять теневые копии;
- Производить ремонт поврежденных баз данных.

Запуск `GFIX` осуществляется следующим образом:

```
gfix [<опции>] <имя базы данных>
```

Если база данных состоит из нескольких файлов, то при запуске `gfix` необходимо указать первичный файл.

Для того, чтобы выполнить любую операцию над базой данных с помощью `gfix`, необходимо пройти процедуру идентификации и аутентификации — указать имя и пароль пользователя, который имеет право на запуск сервиса `gfix` (подробнее см. п. 9.3). Для указания имени и пароля пользователя используются переключатели `-user` и `-pa[ssword]`, соответственно, например:

```
gfix -user testuser -pa pass -role rdb$admin [опции] <имя базы данных>
```

Необязательный переключатель `-role` задаёт имя роли, права которой будут учитываться при выполнении каких-либо действий утилиты. Поэтому если пользователь не указывает роль, то он получает права только тех ролей, которые ему назначены с `DEFAULT`.

Имя пользователя и пароль можно не указывать, если в системе установлены контекстные переменные `ISC_USER` и `ISC_PASSWORD`. А также если используетсядоверительная аутентификация Windows (`Win_Sspi`) илидоверенная через механизм GSSAPI (`gss`), и пользователь системы, от имени которого запускается утилита ISQL, является членом группы администраторов.

Набор всех возможных опций утилиты `GFIX`, представлен ниже.

Таблица 8.11 — Опции GFIX

Операция	Описание
`-ac(tivate_shadow) <теневая копия>`	Параметр предназначен для активации теневой копии. `<теневая копия>` -адрес и имя файла теневой копии (или первого из файлов).
`-at(tach) <n>`	Дополнительный параметр к `-shut`. Предназначен для запрета новых соединений с БД. `<n>` указывает количество секунд, через которое произойдет отключение БД. Отключение отменится, если к этому времени еще останутся активные соединения.
`-b(uffers) <n>`	Устанавливает новый размер страничного кэша БД. `<n>` - количество страниц.
`-co(mmit) {<ID> | all}`	Завершает подтверждением зависшую транзакцию с идентификатором `<ID>`, или все зависшие транзакции (`all`)
`-ca(che)`	Параметр не используется.
`-fu(ll)`	Используется вместе с `-v(alidate)` для более глубокой проверки на уровне записей; освобождает неназначенные фрагменты записей.
`-fo(rce_shutdown) <n>`	Дополнительный параметр к `-shut`. Предназначен для принудительного закрытия базы данных. `<n>` указывает количество секунд, через которое произойдет закрытие. Если остались активные пользователи, они отключатся, последние результаты их работы будут потеряны. Такое средство нужно применять с осторожностью, как последнюю возможность.
`-fe(tch_password)`	Извлекает пароль из файла.
`-g(uid)`	Изменяет GUID базы данных. GUID генерируется случайно.
`-h(ousekeeping) <n>`	Изменяет интервал транзакций для автоматической чистки `sweep`. `<n>` устанавливает новый интервал. Если `n = 0`, автоматическая чистка запрещена.
`-i(gnore)`	Игнорировать ошибки контрольных сумм при проверке или чистке.
`-icu`	Исправляет базу данных для работы с текущей версией ICU.
`-k(ill_shadow) <база данных>`	Удаляет все неиспользуемые теневые копии базы данных.
`-l(ist)`	Показывает некоторые сведения (в том числе ID) о всех зависших транзакциях.

Операция	Описание
-me(nd)	Подготавливает повреждённую базу данных для резервного копирования. Помечает повреждённые записи как неиспользуемые.
-mo(de) {read_write | read_only}	Устанавливает режим записи для базы данных - только для чтения или чтение/запись. Этот параметр может принимать два значения.
-nol(inger)	Останавливает указанную базу данных, после того как последнего соединения не стало, независимо от установок LINGER в базе данных.
-n(o_update)	Используется вместе с -v(alidate) для проверки разрушенных или неразмещенных структур. Если таковые есть, они отобразятся в сообщении, но не будут исправлены.
-o(nline) {single|multi|normal}	Возвращает в режим "online" базу, остановленную с помощью -shut. Опция normal позволяет устанавливать соединение с БД любым авторизованным пользователям, а не только SYSDBA и владельцем БД. Опция multi позволяет устанавливать соединение с БД только SYSDBA и владельцем БД. Опция single похожа на multi, за тем исключением, что только один из пользователей – SYSDBA или владелец БД – может соединиться.
-pr(ompt)	Используется вместе с -l(ist). Выводит подсказки при восстановлении транзакций.
-pa(ssword) <пароль>	Пароль пользователя для работы с gfix.
-par(allel) <n>	Количество параллельных рабочих потоков для сборки мусора. Включает многопоточную сборку мусора, если MaxParallelWorkers > 1 (см. firebird.conf). Вместо этого параметра утилиты можно указать параметр конфигурации ParallelWorkers.
-replica {GUID}	Активация режима репликации.
-role	Роль пользователя для работы с gfix.
-r(ollback) {<ID> | all}	Завершает откатом зависшую транзакцию с идентификатором <ID>, или все зависшие транзакции (all)
-sq(l_dialect) <n>	Изменяет диалект базы данных. <n> может быть 1 или 3.
-sw(eep)	Запускает сборку мусора в БД.
-sh(utdown) {single|multi|full}	Закрывает базу данных. Используется с одним из дополнительных параметров -attach, -force или -tran. Опция multi позволяет устанавливать соединение с БД только SYSDBA и владельцем БД. Опция single похожа на multi, за тем исключением, что только один из пользователей – SYSDBA или владелец БД – может соединиться. Опция full не позволяет соединиться с БД никому, даже SYSDBA и владельцу БД.
-tw(o_phase) {<ID> | all}	Автоматическое двухфазное восстановление limbo транзакции с номером <ID>, или всех транзакций (all).

Операция	Описание
`-tra(nsaction) <n>`	Дополнительный параметр к `-shut`. Предназначен для запрета запуска новых транзакций. `<n>` указывает количество секунд, через которое произойдет отключение БД. Отключение отменится, если к этому времени еще останутся активные транзакции.
`-tru(sted)`	Используется trusted авторизацию.
`-u(se) {full | reserve}`	Включает 100% заполнение страниц БД (`full`) или 80% заполнение по умолчанию (`reserve`). 100%-е заполнение имеет смысл для баз только для чтения.
`-user`	Имя пользователя для работы с `gfix`.
`-v(alidate)`	Проверяет базу данных на уровне страниц и освобождает страницы, помеченные как используемые, но никому не принадлежащие (`orphans`).
`-w(rite) {sync|async}`	Переключает режимы синхронной/асинхронной записи Forced Writes.
`-z`	Выводит версию Firebird и утилиты `gfix`.

Активация теневой (оперативной) копии

Файл теневой копии является дополнительной копией первичного файла(ов) базы данных. Для любой данной базы данных может существовать несколько теневых копий, которые могут быть активированы и деактивированы по умолчанию с помощью утилиты `gfix`.

Для активации теневой копии используется команда `-ac[tivate]`:

```
gfix -activate <файл теневой копии>
```

Это делает файл теневой копии новым файлом базы данных, и пользователи могут продолжать нормальную обработку данных и без потерь.

В случае, если ваш основной файл(ы) базы данных поврежден или нечитабелен, администратор базы данных может активировать теневой файл. После активации файл больше не является теневым файлом, и для его замены необходимо создать новый. Кроме того, теневой файл должен быть переименован на имя старого файла базы данных, который он заменяет.

Следует отметить, что активация теневой копии, в то время как сама база данных активна, может привести к повреждению тени. Перед ее активацией убедитесь, что файл базы данных действительно недоступен.

Администратор базы данных может настроить базу данных для автоматического создания нового теневого файла в случае активации текущей тени. Это позволяет обеспечить непрерывную поставку теневых файлов и предотвращает работу базы данных без нее.

Удаление теневых копий

Удаление всех недоступных теневых копий конкретной базы данных производится командой:

```
gfix -kill <база данных>
```

В случае, если база данных, работающая с теневыми файлами, теряет тень, или по какой-либо

причине тень становится непригодной, база данных перестанет принимать новые соединения до тех пор, пока администратор базы не уничтожит поврежденную тень и, в идеале, создаст новую тень для замены сломанной.

При соединении с локальной базой данных пользователь может не обладать правом на запуск сервиса `gfix`. Однако он сможет выполнить ту или иную операцию, только если у него есть соответствующие права. Например, для удаления теневой копии пользователь должен обладать правом на DDL-операцию `DROP SHADOW` — подробнее см. п. 9.3.

Установка размера кэша базы данных

Кэш (или буфер) базы данных - это оперативная память, выделяемая сервером для работы с базой данных. Операции в оперативной памяти происходят гораздо быстрее, чем если данные постоянно считываются с диска. Размер кэша указывается в страницах БД. Если размер страницы установлен 8192, то кэш в 5000 страниц займет примерно 40 мегабайт ОЗУ. Если сразу много пользователей одновременно обращаются к базе данных, может случиться, что серверу не хватит выделенной оперативной памяти. В этом случае он начнет работать с диском, что замедлит производительность БД. Изменить размер кэша для базы данных можно командой:

```
gfix -b[uffers] <кол-во страниц> <база данных>
```

Это действие применяется только к указанной вами базе данных. На другие базы данных, работающие на том же сервере, это изменение не повлияет.

Другим способом установить размер кэша по умолчанию для всех вновь создаваемых БД, является изменение конфигурационного файла `firebird.conf`, а именно параметра `DefaultDbCachePages`.

Однако более предпочтительным способом для этих целей является утилита `gfix`, так как она позволяет установить собственный размер кэша для каждой базы данных. Если какой-то базой данных пользуются реже, размер кэша для нее можно оставить по умолчанию, или даже уменьшить.

Управление limbo транзакциями

«Застрявшей» («зависшей») транзакцией (limbo) называют транзакцию, которая работает одновременно с двумя или более базами данных. При завершении такой транзакции, Ред База Данных совершает двухфазное подтверждение Commit, гарантируя, что изменения будут внесены либо во все БД, либо ни в одну. При этом подтверждения в базах данных будут даваться по очереди. Если в это время возникнет сбой системы, то может получиться, что в каких то БД изменения были сделаны, а в каких то нет. При этом транзакция переходит в неопределенное состояние, когда сервер не знает, следует ли подтвердить эту транзакцию, или откатить.

`Gfix` предоставляет несколько команд, позволяющих управлять этими транзакциями limbo.

Команда `gfix -l[ist]` будет отображать сведения о limbo транзакциях. Если команда не выдала результата, то зависших транзакций нет, и дальнейшие действия не требуются:

```
gfix -l[ist] <база данных>
```

Эту команду можно дополнить ключом `-pr[ompt]`, и тогда вам будет предложено выполнить `Commit` или `Rollback` для каждой обнаруженной транзакции limbo. В этом случае команда будет такой:

```
gfix -l[ist] -pr[ompt] <база данных>
```

Если обнаружено больше одной транзакции в состоянии `in-limbo`, то администратор может подтвердить (или откатить) все limbo транзакции или какую-то одну, по идентификатору:

```
gfix -commit {all | <ID>} <база данных>
gfix -rollback {all | <ID>} <база данных>
```

После срабатывания этих команд следует заново запустить команду `-list`, чтобы убедиться, что limbo транзакций больше не осталось.

`Gfix` можно использовать для автоматического двухфазного восстановления limbo транзакции с идентификатором, или всех транзакций:

```
gfix -tw[o_phase] {all | <ID>} <база данных>
```

Эти три команды можно также использовать вместе с ключом `-pr[ompt]`.

> Так как при запуске `gfix` можно указать только одну пару имяпользователя/пароль, то при восстановлении зависших двухфазныхтранзакций, которые работали с базами данных на разных серверах, логин и пароль пользователя, выполняющего восстановление, должны совпадать на всех серверах.

Установка режима доступа для базы данных

База данных может работать в одном из двух режимов доступа: только для чтения, или для чтения / записи (по умолчанию). Если вам понадобилось поменять режим, выполните команду:

```
gfix -mo {read_only | read_write} <база данных>
```

Например, если вы хотите разместить базу данных на CD диске, у вас не получится это сделать в режиме «чтения-записи». После того, как база данных будет заполнена данными, ее следует изменить в режим только для чтения, а затем использовать на CD диске (или других файловых системах только для чтения) без проблем.

Чистка базы данных

Вследствие того, что СУБД «Ред База Данных» имеет версионную архитектуру, со временем в ней могут накапливаться устаревшие версии записей, которые не нужны ни одной активной транзакции. В обычных условиях такие записи успешно удаляются «сборщиком мусора», но при возникновении ошибок в работе сервера «Ред База Данных» (например, из-за аппаратного сбоя), в базе данных могут остаться зависшие транзакции или фрагменты записей, которые не могут быть удалены обычным «сборщиком мусора». Большое число таких «мусорных» записей может привести к значительному росту размера БД и падению производительности. Для удаления таких записей применяется процедура чистки (`sweep`). Чистка может производится в ручном и автоматическом режиме. В автоматическом режиме администратор задает интервал чистки — разницу между Oldest Transaction (или Oldest Interesting Transaction, OIT – находится в любом состоянии, кроме подтвержденного) и Oldest Snapshot Transaction (OST). По умолчанию этот интервал равен 20000 транзакций. Изменить этот параметр для конкретной базы данных можно с помощью опции `-h[ouskeeping]`:

```
gfix -h[ouskeeping] <интервал> <база данных>
```

Если интервал равен 0, то в этом случае автоматическая чистка будет отменена.

Вручную чистку можно произвести с помощью команды `-sweep`:

```
gfix -sweep [-i[gnore]] <база данных>
```

Ключ `-i[gnore]` заставляет Ред Базу Данных игнорировать ошибки контрольной суммы на страницах базы данных. Лучше не использовать эту опцию, однако, если в вашей базе данных возникли некоторые проблемы, возможно, эта опция станет необходимой.

Сборка устаревших версий записей производится параллельно с обработкой запросов пользователя, не блокируя их работу.

Закрытие (блокировка) базы данных

Закрытие базы данных означает, что база данных переводится в особый режим, в котором к ней запрещены некоторые подключения, в зависимости от указанного состояния. Полноценная работа с закрытой базой данных возможна только после обратного ее перевода в открытый режим.

Закрытие базы данных может быть необходимо для получения монопольного доступа к ней и проведения действий по восстановлению структуры базы и/или хранящихся в ней данных.

Для закрытия БД используется команда `-sh[ut]`. Совместно с этой командой указывается режим отключения существующих соединений и время ожидания (в секундах) до полной блокировки:

```
gfix -sh[ut] {full|single|multi} {-at[tach]|-tr[an]|-f[orce]} <таймаут> <база
данных>
```

Состояние `full` означает, что запрещены любые подключения, даже `SYSDBA` и владельца базы данных.

Состояние `multi` означает, что пользователи `SYSDBA` и владелец базы данных могут неограниченно подключаться к базе данных. Все остальные соединения запрещены. Это состояние используется по умолчанию при блокировке базы данных.

Состояние `single` означает, что пользователю `SYSDBA` или владельцу базы данных позволено одно подключение к базе данных. Все остальные соединения запрещены.

Режим `-at[tach]` означает, что все новые соединения к базе данных запрещены. Существующие соединения при этом не разрываются. Число `<таймаут>` определяет количество секунд, которое сервер будет ждать до завершения всех активных соединений к базе данных. Если после этого не останется ни одного активного соединения, то база данных будет закрыта. В противном случае, закрытие БД будет отменено.

При указании режима `-tra[nsaction]` сервер заблокирует запуск новых транзакций в закрываемой базе данных. Если по истечении `<таймаут>` секунд к базе не останется ни одного активного подключения, то база данных будет закрыта. В противном случае закрытие БД будет отменено.

В режим `-fo[rce_shutdown]` сервер будет ждать завершения всех активных соединений к базе данных. По истечении времени `<таймаут>` база будет закрыта вне зависимости от того, есть ли еще к ней активные соединения. В этом режиме возможна потеря данных, но он гарантирует, в отличие от двух предыдущих, что база будет переведена в закрытое состояние.

Перевести закрытую базу снова в открытое состояние можно только с помощью команды `-on[line]`

```
gfix -o[nline] {single|multi|normal} <база данных>
```

Состояние `normal` означает, что к базе данных могут подключаться любые авторизованные пользователи. Этот режим используется по умолчанию при включении базы данных.

Использование пространства страниц базы данных

> Изменять режим заполнения страниц с данными можно только, если база находится в режиме `read_write`.

Когда пишется страница базы данных, Ред База Данных резервирует 20% страницы для будущего использования.

Для использования всего доступного пространства страницы базы данных вы можете использовать команду `-use full`. Если впоследствии вы захотите вернуться к режиму по умолчанию, введите команду `-use reserve`, чтобы использовать только 80% каждой страницы.

```
gfix -use {full|reserve} <база данных>
```

Проверка и исправление баз данных

При некорректном завершении работы сервера, аппаратном или программном сбое возможно появление различных ошибок в базе данных. Проверка базы данных с помощью утилиты `gfix` поможет найти такие фрагменты в базе данных и выбрать способ исправления той или иной ошибки. Проверку базы данных рекомендуется производить не только после аппаратных или программных сбоев в работе сервера, но и при возникновении любой ошибки при работе с базой данных, которая не связана с некорректно построенным запросом или проблемами с сетью, при ошибках резервного копирования/восстановления, а также с определенной периодичностью в профилактических целях.

Перед проверкой (или исправлением — см. далее) базы данных необходимо получить исключительный доступ к ней, как это описано в п. 8.4.

Проверка базы данных выполняется с помощью команды `-v[alidate]`:

```
gfix -v[alidate] <база данных>
```

По умолчанию при проверке базы данных `gfix` освобождает неиспользуемые страницы в базе данных[9], а также обнаруживает разрушенные структуры данных.

По умолчанию проверка работает на уровне страниц. Для проверки и на уровне страниц и на уровне записей используйте команду:

```
gfix -v[alidate] -full <база данных>
```

которая освобождает неиспользуемые фрагменты записей.

Для того, чтобы `gfix` производил только проверку базы данных без освобождения неиспользуемых страниц, необходимо указать опцию `-n[o_update]`:

```
gfix -v[alidate] -n[o_update] <база данных>
```

Для того, чтобы `gfix` не проверял ошибки контрольных сумм, используется опция `-i[gnore]`[10] команды `-v[alidate]`:

```
gfix -v[alidate] -i[gnore] <база данных>
```

Результаты работы утилиты `gfix` сохраняются в лог-файле сервера – `firebird.log`.

После проверки базы данных, если были найдены ошибки, можно попытаться исправить базу данных. Для этого существует опция `-mend`. Однако и она не в состоянии исправить все ошибки и может привести к потере данных (поврежденных записей).

```
gfix -mend <база данных>
```

Лучший способ избежать потери данных - регулярно делать резервное копирование и проверять копии на возможность восстановления. При попытке исправить поврежденную базу данных

[9] Страницы, которые были назначены какой-либо структуре данных, но не были использованы вследствие аппаратных или программных сбоев

[10] При чтении записи сервер перебирает не более 10 млн. версий записи, чтобы исключить возможностьбесконечного цикла.

всегда работайте с копией основного файла, а не с оригиналом. Использование опции `-mend` может привести к «бесшумному» удалению данных.

Изменение режима записи на диск

Многие ОС применяют кэширование операций ввода-вывода. При этом для буферизации используется некоторый объем быстрой памяти (который может быть как частью оперативной памяти сервера, так и специальной памятью, встроенной в сам диск). Это повышает производительность приложений, их интенсивность записи, но пользователь не будет уверен, когда его данные будут занесены на физический диск.

При работе с базой данных крайне желательно, чтобы данные были безопасно сохранены. В Ред Базе Данных можно указать должны ли данные сразу записываться на физический диск по `commit` или же доверить запись ОС.

```
gfix -write {sync|async} <база данных>
```

По умолчанию, все БД работают с включенным Forced Writes (`sync`) и отключать этот режим не рекомендуется. Если все же вы не удовлетворены производительностью БД, и при этом стопроцентно уверены в своем серверном оборудовании, можете попробовать отключить Forced Writes.

Активация режима репликации

Для активации режима репликации для базы данных используется команда:

```
gfix -replica {GUID} <имя базы данных>
```

Здесь указывается GUID мастер-базы. GUID это уникальный идентификатор БД, используемый для её опознавания в системе репликации. Узнать его можно с помощью утилиты `gstat -h`. Он генерируется при создании базы (либо при первом коннекте, если еще не задан). Он будет автоматически пересоздан при восстановлении базы из бэкапа и при выполнении команды `nbackup -f` (выход из режима блокировки базы без влития изменений из дельта-файла).Также его можно сменить с помощью команды `gfix -g`, которая должна выполняться в эксклюзивном режиме с правами администратора. Это может быть полезно при копировании БД средствами файловой системы.

Удаляется `GUID` для слейв-базы в момент отключения режима реплики.

Снятие режима реплики производится с помощью команды:

```
gfix -replica {} <имя базы данных>
```

8.5 Утилита GSTAT

Утилита `gstat` предназначена для получения полной информации о базе данных. `Gstat` даёт информацию о дате создания базы данных, размере страниц базы данных, количестве теневых копий, таблицах и другую.

Для того, чтобы работать с `gstat`, необходимо пройти процедуру идентификации и аутентификации — указать имя и пароль пользователя, который имеет право на запуск сервиса `gstat` (подробнее см. п. 9.3). Для указания имени и пароля пользователя используются переключатели `-user` и `-password`, соответственно, например:

```
gstat -u testuser -p pass -role rdb$admin [<опции>] <база данных>
```

Необязательный переключатель `-role` задаёт имя роли, права которой будут учитываться при выполнении каких-либо действий утилиты. Поэтому если пользователь не указывает роль, то

он получает права только тех ролей, которые ему назначены с DEFAULT.

Имя пользователя и пароль можно не указывать, если в системе установлены контекстные переменные ISC_USER и ISC_PASSWORD. А также, если используетсядоверительная аутентификация Windows (Win_Sspi) илидоверенная через механизм GSSAPI (gss), и пользователь системы, от имени которого запускается утилита ISQL, является членом группы администраторов.

Запуск GSTAT осуществляется одним из следующих способов:

```
gstat [<опции>] <база_данных>
gstat <база_данных> [<опции>]
```

Таблица 8.12 — Опции GSTAT

Переключатель	Описание переключателя
-a(ll)	Это значение по умолчанию, если вы не запросили -index, -data или -all. Отыскивает и отображает статистику по -index и -data.
-d(ata)	Статистика страниц данных – информация о таблицах содержащихся в базе данных. Пользовательские и системные индексы, системные таблицы не анализируются.
-e(ncryption)	Статистика по зашифрованным и незашифрованным страницам БД
-h(eader)	Статистика заголовочной страницы - это информация о глобальных свойствах всей базы данных, хранящаяся на заголовочной страницы каждой базы данных. Эта информация также выводится, если использовать любой другой переключатель gstat.
-i(ndex)	Статистика индексов – информация об индексах в базе данных. Пользовательские и системные таблицы, системные индексы не анализируются.
-s(ystem)	Эквивалент переключателям -a[ll] -s[ystem]. Это модификатор, который дополняет выводы -data или -index анализом системных таблиц (индексов).
-u(sername)	Имя пользователя
-p(assword)	Пароль пользователя
-fetch <имя файла>|stdin|/dev/tty	Файл, из которого будет считан пароль пользователя
-r(ecord)	Модификатор для переключателей -data и system. Он добавляет записи о средних длинах записей и версий для любых таблиц.
-t(able) <ТАБЛИЦА> [<ТАБЛИЦА>] ...	Анализ таблицы или списка таблиц и любых индексов, принадлежащих указанным таблицам. За этим параметром следует список имен таблиц, которые вы хотите проанализировать. Список должен быть в верхнем регистре, и каждая таблица разделяется пробелом.
-b(lob) TABLE.COLUMN [TABLE.COLUMN] ...	За переключателем -b следует список имен таблиц и столбцов, где хранятся ссылки на файловые блобы. Список должен быть в верхнем регистре, и каждая TABLE.COLUMN разделяется пробелом. Выводится информация для каждого элемента TABLE.COLUMN: общее количество ссылок, количество отсутствующих файлов (ссылка есть, файла нет), общий размер файлов по этим ссылкам, некорректные ссылки (ссылки, для которых не удалось установить путь к файлу). Если указано несколько таблиц — выводится также суммарная статистика по всем указанным таблицам.
-role	Имя роли

Переключатель	Описание переключателя
-tr	Использовать доверительную аутентификацию
-z	Показать версию сервера и утилиты

Статистика заголовочной страницы

Запустив утилиту gstat с ключом -header можно получить статистические данные о нашей базе данных. Часть информации является статичной, часть – меняется в зависимости от происходящих в базе данных изменений. Пример информации с заголовочной страницы приводится ниже:

```
Database "d:\RedDataBases\testdb.fdb"
Database header page information:
        Flags                0
        Generation           103
        System Change Number 3
        Page size            8192
        Server               RedDatabase
        ODS version          12.0
        Oldest transaction   91
        Oldest active        92
        Oldest snapshot      92
        Next transaction     92
        Autosweep gap        1
        Sequence number      0
        Next attachment ID   65
        Implementation       HW=AMD/Intel/x64 little-endian OS=WindowsCC=MSVC
        Shadow count         0
        Page buffers         0
        Next header page     0
        Database dialect     3
        Creation date        Sep 6, 2017 16:58:58
        Attributes           force write
  Variable header data:
      Database backup GUID: {49E55285-939F-4960-94A3-3681659D2341}
      Database GUID: {F87421F8-92F4-49C0-3D8F-3B88FDA010C1}
      *END*
```

Flags – это набор флагов, определяющий важные особенности поведения базы данных. Флаги устанавливаются только с помощью специальных инструментов вроде gfix, изменять флаги с помощью других инструментов опасно - это может привести к порче базы данных. Значения флагов можно посмотреть в таблице 8.13

Надо сказать, что при получении статистики показывается, что значение параметра Flags всегда равно нулю, вне зависимости от установленных флагов.

Generation – это счетчик, который увеличивается на единицу всякий раз, когда заголовочная страница записывается на диск.

`Page size` – размер страницы базы данных, исчисляется в байтах. Все файлы одной базы данных состоят из страниц одинакового размера, который устанавливается при создании базы данных и при восстановлении базы данных из резервной копии. Множество важнейших параметров сервера зависят от размера страницы - например, кэш базы данных. Рекомендуют создавать базу данных с размером страниц не менее 4096 байт, а лучше 8192 или 16384 байта.

`System Change Number` – маркер (число от 0 до 3), которым помечаются страницы базы данных при создании инкрементных копий с помощью утилиты `nbackup`. `Nbackup` выполняет резервное копирование страниц базы данных, копируя страницы, которые были изменены с момента последней резервной копии непосредственно предшествующего уровня. Если делается резервная копия уровня 0, копируются все страницы, а если уровня 1 – копируются только те страницы, которые были изменены после последнего уровня 0. Чтобы иметь возможность находить измененные страницы, Ред База Данных использует маркер, который называется SCN. Это число увеличивается при каждом изменении состояния резервной копии и не зависит от уровня резервного копирования.

- 0 – страницы перед резервным копированием;
- 1 – страницы, записанные в дельта-файл во время резервного копирования;
- 2 – страницы, записанные во время слияния дельта-файла с основной резервной копией;
- 3 – страницы, записанные после окончания первого бэкапа и слияния.

Резервное копирование и восстановление с помощью `gbak` не восстанавливает содержимое таблицы `RDB$BACKUP_HISTORY` и сбрасывает SCN всех страниц обратно на 0. Таким образом, восстановление с помощью `gbak` перепишет всю базу данных (и может даже изменить размер страницы). Это делает предыдущие резервные копии с помощью `nbackup` бессмысленными в качестве отправной точки для последующих резервных копий: нужно начать заново с уровня 0.

`Server` – чьим сервером была создана база данных: `RedDatabase` или `Firebird`.

`ODS version` – версия структуры базы данных на диске (On-Disk Structure). Представляет собой два числа, разделенные точкой. "Целая" часть - это основная версия ODS, которая зависит от версии сервера, создавшего данную базу данных. Главная версия определяет основные возможности работы с базой данных, и ее значение присваивается при создании (восстановлении) базы данных.
"Дробная" часть (после точки) - это минорная версия ODS, которая может меняться (точнее, увеличиваться) в течение жизни базы данных в зависимости от того, под управлением какой версии сервера работают с этой базой данных.

`Oldest transaction` – идентификатор старейшей заинтересованной транзакции в базе данных (`Oldest Interesting Transaction` или `OIT`). Первая транзакция с состоянием, отличным от `commit`. На практике это:

- `Oldest Active Transaction`;
- номер самой старой транзакции, завершенной "настоящим" `rollback` (выполняется если пользователь в одной транзакции модифицировал более 100000 записей);
- транзакция в состоянии `in-limbo` (полузавершенная) двухфазного `commit`.

Значение этого параметра часто сравнивается с `Next transaction`. Разница этих параметров показывает количество мусора в базе данных и можно судить о целесообразность выполнения резервного копирования.

`Oldest active` – старейшая транзакция, которую пользователь стартовал, и до сих пор не завершил по `commit` или `rollback` (`Oldest Active Transaction` или `OAT`).

`Oldest snapshot` – номер последней транзакции `snapshot`, которая влияет на процесс сборки мусора (`Oldest Snapshot Transaction` или `OST`). Дело в том, что только транзакции с уровнем изоляции `snapshot` вызывают появление мусорных версий записей в базе данных. `OST` – это старейший «номер snapshot» для всех активных транзакций. Транзакция *read-only, read committed* не имеет «номера snapshot». Транзакция *read-write, read committed* имеет «номер snapshot» равный своему собственному номеру. Для транзакции *snapshot* «номер snapshot» – это номер `Oldest Active`

`Transaction` на момент ее старта. Номер `Oldest Snapshot Transaction` обновляется, когда стартует новая транзакция (или выполняется `commit retaining`) или когда стартует `sweep`.

`Autosweep gap` – разница между `Oldest transaction` и `Oldest snapshot`. Показывает необходимость сборки мусора. Когда значение достигает `Sweep interval` (по умолчанию значение `Sweep interval` равно 20000) запускается автоматическая сборка мусора.Значение может быть отрицательным при долгоживущих транзакциях `snapshot`.

`Next transaction` – это просто номер, который будет присвоен следующей транзакции при ее старте.

Если вы заметили, что разница между `OIT` и `Next transaction` все увеличивается, значит какие-то транзакции не подтверждаются должным образом и следовательно может накапливаться все большее число мусорных записей. В конце концов вы увидите, что время запуска базы данных увеличивается, а производительность падает. В таком случае возможно следует запустить `gfix` для сборки мусора вручную.

`Sequence number` – порядковый номер файла базы данных. Для базы данных, состоящей из одного файла, он всегда равен нулю. Второй файл в базе данных будет иметь номер 1 и т д.

`Next attachment ID` – номер следующего соединения к этой базе данных. Каждый раз, когда приложение подключается к базе данных, это число увеличивается на один. Запуск и завершение работы базы данных также увеличивает этот номер. Запуск `gstat` не изменяет идентификатор.

`Implementation` – архитектура аппаратуры, на которой была создана база данных.

`Shadow count` – число файлов оперативных копий, которые определены для данной базы данных.

`Page buffers` – размер кэша базы данныхв страницах. Ноль означает, что база данных использует значение по умолчанию сервера (`DefaultDbCachePages` в `firebird.conf`).

`Next header page` – номер следующей заголовочной страницы. Всегда равен нулю. Собственно говоря, любая страница в базе данных имеет ссылку на номер следующей страницы такого же типа, но так как заголовочная страница всегда единственная в каждом файле базы данных, то у нее эта ссылка обнулена.

`Database dialect` – диалект SQL базы данных.

`Creation date` – дата создания базы данных или последнего восстановления из резервной копии.

`Attributes` – различные атрибуты базы данных. Очевидно, что значения этих атрибутов соответствуют флагам, хранящимся в параметре `Flags`.

Таблица 8.13 — Атрибуты и флаги файла базы данных

Атрибут	Значение флага		Расшифровка
`active shadow`	`0x1`	`1`	Файл является активным Shadow-файлом
`force write`	`0x2`	`2`	Режим принудительной записи включен (forced write on)
`crypt process, plugin <имя плагина>`	`0x4`	`4`	Идет процесс шифрования в фоновом режиме указанным плагином
`no reserve`	`0x8`	`8`	Указывает, что на страницах не резервируется место для старых версий данных. Это позволяет более плотно упаковывать данные на каждой странице, в силу чего база данных занимает меньше дискового пространства. Это идеал для баз данных только для чтения.

Атрибут	Значение флага	Расшифровка
read only	0x20 32	База данных работает в режиме Read Only. Если флаг не установлен, то допустимы как чтение, так и запись
encrypted, plugin <имя плагина>	0x40 64	База данных зашифрована указанным плагином
replica	0x100 256	База данных является объектом репликации (слейвом)
multi-user maintenance	0x80 128	База данных блокирована в режиме multi
single-user maintenance	0x1080 4224	База данных блокирована в режиме single
full shutdown	0x1000 4096	База данных блокирована в режиме full
backup lock	0x400 1024	Основной файл базы данных временно заблокирован. Изменения фиксируются во временном файле дельты.
backup merge	0x800 2048	Объединение файла дельты с основным файлом базы данных

Variable header data – переменные данные заголовочной страницы. Например, интервал очистки (sweep interval), GUID базы даных, GUID последней резервной копии и другое.

Анализ всей базы данных

Если не заданы никакие опции утилиты gstat по умолчанию выполняется анализ всей базы данных. Будет собрана информация о таблицах и индексах, содержащихся в базе данных. Поскольку вывод, скорее всего, будет очень большим, рекомендуется передать вывод в файл:

```
gstat d:\RedDataBases\testdb.fdb > d:\RedDataBases\testdb.txt
```

Статистика страниц данных

Следующая команда

```
gstat -data <база данных>
```

просматривает в базе данных таблицу за таблицей, отображая итоговую информацию о страницах данных. Для включения в отчет системных таблиц добавьте переключатель -system.

Вывод о каждой таблице будет примерно следующий:

```
TEST_TABLE (338)
    Primary pointer page: 734, Index root page: 735
    Pointer pages: 7, data page slots: 20448
    Data pages: 20448, average fill: 85%
    Primary pages: 13998, secondary pages: 6450, swept pages: 13998
    Empty pages: 0, full pages: 20446
    Fill distribution:
         0 - 19% = 0
        20 - 39% = 0
```

```
        40 - 59% = 0
        60 - 79% = 0
        80 - 99% = 0
```

Таблица 8.14 — Вывод `gstat -d[ata]`

Элемент	Описание
`Primary pointer page`	Номер первой страницы косвенных указателей на страницы, хранящие данные таблицы
`Index root page`	Номер страницы, которая является первой страницей указателей на индексы таблицы
`Pointer pages`	Общее количество страниц косвенных указателей на страницы, хранящие данные таблицы
`data page slots`	Количество указателей на страницы базы данных, содержащихся на страницах указателей. Должно равняться числу страниц данных
`Data pages`	Общее количество страниц, в которых хранятся данные таблицы. Этот счетчик включает страницы, хранящие неподтвержденные версии записей и мусор, потому что `gstat` не может их отличить друг от друга
`Average fill`	Обобщающая гистограмма распределения использования памяти для всех страниц, выделенных в таблице.
`Primary pages`	Количество страниц, равное (`Data pages - Secondary pages`)
`Secondary pages`	Количество страниц, на которых не хранятся первичные версии записей.
`Swept pages`	Количество страниц, которые имеют только первичные версии записей, и все они были созданы подтвержденными транзакциями. Такие страницы данных должны быть пропущены процедурой `sweep`.
`Empty pages`	Количество страниц, на которых нет записей
`Full pages`	Количество полностью заполненных страниц
`Fill distribution`	Это гистограмма из пяти 20-процентных "полос" , каждая из которых показывает количество страниц данных, чье среднее заполнение попадает в этот диапазон. Процент заполнения определяется соотношением пространства каждой страницы, содержащей данные. Сумма этих чисел дает общее количество страниц, содержащих данные

Анализ индексов

Следующая команда

```
gstat -i[ndex] <база данных>
```

отыскивает и отображает статистику по индексам в базе данных: средняя длина ключа (в байтах), общее количество дубликатов и максимальное количество дубликатов одного ключа и другое. Вы можете добавить переключатель `-system`, чтобы включить сведения о системных индексах в отчет.

К сожалению, не существует способа получить статистику по одному индексу. Однако вы можете ограничить результат одной таблицей, используя переключатель `-t`, за которым следует имя таблицы. Вы можете записать разделенный пробелам и список имен таблиц для получения результатов более чем по одной таблице.

Данные индексной страницы отображаются так:

```
TEST_TABLE (128)
     Index IND1 (0)
        Root page: 756595, depth: 4, leaf buckets: 232528, nodes: 50422773
        Average node length: 11.90, total dup: 0, max dup: 0
        Average key length: 8.11, compression ratio: 1.11
        Average prefix length: 3.89, average data length: 5.11
        Clustering factor: 6485823, ratio: 0.13
        Fill distribution:
             0 - 19% = 1
            20 - 39% = 0
            40 - 59% = 0
            60 - 79% = 0
            80 - 99% = 232527
```

Таблица 8.15 — Вывод `gstat -i[ndex]`

Элемент	Описание
`Root page`	Номер корневой страницы индекса
`Depth`	Количество уровней в странице индексного дерева. Если глубина дерева индексной страницы превышает 3, то доступ к записям через индекс не будет максимально эффективным. Для уменьшения глубины дерева индексной страницы увеличьте размер страницы. Если увеличение размера страницы не уменьшает глубины, снова увеличьте размер страницы.
`Leaf buckets`	Количество страниц самого низкого уровня (листовых) в дереве индекса. Это страницы, которые содержат указатели на записи. Страницы высокого уровня содержат косвенные связи.
`Nodes`	Общее количество записей, индексированных в дереве. Должно быть равно количеству индексированных строк в дереве, хотя отчет `gstat` может включать узлы, которые были удалены, но не вычищены в процессе сборки мусора. Может также включать множество элементов для записей, у которых был изменен индексный ключ.
`Average node length`	Средний размер узлов в байтах
`Total dup`	Общее количество строк дубликатов индекса
`Max dup`	Количество дублирующих узлов в "цепочке" , имеющих наибольшее количество дубликатов. Всегда будет нулем для уникальных индексов. Если число велико по сравнению с числом в `Total dup`, то это признак плохой селективности.
`Average key length`	Средний размер ключа в байтах с учетом сжатия. К длине каждого ключа прибавляется от 1 до 5 байт в зависимости от размера ключа и префикса. Затем высчитывается средний размер ключа.
`Compression ratio`	Отношение средней длины ключа без учета сжатия (`Average prefix length + Average data length`) к средней длине ключа с учетом сжатия (`Average key length`).
`Average prefix length`	Средний размер, занимаемый префиксами узлов, в байтах.
`Average data length`	Средняя длина каждого ключа в байтах. Она, скорее всего, меньше, чем фактическая сумма размеров столбцов, поскольку Ред База Данных использует индексное сжатие для уменьшения объема данных, хранящихся на странице листа индекса.

Элемент	Описание
`Clustering factor`	Это мера того, насколько много операций ввода-вывода будет осуществлять база данных, если бы ей пришлось читать каждую строку таблицы посредством индекса, в порядке индекса. То есть она показывает, насколько упорядочены строки в таблице по значениям индекса. Если значение близко к общему количеству страниц, значит таблица очень хорошо упорядочена. В этом случае записи индекса на одной странице листа индекса обычно указывают на строки, находящиеся в одних и тех же страницах данных. Если значение близко к общему количеству строк, значит, таблица весьма неупорядочена. В этом случае маловероятно, что записи индекса на одной странице листа индекса указывают на те же страницы данных.
`Ratio`	Отношение `Clustering factor` к общему количеству узлов в индексе.
`Fill distribution`	Это гистограмма с пятью 20-процентными полосами, каждая из которых показывает количество индексных страниц, чей средний процент заполнения находится в указанном диапазоне. Процент заполнения определяется соотношением пространства каждой страницы, содержащей данные. Сумма таких чисел дает общее количество страниц, содержащих индексные данные

Статистика по размерам и версиям записей

Следующая команда:

```
gstat -r[ecord] [-d] <база данных>
```

отображает статистику по размерам и версиям записей.

Таблица 8.16 — Вывод `gstat -r[ecord]`

Элемент	Описание
`Total formats`	Общее количество форматов в таблице `RDB$FORMATS`
`Used formats`	Количество используемых форматов
`Average record length`	Средний размер сжатой записи в байтах
`Total records`	Общее количество строк в таблице.
`Average version length`	Среднее значение длины старых версий в байтах
`Total versions`	Общее количество старых версий в таблице
`Max versions`	Максимальная цепочка старых версий для записи
`Average fragment length`	Средний размер фрагмента в байтах
`Total fragments`	Количество всех фрагментов во всех записях
`Max fragments`	Максимальное количество фрагментов в одной записи
`Average unpacked length`	Средний размер записи в байтах (без сжатия)
`Compression ratio`	Отношение несжатых данных к сжатым
`Big record pages`	Количество страниц, которые полностью заняты только одной записью
`Blobs`	Количество всех блобов (0, 1 и 2 уровня)

Элемент	Описание
Total length	Размер, занимаемый блобами, в байтах
Blob pages	Количество страниц с блобами
Level <n>	Количество блобов каждого уровня

Статистика по файловым блобам

В утилите gstat можно воспользоваться опцией, в которой указываются таблицы, где хранятся ссылки на файловые блобы и которая будет подсчитывать для каждой указанной таблицы количество и общий размер файлов по ссылкам, количество ненайденных файлов (ссылка есть, файла нет), некорректные ссылки (не удалось установить путь к файлу). Null, пустые строки, а также строки, состоящие только из пробелов, ссылками не считаются и не увеличивают счётчик ссылок.

```
gstat <база данных> -b TABLE.COLUMN [TABLE.COLUMN] ...
```

Если указано несколько столбцов одной таблицы, то при выводе произойдёт группировка по таблице. Если передано больше одного поля, то выведется суммарная статистика.

Файловые блобы хранятся в каталогах, настраиваемых в файле directories.conf. Они управляются тремя системными функциями: CREATE_FILE, READ_FILE, DELETE_FILE.

CREATE_FILE создаёт файл из указанного блоба и возвращает ссылку на него. Эта ссылка может быть сохранена в таблице в обычном текстовом поле. READ_FILE может по этой ссылке прочитать файл и вернуть его содержимое в виде блоба.

Опция -b при вызове gstat указывается после пути к базе данных.

-b можно совмещать с остальными параметрами, кроме -e и -b. Также, запрещается использование с -i без -d и -t.

```
gstat d:/work/test.fdb -b BAT.VCHAR BAT.BL LINKS.B1 LINKS.B2 LINKS.B3
================================================================================
Database file sequence:
File D:\WORK\TEST.FDB is the only file

Analyzing file blobs ...
BAT (130)
    'VCHAR' field:
          Links'count: 2
          Missing files'count: 0
          Unresolved links'count: 0
          Blob files'size: 26 bytes

    'BL' field:
          Links'count: 2
          Missing files'count: 0
          Unresolved links'count: 0
          Blob files'size: 22 bytes

LINKS (128)
    'B1' field:
          Links'count: 2
          Missing files'count: 0
          Unresolved links'count: 1
          Blob files'size: 7 bytes

    'B2' field:
          Links'count: 2
```

```
          Missing files'count: 1
          Unresolved links'count: 1
          Blob files'size: 0 bytes

    'B3' field:
          Links'count: 2
          Missing files'count: 0
          Unresolved links'count: 1
          Blob files'size: 13 bytes

Total links'count: 10
Total missing files'count: 1
Total unresolved links'count: 3
Total blob files'size: 68 bytes
```

8.6 Утилита GSEC

GSEC — это утилита для работы с базой данных безопасности (содержащей информацию о пользователях СУБД). Она позволяет привилегированному пользователю управлять учетными записями пользователей для различных баз данных. Используя различные опции, можно добавлять, изменять или удалять учетные записи пользователей из базы данных безопасности.

> Утилита GSEC устарела. Вместо неё лучше использовать SQl команды для управления пользователями (подробнее см. Работа с пользователями 9.3) или Services API.

Информация о всех пользователях хранится в обычной базе данных, называемой базой данных безопасности. По умолчанию база данных безопасности располагается в директории «Ред Базы Данных» и называется security3.fdb (см. Приложение Д).

GSEC может быть запущена как в интерактивном, так и в командном режиме, и может отображать подсказки с перечислением всех опций.

Синтаксис GSEC:

```
gsec [ <опции> ... ] <команда> [ <параметры> ... ]
```

Таблица 8.17 — Опции GSEC

Опция	Описание опции
-user <имя пользователя>	Имя пользователя
-password <пароль>	Пароль пользователя
-fetch_password <файл>	Файл, из которого будет считан пароль пользователя
-role <имя роли>	Название роли, чьи права будет использовать указанный пользователь
-trusted	Использовать доверительную аутентификацию
-database <БД_безопасности>	Путь к файлу базы данных безопасности
-z	Отобразить версию GSEC и сервера «Ред Базы Данных»

Таблица 8.18 — Команды GSEC[11]

Команда	Описание команды
add <имя> [<параметр> ...]	Добавление нового пользователя

Команда	Описание команды
`delete <имя>`	Удаление пользователя
`display`	Вывод информации обо всех зарегистрированных пользователях
`display <имя>`	Вывод информации о конкретном пользователе
`modify <имя> <параметр> [ <параметр> ... ]`	Изменение информации о пользователе
`mapping {set | drop}`	Включает или выключает флаг `AUTO ADMIN MAPPING` для автоматического предоставления роли `RDB$ADMIN` администраторам Windows в текущей базе данных, если используется доверительная авторизация
`{? | help}`	Отображает все опции и команды `GSEC`
`z`	Отобразить версию `GSEC` и сервера «Ред Базы Данных»
`quit`	Выход из интерактивного режима

Таблица 8.19 — Параметры `GSEC`

Параметр	Описание параметра
`-pw <пароль>`	Пароль пользователя
`-uid <uid>`	Идентификатор пользователя
`-gid <uid>`	Идентификатор группы
`-fname <имя>`	Полное имя пользователя
`-mname <отчество>`	Отчество
`-lname <фамилия>`	Фамилия
`-admin {yes | no}`	Установка флага, имеет ли пользователь роль `RDB$ADMIN`

Для того, чтобы выполнить любую операцию с помощью `gsec`, необходимо пройти процедуру идентификации и аутентификации — указать имя и пароль пользователя, который имеет право на запуск сервиса `gsec` (подробнее см. п. 9.3). Для указания имени и пароля пользователя используются переключатели `-user` и `-pa[ssword]`, соответственно, например:

```
gsec -user testuser -password pass -role rdb$admin <команда> [<параметры>...]
```

Необязательный переключатель `-role` задаёт имя роли, права которой будут учитываться при выполнении каких-либо действий утилиты. Поэтому если пользователь не указывает роль, то он получает права только тех ролей, которые ему назначены с `DEFAULT`. Права на внесение изменений в базу данных безопасности имеют пользователи с административными привилегиями. При этом роль `RDB$ADMIN` должна быть указана в переключателе `-role`. Остальные пользователи имеют права только на изменение своей учетной записи.

Имя пользователя и пароль можно не указывать, если в системе установлены контекстные переменные `ISC_USER` и `ISC_PASSWORD`. А также если используетсядоверительная аутентификация Windows (`Win_Sspi`) илидоверенная через механизм GSSAPI (`gss`), и пользователь системы, от имени которого запускается утилита ISQL, является членом группы администраторов.

Пример запуска утилиты `GSEC` в интерактивном режиме:

```
gsec -user sysdba -password masterkey
```

Пример вывода информации о пользователях:

```
GSEC> display
user name uid gid full name
------------------------------------------------------------
SYSDBA 0 0 Sql Server Administrator
TEST_USER 0 0 John Smith
```

GSEC можно использовать для управления базой данных безопасности на удаленном сервере. Для этого необходимо указать в командной строке имя:

```
gsec -database 192.168.10.1:/opt/RedDatabase/security3.fdb -user sysdba -password
masterkey
```

Для выхода из интерактивного режима утилиты GSEC используется команда q[uit]:

```
GSEC> quit
```

8.7 Утилита rdb_lock_print

Утилита rdb_lock_print является инструментом, формирующим статистические данные файла блокировок, что помогает при решении сложных проблем взаимных блокировок.

Исполняемый модуль rdb_lock_print находится в каталоге bin установки СУБД «Ред База Данных». Синтаксис вызова утилиты может выглядеть одним из следующих способов:

```
rdb_lock_print -d <database_name> [<опции>...]
rdb_lock_print -f <lock_file> [<опции>...]
```

Где параметр <database_name> задает имя и путь к БД. Параметр lock_file задает имя и путь к lock - файлу. Путь к файлу таблицы блокировок проходит через /tmp/firebird в ОС Linuxи C:/ProgramData/firebird в ОС Windows.

С помощью утилиты rdb_lock_print может быть выведена таблица блокировок в удобном пользователю формате. Отчет вывода разбит на блоки в следующей последовательности:

1. LOCK_HEADER BLOCK: заголовок блока;
2. OWNER BLOCK: группы владельцев;
3. REQUEST BLOCK: группы запросов. Каждый владелец выводится с его запросами;
4. LOCK BLOCK: группы блокировок;
5. History: история событий.

Утилита rdb_lock_print имеет несколько опций, приведенных в таблице 8.20. Если никаких опций не задано, то выводится заголовок блока LOCK_HEADER BLOCK, в котором описано основная конфигурация и состояние таблицы блокировок.

Таблица 8.20 — Опции rdb_lock_print

Опция	Описание
-a	Выводит содержимое таблицы блокировок: заголовок блока (LOCK_HEADER BLOCK), группы блоков (LOCK BLOCK), группы владельцев (OWNER BLOCK), группы запросов (REQUEST BLOCK) и историю событий (History).
-c	Запрашивает мьютекс (mutex) менеджера блокировок распечатать цельное последовательное представление таблицы блокировок. При этом останавливаются все процессы доступа к базе данных.
-h	Выводит недавнюю историю событий

Опция	Описание
`-i [a,o,t,w] [<N> [<M>]]`	Выдает интерактивный отчет, отслеживающий текущую деятельность таблицы блокировок. Выводит выбранные параметры менеджера блокировок в течение n секунд с интервалом m секунд. Значения по умолчанию 1 сек для обоих значений. Если указано только `-i` — выводится вся информация.
`-l`	Выводит группы блоков
`-m`	Вывод в HTML формате
`-n`	Выводит только ожидающих завершения владельцев (если указано `-o`) или ожидающих завершения групп блоков (если указано `-l`)
`-o | -p`	Выводит группы владельцев
`-q <N>`	Находит владельцев взаимных блокировок и печатает их PID (`N` – интервал сканирования)
`-r`	Выводит информацию о группе блока запросов
`-s <номер>`	Выводит группы блокировок ресурсов заданной серии (действует только если указан спецификатор `-l`)
`-w`	Выводит временный список блокировок (групп запросов), блокирующих другие запросы на блокировку, для каждого владельца (действует только с указанием `-o`)

Результаты использования утилиты могут оказаться достаточно большими, поэтому удобнее отчет выводить в файл, указав: `> <путь к файлу>`.

Блок LOCK_HEADER

Первая группа всех отчетов `rdb_lock_print`. Каждый отчет выводит только одну группу заголовка. На рисунке 8.1 представлено, как может выглядеть блок `LOCK_HEADER`.

```
LOCK_HEADER BLOCK
    Version: 145, Active owner:      0, Length: 1048576, Used:  23128
    Flags: 0x0001
    Enqs:     18, Converts:     40, Rejects:      0, Blocks:      0
    Deadlock scans:      0, Deadlocks:      0, Scan interval:  10
    Acquires:    109, Acquire blocks:      0, Spin count:   0
    Mutex wait: 0.0%
    Hash slots: 1009, Hash lengths (min/avg/max):    0/   0/   3
    Remove node:      0, Insert queue:      0, Insert prior:      0
    Owners (1): forward:  20840, backward:  20840
    Free owners: *empty*
    Free locks (2): forward:  22808, backward:  23064
    Free requests (2):  forward:  22744, backward:  23000
    Lock Ordering: Enabled
```

Рисунок 8.1 — Пример группы заголовка блока

`Version:` Номер версии менеджера блокировок.

`Active owner:` Владелец, который управляет таблицей блокировок в настоящий момент. Если в таблицу блокировок не пишет ни один процесс, то активным владельцем будет 0.

`Length:` Общий объем памяти, выделенный таблице блокировок (в байтах).

`Used:` Наибольшая величина смещения в таблице блокировок, которая используется в настоящий момент.

`Flags:` Определены два битовых флага:

- `LHB_shut_manager` — показывает, что БД остановлена;
- `LHB_lock_ordering` — блокировки назначаются в порядке запросов FIFO.

`Enqs:` Число запросов, полученных на блокировку (не включает запросы, которые пришли и ушли).

`Converts:` Запросы на повышение уровня блокировки.

`Rejects:` Запросы, которые не могут быть удовлетворены.

`Blocks:` Запросы, которые не могут быть удовлетворены немедленно.

`Deadlock scans:` Показывает число просмотров менеджером блокировок цепочки блокировок и владельцев для поиска взаимных блокировок. Менеджер приступает к просмотру, когда процесс ожидает блокировки в течение `Scan interval` секунд.

`Deadlocks:` Число найденных взаимных блокировок.

`Scan interval:` Время (в секундах), которое ожидает менеджер блокировок до того как запустить к поиску взаимных блокировок.

`Acquires:` Сколько раз владелец запрашивает исключительное управление таблицей блокировок, чтобы выполнить изменения.

`Acquire blocks:` Сколько раз владелец находился в состоянии ожидания при запросе исключительного управления таблицей блокировок.

`Spin count:` Режим ожидания взаимной блокировки, когда повторяется запрос к таблице блокировок. По умолчанию отключено(равно 0).

`Mutex wait:` Процент попыток, которые были заблокированы, когда владелец старался обратиться к таблице блокировок Сколько раз (в процентах) владелец находился в состоянии ожидания, когда пытался обратиться к таблице блокировок.

`Hash slots:` Число слотов кэширования блокировок.

`Hash lengths:` Длина цепочки кэширования.

`Remove node:` Владелец сообщает о намерении удалить узел из таблицы, когда зависает при запросе к таблице блокировок.

`Insert queue:` Владелец сообщает о намерении добавить узел в таблицу, когда зависает при запросе к таблице блокировок.

`Insert prior:` Указывает, где размещается ошибочное добавление узла.

`Owners:` Количество владельцев, соединенных с таблицей блокировок.

`Free owners:` Количество блоков владельцев, выделенных владельцам, которые завершили их соединения, оставив блоки неиспользованными.

`Free locks:` Количество групп блокировок, которые были освобождены и не использованы повторно.

`Free requests:` Количество групп запросов, которые были освобождены и не использованы повторно.

`Lock ordering:` Определяет порядок получение запросов на блокировку (в порядке их поступления). Включено, если флаг `LHB_lock_ordering` установлен.

Блок OWNER

Идентифицирует конкретного владельца. Владельцы делятся на несколько типов, которым сопоставляются числа: 1 – процесс, 2 – база данных, 3 – клиентское соединение, 4 – транзакция, 255 – фиктивный процесс.

Заголовок блока содержит число, которое используется в качестве идентификатора владельца в таблице блокировок.

```
OWNER BLOCK  20840
    Owner id: 26834955665412, type: 1, pending:      0
    Process id:   6248 (Alive), thread id:   8928
    Flags: 0x00
    Requests (15):  forward:  20952, backward:  22872
    Blocks: *empty*
```

Рисунок 8.2 — Пример группы владельцев

`Owner id:` Идентификатор владельца.

`Type:` Тип владельца.

`Pending:` Ожидание завершения блокировки, запрошенную владельцем, но не полученную. Показывает смещение группы запроса блокировки.

`Process id:` Идентификатор процесса.

`Thread id:` Идентификатор потока.

`Flags:` Биты, которые определяют состояние. Процесс в одно и то же время может находиться более чем в одном состоянии.

`Requests:` Запросы на блокировку (обработанные или ожидающие завершения) связанные с этим процессом.

`Forward:` Ссылается на последующий элемент в очереди запросов, принадлежащих этому процессу. Число задает смещение.

`Backward:` Ссылается на предыдущий элемент в очереди запросов, принадлежащих этому процессу. Число задает смещение.

`Blocks:` Временный список блокировок (групп запросов), блокирующих другие запросы на блокировку, которыми владеет этот процесс.

Блок REQUEST

Выводит все запросы владельца. На рисунке 8.3 показано как выглядит блок `REQUEST BLOCK`.

```
REQUEST BLOCK  23000
    Owner:  20840, Lock:  23064, State: 6, Mode: 6, Flags: 0x00
    AST: 0x0000000000000000, argument: 0x000000000483A2F0
    lrq_own_requests: *empty*
    lrq_lbl_requests:   forward:     24, backward:  22744
    lrq_own_blocks  : *empty*
```

Рисунок 8.3 — Пример блока конкретного запроса

`Owner:` Смещение группы владельца в таблице блокировок, выполнившего запрос.

`Lock:` Смещение группы блокировки, которая описывает блокируемый ресурс.

`State:` Состояние блокировки, которое назначено этому ресурсу.

`Mode:` Режим блокировки – состояние, которое было запрошено для блокировки.

`Flags:` Флаг запроса содержит биты: 1 – блокирование, 2 – ожидание завершения, 4 – конвертирование, 8 – отмена. Они могут комбинироваться.

`AST:` Адрес подпрограммы, которая вызывается, когда кто-либо еще хочет получить блокировку на ресурс, используемый настоящим запросом.

`Argument:` Адрес аргумента, который может понадобиться подпрограмме AST.

Блок LOCK

Группы блокировок представляют блокируемые ресурсы различных типов, соответствующих определенным сериям:

Таблица 8.21 — Типы ресурсов

Серия	Символ	Тип ресурса
1	LCK_database	Сама база данных
2	LCK_relation	Отношение
3	LCK_bdb	Страница базы данных
4	LCK_tra	Транзакция
5	LCK_rel_exist	Существование отношения
6	LCK_idx_exist	Существование индекса
7	LCK_attachment	Подключение
8	LCK_shadow	Теневая копия
9	LCK_sweep	Сборка мусора
10	LCK_retaining	Самая молодая транзакция, подтвержденная с сохранением контекста
11	LCK_expression	Механизм кэширования индекса по выражению
12	LCK_prc_exist	Существование процедуры
13	LCK_update_shadow	Синхронное изменение теневой копии
14	LCK_backup_alloc	Страница размещения таблицы в резервном файле
15	LCK_backup_database	Защита записи в файл БД
16	LCK_backup_end	Защита согласованного удаления резервного файла
17	LCK_rel_partners	Связанные таблицы
18	LCK_page_space	Идентификатор страничного пространства
19	LCK_dsql_cache	DSQL кэш
20	LCK_monitor_state	Сбрасывание данных мониторинга
21	LCK_tt_exist	Существование TextType
22	LCK_cancel	Отмена операции
23	LCK_btr_dont_gc	Предотвращение удаления страницы B-дерева из индекса
24	LCK_shared_counter	Общий счетчик всей БД
25	LCK_rel_gc	Разрешение сборки мусора для отношения

Серия	Символ	Тип ресурса
26	LCK_fun_exist	Существование функции
27	LCK_rel_rescan	Принудительное повторное сканирование отношения
28	LCK_crypt	Однопоточное шифрование
29	LCK_crypt_status	Сообщает об изменении статуса шифрования базы данных
30	LCK_record_gc	Сборка мусора на уровне записи
31	LCK_alter_database	ALTER DATABASE
32	LCK_repl_txn	Транзакция репликации

На рисунке 8.4 представлен вывод группы блокировки ресурса серии 8. Блок LOCK_BLOCK идентифицирует группу описания заблокированного ресурса.

```
LOCK BLOCK  22552
    Series: 20, Parent:  21016, State: 2, size: 8 length: 4 data: 0
    Key: 000000, Flags: 0x00, Pending request count:      0
    Hash que (3):   forward:    380, backward:  22424
    Requests (1):   forward:  22488, backward:  22488
        Request  22488, Owner:  20840, State: 2 (2), Flags: 0x00
```

Рисунок 8.4 — Пример группы блокировок

Series: Тип ресурса.

Parent: Родитель для всех блокировок, связанных с конкретным типом ресурса.

State: Наивысшее текущее состояние блокировки.

Size: Длина части группы блокировки, содержащей ключ (в байтах).

Length: Фактическая длина ключа.

Data: Данные. Являются целым числом.

Key: Идентификатор заблокированного ресурса.

Hash queue: Начало и конец очереди хэш для ключей ресурсов.

Requests: Показывает число запросов на блокировку этого ресурса и указатели вперед и назад на группы блокировок.

Request: Список запросов. Показывает идентификатор запрашиваемой группы, процесс, выполняющий запрос и фактическое состояние блокировки с запрашиваемым (в круглых скобках).

Flags: Флаг запроса содержит биты: 1 – блокирование, 2 – ожидание завершения, 4 – конвертирование, 8 – отмена. Они могут комбинироваться.

Блок History

Менеджер блокировок запоминает действия, которые он выполнял для каждого владельца. Эти действия можно просмотреть в блоке History и блоке Event log.

```
History:
    ENQ:    owner =  20840, lock =      0, request =  20952
    GRANT:  owner =  20840, lock =  21016, request =  20952
    ENQ:    owner =  20840, lock =      0, request =  21088
    GRANT:  owner =  20840, lock =  21152, request =  21088
    ENQ:    owner =  20840, lock =      0, request =  21208
    GRANT:  owner =  20840, lock =  21272, request =  21208
    CONVERT:    owner =  20840, lock =  21272, request =  21208
    GRANT:  owner =  20840, lock =  21272, request =  21208
    CONVERT:    owner =  20840, lock =  21272, request =  21208
    GRANT:  owner =  20840, lock =  21272, request =  21208
```

Рисунок 8.5 — Пример вывода записей истории

`GRANT:` Предоставление блокировки на ресурс.

`ENQ:` Помещение в очередь запрос .

`DENY:` Отказ в запросе на ресурс.

`POST:` Отправка сообщения владельцу по поводу ресурса.

`DEQ:` Снятие блокировки владельцем.

`SCAN:` Сканирование взаимных блокировок.

`WAIT:` Владелец в состоянии ожидания.

`CONVERT:` Повышение уровня блокировки.

8.8 Утилита rdbsvcmgr

Утилита предоставляет интерфейс командной строки для `Services API`, обеспечивая доступ к любой службе, которая реализуется в СУБД.

`Services API` - программный интерфейс для запуска служебных задач (например, бэкап, рестор, проверка базы данных, сбор статистики и т.д.) без использования утилит. СУБД выполняет эти задачи с помощью менеджера сервисов (`service manager`). Утилита `rdbsvcmgr` обеспечивает доступ к сервисам.

Синтаксис вызова утилиты:

```
rdbsvcmgr <имя сервиса> <опции>
```

Имя сервиса должно быть `service_mgr`, может иметь префикс с именем хоста в соответствии с общими правилами (`host:service_mgr, \\\host\service_mgr`).

Опции в точности соответствуют тегам `SPB`, используемым в сокращенном виде. Удалите части тега `isc_`, `spb_`, `svc_` и вы получите опцию. Например, `isc_action_svc_backup` нужно указать как `action_backup`, `isc_spb_dbname => dbname`, `isc_info_svc_implementation => info_implementation`, `isc_spb_prp_db_online => prp_db_online` и так далее.

В одном вызове `rdbsvcmgr` можно указать либо одно действие, либо несколько информационных элементов.

8.8.1 Подключение к менеджеру сервисов

Чтобы подключиться к удалённому менеджеру сервисов нужно указать имя локальной или удаленной службы. Эта строка похожа на строки подключения к базе данных, поскольку синтаксис определяет сетевой протокол, используемый для подключения клиентского приложения к менеджеру сервисов на хосте сервера.

Таблица 8.22 — Подключение к менеджеру сервисов

Опция	Описание
`user | user_name <имя пользователя>`	Имя пользователя.
`role | sql_role_name <роль>`	Роль.
`password <пароль>`	Пароль.
`fetch_password <файл>`	Считать пароль из файла.
`trusted_auth`	Использовать доверительную аутентификацию.
`expected_db <база данных>`	база данных безопасности, если она отличается от базы по умолчанию (security3.fdb).

Запуск `rdbrepldiff` через сервис:

```
./rdbsvcmgr localhost:service_mgr -user SYSDBA -password masterkey -action_diff
-dbname localhost:/my/other/database.fdb -dif_simple -dif_replica
john:smith@/my/replica/database2.fdb
```

8.8.2 Информационные запросы

Вы можете использовать следующие опции для получения информации о конфигурации сервера.

Таблица 8.23 — Информационные запросы

Опция	Описание
`info_server_version`	Версия сервера.
`info_implementation`	Аппаратная платформа, на которой была создана база данных.
`info_user_dbpath`	Путь к базе данных безопасности сервера.
`info_get_env`	Расположение корневого каталога сервера.
`info_get_env_lock`	Расположение файла таблицы блокировок на сервере.
`info_get_env_msg`	Расположение файла сообщений на сервере.
`info_svr_db_info`	Количество подключений к базе данных и баз данных, активных в данный момент на сервере.
`info_version`	Версия `Service Manager`.
`info_capabilities`	Битовая маска возможностей, поддерживаемых сервером.

Отображение баз данных, используемых на сервере, а также версии СУБД

```
./rdbsvcmgr yourserver:service_mgr user sysdba password masterkey
info_server_version info_svr_db_info
```

8.8.3 Действия

В одном вызове менеджера сервисов можно выполнять только одну задачу. Вы можете поддерживать несколько подключений к менеджеру сервисов и выполнять задачу в каждом подключении.

Бэкап

Действие `action_backup` позволяет выполнять резервное копирование базы данных. Аналогично `gbak -b`.

Таблица 8.24 — Опции action_backup

Опция	Описание
`dbname <имя пользователя>`	Путь к основному файлу базы данных.
`verbose`	Подробный вывод о бэкапе.
`bkp_file <файл резервной копии>`	Путь к файлу резервной копии.
`bkp_length <n>`	Длина файла резервной копии в байтах.
`bkp_factor <n>`	Использовать блокирующий фактор n.
`bkp_parallel_workers <n>`	Количество параллельных рабочих потоков для бэкапа. Включает многопоточное выполнение резервного копирования.
`bkp_ignore_checksums`	Игнорировать контрольные суммы, а также поврежденные `BLOB` при бэкапе и `BLR` при ресторе.
`bkp_ignore_limbo`	Игнорировать зависшие двухфазные транзакции (`limbo`).
`bkp_metadata_only`	Производит резервное копирование только метаданных.
`bkp_no_garbage_collect`	Не собирать мусор во время резервного копирования.
`bkp_old_descriptions`	Производит резервное копирование метаданных в формате старого стиля, т.е. в режиме совместимости со старыми базами данных.
`bkp_non_transportable`	Создаёт резервную копию в нетранспортабельном формате.
`bkp_convert`	Преобразование внешних файлов во внутренние таблицы.
`bkp_no_triggers`	Не запускать триггеры уровня базы данных.
`verbint <n>`	Подробный вывод лога действии с заданным интервалом обработанных записей.
`bkp_skip_data <шаблон>`	Пропускать данные таблиц, имена которых соответствуют указанному здесь регулярному выражению в SQL-синтаксисе (см. оператор `SIMILAR TO`).
`bkp_keep_data <шаблон>]`	Включать только данные таблиц, имена которых соответствуют указанному здесь регулярному выражению в SQL-синтаксисе (см. оператор `SIMILAR TO`).
`bkp_stat TDRW`	Вывести статистику в процессе работы (работает вместе с `-verbose`)). T — время от начала работы `gbak`; D — время прошедшее с последнего вывода на экран; R — число страниц прочитанных с последнего вывода на экран; W — число страниц записанных с последнего вывода на экран.

Бэкап без сборки мусора с перенаправлением подробного вывода о процессе в файл:

```
isql -user sysdba -pas masterkey -g -b -se localhost:service_mgr -v -y
/backup/emp_back.log employee /backup/employee.fbk
```

Рестор

Действие `action_restore` позволяет выполнить восстановление базы данных из резервной копии. Аналогично `gbak -c`.

Таблица 8.25 — Опции action_restore

Опция	Описание
`bkp_file <файл резервной копии>`	Путь к файлу резервной копии.
`dbname <база данных>`	Путь к файлу базы данных.
`res_length <n>`	Количество страниц в восстановленном файле базы данных.
`verbose`	Подробный вывод о ресторе.
`res_buffers <n>`	Задать размер страничного кэша для БД.
`res_page_size <n>`	Установить размер страницы.
`res_access_mode [prp_am_readonly | prp_am_readwrite]`	Режим доступа `"read_only"` или `"read_write"`.
`res_deactivate_idx`	Деактивировать индексы во время восстановления.
`res_no_shadow`	Восстановить без создания теневых копий.
`res_no_validity`	Отключение всех ограничений проверки данных (`check`) и ограничений `NOT NULL`.
`res_one_at_a_time`	Подтверждать транзакцию после восстановления каждой таблицы.
`res_replace`	Заменить базу данных, если она существует.
`res_create`	Восстановить, но не перезаписывать существующую базу данных.
`res_use_all_space`	Не резервировать место под версии записей.
`res_fix_fss_data <кодировка>`	Исправить кодировку данных.
`res_fix_fss_metadata <кодировка>`	Исправить кодировку метаданных.
`res_metadata_only`	Производит восстановление только метаданных.
`verbint <n>`	Подробный вывод лога действии с заданным интервалом обработанных записей.
`res_skip_data <шаблон>`	Пропускать данные таблиц, имена которых соответствуют указанному здесь регулярному выражению в SQL-синтаксисе (см. оператор `SIMILAR TO`).
`res_keep_data <шаблон>`	Включать только данные таблиц, имена которых соответствуют указанному здесь регулярному выражению в SQL-синтаксисе (см. оператор `SIMILAR TO`).

Опция	Описание
`res_stat TDRW`	Вывести статистику в процессе работы (работает вместе с `-verbose`). T — время от начала работы `gbak`; D — время прошедшее с последнего вывода на экран; R — число страниц прочитанных с последнего вывода на экран; W — число страниц записанных с последнего вывода на экран
`res_parallel_workers <n>`	Количество параллельных рабочих потоков для рестора. Включает многопоточное выполнение восстановления БД, если `MaxParallelWorkers > 1` (см. `firebird.conf`). Вместо этого параметра утилиты можно указать параметр конфигурации `ParallelWorkers`.

Настройка базы данных

Действие `action_properties` позволяет настроить параметры базы данных. Аналогично `gfix`.

Таблица 8.26 — Опции action_properties

Опция	Описание
`dbname <база данных>`	Путь к основному базы данных.
`prp_page_buffers <n>`	Устанавливает новый размер страничного кэша БД в страницах. `<n>` - количество страниц.
`prp_sweep_interval <n>`	Изменяет интервал транзакций для автоматической сборки мусора (sweep). `<n>` устанавливает новый интервал. Если `n = 0`, автоматическая сборки мусора не выполняется.
`prp_shutdown_db <n>`	Закрывает базу данных, когда нет соединений с базой данных, или по истечении указанного таймаута.
`prp_deny_new_transactions <n>`	Завершает работу базы данных, если в конце указанного таймаута нет активных транзакций; запрещает новые транзакции в течение этого периода ожидания; завершается ошибкой, если в конце периода ожидания есть активные транзакции.
`prp_deny_new_attachments <n>`	Завершает работу базы данных, если в конце указанного таймаута нет активных подключений; запрещает новые подключения к базе данных в течение периода ожидания; завершится ошибкой, если в конце периода ожидания есть активные подключения.
`prp_set_sql_dialect <n>`	Изменяет диалект базы данных. <n> может быть 1 или 3.
`prp_access_mode [prp_am_readonly | prp_am_readwrite]`	Устанавливает режим записи для базы данных - только для чтения или чтение/запись. Этот параметр может принимать два значения.
`prp_reserve_space [prp_res_use_full | prp_res]`	Включает 100% заполнение страниц БД (full) или 80% заполнение по умолчанию (reserve). 100%-е заполнение имеет смысл для баз только для чтения.
`prp_write_mode [prp_wm_async | prp_wm_sync]`	Переключает режимы синхронной/асинхронной записи `Forced Writes`.

Опция	Описание
prp_activate <теневая копия>	Параметр предназначен для активации теневой копии. <теневая копия> - адрес и имя файла теневой копии (или первого из файлов).
prp_db_online	Возвращает в режим "online" базу, остановленную с помощью -shut
prp_force_shutdown <n>	Предназначен для принудительного закрытия базы данных. <n> указывает количество секунд, через которое произойдет закрытие. Если остались активные пользователи, они отключатся, последние результаты их работы будут потеряны. Такое средство нужноприменять с осторожностью.
prp_attachments_shutdown <n>	Предназначен для запрета новых соединений с БД. <n> указывает количество секунд, через которое произойдет отключение БД. Отключение отменится, если к этому времени еще останутся активные соединения.
prp_transactions_shutdown <n>	Предназначен для запрета запуска новых транзакций. <n> указывает количество секунд, через которое произойдет отключение БД. Отключение отменится, если к этому времени еще останутся активные транзакции.
prp_shutdown_mode [prp_sm_normal | prp_sm_multi | prp_sm_single | prp_sm_full]	Останавливает базу данных. Используется с одним из дополнительных параметров -attach, -force или -tran. Опция multi позволяет устанавливать соединение с БД только SYSDBA и владельцем БД. Опция single похожа на multi, за тем исключением, что только один из пользователей – SYSDBA или владелец БД – может соединиться. Опция full не позволяет соединиться с БД никому, даже SYSDBA или владельцу БД.
prp_online_mode [prp_sm_normal | prp_sm_multi | prp_sm_single | prp_sm_full]	Возвращает в режим "online"базу, остановленную с помощью -shut. Опция normal позволяет устанавливать соединение с БД любым авторизованным пользователям, а не только SYSDBA и владельцем БД. Опция multi позволяет устанавливать соединение с БД только SYSDBA и владельцем БД. Опция single похожа на multi за тем исключением, что допускается только одно подключение от SYSDBA или владельца БД.
prp_nolinger	Останавливает указанную базу данных, после того как последнего соединения не стало, независимо от установок LINGER в базе данных.

Проверка базы данных

Действие action_repair позволяет инициировать проверку согласованности базы данных и исправление найденных ошибок. Аналогично gfix с опциями -validate, -full и -mend.

Таблица 8.27 — Опции action_repair

Опция	Описание
dbname <база данных>	Путь к файлу базы данных.

Опция	Описание
rpr_commit_trans <ID> | all	Завершает подтверждением зависшую (in limbo) двухфазную транзакцию с идентификатором <ID>, или все зависшие транзакции (all).
rpr_rollback_trans <ID> | all	Завершает откатом зависшую (in limbo) двухфазную транзакцию с идентификатором <ID>, или все зависшие транзакции (all).
rpr_recover_two_phase <ID> | all	Автоматическое двухфазное восстановление limbo транзакции с номером <ID>, или всех транзакций (all).
rpr_commit_trans_64 [int64 value]	Завершает подтверждением зависшую (in limbo) двухфазную транзакцию с идентификатором <ID>, или все зависшие транзакции (all).
rpr_rollback_trans_64 [int64 value]	Завершает откатом зависшую (in limbo) двухфазную транзакцию с идентификатором <ID>, или все зависшие транзакции (all).
rpr_recover_two_phase_64 [int64 value]	Автоматическое двухфазное восстановление limbo транзакции с номером <ID>, или всех транзакций (all).
rpr_check_db	Проверка базы данных без исправления каких-либо проблем.
rpr_ignore_checksum	Игнорировать ошибки контрольных сумм при проверке.
rpr_kill_shadows	Удаляет все неиспользуемые теневые копии базы данных.
rpr_mend_db	Подготавливает повреждённую базу данных для резервного копирования. Помечает повреждённые записи как неиспользуемые.
rpr_validate_db	Проверяет базу данных на уровне страниц и освобождает страницы, помеченные как используемые, но никому не принадлежащие (orphans).
rpr_full	Используется вместе с -validate_db для более глубокой проверки на уровне записей; освобождает неназначенные фрагменты записей.
rpr_sweep_db	Запускает принудительную сборку мусора в БД.
rpr_list_limbo_trans	Выводит список «зависших» транзакций (limbo).
rpr_icu	Исправляет базу данных для работы с текущей версией ICU.
rpr_par_workers <n>	Количество параллельных рабочих потоков для сборки мусора. Включает многопоточную сборку мусора, если MaxParallelWorkers > 1 (см. firebird.conf). Вместо этого параметра утилиты можно указать параметр конфигурации ParallelWorkers.

Статистика базы данных

Действие action_db_stats выводит статистику базы данных. Аналогично gstat.

Таблица 8.28 — Опции action_db_stats

Опция	Описание
dbname <база данных>	Путь к файлу базы данных.

Опция	Описание
sts_record_versions	Добавляет статистику о средних длинах записей, количестве версий и информацию о BLOB.
sts_nocreation	Не выводить дату создания.
sts_table <таблица>	Анализ таблицы или списка таблиц и любых индексов, принадлежащих указанным таблицам. За этим параметром следует список имен таблиц, которые вы хотите проанализировать. Список должен быть в верхнем регистре, и каждая таблица разделяется пробелом.
sts_data_pages	Статистика страниц данных – информация о таблицах содержащихся в базе данных. Пользовательские и системные индексы, системные таблицы не анализируются.
sts_hdr_pages	Статистика заголовочной страницы - это информация о глобальных свойствах всей базы данных, хранящаяся на заголовочной странице каждой базы данных.
sts_idx_pages	Статистика индексов – информация об индексах в базе данных. Пользовательские и системные таблицы, системные индексы не анализируются.
sts_sys_relations	Статистика для системных таблиц и индексов в дополнение к пользовательским таблицам и индексам.
sts_encryption	Статистика по зашифрованным и незашифрованным страницам БД.

Отображение информации заголовка базы данных employee на локальном компьютере:

```
./rdbsvcmgr service_mgr user sysdba password masterkey action_db_stats dbname
employee sts_hdr_pages
```

Работа с пользователями

Можно использовать Services API для отображения, добавления, удаления и изменения пользователей аналогично gsec.

Действие action_display_user выводит информацию обо всех зарегистрированных пользователях. Действие action_display_user_adm дополнительно показывает являются ли они администраторами БД.

Таблица 8.29 — Опции action_display_user и action_display_user_adm

Опция	Описание
dbname <БД_безопасности>	Путь к файлу базы данных безопасности.
sec_username <имя пользователя>	Имя пользователя, для которого будет выполнена операция.
sql_role_name <имя роли>	Название роли, чьи права будет использовать указанный пользователь.

Действие action_add_user нужно для добавления нового пользователя. Аналогично gsec -add.

Таблица 8.30 — Опции action_add_user

Опция	Описание
`dbname <база данных>`	Путь к файлу базы данных
`sec_username <имя пользователя>`	Имя пользователя
`sql_role_name <имя роли>`	Название роли, чьи права будет использовать указанный пользователь.
`sec_password <пароль>`	Пароль.
`sec_groupname <имя группы>`	Группа.
`sec_firstname <имя>`	Имя.
`sec_middlename <отчество>`	Отчество.
`sec_lastname <фамилия>`	Фамилия.
`sec_userid <uid>`	Идентификатор пользователя.
`sec_groupid <gid>`	Идентификатор группы.
`sec_admin [int32 value]`	Если значение не равно 0, пользователю будут назначены права администратора

Действие `action_delete_user` нужно для удаления нового пользователя. Аналогично `gsec -delete`.

Таблица 8.31 — Опции gsec -delete

Опция	Описание
`dbname <база данных>`	Путь к файлу базы данных.
`sec_username <имя пользователя>`	Имя пользователя.
`sql_role_name <роль>`	Роль.

Действие `action_modify_user` нужно для удаления изменения информации о пользователе. Аналогично `gsec -modify`.

Таблица 8.32 — Опции action_modify_user

Опция	Описание
`dbname <база данных>`	Путь к файлу базы данных.
`sec_username <имя пользователя>`	Имя пользователя.
`sql_role_name <роль>`	Роль.
`sec_password <пароль>`	Пароль.
`sec_groupname <имя группы>`	Группа.
`sec_firstname <имя>`	Имя.
`sec_middlename <отчество>`	Отчество.
`sec_lastname <фамилия>`	Фамилия.
`sec_userid <uid>`	Идентификатор пользователя.
`sec_groupid <gid>`	Идентификатор группы.

Опция	Описание
sec_admin [int32 value]	

Инкрементный бэкап

Действие action_nbak позволяет создавать инкрементные резервные копии.

Таблица 8.33 — Опции action_nbak

Опция	Описание
dbname <база данных>	Путь к файлу базы данных.
nbk_file <резервный файл>	Путь к файлу резервной копии.
nbk_level <уровень>	Уровень резервной копии.
nbk_guid <GUID>	GUID предыдущего бекапа, относительно которого нужно создавать инкремент.
nbk_no_triggers	Не запускать триггеры базы данных.
nbk_direct [ON | OFF]	Включает/выключает небуферизованный ввод/вывод при чтении БД.

Восстановление из инкрементных резервных копий

Действие action_nrest нужно для восстановления базы данных из инкрементных резервных копий.

Таблица 8.34 — Опции action_nrest

Опция	Описание
dbname <база данных>	Путь к файлу базы данных.
nbk_file <резервный файл>	Путь к файлу резервной копии.
nbk_inplace	Восстанавливать инкремент в существующую базу данных.

Аудит

Таблица 8.35 — Действия для ведения аудита

Действие	Описание
action_trace_start -trc_cfg <конфигурационный файл> -trc_name <имя сессии>	Запускает сессию пользовательской трассировки.
action_get_fb_log, action_get_ib_log	Отображают лог СУБД.
action_trace_suspend -trc_id <ID сессии>	Приостановление сеанса трассировки.
action_trace_resume -trc_id <ID сессии>	Возобновление сеанса трассировки.

Действие	Описание
action_trace_stop -trc_id <ID сессии>	Завершение сеанса трассировки.
action_trace_list	Вывод списка существующих сеансов трассировки.

Отображения

Таблица 8.36 — Работа с отображениями

Действие	Описание
action_set_mapping -dbname <база данных> -sql_role_name <роль>	Предоставляет администраторам Windows роль RDB$ADMIN в указанной базе данных, если используется доверительная авторизация
action_drop_mapping -dbname <база данных> -sql_role_name <роль>	Снимает с администратора Windows роль RDB$ADMIN в указанной базе данных.

Онлайн-валидация

Действие action_validate позволяет запустить онлайн-проверку базы данных.

Таблица 8.37 — Опции action_validate

Опция	Описание
dbname <база данных>	Путь к файлу базы данных.
val_tab_incl <шаблон>	Шаблон таблиц, включенных в проверку. По умолчанию все пользовательские таблицы включены, системные таблицы не проверяются.
val_tab_excl <шаблон>	Шаблон для таблиц, исключенных из проверки.
val_idx_incl <шаблон>	Шаблон для индексов, включенных в проверку. По умолчанию % – все индексы включены.
val_idx_excl <шаблон>	Шаблон списка индексов для исключения из процесса валидации.
val_lock_timeout <число>	Таймаут ожидания блокировки на таблицу (в секундах). По умолчанию 10 секунд, -1 - бесконечное ожидание.

8.9 Утилита rdbguard

Утилита rdbguard контролирует состояние сервера. Если сервер был нештатно остановлен, Guardian автоматически перезапускает его.

Таблица 8.38 — Опции rdbguard

Опция	Описание
-s(ignore)	Если запуск сервера завершается с ошибкой, выполнять повторные попытки запуска.
-(o)netime	Если запуск сервера завершается с ошибкой, не пытаться запустить его повторно.

Опция	Описание
-(f)orever	Если запуск или выполнение сервера завершается с ошибкой, выполнять его повторный запуск (режим по умолчанию).
-(d)aemon	Запускать сервер в режиме демона, отключаясь от родительского терминала.
-(p)idfile путь_к_файлу	Использовать указанный файл в качестве PID-файла сервера.
-(g)uardpidfile путь_к_файлу	Использовать указанный файл в качестве PID-файла rdbguard.
-(t)imeout таймаут	Время в секундах, которое rdbguard будет ожидать с момента начала остановки сервера, прежде чем убить его процесс. По умолчанию 1800 секунд (30 минут). Если сервер не успевает остановиться в отведенное время, то это может привести к неполной записи на диск измененных страниц из кэша. В этом случае целостность базы данных не нарушается, но может наблюдаться снижение производительности.

8.10 Коды возврата утилит

Для утилит gfix, gstat, rdbsvcmgr, rdbtracemgr, rdblogmgr, rdbreplmgr, hashgen, mint,nbackup,rdb_lock_print:

- 0 – успешно
- 1 – ошибка

Для утилиты gbak:

- 0 – успешно
- 1 – ошибка
- 2 – база данных не переведена в режим online из-за ошибки при восстановлении из резервной копии

Для утилиты rdbrepldiff:

- 0 – успешно (базы идентичны)
- 1 – базы не идентичны
- 2 – ошибка в процессе работы

Для утилиты gsec:

- 0 – успешно
- 15 – невозможно открыть базу данных
- 16 – ошибка в опции утилиты
- 17 – операция не указана
- 18 – имя пользователя не указано
- 19 – ошибка добавления записи
- 20 – ошибка изменения записи
- 21 – ошибка поиска/изменения записи
- 22 – запись не найдена для пользователя:
- 23 – ошибка удаления записи

- 24 – ошибка поиска/удаления записи
- 28 – ошибка поиска/отображения записи
- 29 – недопустимый параметр, опция не определена
- 30 – операция уже указана
- 31 – пароль уже указан
- 32 – идентификатор пользователя уже указан
- 33 – идентификатор группы уже указан
- 34 – проект уже указан
- 35 – организация уже указан
- 36 – имя уже указано
- 37 – отчество уже указано
- 38 – фамилия уже указана
- 40 – указана недопустимая опция
- 41 – указана неоднозначная опция
- 42 – не указана операция для параметров
- 43 – нет допустимых параметров для этой операции
- 44 – указаны несовместимые опции
- 74 – ошибка выделения памяти
- 75 – ошибка
- 76 – недопустимое имя пользователя (допускается не более 31 байта)
- 77 – недопустимый пароль (допускается не более 16 байт)
- 78 – база данных уже указана
- 79 – имя администратора базы данных уже указано
- 80 – пароль администратора базы данных уже указан
- 81 – роль уже указана
- 92 – указана недопустимая в интерактивном режиме опция
- 93 – ошибка при закрытии базы данных безопасности
- 94 – ошибка при выполнении запроса в базе данных безопасности
- 96 – ошибка при получении пароля из файла
- 97 – ошибка при изменении `AUTO ADMINS MAPPING` в базе данных безопасности
- 99 – недопустимый параметр для `-MAPPING`, допускается только `SET` или `DROP`
- 103 – недопустимый параметр для `-ADMIN`, принимается только `YES` или `NO`
- 104 – недостаточно прав для завершения операции
- 200 – пользователь не найден в `LDAP`
- 201 – ошибка при изменении пароля `LDAP`

Для утилиты `rdbserver`:

- 0 – успешно
- 1 – ошибка
- 2 – ошибка запуска

Для утилиты `isql`:

- 0 – успешно
- 1 – ошибка

- 101 – ошибка аутентификации
- 102 – ошибка ввода/вывода
- 103 – ошибка `DSQL`
- 104 – ошибка функции `SHOW`

Глава 9

Администрирование функций безопасности

9.1 Основные термины и определения

Сервер баз данных (далее сервер) — уста новленная СУБД Ред База Данных. Для соединения с сервером по сети по умолчанию используется порт 3050 (настраивается через параметр `RemoteServicePort` в конфигурационном файле Ред База Данных).

Конфигурационный файл сервера — текстовый файл `firebird.conf`, расположенный в корневой директории каталога установки сервера. Файл содержит параметры настройки сервера.

База данных безопасности — база данных с именем `security3.fdb`, расположенная в корневой директории каталога установки сервера. В этой базе хранятся параметры пользователей системы, политики доступа, глобальные роли (см. Приложение Д). Для каждой базы данных база данных безопасности может переопределена в файле `databases.conf` (параметр `SecurityDatabase`). Любая база данных может быть базой данных безопасности для самой себя.

Пользователь — субъект доступа к базам данных сервера.

Политика безопасности — совокупность требований к сложности пароля и параметрам сессий пользователя. Политики назначаются пользователям для повышения общей безопасности системы.

Идентификация — предъявление пользователем имени (логина) для входа в систему.

Аутентификация — процедура подтверждения пользователем того, что он тот, чье имя он предъявил в ходе идентификации.

Фактор аутентификации — данные, предъявленные пользователем, для проверки одного из условий, необходимых для прохождения аутентификации. Факторы могут быть первичными и вторичными. Первичные факторы — пароль, контекст безопасности ОС и сертификат. Вторичные могут быть предъявлены после первичной аутентификации по защищенному каналу, установленному в результате первичной аутентификации. Вторичные факторы могут быть любыми, например, данные биометрии. Их передача шифруется ключами, выработанными в результате первичной аутентификации.

Многофакторная аутентификация — аутентификация с использованием нескольких факторов аутентификации (пароль, сертификат). Факторы, необходимые для прохождения аутентификации, определяются политикой безопасности. Только в режиме многофакторной аутентификации производится проверка соответствия пароля требованиям политики безопасности.

Роль — совокупность прав для доступа к той или иной базе данных. Роли могут назначаться пользователям. Каждый пользователь может иметь произвольное количество ролей в одной или нескольких базах данных. Каждая роль может быть назначена произвольному количеству пользователей.

Администратор сервера БД — пользователь с именем `SYSDBA`. Создается при установке сервера. Обладает всеми правами по управлению работой сервера и полным доступом ко всем базам данных сервера. Пароль для `SYSDBA` по умолчанию – `masterkey`.

Системный администратор — пользователь, которому назначена роль `RDB$ADMIN`. Предназначен для распределения прав пользователей, а также для операций обслуживания баз данных (резервное копирование, восстановление и т.д.).

Системный каталог — совокупность системных таблиц, содержащая информацию обо всех объектах базы данных. Создается при создании БД и изменяется при изменении метаданных в БД.

9.2 Общая модель защиты

Модель защиты описывает совокупность объектов защиты, субъектов доступа к защищаемым объектам и используемые механизмы безопасности, обеспечивающие выполнение заданных требований к безопасности информации.

Объектами защиты в Ред Базы Данных являются:

- хранящиеся в Ред Базе Данных пользовательские данные (записи, поля);
- данные системного каталога (метаданные);
- операции над данными и метаданными.

Субъектами доступа являются пользователи системы, прошедшие процесс идентификации и аутентификации, а также запущенные от их имени процедуры и функции.

Система защиты состоит из следующих механизмов безопасности:

- идентификация и аутентификация;
- разграничение доступа;
- регистрация событий;
- очистка освобождаемых ресурсов;
- контроль целостности информационных объектов.

Идентификация и аутентификация

В Ред База Данных идентификация и аутентификация основана на имени пользователя, его пароле, а также других факторах аутентификации, которые могут быть затребованы сервером в ходе аутентификации.

Для того, чтобы пройти процедуру идентификации, пользователь обязан предъявить свой логин (имя пользователя). Основываясь на этой информации, сервер в дальнейшем определит, как должна происходить аутентификация пользователя, и какие права сможет получить пользователь после прохождения аутентификации.

Все необходимые настройки идентификации и аутентификации задаются администратором Ред База Данных в файле конфигурации сервера.

СУБД Ред База Данных также обеспечивает возможность аутентификации пользователя на сервере с использованием протокола `LDAP`. При аутентификации из службы каталогов запрашивается дополнительная информация о пользователе (ФИО из атрибута `CN`), и на основе этих данных заполняются контекстные переменные на сервере БД.

СУБД Ред База Данных имеет возможность аутентификации пользователя на сервере через протокол `Kerberos` с механизмом `GSSAPI`, обеспечивая более эффективный доступ к ресурсам и взаимную проверку подлинности.

Разграничение доступа

В Ред База Данных доступ субъектов безопасности к защищаемым объектам может быть разграничен следующими способами:

- субъекту (пользователю, роли, процедуре, триггеру, пакету, функции, представлению) могут быть назначены определенные права;
- пользователю или роли может быть назначена одна или несколько ролей, наделенных необходимыми правами.

По сути, назначение пользователю роли эквивалентно включению пользователя в группу, то есть ролям даются определённые права на доступ к защищаемым объектам и работе с ними.

После назначения роли пользователю он может получить права этой роли, если укажет ее при подключении к базе данных. Пользователю может быть назначено несколько ролей. В этом случае,

если пользователь не указал при подключении к базе данных конкретную роль, то он получает права только тех ролей, которые ему назначены с DEFAULT (действует принцип кумулятивного действия ролей). Возможно также назначение одной роли другой.

Специальные учетные записи

SYSDBA

Изначально в системе существует только один пользователь – администратор сервера SYSDBA (пароль по умолчанию – masterkey). Этот пользователь обладает полными правами на выполнение всех функций по управлению работой сервера и работе с базами данных.

> Во время установки сервера или сразу после нее рекомендуется как можно быстрее сменить пароль по умолчанию пользователя SYSDBA.

Пользователи POSIX

В POSIX системах, включая MacOSX, Ред База Данных будет трактовать пользователя POSIX точно так же как и пользователя, хранящегося в собственной базе данных безопасности, до тех пор, пока сервер видит клиента в качестве доверенного хоста. Учетные записи пользователя должны существовать как на клиенте, так и на сервере. Для того чтобы установить доверительные отношения с хостом клиента, необходимо на сервере занести соответствующую запись в файл /etc/hosts.equiv или /etc/gds_hosts.equiv. В файле hosts.equiv прописываются доверительные отношения на уровне операционных систем, которые, соответственно, распространяются на все сервисы (например, rlogin, rsh, rcp). В файле gds_hosts.equiv устанавливаются доверительные отношения между хостами, только для Ред Базы Данных. Формат записи идентичен для обоих файлов, и выглядит следующим образом:

```
hostname [username]
```

В POSIX системах пользователь root может выступать в роли SYSDBA. Ред база данных в этом случае будет трактовать имя пользователя root как SYSDBA, и он будет иметь доступ ко всем базам данных сервера.

Пользователи Windows

В операционных системах семейства Windows NT вы также можете пользоваться учётными записями ОС. Для этого необходимо, чтобы в файле конфигурации firebird.conf (параметр AuthServer) в списке плагинов присутствовал провайдер Win_Sspi. Кроме того, этот плагин должен присутствовать и в списке плагинов клиентской стороны (параметр AuthClient).

Администраторы операционной системы Windows автоматически не получают права SYSDBA при подключении к базе данных (если, конечно, разрешена доверенная авторизация). Имеют ли администраторы автоматические права SYSDBA, зависит от установки значения флага AUTO ADMIN MAPPING.

Владельцы базы данных

Владелец базы данных — это либо текущий пользователь (CURRENT_USER), который был в момент создания, либо пользователь который был указан в параметрах USER и PASSWORD в операторе CREATE DATABASE.

Владелец базы данных является администратором в ней и имеет полный доступ ко всем объектам этой базы данных, даже созданных другими пользователями.

Администраторы

Администратор — это пользователь, которые имеет достаточные права для чтения и записи, создания, изменения и удаления любого объекта в базе данных. В таблице показано, как привилегии «Суперпользователя» включены в различных контекстах безопасности.

Таблица 9.1 — Администраторы

Пользователь	Роль RDB$ADMIN	Замечание
SYSDBA	Автоматически	Существует автоматически на уровне сервера. Имеет полные привилегии ко всем объектам во всех базах данных. Может создавать, изменять и удалять пользователей.
Пользователь root или суперпользователь в POSIX	Автоматически	Также как SYSDBA.
Владелец базы данных	Автоматически	Также как SYSDBA, но только в этой базе данных.
Администраторы Windows	Устанавливается в CURRENT_ROLE, если вход успешен	Также как SYSDBA, если соблюдены следующие условия: • В файле конфигурации firebird.conf (параметр AuthServer) в списке плагинов присутствовал провайдер Win_Sspi. Кроме того, этот плагин должен присутствовать и в списке плагинов клиентской стороны (параметр AuthClient). • Во всех базах данных, где требуется полномочия суперпользователя должен быть включен AUTO ADMIN MAPPING или создано отображение предопределенной группы DOMAIN_ANY_RID_ADMINS на роль RDB$ADMIN. • При входе не указана роль.
Обычный пользователь или пользователь POSIX	Должна быть предварительно выдана и должна быть указана при входе	Также как SYSDBA, но только в тех базах данных, где эта роль предоставлена.
Пользователь Windows	Должна быть предварительно выдана и должна быть указана при входе	Также как SYSDBA, но только в тех базах данных, где эта роль предоставлена. Доступно только если в файле конфигурации firebird.conf (параметр AuthServer) в списке плагинов присутствовал провайдер Win_Sspi. Кроме того, этот плагин должен присутствовать и в списке плагинов клиентской стороны (параметр AuthClient).

Специальные предопределенные роли

Существует ряд предопределённых ролей, предназначенных для выполнения функций поддержки и администрирования СУБД. Роли и их назначение приведены в следующей таблице:

Таблица 9.2 — Предопределенные роли

Имя роли	Назначение роли
RDB$ADMIN	Дает права пользователя SYSDBA, но только в текущей базе данных.
PUBLIC	Роль по умолчанию для вновь создаваемых пользователей. Не имеет никаких прав. Не существует в базе данных в явном виде.

Системная роль RDB$ADMIN, присутствует в каждой базе данных. Привилегии вступают в силу сразу после входа в обычную базу данных с указанием роли RDB$ADMIN, после чего пользователь получает полный контроль над всеми объектами базы данных.

Для предоставления и удаления роли RDB$ADMIN в обычной базе данных используются операторы GRANT и REVOKE, как и для назначения и отмены остальных ролей.

```
GRANT RDB$ADMIN TO <имя пользователя>
REVOKE RDB$ADMIN FROM <имя пользователя>
```

Для использования прав роли RDB$ADMIN пользователь просто указывает её при соединении с базой данных. Он также может указать её позднее с помощью оператора SET ROLE.

Предоставление роли RDB$ADMIN в базе данных безопасности даёт возможность создавать, изменять и удалять учётные записи пользователей. Для этого могут использоваться не только операторы GRANT и REVOKE, но и SQL команды управления пользователями: CREATE USER и ALTER USER, в которых указываются специальные опции GRANT ADMIN ROLE и REVOKE ADMIN ROLE.

```
CREATE USER <имя пользователя> PASSWORD '<пароль>' GRANT ADMIN ROLE
ALTER USER <имя пользователя> REVOKE ADMIN ROLE
```

Для управления учётными записями пользователей пользователь, имеющий права на роль RDB$ADMIN, должен подключиться к базе данных безопасности с этой ролью.

Чтобы управлять пользователями из обычной базы данных, у этого пользователя должны быть права на роль RDB$ADMIN в это базе данных. Он определяет роль при соединении и может в ней выполнить любой SQL запрос. Иначе управление учётными записями посредством SQL запросов недоступно.

То же самое можно сделать используя утилиту gsec указав параметр -admin для сохранения атрибута RDB$ADMIN учётной записи пользователя:

```
gsec -add <имя пользователя> -pw <пароль> -admin yes
gsec -mo <имя пользователя> -admin no
```

Для управления пользователями через утилиту gsec роль RDB$ADMIN должна быть указана в переключателе -role.

В обоих случаях пользователь с правами RDB$ADMIN роли может всегда передавать эту роль другим. Другими словами, WITH ADMIN OPTION уже встроен в эту роль и эту опцию можно не указывать.

Привилегии на роль RDB$ADMIN могут давать только администраторы.

AUTO ADMIN MAPPING

Администраторы операционной системы Windows автоматически не получают права SYSDBA при подключении к базе данных (если, конечно, разрешена доверенная авторизация). Имеют ли администраторы автоматические права SYSDBA зависит от установки значения флага AUTO ADMIN MAPPING. Это флаг в каждой из баз данных, который по умолчанию выключен. Если флаг AUTO ADMIN MAPPING включен, то он действует, когда администратор Windows:

- подключается с помощью доверенной аутентификации
- не определяет никакой роли при подключении

После успешного подключения текущей ролью будет являться RDB$ADMIN.

Включение и выключение флага AUTO ADMIN MAPPING в обычной базе данных осуществляется следующим образом:

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

Альтернативой включения такого флага служит создание отображения предопределённой группы DOMAIN_ANY_RID_ADMINS на роль RDB$ADMIN:

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

Эти операторы могут быть выполнены владельцами баз данных или администраторами.

В обычных базах данных статус `AUTO ADMIN MAPPING` проверяется только во время подключения. Если администратор имеет роль `RDB$ADMIN` потому, что произошло автоматическое отображение во время входа, то он будет удерживать эту роль на протяжении всей сессии, даже если он или кто-то другой в это же время выключает автоматическое отображение. Точно также, включение `AUTO ADMIN MAPPING` не изменит текущую роль в `RDB$ADMIN` для администраторов, которые уже подключились.

Для включения `AUTO ADMIN MAPPING` в базе данных пользователей можно также использовать утилиту командной строки `gsec`:

```
gsec -mapping set
gsec -mapping drop
```

Только `SYSDBA` может включить `AUTO ADMIN MAPPING`, если он выключен, но любой администратор может выключить его.

При выключении `AUTO ADMIN MAPPING` пользователь отключает сам механизм, который предоставлял ему доступ и, таким образом, он не сможет обратно включить `AUTO ADMIN MAPPING`. Даже в интерактивном `gsec` сеансе новая установка флага сразу вступает в силу.

Доступ к административным функциям (системным сервисам)

Дискреционный принцип контроля доступа действует не только в отношении операций над объектами конкретной базы данных, но и в отношении операций над базами данных в целом, а также операций с пользователями — то есть в отношении следующих административных функций:

- резервное копирование/восстановление БД (`GBAK`);
- добавление/изменение/удаление пользователя, получение списка пользователей (`GSEC`);
- получение свойств БД, анализ и восстановление поврежденной БД (`GFIX`);
- получение статистики БД (`GSTAT`).

Чтобы получить привилегии на доступ к административным функциям, следует подключаться к сервисам с указанием конкретной роли. Права применяются на конкретное действие, выполняемое через сервис.

Регистрация событий (аудит)

Настройка регистрации событий в Ред База Данных производится с помощью изменения параметров в конфигурационном файле аудита – `fbtrace.conf`. Задаются следующие параметры:

- включение/отключение регистрации событий;
- формат журнала (текстовый, бинарный, запись в syslog в Linux, в журнал событий в Windows);
- типы регистрируемых событий;
- базы данных, для которых будет включена регистрация событий;
- включение/отключение ротации лог-файлов аудита.

События безопасности регистрируются в отдельном лог-файле (журнале аудита), представляющем собой текстовый или бинарный файл. Также события могут регистрироваться в syslog в Linux, в журнале событий в Windows. По умолчанию используется текстовый формат. Подробнее о типах регистрируемых событий и параметрах, сохраняемых для каждого типа события, см. п. 9.6.

Для настройки системы аудита используется файл `fbtrace.conf`, находящийся в корневом каталоге Ред База Данных. По умолчанию в нем отключена регистрация всех типов событий для всех баз данных, т.е. аудит событий не ведется. Конфигурационный файл `fbtrace.conf` считывается СУБД в начале процесса подключения к БД, таким образом, при изменении параметров конфигурационного файла уже существующие подключения будут пользоваться предыдущей (неизмененной) версией конфигурации, а вновь создаваемые — текущей (измененной) версией. Файл журнала создается в каталоге базы данных и получает имя следующего вида: `<database_name>.fbtrace_bin` или `<database_name>.fbtrace_text`. Расположение и название журналов аудита может быть изменено в конфигурационном файле `fbtrace.conf`.

Файлы журнала создаются только в случае необходимости – когда нужен аудит БД в бинарном либо текстовом виде.

Используется система ротации логов, которая активизируется по достижении файлом журнала аудита заданного администратором максимального размера. Под ротацией понимается создание нового лог-файла, в который в дальнейшем происходит запись всех событий. Удаления или архивации старых лог-файлов не предусмотрено и может осуществляться средствами ОС и планировщиками задач.

Для анализа журнала, записанного в двоичном формате, реализована возможность подключения файла с логом к базе данных в качестве внешней таблицы (подробнее см. п. 9.6). В дальнейшем возможна работа с этим журналом как с обычной таблицей в базе данных (с тем ограничением, что таблица будет доступна только для чтения). То есть существует возможность создать единое хранилище логов для всех баз данных, подключив бинарные файлы аудита к одной или нескольким специально созданным для этой цели базам данных.

Очистка освобождаемых ресурсов

Важная особенность СУБД Ред База Данных — версионная архитектура. Это означает, что при изменении записи создается новая версия записи. Предыдущая версия остается существовать. Каждая транзакция, стартовавшая на сервере, имеет свой номер. Запись также имеет номер создавшей ее транзакции. Таким образом, каждая транзакция может видеть свою версию записи и именно к ней иметь интерес.

Требование обезличивания памяти не распространяется на такие версии записей, которые интересны и используются какой-либо транзакцией. В том случае, если запись не интересна ни одной транзакции, т.е. любая из открытых транзакций не получит эту версию записи, то эта версия записи удаляется. В этот момент будет происходить обезличивание памяти, ранее занятой данной версией записи.

При удалении файлов они должны быть перезаписаны последовательностью нулей, единиц (0xFF), случайных значений столько раз, сколько указано в параметре конфигурации. После этого файл переименовывается некоторое количество раз, чтобы в журнале файловой системы не осталось имени исходного файла.

Функция очистки (обезличивания) освобождаемых ресурсов памяти встроена в Ред База Данных и может настраиваться путём задания различных значений параметра `MemoryWipePasses = <integer>` в конфигурационном файле сервера. Целочисленное значение, настраивающее необходимость и метод обезличивания освобождаемой памяти, задаёт следующие действия:

- 0 - не производить обезличивание;
- 1 - обезличивать освобождаемый ресурс за один проход;
- N - производить заданное количество чередующихся заполнений освобождаемого ресурса нулями и единицами. При этом последний проход в любом случае заполняет освобождаемый блок нулями.

9.3 Дискреционный принцип контроля доступа

Общие сведения

Субъектами доступа в СУБД Ред База Данных являются пользователи, а также роли, представления, процедуры, функции, пакеты и триггеры, запущенные от имени пользователей.

Объекты доступа - это сами базы данных и объекты баз данных – таблицы, представления, процедуры, функции, пакеты, генераторы, домены, исключения, роли, наборы символов, сортировки, BLOB-фильтры и записи. Контроль доступа пользователей к базам данных и объектам баз данных осуществляется через делегирование прав пользователям на различные действия над объектами.

Для каждой пары (субъект – объект) задается явное и недвусмысленное перечисление допустимых типов доступа (читать, писать и т.д.), т.е. тех типов доступа, которые являются санкционированными для данного субъекта к данному ресурсу (объекту).

Объекты базы можно разделить на две группы:

- метаданные («данные о данных», структура базы);
- данные (собственно информация, содержащаяся в базе).

Для первой группы определены операции создания, изменения, и удаления объектов, для второй – добавления, изменения, удаления и выборки данных.

Разрешенные для каждого пользователя системы операции над каждым типом объектов определяются правами пользователя. Доступные для каждого объекта операции сведены в таблицу 9.3:

Таблица 9.3 — Объекты и операции над ними

Объект	Операция
База данных	Создание, изменение, удаление
Таблица/представление	Создание, изменение, удаление
Процедура/функция/пакет	Создание, изменение, удаление, запуск (вызов)
Триггер	Создание, изменение, удаление
Генератор	Создание, изменение, удаление, использование
Домен	Создание, изменение, удаление
Исключение	Создание, изменение, удаление, использование
Роль	Создание, удаление, назначение пользователю/другой роли
BLOB фильтр	Объявление и удаление объявления BLOB фильтра
Сортировка	Добавление сортировки и удаление существующей сортировки
Набор символов	Установка сортировки по умолчанию для набора символов
Записи	Добавление, изменение, удаление, выборка, ссылка на указанные столбцы внешним ключом

Права на DDL-операции с триггерами определяются правами субъекта на таблицу.

Основным механизмом реализации принципа дискреционного доступа в СУБД Ред База Данных является индивидуальное назначение прав непосредственно субъектам доступа или назначение прав ролям с последующим присвоением пользователям (или ролям) необходимых им ролей.

Авторизованный пользователь не имеет никаких привилегий до тех пор, пока какие либо права не будут предоставлены ему явно. При создании объекта только его создатель и SYSDBA имеет привилегии на него и может назначать привилегии другим пользователям, ролям или объектам.

Сервер Ред База Данных можно настроить на работу только с определенными базами данных. Для этого необходимо указать список доступных для сервера баз данных в конфигурационном файле

сервера firebird.conf (параметр DatabaseAccess).

Кроме того, внутри каждой из этих баз можно распределить права пользователей, сделав определенные объекты доступные только определенным пользователям или всем пользователям для одной или нескольких операций. Например, для того, чтобы дать возможность всем пользователям, соединившимся с базой данных, изменять данные в какой-либо таблице, необходимо создать роль, которой нужно дать права на изменение данных в этой таблице, а потом назначить эту роль всем пользователям.

Работа с пользователями

Можно управлять учётными записями пользователей средствами операторов SQL. Такая возможность предоставлена следующим пользователям:

- SYSDBA
- Любому пользователю, имеющему права на роль RDB$ADMIN в базе данных безопасности и права на ту же роль для базы данных в активном подключении (пользователь должен подключаться к базе данных с ролью RDB$ADMIN);
- При включенном флаге AUTO ADMIN MAPPING в базе данных пользователей (security3.fdb или той, что установлена для вашей базы данных в файле databases.conf) — любой администратор операционной системы Windows (при условии использования сервером доверенной авторизации - Win_Sspi) без указания роли. При этом не важно, включен или выключен флаг AUTO ADMIN MAPPING в самой базе данных.

Непривилегированные пользователи могут использовать только оператор ALTER USER для изменения собственной учётной записи.

Для создания новой учетной записи пользователя используется следующий синтаксис:

```
CREATE USER <логин> PASSWORD <пароль>
  [FIRSTNAME <имя пользователя>]
  [MIDDLENAME <отчество пользователя>]
  [LASTNAME <фамилия пользователя>]
  [ACTIVE | INACTIVE]
  [USING PLUGIN 'имя плагина']
  [TAGS (<атрибут> [, <атрибут> ...] )]
  [GRANT ADMIN ROLE]

<атрибут> ::= <имя атрибута> = 'строковое значение'
```

Пользователь должен отсутствовать в текущей базе данных безопасности Ред Базе Данных иначе будет выдано соответствующее сообщение об ошибке.

Начиная с версии 3.0 имена пользователей подчиняются общему правилу наименования идентификаторов объектов метаданных. Таким образом, пользователь с именем "Alex" и с именем "ALEX" будут разными пользователями.

Предложение PASSWORD задаёт пароль пользователя. Максимальная длина пароля зависит от того какой менеджер пользователей задействован (параметр UserManager в файле конфигурации firebird.conf). Для менеджер пользователей SRP эффективная длина пароля ограничена 20 байтами *. Для менеджер пользователей Legacy_UserManager максимальная длина пароля равна 8 байт.

Необязательные предложения FIRSTNAME, MIDDLENAME и LASTNAME задают дополнительные атрибуты пользователя, такие как имя пользователя, отчество и фамилия соответственно.

Кроме того можно задать неограниченное количество пользовательских атрибутов с помощью необязательного предложения TAGS.

Если при создании учётной записи будет указан атрибут INACTIVE, то пользователь будет создан в "неактивном состоянии т.е. подключиться с его учётной записью будет невозможно. При указании атрибута ACTIVE пользователь будет создан в активном состоянии (по умолчанию).

С опцией GRANT ADMIN ROLE создаётся новый пользователь с правами роли RDB$ADMIN в базе данных пользователей (security3.fdb). Это позволяет ему управлять учётными записями пользователей, но не дает ему специальных полномочий в обычных базах данных.

Необязательное предложение USING PLUGIN позволяет явно указывать какой плагин управления пользователями будет использован. По умолчанию используется тот плагин, который был указан первым в списке параметра UserManager в файле конфигурации firebird.conf. Допустимыми являются только значения, перечисленные в параметре UserManager.

Если предложение USING PLUGIN не указано, то при добавлении пользователя он сам добавляется во все плагины из списка параметра DefaultUserManagers (в том числе его атрибуты).

Следует учитывать, что одноименные пользователи, созданные с помощью разных плагинов управления пользователями — это разные пользователи.

Это связано с тем, что методы парольной аутентификации (LegacyAuth, Srp, Multifactor) используют разные таблицы в базе данных безопасности для хранения данных пользователей (см. раздел 9.5). Поэтому для системы аутентификации пользователь, созданный менеджером Legacy_UserManager, никак не связан с пользователем, созданным Srp. У них разные пароли и другая пользовательская информация. Но с точки зрения движка эти пользователи одинаковые, так как движок идентифицирует пользователей по именам.

Для плагина SRP эффективная длина пароля ограничена 20 байтами. Для плагина Legacy_UserManager максимальная длина пароля равна 8 байт.

Для изменения существующей учетной записи пользователя используется следующий синтаксис:

```
ALTER {USER <логин> | CURRENT USER}
{
   [SET]
   [PASSWORD <пароль>]
   [FIRSTNAME <имя пользователя>]
   [MIDDLENAME <отчество пользователя>]
   [LASTNAME <фамилия пользователя>]
   [ACTIVE | INACTIVE]
   [TAGS (<атрибут>|DROP <имя атрибута> [, <атрибут>|DROP <имя атрибута>...] )]
}
[USING PLUGIN 'имя плагина']
[{GRANT | REVOKE} ADMIN ROLE];

<атрибут> ::= <имя атрибута> = 'строковое значение'
```

В операторе ALTER USER должно присутствовать хотя бы одно из необязательных предложений.

Это единственный оператор управления учётными записями, который может также использоваться непривилегированными пользователями для изменения их собственных учетных записей, однако это не относится к опциям GRANT/REVOKE ADMIN ROLE и атрибуту ACTIVE/INACTIVE для изменения которых, необходимы административные привилегии.

Если предложение USING PLUGIN не указано, то при изменении атрибутов пользователя они сами сменяется у соответствующих пользователей в плагинах из списка параметра DefaultUserManagers.

Если в каком-либо плагине из списка нет пользователя, то он добавляется, но только если среди изменяемых атрибутов есть пароль.

Для удаления существующей учетной записи пользователя используется следующий синтаксис:

```
DROP USER <логин>
[USING PLUGIN 'имя плагина'];
```

Если предложение USING PLUGIN не указано, то при удалении пользователя он сам удаляется из всех плагинов из списка параметра DefaultUserManagers.

С помощью оператора RESET USER можно разблокировать заблокированного многофакторного пользователя:

```
RESET USER <имя пользователя>
```

Работа с ролями

Роль — средство задания необходимого набора привилегий к объектам базы данных. Роль можно сравнить с группой пользователей операционной системы, имеющих одинаковые привилегии. Роль можно создать с помощью следующего оператора:

```
CREATE ROLE <имя роли>;
```

Одна роль может быть назначена любому количеству пользователей или ролей[12]. Для назначения роли пользователю используется оператор:

```
GRANT [DEFAULT] <имя роли> [, [DEFAULT] <имя роли> ...]
   TO [USER]|[ROLE] <имя польз-я/роли> [, [USER]|[ROLE] <имя польз-я/роли>...]
   [WITH ADMIN OPTION] [{GRANTED BY | AS} [USER] <имя грантора>]
```

Для того, чтобы отнять роль у пользователя, используется оператор:

```
REVOKE [ADMIN OPTION FOR] [DEFAULT] <имя роли> [, [DEFAULT] <имя роли> ...]
   FROM [USER]|[ROLE] <имя польз-я/роли> [, [USER]|[ROLE] <имя польз-я/роли>...]
   [{GRANTED BY|AS} [USER] <имя грантора>]
```

Для удаления роли используется оператор:

```
DROP ROLE <имя роли>;
```

Пользователь, которому предоставлена роль, должен указать её при входе, для того чтобы получить её привилегии. Но помимо них пользователь получает привилегии всех ролей, назначенных ему с DEFAULT (если такие имеются). Поэтому если пользователь не указывает роль при подключении к серверу, то он получает права только тех ролей, которые ему назначены с DEFAULT. Вход в систему с несколькими ролями не поддерживается. Можно изменить текущую роль с помощью оператора SET ROLE.

Роль, под которой пользователь соединяется с базой данных, задается в операторе CONNECT:

```
CONNECT <имя БД> USER <имя пользователя> PASSWORD <пароль> ROLE <имя роли>;
```

[12] Подробнее о назначении роли на роль см. п. 9.3

Специальные предопределенные роли

Существует ряд предопределённых ролей, предназначенных для выполнения функций поддержки и администрирования СУБД. Роли и их назначение приведены в следующей таблице:

Таблица 9.4 — Предопределенные роли

Имя роли	Назначение роли
RDB$ADMIN	Дает полные права, но только в текущей базе данных.
PUBLIC	Роль по умолчанию для вновь создаваемых пользователей. Не имеет никаких прав. Не существует в базе данных в явном виде.
RDB$SYSADMIN	Даёт права управления пользователями, создания/изменения/удаления базы данных, а также возможность назначать права пользователя. Роль RDB$SYSADMIN должна назначаться в базе данных безопасности.
RDB$DBADMIN	Даёт права управления пользователями, выполнения резервного копирования и восстановления базы из бэкапа, управление настройками базы данных, а также возможность назначать права пользователя.
RDB$USER	Даёт права создавать объекты базы данных и манипулировать ими, выполнять хранимые процедуры.

Распределение прав на операции определения объектов базы данных

Для назначения прав субъектам доступа (пользователям, ролям, триггерам, процедурам, функциям, пакетам, представлениям) используется оператор GRANT. Для снятия этих прав используется оператор REVOKE. В СУБД Ред База Данных пользователь или роль могут получить права на выполнение операций изменения структуры базы данных (DDL-операции) и делегировать свои права другим пользователям или ролям (предложение WITH GRANT OPTION в операторе GRANT).

В общем случае операторы GRANT, REVOKE на DDL-операции определены для объектов следующих типов: PROCEDURE, FUNCTION, PACKAGE, ROLE, TABLE, VIEW, EXCEPTION, GENERATOR, DOMAIN, SEQUENCE, CHARACTER SET, COLLATION, FILTER.

Права на создание, изменение и удаление объектов определяются следующим выражением:

```
GRANT {ALL [PRIVILEGES] | {CREATE|ALTER ANY|DROP ANY} [, {CREATE|ALTER ANY|
                                                        DROP ANY}...] } <объект>
   TO <список получателей привилегий> [WITH GRANT OPTION]
   [{GRANTED BY|AS} [USER] <имя грантора>]
```

и, соответственно, права на снятие DDL-привилегий:

```
REVOKE [GRANT OPTION FOR] {ALL [PRIVILEGES] | {CREATE|ALTER ANY|DROP ANY}
                             [, {CREATE|ALTER ANY|DROP ANY}...] } <объект>
   FROM <список обладателей привилегий>
   [{GRANTED BY|AS} [USER] <имя грантора>]
```

Предложение ALL [PRIVILEGES] объединяет привилегии CREATE, ALTER и DROP на указанный тип объекта.

При предоставлении привилегий пользователям можно указать предложение WITH GRANT OPTION, что позволяет свою очередь предоставлять другим пользователям эти привилегии.

Предложение GRANT OPTION FOR в операторе REVOKE позволяет отменить для соответствующего объекта право предоставления другим объектам эти привилегии.

С помощью предложение GRANTED BY можно предоставлять или отозвать права не от имени текущего пользователя, а от другого пользователя. При использовании оператора REVOKE после GRANTED BY права будут удалены только в том случае, если они были зарегистрированы от удаляющего пользователя. Предложение AS является синонимом GRANTED BY. Предложения GRANTED BY и AS могут использовать только владелец базы данных и администраторы. Даже владелец объекта не может использовать их, если он не имеет административных привилегий.

Кроме назначения прав непосредственно пользователям возможно назначение прав ролям, хранимым процедурам, функциям, пакетам, триггерам и представлениям.

Оператор назначения привилегий на создание, удаление и изменение базы данных имеет несколько отличную форму от оператора назначения DDL привилегий на другие объекты метаданных.

```
GRANT CREATE DATABASE TO <список пользователей и ролей>
GRANT {ALL [PRIVILEGES] | {ALTER|DROP} [,{ALTER|DROP}...]} DATABASE
   TO <список получателей привилегий> [WITH GRANT OPTION]
   [{GRANTED BY|AS} [USER] <имя грантора>]
```

и, соответственно, права на снятие DDL-привилегий на базу данных:

```
REVOKE CREATE DATABASE FROM <список пользователей и ролей>
REVOKE [GRANT OPTION FOR] {ALL [PRIVILEGES]|{ALTER|DROP}[,{ALTER|DROP}]} DATABASE
   FROM <список обладателей привилегий>
   [{GRANTED BY|AS} [USER] <имя грантора>]
```

Привилегия CREATE DATABASE является особым видом привилегий, поскольку она сохраняется в базе данных безопасности. Список пользователей имеющих привилегию CREATE DATABASE можно посмотреть в виртуальной таблице SEC$DB_CREATORS. Привилегию на создание новой базы данных могут выдавать только администраторы в базе данных безопасности.

Привилегии ALTER DATABASE и DROP DATABASE относятся только к текущей базе данных. Привилегии на изменение и удаление текущей базы данных могут выдавать только администраторы.

Пример

Для того, чтобы дать пользователю TestUser возможность создавать таблицы, необходимо выполнить следующую команду:

```
GRANT CREATE TABLE TO TestUser;
```

Теперь пользователь сможет создавать таблицы, например:

```
CREATE TABLE TEST_TABLE (ID integer, Name: VARCAHR(50));
```

При попытке создать какой-либо другой объект базы данных пользователь получит сообщение об ошибке:

```
Statement failed, SQLCODE = -901
There is no privilege for this operation.
```

Аналогично пользователь может получить права на создание, изменение, удаление таблиц, представлений, процедур, функций и других объектов базы данных.

Для того, чтобы лишить пользователя права на изменение структуры базы данных, необходимо выполнить команду REVOKE, например:

```
REVOKE CRATE TABLE FROM TestUser;
```

Теперь при попытке выполнить операцию

```
CREATE TABLE TEST_TABLE_2 (ID integer, Name: VARCAHR(50))
```

пользователь получит сообщение об ошибке следующего вида:

```
Statement failed, SQLCODE = -901
There is no privilege for this operation.
```

Распределение прав на операции манипулирования данными

По аналогии с распределением прав на DDL-операции, администратор SYSDBA и владелец объекта могут распределять права и на операции доступа к данным (добавление, изменение, выборка, удаление). Пользователь или роль могут делегировать свои права другим пользователям или ролям (предложение WITH GRANT OPTION в операторе GRANT). Синтаксис запросов по доступу к DML-операциям сведен в таблицу 9.5:

Таблица 9.5 — Назначение прав на DML операции

Операция	Объект	Назначение прав
SELECT, DELETE, INSERT, UPDATE [(<имя столбца>[, <имя столбца>])], REFERENCES (<имя столбца>[, <имя столбца>...]), ALL [PRIVILEGES]	TABLE, VIEW	GRANT <операция> ON [TABLE] {<имя таблицы>|<имя view>} TO <список получателей привилегий> [WITH GRANT OPTION] [{GRANTED BY|AS} [USER] <имя грантора>] REVOKE [GRANT OPTION FOR] <операция> ON [TABLE] {<имя таблицы> | <имя представления>} FROM <список обладателей привилегий> [{GRANTED BY|AS} [USER] <имя грантора>]
USAGE[13]	EXCEPTION, GENERATOR, SEQUENCE	GRANT USAGE ON {EXCEPTION|GENERATOR|SEQUENCE} <имя> TO <список получателей привилегий> [WITH GRANT OPTION] [{GRANTED BY|AS} [USER] <имя грантора>] REVOKE [GRANT OPTION FOR] USAGE ON {EXCEPTION|GENERATOR|SEQUENCE} <имя> FROM <список обладателей привилегий> [{GRANTED BY|AS} [USER] <имя грантора>]
EXECUTE	PROCEDURE, FUNCTION, PACKAGE	GRANT EXECUTE ON {PROCEDURE|FUNCTION|PACKAGE} <имя> TO <список получателей привилегий> [WITH GRANT OPTION] [{GRANTED BY|AS} [USER] <имя грантора>] REVOKE [GRANT OPTION FOR] EXECUTE ON {PROCEDURE | FUNCTION | PACKAGE} <имя> FROM <список обладателей привилегий> [{GRANTED BY|AS} [USER] <имя грантора>]

Предложение WITH GRANT OPTION означает, что пользователь (или роль), кроме права на ту или иную операцию, получит также возможность передать право на эту операцию другому пользователю или роли. Лишить пользователя возможности делегировать свои права можно с помощью оператора REVOKE GRANT OPTION.

С помощью предложение GRANTED BY можно предоставлять или отозвать права не от имени текущего пользователя, а от другого пользователя. При использовании оператора REVOKE после GRANTED BY права будут удалены только в том случае, если они были зарегистрированы от удаляющего пользователя. Предложение AS является синонимом GRANTED BY. Предложения GRANTED BY и

[13] Привилегия USAGE применяется для использования исключений и генераторов/последовательностей в пользовательских запросах.

AS могут использовать только владелец базы данных и администраторы. Даже владелец объекта не может использовать их, если он не имеет административных привилегий.

Кроме назначения прав непосредственно пользователям возможно назначение прав ролям, хранимым процедурам, функциям, пакетам, триггерам и представлениям.

Роли представляют собой гибкий механизм распределения прав сразу нескольким пользователям – можно дать права на требуемые операции какой-либо роли, а затем назначить эту роль всем пользователям, которым необходимы эти права.

Распределение прав ролям происходит аналогично назначению прав пользователям, только в предложениях GRANT... TO... и REVOKE... FROM... указываются не имена пользователей, а имена ролей.

Пример

Для того, чтобы дать пользователю, право на вставку данных в таблицу Test_Table необходимо выполнить команду:

```
GRANT INSERT ON TABLE Test_Table To TESTUSER;
```

Теперь пользователь TestUser сможет добавлять записи в таблицу Test_Table:

```
INSERT INTO Test_Table (ID, Name) VALUES (1, 'Alex');
```

Попытка же выполнить другие операции манипулирования данными вернет ошибку. Например:

```
SELECT * FROM Test_Table;
```

вернет ошибку:

```
Statement failed, SQLCODE = -551
no permission for read/select access to TABLE TEST_TABLE
```

Для того, чтобы лишить пользователя права добавлять записи в таблицу, необходимо выполнить команду:

```
REVOKE INSERT ON TABLE Test_Table FROM TESTUSER;
```

Теперь при попытке вставить запись в таблицу Test_Table пользователь получит сообщение об ошибке:

```
Statement failed, SQLCODE = -551
no permission for insert/write access to TABLE Test_Table
```

Аналогично пользователю можно дать или отнять права на изменение/удаление/просмотр данных, ссылки на столбцы внешним ключом, запуск процедур/функций/пакетов и использование исключений/генераторов/последовательностей в пользовательских запросах. Причем права на изменение записей могут быть выделены как целиком на всю запись, так и только на определенные столбцы записей. При попытке доступа к неразрешенным столбцам таблиц пользователь получит сообщение об ошибке, например:

```
GRANT UPDATE (Name) ON Test_Table TO TestUser;
COMMIT;
CONNECT 'TestDB.fdb' USER TESTUSER PASSWORD TestPass123;
UPDATE Test_Table SET ID=2, Name='Tom';
```

вернет ошибку:

```
Statement failed, SQLCODE = -551
no permission for update/write access to COLUMN ID
```

так как пользователю TESTUSER разрешено изменять значение только столбца Name.

А оператор

```
UPDATE Test_Table Name='Tom';
```

выполнится без ошибок.

Распределение прав на административные функции

Под административными функциями понимается доступ к системным сервисам через клиентский API Ред База Данных. Такими функциями являются следующие:

- резервное копирование (`Backup Database, GBAK`);
- восстановление базы из бэкапа (`Restore Database, GBAK`);
- получение списка пользователей (`Display User, GSEC`);
- добавление пользователя (`Add User, GSEC`);
- удаление пользователя (`Delete User, GSEC`);
- редактирование пользователя (`Modify User, GSEC`);
- получение свойств БД (`Database Properties, GFIX`);
- анализ и восстановление поврежденной БД (`Repair Database, GFIX`);
- получение статистики БД (`Database Stats, GSTAT`).

Чтобы получить привилегии на доступ к административным функциям, следует подключаться к сервисам с указанием конкретной роли. Права применяются на конкретное действие, выполняемое через сервис. Если пользователь не указал роли при подключении к сервису, то он получает права только тех ролей, которые ему назначены с `DEFAULT`.

К примеру, пользователю `testuser` назначили административные права в базе данных `testdb.fdb`. Если для создания резервной копии применить следующую команду:

```
gbak -BACKUP_DATABASE -user testuser -password pass testdb.fdb testbackupdb.fbk
```

то вернется ошибка:

```
gbak: ERROR:Your user name and password are not defined. Ask your database administrator to set up a Firebird login.
```

Но резервное копирование выполнится без ошибок, если указать роль `rdb$admin`:

```
gbak -b -user testuser -password pass -role rdb$admin testdb.fdb testbackupdb.fbk
```

Кумулятивное действие ролей

В СУБД Ред База Данных действует принцип кумулятивного действия ролей. Это значит, что, привилегии конкретной роли - это объединение привилегий, выданных этой роли, и привилегий ролей, назначенных этой роли.

Правила кумулятивного действия ролей:

- если пользователь не указывает роль при подключении к серверу, то он получает права только тех ролей, которые ему назначены с `DEFAULT`;
- если пользователь при подключении указал конкретную роль, то он получает только её привилегии и привилегии ролей, которые ему назначены с `DEFAULT`;
- пользователь может с помощью оператора `SET ROLE` сменить роль, указанную при подключении. В этом случае привилегии пользователя `CURRENT_USER` будут складываться из привилегий роли, назначенной оператором `SET ROLE` и привилегии ролей, которые ему назначены с `DEFAULT`;

- при подключении происходит проверка, что данная роль существует и назначена данному пользователю;
- циклические ссылки ролей друг на друга недопустимы.

Назначение и отбор у ролей прав других ролей происходит аналогично назначению и отбору прав у пользователей или у ролей:

```
GRANT Role1 TO Role2;
REVOKE Role2 FROM Role1;
```

Попытка выполнить повторный `GRANT` одной роли на другую не даст ошибки – права обеих ролей не могут от этого измениться.

Попытка выполнить циклическое наследование прав между ролями:

```
GRANT Role1 TO Role2;
GRANT Role2 TO Role1;
```

Вернет ошибку при выполнении второго оператора:

```
Statement failed, SQLCODE = -607
unsuccessful metadata update
-role ROLE2 can not be granted to role ROLE1
```

При попытке повторного отбора прав одной роли у другой роли:

```
GRANT Role1 TO Role2;
REVOKE Role1 FROM Role2;
REVOKE Role1 FROM Role2;
```

будет выдано предупреждение:

```
Warning: privileges on ROLE1 is not granted to ROLE2.
```

Аналогичное предупреждение будет выдано и при попытке отнять у роли права той роли, которые небыли ей назначены.

Совпадение имён пользователей и ролей недопустимо. Исключить такие совпадения невозможно, так как роли хранятся непосредственно в БД, а пользователи – в базе данных безопасности, поэтому при переносе БД с одного сервера на другой возможны совпадения имён пользователей и ролей. При таком совпадении действует следующее правило: права всегда назначаются на роль в первую очередь, затем, если роль не найдена – на пользователя.

Выполнение процедур

Существует возможность выполнения процедуры не только с правами вызывающего ее пользователя, но и с правами владельца, то есть того пользователя, который ее создал. В этом случае пользователю для выполнения процедуры нужно будет иметь привилегию только на выполнение процедуры, но не права на объекты, с которыми работает эта процедура, эти права должны быть у владельца процедуры.

То, в контексте безопасности какого пользователя будет выполняться процедура, определяется при ее создании/изменении. Для возможности задания этого в объявление процедуры добавлена опция `SQL SECURITY {DEFINER | INVOKER}`. По умолчанию этот параметр равен `INVOKER`, т.е. процедура выполняется с правами вызвавшего ее пользователя. Значение по умолчанию на уровне всей базы данных можно изменить оператором `ALTER DATABASE SET DEFAULT SQL SECURITY`. Синтаксис объявления процедуры следующий:

```
CREATE PROCEDURE <имя хранимой процедуры>
   [(<входной параметр> [, <входной параметр> ...])]
[RETURNS (<выходной параметр> [, <выходной параметр> ...])]
[SQL SECURITY {DEFINER | INVOKER}]
AS
   [<объявление> [<объявление> ...] ]
BEGIN
   <блок операторов>
END
```

9.4 Строки подключения

Если Вы хотите подключиться к базе данных или создать ее, Вы должны, среди прочего, предоставить клиентскому приложению (или, если Вы программист, подпрограммам, которые Вы вызываете) строку подключения к базе данных. Строка соединения однозначно идентифицирует местоположение базы данных на Вашем компьютере, в локальной сети или даже в интернете.

Строка подключения к локальной базе данных

Строка локального подключения состоит из пути к базе данных и имени файла в формате файловой системы, используемой на серверной машине.

- На Linux и других Unix серверах, например:

```
/opt/RedDatabase/examples/empbuild/employee.fdb
```

- На Windows серверах, например:

```
C:\Biology\Data\Primates\Apes\populations.fdb
```

Многие клиенты пользуются относительными путями для подключения (например, `..\examples\empbuild\employee.fdb`). Но пользоваться ими нужно с осторожностью, т.к. не всегда очевидно как они будут расширены. Случайно можно подключиться к другой базе данных и изменения в них могут привести к катастрофическим последствиям.

Вместо пути к файлу лучше указывать алиас базы данных из файла `databases.conf`.

Получив строку локального подключения **без указания сетевого протокола**, клиент Ред Базы Данных сначала попытается сделать прямое `embedded` соединение с файлом базы данных, **минуя аутентификацию**, но учитывая привилегии и ограничения для предоставленного имени пользователя и/или роли (если провайдер `Engine12` включен в `firebird.conf` или `databases.conf` параметр `Providers`). Если файл базы данных существует, но соединение не устанавливается, поскольку клиентский процесс не имеет необходимых прав доступа к файлу, выполняется попытка подключения клиент-сервер (провайдером `Loopback`) в следующем порядке:

1. используя TCP/IP протокол через `localhost`;
2. на Windows: используя WNET (NetBEUI) протокол ;
3. на Windows: используя XNET протокол.

Можно явно указывать определенный протокол в строке подключения (в стиле URL) и таким образом обойти попытку `embedded` подключения:

- `inet://zappa` (TCP/IP подключение, используя алиас на локальном компьютере)
- `inet:///opt/RedDatabase/examples/citylife.fdb` (TCP/IP подключение, используя абсолютный путь на локальном Posix компьютере)

- `inet://C:\Work\Databases\Drills.fdb` (TCP/IP подключение, используя абсолютный путь на локальном Windows компьютере)
- `wnet://doggybase` (NetBEUI подключение, используя алиас на локальном Windows компьютере)
- `wnet://D:\Fun\Games.fdb` (NetBEUI подключение, используя абсолютный путь на локальном Windows компьютере)
- `xnet://security.db` (XNET подключение, используя алиас на локальном Windows компьютере)
- `xnet://C:\Programmas\Firebird\Firebird_3_0\security3.fdb` (XNET подключение, используя абсолютный путь на локальном Windows компьютере)

Строка подключение через TCP/IP

Если Вы при подключении используете протокол TCP/IP, то спецификация файла базы данных должна выглядеть следующим образом:

```
{<имя сервера>|<IP адрес>}[/<номер порта>|/<имя сервиса>]:{<абс. путь>|<алиас>}
```

Примеры:

- Для Linux/Unix:

```
pongo:/opt/RedDatabase/examples/empbuild/employee.fdb
bongo/3052:fury
112.179.0.1:/var/Firebird/databases/butterflies.fdb
localhost:blackjack.fdb
```

- Для Windows:

```
siamang:C:\Biology\Data\Primates\Apes\populations.fdb
sofa:D:\Misc\Friends\Rich\Lenders.fdb
inca/fb_db:D:\Traffic\Roads.fdb
127.0.0.1:Borrowers
```

Строка подключение через NetBEUI

Если Вы при подключении используете протокол NetBEUI, то спецификация файла базы данных должна выглядеть следующим образом:

```
\\{<имя сервера>|<IP адрес>}[@<номер порта>|@<имя сервиса>]\{<абс. путь>|<алиас>}
```

Примеры:

- Для Windows:

```
\\siamang\C:\Biology\Data\Primates\Apes\populations.fdb
\\sofa\D:\Misc\Friends\Rich\Lenders.fdb
\\inca@fb_db\D:\Traffic\Roads.fdb
\\127.0.0.1\Borrowers
```

URL-подобная строка подключения

Существует также унифицированный URL-подобный синтаксис спецификации удалённого сервера. В этом синтаксисе первым параметром указывается наименование протокола, далее указывается имя сервера или IP адрес, номер порта и путь к первичному файлу базы данных или псевдоним.

```
<протокол>://[{<имя сервера>|<IP адрес>}[:<номер порта>|:<имя сервиса>]/]{<абс.
путь>|<алиас>}
```

В качестве протокола можно указать следующие значения:

- INET — TCP/IP;
- WNET — NetBEUI или протокол именованных каналов;
- XNET — локальный протокол.

Примеры:

- Для Linux/Unix:

```
inet://pongo//opt/RedDatabase/examples/empbuild/employee.fdb
inet://bongo:3052/fury
inet://112.179.0.1//var/Firebird/databases/butterflies.fdb
inet://localhost/blackjack.fdb
```

- Для Windows:

```
inet://siamang/C:\Biology\Data\Primates\Apes\populations.fdb
inet://sofa:4044/D:\Misc\Friends\Rich\Lenders.fdb
wnet://inca:fb_db/D:\Traffic\Roads.fdb
wnet://127.0.0.1/Borrowers
```

9.5 Идентификация и аутентификация

Идентификация и аутентификация пользователей являются основой дискреционного доступа субъектов безопасности в систему. Идентификация — это предъявление пользователем своего имени.

Под аутентификацией понимается процедура проверки того, что субъект безопасности именно тот, за кого он себя выдает (тот, чье имя он предъявил при идентификации). Данная проверка производится с помощью некой уникальной информации и может выполняться несколькими способами в зависимости от установок параметра `AuthServer` в файле конфигурации `firebird.conf`. Этот параметр содержит список доступных методов проверки подлинности. Если проверить подлинность с помощью первого метода не удалось, то сервер переходит к следующему и т.д. Если ни один метод не подтвердил подлинность, то пользователь получает сообщение об ошибке.

Ред База Данных 3.0 поддерживает следующие методы аутентификации:

- Безопасная парольная аутентификация использующая алгоритм хэширования SHA для передачи данных: `Srp, Srp224, Srp256, Srp384, Srp512`. По умолчанию используется Srp256;
- Традиционная (`Legacy_Auth`) аутентификация;
- Доверительная (`Win_Sspi`) аутентификация для ОС Windows;
- Многофакторная аутентификация (`Multifactor`)[14] с применением политик безопасности;
- Доверенная аутентификация через механизм GSSAPI (`Gss`);
- Доверенная аутентификации для выполнения `Execute Statement On External` без указания пароля (`ExtAuth`).

Также Ред База Данных поддерживает возможность аутентификации пользователей сервера баз данных с использованием службы каталогов через протокол LDAP.

Весь функционал, который относится к аутентификации, реализован в виде сторонних плагинов аутентификации: legacy_auth, который реализует традиционные методы аутентификации, унаследованные от предыдущих версий (Legacy_Auth и Win_Sspi), gss, srp и multifactor.

Плагины аутентификации состоят из трех частей:

- Client – подготавливает данные на клиенте для отправки на сервер;
- Server – проверяет пароль на правильность;
- User Manager – добавляет, изменяет и удаляет пользователей на сервере. Это не требуется, если используется какой-либо внешний метод аутентификации, такой как доверенная аутентификация Windows.

Все три части на самом деле являются отдельными плагинами, которые должны быть настроены отдельно в firebird.conf.

Информация о пользователях, зарегистрированных для конкретного сервера Ред Базы Данных, хранится в особой базе данных безопасности — security3.fdb. Для каждой базы данных база данных безопасности может быть переопределена в файле databases.conf (параметр SecurityDatabase). Любая база данных может быть базой данных безопасности для самой себя.

В зависимости от используемого плагина аутентификации, данные о пользователях хранятся в разных таблицах базы данных безопасности. С точки зрения системы аутентификации пользователи с одинаковыми именами, но созданные с помощью различных плагинов аутентификации – это разные пользователи. Но для движка это один и тот же пользователь, потому что он идентифицирует пользователей по именам.

Безопасная парольная аутентификация (SRP)

Ред База Данных 3.0 поддерживает новый метод аутентификации пользователей, реализованный в качестве плагина по умолчанию – Secure Remote Password (SRP) Protocol.

В результате работы данного протокола обе стороны получают длинный секретный ключ, проверяемый на соответствие между сторонами после получения. В случаях, когда помимо аутентификации необходимо шифрование данных, SRP предоставляет более надёжные, чем SSH, и более быстрые, чем протокол Диффи-Хеллмана, средства для достижения этой цели. Он также не зависит от третьих лиц, в отличие от Kerberos.

SSH протокол требует предварительного обмена ключами между сервером и клиентом, когда открытый ключ располагается на сервере. SRP не нуждается в этом. От клиента требуется только логин и пароль. Все обмены происходят, когда соединение установлено.

Кроме того, SRP устойчив к атаке посредника (man-in-the-middle).

Для того, чтобы пользователь Ред База Данных смог пройти парольную аутентификацию, он должен быть предварительно создан с помощью плагина управления пользователями Srp:

```
CREATE USER test PASSWORD 'test'
USING PLUGIN Srp;
```

Данные о пользователях, созданных с помощью Srp плагина, хранятся в базе данных безопасности (security3.fdb) в таблице PLG$SRP (см. Приложение Д).

> Использование нового метода аутентификации не совместимо с базами данных безопасности из предыдущих версий и паролями оттуда. Однако, можно использовать процедуру обновления для миграции пользователей из базы security2.fdb. Старая база данных безопасности поддерживается плагином Legacy_Auth, но в таком случае пропадают преимущества в безопасности версии 3.0.

[14] Для работы данной функции необходимо наличие криптопровайдера и криптоплагина.

Благодаря реализации SRP в плагине длина пароля увеличена с 8 символов до 20 и более. На длину пароля нет ограничения в 20 байт и он может быть использован. хэши различных паролей, длина которых более 20 байт, тоже различны. Предел эффективности наступает из-за ограниченной длины хэша в SHA1 равном 20 байт или 160 бит. Рано или поздно найдётся более короткий пароль с тем же хэшем с помощью атаки Brute Force. Именно поэтому часто говорят, что эффективная длина пароля для алгоритма SHA1 составляет 20 байт.

Для работы данного метода необходимо, чтобы в файле конфигурации `firebird.conf` параметр `AuthServer` в списке значений содержал метод аутентификации `Srp`. Кроме того, этот метод должен присутствовать и в списке методов клиентской стороны – в параметре `AuthClient`.

```
UserManager = Srp
AuthServer = Srp
AuthClient = Srp
```

Для аутентификации в режиме SRP необходимо предъявить имя пользователя и пароль:

```
isql localhost:d:\test.fdb -user testuser -password testpass
```

Традиционная (Legacy_Auth) аутентификация

Традиционная аутентификация подразумевает использование пароля в качестве единственного фактора аутентификации. Пароль может передаваться серверу в виде хэша или в открытом виде. Шифрование пароля происходит по алгоритму `LEGACY`.

Если вы собираетесь использовать данный метод аутентификации, в файле конфигурации `firebird.conf` выставите значения следующих параметров:

```
UserManager = Legacy_UserManager
WireCrypt = Enabled
AuthServer = Legacy_Auth, Srp, Win_Sspi
AuthClient = Legacy_Auth, Srp, Win_Sspi
```

Плагин, отвечающий за `Legacy_Auth` аутентификацию, не предоставляет ключа шифрования трафика. Поэтому следует отключить обязательное (`Required`) шифрование сессий через параметр `WireCrypt` в конфигурации сервера, т.е. выставить значение `Enabled` или `Disabled`.

Для аутентификации в традиционном режиме необходимо предъявить имя пользователя и пароль, например:

```
isql localhost:d:\test.fdb -user testuser -password testpass
```

В этом методе аутентификации учитывается только первые 8 символов любого пароля.

Для того, чтобы пользователь Ред База Данных смог пройти традиционную аутентификацию, он должен быть предварительно создан с помощью плагина управления пользователями `Legacy_UserManager`:

```
CREATE USER test PASSWORD 'test'
USING PLUGIN Legacy_UserManager;
```

Данные о пользователях, созданных в традиционном режиме, хранятся в базе данных безопасности (`security3.fdb`) в таблице `PLG$USERS` (см. Приложение Д).

Доверительная (Win_Sspi) аутентификация

В доверительном режиме аутентификации используется система безопасности операционной системы. В операционных системах семейства Windows NT можно пользоваться учётными записями ОС. Для этого необходимо, чтобы в файле конфигурации `firebird.conf` параметр `AuthServer` в списке значений содержал метод аутентификации `Win_Sspi`. Кроме того, этот метод должен присутствовать и в списке методов клиентской стороны – в параметре `AuthClient`. Также для использования доверительной аутентификации следует отключить обязательное (`Required`) шифрование соединений (параметр `WireCrypt`), поскольку плагин, реализующий `Win_Sspi`, не предоставляет ключ шифрования. Для этого достаточно выставить значение `Enabled` или `Disabled`.

```
AuthServer = Win_Sspi, Srp
AuthClient = Win_Sspi, Srp
WireCrypt = Enabled
```

До Ред Базы Данных 3.0 при включенной доверительной аутентификации, пользователи прошедшие проверку по умолчанию автоматически отображались в `CURRENT_USER`. В версии 3.0 отображение должно быть сделано явно для систем с несколькими базами данных безопасности и включенной доверительной аутентификацией.

Отображение для включения использования доверительной аутентификации Windows во всех базах данных, которые используют текущую базу данных безопасности:

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

В этом режиме при подключении к серверу Ред База Данных не требуется предъявлять имя пользователя и пароль. Если пользователь локального компьютера подключается к серверу, работающему на том же компьютере, то он получает роль `PUBLIC`.

При сетевом соединении происходит проверка принадлежности пользователя домену, в состав которого входит компьютер с работающим сервером БД. Если пользователь не является доменным, то он не имеет прав на подключение к серверу.

Для того, чтобы узнать, с каким именем пользователя и паролем вы подключились в режиме доверительной аутентификации, можно выполнить следующий запрос:

```
select CURRENT_USER from rdb$database;
=======================================
domain\administrator.
```

То есть подключился пользователь с именем `administrator`, который является членом домена `domain`.

Чтобы при подключении доверенного пользователя не указывать никакой дополнительной информации о роли, существует оператор `SET TRUSTED ROLE`, который включает доступ доверенной роли.

Администраторы операционной системы Windows автоматически не получают права `SYSDBA` при подключении к базе данных. Имеют ли администраторы автоматические права `SYSDBA` зависит от установки значения флага `AUTO ADMIN MAPPING`. После успешного "auto admin"подключения текущей ролью будет являться `RDB$ADMIN`.

Оператор `ALTER ROLE` разрешает или запрещает автоматическое предоставление роли `RDB$ADMIN` администраторам Windows в текущей базе данных, если используется доверительная авторизация. По умолчанию автоматическое предоставление роли `RDB$ADMIN` отключено.

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

Оператор `ALTER ROLE` является упрощённым видом оператора создания отображения предопределённой группы `DOMAIN_ANY_RID_ADMINS` на роль `RDB$ADMIN`.

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

В обычных базах данных статус `AUTO ADMIN MAPPING` проверяется только во время подключения. Если Администратор имеет роль `RDB$ADMIN` потому, что произошло автоматическое отображение во время входа, то он будет удерживать эту роль на протяжении всей сессии, даже если он или кто-то другой в это же время выключает автоматическое отображение. Точно также, включение `AUTO ADMIN MAPPING` не изменит текущую роль в `RDB$ADMIN` для администраторов, которые уже подключились.

Рекомендуется явно указывать на то, что ожидается доверительная аутентификация, используя для этого ключ `–tr`. Например, если в системе установлены соответствующие значения переменных окружения `ISC_USER` и `ISC_PASSWORD`, и не будет указан ключ `–tr` при аутентификации, то вместо контекста безопасности пользователя на сервер будут переданы имя пользователя и пароль, соответствующие переменным окружения, как при традиционной аутентификации, что приведет к ошибке, так как на сервере ожидается доверительная аутентификация.

При доверительной аутентификации права на доступ и операции над объектами баз данных могут быть назначены пользователю операционной системы, как обычному пользователю Ред База Данных.

При доверительной аутентификации возможен следующий вариант соединение с БД:

```
isql localhost:d:\test.fdb
```

Многофакторная (Multifactor) аутентификация

В Ред База Данных 3.0 реализована возможность использования многофакторной аутентификации. Требуемые для аутентификации пользователя факторы определяются политикой безопасности, назначенной пользователю. Подробно политики безопасности рассмотрены в п. 9.5.

Для использования только режима многофакторной аутентификации следует в файле конфигурации `firebird.conf` добавить в список значений параметра `AuthServer` метод аутентификации `Multifactor`. Кроме того, этот метод должен присутствовать и в списке методов клиентской стороны – в параметре `AuthClient`. Также в конфигурационном файле сервера и клиента необходимо указать используемый криптоплагин `CryptoPlugin = Crypto_API` (по умолчанию). Как уже было сказано, для многофакторной аутентификации необходимо наличие криптопровайдера КриптоПро CSP.

```
AuthServer = Multifactor, Srp
AuthClient = Multifactor, Srp
UserManager = Multifactor_Manager
WireCrypt = Enabled
CryptoPlugin = Crypto_API
```

Для того, чтобы пользователь Ред База Данных смог пройти многофакторную аутентификацию, он должен быть предварительно создан с помощью плагина управления пользователями

Multifactor_Manager:

```
CREATE USER test PASSWORD 'test'
USING PLUGIN Multifactor_Manager;
```

Для того, чтобы сделать SYSDBA многофакторным, необходимо создать нового пользователя с именем SYSDBA, используя плагин Multifactor_Manager.

Данные о многофакторных пользователях хранятся в базе данных безопасности (security3.fdb) в таблице PLG$MF (см. Приложение Д). Там хранится не только хэш пароля, но и название алгоритма, с помощью которого этот хэш получен.

Доступ к базе данных в режиме многофакторной аутентификации определяется политикой безопасности, в которой определены факторы, обязательные для предъявления пользователем при прохождения процедуры аутентификации. При аутентификации все данные пользователя, кроме имени, передаются только в зашифрованном виде. В настоящее время используются следующие факторы: пароль и сертификат.

Пример:

Если политика безопасности пользователя требует аутентификации по факторам «пароль» и «сертификат», то строка подключения будет иметь следующий вид:

```
isql <имя_БД> -user <имя_домена\имя_пользователя> -password <пароль_пользователя>
-certificate <алиас_сертификата> -pin <пароль_закрытого_ключа>
```

Здесь:

- параметр –user идентифицирует пользователя;
- параметр –password аутентифицирует пользователя по фактору «пароль»;
- параметр –certificate аутентифицирует пользователя по фактору «сертификат», то есть задает сертификат пользователя с помощью алиаса. Алиас представляет собой строку следующего вида: "SubjectCN,IssuerCN,SerialNumber", где SubjectCN – имя владельца сертификата, IssuerCN – название издателя сертификата, SerialNumber – серийный номер сертификата в шестнадцатеричном виде.
- параметр –pin задаёт пароль для закрытого ключа сертификата, если он необходим.

Контейнер с набором ключей и сертификат создаются заранее.

Таблица 9.6 — Параметры конфигурационного файла, имеющие отношение к механизму многофакторной аутентификации

Параметр	Возможное значение	Комментарий
AuthServer, AuthClient	Srp, Win_Sspi, Legacy_Auth, Gss, Multifactor	Параметр AuthServer - набор методов аутентификации, разрешенных на сервере. Параметр AuthClient - набор методов аутентификации, поддерживаемых клиентом.
UserManager	Srp, Legacy_UserManager, Multifactor_Manager	Плагин для управления пользователями в базе данных безопасности
CryptoPlugin	Crypto_API	Имя библиотеки криптопровайдера (расположена в каталоге plugins сервера)
ServerCertificate	<алиас сертификата>	Задаёт алиас сертификата, которым сервер будет удостоверять свою подлинность клиенту.

Параметр	Возможное значение	Комментарий
CertUsernameDN	<атрибут>	Атрибут сертификата, из которого будет извлекаться имя его владельца. По умолчанию CN.
CertUsernamePattern	<регулярное выражение>	Регулярное выражение, применяемое к атрибуту сертификата с именем пользователя для извлечения самого этого имени. Использует синтаксис SQL, по умолчанию не задано.
CertVerifyChain	0 | 1	Задаёт / отключает проверку цепочки сертификации пользовательского сертификата.
TrustedCertificate	<алиас>	Задаёт алиас сертификата, которому сервер будет доверять. Если пользователь предъявляет этот сертификат, он будет аутентифицирован с указанным именем без пароля и без проверки его сертификата.

Политики безопасности

Общие сведения

Политики безопасности (политики учетных записей) позволяют контролировать следующие параметры безопасности системы:

- сложность пароля при его задании;
- количество предыдущих паролей, которые не должен повторять вновь заданный;
- срок действия пароля;
- количество допустимых неудачных попыток аутентификации;
- период времени неиспользования учетных записей пользователей;
- требования к дополнительным факторам для прохождения аутентификации (цифровые сертификаты).

Политики учетных записей создаются и хранятся в базе данных безопасности security3.fdb в таблице PLG$POLICIES (см. Приложение Д).

Политика должна определять реакцию системы на неудачные попытки входа в систему и блокировать пользователя временно или постоянно. Это позволяет управлять сложностью пароля и общей защищенностью базы данных. Также политика не позволяет производить подбор пароля и блокирует пользователя, от имени которого может производится атака сервера.

> Политики безопасности применимы только для многофакторных пользователей при многофакторном подключении к БД.

Создание политик безопасности

Для создания, изменения или удаления политики безопасности администратору необходимо соединиться с какой-либо базой данных. Для создание политики используется оператор CREATE POLICY. Синтаксис этого оператора приведен ниже:

```
CREATE POLICY <имя политики> [AS <параметр>=<значение> [,<параметр>=<значение>...]]
```

Хотя сами политики хранятся в базе данных безопасности security3.fdb, создать их можно, соединившись с любой базой данных. Однако пользователь при этом должен иметь права на запись в базу данных безопасности security3.fdb.

Возможные параметры политик следующие:

- `AUTH_FACTORS` — факторы аутентификации (представлены в таблице 9.8);
- `PSWD_NEED_CHAR` — минимальное количество букв в пароле;
- `PSWD_NEED_DIGIT` — минимальное количество цифр в пароле;
- `PSWD_NEED_DIFF_CASE` — требование использования различных регистров букв в пароле;
- `PSWD_MIN_LEN` — минимальная длина пароля;
- `PSWD_VALID_DAYS` — срок действия пароля;
- `PSWD_UNIQUE_COUNT` — количество последних не повторяющихся паролей;
- `MAX_FAILED_COUNT` — количество неудачных попыток входа;
- `MAX_UNUSED_DAYS` — максимальное время неактивности учетных записей пользователя, в днях.

Таблица 9.7 — Символическое обозначение факторов аутентификации

Символическое обозначение	Расшифровка
CERT_X509	Сертификат пользователя
PASSWORD	Пароль

Следующий пример демонстрирует создание политики:

```
CREATE POLICY TestPolicy AS
    AUTH_FACTORS = (CERT_X509, PASSWORD),
    PSWD_NEED_CHAR = 5,
    PSWD_NEED_DIGIT = 3,
    PSWD_MIN_LEN = 8,
    PSWD_NEED_DIFF_CASE = true,
    PSWD_VALID_DAYS = 15,
    PSWD_UNIQUE_COUNT = 5,
    MAX_FAILED_COUNT = 5,
    MAX_UNUSED_DAYS = 45;
```

Назначение политики пользователям

Для того, чтобы назначить созданную политику пользователю, необходимо выполнить следующую команду:

```
GRANT POLICY <имя_политики> TO <имя_пользователя>;
```

После назначения пользователям политик, касающихся требований к паролям, администратор должен сменить этим пользователям пароли, чтобы они удовлетворяли требованиям сопоставленных этим пользователям политик безопасности.

Политики назначаются только пользователям, но не ролям. Назначить политику несуществующему пользователю нельзя. У пользователя может быть только одна политика. Таким образом, чтобы отменить предыдущую политику и назначить новую, нужно просто еще раз выполнить оператор `GRANT POLICY <новая_политика>`.

В случае, если при смене пароля пользователя будет введен неправильный старый пароль пользователя, будет выдано сообщение:

```
Your user name and password are not defined. Ask your database administrator to
set up a Firebird login. unable to open database
```

Если при смене пароля пользователь введет пароль, который не удовлетворяет установленной для пользователя политике безопасности, то будет выдано сообщение:

```
Statement failed, SQLSTATE = HY000
modify record error
```

При попытке соединиться с базой данных с некорректными учетными данными пользователя пользователю будет отказано. Здесь можно выделить следующие случаи:

- соединение с предъявлением логина несуществующего пользователя;
- соединение с предъявлением неправильного пароля для существующего пользователя.

В обоих случаях пользователь получит следующее сообщение об ошибке:

```
Statement failed, SQLCODE = -902
Your user name and password are not defined. Ask your database administrator to
set up a Firebird login.
```

Когда достигается значение `MAX_FAILED_COUNT` при неудачных попытках соединиться с базой данных, пользователь блокируется. Для его разблокировки следует подключиться к базе безопасности `security3.fdb` и установить значение столбца `PLG$FAILED_COUNT` (см. таблицу `PLG$MF`), выполнив запрос:

```
UPDATE PLG$MF SET PLG$FAILED_COUNT=<значение> WHERE PLG$USER_NAME = '<имя_польз-я>'
```

При установке значения параметра в 0 пользователь блокироваться не будет.

Если пользователь при прохождении аутентификации предъявил все требуемые политикой безопасности факторы аутентификации, при этом эти факторы удовлетворяют ограничениям политики для этого пользователя и позволяют однозначно идентифицировать пользователя, то аутентификация субъекта доступа считается успешной.

Вновь созданным пользователям соответствует политика безопасности по умолчанию – `DEFAULT`. В ней отсутствуют какие-либо требования к паролям или сессиям пользователей. То есть для того, чтобы отменить требования политики для определенного пользователя, ему необходимо назначить политику по умолчанию:

```
GRANT POLICY "DEFAULT"TO <имя_пользователя>
```

Таким образом, использование политик позволяет повысить общую безопасность системы, а именно:

- запретить пользователям использовать слишком простые пароли;
- требовать от пользователей регулярной смены паролей;
- ограничить число неудачных попыток аутентификации, что в совокупности с требованиями к сложности и сроку действия паролей исключает подбор пароля злоумышленником.

Кроме того, в случае, если пользователь не прошел процедуру аутентификации, он не получит информации о том, какой именно из предъявленных им факторов является неправильным.

Аутентификация доверенным пользователем

В `firebird.conf` добавлен параметр конфигурации `TrustedUser` для аутентификации по паролю с именем другого пользователя. По умолчанию пустой.

При соединении с базой данных кроме имени пользователя (`isc_dpb_user_name`) можно указать эффективный логин (`isc_dpb_effective_login`). В `isql` для этого добавлен ключ `-l`.

Если задан эффективный логин, то после успешной аутентификации любого плагина проверяется задан ли параметр `TrustedUser`:

- Если не задан - ошибка аутентификации: попытка подмены логина с отключенной опцией.
- Если параметр задан, но логин пользователя не совпадает с доверенным - тоже ошибка: попытка подмены логина не доверенным пользователем.
- Если параметр задан и логин пользователя совпадает с доверенным, то при подключении к базе данных его имя заменяется на указанный им эффективный логин.

Для ядра СУБД это подключение будет выглядеть как обычное, информация о подмене логина до него не доходит.

Работает для всех плагинов аутентификации.

Доверенная аутентификация через механизм GSSAPI (Gss)

Доверенная аутентификация через механизм GSSAPI доступна только в промышленной редакции СУБД Ред База Данных. Подробнее различия функционала редакций описаны в разделе "Редакции СУБД Ред База Данных 3.0".

Ред База Данных 3.0 поддерживает Kerberos аутентификацию через GSSAPI (Generic Security Services Application Programming Interface). GSSAPI – это абстрактный уровень над Kerberos 5, предназначенный для решения проблемы несовместимости схожих сервисов безопасности. GSSAPI обеспечивает автоматическую аутентификацию (single sign-on), для систем, которые её поддерживают.

Принцип работы:

- Для аутентификации клиента используется контекст безопасности, полученный при входе в систему или из других источников типа kinit.
- Из контекста безопасности клиент извлекает токен, который отправляется на сервер СУБД.
- Сервер СУБД проверяет токен через GSSAPI (по умолчанию для проверки используется механизм Kerberos).
- Если токен проходит проверку, сервер извлекает из него имя пользователя, которое будет использовано для подключения к базе.

К достоинствам данной аутентификации можно отнести преимущества технологии единого входа (SSO):

- ввод пользователем своих учётных данных только один раз;
- отсутствие ввода пароля;
- безопасность;
- централизованное хранение учётных данных пользователей.

При работе с Kerberos GSSAPI использует стандартные учётные записи в формате `servicename/hostname@realm`.

Имя пользователя при такой аутентификации формируется из имени учетной записи в базе данных `KDC`.

Такой вид аутентификации работает только при передаче данных через сетевой протокол.

Если пользователь указал логин, то данный метод аутентификации будет проигнорирован сервером.

Для работы данного метода аутентификации необходимо, чтобы в файле конфигурации `firebird.conf` параметр `AuthServer` в списке значений содержал метод аутентификации `Gss`. Кроме того, этот метод должен присутствовать и в списке методов клиентской стороны – в параметре `AuthClient`. Так как плагин, отвечающий за `GSSAPI` аутентификацию, не предоставляет ключа шифрования трафика, следует отключить обязательное (`Required`) шифрование сессий через параметр `WireCrypt` в конфигурации сервера, т.е. выставить значение `Enabled` или `Disabled`.

Если вы собираетесь использовать данный метод аутентификации, в файле конфигурации `firebird.conf` выставите значения следующих параметров:

```
AuthServer = Gss, Srp
AuthClient = Gss, Srp
WireCrypt = Disabled
KrbServerKeyfile =
GssServiceName =
GssHostName =
GSSLibrary =
```

где

- `KrbServerKeyfile` — путь до файла, содержащий долговременный ключ сервиса, который будет использовать СУБД для аутентификации в Kerberos. Этот ключевой файл создаётся в центре распределения ключей (KDC), например командой утилитой в Active Directory;
- `GssServiceName` — имя сервиса (по умолчанию «`rdb_server`»), созданное на сервере Kerberos для аутентификации СУБД;
- `GssHostName` — DNS-адрес сервера СУБД (например «`rdb.example.com`»);
- `GSSLibrary` — динамическая библиотека GSSAPI (`libgssapi_krb5.so`). Поддерживается также библиотека `libvas-gssapi.so` от One Identity Authentication Services. При её использовании СУБД после аутентификации определяет группы, назначенные пользователю в домене, и назначает ему одноимённые роли, существующие в базе данных.

При доверенной аутентификации возможен следующий вариант соединение с БД:

```
isql localhost:d:\test.fdb
```

Доверенная аутентификации для выполнения Execute Statement On External без указания логина и пароля (ExtAuth)

Для выполнения оператора `EXECUTE STATEMENT ON EXTERNAL`, если внешний источник данных находится на другом сервере, предложения `AS USER <имя пользователя>` и `PASSWORD <пароль>` являются обязательными.

```
EXECUTE STATEMENT 'SELECT * FROM RDB$DATABASE'
ON EXTERNAL 'server:db1' AS USER 'MYUSER' PASSWORD 'mypassword'
```

Значения имени пользователя и пароля передаются в открытой форме, что небезопасно. Например, если ESOE (сокр. от `EXECUTE STATEMENT ON EXTERNAL`) вызывается из кода хранимой процедуры, подключенные пользователи могут видеть пароль.

Для безопасного подключения в Ред Базе Данных был разработан плагин аутентификации `ExtAuth` специально для ESOE, который устанавливает доверительную связь между серверами Ред Базы Данных и выполняет аутентификацию ESOE без логина и пароля:

```
EXECUTE STATEMENT 'SELECT * FROM RDB$DATABASE'
ON EXTERNAL 'server:db1';
```

Для использования данной возможности следует в файле конфигурации `firebird.conf` добавить в список значений параметров `AuthServer` и `AuthClient` метод аутентификации `ExtAuth` на всех серверах, которые будут "доверять"друг другу.

```
AuthServer = Srp, ExtAuth
AuthClient = Srp, ExtAuth
```

Затем необходимо сгенерировать ключевой файл для плагина. Этот ключ должен быть размещен на всех серверах Ред Базы Данных, которые должны доверять друг другу.

Чтобы сгенерировать ключевой файл `ExtAuth.conf`, запустите исполняемый файл `extauth_keygen`. Ключевой файл содержит три параметра:

- `Key` – сам ключ;
- `IgnoreLogin` – игнорировать явное указание логина в операторе ESOE (по умолчанию логин не игнорируется). Если значение параметра `No` и указан логин, то плагин `ExtAuth` перестает действовать.
- `IgnorePassword` – игнорировать явное указание пароля в операторе ESOE (по умолчанию пароль не игнорируется). Если значение параметра `No` и указан пароль, то плагин `ExtAuth` перестает действовать.

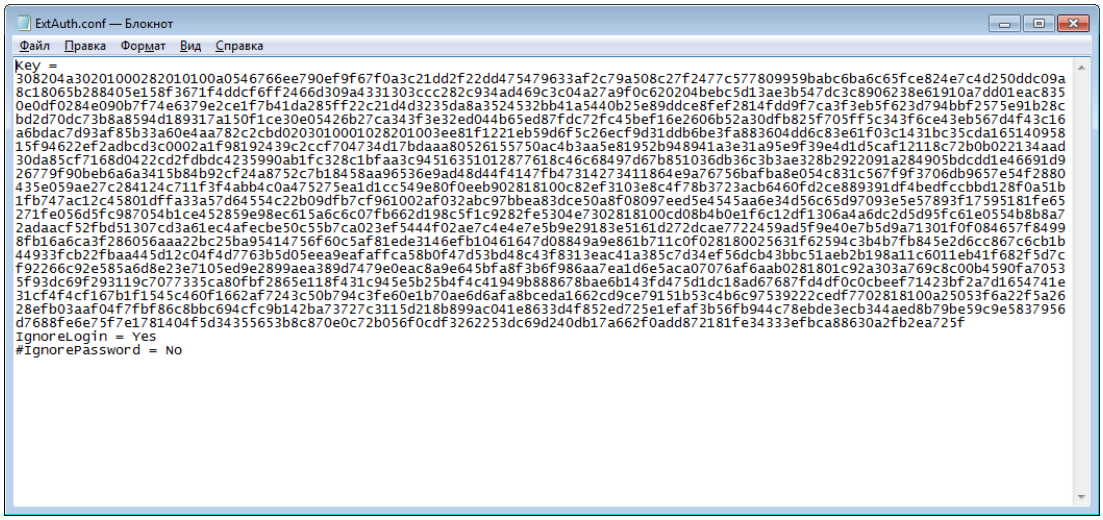


Рисунок 9.1 — Содержимое файла `ExtAuth.conf`

Потом скопируйте файл ключа на все доверенные сервера Ред Базы Данных в папку `plugins`. Ключ, созданный на Windows, можно использовать в Linux, и наоборот.

> Другое имя ключевого файла и его расположение не допускается.

Чтобы использовать плагин аутентификации внутри конкретной базы данных, необходимо создать отображение между пользователями плагина `ExtAuth` и обычными пользователями Ред Базы Данных.

Например, чтобы запустить ESOE от имени пользователя `MYUSER`:

```
EXECUTE STATEMENT 'SELECT * FROM RDB$DATABASE'
ON EXTERNAL 'server:db1';
```

нужно сопоставить пользователя `MYUSER` с фактическим пользователем в целевой базе данных `db1`. Давайте предположим, что есть пользователь `MYUSER2` в целевой базе данных, в этом случае нужно создать следующее отображение в целевой базе данных `db1`:

```
CREATE MAPPING cluster_auth USING PLUGIN extauth
FROM USER MYUSER TO user MYUSER2;
```

Оба пользователя должны существовать.
Можно создать отображение между пользователями для всех баз данных на сервере.

Аутентификация по протоколу LDAP

Аутентификация по протоколу LDAP доступна в стандартной редакции СУБД Ред База Данных. Подробнее различия функционала редакций СУБД Ред База Данных описаны в разделе "Редакции СУБД Ред База Данных 3.0".

Ред База Данных поддерживает возможность аутентификации пользователей сервера баз данных с использованием службы каталогов через протокол LDAP. Как известно, учётные данные пользователей хранятся в БД безопасности – security3.fdb. Также существует возможность добавить дополнительный источник учётной информации – службу каталогов на основе сервера OpenLDAP и Active Directory. При этом LDAP используется именно как дополнение к традиционной схеме безопасности сервера. При проверке пользовательских учётных данных, если заданы параметры подключения к LDAP, сервер для всех пользователей, кроме SYSDBA, сначала пытается проверить наличие учетной записи пользователя в каталоге LDAP и если он там не найден, то выполняется также поиск в БД безопасности security3.fdb. Для SYSDBA поиск выполняется только в БД безопасности.

Если проверка в БД security3.fdb прошла неуспешно и не задан адрес LDAP-сервера или пользователь не найден в LDAP-сервере, выдается сообщение об ошибке аутентификации.

С точки зрения конечного пользователя всё это работает совершенно прозрачно и ему не нужно выполнять никаких дополнительных действий.

Параметры конфигурации LDAP

Для аутентификации по протоколу LDAP в firebird.conf должны быть заданы настройки подключения к серверу каталогов в параметрах вида «LDAP...».

Параметры вида «LDAP...» описаны в разделе Настройки LDAP 6.7.

Существует два основных метода аутентификации по протоколу LDAP: традиционный метод и метод bind.

Традиционный метод аутентификации по протоколу LDAP

При использовании этого метода LDAP используется как хранилище данных, аналогичное БД безопасности.

Для аутентификации с помощью этого метода в LDAP должны быть созданы атрибуты, в которых СУБД будет хранить нужную информацию. Атрибуты описаны в приложении Г.

Необходимо добавить к конфигурации LDAP схему, аналогичную приведенной в приложении Г. Пользователям должен быть назначен класс rdbAuth.

Пароли пользователя хранятся в атрибутах rdbPassword - для метода Legacy_Auth, rdbSrpVerifier - для метода Srp и rdbSecurePassword - для метода Multifactor. Это пароли для аутентификации в СУБД, они могут отличаться от пароля пользователя в LDAP (userPassword).

Если настроен параметр LDAPPasswordSync и пароль меняется средствами СУБД, то можно синхронизировать пароли для аутентификации в СУБД с паролями пользователя в LDAP. Если пароль пользователя в LDAP изменяется сторонними утилитами, то значения в атрибутах rdbPassword, rdbSrpVerifier и rdbSecurePassword не будут синхронизированы.

Для определения списка групп, к которым принадлежит пользователь в LDAP, могут использоваться различные схемы. Используемая схема определяется настройкой сервера LDAP/ AD. Указать

конкретную схему можно в параметре LDAPMembershipFilter. В нём в качестве имени предполагаемого пользователя указывается шаблон %u, а в качестве DN пользователя указывается %d. Три основные схемы определения групп:

- LDAPMembershipFilter = memberUid=%u;
- LDAPMembershipFilter = member=uid=%u,ou=users,dc=example,dc=com;
- LDAPMembershipFilter = member=%d.

При аутентификации данном способом клиент передает имя пользователя и хэш пароля серверу СУБД. Пользователь будет аутентифицироваться по информации из его атрибутов в LDAP. Сервер ищет заданного пользователя в LDAP, считывает хэш его пароля из атрибута, соответствующего методу аутентификации и сверяет хэш в LDAP с хэшем, переданным пользователем.

Пример настройки для OpenLDAP:

```
# Адрес сервера LDAP
LDAPServer=10.81.1.1
# DN служебной учетной записи, от имени которой СУБД будет подключаться к LDAP
LDAPUserDN=uid=rdb,ou=users,dc=example,dc=com
# Пароль от служебной учетной записи
LDAPPassword=12345
# Атрибут LDAP, по которому будет выполняться поиск пользователя, имя которого
передает клиент
LDAPUserPrefix=uid
# Фильтр для поиска пользователя в каталоге LDAP
LDAPUserFilter = uid=%u
# Стартовая ветка для поиска пользователя (поиск выполняется рекурсивно по всем
вложенным веткам)
LDAPUserBase=ou=users,dc=example,dc=com
# Стартовая ветка для поиска групп пользователя (поиск выполняется рекурсивно по
всем вложенным веткам)
LDAPGroupBase=ou=roles,dc=example,dc=com
# Фильтр для поиска групп пользователя
LDAPMembershipFilter=memberUid=%u
```

Метод bind

Этот метод используется, если требуется аутентифицировать пользователя в СУБД его доменным паролем, при этом не создавая в LDAP дополнительные атрибуты. Такой способ больше подходит для аутентификации в ActiveDirectory.

При использовании этого метода пользователь, при подключении к БД, указывает доменный пароль, который клиент передает серверу в открытом виде, чтобы СУБД могла применить функцию bind к LDAP от имени данного пользователя.

Пользователь будет аутентифицироваться в LDAP по его реальному логину методом bind одним из следующих способов:

- Если задан параметр LDAPUserDN, то сначала сервер будет подключаться к LDAP от этого общего пользователя. Затем сервер будет искать DN реального пользователя и отключится от LDAP. Далее сервер делает bind с полным именем (DN) и паролем реального пользователя;
- Если параметр LDAPUserDN не задан, то сервер сразу сделает bind с именем и паролем реального пользователя;
  - Если задан параметр LDAPUserBase, то DN пользователя формируется, как LDAPUserPrefix + <имя пользователя> + LDAPUserBase. В этом случае пользователи должны находиться в одной ветке LDAP, указанной в LDAPUserBase;

  - Если параметр LDAPUserBase не задан, то сервер сначала попытается сделать bind с именем пользователя в том виде, в котором его передал клиент.

- Если аутентифицироваться не получилось и задан параметр LDAPDomain, то будет выполнена попытка сделать bind используя UPN (UserPrincipalName) в виде <имя пользователя>@<значение LDAPDomain>. Такой способ работает для входа в ActiveDirectory.

Пример настройки для ActiveDirectory:

```
# Адрес сервера LDAP
LDAPServer=10.81.1.1
# Шифрование по SSL
LDAPEncryption=SSL
# Атрибут LDAP, по которому будет выполняться поиск пользователя, имя которого
передает клиент
LDAPUserPrefix=sAMAccountName
# Фильтр для поиска пользователя в каталоге LDAP
LDAPUserFilter=&(objectClass=user)(sAMAccountName=%u)
# Стартовая ветка для поиска групп пользователя (поиск выполняется рекурсивно по
всем вложенным веткам)
LDAPGroupBase=ou=groups,dc=example,dc=com
# Фильтр для поиска групп пользователя
LDAPMembershipFilter=member=%d
# Имя домена
LDAPDomain = example.com
# Отключение проверки сертификата сервера LDAP
VerifyLdapServer=0
```

Если пользователь не найден в LDAP, сервер пытается проверить пользователя через security3.fdb. Если он там найден, аутентификация выполняется по считанным из неё данным. Если не найден или найден, но его пароль не совпадает, то сообщается об ошибке аутентификации.

Для SYSDBA поиск выполняется только в БД безопасности.

Процесс аутентификации при передаче пароля в зашифрованном виде

Пользователь при подключении к БД передаёт пароль в *зашифрованном виде* в тэге isc_dpb_password_enc. В файле конфигурации задан параметр LDAPServer.

Сервер для всех пользователей, кроме SYSDBA, сначала пытается проверить наличие учетной записи пользователя в каталоге LDAP. Сначала выполняется подключение к LDAP-серверу от имени общего пользователя LDAPUserDN (указывается обязательно). Затем на сервере происходит поиск нужного пользователя относительно указанной ветви LDAPUserBase. Если пользователь найден, то сверяются хэш его пароля в LDAP с хэшем предоставленного им пароля.

Пароль пользователя для Legacy/Multifactor/Srp-аутентификации хранится в LDAP в атрибуте «rdbPassword»/«rdbSecurePassword»/«rdbSrpVerifier» в том же виде, в котором он сохраняется в security3.fdb.

Если пользователь не найден в LDAP, сервер пытается проверить пользователя через security3.fdb. Если он там найден, аутентификация выполняется по считанным из неё данным.

Если пользователь не найден в LDAP и не найден в security3.fdb, проверка пароля считается неуспешной.

Процесс аутентификации по сертификату

При многофакторной аутентификации по сертификату сначала проверяется является ли пользовательский сертификат доверенным. Для этого он сравнивается с сертификатом, заданным в па-

раметре конфигурации TrustedCertificate. При аутентификации по доверенному сертификату наличие пользователя в security3.fdb не требуется.

Для успешной аутентификации по доверенному сертификату необходимо, чтобы в security3.fdb присутствовали служебные таблицы для многофакторной аутентификации Для этого должен быть создан хотя бы один многофакторный пользователь.

Если сертификат не доверенный, то он верифицируется и после этого проверяется, задан ли параметр конфигурации LDAPUserCertificate. Если он не задан, проверка сертификата считается успешной. Если параметр задан, то выполняется подключение к LDAP-серверу и скачивание оттуда сертификата пользователя. Затем полученный сертификат сравнивается с сертификатом, предоставленным пользователем. Если они одинаковы, проверка сертификата считается успешной. Если не удалось подключиться к LDAP, получить оттуда сертификат пользователя или сертификаты не совпадают, проверка сертификата считается неуспешной.

Если проверка сертификата через LDAP прошла успешно, имя пользователя, извлечённое из сертификата, может отличаться от имён пользователей, которые заданы другими факторами аутентификации.

Имя пользователя из сертификата (из секции Subject) извлекается с помощью двух параметров конфигурации.

- CertUsernameDN – содержит DN атрибута пользователя внутри сертификата. По умолчанию используется атрибут CN.
- CertUsernamePattern – задаёт регулярное выражение в синтаксисе SQL, которое извлекает подстроку из содержимого найденного атрибута. По умолчанию используется пустой шаблон, что означает использование содержимого атрибута целиком.

Если указанный атрибут не найден в сертификате или результатом применения к нему регулярного выражения оказалась пустая строка, возвращается ошибка.

Верификация сертификата пользователя выполняется в два этапа. На первом этапе проверяется наличие у пользователя доступа к закрытому ключу, соответствующему предъявленному сертификату. Это выполняется генерацией сессионного ключа по алгоритму Диффи-Хеллмана, в которой участвуют пары открытых и закрытых ключей клиента и сервера. На втором этапе выполняется проверка самого сертификата и его цепочки сертификации. Верификация может быть отключена параметром конфигурации «VerifyLdapServer» (по умолчанию – включен). Верификация считается неуспешной в следующих случаях:

- не удалось построить цепочку сертификации;
- любой сертификат из цепочки отозван;
- любой сертификат из цепочки просрочен;
- любой сертификат из цепочки не прошёл проверку подписи;
- любой сертификат из цепочки используется не по назначению;
- цепочка основана на недоверенном корневом центре сертификации;
- цепочка сертификации содержит цикл;
- цепочка сертификации построена не полностью;
- не удалось проверить статус отзыва для любого сертификата из цепочки.

При неудачной верификации пользователю сообщается об ошибке проверки фактора аутентификации, а в firebird.log записывается сообщение с информацией о владельце сертификата и типе произошедшей ошибки.

Информация о пользователях, получаемая из LDAP

После аутентификации через LDAP можно запросить информацию о пользователе с помощью функций LDAP_ATTR:

```
LDAP_ATTR(<имя атрибута> [, <имя пользователя> ])
```

Важно учитывать, что получить атрибуты пользователя, указанного в необязательном аргументе функции LDAP_ATTR, может только SYSDBA, владелец базы данных, пользователь с ролью RDB$ADMIN, пользователь, принадлежащий группе LDAP_ADMIN каталога LDAP или пользователь, имя которого указано в качестве аргумента.

Можно узнать где пользователь прошел проверку подлинности: через базу данных безопасности или в LDAP:

```
select RDB$GET_CONTEXT('AUTHDATA', 'AUTH_TYPE') from rdb$database;
```

и с помощью какого плагина аутентификации:

```
select RDB$GET_CONTEXT('AUTHDATA', 'AUTH_PLUGIN') from rdb$database;
```

Также можно узнать ФИО пользователя. При аутентификации через LDAP эта информация считывается из атрибута пользователя "CN":

```
select RDB$GET_CONTEXT('AUTHDATA', 'USER_FIRST_NAME') from rdb$database;
select RDB$GET_CONTEXT('AUTHDATA', 'USER_MIDDLE_NAME') from rdb$database;
select RDB$GET_CONTEXT('AUTHDATA', 'USER_LAST_NAME') from rdb$database;
```

Если заданы параметры LDAPGroupBase и LDAPMembershipFilter, то осуществляется поиск групп, к которым принадлежит пользователь в LDAP. После этого ему назначаются роли, соответствующие найденным группам пользователя из LDAP. Заполняются переменные LDAP_ROLES и LDAP_ROLES_DN – список ролей пользователя и DN ролей пользователя, полученных из каталога LDAP.

Список всех групп, которым принадлежит пользователь, можно получить с помощью функции LDAP_USER_GROUPS:

```
LDAP_USER_GROUPS(<имя пользователя> [, <фильтр списка групп> ])
```

Для получения полного списка групп пользователей каталога LDAP можно использовать функцию LDAP_GROUPS:

```
LDAP_GROUPS([ <фильтр списка групп> ])
```

Получение полного списка групп и списка групп для конкретного пользователя доступно только для SYSDBA, владельца базы данных, пользователю с ролью RDB$ADMIN или пользователю, принадлежащему группе LDAP_ADMIN каталога LDAP.

Результирующие списки групп можно отфильтровать во время запроса с помощью необязательных аргументов функций LDAP_GROUPS и LDAP_USER_GROUPS:

```
select LDAP_GROUPS('objectClass=posixGroup') from rdb$database;
select LDAP_USER_GROUPS('TEST-USER', 'objectClass=posixGroup') from rdb$database;
```

Изменение пароля в LDAP

Пароль пользователя в LDAP может меняться/задаваться с использованием утилиты GSEC или сервисов. Для этого используется параметр конфигурации LDAPPasswordSync. В нём через «;» указываются пароли, которые необходимо сменить (по умолчанию – все возможные). При изменении пароля указанный пользователь сначала ищется в security3.fdb и, если он там найден, его пароль меняется. Затем пользователь ищется в LDAP (если задан адрес LDAP-сервера). В LDAP меняются

следующие атрибуты:

- `userPassword` – пароль пользователя в Linux (записывается хэш `SHA1` в кодировке `BASE64`);
- `sambaLMPassword` – `LMHash` для `Samba`;
- `sambaNTPassword` – `MD4` хэш для `Samba`;
- `rdbPassword` – пароль пользователя на сервере БД, зашифрованный алгоритмом `LEGACY` , используемым при обычной аутентификации;
- `rdbSecurePassword` – многофакторный пароль пользователя на сервере БД, зашифрованный каким-либо алгоритмом из криптоплагина;
- `rdbPasswordAlgorithm` – алгоритм шифрования многофакторного пароля на сервере БД (в кодировке `UTF-8`);
- `rdbPasswordHistory` – история смены многофакторных паролей;
- `rdbSrpSalt` – соль для аутентификации по протоколу SRP;
- `rdbSrpVerifier` – верификатор пользователя для протокола SRP.

Если меняется пароль у многофакторного пользователя, опция `isc_spb_multi_factor_auth` в сервисах), для формирования `rdbSecurePassword` используется алгоритм хэширования, указанный в firebird.conf (по умолчанию ГОСТ Р 34.11-94).

Если сменить какой-либо пароль не получилось, в `firebird.log` записывается сообщение об ошибке с указанием имени пользователя, атрибута и описания ошибки. Например, пароль не получится сменить, если в схеме `LDAP` нет соответствующего ему атрибута. При ошибке смены одного из паролей выполняется попытка сменить остальные пароли.

При ошибке смены `rdbSecurePassword` два связанных с ним атрибута не меняются.

Если атрибут, соответствующий паролю, у пользователя не задан, но он имеется в схеме `LDAP`, нужный атрибут создаётся.

Если пользователь найден, но хотя бы один из паролей не получилось задать, пользователю возвращается ошибка «`error changing ldap password`».

Если пользователь не найден ни в `security3`, ни в `LDAP`, возвращается ошибка «`record not found for user`».

Схемы LDAP

Так как сервер при работе с `LDAP` использует ряд атрибутов, не входящих в стандартные схемы, то для полноценной работы сервера необходимо добавить к конфигурации `LDAP` схему, аналогичную приведенной в приложении Г.

Чтобы сервер мог использовать эти атрибуты, администратор `LDAP` должен назначить пользователю класс `rdbAuth`. У пользователя, от имени которого сервер выполняет подключение к `LDAP (LDAPUserDN)` должны быть права на чтение всех этих атрибутов и на запись в атрибуты `rdbPassword, rdbSecurePassword, rdbSrpVerifier, rdbSrpSalt, rdbPasswordAlgorithm, rdbPasswordHistory, rdbPolicy, rdbPasswordTime, rdbFailedCount, rdbAccessTime`. При этом сервер сможет создавать нужные атрибуты запросами к `LDAP`.

Таблица 9.8 — Атрибуты пользователя LDAP

Атрибут	Описание
`userPassword`	Пароль пользователя, используемый обычно в `UNIX-системах`.
`sambaLMPassword`	Пароль пользователя, используемый `SAMBA-протоколом`.
`sambaNTPassword`	Пароль пользователя, используемый `SAMBA-протоколом`.
`sambaPwdLastSet`	Время последнего изменения атрибутов `sambaLMPassword` и `sambaNTPassword` в формате `UNIX`.
`rdbPassword`	Пароль для пользователя метода `Legacy_Auth`.

Атрибут	Описание
rdbSrpVerifier	Верификатор пользователя для протокола SRP.
rdbSrpSalt	Соль для аутентификации по протоколу SRP.
rdbSecurePassword	Пароль для пользователя метода Multifactor.
rdbPasswordAlgorithm	Алгоритм шифрования многофакторного пароля на сервере БД (в кодировке UTF-8).
rdbPasswordHistory	История смены многофакторных паролей.
rdbActive	Атрибут показывает активен пользователь или заблокирован. Значение true, если пользователь активен и false, если пользователь заблокирован.
rdbPolicy	Название политики безопасности. Если у пользователя этот атрибут не найден, для него применяется политика безопасности по умолчанию (DEFAULT).
rdbPasswordTime	Время последней смены ГОСТ-пароля пользователем.
rdbFailedCount	Число неудачных попыток аутентификации с момента последнего успешного входа.
rdbAccessTime	Время, до которого вход пользователя на сервер запрещён.

Отображение объектов безопасности

С введением поддержки множества баз данных безопасности в Ред базе данных появились новые проблемы, которые не могли произойти с единой глобальной базой данных безопасности. Кластеры баз данных, использующие одну и ту же базу данных безопасности, были эффективно разделены. Отображения предоставляют средства для достижения той же эффективности, когда множество баз данных используют каждая свою базу данных безопасности. В некоторых случаях требуется управление для ограничения взаимодействия между такими кластерами. Например:

- когда EXECUTE STATEMENT ON EXTERNAL DATA SOURCE требует обмена данными между кластерами;
- когда SYSDBA доступ к базам данных необходим от других кластеров, использующих службы;
- аналогичные проблемы существовали в Ред базе данных 2.5 и 2.6 под Windows, из-за поддержки доверительной аутентификации: два отдельных списка пользователей — один в базе данных безопасности, а другой в Windows, и необходимо связать их.

Единое решение для всех этих случаев является отображение информации о пользователе, входящего в систему, на внутренние объекты безопасности — CURRENT_USER и CURRENT_ROLE.

Создание отображения объекта безопасности выглядит следующим образом:

```
CREATE [GLOBAL] MAPPING <имя отображения>
USING {
    PLUGIN <имя плагина> [IN <имя базы данных>]
  | ANY PLUGIN [IN <имя базы данных> | SERVERWIDE]
  | MAPPING [IN <имя базы данных>]
  | '*' [IN <имя базы данных>] }
FROM { ANY <тип отоб-го объекта> | <тип отоб-го объекта> <имя отоб-го объекта> }
TO { USER | ROLE } [<имя объекта, на которое произведено отображение>]
```

Оператор CREATE MAPPING создаёт отображение объектов безопасности (пользователей, групп, ролей) одного или нескольких плагинов аутентификации на внутренние объекты безопасности – CURRENT_USER и CURRENT_ROLE.

Если присутствует опция GLOBAL, то отображение будет применено не только для текущей базы данных, но и для всех баз данных находящимся в том же кластере, в том числе и базы данных безопасности. Одноименные глобальные и локальные отображение являются разными объектами.

Предложение USING описывает источник отображения. Оно имеет весьма сложный набор опций:

- явное указание имени плагина (опция PLUGIN) означает, что оно будет работать только с этим плагином;
- оно может использовать любой доступный плагин (опция ANY PLUGIN), даже если источник является продуктом предыдущего отображения;
- оно может быть сделано так, чтобы работать только с обще серверными плагинами (опция SERVERWIDE);
- оно может быть сделано так, чтобы работать только с результатами предыдущего отображения (опция MAPPING);
- вы можете опустить использование любого из методов, используя звёздочку (*) в качестве аргумента;
- оно может содержать имя базы данных (опция IN), из которой происходит отображение объекта FROM.

Предложение FROM описывает отображаемый объект. Оно принимает обязательный аргумент — тип объекта. Особенности:

- при отображении имён из плагинов, тип определяется плагином;
- при отображении продукта предыдущего отображения, типом может быть только USER и ROLE;
- если имя объекта будет указано явно, то оно будет учитываться при отображении;
- при использовании ключевого слова ANY будут отображены объекты с любыми именами данного типа.

В предложении TO указывается пользователь или роль, на которого будет произведено отображение. NAME является не обязательным аргументом. Если он не указан, то в качестве имени объекта будет использовано оригинальное имя из отображаемого объекта.

Воспользоваться оператором создания отображений может SYSDBA, владелец базы данных (если отображение локальное), пользователь с ролью RDB$ADMIN, пользователь root (Linux).

Синтаксис позволяет изменять любые опции существующего отображения (ALTER MAPPING) и удалять отображение (DROP MAPPING).

Примеры

1. Включение доступа определённому пользователю из другой базы данных к текущей базе данных под другим именем.

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

2. Включение SYSDBA сервера (от основной базы данных безопасности) для доступа к текущей базе данных.

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

3. Обеспечение гарантирование, что у пользователей, которые подключаются традиционным плагином аутентификации не слишком много прав.

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

9.6 Аудит

Аудит событий реализован на основе утилиты `FBTrace`. Ред База Данных отличает пользовательскую трассировку и системный аудит, которые с помощью Services API позволяют отслеживать и анализировать все, что происходит в базе данных в режиме реального времени. Средства трассировки и аудита позволяют серверу отслеживать и записывать в лог-файлы такие события: соединения и отсоединения от БД (создания и удаления БД), операции DML и DDL, выполнение хранимых процедур и т.д. По умолчанию лог-файлы размещаются в том же каталоге, что и отслеживаемая (логируемая) БД, и имеют имя вида: `<имя_базы.fbtrace_text>` или `<имя_базы.fbtrace_bin>` для текстового и бинарного формата лога соответственно. Запись в лог для каждой конкретной БД начинает вестись с момента ее создания или присоединения к ней и до момента отсоединения от нее или ее удаления. Регистрируются события, завершившиеся как удачно, так и неудачно (с ошибкой).

Сессию системного аудита запускает сам сервер. Это означает, что нет необходимости взаимодействия с пользователем. События, которые будут отслеживаться в этой сессии, задаются в конфигурационном файле и читаются при старте сессии.

Параметр `AuditTraceConfigFile` в файле конфигурации firebird.conf задает имя и расположение файла с настройками системного аудита. Этот параметр по умолчанию имеет значение `fbtrace.conf`. Но по умолчанию он не включен (`enabled false`), что означает отсутствие сессий системного аудита. Файл с шаблоном настроек fbtrace.conf находится в корневом каталоге и содержит список отслеживаемых событий и указывает размещение логов трассировки для каждого события. Это позволяет достаточно гибко настроить параметры аудита различных событий для любой базы данных, при этом логирование будет осуществляться в отдельные файлы.

Что касается пользовательской трассировки, она нуждается в запуске пользователем явно. При запуске сессии пользовательской трассировки из приложения задаются ее конфигурация и имя (необязательный параметр). Конфигурация сессии представляет собой текстовый файл, составленный в соответствии с правилами и синтаксисом, приведенными в файле fbtrace.conf. Любой пользователь может инициировать и управлять сессией трассировки. Обычный пользователь может управлять сессиями только в своих соединениях и не может управлять сессиями, начатыми другими пользователями. Администраторы могут управлять любыми сессиями.

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

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

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

С новой утилитой командной строки `rdbtracemgr`, которая может быть найдена в корневом каталоге установки сервера, добавлена трассировка в интерактивном режиме. Эта утилита включает следующий функционал:

- Запуск пользовательской трассировки и отображение выходных данных сервера (`START`)
- Завершение сеанса трассировки (`STOP`)
- Приостановление и возобновление сеанса трассировки (`SUSPEND` и `RESUME`)
- Список всех доступных в настоящее время сеансов трассировки (`LIST`)

Синтаксис вызова утилиты следующий:

```
rdbtracemgr {-SE[RVICE] <имя сервиса> | -U[SER] <имя пользователя> | -P[ASSWORD]
<пароль> | -FE[TCH] <путь / имя файла> | -T[RUSTED]}
{-STA[RT] | -STO[P] | -SU[SPEND] | -R[ESUME] | -L[IST]}
[-N[AME] <имя сессии> | -I[D] <ID сессии> | -C[ONFIG] <конфигурационный файл>]
```

Для скрытия пароля появилась возможность считывания пароля из файла. Параметр `-fe[tch]` можно использовать вместо параметра `-p[assword]`. Параметр принимает одно значение—строку без кавычек и апострофов, содержащую путь и имя файла с паролем. Файл должен быть доступен текущему пользователю операционной системы для чтения.

Типы событий аудита

В логе аудита могут быть зарегистрированы следующие события:

- начало и окончание ведения аудита для БД;
- присоединение к БД и отсоединение от нее;
- присоединение к сервису и отсоединение от него;
- старт сервиса, запрос к сервису;
- подготовка, выполнение и освобождение запроса к БД, а также выборка записей;
- компиляция и выполнение BLR- и DYN-запросов;
- начало и окончание выполнения хранимой процедуры;
- начало и окончание выполнения хранимой функции;
- начало и окончание выполнения триггера;
- установка значения контекстной переменной;
- начало и завершение транзакции;
- возникновение ошибок и предупреждений;
- сборка мусора.

По умолчанию система аудита выключена.

Несанкционированной попыткой выполнения действия считается такая, при которой не была пройдена аутентификация, либо не оказалось прав на выполнение действия. Неуспешной – любая другая неудачная попытка, закончившаяся ошибкой.

Сообщения о вызове сервисов и предъявлении факторов аутентификации записываются в лог-файл базы `security3.fdb`. В unix-системах у суперсервера недостаточно прав для записи в данный файл. Для решения этой проблемы сервер должен быть запущен от имени `root` (нужно выполнить скрипт `restoreRootRunUser.sh` из каталога `bin`), либо лог-файл может быть создан вручную и пользователь `firebird` должен иметь право на запись в него.

Возможно использование системы ротации логов, которая активизируется по достижении файлом журнала аудита заданного пользователем максимального размера. При этом рабочий лог-файл переименовывается в файл с именем `<log_filename>.<текущая дата и время>.<log_ext>`, где дата и время записываются в виде `<YYYY-MM-DDThh-mm-ss>`, `<log_ext>` – расширение лог-файла. Этот файл упаковывается,в архив ZIP в Windows-системах,в GZIP - в Linux-системах. Директория, в которой будет создан архив, задаётся параметром `archive_directory`. После переименования рабочего лога, создается новый файл с именем переименованного. Этот новый файл используется в дальнейшем в качестве рабочего лога. Удаление старых лог-файлов не предусмотрено и может осуществляться средствами ОС и планировщиками задач.

Настройка аудита. Параметры конфигурационного файла

Настройка регистрации событий происходит с помощью изменения параметров в файле `fbtrace.conf`, расположенном в каталоге установки Ред База Данных.

Файл конфигурации может состоять из двух секций

```
database
{
...
}
services
{
...
}
```

Различные `database ...` секции состоят из параметров, отвечающие за логирование событий на уровне запросов. `services` секция может быть только в единственном экземпляре, позволяет делать трейс для широкого круга вызовов Services API (таких как резервирование, восстановление данных и т.д.)

Строка, следующая за символом `#`, считается комментарием. В параметрах, значения которых допускают использование регулярных выражений, используется синтаксис регулярных выражений SQL (аналогично оператору `SIMILAR TO`). Значение параметра `true` означает что параметр включен, `false` – что он выключен.

В текстовом режиме по умолчанию включено логирование единственного типа событий — завершения выполнения SQL-запросов (если включен сам аудит).

В бинарном режиме автоматически регистрируются события всех типов. Игнорируются все параметры, за исключением `enabled`, `format`, `log_filename`, `max_log_size`, `rotate_log` и `archive_directory`.

В конфигурационном файле настраиваются следующие параметры:

enabled = true/false – вести аудит или нет. По умолчанию аудит выключен.

format = 0/1/2 – формат лог-файла. 0 – текстовый (по умолчанию), 1 – бинарный, 2 - запись в системный лог.

В Linux запись в системный лог осуществляется вызовом функции syslog, обычно события регистрируются в системный лог-файл (например, `/var/log/messages`). При этом используются следующие категории событий:

- `LOG_AUTH` - для событий аутентификации;
- `LOG_AUTHPRIV` - для событий безопасности;
- `LOG_DAEMON` - для остальных событий.

Приоритет событий:

- `LOG_ERR` - для событий, завершившихся с ошибкой;
- `LOG_NOTICE` - для событий безопасности;

- `LOG_INFO` - для всех остальных событий.

В Windows события регистрируются в системный журнал событий. Для успешных событий используется тип сообщения `EVENTLOG_INFORMATION_TYPE`, для неуспешных - `EVENTLOG_ERROR_TYPE`. В качестве источника будет указан `Red Database SQL Server`.

log_filename = <**строка**> — имя файла лога. Если этот параметр не задан, лог-файл создаётся в той же папке, где находиться БД и имеет имя вида `<имя_базы.fbtrace_text>` или `<имя_базы.fbtrace_bin>`. Возможно использование регулярных выражений в этом параметре. Например, с их помощью можно разбить путь к БД на группы и обращаться к этим группам конструкциями вида \1, \2 и т.д. (см. примеры ниже). При явном указании имени файла, все события от всех логируемых БД будут сохранятся в данном файле.В этом параметре разделитель каталогов Windows - символ обратной косой черты \ - должен дублироваться.

max_log_size = <**число**> — задает максимальный размер log-файлов в мегабайтах. Если значение параметра равно 0, то размер файла журнала не ограничен, ротация логов не используется. Значение по умолчанию для текстового формата лог-файла – 50. Значение по умолчанию для бинарного формата – 500.

rotate_log = **true/false** — определяет, использовать ли систему ротации логов. Если значение равно `true`, то при достижении логом размера `max_log_size` будет выполняться его ротация. Если значение равно `false`, то при достижении логом размера `max_log_size` запись в лог будет остановлена, сессия трассировки будет отключена. Значение по умолчанию для текстового формата лог-файла – `true`. Значение по умолчанию для бинарного формата - `false`.

archive_directory = <**строка**> — путь к директории, в которую будут записываться логи после ротации. Значение по умолчанию – пусто, то есть архив будет создан в той же директории, где лог. Если операция записи в журнал завершилась с ошибкой исчерпания места, то будет произведена принудительная ротация логов.

include_user_filter = <**регулярное выражение**> — регулярное выражение, которому должно соответствовать имя пользователя, от которого выполняется соединение с базой данных. Аудит будет работать только для тех подключений, которые прошли эту проверку. Значение по умолчанию – пусто, то есть в лог будут включены все подключения.

exclude_user_filter = <**регулярное выражение**> — регулярное выражение, противоположное `include_user_filter`. Подключения от пользователей, совпавших с этим выражением не будут регистрироваться. Значение по умолчанию – пусто, то есть в лог будут включены все подключения.

include_process_filter = <**регулярное выражение**> — регулярное выражение, которому должно соответствовать название пользовательского процесса, выполняющего соединение с базой данных. Аудит будет работать только для тех подключений, которые прошли эту проверку. Значение по умолчанию – пусто, то есть в лог будут включены все подключения.

exclude_process_filter = <**регулярное выражение**> — регулярное выражение, противоположное `include_process_filter`. Подключения от процессов, совпавших с этим выражением не будут регистрироваться. Значение по умолчанию – пусто, то есть в лог будут включены все подключения.

include_filter = <**регулярное выражение**> — этот параметр задаёт регулярное выражение в синтаксисе SQL (`SIMILAR TO`), которому должен удовлетворять текст SQL-запроса. Если текст запроса не удовлетворяет заданному здесь шаблону, этот запрос не записывается в лог. Значение по умолчанию – пусто, то есть в конечный лог будут включены все запросы. Применяется только для текстового формата лог-файла.

exclude_filter = <**регулярное выражение**> — задаёт регулярное выражение в синтаксисе SQL (`SIMILAR TO`), которому не должен удовлетворять текст SQL-запроса. Аналогично

`include_filter`. Значение по умолчанию – пусто. Применяется только для текстового формата лог-файла.

log_connections = true/false — определяет, записывать ли события присоединения/отсоединения к БД в лог-файл. Применяется только для текстового формата лог-файла. Неудачные попытки присоединения не записываются через пользовательский трейс.

connection_id = <число> — задаёт номер (идентификатор) подключения на сервере, которое будет отслеживаться. По умолчанию равно 0, т.е. отслеживаются все подключения. Применяется только для текстового формата лог-файла.

log_transactions = true/false — определяет, записывать ли события начала и завершения транзакций в лог-файл.[15] Применяется только для текстового формата лог-файла.

log_statement_prepare = true/false — определяет, записывать ли события подготовки запросов к БД в лог-файл. Применяется только для текстового формата лог-файла.

log_statement_free = true/false — определяет, записывать ли события освобождения запросов к БД в лог-файл. Применяется только для текстового формата лог-файла.

log_statement_start = true/false — определяет, записывать ли события начала выполнения запросов к БД в лог-файл. Применяется только для текстового формата лог-файла.

log_statement_finish = true/false — определяет, записывать ли события окончания выполнения запросов к БД в лог-файл. Применяется только для текстового формата лог-файла.

log_procedure_start = true/false — определяет, записывать ли события начала выполнения хранимых процедур. Применяется только для текстового формата лог-файла.

log_procedure_finish = true/false — определяет, записывать ли события завершения выполнения хранимых процедур. Применяется только для текстового формата лог-файла.

log_procedure_compile = true/false — оопределяет, записывать ли события парсинга кода (`BLR`) хранимых процедур. Также выводит план доступа процедуры. Применяется только для текстового формата лог-файла.

log_function_start = true/false — определяет, записывать ли события начала выполнения хранимых функций. Применяется только для текстового формата лог-файла.

log_function_finish = true/false — определяет, записывать ли события завершения выполнения хранимых функций. Применяется только для текстового формата лог-файла.

log_function_compile = true/false — определяет, записывать ли события парсинга кода (BLR) хранимых функций. Также выводит план доступа функции. Применяется только для текстового формата лог-файла.

log_trigger_start = true/false — определяет, записывать ли события начала выполнения триггеров. Применяется только для текстового формата лог-файла.

log_trigger_finish = true/false — определяет, записывать ли события завершения выполнения триггеров. Применяется только для текстового формата лог-файла.

log_trigger_compile = true/false — определяет, записывать ли события парсинга кода (BLR) триггеров. Также выводит план доступа триггера. Применяется только для текстового формата лог-файла.

log_context = true/false — определяет, записывать ли события изменений значений контекстных переменных в лог-файл. Применяется только для текстового формата лог-файла.

[15] При подтверждении транзакции записывается операция `commit`, при откате - `rollback`

log_errors = true/false — включает/отключает запись об ошибках. Применяется только для текстового формата лог-файла.

log_warnings = true/false — включает/отключает запись о предупреждениях. Применяется только для текстового формата лог-файла.

include_gds_codes = <GDS-коды> — это список GDS-кодов ошибок или предупреждений. Если список пустой, то в конечный лог будут включены все ошибки. Иначе в лог будут записываться только ошибки из этого списка. Применяется только для текстового формата лог-файла.

exclude_gds_codes = <GDS-коды> — это список GDS-кодов ошибок или предупреждений. Если список пустой, то в конечный лог будут включены все ошибки. Иначе в лог будут записываться ошибки, не входящие в этот список. Применяется только для текстового формата лог-файла.

log_initfini = true/false — определяет, записывать ли события начала/окончания ведения аудита БД в лог-файл. Применяется только для текстового формата лог-файла.

log_sweep = true/false — определяет, записывать ли события процесса сборки мусора в лог-файл. Применяется только для текстового формата лог-файла.

print_plan = true/false — включает/отключает печать планов запросов.

explain_plan = true/false — включает/отключает печать расширенных планов запросов.

print_perf = true/false — включает/отключает печать статистики выполнения запросов.

log_blr_requests = true/false — определяет, записывать ли события прямого выполнения откомпилированных запросов во внутреннем представлении сервера - BLR. Применяется только для текстового формата лог-файла.

print_blr = true/false — если параметр установлен в `true`, то содержимое BLR-запросов будет преобразовываться в текстовое представление. Если параметр установлен в `false`, то BLR-запрос будет сохранен в двоичном виде (последовательность байт). Этот параметр будет работать только для текстового формата лога. В бинарном формате содержимое BLR всегда будет сохраняться в двоичном виде.

log_dyn_requests = true/false — определяет, записывать ли события прямого выполнения откомпилированных запросов на изменение метаданных (DDL) во внутреннем представлении сервера - DYN. Применяется только для текстового формата лог-файла.

print_dyn = true/false — если параметр установлен в `true`, то содержимое DYN-запросов будет преобразовываться в текстовое представление. Если параметр установлен в `false`, то DYN-запрос будет сохранен в двоичном виде (последовательность байт). Этот параметр будет работать только для текстового формата лога. В бинарном формате содержимое DYN всегда будет сохраняться в двоичном виде.

log_privilege_changes = true/false — включает/отключает запись событий, связанных с изменением правил разграничения доступа. Применяется только для текстового формата лог-файла.

log_changes_only = true/false — включает/отключает запись только тех событий, которые изменяли данные в базе. Применяется только для текстового формата лог-файла.

log_security_incidents = true/false — включает/отключает запись событий, связанных с нарушением безопасности сервера (инциденты безопасности). Если этот параметр включен, все такие события будут регистрироваться независимо от того, включена ли регистрация событий данного типа в других настройках. Неудачные попытки присоединения не записываются через пользовательский трейс. По умолчанию отключено.

print_security_level = true/false — включает/отключает печать уровня важности события. Уровни важности:

- `EMERG` - аварийный;
- `FATAL` - фатальный;
- `CRITICAL` - критический;
- `HIGH` - высокий;
- `MEDIUM` - средний;
- `LOW` - низкий;
- `DEBUG` - отладочный.

По умолчанию отключено.

print_security_type = true/false — включает/отключает печать типа события. Типы событий:

- `AUTHENTICATION` - аутентификация;
- `IDENTIFICATION` - идентификация;
- `USER MANAGEMENT` - управление пользователями и ролями;
- `ACCESS MANAGEMENT` - управление доступом;
- `START/STOP COMPONENT` - запуск/остановка сервера;
- `RESTORE` - восстановление из резервной копии;
- `ACCESS TO PROTECTED INFO` - изменение правил разграничения доступа;
- `VALIDATION OF THE SOFTWARE COMPONENTS` - проверка целостности объектов контроля;
- `OTHER` - другое.

По умолчанию отключено.

time_threshold = <число> – минимальное время выполнения запросов, процедур, транзакций и т.д. События, время выполнения которых меньше указанного, не будут регистрироваться в журнале. Значение по умолчанию – 100 мс. Применяется только для текстового формата лог-файла.

max_sql_length = <число> — максимальная длина одной записи SQL-запроса в лог-файле, в байтах. Значение по умолчанию – 0 (неограниченно), максимальное значение – 64K. Если длина запроса больше указанного здесь значения, запрос будет обрезан. Применяется только для текстового формата лог-файла[16].

max_blr_length = <число> — максимальная длина BLR-запроса, сохраняемого в лог, в байтах. Значение по умолчанию – 500 байт. Максимальное значение – 64K. Если длина запроса больше указанного здесь значения, запрос будет обрезан. Применяется только для текстового формата лог-файла.

max_dyn_length = <число> — максимальная длина DYN-запроса, сохраняемого в лог, в байтах. Значение по умолчанию – 500 байт. Максимальное значение – 64K. Если длина запроса больше указанного здесь значения, запрос будет обрезан. Применяется только для текстового формата лог-файла.

max_arg_length = <число> — максимальная длина одного параметра запроса / процедуры в лог-файле. Значение по умолчанию – 0 (неограниченно). Максимальное значение – 64K. Если длина параметра больше указанного здесь значения, параметр будет обрезан. Применяется только для текстового формата лог-файла.

[16] Для бинарного формата лог-файла запрос будет записан в лог целиком, однако при подключении такого лога к базе данных максимальный размер запросов не может превышать размер страницы БД. В противном случае запрос будет обрезан.

max_arg_count = <**число**> — максимальное количество параметров запроса / процедуры, которое заносится в лог-файл. Значение по умолчанию – 0 (неограниченно). Параметры, номера которых больше указанного здесь значения, отображаться не будут. Применяется только для текстового формата лог-файла.

log_services = **true**/**false** – включает или отключает аудит событий присоединения/отсоединения и старта сервиса.

log_service_query = **true**/**false** – определяет, записывать ли события запросов к сервису.

cancel_on_error = **true**/**false** – определяет, надо ли отменять текущую логируемую операцию при возникновении ошибки записи в файл трейса. Согласно общему поведению, при возникновении подобной ошибки сессия дальнейшего логирования прекращается.

События, которые могут быть отменены:

- Присоединение к серверу;
- Старт транзакции;
- Выполнение процедуры;
- Выполнение функции;
- Выполнение триггера;
- Подготовка DSQL;
- Выполнение DSQL;
- Присоединение к сервису;
- Старт сервиса;
- Запрос к сервису;
- Отсоединение от сервиса.

При помощи регулярных выражений можно задавать конкретные БД для логирования. Примеры конфигурационных файлов аудита:

Пример 1:

```
#Лог ведется для баз с именами test.fdb, azk2.fdb, rules.fdb
database = %[\\/](test|azk2|rules).fdb
{
   enabled = true
   # Логи сохраняются в файлы test.log, azk2.log, rules.log соответственно
   log_filename = \1.log
}
```

Пример 2:

```
#Для всех БД на диске C с расширением fdb - формат лога текстовый
database = C:%.fdb
{
   enabled = true
   format = 0
}
#Для всех БД с расширением fdb на диске D - формат лога бинарный
database = D:%.fdb
{
   enabled = true
```

```
   format = 1
}
```

В регулярных выражениях можно использовать группировку - ()

Пример 3:

```
#Первая группа (%[\\/]) - любой путь к файлам БД
#Вторая группа (test|azk) - имя файла БД test или azk
database = (%[\\/])(test|azk).fdb
{
   enabled = true
   #\1 - Первая группа - путь.
   #\2 - Вторая группа - имя файла
   log_filename = \1\\logs\\\2.log
}
```

То есть будут создаваться лог-файлы с именами <имя_базы.log> в каталоге logs расположенном на одном уровне с каталогом, содержащим базы.

Текстовый файл аудита

Журнал аудита содержит записи в хронологическом порядке по времени завершения события. Далее будут описаны все варианты записей в зависимости от типа события.

Результат UNAUTHORIZED записывается, если возникла ошибка, связанная с недостаточными правами доступа или неверными данными при аутентификации. В случае возникновения других ошибок результатом будет FAILED.

Начало/окончание ведения аудита

Включить ведение записей об этом событии позволяет параметр log_initfini. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>)
    SESSION_<ID сессии> <имя сессии>
    <путь к базе данных>
```

Присоединение/отсоединение от БД

Включить ведение записей об этом событии позволяет параметр log_connections. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) <событие>,
<важность события>, <тип события>
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>

<тип события>:= {CREATE_DATABASE| ATTACH_DATABASE| DROP_DATABASE| DETACH_DATABASE}

<сведения о соединении> ::=
   <путь к БД> (ATT_<ID соединения>, <имя пользователя>:<роль>, <кодировка>,
                <протокол соединения>:<IP адрес или имя компьютера>:<MAC-адрес>)
```

В случае неуспешной или несанкционированной попытки выполнения присоединения или отсоединения от БД в типе события фиксируется результат FAILED или UNAUTHORIZED.

Неудачные попытки присоединения не записываются через пользовательский трейс.

Начало/завершение транзакции

Включить ведение записей об этом событии позволяет параметр `log_transactions`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) <событие>
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
        (TRA_<ID транзакции>,<уровень изоляции>|<режим разр. бл-к>|<режим доступа>)
    [<глобальные счетчики>]
    [<табличные счетчики>]

<тип события>:= {START_TRANSACTION | COMMIT_RETAINING | COMMIT_TRANSACTION |
                 ROLLBACK_RETAINING | ROLLBACK_TRANSACTION}

<сведения о соединении> ::=
   <путь к БД> (ATT_<ID соединения>, <имя пользователя>:<роль>, <кодировка>,
                <протокол соединения>:<IP адрес или имя компьютера>:<MAC-адрес>)

<уровень изоляции> ::= {CONSISTENCY | CONCURRENCY | {READ_COMMITTED|REC_VERSION} |
                        {READ_COMMITTED|NO_REC_VERSION}}

<режим разрешения блокировок> ::= {WAIT [N] | NOWAIT};

<режим доступа к данным> := {READ_ONLY | READ_WRITE};

<глобальные счетчики> ::= m1 ms, m2 read(s), m3 write(s), m4 fetch(es), m5 mark(s)

<табл. счетчики> ::= Table Natural Index Update Insert Delete Backout Purge Expunge
                     ************************************************************
```

В случае неуспешной или несанкционированной попытки выполнения старта или завершения транзакции в типе события фиксируется результат `FAILED` или `UNAUTHORIZED`.

Записи содержат табличные и глобальные счетчики, только если в настройках включен параметр `print_perf`.

Таблица 9.9 — Табличные счётчики

`Table`	Имя таблицы
`Natural`	Количество записей, считанных последовательно (без индекса)
`Index`	Количество записей, считанных по индексу
`Update`	Количество обновлённых записей
`Insert`	Количество вставленных записей
`Delete`	Количество удалённых записей
`Backout`	Количество записей, для которых были восстановлены предыдущие версии из-за отката транзакции или точки сохранения
`Purge`	Количество записей, для которых были удалены устаревшие версии, не нужные ни одной активной транзакции

Expunge	Количество вычищенных средствами сборки мусора записей (expunged records). Это происходит, когда запись, которая не видна какой-либо активной транзакции, удаляется и удаляющая транзакция подтверждается (commit). Удалённая запись и все ранее подтверждённые версии этой записи удаляются, чтобы занятое ими дисковое пространство можно было повторно использовать. Если запись всё ещё видна для более старых snapshot транзакций, конечно, удаление может произойти только после завершения этих транзакций
Lock	Количество записей прочитанных с использованием предложения WITH LOCK
Wait	Количество попыток обновления/модификации/блокировки записей принадлежащих нескольким активным транзакциям. Транзакция находится в режиме WAIT
Conflict	Количество неудачных попыток обновления/модификации/блокировки записей принадлежащих нескольким активным транзакциям. В таких ситуациях сообщается о конфликте обновления (UPDATE CONFLICT)
BVersion	Количество прочитанных версий при поиске видимых версий записей
Fragment	Количество прочитанных фрагментов записей
Refetch	Количество повторно прочитанных записей

Таблица 9.10 — Глобальные счётчики

ms	Время выполнения
read(s)	Количество страниц, считанных с диска
write(s)	Количество страниц, записанных на диск
fetch(es)	Количество страниц, считанных из страничного кэша
mark(s)	Количество страниц, изменённых в страничном кэше

Подготовка запросов к БД

Включить ведение записей об этом событии позволяет параметр log_statement_prepare. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) PREPARE_STATEMENT
<важность события>, <тип события>
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
        (TRA_<ID транзакции>,<уровень изоляции>|<режим разр. бл-к>|<режим доступа>)
Statement <идентификатор запроса>:
-------------------------------------------------------------------------------
<содержимое запроса>
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
[<план запроса>]
<время выполнения> ms

<сведения о соединении> ::=
   <путь к БД> (ATT_<ID соединения>, <имя пользователя>:<роль>, <кодировка>,
```

```
                <протокол соединения>:<IP адрес или имя компьютера>:<MAC-адрес>)
<уровень изоляции> ::= {CONSISTENCY | CONCURRENCY | {READ_COMMITTED|REC_VERSION} |
                        {READ_COMMITTED|NO_REC_VERSION}}
<режим разрешения блокировок> ::= {WAIT [N] | NOWAIT};
<режим доступа к данным> := {READ_ONLY | READ_WRITE};
```

В случае неуспешной или несанкционированной попытки подготовки запроса в типе события фиксируется результат `FAILED` или `UNAUTHORIZED`.

План выполнения запроса выводится, если включен параметр `print_plan`.

Освобождение запросов к БД

Включить ведение записей об этом событии позволяет параметр `log_statement_free`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) {FREE_STATEMENT |
CLOSE_CURSOR} <важность события>, <тип события>
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
Statement <идентификатор запроса>:
-------------------------------------------------------------------------------
<содержимое запроса>
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
<план запроса>
```

План выполнения запроса выводится, если включен параметр `print_plan`.

Начало/окончание выполнения запросов к БД

Включить ведение записей об этом событии позволяют параметры `log_statement_start` и `log_statement_finish`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) <событие>,
<важность события>, <тип события>
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
        (TRA_<ID транзакции>,<уровень изоляции>|<режим разр. бл-к>|<режим доступа>)
Statement <идентификатор запроса>:
-------------------------------------------------------------------------------
<содержимое запроса>
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
[<план запроса>]
<параметры выполнения запроса>
[<количество выбранных записей> records fetched]
 [sorting memory usage: total: <суммарный размер кэша>, cached: <размер RAM кэша>,
on disk: <размер дискового кэша>]
    [<глобальные счетчики>]
    [<табличные счетчики>]
<тип события>:= {EXECUTE_STATEMENT_START | EXECUTE_STATEMENT_FINISH}
```

В случае неуспешной или несанкционированной попытки выполнения запроса в типе события фиксируется результат `FAILED` или `UNAUTHORIZED`.

В типе события EXECUTE_STATEMENT_FINISH, при использовании сортировки, будет добавлена запись `sorting memory usage` с счетчиками выделенного кэша. В противном случае, если сортировка не использовалась, к `records fetched` будет добавлено `without sorting`.

План выполнения запроса выводится, если включен параметр `print_plan`.

Записи содержат табличные и глобальные счетчики (см. таблицу 9.9 и таблицу 9.10), только если в настройках включен параметр `print_perf`.

Изменение значений контекстных переменных

Включить ведение записей об этом событии позволяет параметр `log_context`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) SET_CONTEXT
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
        (TRA_<ID транзакции>,<уровень изоляции>|<режим разр. бл-к>|<режим доступа>)
[<пространство_имен>] <имя_переменной> = <значение_переменной>
```

Изменение правил разграничения доступа

Включить ведение записей об этом событии позволяет параметр `log_privilege_changes`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) PRIVILEGES_CHANGE
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
        (TRA_<ID транзакции>,<уровень изоляции>|<режим разр. бл-к>|<режим доступа>)
    Executed by <executor> as <grantor>,operation:{ADD|DELETE} PRIVILEGE <прив-ия>
    <объект> for <имя пользователя>
    Attachment: <ID_соединения>, Transaction: <ID_транзакции>

<привилегия> ::= {ALL | INSERT | UPDATE | DELETE | SELECT | EXECUTE | REFERENCE |
CREATE | ALTER | ALTER ANY | DROP | DROP ANY | ROLE | ENCRYPTION KEY}
```

Начало/завершение выполнения хранимых процедур (функций)

Включить ведение записей об этом событии позволяют параметры `log_procedure_start` и `log_procedure_finish` (`log_function_start`, `log_function_finish`). Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) <событие>
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
        (TRA_<ID транзакции>,<уровень изоляции>|<режим разр. бл-к>|<режим доступа>)
Procedure (Function) <имя процедуры (функции)>:
<входные параметры хранимой процедуры (функции)>
[returns: <выходное значение функции>]
<количество выбранных записей> records fetched
 [sorting memory usage: total: <суммарный размер кэша>, cached: <размер RAM кэша>,
on disk: <размер дискового кэша>]
    [<глобальные счетчики>]
    [<табличные счетчики>]
```

```
<тип события>:= {EXECUTE_PROCEDURE_START | EXECUTE_FUNCTION_START |
                 EXECUTE_PROCEDURE_FINISH | EXECUTE_FUNCTION_FINISH}
```

В случае неуспешной или несанкционированной попытки выполнения хранимой процедуры или функции в типе события фиксируется результат `FAILED` или `UNAUTHORIZED`.

В типе события EXECUTE_PROCEDURE_FINISH/EXECUTE_FUNCTION_FINISH, при использовании сортировки, будет добавлена запись `sorting memory usage` с счетчиками выделенного кэша. В противном случае, если сортировка не использовалась, к `records fetched` будет добавлено `without sorting`.

Записи содержат табличные и глобальные счетчики (см. таблицу 9.9 и таблицу 9.10), только если в настройках включен параметр `print_perf`.

Начало/завершение выполнения триггеров

Включить ведение записей об этом событии позволяют параметры `log_trigger_start` и `log_trigger_finish`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>)
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
        (TRA_<ID транзакции>,<уровень изоляции>|<режим разр. бл-к>|<режим доступа>)
        <имя_триггера> [FOR <имя_таблицы (представления)>] ({ON <событие БД>} |
        {BEFORE | AFTER} <событие таблицы (представления) или DDL-событие>)
    [<глобальные счетчики>]
    [<табличные счетчики>]

<тип события>:= {EXECUTE_TRIGGER_START | EXECUTE_TRIGGER_FINISH}
```

В случае неуспешной или несанкционированной попытки выполнения триггера в типе события фиксируется результат `FAILED` или `UNAUTHORIZED`.

Записи содержат табличные и глобальные счетчики (см. таблицу 9.9 и таблицу 9.10), только если в настройках включен параметр `print_perf`.

Парсинг кода хранимых процедур/функций/триггеров

Включить ведение записей об этих событиях позволяют параметры `log_procedure_compile`, `log_function_compile` и `log_trigger_compile`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) COMPILE_PROCEDURE
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>

Procedure/Function/Trigger <имя>:
----------------------------------------
<план процедуры/функции/триггера>
<время выполнения> ms
```

Компиляция BLR-запросов перед исполнением

Включить ведение записей об этом событии позволяет параметр `log_blr_requests`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) COMPILE_BLR
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
        (TRA_<ID транзакции>,<уровень изоляции>|<режим разр. бл-к>|<режим доступа>)
Statement <идентификатор запроса>:
-------------------------------------------------------------------------------
<содержимое запроса>
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
<время выполнения> ms
```

В случае неуспешной или несанкционированной попытки компиляции BLR-запроса в типе события фиксируется результат `FAILED` или `UNAUTHORIZED`.

Выполнение BLR-запросов

Включить ведение записей об этом событии позволяет параметр `log_blr_requests`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) EXECUTE_BLR
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
        (TRA_<ID транзакции>,<уровень изоляции>|<режим разр. бл-к>|<режим доступа>)
Statement <идентификатор запроса>:
-------------------------------------------------------------------------------
<содержимое запроса>
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    [<глобальные счетчики>]
    [<табличные счетчики>]
```

В случае неуспешной или несанкционированной попытки выполнения BLR-запроса в типе события фиксируется результат `FAILED` или `UNAUTHORIZED`.

Записи содержат табличные и глобальные счетчики (см. таблицу 9.9 и таблицу 9.10), только если в настройках включен параметр `print_perf`.

Выполнение DYN-запросов

Включить ведение записей об этом событии позволяет параметр `log_dyn_requests`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) EXECUTE_DYN
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
        (TRA_<ID транзакции>,<уровень изоляции>|<режим разр. бл-к>|<режим доступа>)
-------------------------------------------------------------------------------
<содержимое запроса>
<время выполнения> ms
```

В случае неуспешной или несанкционированной попытки выполнения DYN-запроса в типе события фиксируется результат `FAILED` или `UNAUTHORIZED`.

Присоединение/отсоединение к сервисам

Включить ведение записей об этом событии позволяет параметр `log_services`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>)
{ATTACH_SERVICE|DETACH_SERVICE} <важность события>, <тип события>
     service_mgr, (Service <ID сервиса> , <имя пользователя>,
                   <протокол соединения>:<IP адрес или имя компьютера>:<MAC-адрес>,
                   <клиентский процесс>:<ID клиентского процесса>)
```

В случае неуспешной или несанкционированной попытки присоединения (отсоединения) к сервису в типе события фиксируется результат `FAILED` или `UNAUTHORIZED`.

Старт сервиса

Включить ведение записей об этом событии позволяет параметр `log_services`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) START_SERVICE
     service_mgr, (Service <ID сервиса> , <имя пользователя>,
                   <протокол соединения>:<IP адрес или имя компьютера>:<MAC-адрес>,
                   <клиентский процесс>:<ID клиентского процесса>)
    <тип запроса к сервису>
    <опции, переданные сервис-менеджеру от клиента при запуске>
```

В случае неуспешной или несанкционированной попытки старта сервиса в типе события фиксируется результат `FAILED` или `UNAUTHORIZED`.

Запросы к сервису

Включить ведение записей об этом событии позволяют параметры `log_services` и `log_service_query`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) QUERY_SERVICE
     service_mgr, (Service <ID сервиса> , <имя пользователя>,
                   <протокол соединения>:<IP адрес или имя компьютера>:<MAC-адрес>,
                   <клиентский процесс>:<ID клиентского процесса>)
    <тип запроса к сервису>
    [Send portion of the query: <данные, переданные сервис-менеджеру> ]
    [Receive portion of the query: <данные, полученные сервис-менеджером>]
```

В случае неуспешной или несанкционированной попытки выполнения запроса к сервису в типе события фиксируется результат `FAILED` или `UNAUTHORIZED`.

Событие с ошибкой или предупреждением

Включить ведение записей об ошибках (предупреждениях) позволяет параметр `log_errors` (`log_warnings`). Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) {ERROR AT|WARNING
AT} <...>
     <сведения о соединении>
```

```
    <клиентский процесс>:<ID клиентского процесса>
<ошибки>
```

Чистка базы данных

Включить ведение записей о чистке базы данных позволяет параметр `log_sweep`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>)
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
Transaction counters:
     Oldest interesting <OIT>
     Oldest active <OAT>
     Oldest snapshot <OST>
     Next transaction <Next>
[<глобальные счетчики>]
[<табличные счетчики>]

<тип события> ::= {SWEEP_START | SWEEP_FINISH | SWEEP_FAILED | SWEEP_PROGRESS}
```

Записи содержат табличные и глобальные счетчики (см. таблицу 9.9 и таблицу 9.10), только если в настройках включен параметр `print_perf`.

Адаптер для подключения бинарного файла аудита

Это инструмент анализа журнала аудита в бинарном формате, который позволяет подключать двоичные журналы в виде внешних таблиц и использовать SQL-запросы для поиска и фильтрации данных в них. Удаление и редактирование записей запрещено. Используется понятие версии формата бинарного лог-файла. Она проверяется при инициализации аудита (событие начала ведения аудита), если лог-файл уже существует и имеет бинарный формат. Если версия формата лога, используемая в Ред База Данных, отличается от версии формата существующего файла, то производится ротация (переименование существующего лога, создание рабочего лог-файла с таким же именем).

Подключение журнала производится с помощью SQL-запроса вида:

```
CREATE TABLE <table_name> EXTERNAL [FILE] <filespec> ADAPTER 'fbtrace'
[(<col_defs>)]
```

- `table_name` – имя таблицы, которая будет хранить данные аудита;
- `filespec` – имя лог-файла, который должен быть подключен;
- `ADAPTER` – ключевое слово, необходимое для подключения в качестве внешней таблицы файла с нестандартным форматом данных;
- `fbtrace` – название адаптера, предназначенного для обработки бинарного лога системы аудита;
- `col_defs` – описание полей таблицы аудита (см. таблицу 9.11).

При подключение бинарного лога необходимо в `firebird.conf` настроить параметр `ExternalFileAccess`, по умолчание он выключен и подключение внешних файлов невозможно.

Таблица 9.11 — Поля создаваемой таблицы аудита по типам событий

Общие поля таблицы аудита		
Имя поля	**Тип**	**Комментарий**
EVENT_TIME	TIMESTAMP	Время регистрации события
EVENT_PROCESS_ID	INTEGER	Идентификатор процесса, вызвавшего событие
EVENT_OBJECT_ID	VARCHAR(16)	Идентификатор объекта, вызвавшего событие (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
EVENT_NUMBER	INTEGER	Номер события, записанного процессом сервера

Начало ведения аудита		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	TRACE INIT
EVENT_DATABASE	BLOB TEXT	База данных, для которой ведется аудит

Окончание ведения аудита		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	TRACE FINI

Присоединение к базе данных		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	CREATE DATABASE или ATTACH DATABASE
EVENT_USER	BLOB TEXT	Пользователь, от имени которого выполнено присоединение к БД
EVENT_PROTOCOL	BLOB TEXT	Протокол, по которому произошло подключение к БД
EVENT_HOSTNAME	BLOB TEXT	Имя хоста, с которого произошло подключение к БД
EVENT_HW_ADDRESS	BLOB TEXT	Физический адрес клиента (MAC-адрес адаптера)
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором произошло подключение к БД
EVENT_RESULT	VARCHAR(14)	Результат подключения (успешно, неуспешно, несанкционированно)

Отсоединение от базы данных		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	DROP DATABASE или DETACH DATABASE
EVENT_USER	BLOB TEXT	Пользователь, от имени которого выполнено отключение от БД
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором произошло отключение от БД

Начало транзакции		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	START TRANSACTION
EVENT_USER	BLOB TEXT	Пользователь, от имени которого стартовала транзакция
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором стартовала транзакция
TRANS_ID	BIGINT	Идентификатор транзакции
TRANS_OPT	BLOB TEXT	Параметры транзакции
EVENT_RESULT	VARCHAR(14)	Результат старта транзакции (успешно, неуспешно, несанкционированно)

Завершение транзакции		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	COMMIT RETAINING, COMMIT TRANSACTION, ROLLBACK RETAINING или ROLLBACK TRANSACTION
EVENT_USER	BLOB TEXT	Пользователь, от имени которого завершилась транзакция
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором завершилась транзакция
TRANS_ID	BIGINT	Идентификатор транзакции
PERF_INFO	BLOB TEXT	Статистика производительности запроса
PERF_TIME	BIGINT	Время выполнения запроса
EVENT_RESULT	VARCHAR(14)	Результат завершения транзакции (успешно, неуспешно, несанкционированно)

Изменение значения контекстной переменной		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	SET CONTEXT
EVENT_USER	BLOB TEXT	Пользователь, от имени которого произвели изменения
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором выполнили изменения контекстной переменной
TRANS_ID	BIGINT	Идентификатор транзакции, в которой произвели изменения
VAR_NS	BLOB TEXT	Пространство имен, для которого устанавливается контекстная переменная
VAR_NAME	BLOB TEXT	Имя контекстной переменной
VAR_VALUE	BLOB TEXT	Значение контекстной переменной

Начало выполнения хранимой процедуры		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	START FUNCTION или START PROCEDURE
EVENT_USER	BLOB TEXT	Пользователь, от имени которого выполняется процедура

EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором выполняется хранимая процедура
TRANS_ID	BIGINT	Идентификатор транзакции
PROC_NAME	BLOB TEXT	Имя хранимой процедуры или функции
PROC_PARAMS	BLOB TEXT	Параметры хранимой процедуры или функции
EVENT_RESULT	VARCHAR(14)	Результат начала выполнения процедуры (успешно, неуспешно, несанкционированно)

Завершение выполнения хранимой процедуры		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	FINISH FUNCTION или FINISH PROCEDURE
EVENT_USER	BLOB TEXT	Пользователь, от имени которого завершилось выполнение процедуры
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором завершилось выполнение процедуры
TRANS_ID	BIGINT	Идентификатор транзакции
PROC_NAME	BLOB TEXT	Имя хранимой процедуры или функции
PROC_PARAMS	BLOB TEXT	Параметры хранимой процедуры или функции
FUNC_RESULT	BLOB TEXT	Возвращаемый результат хранимой функции
PERF_INFO	BLOB TEXT	Статистика производительности хранимой процедуры или функции
PERF_TIME	BIGINT	Время выполнения хранимой процедуры или функции
ROW_FETCHED	BIGINT	Число выбранных записей
EVENT_RESULT	VARCHAR(14)	Результат завершения выполнения процедуры (успешно, неуспешно, несанкционированно)

Подготовка запроса		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	PREPARE STATEMENT
EVENT_USER	BLOB TEXT	Пользователь, от имени которого шла подготовка запроса
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором шла подготовка запроса
TRANS_ID	BIGINT	Идентификатор транзакции
STMT_ID	BIGINT	Идентификатор запроса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
STMT_SQL	BLOB TEXT	Содержимое SQL-запроса
PERF_TIME	BIGINT	Время подготовки запроса
STMT_ACCESS_PATH	BLOB TEXT	План запроса
EVENT_RESULT	VARCHAR(14)	Результат подготовки запроса (успешно, неуспешно, несанкционированно)

Начало выполнения запроса

Имя поля	Тип	Комментарий
EVENT_TYPE	CHAR(24)	START EXECUTE STATEMENT
EVENT_USER	BLOB TEXT	Пользователь, от имени которого выполняется запрос
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором выполняется запрос
TRANS_ID	BIGINT	Идентификатор транзакции
STMT_ID	BIGINT	Идентификатор запроса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
STMT_PARAMS	BLOB TEXT	Параметры выполнения запроса
EVENT_RESULT	VARCHAR(14)	Результат начала выполнения запроса (успешно, неуспешно, несанкционированно)

Окончание выполнения запроса		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	FINISH EXECUTE STATEMENT
EVENT_USER	BLOB TEXT	Пользователь, от имени которого выполняется запрос
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором выполняется запрос
TRANS_ID	BIGINT	Идентификатор транзакции
STMT_ID	BIGINT	Идентификатор запроса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
STMT_PARAMS	BLOB TEXT	Параметры выполнения запроса
PERF_INFO	BLOB TEXT	Статистика производительности запроса
PERF_TIME	BIGINT	Время выполнения запроса
ROW_FETCHED	BIGINT	Число выбранных записей
EVENT_RESULT	VARCHAR(14)	Результат завершения выполнения запроса (успешно, неуспешно, несанкционированно)

Освобождение запроса		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	FREE STATEMENT или CLOSE CURSOR
EVENT_USER	BLOB TEXT	Пользователь, от имени которого происходит освобождение запроса
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором освобождается запрос
STMT_ID	BIGINT	Идентификатор запроса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)

Компилирование BLR-запроса		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	COMPILE BLR
EVENT_USER	BLOB TEXT	Пользователь, от имени которого компилируется запрос

EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором компилируется запрос
TRANS_ID	BIGINT	Идентификатор транзакции
STMT_ID	BIGINT	Идентификатор BLR-запроса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
BLR_DATA	BLOB BLR	Содержимое BLR-запроса
PERF_TIME	BIGINT	Время компилирования BLR-запроса
EVENT_RESULT	VARCHAR(14)	Результат компилирования BLR-запроса (успешно, неуспешно, несанкционированно)

Выполнение BLR-запроса		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	EXECUTE BLR
EVENT_USER	BLOB TEXT	Пользователь, от имени которого выполняется BLR-запрос
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором выполняется BLR-запрос
TRANS_ID	BIGINT	Идентификатор транзакции
STMT_ID	BIGINT	Идентификатор BLR-запроса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
PERF_INFO	BLOB TEXT	Статистика производительности BLR-запроса
PERF_TIME	BIGINT	Время выполнения BLR-запроса
EVENT_RESULT	VARCHAR(14)	Результат выполнения BLR-запроса (успешно, неуспешно, несанкционированно)

Выполнение DYN-запроса		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	EXECUTE DYN
EVENT_USER	BLOB TEXT	Пользователь, от имени которого выполняется DYN-запрос
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором выполняется DYN-запрос
TRANS_ID	BIGINT	Идентификатор транзакции
DYN_DATA	BLOB DYN	Содержимое DYN-запроса
PERF_TIME	BIGINT	Время выполнения DYN-запроса
EVENT_RESULT	VARCHAR(14)	Результат выполнения DYN-запроса (успешно, неуспешно, несанкционированно)

Присоединение к сервису		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	ATTACH SERVICE
EVENT_USER	BLOB TEXT	Пользователь, от имени которого выполняется присоединение к сервису

SVC_ID	VARCHAR(16)	Идентификатор сервиса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
EVENT_PROTOCOL	BLOB TEXT	Протокол, по которому произошло подключение к сервису
EVENT_HOSTNAME	BLOB TEXT	Имя хоста, с которого произошло подключение к сервису
EVENT_HW_ADDRESS	BLOB TEXT	Физический адрес клиента (MAC-адрес адаптера)
EVENT_DATABASE	BLOB TEXT	База данных, для которой присоединились к сервису
EVENT_RESULT	VARCHAR(14)	Результат присоединения к сервису (успешно, неуспешно, несанкционированно)

Старт сервиса		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	START SERVICE
SVC_ID	VARCHAR(16)	Идентификатор сервиса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
SVC_NAME	BLOB TEXT	Имя вызываемого сервиса
SVC_SWITCHES	BLOB TEXT	Параметры запуска сервиса
EVENT_RESULT	VARCHAR(14)	Результат старта сервиса (успешно, неуспешно, несанкционированно)

Запрос к сервису		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	QUERY SERVICE
SVC_ID	VARCHAR(16)	Идентификатор сервиса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
SVC_QUERY	BLOB TEXT	Тип запроса к сервису
EVENT_RESULT	VARCHAR(14)	Результат запроса к сервису (успешно, неуспешно, несанкционированно)

Отсоединение от сервиса		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	DETACH SERVICE
SVC_ID	VARCHAR(16)	Идентификатор сервиса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
EVENT_RESULT	VARCHAR(14)	Результат отсоединения от сервиса (успешно, неуспешно, несанкционированно)

Проверка предъявленного фактора аутентификации		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	VERIFY AUTH FACTOR
EVENT_USER	BLOB TEXT	Пользователь, проходящий аутентификацию
AUTH_FACTOR_TYPE	BLOB TEXT	Тип фактора, предъявленного при аутентификации

EVENT_RESULT	VARCHAR(14)	Результат проверки предъявленных факторов аутентификации (успешно, неуспешно, несанкционированно)

Начало выполнения триггера		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	START TRIGGER
EVENT_USER	BLOB TEXT	Пользователь, от имени которого выполняется триггер
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором выполняется триггер
TRANS_ID	BIGINT	Идентификатор транзакции
TRIG_NAME	BLOB TEXT	Название триггера
TRIG_ACTION	BLOB TEXT	Событие, на которое срабатывает триггер
EVENT_RESULT	VARCHAR(14)	Результат начала выполнения триггера (успешно, неуспешно, несанкционированно)

Завершение выполнения триггера		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	FINISH TRIGGER
EVENT_USER	BLOB TEXT	Пользователь, от имени которого выполняется триггер
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором выполняется триггер
TRANS_ID	BIGINT	Идентификатор транзакции
TRIG_NAME	BLOB TEXT	Название триггера
TRIG_ACTION	BLOB TEXT	Событие, на которое срабатывает триггер
PERF_INFO	BLOB TEXT	Статистика производительности триггера
PERF_TIME	BIGINT	Время выполнения триггера
EVENT_RESULT	VARCHAR(14)	Результат завершения выполнения триггера (успешно, неуспешно, несанкционированно)

Изменение правила разграничения доступа		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	ADD PRIVILEGE или DELETE PRIVILEGE
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором происходит изменение правил разграничения доступа
TRANS_ID	BIGINT	Идентификатор транзакции
EVENT_USER	BLOB TEXT	Пользователь, от имени которого происходит изменение правил разграничения доступа
PRIVILEGE_GRANTOR	BLOB TEXT	Пользователь, выдавший привилегию
PRIVILEGE_OBJECT	BLOB TEXT	Объект, на который выдали привилегию
PRIVILEGE_GRANTEE	BLOB TEXT	Кому выдана привилегия
PRIVILEGE	BLOB TEXT	Выданная привилегия

PRIVILEGE_OPTION	INTEGER	Наличие GRANT OPTION при выдаче привилегии
EVENT_RESULT	VARCHAR(14)	Результат изменения правил разграничения доступа (успешно, неуспешно, несанкционированно)

Ошибка		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	WARNING или ERROR
PROC_NAME	BLOB TEXT	Название функции, в которой произошла ошибка
ERROR_MESSAGE	BLOB TEXT	Сообщение об ошибке

Сборка мусора		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	SWEEP
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором произошло отключение от БД
SWEEP_OIT	BIGINT	Oldest Interesting Transaction
SWEEP_OAT	BIGINT	Oldest Active Transaction
SWEEP_OST	BIGINT	Oldest Snapshot Transaction
SWEEP_NEXT	BIGINT	Next Transaction
PERF_TIME	BIGINT	Время выполнения сборки мусора
PERF_INFO	BLOB TEXT	Статистика производительности запроса
EVENT_RESULT	VARCHAR(14)	Состояние сборки мусора (SWEEP_START | SWEEP_FINISH | SWEEP_FAILED | SWEEP_PROGRESS | UNKNOWN)

Значения некоторых полей могут быть пустыми в зависимости от типа события.

Так как в случае подключения лог-файла структура таблицы заранее известна, то при указании ключевого слова ADAPTER определение ее полей не обязательно. Если пользователь указывает тип адаптера без перечисления полей таблицы, создается таблица соответствующего адаптера со всеми возможными полями. Если же в запросе одновременно задается и тип адаптера, и структура таблицы, перечисленные поля должны быть подмножеством полей таблицы адаптера. Названия полей и их типы также жестко определяются типом адаптера. Порядок объявления полей не учитывается.

Если одно из полей задано неверно (ему не соответствует ни одно поле в таблице адаптера), в файл firebird.log записывается соответствующая ошибка и выполнение запроса прерывается.

Если структура таблицы указана верно, то не перечисленные в ней поля таблицы адаптера игнорируются.

При открытии бинарного лог-файла определяется версия его формата. Если она отличается от номера версии, которая является рабочей для данной версии Ред База Данных, в файл firebird.log записывается соответствующая ошибка

Если производится попытка использования адаптера в БД, ODS которой не поддерживает этого, в файл firebird.log записывается соответствующая ошибка.

Пример подключения журнала с набором конкретных полей:

```
CREATE TABLE log EXTERNAL FILE '/home/tester/employee.fdb.fbtrace_bin' ADAPTER
'fbtrace' (
```

```
    EVENT_TYPE CHAR(20),
    EVENT_DATABASE BLOB SUB_TYPE 1,
    EVENT_USER BLOB SUB_TYPE 1,
    EVENT_RESULT VARCHAR(14)
);
```

9.7 Агрегатный аудит

Агрегатный аудит собирает метрики, которые агрегируются по событиям и транзакциям. Метриками являются следующие значения запросов: `read`, `write`, `fetch`, `mark` и время выполнения. Аудит хранит значения метрик во время работы сервера. При выключении/перезагрузке сервера сохранённая статистика будет очищена.

9.7.1 Настройка агрегатного аудита

1. Укажите настройки для агрегатного аудита в файле `fbtrace.conf` или создайте отдельный файл конфигурации:

```
database [= /путь/к/бд]
{
  enabled = true
  format = aggtrace
  reset_counters = false
  max_sql_length = 0
  max_plan_length = 0
  max_log_size = 2048
}
```

   Опция `enabled = true` включает ведение аудита.
   Опция `format = aggtrace` указывает, что нужно использовать агрегатный аудит.
   Опция `reset_counters` определяет, сбрасывать ли значения счётчиков для события. При `reset_counters = true` значения `perf`, `count`, `succeed` и `failed` у события будут обнуляться при вызове агрегатного аудита через сервисы с любыми опциями, кроме `-atrc_clear`.
   Опция `max_sql_length` определяет максимальную допустимую длину `SQL-запроса`, которая может храниться. Запросы, длина которых больше, будут обрезаны до указанного значения. Значение 0 указывает, что длина не ограничена.
   Опция `max_plan_length` определяет максимальную допустимую длину плана `SQL-запроса`, которая может храниться. Планы, длина которых больше, будут обрезаны до указанного значения. Значение 0 указывает, что длина не ограничена.
   Опция `max_log_size` задает максимальный размер `log-файлов` в мегабайтах. Допускается значение от 5 до 4096. Если значение параметра равно 0 (по умолчанию), то размер файла журнала не ограничен.
   Настройки агрегатного аудита перечитываются для каждого соединения.

2. В `firebird.conf` в параметре `AuditTraceConfigFiles` укажите конфигурационный файл агрегатного аудита:

```
AuditTraceConfigFiles = fbtrace.conf
```

3. В `firebird.conf` для параметра `TracePlugin` укажите значение `aggtrace`.

```
TracePlugin = aggtrace
```

Можно использовать несколько плагинов. Для этого нужно указать их через запятую.

9.7.2 Запуск агрегатного аудита

Запросить значения, собранные агрегатным аудитом, можно следующей командой:

```
./rdbsvcmgr -service_mgr -user <имя пользователя> -password <пароль>
-action_aggtrace [опции]
```

Набор всех возможных опций представлен ниже.

Таблица 9.12 — Опции агрегатного аудита

Опция	Описание
`-atrc_statement`	Запросить метрики событий. Если указана какая-либо `"get"` опция (`atrc_get_new`, `atrc_get_old`, `atrc_get_all`, `atrc_with_query`, `atrc_get` или `atrc_get_updated`), то `atrc_statement` будет применена автоматически.
`-atrc_get`	Запросить метрики новых и обновлённых запросов.
`-atrc_get_updated`	Запросить метрики только обновлённых запросов т.е тех, у которых изменились значения метрик.
`-atrc_get_new`	Запросить метрики только новых запросов.
`-atrc_get_old`	Запросить метрики только старых запросов, то есть тех, которые уже запрашивались ранее, но значения метрик не изменились.
`-atrc_get_all`	Запросить метрики по всем статусам.
`-atrc_with_query`	Добавлять текст запроса в вывод.
`-atrc_with_plan`	Добавлять план запроса в вывод.
`-atrc_text <хэш запроса>|<хэш плана>`	Вывести запрос или план запроса по указанному хэшу.
`-atrc_transaction`	Запросить метрики событий транзакций.
`-atrc_get_version`	Вывести версию плагина агрегатного аудита.
`-atrc_clear`	Очистить сохранённую статистику.

9.7.3 Вывод собранных метрик

Схема вывода результата работы `aggtrace.schema.json` расположена в каталоге `doc` установки сервера.

Метрики событий

Аудит агрегирует значения по следующим параметрам:

- `dbname` - полный путь к базе данных;
- `event` - событие;
- `query` - текст запроса.

Формат вывода метрик событий:

```
[
```

```
{
  "event": "<событие>",
  "hash": "<хэш>",
  "status": "<статус>",
  "perf": [
    [
      <среднее количество страниц, считанных из страничного кэша (fetch)>,
      <минимальное количество страниц, считанных из страничного кэша (fetch)>,
      <максимальное количество страниц, считанных из страничного кэша (fetch)>,
      <общее количество страниц, считанных из страничного кэша (fetch)>
    ],
    [
      <среднее количество прочитанных (read) страниц базы данных>,
      <минимальное количество прочитанных (read) страниц базы данных>,
      <максимальное количество прочитанных (read) страниц базы данных>,
      <общее количество прочитанных (read) страниц базы данных
    ],
    [
      <среднее количество страниц, изменённых в страничном кэше (mark)>,
      <минимальное количество страниц, изменённых в страничном кэше (mark)>,
      <максимальное количество страниц, изменённых в страничном кэше (mark)>,
      <общее количество страниц, изменённых в страничном кэше (mark)>
    ],
    [
      <среднее количество страниц, записанных на диск (write)>,
      <минимальное количество страниц, записанных на диск (write)>,
      <максимальное количество страниц, записанных на диск (write)>,
      <общее количество страниц, записанных на диск (write)>
    ],
    [
      <средний объём памяти, выделенный под сортировки>,
      <минимальный объём памяти, выделенный под сортировки>,
      <максимальный объём памяти, выделенный под сортировки>,
      <общий объём памяти, выделенный под сортировки>
    ],
    [
      <средний объём памяти, выделенный под сортировки на диске>,
      <минимальный объём памяти, выделенный под сортировки на диске>,
      <максимальный объём памяти, выделенный под сортировки на диске>,
      <общий объём памяти, выделенный под сортировки на диске>
    ],
    [
      <средний объём памяти, кэшированной в процессе сортировки>,
      <минимальный объём памяти, кэшированной в процессе сортировки>,
      <максимальный объём памяти, кэшированной в процессе сортировки>,
      <общий объём памяти, кэшированной в процессе сортировки>
    ],
    [
      <среднее время выполнения (ns)>,
      <минимальное время выполнения (ns)>,
      <максимальное время выполнения (ns)>
      <общее время выполнения (ns)>,
    ]
```

```
    ],
    "count": 1,
    "succeed": 1,
    "failed": 0,
    "dbname": "<полный путь к базе данных>",
    "query_hash": "<хэш запроса>",
    "query": "<текст запроса>",
    "plan_hash": "<хэш плана запроса>",
    "plan": "<план запроса>"
  },
  ...
]
```

Описание значений в выводе:

- `event` – Название события. Отслеживаются следующие события:
  - Завершение выполнения хранимой процедуры (`FINISH PROCEDURE`);
  - Завершение выполнения триггера (`FINISH TRIGGER`);
  - Подготовка запроса (`PREPARE STATEMENT`);
  - Начало выполнения запроса (`START STATEMENT`);
  - Завершение выполнения запроса (`FINISH EXECUTE STATEMENT`);
  - Освобождение запроса или закрытие курсора (`FREE STATEMENT`);
  - Завершение выполнения хранимой функции (`FINISH FUNCTION`).
- `hash` – Хэш агрегированного значения;
- `status` – Статус запроса:
  - `new` – Запрос, метрики которого ранее не запрашивались из агрегатного трейса;
  - `old` – Старый запрос, у которого сохранённые значения мметрик не изменились;
  - `updated` – Обновлённый запрос, у которого обновились сохранённые значения метрик.
- `perf` – Собранные значения метрик;
- `count` – Количество выполнений запроса;
- `succeed` – Количество успешных выполнений;
- `failed` – Количество выполнений, завершённых с ошибкой;
- `dbname` – Полный путь к базе данных;
- `query_hash` - Хэш запроса;
- `query` – Текст запроса;
- `plan_hash`- Хэш плана запроса;
- `plan` – План запроса.

Метрики транзакций

Формат вывода метрик транзакций:

```
[
  {
    "dbname": "<база данных>",
    "hash": "<хэш>",
    «тип транзакции>":{
      "perf": [
        [
```

```
        <среднее количество страниц, считанных из страничного кэша (fetch)>,
        <минимальное количество страниц, считанных из страничного кэша (fetch)>,
        <максимальное количество страниц, считанных из страничного кэша (fetch)>,
        <общее количество страниц, считанных из страничного кэша (fetch)>
      ],
      [
        <среднее количество прочитанных (read) страниц базы данных>,
        <минимальное количество прочитанных (read) страниц базы данных>,
        <максимальное количество прочитанных (read) страниц базы данных>,
        <общее количество прочитанных (read) страниц базы данных>
      ],
      [
        <среднее количество страниц, изменённых в страничном кэше (mark)>,
        <минимальное количество страниц, изменённых в страничном кэше (mark)>,
        <максимальное количество страниц, изменённых в страничном кэше (mark)>,
        <общее количество страниц, изменённых в страничном кэше (mark)>
      ],
      [
        <среднее количество страниц, записанных на диск (write)>,
        <минимальное количество страниц, записанных на диск (write)>,
        <максимальное количество страниц, записанных на диск (write)>,
        <общее количество страниц, записанных на диск (write)>
      ],
      [
        <средний объём памяти, выделенный под сортировки>,
        <минимальный объём памяти, выделенный под сортировки>,
        <максимальный объём памяти, выделенный под сортировки>,
        <общий объём памяти, выделенный под сортировки>
      ],
      [
        <средний объём памяти, выделенный под сортировки на диске>,
        <минимальный объём памяти, выделенный под сортировки на диске>,
        <максимальный объём памяти, выделенный под сортировки на диске>,
        <общий объём памяти, выделенный под сортировки на диске>
      ],
      [
        <средний объём памяти, кэшированной в процессе сортировки>,
        <минимальный объём памяти, кэшированной в процессе сортировки>,
        <максимальный объём памяти, кэшированной в процессе сортировки>,
        <общий объём памяти, кэшированной в процессе сортировки>
      ],
      [
        <среднее время выполнения (ns)>,
        <минимальное время выполнения (ns)>,
        <максимальное время выполнения (ns)>
        <общее время выполнения (ns)>,
      ]
    ],       "count": <общее количество событий>,
    "succeed": <количество успешно выполненных событий>,
    "failed": <количество событий, завершённых сошибкой>
  },
  "ROLLBACK_TRANSACTION":{
    "perf": [...],
```

```
      "count": <общее количество событий>
    },
    "ROLLBACK_RETAINING:{
      "perf": [...],
      "count": <общее количество событий>
    },
    "COMMIT_TRANSACTION:{
      "perf": [...],
      "count": <общее количество событий>
    },
    "COMMIT_RETAINING:{
      "perf": [...],
      "count": <общее количество событий>
    },
    "FAILED ROLLBACK_TRANSACTION:{
      "perf": [...],
      "count": <общее количество событий>
    },
    "FAILED ROLLBACK_RETAINING:{
      "perf": [...],
      "count": <общее количество событий>
    },
    "FAILED COMMIT_TRANSACTION:{
      "perf": [...],
      "count": <общее количество событий>
    },
    "FAILED COMMIT_RETAINING:{
      "perf": [...],
      "count": <общее количество событий>
    }
  }
]
```

Описание значений в выводе:

- `dbname` – Полный путь к базе данных;
- `hash` – Хэш агрегированного значения;
- `тип транзакции` - Выполненная транзакция;
- `perf` – Собранные значения метрик для транзакции;
- `count` – Количество запусков транзакции;
- `succeed` – Количество успешных выполнений;
- `failed` – Количество выполнений, завершённых с ошибкой;

Вывод версии

Формат вывода версии:

```
{
  "version": "<версия агрегатного аудита>"
}
```

9.8 Сбор метрик в формате Prometheus

Приложение для экспорта метрик СУБД доступно по ссылке https://github.com/red-soft-ru/rdb_prometheus_exporter.

Для его работы требуется написать конфигурационный файл в формате JSON следующего содержания:

```
{
"port": 8000,
"login": "SYSDBA
"password": "masterkey
"RDB_port": 3050,
"utilities": "/opt/RedDatabase/bin"
"databases": {
"[db_alias]": "localhost:[insert_path_to_database]
"[another_db_alias]": "localhost:[insert_path_to_another_database]"
},
"trace": "[insert_path_to_fdbtrace_text]"
}
```

Где:

- `port` - порт, на который будут отправляться метрики;
- `login` - имя пользователя, от которого экспортёр будет подключаться к СУБД. Желательно использовать пользователя с административными привилегиями для мониторинга всех подключений;
- `password` - пароль пользователя;
- `RDB_port` - порт, на котором работает СУБД;
- `utilities` - путь к утилитам СУБД;
- `databases` - перечень баз данных для мониторинга;
- `trace` - путь к файлу аудита для анализа статистики выполняемых запросов.

Конфигурационный файл должен называться `exporter_conf.json` и находиться в корневом каталоге утилиты экспортёра.

Запущенная утилита собирает следующие метрики:

Таблица 9.13 — Собираемые метрики

`db_size`	Размер базы данных
`diff_oldt_nt`	Разница между значениями `Oldest transaction` и `Next transaction`
`active_users`	Количество активных пользователей
`mon_io_stats`	Статистика ввода/вывода: количество прочитанных и измененных страниц в кэше и на диске
`mon_memory_usage`	Статистика по использованию памяти
`mon_database`	Информация о базе данных
`mon_attachment`	Информация об активных подключениях к базе данных
`mon_transaction`	Информация о запущенных транзакциях

`mon_statement`	Информация о запросах к базе данных
`mon_call_stack`	Стеки вызовов хранимых процедур и триггеров
`system_memory`	Информация о системной оперативной памяти: • `used`: занятая; • `available`: свободная; • `total`: общий объем.
`system_cpu`	Информация об использовании процессора: • `percent`: текущая загрузка в процентах; • `frequency`: частота процессора.
`trace_statements`	Информация о выполненных в СУБД запросах: • `OK`: количество успешно выполненных запросов; • `FAIL`: количество запросов, завершившихся с ошибкой; • `EXEC_TIME`: время выполнения всех запросов.

Глава 10

Встроенный сервер

10.1 Общие сведения

В СУБД Ред База Данных существует специальный режим работы, называемый «встроенным» (`embedded`). Этот режим предназначен для прямого локального доступа приложения клиента к файлу базы данных минуя клиент-серверное сетевое подключение. При этом отдельная настройка сервера не требуется.

10.2 Установка embedded сервера

В ОС Windows

Встроенный режим не требует процедуры установки СУБД как таковой. Загрузите с официального сайта Ред Базы Данных архив вида: `bin/windows/x86_64/RedDatabase-OE-3.0.X.X-windows-x86_64.zip` для 64-х разрядной версии, или `bin/windows/x86/RedDatabase-OE-3.0.X.X-windows-x86.zip` для 32-х битной версии. Обратите внимание, что битность дистрибутива должна совпадать с битностью вашей ОС: вы должны использовать `x86_64` версию, если вы компилируете свою программу под 64-бит, и x86, если вы пишете 32-битные программы.

Распакуйте соответствующий дистрибутив в рабочий каталог вашей программы. Для работы встроенного режима достаточно скопировать файлы из корневого каталога дистрибутива СУБД Ред База Данных, а также все файлы из папки `/intl` и `/plugins` (достаточно файла `engine12.dll`).

В ОС Linux

Данная установка производится аналогично установке под Windows.В данном режиме в качестве провайдера используется библиотека `/plugins.libengine12.so`. Для работы встроенного сервера достаточно скопировать файлы из корневого каталога дистрибутива СУБД Ред База Данных, и папок `/bin`, `/intl` и `/plugins` (достаточно файла `libengine12.so`).

В системе должна быть установлена библиотека `libicu`.

В случае возникновения ошибки подключения:

```
«error while loading shared libraries: libtinfo.so.5»
```

потребуется установить пакет `ncurses-compat-libs`.

10.3 Подключение

Для подключения в режиме встроенного сервера через API Ред Базы Данных клиенту следует указать в строке подключения *локальный путь* (без хоста) к файлу базы данных (или его алиас, прописанный в файле `databases.conf`).

Встроенный сервер не требует аутентификации. Тем не менее, имя пользователя и, если необходимо, роль могут быть указаны в параметрах подключения, поскольку они используются для контроля доступа к объектам базы данных. По умолчанию встроенный сервер будет использовать имя текущего пользователя компьютера.

Можно легко переключаться с `embedded` на полноценный клиент-серверный режим без изменения строк кода приложения, только поменяв строку подключения к базе данных.

10.4 Несколько одновременных подключений

До версии 2.6 `embedded` сервер в Windows не мог подключиться к базе данных, которая уже имела соединения с полноценным сервером или существующим экземпляром встроенного. Верно и обратное: полноценный сервер не мог подключиться к базе данных, к которой был подключен встроенный сервер. Так было потому, что предыдущие версии встроенного сервера в Windows были реализованы как суперсервер, который по разным причинам требует монопольной блокировки файла базы данных.В версии 2.6 `embedded` сервер мог совместно использовать базу данных с другим встроенным сервером и с автономным сервером `Superclassic` или `Classic` на всех платформах.

По умолчанию встроенный сервер запускается на всех доступных платформах(Windows, Linux)в режиме `SuperServer`.

Настройка архитектуры находится в конфигурационном файле `firebird.conf`:

```
ServerMode = Super
```

В этом режиме встроенный сервер создает эксклюзивную блокировку для файла базы данных на подключение и, в то время как он подключен, предотвращает подключения от других экземпляров. При такой конфигурации невозможно, например, иметь клиент-серверные подключения одновременно с клиентами браузера, подключенными к одной и той же базе данных через приложение интрасети, использующее встроенный механизм.

Решение состоит в том, чтобы запустить встроенный сервер как процесс `[Super]Classic` совместно с вашим сетевым сервером работающим также в режимах `Superclassic` или `Classic`. Раскомментируйте параметр `ServerMode` в конфигурационном файле `firebirds.conf` вашего встроенного сервера и установите для него значение `Classic` или `SuperClassic`:

```
ServerMode = Classic
```

Обратите внимание, что для встроенного сервера режимы `Classic` и `SuperClassic` эквивалентны.

10.5 Запуск инструментов администрирования

Нет необходимости создавать отдельную файловую среду, если вы планируете использовать встроенное соединение для запуска инструментов администрирования, таких как `gbak` или `gfix`, или запуска привилегированного DDL в режимах `Classic` или `Superclassic`. Просто сохраните исходную конфигурацию провайдеров и войдите в систему, используя в строке подключения *локальный путь* к файлу базы данных и любое имя пользователя, необходимое для выполнения задач.

10.6 Примеры подключения

Для консольного подключения к базе данных в режиме встроенного сервера можно использовать утилиту `isql` из состава дистрибутива.

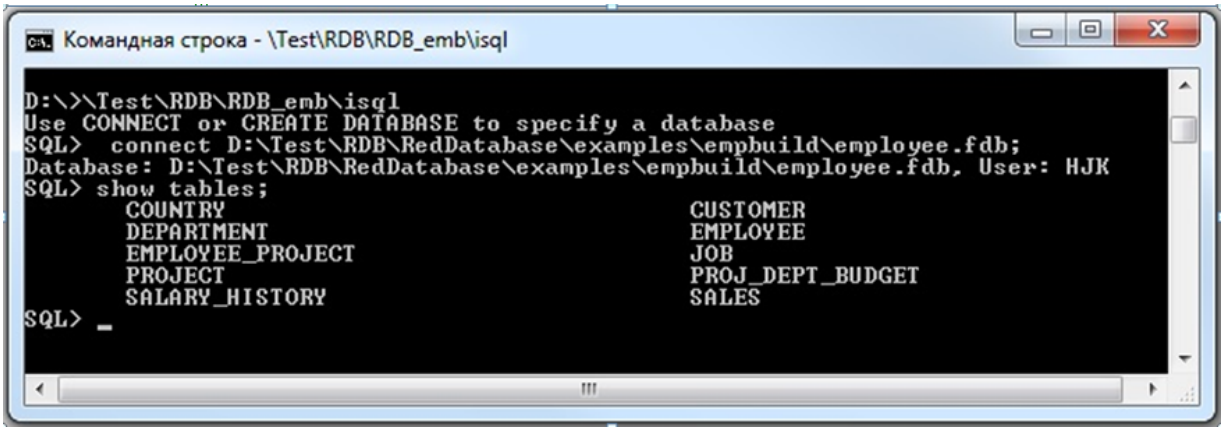


Рисунок 10.1 — Консольное подключение в режиме встроенного сервера через утилиту isql для Windows

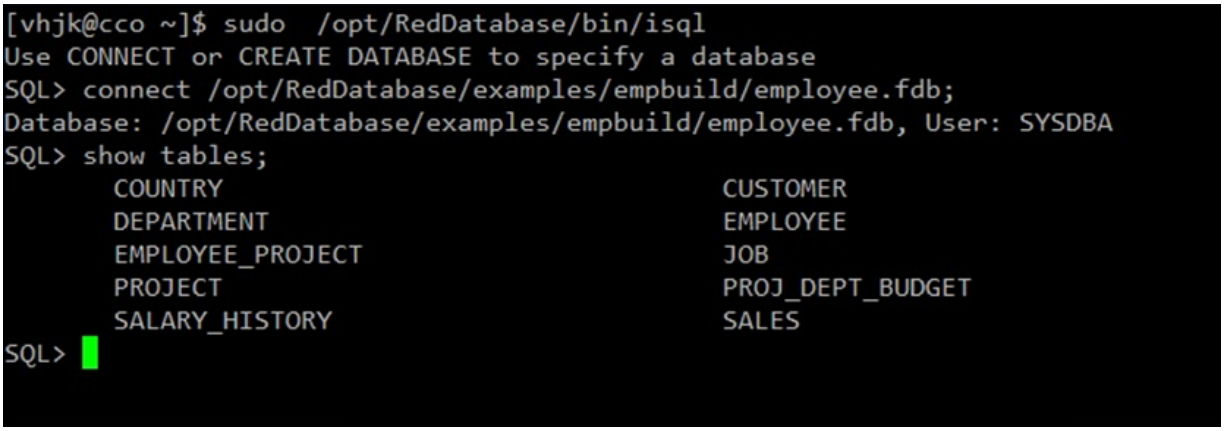


Рисунок 10.2 — Консольное подключение в режиме встроенного сервера через утилиту isql для Linux

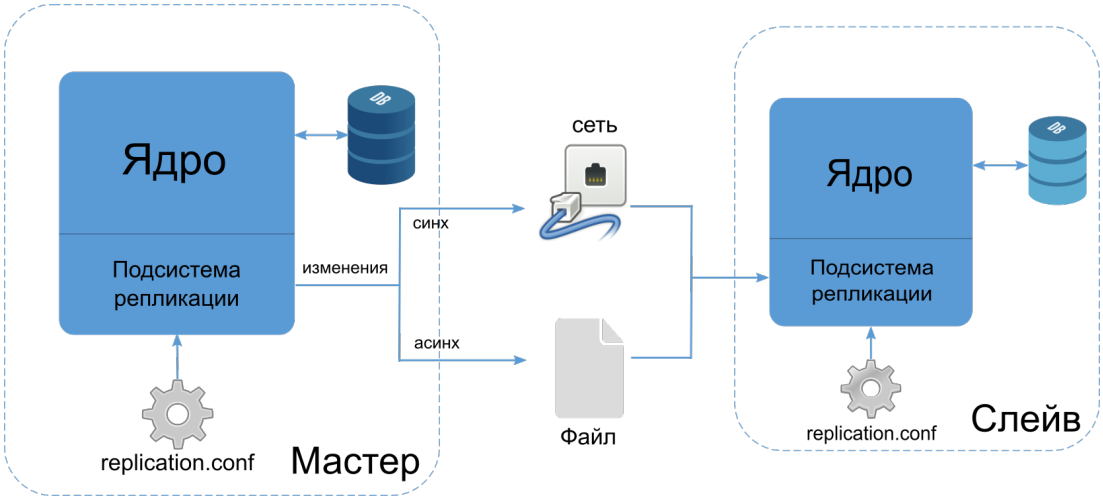
Рисунок 10.3 — Пример кода подключения к базе данных на языке Python.

Глава 11

Репликация

11.1 Особенности репликации

Большая часть репликаторов, написанных для Firebird, представляет собой внешние по отношению к серверу приложения. Они хорошо подходят для миграции данных между различными базами данных или даже между различными СУБД, но плохо подходят для поддержания копии БД в рамках решения задачи создания отказоустойчивого кластера. Основной проблемой при этом является производительность, т.к. сторонние средства создают дополнительную нагрузку при синхронизации данных, причём она обычно избыточна и существенно замедляет работу СУБД. Кроме того, не все средства репликации могут гарантировать идентичность данных в основной базе и в копии при возникновении сбоя.



Встроенная репликация предназначена для обеспечения повышенной отказоустойчивости в случае повреждения физической структуры файла базы данных, вызванной техническим сбоем оборудования, программным сбоем операционной системы или самой СУБД. Она подразумевает перенос любых изменений данных с основного рабочего сервера на один или несколько резервных серверов, гарантируя, таким образом, их идентичность с точки зрения хранящихся на них данных. При этом главный сервер принято называть мастером, а резервные – слейвами.

Элемент репликации представляет собой запись в таблице, любые изменения которой передаются на резервные сервера. Также передаются и контекстные изменения, а именно соединения и транзакции, в которых происходит изменение записи. Таким образом, основными событиями репликации являются:

- создание и завершение соединений;
- начало, конец и откат каждой транзакции;
- начало, конец и откат точки сохранения;
- вставка, обновление и удаление записей;
- изменение генераторов.

Однозначное соответствие строк на основном и резервном серверах определяется уникальным индексом. Механизм репликации не содержит каких-либо средств, гарантирующих логическую

непротиворечивость данных между основной и резервными копиями баз данных. Полноценно реплицируются только таблицы, содержащие уникальный индекс, например, первичный ключ. В случае отсутствия или неактивности такого индекса возможна репликация только операций вставки записи.

В процессе работы репликации возможны конфликты. Конфликтом считается обнаруженное несоответствие между данными на основном и резервном серверах. Примерами конфликтов являются:

- наличие строки в реплике при попытке ее вставки;
- отсутствие строки в реплике при ее изменении или удалении;
- наличие транзакции при попытке её запуска;
- отсутствие транзакции при её завершении.

Конфликты DML-операций в режиме «приоритет мастера» будут решены следующим образом:

- вставка уже существующей на реплике записи приведёт к её обновлению;
- обновление несуществующей записи превращается во вставку;
- удаление несуществующей записи игнорируется.

Если приоритет мастера отключен, или произошел конфликт другого вида, это расценивается как ошибка репликации, которая приведёт либо к отключению репликации и продолжению работы мастера в штатном режиме, либо к отмене операции, во время которой произошла ошибка и сообщению о ней пользователю.

11.2 Основные понятия

Мастер — сервер, на котором находится база данных, которая будет реплицироваться.
Слейв — сервер, на который будет реплицироваться база данных с мастера.
Реплика — база данных на слейве.

11.3 Синхронная и асинхронная репликация

> Синхронная и асинхронная репликация доступна только в промышленной редакции. Подробнее различия функционала редакций описаны в разделе "Редакции СУБД Ред База Данных 3.0".

Существует два основных режима работы репликации: синхронный и асинхронный. В первом случае при наступлении события репликации в мастер-базе, оно должно быть также выполнено на всех слейв-базах для того, чтобы событие считалось завершенным. Логически это означает, что существует лишь одна версия данных. Основной недостаток синхронной репликации – она создаёт дополнительную задержку в работе мастера, т.к. СУБД должна дождаться ответа от реплики, чтобы считать операцию завершенной.

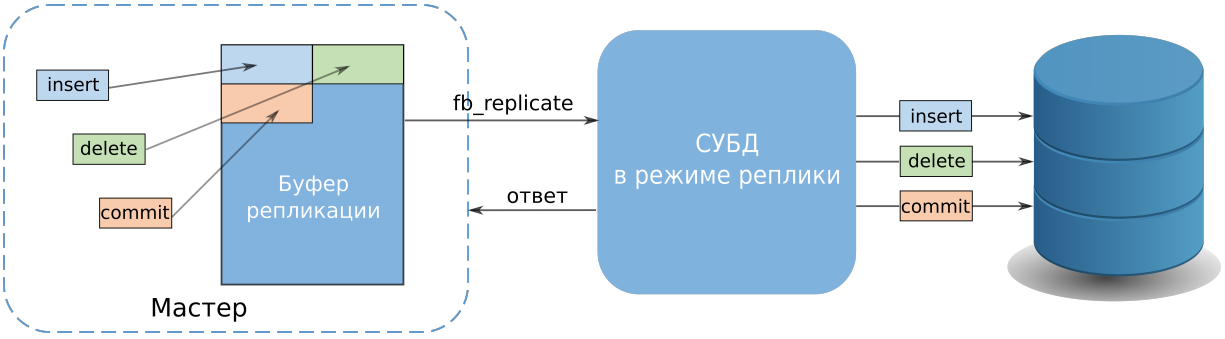
В случае асинхронной репликации обновления мастер-базы пишутся в журналы, которые распространяются на реплику спустя некоторое время, а не в той же транзакции. Таким образом, при асинхронной репликации уменьшаются простои мастера, но при этом вводится задержка, в течение которой реплики могут быть фактически неидентичными. Кроме того, больше вероятность конфликтов из-за того, что процесс копирования журналов репликации выполняется сторонними средствами.

Рассмотрим подробнее принципы работы и особенности разных режимов репликации.

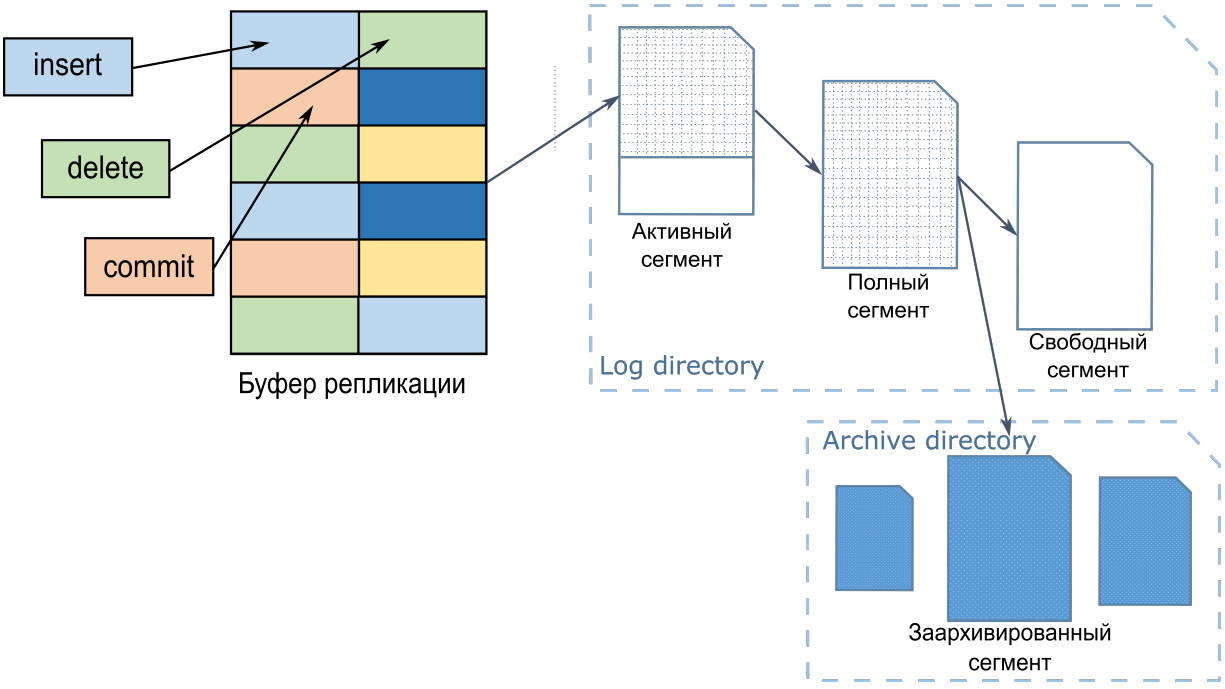
При синхронной репликации в процессе подключения к основному серверу проверяется наличие и доступность резервных серверов, после чего устанавливается с ними постоянное соединение. При ошибке подключения к слейву подключение к мастеру также завершается с ошибкой. Подключение к слейву может выполняться от имени текущего пользователя, либо от имени пользователя, заданного в конфигурации.

При наступлении событий в рамках транзакции, они записываются в буфер репликации, который передаётся на слейв либо при его заполнении, либо при наступлении определённых событий, например, коммита транзакции или завершения соединения. Слейв, получив репликационный пакет, должен выполнить все указанные в нём действия. При ошибке в любом из них применение пакета прекращается, а сообщение об ошибке репликации отправляется на мастер.

При синхронной репликации особую проблему составляет коммит транзакции. При наличии сложной отложенной работы время коммита может существенно увеличиться, если он будет последовательно выполняться на мастере и на слейве. Поэтому при репликации коммита на слейв посылается специальное событие подготовки коммита, и отложенная работа на мастере и слейве выполняется параллельно.



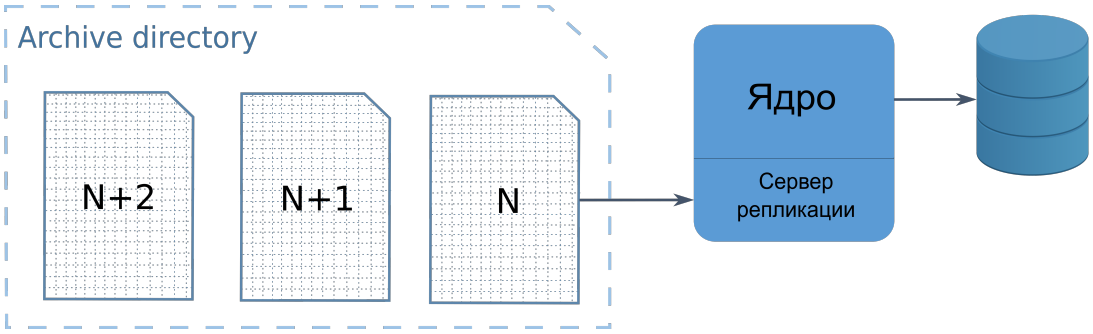
Асинхронная репликация также использует буферизацию событий репликации, но не посылает содержимое буфера на слейв, а пишет его в журнал репликации. Он представляет собой набор бинарных файлов (сегментов журнала) фиксированного размера, хранящиеся в заданной в конфигурации репликации директории. СУБД сбрасывает буферы с событиями репликации в текущий (активный) сегмент до тех пор, пока он не будет заполнен. После этого отдельный поток выполняет архивацию сегмента с помощью внешней команды. По окончании архивации сегмент считается свободным и может снова использоваться для записи событий репликации.



В данной модели существует четыре возможных состояния сегментов журнала:

- `USED` – сегмент в данный момент используется. В конкретный момент времени он единственный. Когда сегмент достигает заданного размера, он считается заполненным и запись в него прекращается. То же самое происходит при простое сегмента в течении минуты.
- `FULL` – сегмент заполнен и ждет архивации. Ей занимается отдельный поток, который оповещается с помощью семафора. Также возможна ручная архивация с помощью утилиты rdblogmgr.
- `ARCH` – сегмент в процессе архивации.
- `FREE` – помечается после архивации, как возможный к использованию. Как только сегмент `USED` переходит в состояние `FULL`, ищутся сегменты в состоянии `FREE` с целью использования их снова. Если такие не найдены, создается новый.

Применение журнала к реплике выполняется сервером репликации. В случае Суперсервера или Суперклассика, это отдельный поток в запущенной СУБД. Он читает архивные сегменты из указанного каталога и применяет их к БД. Если в системе установлен и работает классик, то сервер репликации запускается отдельным процессом суперклассика с ключом `-R`. Также сегменты журнала могут быть применены вручную утилитой `rdblogmgr`.



11.4 Настройка системы репликации

Файл конфигурации

Для настройки системы репликации используется файл `replication.conf`, находящийся в корневом каталоге СУБД. По умолчанию он содержит блок `<database>`, параметры которого закомментированы. Для обозначения комментариев используется символ `#`. Конфигурационный файл `replication.conf` считывается СУБД в начале процесса подключения к БД.

Файл конфигурации разбит на блоки:

- `database` - основной блок с параметрами по умолчанию;
- `database = <путь к базе данных>` - блок с параметрами для базы, требующей репликации;
- `replica = <путь до слейв-базы>` - блок с параметрами для слейва (в случае асинхронной репликации).

Первое слово в строке внутри блока, начинающейся не с символа комментария, считается названием параметра. Справа от имени параметра, после символа равенства, указывается значение параметра.

Поддерживаются следующие параметры:

buffer_size

Указывает размер накопляемого буфера операций на мастер-базе перед отправкой их на слейв-базы или записью в журнал. Этот параметр не влияет на такие операции, как `commit` или `rollback`; при выполнении этих операций буфер отправляется немедленно и ожидается подтверждение от всех удачного исполнения.

```
buffer_size = 1048576 # 1MB
```

disable_on_error

Параметр может настраиваться только для синхронного режима. Если установлен в `true`, то при возникновении ошибки, связанной с репликацией, мастер отключается от слейва и продолжает работать без репликации. В противном случае (поведение по умолчанию) мастер сообщит об ошибке.

В асинхронном режиме при возникновении ошибки репликации выдается сообщение, слейв отключается, дальнейшее применение журналов невозможно.

```
disable_on_error = true
```

compress_records

Если установлен в `true`, то перед отправкой записи будут сжиматься алгоритмом `RLE`. Необходим для сжатия трафика, передаваемого по сети. Настраивается только для синхронного режима. Для асинхронного всегда `true`.

```
compress_records = true
```

master_priority

Параметр может настраиваться только для синхронного режима. Для асинхронного он всегда `true`. Если установлен в `true`, то при добавлении, изменении или удалении записей, которые как-то конфликтуют с мастер-базой, будет выбираться стратегия с изменением операции. Например, если новая запись добавляется в таблицу, а на слейве такая запись уже существует, то запись изменяется; если запись изменяется, а на слейве ее не существует, то она добавляется; если запись удаляется, а на слейве ее нет, то она игнорируется.

```
master_priority = true
```

include_filter

Строковый параметр в формате регулярного выражения в синтаксисе SQL, который задает, какие таблицы базы данных необходимо включить в репликацию. По умолчанию реплицируются все таблицы.

```
include_filter = (W$|ORD$)%
```

exclude_filter

Строковый параметр в формате регулярного выражения в синтаксисе SQL, который задает, какие таблицы базы данных необходимо исключить из репликации. По умолчанию реплицируются все таблицы.

```
exclude_filter = (TEMP$|LOG$)%|(SYS_DB_LOG)
```

exclude_without_pk

Если параметр включен, то из репликации будут исключены таблицы без первичного ключа (или уникального индекса). Если параметр выключен (по умолчанию), то при репликации таблиц без первичного ключа выдается ошибка.

```
exclude_without_pk = true
```

alert_command

Внешняя программа, которая выполняется, когда возникает критическая ошибка. Эта команда выполняется один раз в каждой неудачной сессии репликации. Обратите внимание, что программа выполняется синхронно, и сервер ждет ее завершения, прежде чем продолжить свою деятельность.

```
alert_command = crash_replication.bat
```

log_directory

Используется только для асинхронного режима. На мастере данный параметр указывает на каталог для хранения файлов журнала транзакций. Если значение параметра пустое, то ведение журнала отключено. Если ведение журнала включено, то параметр `compress_records` (см. выше) тоже считается включенным.

В секции `replica = <путь до слейв-базы>` параметр `log_directory` указывает, где искать архивные журналы для применения к слейву, находящемуся по пути `<путь до слейв-базы>`.

```
log_directory = d:\replication_directory
```

log_file_prefix

Префикс для имен файлов журнала репликации. Если параметр не указан, то в качестве префикса файлов журнала будет имя файла базы данных (без пути). Используется только для асинхронного режима.

```
log_file_prefix = warehouse_log
```

log_segment_size

Максимально допустимый размер для одного сегмента репликации. Он должен быть, по крайней мере, в два раза больше размера, указанного в `buffer_size`. Используется только для асинхронного режима.

```
log_segment_size = 16777216 # 16MB
```

log_segment_count

Максимально допустимое число полных сегментов репликации. После того, как предел достигнут, процесс репликации задерживается на `log_archive_timeout` секунд (см. ниже), чтобы догнать процесс архивирования файлов сегментов. Если какой-либо из полных сегментов не архивируется и отмечен для повторного использования в течение тайм-аута, то репликация завершается с ошибкой. Ноль означает неограниченное количество сегментов, ожидающих архивирования. Используется только для асинхронного режима.

```
log_segment_count = 8
```

log_archive_directory

Данный параметр определяет директорию, куда будут архивироваться сегменты журнала (использоваться для подстановки в переменную `$(archpathname)`, см. ниже). Используется только для асинхронного режима.

```
log_archive_directory = /temp/archive_logs
```

log_archive_command

Используется только для асинхронного режима. Программа, которая выполняется, когда некоторый сегмент репликации становится полным и нуждается в архивировании. Эта программа должна возвращать ноль, только если архивирование успешно выполнено. Например, он должен вернуться не ноль, если целевой архив уже существует.

Доступны специальные предопределенные переменные:

- `$(logfilename)` - имя файла (без пути) сегмента, который нужно заархивировать;
- `$(logpathname)` - полный путь к файлу сегмента, который нужно заархивировать;
- `$(archfilename)` - имя файла (без пути) для заархивированного сегмента;
- `$(archpathname)` - полный путь к файлу заархивированного сегмента.

```
log_archive_command = "test ! -s $(archpathname) && cp $(logpathname)
$(archpathname)"
```

```
log_archive_command = "copy $(logpathname) $(archpathname)"
```

log_archive_timeout

Время ожидания (в секундах) заполнения активного сегмента. Если в течении этого времени не было никаких изменений в базе данных, то текущий сегмент помечается как полный и отправляется на архивацию.

Это позволяет минимизировать разрыв репликации, если база данных редко изменяется. Значение 0 означает отсутствие промежуточного архивирования, т.е. сегменты архивируются только после достижения их максимального размера (определяется `log_segment_size`).

Используется только для асинхронного режима. По умолчанию значение параметра равно 60.

```
log_archive_timeout = 30
```

log_group_flush_delay

Параметр для задержки (в миллисекундах) между коммитами. По умолчанию параметр принимает значение 0, то есть во время коммита происходит скидывание буфера в активный сегмент. Этот параметр задает задержку перед скидыванием буфера. Так, коммиты будут писаться в буфер до тех пор, пока он не заполнится или пока не будет простоя между коммитами. Используется только для асинхронного режима.

```
log_group_flush_delay = 30
```

replica_database

Задание этого параметра включает синхронную репликацию. Он указывает на базу данных реплики.

Значение параметра можно указывать в двух форматах. В первом варианте значение указывается одной строкой и есть ограничение на использование символа `@` в имени пользователя и пароле:

```
replica_database = [<login>:<password>@]<адрес сервера>:<путь к БД>
```

Во втором варианте `replica_database` определяется в виде подсекции:

```
replica_database = <адрес сервера>:<путь к БД>
    username[_env | _file] = <имя пользователя> | <переменная окружения> |
```

```
<путь к файлу>
    password[_env | _file] = <пароль> | <переменная окружения> | <путь к
файлу>
```

При использовании суффикса `_env` значение будет взято из заданной переменной окружения.

При использовании суффикса `_file` значение будет взято из первой строки определённого файла.

Можно указывать как абсолютный, так и относительный путь (относительно корневого каталога инсталляции Ред Базы Данных). Если такого файла не существует, либо файл пустой, то в лог будут записаны следующие ошибки:

```
"<replica> specifies missing file: <...>/replicaton_pass" -- файл не найден
"<replica> specifies empty file: <...>/replication_pass" -- файл пустой
```

В архитектурах `SuperServer` и `SuperClassic` подключение к базе данных реплики происходит при первом соединении пользователя с мастером. А отключение происходит, когда последний пользователь отсоединяется от базы данных мастера.

В архитектуре `Classic` каждый серверный процесс сохраняет свое собственное активное соединение с базой данных реплики.

owner_auth

В блоке `replica = <путь до слейв-базы>` параметр `owner_auth` является идентификатором для авторизации к слейву. Значение параметра имеет следующий формат:

```
<login>:<password>
```

Для асинхронного режима данный параметр можно не задавать в случае, если в системе установлены контекстные переменные `ISC_USER` и `ISC_PASSWORD`, и сервер для применения журналов репликации запущен с этими контекстными переменными.

master_database

Данный параметр в блоке `replica = <путь до слейв-базы>` содержит строку подключения к мастеру репликации следующего формата:

```
[<login>:<password>@]<адрес сервера>:<путь к БД>
```

db_copy_command

Консольная команда для пересоздания слейва. Данная команда должна возвращать 0, только если копирование было выполнено успешно. Доступны следующие переменные:

- `$(masterdb)` — строка соединения для доступа к мастер-базе;
- `$(masteruser)` — имя пользователя для подключения к мастер-базе;
- `$(masterpwd)` — пароль для подключения к мастер-базе;
- `$(replicadb)` — полный путь к БД реплики;
- `$(replicauser)` — имя пользователя для подключения к слейву;
- `$(replicapwd)` — пароль для подключения к слейву;
- `$(guid)` — GUID мастер-базы.

Пример команды, которая делает копию базы с помощью `nbackup`, устанавливают реплику и копирует базу по пути `replicadb`:

- для Windows:

```
db_copy_command = "del d:\temp\repl\tpcc-rdb.fdb && nbackup -u
$(masteruser) -p $(masterpwd) -d on -b 0 $(masterdb)
d:\temp\repl\tpcc-rdb.fdb && nbackup -f d:\temp\repl\tpcc-rdb.fdb &&
gfix -user $(replicauser) -password $(replicapwd) -replica $(guid)
d:\temp\repl\tpcc-rdb.fdb && move /Y d:\temp\repl\tpcc-rdb.fdb
$(replicadb)"
```

- для Linux:

```
db_copy_command = "export
FIREBIRD=home/dimitr/firebird/rdb/trunk/firebird/gen/ firebird; export
LD_LIBRARY_PATH=/home/dimitr/firebird/rdb/trunk/firebird/gen/
firebird/lib; export PATH=$FIREBIRD/bin:$PATH; rm -f /tmp/tpcc-rdb.fdb
&& nbackup -u $(masteruser) -p $(masterpwd) -d on -b 0 $(masterdb)
/tmp/tpcc-rdb.fdb && nbackup -f /tmp/tpcc-rdb.fdb && gfix -user
$(replicauser) -password $(replicapwd) -replica $(guid)
/tmp/tpcc-rdb.fdb && mv -f /tmp/tpcc-rdb.fdb $(replicadb)"
```

Настройка базы данных для асинхронной репликации

Асинхронная репликация — процесс, при котором все обновления в главной базе данных пишутся в журналы, которые впоследствии архивируются и применяются к базе-реплике. Главная задача в настройке асинхронной репликации — обеспечить доставку журналов с главной базы на реплику.

Подготовительные настройки

Установите СУБД «Ред База Данных» промышленной (Enterprise) редакции на мастер и на слейв. На слейве выберите архитектуру СУБД Super. Установку и настройку необходимо выполнять под учетной записью с правами суперпользователя.

На мастере и на слейве требуется установить `nfs-utils`:

```
sudo yum install nfs-utils
```

На реплике требуется поставить `autofs`:

```
sudo yum install autofs
```

Настройка доступности журналов репликации

1. Запустите службу `rpcbind` на слейве и на мастере и `nfs-server` на мастере:

```
sudo systemctl start rpcbind
sudo systemctl start nfs-server
```

2. Создайте на мастере папки для хранения логов и архивов репликации. Например, так:

```
sudo mkdir -p /home/logsrdb/logs /home/logsrdb/arch
```

Назначьте на них права:

```
sudo chmod -R 777 /home/logsrdb
sudo chown -R firebird:firebird /home/logsrdb
```

3. На мастере отредактируйте файл `/etc/exports` для того, чтобы расшарить папку, которая будет доступна со слейва. Для этого в `/etc/exports` нужно добавить строку по образцу:

```
<путь_до_расшаренной_папки> <ip_слейва>/<маска_сети>(атрибуты)
```

Например:

```
/home/logsrdb/ 10.81.1.0/24(rw,sync)
```

Так же необходимо добавить правила в firewall на мастере:

```
sudo firewall-cmd -permanent -zone=public -add-service=nfs
sudo firewall-cmd -permanent -zone=public -add-service=mountd
sudo firewall-cmd -permanent -zone=public -add-service=rpc-bind
```

Откройте порт 3050 (порт по умолчанию для СУБД Ред База Данных) на мастере

```
sudo firewall-cmd -permanent -add-port=3050/tcp
```

Перезапустите firewall

```
sudo firewall-cmd -reload
```

Перезапустите `nfs-server` и добавьте его в автозагрузку

```
sudo systemctl restart nfs-server
sudo systemctl enable nfs-server
```

4. На слейве необходимо организовать автомонтирование каталога с мастера, для этого:
   4.1 Добавьте строку для монтирования в файл `/etc/auto.master`:

```
/mnt/repl /etc/auto.nfs -ghost
```

   4.2 На слейве создайте файл `/etc/auto.nfs` и впишите туда строку

```
logsrdb -rw,soft,intr 10.81.1.187 (ip-адрес мастера):/home/logsrdb
(путь до логов на мастере)
```

   4.3 Запустите `autofs` и добавьте его в автозагрузку.

```
sudo systemctl start autofs.service
sudo systemctl enable autofs.service
```

   4.4 На слейве проверьте доступность каталогов, которые были расшарены на мастере, например выполнив команду на слейве:

```
ll /mnt/repl/logsrdb/
```

   Где `/mnt/repl/logsrdb/` — путь монтирования каталогов из файлов `/etc/auto.master` и `/etc/auto.nfs/` При этом, должны отобразиться каталоги `arch` и `logs` с правами

на них, например:

```
drwxrwxrwx. 2 firebird firebird 52 июн 15 15:03 arch
drwxrwxrwx. 2 firebird firebird 29 июн 11 12:27 logs
```

Настройка репликации

1. Остановите СУБД на слейве:

```
sudo systemctl stop firebird
```

2. Настройте `replication.conf` на слейве. Заполните секцию `replica` по образцу:

```
replica = <путь_до_бд_реплики>
{
    owner_auth = <имя_пользователя_администратора>:<пароль_администратора>
    log_directory = <путь_папки_с_архивированными_логами_с_мастера>
    master_database =
        <имя_пользователя_администратора>:<пароль_администратора>@
        <ip_адрес_мастера >:<путь_до_БД_на_мастере>
}
```

Например:

```
replica = /opt/db/red.fdb
{
    owner_auth = sysdba:masterkey
    log_directory = /mnt/repl/logsrdb/arch
    master_database = sysdba:masterkey@10.81.1.187:/opt/db/red.fdb
}
```

Если реплицируемых баз несколько, например две, то создайте и заполните второй блок `replica` по аналогии с первым.

3. На слейве необходимо создать файл `replication.log` и назначить на него права пользователя `firebird`:

```
sudo touch /opt/RedDatabase/replication.log
sudo chmod 766 /opt/RedDatabase/replication.log
sudo chown firebird:firebird /opt/RedDatabase/replication.log
```

4. Настройте `replication.conf` на мастере. Заполните по образцу секцию `database`, закомментировав ее:

```
database = <путь_до_базы_данных_на_мастере>
{
    log_directory = <путь_до_логов>
    log_archive_directory = <путь_до_папки_с_архивами_на_мастере>
    log_archive_command = "test ! -s $(archpathname) &&
                           cp $(logpathname) $(archpathname)"
}
```

При этом если в базе данных присутствуют таблицы без первичного ключа или уникального индекса, то следует включить следующий параметр — `exclude_without_pk`. (т.е. добавить `exclude_without_pk = true`) в секцию `database`.

Например:

```
#database = /opt/db/red.fdb
#{
    #exclude_without_pk = true # см. примечание выше по этому параметру
    #log_directory = /home/logsrdb/logs
    #log_archive_directory = /home/logsrdb/arch
    #log_archive_command = "test ! -s $(archpathname) &&
                            cp $(logpathname) $(archpathname)"
#}
```

5. Остановите СУБД на мастере:

```
sudo systemctl stop firebird
```

6. Убедитесь, что процессов СУБД не осталось.

```
sudo ps -ef | grep rdbserver
```

или

```
sudo lsof <путь_до_базы_данных>
```

Если остались незавершенные процессы, то принудительно завершите их:

```
sudo killall rdbserver
```

7. Раскомментируйте настроенные параметры на мастере в `replication.conf`, секцию `database`.
8. Заблокируйте базу данных на мастере с помощью `nbackup`:

```
sudo /opt/RedDatabase/bin/nbackup -L <путь_до_бд>
```

9. Запустите СУБД на мастере:

```
sudo systemctl start firebird
```

10. Проверьте, что изменения в базе данных пишутся в дельту, создаются логи и архивы репликации, т. е. в каталоге с базой появился файл вида `red.fdb.delta` (дельта с заблокированной базы), а в каталогах `/home/logsrdb/logs` и `/home/logsrdb/arch` начали появляться файлы (логи и архивы репликации соответственно).
11. Создаем на слейве каталог для базы:

```
sudo mkdir /opt/db/
```

И копируем БД с мастера на слейв, например, с помощью `scp`. Пример (в приведенном виде выполняется на слейве):

```
sudo scp -r root@10.81.1.187:/opt/db/red.fdb /opt/db
```

И назначаем на неё соответствующие права:

```
sudo chmod -R 760 /opt/db
sudo chown -R firebird:firebird /opt/db
```

12. После копирования базы данных на слейв, разблокируем ее на мастере:

```
sudo /opt/RedDatabase/bin/nbackup -N /opt/db/red.fdb
```

где `/opt/db/red.fdb` — путь до базы данных

13. Разблокируем базу данных на слейве:

```
sudo /opt/RedDatabase/bin/nbackup -F /opt/db/red.fdb
```

14. Переводим базу на слейве в режим реплики:

```
sudo /opt/RedDatabase/bin/gfix -replica <GUID_базы_данных_с_мастера> -u
<имя_администратора> -p <пароль_администратора> <путь_до_базы_данных>
```

Здесь указывается GUID мастер-базы, который можно узнать из вывода `GSTAT -h` (на базе мастера), пример:

```
sudo /opt/RedDatabase/bin/gfix -replica F02EAEE7-44D0-4FCD-3D9D-BD4B83BEFCD2
-user sysdba -password masterkey /opt/db/red.fdb
```

где `/opt/db/red.fdb` — путь до базы данных слэйва.

15. Убеждаемся, что пользователь `firebird` имеет доступ к расшаренной папке с архивированными логами и к базе данных на слейве:

```
ll /mnt/repl/logsrdb/
ll /opt/db/
```

16. Включаем СУБД на слейве и убеждаемся, контрольный файл создан и архивы начали вливаться:

```
sudo systemctl start firebird
ll /mnt/repl/logsrdb/arch/\{*
```

здесь `/mnt/repl/logsrdb/arch/` - путь до каталога с архивами + символы `\{*`

```
sudo /opt/RedDatabase/bin/rdbreplmgr -s <путь_до_базы_данных_на_слейве>
```

Например:

```
sudo /opt/RedDatabase/bin/rdbreplmgr -s /opt/db/red.fdb
```

Вывод будет примерно следующиего содержания:

```
Status for replica /opt/db/red.fdb:
Master database: 10.81.1.187:/opt/db/red.fdb
Master GUID: 71615D98-0551-48D3-EE94-9441AFF3FE04
Archive directory: /mnt/repl/logsrdb/arch/
Control file: /mnt/repl/logsrdb/arch/71615D98-0551-48D3-EE94-9441AFF3FE04
Current segment: 16 (as of 2020-06-17 09:36:35)
Oldest segment: absent
Total segments in the queue: 1
```

Где

- `Current segment` — количество обработанных сегментов, должно постоянно расти;
- `Oldest segment` — количество старых сегментов, должно быть в значении «absent»;
- `Total segments in the queue` — количество сегментов в очереди на обработку, в идеале должно стремиться к 0.

17. После настройки репликации необходимо дополнительно настроить сборку мусора на слейве. Запускаться сборка мусора будет через планировщик `cron`. Чтобы добавить задачу на выполнение в `cron` необходимо выполнить команду:

```
sudo crontab -e
```

После чего откроется редактор задач планировщика, в котором нужно будет прописать запуск сборки мусора, указав время выполнения, например:

```
10 23 * * * /opt/RedDatabase/bin/gfix -sweep -user sysdba -password masterkey
localhost:ALIAS_DB
```

В этом примере сборка мусора будет запускаться каждый день в 23:10. Расписание запуска задач в `crontab` можно посмотреть с помощью команды:

```
sudo crontab -l
```

Настройка базы данных для синхронной репликации

Инициализация синхронной реплики

На слейве рекомендуется установить СУБД Ред База Данных с архитектурой классик.

1. Остановить СУБД на мастере, убедиться что не осталось процессов сервера в системе.
2. В `replication.conf` прописать секцию синхронной реплики `database = /путь/к/бд`. В ней задать обязательный параметр подключения к реплике в виде:

```
replica_database = [<login>:<password>@]<database connection string>
```

Их может быть несколько.

3. Существует два способа инициализации реплики: из физической копии и из логической копии.

3.1 Для инициализации реплики из физической копии необходимо скопировать базу данных на слейв, например, командой:

```
scp -c arcfour
```

3.2 Для инициализации реплики из логической копии необходимо создать резервную копию базы данных:

```
gbak -B <база_данных-источник> <файл резервной копии>
```

При наличии триггеров, способных рассинхронизировать мастер и слейв (например, триггеры на отключение или подключение), нужно использовать опцию `-nod` при создании резервной копии.
Далее нужно восстановить базу данных из резервной копии на стороне слейва:

```
gbak -C <файл резервной копии> <база_данных>
```

4. Для базы-реплики активируется режим репликации с помощью опции `-replica` утилиты `GFIX`:

```
gfix -replica <GUID> -user <имя пользователя> -password <пароль>
<replica_database>
```

   Здесь указывается `GUID` мастер-базы, который можно узнать из вывода `GSTAT -h`.

5. Запустить СУБД на мастере.

Перевести реплику в режим штатной работы можно командой:

```
gfix -replica {} <replica_database>
```

Информация о режиме репликации

Узнать, в каком режиме находится база данных, можно с помощью утилиты `gstat -h`.

```
gstat -h d:\db\R_SLAVE.FDB
---------------------------------
Database header page information:
...
Attributes force write, replica
Variable header data:
Database GUID: 6C81FDE1-9978-417C-11BD-FFA63E5AA6A0
Replication master GUID: 6C81FDE1-9978-417C-11BD-FFA63E5AA6A0
```

Как видно из примера, для слейва в атрибутах БД указан режим `replica`, а в дополнительных данных – `GUID` этой БД и `GUID` мастер-базы. В данном случае они одинаковы, потому что реплика была создана копированием мастер-базы.

Более подробная информация доступна через виртуальную таблицу `MON$REPLICATION`, которая состоит из следующих полей:

Таблица 11.1 — Опции `MON$REPLICATION`

Опция	Описание
`MON$TYPE`	Режим репликации БД: 1 – мастер с синхронной репликацией; 2 – мастер с асинхронной репликацией; 3 – слейв;
`MON$CONNECTION_STRING`	Строка подключения к реплике для мастера в синхронном режиме, каталог журнала для мастера в асинхронном режиме или GUID мастер-базы для слейва;
`MON$ACTIVE`	Для мастера в синхронном режиме и слейва здесь будет 1 если репликация активна и 0, если нет; для мастера в асинхронном режиме – последовательный номер активного сегмента журнала;
`MON$LAST_MODIFIED`	Время последней активности репликации (отправки сообщения на реплику или записи в лог на мастере, получения репликационного пакета на слейве);
`MON$WAITFLUSH_COUNT`	Количество сбросов буфера на синхронную реплику или в журнал репликации для текущего подключения;
`MON$WAITFLUSH_TIME`	Время, затраченное на сброс буферов на синхронную реплику или в журнал репликации для текущего подключения;

MON$WAITFLUSH_TRANSFER	Количество байт, сброшенных на синхронную реплику или в журнал репликации для текущего подключения;
MON$BACKGROUND_COUNT	Аналогично MON$WAITFLUSH_COUNT, но выполненное в фоне отдельным потоком;
MON$BACKGROUND_TIME	Аналогично MON$WAITFLUSH_TIME, но выполненное в фоне отдельным потоком;
MON$BACKGROUND_TRANSFER	Аналогично MON$WAITFLUSH_COUNT, но выполненное в фоне отдельным потоком;

11.5 Утилиты

Утилита настройки журнала репликации (rdblogmgr)

Данная утилита предназначена для вывода детализации текущего состояния журнала асинхронной репликации (общее состояние журнала, настройки журнала конфигурации, список использованных сегментов). Дополнительно, утилита `rdblogmgr` позволяет выполнить ручное архивирование заданного сегмента журнала или всех сегментов, а также принудительно помечает используемый сегмент как полный для возможности его архивирования.

Для запуска утилиты необходимо выполнить следующую команду:

```
rdblogmgr -D[atabase] <имя_базы_данных> -U[ser] <имя_пользователя> -P[assword]
<пароль>
```

Таблица 11.2 — Опции `rdblogmgr`

Опция	Описание
-D[atabase] <имя_базы_данных>	Имя базы данных
-U[ser] <имя_пользователя>	Имя пользователя
-P[assword] <пароль>	Пароль пользователя
-G[uid] <guid>	guid базы данных
-C[onfig]	Показывает настройки журнала конфигурации (блок `Log configuration`)
-S[egments]	Показывает список используемых сегментов из каталога `log_directory` (блок `Available log segments`). Показывает в каком состоянии находится тот или иной сегмент, его размер.
-A[rchive] <номер_сегмента>	Архивирует отдельный сегмент журнала
-A[rchive] all	Архивирует все сегменты журнала
-F[orce]	Принудительно помечает сегмент как полный, если даже тот используется.
-Z	Показать версии утилиты и сервера
-?	Вывод доступных опций

При каждом запуске утилиты выводится общее состояние журнала (блок Log status).

Ручное принудительное архивирование логов можно использовать, если не успевает фоновое архивирование (был сбой связи и сегменты долго не архивировались) или если база мастера вышла из строя и нужно обновить последний сегмент на слейв.

Если сегмент активный, то он будет заархивирован только при указании опции `-F[orce]`.

Приведем пример:

```
rdblogmgr -D /tmp/firebird/RDB-tpcc.fdb -U sysdba -P masterkey -C -S
------------------------------------------------
Log status:
Current sequence: 2
Last modified: 2015-08-06 12:37:07
Active segment: RDB-tpcc.fdb.log-001, size: 49051
Total log size: 3483922 bytes in 2 segments
Free segments: 0, full segments: 1, archived segments: 0

Log configuration:
Log directory = /tmp/firebird/RDB-tpcc-log/
Log file prefix = RDB-tpcc.fdb
Max segment count = 0
Max segment size = 16777216
Archive directory =
Archive command =
Archive timeout = 0

Available log segments:
File name: RDB-tpcc.fdb.log-000
Sequence: 1
State: full
Size in use: 3434871 bytes
File name: RDB-tpcc.fdb.log-001
Sequence: 2
State: used
Size in use: 49051 bytes
```

Утилита для применения файлов асинхронной репликации к базе (rdbreplmgr)

Утилита `rdbreplmgr`, ориентированная на слейв, накатывает журналы на реплику в ручном режиме, выводит информацию о состоянии асинхронной репликации, а также создает копию мастер-базы, если определен параметр конфигурации `db_copy_command`.

Для запуска утилиты необходимо выполнить следующую команду:

```
rdbreplmgr <команды> [<опции>] <replica_name>
```

Таблица 11.3 — Команды и опции `rdbreplmgr`

Команды и опции	Описание
`-C[reate]`	Создает копию мастер-базы
`-A[pply]`	Подключается к реплике и накатывает журналы из каталога, указанного в replication.conf
`-L[og]`	Вывод детальной информации журнала
`-S[tatus]`	Выводит информацию о состоянии репликации
`-U[ser] <username>`	Имя пользователя

-P[assword] <password>	Пароль пользователя
-G[uid]	Ручное указание guid-мастера
-V[erbose]	Подробный вывод о проверке
-Z	Показать версии утилиты и сервера
?	Помощь

Команда -Create выполняет программу, определенную в параметре db_copy_command, для пересоздания слейва. Пример такой команды см. выше.

Команда -Log <путь_до_журнала> выводит информацию об операциях, которые хранятся внутри журнала репликации. Пример выполнения команды -Log:

```
Report for log file <путь_до_журнала>:
Version: 1
State: arch
GUID: 478F611E-D0CC-418F-5D90-F461CAFA657A
Sequence: 5
Protocol: 2
Length: 2494
Log file segments:
Segment started at 2021-12-21 15:30:28 (1063 bytes):
StartTransaction: att_id=4585, tra_id=32432
StartSavepoint: tra_id=32432
...
```

Для вывода информации о журналах mc с помощью rdbreplmgr -L необходимо в /etc/mc/mc.ext прописать следующее:

```
regex/log-[0-9]3$
View=%viewascii /opt/RedDatabase/bin/rdbreplmgr -L %p
regex/arch-[0-9]9$
View=%viewascii /opt/RedDatabase/bin/rdbreplmgr -L %p
```

Пример выполнения команды -Status:

```
Status for replica d:\repl \slave_async.fdb:
Master database: d:\temp \repl \master.fdb
Master GUID: {232822C2-9A04-498D-0C8B-971C3E0DF421}
Archive directory: d:\temp\repl\logs\
Control file: d:\temp\repl\logstext {232822C2-9A04-498D-0C8B-971C3E0DF421}
Current segment: 2
Oldest segment: absent
Total segments in the queue: 1
```

Утилита проверки целостности данных (rdbrepldiff)

Для выполнения проверки целостности данных после репликации используется утилита rdbrepldiff. Суть работы данной утилиты — произвести считывание всех таблиц и всех данных в мастер-базе и слейв-базе и сверить их идентичность. Утилиту можно запускать периодически, когда сервера слабо нагружены или в моменты, когда есть предположения, что данные на мастере и на слейве, в связи с какой-либо ошибкой, отличаются друг от друга и необходимо применить какие-то меры для их синхронизации.

Для того, чтобы проверить целостность базы данных, необходимо выполнить команду:

```
rdbrepldiff [опции] -D[atabase] <master_name> -R[eplica] <replica_name>
```

Спецификация базы данных задается в формате `<адрес сервера>:<путь к БД>`.

Таблица 11.4 — Опции `rdbrepldiff`

Опции	Описание
`-U[ser] <username>`	Имя пользователя
`-P[assword] <password>`	Пароль пользователя
`-C[onfig] <config>`	Имя измененного файла конфигурации для возможности сравнивать не все сущности базы данных, а только необходимые. При этом процесс репликации не прерывается
`-M[etadata]`	Производит проверку только метаданных
`-S[ynchronize]`	Ожидание синхронизации мастера и слейва в течение одной минуты или времени, указанного в параметре `-Timeout`
`-N[odbtriggersoff]`	Включает ипользование триггеров при подключении к мастеру
`-T[imeout]`	Тайм-аут ожидания синхронизации (в секундах)
`-V[erbose]`	Подробный вывод о проверке
`-Z`	Показать версии утилиты и сервера
`?`	Помощь

Для проверки должен быть указан адрес мастер-сервера и адрес слейв-сервера. После запуска утилита подключается к ним (одновременно) и выполняется чтение метаданных обоих баз. Выбирается первая таблица и идет чтение из таблицы обоих баз, сравниваются записи. Если в одной из таблиц оказалось больше данных или если данные не совпадают, то таблицы считаются не идентичными (выдается сообщение об ошибке). Этот процесс продолжается для всех остальных таблиц. Если все таблицы идентичны, то делается вывод, что базы идентичны.

Глава 12

Настройка производительности Ред Базы Данных

Производительность конкретной системы зависит от аппаратного и программного обеспечения. Причины плохой производительности могут быть с обеих сторон - слабая дисковая подсистема, мало памяти, плохое управление транзакциями в приложении, плохие запросы и т.д.

В первом разделе будет описано, как подобрать эффективное аппаратное обеспечение для Ред Базы Данных. Далее даются рекомендации по диагностике производительности системы. Заключительная часть главы посвящена проблемам производительности и пути их устранения.

Производительность СУБД Ред База Данных 3.0 описана в Приложении З

12.1 Выбор аппаратного обеспечения

Прежде чем подобрать эффективное аппаратное обеспечение для БД, следует понять как Ред База Данных использует его компоненты: CPU, RAM, HDD/SSD.

Основные операции взаимодействия БД с аппаратным обеспечением

При старте СУБД процесс сервера занимает в RAM минимальный объем (несколько мегабайт), и не производит никаких интенсивных операций с CPU или RAM.

При соединении с БД сервер начинает читать её метаданные и создавать соответствующие объекты в памяти, отчего размер процесса увеличивается пропорционально количеству используемых таблиц, индексов, триггеров и других метаданных. Использование памяти увеличивается, но CPU практически не задействован на этом этапе.

Когда клиент начинает выполнять SQL-запросы (включая хранимые процедуры), сервер выполняет соответствующие операции, обращающиеся к "железу". Среди этих операций мы можем выделить следующие базовые операции, потребляющие определенный набор системных ресурсов. Они представлены в таблице с указанием интенсивности потребления (1 означает небольшую интенсивность, 10 – максимальную):

Таблица 12.1 — Потребление системных ресурсов

	Чтение страниц БД с диска	**Запись страниц БД на диск**	**Чтение страниц БД из кэша**	**Запись страниц БД в кэш**	**Чтение страниц данных из GTT**	**Запись страниц данных в GTT**	**Сортировка записей**	**Обработка SQL запроса**
CPU	1	1	1	1	1	1	5	10
RAM	5	5	5	5	5	5	5	2
Disc IO	10	10	1	1	1	1	1	1

[16] В не интерактивном режиме команды вводятся с префиксом «-», например -add

Видно, что наиболее тяжелыми операциями являются те, которые включают работу с диском, так как дисковая подсистема является наиболее медленным компонентом "железа".

Процессор

При выборе процессора следует принять во внимание следующие вещи:

- **Насколько сложные запросы преобладают**
  Ред базы данных всегда исполняет один запрос на одном ядре, поэтому сложные или плохо оптимизированные запросы могут занимать до 100% одного ядра, заставляя остальные запросы переместиться на менее загруженные ядра. Поэтому чем больше ядер, тем меньше шанс, что все процессорные мощности будут заняты.
  Если преобладают простые короткие SQL запросы, все запросы хорошо отлажены, то CPU не будет являться узким местом производительности, и можно выбрать модель с меньшим количеством ядер.
  Если преобладает большое количество медленных запросов, возвращающих большое количество данных, то необходим процессор с большим количеством ядер.
- **Число активных соединений**
  Для грубой оценки необходимого количества ядер в CPU можно пользоваться правилом от 10 до 30 соединений на 1 ядро. 10 пользователей/ядро – приложение с преобладанием сложных и медленных запросов, 30 соединений/ядро – приложение с преобладанием простых, хорошо отлаженных запросов.

Память

При выборе RAM следует уделить внимание двум моментам:

- **Модули памяти должны быть с коррекцией ошибок (ECC RAM)**
  ECC RAM значительно снижает количество ошибок при работе с памятью и настоятельно рекомендуется для использования в промышленных системах.
- **Правильно рассчитать объем RAM**
  Ред База Данных 3.0 архитектуры Classic запускает отдельный процесс для обслуживания каждого соединение, SuperClassic запускает отдельный поток для каждого соединения, но практически с той же структурой потребления памяти – каждое соединение имеет свой независимый страничный кэш. SuperServer запускает один процесс с общим страничным кэшем для всех соединений с каждой базой данных. При подключении к нескольким базам для каждой из них будет выделен свой объем страничного кэша, заданный настройками в файлах конфигурации или в самой базе данных.
  Таким образом, потребление памяти складывается из следующих основных параметров:
    1. Количество соединений
    2. Размер страницы базы данных
    3. Размер объектов метаданных (пропорционален количеству таблиц, триггеров, хранимых процедур и др.)
    4. Размер страничного кэша (определяется параметрами в заголовке БД или в `firebird.conf` или в свойствах конкретного соединения)
    5. Размер кэша для сортировок (определяется параметром в `firebird.conf`)
    6. Для Classic/SuperClassic – размер таблицы блокировок

  Ниже представлены формулы приблизительного расчета необходимого объема памяти для Ред Базы Данных:

  – **для Classic**

> Количество соединений * ((Кол-во страниц в кэше * Размер страницы) + Размер кэша для сортировок)

– **для SuperClassic**

> Количество соединений * (Кол-во страниц в кэше * Размер страницы) + Размер кэша для сортировок

– **для SuperServer**

> (Кол-во страниц в кэше * Размер страницы) + Размер кэша для сортировок

Реальное значение потребления памяти может отличаться, так как в этом расчете не учитывается объем памяти под метаданные, под битовые маски индексов, и т.д., что может увеличить расход памяти, но одновременно предполагается, что память под сортировки будет использована полностью во всех соединениях, чего обычно не происходит.

Количество RAM сверх рассчитанного минимального будет эффективно использоваться операционной системой для кэширования файла БД.

Когда база данных уже находится в эксплуатации, можно просто посмотреть средний размер памяти, используемый процессом СУБД (с помощью TaskManager или ProcessExplorer).

Дисковая подсистема

Чтобы уменьшить конкуренцию за дисковый ввод-вывод между операциями с файлом БД, сортировками и резервными копиями, а также уменьшить шанс одновременной потери и БД и резервных копий, рекомендуется иметь 3 разных диска (или raid-массива):

- **Для базы данных**
  Для работы с базой данных лучше всего использовать SSD-диски, так как они обеспечивают отличное масштабирование при параллельном вводе-выводе. Обязательно следует использовать диски промышленного класса, с увеличенным числом циклов перезаписи, иначе велик риск потери данных из-за поломки SSD.
  При этом рекомендуется оставлять до 30% свободного места на диске (из-за повышенного износа) и планировать замену диска примерно раз в 3 года.
  Наилучшим выбором является использование SSD эксклюзивно для работы с базой данных, так как любые операции ввода-вывода сокращают срок службы дисков.
  Если SSD диск оказывается слишком дорогим решением или размер БД слишком велик, то можно подобрать альтернативные варианты в порядке уменьшения приоритета:
  1. HDD с интерфейсом SAS
  2. диски SATA с интерфейсом nSAS
  3. обычные диски SATA
- **Для временных файлов**
  Так как временные файлы на диске возникают только при отсутствии достаточного количества RAM, то конечно, лучше всего вообще избегать их появления на диске.
  Но все-таки Ред База Данных требует указания папки, где будут хранится временные файлы (параметры TempTableDirectory и TempDirectories). Обычно его оставляют по умолчанию, т.е. используется стандартный временный каталог ОС. Чтобы предупредить исчерпание свободного места на системном диске, в `firebird.conf` указывают второй диск в качестве дополнительного резервного места.
  Следует учитывать, что при создании индекса при восстановлении верифицированного бэкапа (созданного утилитой gbak) создается временный файл, который содержит все

ключи этого индекса. Если база данных достаточно велика, то размер индекса для какой-нибудь большой таблицы может быть также значительным.

- **Для резервных копий**
  При резервном копировании (верифицированном или неверифицированном) происходит чтение файла базы данных (всего или части), и запись резервной копии (полной или частичной). Операции записи при создании резервной копии идут последовательно, это означает, что обычные недорогие жесткие диски с интерфейсом SATA (HDD SATA) хорошо подойдут для хранения резервных копий, так как скорость последовательной записи у них довольно велика.
  На диске для резервных копий всегда должно оставаться свободное место в размере последнего бэкапа+10%. В этом случае есть возможность создать свежий бэкап, убедиться в корректном завершении процесса копирования и только потом удалить предыдущий бэкап.

Под «отдельными дисками» понимается, что физически потоки данных должны идти через разные каналы ввода-вывода. Если создать 3 логических диска на одном физическом диске, никакого улучшения производительности не произойдет. Однако, если 3 логических диска будут организованы на устройстве хранения данных (СХД), оснащенном многоканальными контроллерами, то производительность может возрасти, так как устройство может распределять потоки данных между контроллерами.

RAID

Для базы данный и бэкапов следует увеличивать надежность дисковой подсистемы путем объединения дисков в RAID.

Для SSD дисков следует обязательно использовать RAID1 – это 2 «зеркальных» диска, на которые одновременно пишутся изменения, что значительно уменьшит шанс полной потери данных.

Для HDD дисков, используемых для бэкапов, достаточно использовать RAID1, который обеспечит надежное хранение бэкапов и приемлемую скорость записи и чтения.

HDD диски, используемые для БД, необходимо объединять в RAID10 (минимум 4 диска), который обеспечит оптимальное сочетание стоимости, надежности и производительности.

12.2 Диагностика системы

Чтобы собрать информацию о производительности системы, следует выполнить следующие шаги:

- Измерить параметр IOPS произвольной записи, используя IOMeter:
  - размер блока: 8/16 KB;
  - режим доступа: произвольный (Random);
  - отношение количества операций чтения к количеству операций записи (англ. read/write ratio): 0/100
  - размер очереди (англ. queue depth): 5

  Использовать тестовый файл размером не менее 10 Гб = 19531250 секторов (когда размер сектора составляет 512 байт). По возможности все тесты на IOMeter должны быть проведены на простаивающей системе.
- Измерить параметр IOPS произвольного чтения с теми же установками, описанными выше (только с отношением чтения/записи = 100/0).
- Измерить скорость последовательной записи при помощи IOMeter:
  - размер блока: 64 KB;
  - режим доступа: последовательный (Sequential);
  - отношение количества операций чтения к количеству операций записи (англ. read/write ratio): 0/100

  - размер очереди (англ. queue depth): 1

- Дисковая подсистема довольно часто становится узким местом в работе. Поэтому очень важно уметь диагностировать проблемы с дисками. Одним из основных инструментов для наблюдения за производительностью дисковой подсистемы являются счетчики производительности:
  - % Disk Write Time (процент загруженности диска операциями записи)
  - % Disk Read Time (процент загруженности диска операциями чтения)
  - Avg. Disk sec/Write (среднее время в секундах, требуемое для выполнения диском одной операции записи)
  - Avg. Disk sec/Read (среднее время в секундах, требуемое для выполнения диском одной операции чтения)
  - Disk Read Bytes/sec (средняя скорость чтения)
  - Disk Reads/sec (количество обработанных за секунду запросов на чтение)
  - Disk Write Bytes/sec (средняя скорость записи)
  - Disk Writes/sec (количество обработанных за секунду запросов на запись)
- Включить аудит событий, изменив параметры в конфигурационном файле `fbtrace.conf`:
  - `enabled = true,`
  - `time_threshold = 1000`
- Собирать данные менеджера блокировок с регулярными интервалами:

```
rdb_lock_print -a -d </path/to/db> > db_lock_print.txt
```

- Мониторить характеристики процессора, используя CPU-Z.Диспетчеру задач Windows доверять не стоит.
- С помощью утилиты `gstat` проверить нет ли старых активных транзакций, которые препятствуют сборке мусора:

```
gstat -h </path/to/db>
```

- Проверить таблицы мониторинга или сохранить результаты этих таблиц во внешние файлы:

```
MON$DATABASE;MON$ATTACHMENTS;MON$TRANSACTIONS;MON$STATEMENTS;
MON$CALL_STACK; MON$IO_STATS; MON$RECORD_STATS; MON$CONTEXT_VARIABLES;
MON$MEMORY_USAGE;
```

12.3 Узкие места и их устранение

Узкое место записи данных на диск с БД

Признаки

- Высокий показатель % Disk Write Time (>75%)
- Среднее значение счетчика Disk Writes/sec близко или превышает параметр IOPS произвольной записи, измеренный IOMeter.
- Запросы Insert/update/delete, которые обрабатывают небольшое количество строк и обычно выполняются за время менее 100 мсек, выполняются за одну и более секунды и появляются в журнале аудита.

Рекомендации

- Убедиться, что количество кэшируемых страниц достаточно большое. Рекомендуемое значение – 400. Установить другое значение кэшируемых страниц можно с помощью утилиты `gfix`:

```
GFIX -buffers 400 <db_name>
```

- Улучшить показатели параметра IOPS произвольной записи можно применением следующих мер:
    1. Избегайте применять RAID5. Лучше использовать RAID1 или RAID10
    2. Используйте SSD для хранения БД
- Разобраться какие запросы выполняют столько IO-операций (с помощью таблиц мониторинга и аудита)
- Как временный, запасной вариант, можно установить режим асинхронной записи данных – в этом случае изменения и новые данные хранятся в памяти и периодически сбрасываются на диск подсистемой ввода/вывода операционной системы. Общепринятое название такой конфигурации - forced writes off . По умолчанию БД устанавливается с включенным режимом принудительной записи - forced writes - синхронная запись. В этом режиме изменения и новые данные сразу записываются на диск.
  Затем в конфигурационном файле `firebird.conf` отключить параметры `MaxUnflushedWrites` и `MaxUnflushedWriteTime`.

Узкое место чтения данных с диска БД

Признаки

- Высокий показатель % Disk Read Time ($> 75\%$)
- Среднее значение счетчика Disk Reads/sec близко или превышает параметр IOPS произвольного чтения, измеренный IOMeter в течение нескольких минут.
- Запросы на чтение и обновление выполняются чрезвычайно долго и появляются в журнале аудита.

Рекомендации

- Убедитесь, что ОС можно использовать всю оперативную память для кэширования. Для этого можно проверить установленные границы размера файла системного кэша в результате использования утилиты `SetSystemFileCacheSize`. Она должна показывать:

```
Current cache working set size
min: none set
max: none set
```

  Если утилита показывается что-то иное, установите `FileSystemCacheSize=0` в конфигурационном файле `firebird.conf`. При внесении изменений потребуется перезагрузка.
- Убедитесь, что достаточно оперативной памяти доступно для кэширования.
- Разобраться какие запросы выполняют столько IO-операций (с помощью таблиц мониторинга и аудита)

Проблемы с троттлингом процессора

Признаки

- Измерения с помощью CPU-Z показали, что тактовая частота процессора ниже заявленной

Когда система задушена, то все остальные операции тоже начинают занимать большее время, и система все больше склоняется к образованию "узких мест"в других местах (подсистемах, уровнях) из-за взаимных блокировок.

Рекомендации

- В разделе Электропитание панели управления Windows выберите план «Высокая производительность». Далее «Настройка плана электропитания» -> «Изменить дополнительные параметры питания» , далее выбрать пункт «Управление питанием процессора», где «Минимальное состояние процессора» должно равняться 100 %. С помощью CPU-Z убедитесь, что проблема решена.
- Если предыдущий шаг не помог, значит BIOS или прошивка снизили частоту процессора.Прочтите руководство Server's BIOS и выберите настройки, которые соответствуют параметру «Performance Optimized». Возможно потребуется отключить Intel SpeedStep в BIOS.
- Не исключено, что тактовая частота процессора снижена из-за проблем с охлаждением. Измерьте температуру процессора и чипсета (например с помощью утилиты Open Hardware Monitor).

Проблемы с сортировкой

При выполнении сортировок Ред База Данных выполняет ее в памяти (в адресном пространстве процесса сервера), пока размер используемой памяти для всех выполняемых одновременно сортировок не достигнет предела, установленного параметром `TempCacheLimit` (`firebird.conf`). При превышении этого лимита создается временный файл (с соответствующим флагом операционной системы) в папке временных файлов, и в нем выполняется сортировка. В случае, если в системе есть свободная память (RAM), то файл сортировки будет кэширован на уровне ОС и сортировка будет производиться в памяти. Измерения показывают, что запросы с большими сортировками выполняются в 2 раза быстрее в ОЗУ, а не на диске.

Значение параметра `TempCacheLimit` рекомендуеся сделать больше суммарного размера временных файлов в каталогах `TempDirectories`. При наличии свободной памяти можно увеличить значение параметра и понаблюдать за эффектом с помощью аудита.

При включенном логировании операций типа `finish`, в трейсе отразится объём кэша, выделенного для сортировки.

Имя временного файла формируется следующим образом:

```
<тип временного объекта>_att<id подключения>_stmt<id запроса>_<случайная строка>
```

Например: `/tmp/fb_sort_att342316_stmt106_M2OSLu`.

Проблемы со сборкой мусора

Признаки

- Большая разница между «Oldest transaction» и «Next transaction» при выводе статистики: `gstat -h`
- Существует большая разница между Oldest active (или Oldest Snapshot) и Next transaction.

- Запросы выполняются неожиданно долго

Удержание Oldest active и Oldest Snapshot возможны по следующим причинам:

- некоторое приложение работает, и держит открытой хотя бы одну транзакцию - например, пользователь не закрыл приложение, и какая-либо его форма очень долго открыта;

- приложения работают длительное время, и «теряют» идентификаторы транзакций в коде;

- используемая библиотека компонент или драйвер работает с «транзакцией по умолчанию», которая может длиться очень долго.

Рекомендации

- Посмотреть таблицы мониторинга, чтобы узнать какая транзакция активна больше всего времени, принять необходимые меры по ее закрытию и провести принудительную сборку мусора `gfix -sweep n <db_name>`
- Процедуру чистки (sweep) базы данных выполнять по крайней мере раз в неделю. Один из способов — это вызвать принудительную сборку мусора с помощью команды `gfix -sweep n <db_name>`.Вы можете запланировать запуск команды с помощью планировщика Windows.
- Выполнять периодическое резервное копирование-восстановление БД по крайней мере раз в год или после крупного изменения в данных.

Длительное ожидание блокировок

Признаки

- Запросы выполняются неожиданно долго
- В менеджере блокировок процент попыток (Mutex wait), которые были заблокированы, когда владелец старался обратиться к таблице блокировок, высокий (более 20%)

Рекомендации

- Обычно причина высокого показателя Mutex wait кроется с недостаточно оптимальной структуре хэш-таблицы. Не рекомендуется, чтобы среднее значение «Hash lengths» было больше 3 или максимальное больше 10. Рекомендуемое минимальное число слотов `LockHashSlots=30011` (`firebird.conf`). В качестве значения размера хэш-таблицы лучше указывать простое число.

Неэффективное использование таблиц мониторинга

Признаки

- Запросы выполняются неожиданно долго
- Дамп лок-таблицы показывает цепочки ожидания для блокировки 20-го типа (series 20) (номер серии недавно изменялся, он актуален для Ред Базы версии 2.5.0.10954 или новее)

Сбор информации для MON$-таблиц требует глобальной блокировки БД и является достаточно длительной операцией. Не рекомендуется обращаться к MON$-таблицам на рабочем сервере чаще, чем раз в несколько минут.

Рекомендации

- Автоматические процессы, процедуры или триггеры не должны обращаться к MON$ таблицам так часто. Эта проблема должна быть перенаправлена на разработчиков приложений для исправления.

Проблемы с блокировками транзакций

Признаки

- DML запросы выполняются неожиданно долго
- Дамп лок-таблицы показывает цепочки ожидания для блокировки 4го типа

Эта проблема вызвана одновременным обновлением одинаковых наборов записей в режиме WAIT разрешения блокировок или неправильным выбором параметров транзакций (например, использование параметра no_rec_version)

Рекомендации

- Этот вопрос должен быть переадресован разработчикам приложения на исправление

Узкое место в подсистеме памяти

Признаки

- Запросы, обрабатывающие большие объемы данных, выполняются неожиданно долго
- Большое потребление ресурсов центрального процессора, и сервер является основным их потребителем
- Мониторинг процессора показывает, что процент времени нахождения в режиме ядра высок (20% и более)

Причина таких признаков заключается в том, что во время чтения страницы БД должны быть скопированы с кэша файловой системы на буферный кэш Firebird. Существуют ограничения в пропускной способности памяти, что особенно заметно в крупных многопроцессорных конфигурациях, и тогда каналы обмена с памятью становятся узким местом.

Рекомендации

- Исключить проблемы уровня ОС
- Уменьшить размер страниц (с 16K до 8K) и увеличить размер буферного кэша
- Оптимизировать проблемные запросы (проверить по таблицам мониторинга и с помощью аудита)
- Убедиться в правильной настройке кэша сервера (возможно он слишком мал), проверить параметры конфигурационного файла:
  - `TempBlockSize` - можно увеличить до 2 или 3 Мб, но не до 16 Мб;
  - `TempCacheLimit` - сделать значение больше суммарного размера временных файлов в каталогах `TempDirectories`;
  - `TempDirectories` - каталоги для хранения данных сортировок указываются через точку с запятой, следуют в порядке очереди, пока не заполнится первая директория, вторая не будет использоваться. Указать первой директорией RAM диск, потом SSD или HDD;
  - `TempTableDirectory` - задать каталог для хранения данных глобальных временных таблиц и временных блобов.
- Достигнуты границы масштабируемости ядра Windows. Переходите на ОС которая позволяет полное профилирование системы (Linux, Unix).

Проблемы с блокировкой страниц

Признаки

- Запросы выполняются неожиданно долго
- Дамп лок-таблицы показывает цепочки ожидания для блокировки 3-го типа (series 3)

Это проблема могла быть вызвана выше описанными: узкое место записи данных на диск с БД, длительное ожидание блокировок, узкое место в подсистеме памяти.

Если проблемы не в этом, то причины могут быть следующие:

- много одновременных соединений выполняют крупные изменения в разных записях одинаковых таблиц (может быть выведено из журнала аудита)
- чрезвычайно высокий рост числа транзакций, сгенерируемых приложениями (может быть выведено из быстрого роста счетчиков транзакций)

Если и эти причины исключены, то можно получить номера страниц, используя ключ Key - идентификатор заблокированно ресурса, и выяснить, какие именно страницы вызывают проблемы и какие таблицы, индексы или другие структуры являются узким местом.

Рекомендации

- Разрешить причины, связанные с другими выше описанными проблемами
- Перевести задачу на разработчиков приложения для того, чтобы они могли исключить проблемы, возникающие при одновременных изменениях в таблицах или высоком росте числа транзакций
- Можно обратиться в Ред Софт, если все обычные причины исключены, а проблема осталась.

Долго выполнимые запросы

Признаки

- Пользователи отмечают плохую производительность, зависание клиента (или зависание всей системы, если запрос выполняется сервером, когда удерживается некоторая блокировка)
- В случае зависания проблемный запрос можно увидеть в таблице `MON$STATEMENTS` со строкой `MON$STATE <> 0`
- Проблемы с производительностью при определенных запросах
- В таблице мониторинга `MON$RECORD_STATS` количество неиндексных чтений гораздо больше индексных (`MON$RECORD_SEQ_READS > MON$RECORD_IDX_READS`)

Рекомендации

- Если система зависла и вы определили длительный запрос в `MON$STATEMENTS`, убейте его удалением соответствующей строки
- Запустить запрос отдельно с включенным отображением плана и статистики:

```
set stats on;
set plan on;
<запрос>
.......................
Current memory = 2574604
```

```
Delta memory = -5660
Max memory = 2653012
Elapsed time= 0.03 sec
Buffers = 512
Reads = 1
Writes 0
Fetches = 92
```

`Reads` - показатель дискового I/O, сколько страниц считано с диска в кэш сервера (тех, которые еще не находились в кэше). Первое выполнение запроса показывает точные данные. При повторном вызове, если `Reads = 0`, то все нужные страницы уже в кэше, а если значение больше 0, то в кэше не хватает места, чтобы поместить необходимые страницы, то есть на самом деле считано страниц `Buffers + Reads`.
`Writes` - страницы, записанные из кэша на диск (или в кэш операционной системы). Insert, update, delete - влияют на этот показатель, также влияет сборка мусора при select.
`Fetches` - индикатор операций с памятью (включая CPU). Сколько было обращений к странице в кэше, чтение записей, чтение ключей, чтение страниц указателей, и т.п.

- Опыт показывает, что наиболее значительные проблемы с производительностью вызываются отсутствием нужных индексов. Проверьте, существуют ли индексы, которые может использовать данный запрос. Если нет — постройте их.
  При наличие нескольких индексов оптимизатор выберет имеющие лучшую селективность и отбросит имеющие худшую, поэтому необходимо ее периодически пересчитывать (вручную или по расписанию, т.к. сама по себе она не пересчитывается).

Проблемы с Антивирусом

Признаки

- Система медленно работает

Антивирус перехватывает большинство системных вызовов и поэтому может вызвать замедление в совершенно неожиданных местах. Наблюдалось замедление ввода-вывода, замедление сети, замедления доступа к памяти.

Рекомендации

- Отключите или удалите антивирус. Если производительность улучшилась, попробуйте выборочно отключить антивирус для временной папки Firebird (C:\ProgramData\firebird)

Производительность файловой системы

Признаки

- Показатель IOPS производительности системы хранения данных значительно медленнее, чем производительность самого устройства. Вы можете ожидать значения IOPS для каждого SSD по крайней мере 5000 (для пары, если они "зеркальные") и по крайней мере 130 IOPS для каждого жесткого диска (для пары, если они "зеркальные").

Рекомендации

- Рекомендуется иметь размер кластера файловой системы NTFS совпадающий с размером страницы БД (8 или 16 Кб)

- Для многоуровневых конфигураций (например виртуализация) должны быть оценены накладные расходы на каждом уровне в плане задержек и пропускной способности операций ввода-вывода. Конфигурация должна быть настроена, чтобы обеспечить приемлемый уровень производительности

Глава 13

Коннектор CDC

13.1 Настройка коннектора CDC

Коннектор реализуется в виде плагина репликации `Debezium_Connector`, использующем `JAVA` библиотеки платформы `Debezium` с открытым исходным кодом для сбора измененных данных. Для включения коннектора плагин должен быть доступен в конфигурации плагинов `plugins.conf`, а также в конфигурации репликации `replication.conf` в секции `database` в настройке `plugin` должен быть указан `Debezium_Connector`. Параметры коннектора задаются в конфигурационном файле `reddatabase.connector.yaml`.

13.1.1 Параметры коннектора

В конфигурационном файле `reddatabase.connector.yaml` настраиваются следующие параметры:

connector.name – Имя коннектора.

connector.journal.directory - Каталог для записи журналов репликации (обязательный параметр).

connector.journal.check.interval.ms - Период просмотра наличия журналов транзакций в миллисекундах (обязательный параметр).

kafka.producer.bootstrap.servers - Список пар хост/порт, используемых для установления начального соединения с кластером `Kafka`. Клиент будет использовать все серверы независимо от того, какие серверы указаны здесь для начального подключения - этот список влияет только на начальные хосты, используемые для обнаружения полного набора серверов. Список имет вид:`host1:port1,host2:port2,.....` Поскольку эти серверы используются только для начального подключения для обнаружения полного состава кластера (который может динамически меняться), этот список не обязательно должен содержать полный набор серверов (однако вы можете захотеть несколько серверов на случай, если какой-либо сервер не работает).

kafka.producer.retries - При значении больше нуля клиент будет повторно отправлять любую запись, отправка которой не удалась из-за потенциально скоротечной ошибки. Обратите внимание, что такая повторная отправка ничем не отличается от того, если бы клиент повторно отправил запись после получения ошибки. Разрешение повторных запросов без установки `max.in.flight.requests.per.connection` равным 1 потенциально изменит порядок записей, поскольку если две партии отправляются в один раздел, и первая не удалась и была повторно отправлена, а вторая удалась, то записи из второй партии могут появиться первыми. Кроме того, обратите внимание, что запросы на производство будут отклонены до того, как будет исчерпано количество повторных попыток, если таймаут, настроенный в параметре `delivery.timeout.ms`, истечет до успешного подтверждения. Обычно пользователи предпочитают оставлять этот параметр без настройки и вместо него использовать `delivery.timeout.ms` для управления поведением повторных попыток.

kafka.producer.retry.backoff.ms - Количество времени ожидания перед повторной попыткой неудачного запроса к данному тематическому разделу. Это позволяет избежать многократной отправки запросов в замкнутом цикле при некоторых сценариях отказа.

kafka.producer.transactional.id - Используется для доставки транзакций. Обеспечивает семантику надежности, охватывающую несколько сессий `Producer`, поскольку позволяет клиенту гарантировать, что транзакции, использующие тот же `transactional.id`, были завершены до начала любых новых транзакций. Если `transactional.id` не указан, то производитель ограничивается идемпотентной доставкой. Если `transactional.id` настроен, то подразумевается `enable.idempotence`. По умолчанию `transactional.id` не настроен, что означает, что транзакции не могут быть использованы.

kafka.producer.max.block.ms - Время, в течение которого будут блокироваться методы `send()`, `partitionsFor()`, `initTransactions()`, `sendOffsetsToTransaction()`, `commitTransaction()` и `abortTransaction()` `KafkaProducer`. Для `send()` этот таймаут ограничивает общее время ожидания выборки метаданных и выделения буфера (блокировка в пользовательских сериализаторах или разделителе не учитывается в этом таймауте). Для `partitionsFor()` этот таймаут ограничивает время ожидания метаданных, если они недоступны. Методы, связанные с транзакциями, всегда блокируются, но могут прерваться, если координатор транзакций не может быть обнаружен или не отвечает в течение таймаута

kafka.producer.delivery.timeout.ms - Верхняя граница времени сообщения об успехе или неудаче после возврата вызова `send()`. Это ограничивает общее время задержки записи перед отправкой, время ожидания подтверждения от брокера (если ожидается), а также время, допустимое для повторных неудач отправки. `Producer` может сообщить о неудаче отправки записи раньше этого значения, если произошла неустранимая ошибка, исчерпаны все повторные попытки, или запись добавлена в пакет, который достиг более раннего срока истечения доставки. Значение этого параметра должно быть больше суммы `request.timeout.ms` и `linger.ms` или равно ей.

kafka.producer.request.timeout.ms - Конфигурация контролирует максимальное количество времени, в течение которого клиент будет ожидать ответа на запрос. Если ответ не получен до истечения таймаута, клиент повторно отправит запрос, если это необходимо, или отклонит запрос, если количество повторных попыток исчерпано. Это значение должно быть больше, чем `replica.lag.time.max.ms` (конфигурация брокера), чтобы уменьшить вероятность дублирования сообщений из-за ненужных повторных попыток производителя.

kafka.topic.prefix - Логическое имя, которое идентифицирует и обеспечивает пространство имен для конкретного сервера Ред Базы Данных, на котором фиксируются изменения. Логическое имя должно быть уникальным для всех других коннекторов, поскольку оно используется в качестве префикса для всех имен тем `Kafka`, которые получают события, испускаемые этим коннектором. В префиксе серверадолжны использоваться только буквенно-цифровые символы, дефисы, точки и символы подчеркивания.

key.converter - Конвертер ключей.

value.converter - Конвертер значений.

```
connector.name: reddatabase

kafka.producer.bootstrap.servers: localhost:9092
kafka.producer.retries: 3
kafka.producer.retry.backoff.ms: 1000
kafka.producer.max.block.ms: 1000
kafka.producer.timeout.ms: 1000
kafka.producer.transactional.id: reddatabase-transaction-id
kafka.topic.prefix: reddatabase_topic

key.converter: org.apache.kafka.connect.json.JsonConverter
value.converter: org.apache.kafka.connect.json.JsonConverter
```

Логирование действий коннектора осуществляется с помощью библиотеки Logback. Пример конфигурации логирования в файл:

```
<configuration>
  <appender name="FILE"class="ch.qos.logback.core.FileAppender»
    <file>/tmp/debezium-connector-reddatabase.log</file>

    <encoder>
      <pattern>%date %level [%thread] %logger50 [%file:%line] %msg%n</pattern>
    </encoder>
  </appender>
  <root level="debug»
    <appender-ref ref="FILE"/>
  </root>
</configuration>
```

13.1.2 Наименование тем

По умолчанию коннектор записывает события изменений для всех операций `INSERT`, `UPDATE` и `DELETE`, происходящих в таблице, в одну тему `Apache Kafka`, специфичную для этой таблицы. Коннектор использует следующее соглашение для наименования тем событий изменений:

```
<kafka.topic.prefix>.<имя таблицы>
```

13.2 Фильтрация таблиц и столбцов

Коннектор отслеживает изменения для всех таблиц, для которых включена репликация. Список реплицируемых таблиц является вариантом реализации списка отслеживаемых таблиц (так называемый `whitelist/includelist`). Включение таблицы в список реплицируемых осуществляется командой `ALTER DATABASE INCLUDE TABLE <ТАБЛИЦА> TO PUBLICATION`:

```
ALTER DATABASE INCLUDE TABLE TEST_TABLE TO PUBLICATION
```

Возможно включение нескольких таблиц в список реплицируемых:

```
ALTER DATABASE INCLUDE TABLE T1, T2, T3 TO PUBLICATION
```

При желании можно добавить все доступные таблицы (в том числе и те, которые будут созданы в процессе) в список реплицируемых:

```
ALTER DATABASE INCLUDE ALL TO PUBLICATION
```

Исключить конкретные таблицы из списка реплицируемых:

```
ALTER DATABASE EXCLUDE TABLE T1, T2, T3 FROM PUBLICATION;
```

Исключить все таблицы из списка реплицируемых:

```
ALTER DATABASE EXCLUDE ALL FROM PUBLICATION;
```

Кроме инструментов Ред Базы Данных, коннектор имеет встроенные механизмы фильтрации таблиц и столбцов. Настройка фильтрации таблиц и столбцов осуществляется следующими параметрами:

- `table.include.list` - Необязательный список регулярных выражений, разделенных запятыми, которые соответствуют идентификаторам таблиц, изменения которых необходимо отслеживать. Коннектор не фиксирует изменения в таблицах, не включенных в `table.include.list`. Каждый идентификатор имеет вид: `<Имя_БД>.<Имя_Таблицы>`. По умолчанию коннектор фиксирует изменения в каждой несистемной таблице, добавленной в список реплицируемых. Несовместим с параметром `table.exclude.list`;
- `table.exclude.list` - Необязательный список регулярных выражений, разделенных запятыми, которые соответствуют идентификаторам таблиц, изменения которых не отслеживаются. Коннектор фиксирует изменения в любой таблице, добавленной в список репликации, но не указанной в `table.exclude.list`. Каждый идентификатор имеет вид `<Имя_БД>.<Имя_Таблицы>`. Несовместим с параметром `table.include.list`;
- `column.include.list` - Необязательный список регулярных выражений, разделенных запятыми, которые соответствуют именам столбцов для включения в событие изменения. Полные имена столбцов имеют вид: `<Имя_БД>.<Имя_Таблицы>.<Имя_Столбца>`. Несовместим с параметром `column.exclude.list`;
- `column.exclude.list` - Необязательный список регулярных выражений, разделенных запятыми, которые соответствуют именам столбцов для исключения из события изменения. Полные имена столбцов имеют вид: `<Имя_БД>.<Имя_Таблицы>.<Имя_Столбца>`. Несовместим с параметром `column.include.list`.

Если в настройках указаны оба параметра `table.include.list` и `table.exclude.list`, то приоритет будет за `table.exclude.list`. Такое же поведение касается параметров `column.include.list` и `column.exclude.list` - приоритет у `column.exclude.list`. Включение таблиц в список реплицируемых (`ALTER DATABASE INCLUDE TABLE <ТАБЛИЦА> TO PUBLICATION`) является обязательным условием отслеживания изменений таблиц. Без включения репликации для таблиц использование парамтеров `table.include.list`, `table.exclude.list`, `column.include.list`, `column.exclude.list` не имеет смысла.

Примеры:

```
# Включение таблицы TABLE1 из БД /opt/databases/employee.fdb в список для
отслеживания:
table.include.list: /opt/databases/employee.fdb.table1

# Включение таблицы TABLE1 из ЛЮБОЙ БД в список для отслеживания:
table.include.list: .*table1
# Включение таблицы TABLE1 из ЛЮБОЙ БД и таблицы TEST_TABLE из любой БД с именем
data.fdb в список для отслеживания:
table.include.list: .*table1, .*data.fdb.test_table

# Включение таблицы TABLE1 из ЛЮБОЙ employee.fdb в список для отслеживания:
table.include.list: .*employee.fdb.table1

# Исключение таблицы TABLE1 базы данных /opt/databases/employee.fdb из списка
отслеживаемых:
table.include.list: /opt/databases/employee.fdb.table1

# Включение столбца ID таблицы TABLE1 из БД /opt/databases/employee.fdb в сообщение
изменения:
column.include.list: /opt/databases/employee.fdb.table1.id

# Включение столбца ID таблицы TABLE1 из ЛЮБОЙ БД в сообщение изменения:
column.include.list: .*table1.id

# Включение столбцов ID и NAME ЛЮБОЙ таблицы ЛЮБОЙ БД в сообщение изменения:
```

```
column.include.list: .*id, *.name

# Исключение столбца F_VCHAR ЛЮБОЙ таблицы ЛЮБОЙ БД из сообщения изменения:
column.exclude.list: .*f_vchar
```

13.3 События изменения данных

Каждое событие изменения данных, которое формирует коннектор Ред Базы, имеет ключ и значение. Структура ключа и значения зависит от таблицы, из которой исходят события изменения. Коннектор, как и `Kafka Connect`, разработан с учетом непрерывных потоков сообщений о событиях. Каждый ключ и значение сообщения состоит из двух частей: схемы и полезной нагрузки. Схема описывает структуру полезной нагрузки, а полезная нагрузка содержит фактические данные. Для каждой измененной таблицы, ключ события изменения формируется из полей в первичном ключе (или уникальном ключевом ограничении) таблицы на момент создания события.

Как и ключ сообщения, значение сообщения о событии изменения имеет раздел схемы и раздел полезной нагрузки. Раздел полезной нагрузки каждого значения события изменения, создаваемого коннектором, имеет структуру конверта со следующими полями:

- `op` - обязательное поле, содержащее строковое значение, описывающее тип операции: `CREATE` - создание (или вставка), `UPDATE` - обновление, `DELETE` - удаление;
- `before` - необязательное поле, которое, если присутствует, содержит состояние строки до наступления события;
- `after` - необязательное поле, которое, если присутствует, содержит состояние строки после произошедшего события. Структура описывается той же схемой, которая использовалась ранее;
- `source` - обязательное поле, содержащее структуру, описывающую исходные метаданные для события, которая в случае Ред Базы содержит такие поля: версия коннектора, имя коннектора, имя сервера, номер транзакции, время (по системным часам в `JVM`, выполняющей задачу `Kafka Connect`), в которое коннектор обработал событие, имя базы данных, имя таблицы.

Пример формирования события изменения данных для операции INSERT:

```
INSERT INTO TABLE11 VALUES(42,'NEW VALUE');
```

Когда в кластер Kafka попадает надгробная запись, то соответствующий ключ удаляется из хранилища состояний, освобождая таким образом место.

Если используется Jdbc Sink Connector, то включение tombstone records необходимо для генерации DELETE запросов для результирующей БД.

Отключить генерацию tombstone можно настройкой tombstones.on.delete:

```
tombstones.on.delete: false
```

13.4 События изменения схем

Коннектор формирует события изменения схемы, которые формируются из выполняемых DDL операций на сервере. Коннектор отправляет сообщение каждый раз, когда происходит изменение структуры базы данных. Сообщения, которые коннектор отправляет в тему изменения схемы, содержат полезную нагрузку и также содержат схему о событии изменения. Полезная нагрузка сообщения о событии изменения схемы включает следующие элементы:

- databaseName\schemaName - Идентифицирует базу данных и схему, содержащую изменение.
- ddl - Это поле содержит DDL, который отвечает за изменение схемы.
- tableChanges Массив из одного или нескольких элементов, которые содержат изменения схемы, созданные командой DDL.
- type - Описывает вид изменения. Значение является одним из следующих: CREATE - Таблица создана. ALTER - Таблица изменена. DROP - Таблица удалена.
- id - Полный идентификатор таблицы, которая была создана, изменена или удалена.
- table - Представляет метаданные таблицы после примененного изменения.
- primaryKeyColumnNames - Список столбцов, составляющих первичный ключ таблицы.
- columns - Метаданные для каждого столбца в измененной таблице.

Пример формирования события изменения схемы при выполнение операции CREATE TABLE:

```
create table table24(f_id integer primary key, f_vchar varchar(100));
```

13.5 Журналы транзакций

Коннектор для каждой репликационной транзакции создает журнал транзакции. Журнал транзакции используется для сохранения данных транзакции (DDL и DML событий) и отправки в кластер Kafka. Так же журналы используются для повторной отправки, если кластер был недоступен, или возникли другие ошибки, при первичной отправке сообщений транзакции. Для включения записи журналов транзакций необходимо указать путь к каталогу журналов в опции connector.journal.directory, например:

```
connector.journal.directory: /tmp/debezium-connector-reddatabase
```

В случае возниконовения ошибок при отправке сообщений журналы транзакции повторно вычитываются и, при первой доступности кластера Kafka, сообщения отправляются снова. При успешном отправлении всех сообщений репликационной транзакции журнал транзакции удаляется. Период просмотра наличия журналов транзакций и их повторной отправки по умолчанию равен 0. Период можно изменить через опцию connector.journal.check.interval.ms, которая принимает положительное число в миллисекундах. Например, задать время просмотра наличия журналов транзакций и их отправки равное 10 минутам:

```
connector.journal.check.interval.ms: 600000
```

Имена журналов транзакций имеют следующий формат: <Случайный UUID>.log[.live]. Журнал незафиксированной транзакции имеет расширение .live, журналы зафиксированных транзакций представлены с расширением .log. Коннектор отправляет сообщения только из журналов зафиксированных транзакций (с расширением .log). Если серверная транзакция была отменена (rollback), журнал репликационной транзакции удаляется в любом случае.

Просмотр журналов транзакций и отправка сообщений начинается при загрузке плагина.

Приложение А Описание таблиц мониторинга

Система «Ред База Данных» предоставляет возможность отслеживать работу с конкретной базой данных, выполняемую на стороне сервера. Для этих целей используются таблицы мониторинга. Эти таблицы являются виртуальными в том смысле, что до обращения к ним со стороны пользователя, никаких данных в них не записано. Они фактически заполняются данными только в момент запроса пользователя. При этом описания этих таблиц в базе данных присутствуют постоянно.

Список таблиц мониторинга представлен в таблице А.1.

Таблица А.1 — Список таблиц мониторинга «Ред База Данных»

Таблица	Описание
MON$DATABASE	Сведения о базе данных, с которой выполнено соединение
MON$REPLICATION	Сведения о статусе репликации базы данных
MON$ATTACHMENTS	Сведения о текущих соединениях с базой данных
MON$TRANSACTIONS	Запущенные на выполнение транзакции
MON$STATEMENTS	Подготовленные к выполнению запросы
MON$CALL_STACK	Обращения к стеку активными запросами хранимых процедур и триггеров
MON$IO_STATS	Статистика по вводу-выводу
MON$RECORD_STATS	Статистика на уровне записей
MON$TABLE_STATS	Статистика на уровне таблиц
MON$CONTEXT_VARIABLES	Сведения о пользовательских контекстных переменных
MON$MEMORY_USAGE	Статистика использования памяти

MON$DATABASE

Таблица А.2 — Сведения о базе данных, с которой выполнено соединение

Идентификатор столбца	Тип данных	Описание
MON$DATABASE_NAME	VARCHAR(255)	Полный путь и имя первичного файла базы данных или псевдоним базы данных.
MON$PAGE_SIZE	SMALLINT	Размер страницы файлов базы данных в байтах.
MON$ODS_MAJOR	SMALLINT	Старшая версия ODS.
MON$ODS_MINOR	SMALLINT	Младшая версия ODS.
MON$OLDEST_TRANSACTION	BIGINT	Номер старейшей заинтересованной транзакции — OIT, Oldest Interesting Transaction.
MON$OLDEST_ACTIVE	BIGINT	Номер старейшей активной транзакции — OAT, Oldest Active Transaction.

Идентификатор столбца	Тип данных	Описание
MON$OLDEST_SNAPSHOT	BIGINT	Номер транзакции, которая была активной на момент старта транзакции OAT, — транзакция OST, Oldest Snapshot Transaction.
MON$NEXT_TRANSACTION	BIGINT	Номер следующей транзакции.
MON$PAGE_BUFFERS	INTEGER	Количество страниц, выделенных в оперативной памяти для кэша.
MON$SQL_DIALECT	SMALLINT	SQL диалект базы данных: 1 или 3.
MON$SHUTDOWN_MODE	SMALLINT	Текущее состояние останова (shutdown) базы данных: 0 — база данных активна (online), 1 — останов для нескольких пользователей (multi-user shutdown), 2 — останов для одного пользователя (single-user shutdown), 3 — полный останов (full shutdown).
MON$SWEEP_INTERVAL	INTEGER	Интервал чистки (sweep interval).
MON$READ_ONLY	SMALLINT	Признак, является ли база данных только для чтения, read only (значение 1) или для чтения и записи, read-write (значение 0).
MON$FORCED_WRITES	SMALLINT	Указывает, установлен ли для базы режим синхронного вывода (forced writes, значение 1) или режим асинхронного вывода (значение 0).
MON$RESERVE_SPACE	SMALLINT	Флаг, указывающий на полное заполнение страниц БД (full) или 80% заполнение по умолчанию (reserve). 100%-е заполнение имеет смысл для баз только для чтения.
MON$CREATION_DATE	TIMESTAMP	Дата и время создания базы данных.
MON$PAGES	BIGINT	Количество страниц, выделенных для базы данных на внешнем устройстве.
MON$STAT_ID	INTEGER	Идентификатор статистики.
MON$BACKUP_STATE	SMALLINT	Текущее физическое состояние backup: 0 — база не затронута бэкапом, 1 — база заблокирована для резервирования, 2 — объединение временного файла дельты и основного файла базы данных.
MON$CRYPT_PAGE	BIGINT	Количество зашифрованных страниц в процессе шифрования/дешифрования; ноль если этот процесс закончился или не начинался.
MON$OWNER	CHAR(31)	Владелец базы данных.
MON$SEC_DATABASE	CHAR(7)	Отображает, какой тип базы данных безопасности используется: `Default` - база данных безопасности по умолчанию, т.е. `security3.fdb`; `Self` — в качестве базы данных безопасности используется текущая база данных; `Other` — в качестве базы данных безопасности используется другая база данных.

MON$REPLICATION

Таблица А.3 — Сведения о статусе репликации базы данных

Идентификатор столбца	Тип данных	Описание
MON$TYPE	SMALLINT	Тип состояния базы данных: 1 – мастер с синхронной репликацией; 2 – мастер с асинхронной репликацией; 3 – слейв с репликацией.
MON$CONNECTION_STRING	VARCHAR(255)	Строка подключения: - для мастера в синхронном режиме – путь к базе в формате [<login>:<password>@]<path_to_database>; - для мастера в асинхронном режиме – путь к каталогу с журналами репликации; - для слейва – GUID базы мастера.
MON$ACTIVE	BIGINT	Статус активности подключения: • 0 – отключена (для мастера с синхронной репликацией и слейва); • 1 – подключена (для мастера с синхронной репликацией); • <ATTACHMENT_ID> – идентификатора подключение мастера к реплике (для реплики с синхронной репликацией); • <номер_файла> – номер текущего файла журнала (для мастера с асинхронной репликацией).
MON$LAST_MODIFIED	TIMESTAMP	Содержит время последней успешной отправки буфера на слейв.
MON$WAITFLUSH_COUNT	INTEGER	Количество отправленных пакетов (с задержкой мастера, при коммитах).
MON$WAITFLUSH_TIME	BIGINT	Количество времени мастера для отправки пакетов (в микросекундах).
MON$WAITFLUSH_TRANSFER	BIGINT	Размер отправленных данных слейву (в байтах).
MON$BACKGROUND_COUNT	INTEGER	Количество отправленных пакетов (в фоне, в рамках одной транзакции).
MON$BACKGROUND_TIME	BIGINT	Количество времени мастера для отправки пакетов (в фоне, в рамках одной транзакции).
MON$BACKGROUND_TRANSFER	BIGINT	Размер отправленных данных слейву (в фоне, в рамках одной транзакции).

MON$ATTACHMENTS

Таблица А.4 — Сведения о текущих соединениях с базой данных

Идентификатор столбца	Тип данных	Описание
MON$ATTACHMENT_ID	BIGINT	Идентификатор соединения.
MON$SERVER_PID	INTEGER	Идентификатор серверного процесса.
MON$STATE	SMALLINT	Состояние соединения: 0 — бездействующее, 1 — активное. Соединение считается активным, если в нем есть хотя бы одна транзакция с хотя бы одним открытым запросом.
MON$ATTACHMENT_NAME	VARCHAR(255)	Полный путь к файлу и имя первичного файла базы данных.
MON$USER	CHAR(31)	Имя пользователя, соединенного с базой данных.
MON$ROLE	CHAR(31)	Имя роли, указанное при соединении. Если роль во время соединения не была задана, поле содержит текст NONE.
MON$REMOTE_PROTOCOL	VARCHAR(10)	Имя удаленного протокола.
MON$REMOTE_ADDRESS	VARCHAR(255)	Удаленный адрес (адрес и имя сервера).
MON$REMOTE_PID	INTEGER	Идентификатор удаленного клиентского процесса.
MON$CHARACTER_SET_ID	SMALLINT	Идентификатор набора символов в соединении.
MON$TIMESTAMP	TIMESTAMP	Дата и время начала соединения.
MON$GARBAGE_COLLECTION	SMALLINT	Разрешена ли сборка мусора для этого соединения: 1-разрешена, 0 - нет
MON$REMOTE_PROCESS	VARCHAR(255)	Полное имя файла и путь к исполняемому файлу, который установил данное соединение
MON$STAT_ID	INTEGER	Идентификатор статистики.
MON$CLIENT_VERSION	VARCHAR(255)	Версия клиентской библиотеки.
MON$REMOTE_VERSION	VARCHAR(255)	Версия удалённого протокола.
MON$REMOTE_HOST	VARCHAR(255)	Имя удалённого хоста.
MON$REMOTE_OS_USER	VARCHAR(255)	Имя пользователя в операционной системе.
MON$AUTH_METHOD	VARCHAR(255)	Метод аутентификации, используемый при подключении.
MON$SYSTEM_FLAG	SMALLINT	Флаг того, что подключение системное: 0 — пользовательское подключение; 1 — системное подключение.
MON$REPL_WAITFLUSH_-COUNT	INTEGER	Количество отправленных пакетов резервным базам данных.
MON$REPL_WAITFLUSH_-TIME	BIGINT	Время (в мс.), которое основной сервер ожидает ответа от резервных серверов.
MON$LAST_ACTIVITY_TIME	TIMESTAMP	Время последнего обращения пользователя к серверу

MON$TRANSACTIONS

Таблица A.5 — Описывает запущенные на выполнение транзакции

Идентификатор столбца	Тип данных	Описание
MON$TRANSACTION_ID	BIGINT	Идентификатор (номер) транзакции.
MON$ATTACHMENT_ID	BIGINT	Идентификатор соединения.
MON$STATE	SMALLINT	Состояние транзакции: 0 — бездействующая (нет открытых запросов), 1 — активная (есть открытые запросы).
MON$TIMESTAMP	TIMESTAMP	Дата и время старта транзакции.
MON$TOP_TRANSACTION	INTEGER	Верхний предел используемый транзакцией чистильщика (sweeper) при продвижении глобального OIT. Все транзакции выше этого порога считаются активными. Обычно он эквивалентен MON$TRANSACTION_ID, но использование COMMIT RETAINING или ROLLBACK RETAINING приводит к тому, что MON$TOP_TRANSACTION останется неизменным ("зависшим") при увеличении идентификатора транзакции.
MON$OLDEST_TRANSACTION	INTEGER	Номер старейшей заинтересованной транзакции — OIT, Oldest Interesting Transaction.
MON$OLDEST_ACTIVE	INTEGER	Номер старейшей активной транзакции — OAT, Oldest Active Transaction.
MON$ISOLATION_MODE	SMALLINT	Режим (уровень) изоляции: 0 — consistency (snapshot table stability), 1 — concurrency (snapshot), 2 — read committed record version, 3 — read committed no record version.
MON$LOCK_TIMEOUT	SMALLINT	Время ожидания: -1 — бесконечное ожидание (wait), 0 — транзакция no wait, другое число — время ожидания в секундах (lock timeout).
MON$READ_ONLY	SMALLINT	Признак, является ли транзакцией только для чтения, read only (значение 1) или для чтения и записи, read-write (0).
MON$AUTO_COMMIT	SMALLINT	Признак, используется ли автоматическое подтверждение транзакции auto-commit (значение 1) или нет (0).
MON$AUTO_UNDO	SMALLINT	Признак, используется ли автоматическая отмена транзакции auto-undo (значение 1) или нет (0).
MON$STAT_ID	INTEGER	Идентификатор статистики.

MON$STATEMENTS

Таблица А.6 — Подготовленные к выполнению запросы

Идентификатор столбца	Тип данных	Описание
MON$STATEMENT_ID	BIGINT	Идентификатор запроса.
MON$ATTACHMENT_ID	BIGINT	Идентификатор соединения.
MON$TRANSACTION_ID	BIGINT	Идентификатор транзакции.
MON$STATE	SMALLINT	Состояние запроса: 0 — бездействующий (idle), 1 — активный (active), 2 — приостановленный (stalled). То есть запрос или курсор "живой" , но в данный момент не выполняется. Например, это состояние является перерывом между клиентскими фетчами, т.е. запрос создал курсор, он еще недофетчен и в данный момент фетч не делается (обрабатывает предыдущую порцию).
MON$TIMESTAMP	TIMESTAMP	Дата и время старта запроса.
MON$SQL_TEXT	BLOB TEXT	Текст запроса на языке SQL.
MON$STAT_ID	INTEGER	Идентификатор статистики.
MON$EXPLAINED_PLAN	BLOB TEXT	План запрос в расширенной форме.

MON$CALL_STACK

Таблица А.7 — Обращения к стеку запросами хранимых процедур и триггеров

Идентификатор столбца	Тип данных	Описание
MON$CALL_ID	BIGINT	Идентификатор обращения.
MON$STATEMENT_ID	BIGINT	Идентификатор верхнего уровня SQL— запроса, инициировавшего цепочку обращений.
MON$CALLER_ID	BIGINT	Идентификатор обращающегося триггера, хранимой функции или хранимой процедуры.
MON$OBJECT_NAME	CHAR(31)	Имя объекта PSQL.
MON$OBJECT_TYPE	SMALLINT	Тип объекта PSQL: • 2 — триггер; • 5 — хранимая процедура; • 15 — хранимая функция.
MON$TIMESTAMP	TIMESTAMP	Дата и время старта обращения.
MON$SOURCE_LINE	INTEGER	Номер исходной строки SQL-запроса, выполняющегося в настоящий момент.
MON$SOURCE_COLUMN	INTEGER	Номер исходного столбца SQL-запроса, выполняющегося в настоящий момент.
MON$STAT_ID	INTEGER	Идентификатор статистики.
MON$PACKAGE_NAME	CHAR(31)	Имя пакета для упакованных процедур/функций.

MON$IO_STATS

Таблица A.8 — Статистика по вводу-выводу

Идентификатор столбца	Тип данных	Описание
MON$STAT_ID	INTEGER	Идентификатор статистики.
MON$STAT_GROUP	SMALLINT	Группа статистики: 0 — база данных (database), 1 — соединение с базой данных (connection), 2 — транзакция (transaction), 3 — запрос (statement), 4 — вызов (call).
MON$PAGE_READS	BIGINT	Количество прочитанных (read) страниц базы данных.
MON$PAGE_WRITES	BIGINT	Количество записанных (write) страниц базы данных.
MON$PAGE_FETCHES	BIGINT	Количество страниц, считанных из страничного кэша (fetch).
MON$PAGE_MARKS	BIGINT	Количество отмеченных (mark) страниц базы данных. Это "грязные" страницы, т.е. изменённые в памяти (в кэше), но пока не записанные на диск.

MON$RECORD_STATS

Таблица A.9 — Статистика на уровне записей

Идентификатор столбца	Тип данных	Описание
MON$STAT_ID	INTEGER	Идентификатор статистики.
MON$STAT_GROUP	SMALLINT	Группа статистики: 0 — база данных (database), 1 — соединение с базой данных (connection), 2 — транзакция (transaction), 3 — запрос (statement), 4 — вызов (call).
MON$RECORD_SEQ_READS	BIGINT	Количество последовательно считанных записей (read sequentially).
MON$RECORD_IDX_READS	BIGINT	Количество записей, прочитанных при помощи индекса (read via an index).
MON$RECORD_INSERTS	BIGINT	Количество добавленных записей (inserted records).
MON$RECORD_UPDATES	BIGINT	Количество измененных записей (updated records).
MON$RECORD_DELETES	BIGINT	Количество удаленных записей (deleted records).

Идентификатор столбца	Тип данных	Описание
MON$RECORD_BACKOUTS	BIGINT	Количество возвращенных в базу данных записей (backed out records). Это происходит при откате транзакции. Если обнаружена версия записи, созданная в результате отката транзакции, она помечается для сборки мусора, а предыдущая подтвержденная версия переносится в основной слот на странице данных.
MON$RECORD_PURGES	BIGINT	Количество удаленных ненужных записей (purged records). Это происходит, когда сборщик мусора удаляет старые версии записи, которые больше не нужны какой-либо активной транзакции.
MON$RECORD_EXPUNGES	BIGINT	Количество вычищенных средствами сборки мусора записей (expunged records). Это происходит, когда запись, которая не видна какой-либо активной транзакции, удаляется и удаляющая транзакция подтверждается (commit). Удаленная запись и все ранее подтвержденные версии этой записи удаляются, чтобы занятое ими дисковое пространство можно было повторно использовать. Если запись все еще видна для более старых snapshot транзакций, конечно, удаление может произойти только после завершения этих транзакций.
MON$RECORD_LOCKS	BIGINT	Количество записей прочитанных с использованием предложения WITH LOCK.
MON$RECORD_WAITS	BIGINT	Количество попыток обновления/модификации/блокировки записей принадлежащих нескольким активным транзакциям. Транзакция находится в режиме WAIT.
MON$RECORD_CONFLICTS	BIGINT	Количество неудачных попыток обновления/модификации/блокировки записей принадлежащих нескольким активным транзакциям. В таких ситуациях сообщается о конфликте обновления (UPDATE CONFLICT).
MON$BACKVERSION_READS	BIGINT	Количество прочитанных версий при поиске видимых версий записей.
MON$FRAGMENT_READS	BIGINT	Количество прочитанных фрагментов записей.
MON$RECORD_RPT_READS	BIGINT	Количество повторно прочитанных записей.

MON$TABLE_STATS

Таблица A.10 — Статистика на уровне таблицы

Идентификатор столбца	Тип данных	Описание
MON$STAT_ID	INTEGER	Идентификатор статистики.

Идентификатор столбца	Тип данных	Описание
MON$STAT_GROUP	SMALLINT	Группа статистики: 0 — база данных (database), 1 — соединение с базой данных (connection), 2 — транзакция (transaction), 3 — запрос (statement), 4 — вызов (call).
MON$TABLE_NAME	CHAR(31)	Имя таблицы.
MON$RECORD_STAT_ID	INTEGER	Ссылка на MON$RECORD_STATS.

MON$CONTEXT_VARIABLES

Таблица A.11 — Сведения о пользовательских контекстных переменных

Идентификатор столбца	Тип данных	Описание
MON$ATTACHMENT_ID	BIGINT	Идентификатор соединения. Содержит корректное значение только для контекстных переменных уровня соединения, для переменных уровня транзакции устанавливается в NULL.
MON$TRANSACTION_ID	BIGINT	Идентификатор транзакции. Содержит корректное значение только для контекстных переменных уровня транзакции, для переменных уровня соединения устанавливается в NULL.
MON$VARIABLE_NAME	VARCHAR(80)	Имя контекстной переменной.
MON$VARIABLE_VALUE	VARCHAR(255)	Значение контекстной переменной.

MON$MEMORY_USAGE

Таблица A.12 — Статистика использования памяти

Идентификатор столбца	Тип данных	Описание
MON$STAT_ID	INTEGER	Идентификатор статистики.
MON$STAT_GROUP	SMALLINT	Группа статистики: 0 — база данных (database), 1 — соединение с базой данных (connection), 2 — транзакция (transaction), 3 — запрос (statement), 4 — вызов (call).
MON$MEMORY_USED	BIGINT	Количество используемой памяти (в байтах). Информация о высокоуровневом распределении памяти, выполненной сервером из пулов. Может быть полезна для отслеживания утечек памяти и чрезмерного потребления памяти в соединениях, процедурах и т.д.

Идентификатор столбца	Тип данных	Описание
MON$MEMORY_ALLOCATED	BIGINT	Количество памяти, выделенной ОС (в байтах). Информация о низкоуровневом распределении памяти, выполненном менеджером памяти — объем памяти, выделенный операционной системой, что позволяет контролировать физическое потребление памяти. Обратите внимание, не все записи этого столбца имеют ненулевые значения. Малые выделения памяти здесь не фиксируются, а вместо этого добавляются к пулу памяти базы данных. Только MON$DATABASE (MON$STAT_GROUP = 0) и связанные с выделением памяти объекты имеют ненулевое значение.
MON$MAX_MEMORY_USED	BIGINT	Максимальное количество байт, используемое данным объектом.
MON$MAX_MEMORY_ALLOCATED	BIGINT	Максимальное количество байт, выделенное ОС данному объекту.

Приложение Б Описание системных таблиц

При первоначальном создании базы данных система управления базами данных создает множество системных таблиц. В системных таблицах хранятся метаданные — описания всех объектов базы данных.

Список системных таблиц в алфавитном порядке представлен в таблице Б.1.

Таблица Б.1 — Список системных таблиц «Ред База Данных»

Таблица	Содержание
RDB$AUTH_MAPPING	Сведения об отображении объектов безопасности
RDB$BACKUP_HISTORY	Хранит историю копирования базы данных
RDB$CHARACTER_SETS	Доступные в базе данных наборы символов
RDB$CHECK_CONSTRAINTS	Соответствие имен триггеров именам ограничений, связанных с характеристиками NOT NULL, ограничениями CHECK и предложениями ON UPDATE и ON DELETE в ограничениях внешнего ключа
RDB$COLLATIONS	Порядки сортировки для всех наборов символов
RDB$DATABASE	Основные данные о базе данных
RDB$DB_CREATORS	Содержит сведения о пользователях имеющих права на создание базы данных. Используется только в том случае, если текущая база данных назначена как база данных безопасности.
RDB$DEPENDENCIES	Сведения о зависимостях между объектами базы данных
RDB$EXCEPTIONS	Пользовательские исключения базы данных
RDB$FIELDS	Характеристики столбцов и доменов, как системных, так и созданных пользователем
RDB$FIELD_DIMENSIONS	Размерности столбцов, являющихся массивами
RDB$FILES	Сведения о вторичных файлах и файлах оперативных копий
RDB$FILTERS	Данные о BLOB – фильтрах
RDB$FORMATS	Данные об изменениях таблиц
RDB$FUNCTIONS	Сведения о внешних или хранимых функциях
RDB$FUNCTION_ARGUMENTS	Характеристики параметров внешних или хранимых функций
RDB$GENERATORS	Сведения о генераторах (последовательностях)
RDB$INDEX_SEGMENTS	Сегменты и позиции индексов
RDB$INDICES	Определение индексов базы данных (созданных пользователем или системой)
RDB$LOG_FILES	В настоящей версии не используется
RDB$PACKAGES	Сведения о PSQL пакетах
RDB$PAGES	Сведения о страницах базы данных
RDB$PROCEDURE_PARAMETERS	Параметры хранимых процедур

Таблица	Содержание
RDB$PROCEDURES	Описания хранимых процедур
RDB$REF_CONSTRAINTS	Описания именованных ограничений базы данных (внешних ключей)
RDB$RELATION_CONSTRAINTS	Описание всех ограничений на уровне таблиц
RDB$RELATION_FIELDS	Характеристики столбцов таблиц
RDB$RELATIONS	Заголовки таблиц и представлений
RDB$ROLES	Определение ролей
RDB$SECURITY_CLASSES	Списки управления доступом
RDB$TRANSACTIONS	Состояние транзакций при обращении к нескольким базам данных
RDB$TRIGGER_MESSAGES	Сообщения триггеров
RDB$TRIGGERS	Описания триггеров
RDB$TYPES	Описание перечислимых типов данных
RDB$USER_PRIVILEGES	Полномочия пользователей системы
RDB$VIEW_RELATIONS	Описывает представления. Не используется в настоящей версии

RDB$AUTH_MAPPING

Таблица содержит сведения об отображении объектов безопасности.

Идентификатор столбца	Тип данных	Описание
RDB$MAP_NAME	CHAR(31)	Имя отображения.
RDB$MAP_USING	CHAR(1)	Является ли аутентификация общесерверной (S) или обычной (P).
RDB$MAP_PLUGIN	CHAR(31)	Имя плагина аутентификации, из которого происходит отображение.
RDB$MAP_DB	CHAR(31)	Имя базы данных, в которой прошла аутентификация. Из неё происходит отображение.
RDB$MAP_FROM_TYPE	CHAR(31)	Тип объекта, который будет отображён.
RDB$MAP_FROM	CHAR(255)	Имя объекта, из которого будет произведено отображение.
RDB$MAP_TO_TYPE	SMALLINT	Тип объекта, в который будет произведено отображение: 0 — USER; 1 — ROLE.
RDB$MAP_TO	CHAR(31)	Наименование объекта, в который будет произведено отображение (имя пользователя или роли).
RDB$SYSTEM_FLAG	SMALLINT	0 – определён пользователем; 1 –определён в системе.
RDB$DESCRIPTION	BLOB TEXT	Произвольное текстовое описание порядка сортировки.

RDB$BACKUP_HISTORY

Таблица хранит историю копирования базы данных при помощи утилиты `nbackup`.

Идентификатор столбца	Тип данных	Описание
RDB$BACKUP_ID	INTEGER	Присваиваемый системой идентификатор.
RDB$TIMESTAMP	TIMESTAMP	Дата и время выполнения копирования.
RDB$BACKUP_LEVEL	INTEGER	Уровень копирования.
RDB$GUID	CHAR(38)	Уникальный идентификатор.
RDB$SCN	INTEGER	Системный номер.
RDB$FILE_NAME	VARCHAR(255)	Полный путь и имя файла копии.

RDB$CHARACTER_SETS

Содержит наборы символов, доступные в базе данных.

Идентификатор столбца	Тип данных	Описание
RDB$CHARACTER_SET_NAME	CHAR(31)	Имя набора символов.
RDB$FORM_OF_USE	CHAR(31)	Не используется.
RDB$NUMBER_OF_CHARACTERS	INTEGER	Количество символов в наборе. Для существующих наборов символов не используется.
RDB$DEFAULT_COLLATE_NAME	CHAR(31)	Имя порядка сортировки по умолчанию для набора символов.
RDB$CHARACTER_SET_ID	SMALLINT	Уникальный идентификатор набора символов.
RDB$SYSTEM_FLAG	SMALLINT	Системный флаг: имеет значение 1, если набор символов был определен в системе при создании базы данных; значение 0 для набора символов, определенного пользователем.
RDB$DESCRIPTION	BLOB TEXT	Произвольное текстовое описание набора символов.
RDB$FUNCTION_NAME	CHAR(31)	Имя внешней функции для наборов символов, определенных пользователем, доступ к которым осуществляется через внешнюю функцию.
RDB$BYTES_PER_CHARACTER	SMALLINT	Количество байтов для представления одного символа.
RDB$SECURITY_CLASS	CHAR(31)	Может ссылаться на класс безопасности, определённый в таблице RDB$SECURITY_CLASSES для применения ограничений управления доступом для всех пользователей этого набора символов.
RDB$OWNER_NAME	CHAR(31)	Имя пользователя — владельца (создателя) набора символов.

RDB$CHECK_CONSTRAINTS

Описывает соответствие имен триггеров именам ограничений, связанных с характеристиками NOT NULL, ограничениями CHECK и предложениями ON UPDATE, ON DELETE в ограничениях внешнего ключа.

Идентификатор столбца	Тип данных	Описание
RDB$CONSTRAINT_NAME	CHAR(31)	Имя ограничения. Задается пользователем или автоматически генерируется системой.
RDB$TRIGGER_NAME	CHAR(31)	Для ограничения CHECK — это имя триггера, который поддерживает данное ограничение. Для ограничения NOT NULL — это имя столбца, к которому применяется ограничение. Для ограничения внешнего ключа — это имя триггера, который поддерживает предложения ON UPDATE, ON DELETE.

RDB$COLLATIONS

Порядки сортировки для наборов символов.

Идентификатор столбца	Тип данных	Описание
RDB$COLLATION_NAME	CHAR(31)	Имя порядка сортировки.
RDB$COLLATION_ID	SMALLINT	Идентификатор порядка сортировки. Вместе с идентификатором набора символов является уникальным идентификатором порядка сортировки.
RDB$CHARACTER_SET_ID	SMALLINT	Идентификатор набора символов. Вместе с идентификатором порядка сортировки является уникальным идентификатором.
RDB$COLLATION_ATTRIBUTES	SMALLINT	Атрибуты сортировки. Представляет собой битовую маску, где 1-й бит показывает учитывать ли конечные пробелы при сравнении (0 - NO PAD; 1 - PAD SPACE); 2-й бит показывает является ли сравнение чувствительным к регистру символов (0 - CASE SENSITIVE, 1 - CASE INSENSITIVE); 3-й бит показывает будет ли сравнение чувствительным к акцентам (0 - ACCENT SENSITIVE, 1 - ACCENT INSENSITIVE). Таким образом, значение 5 означает, что сравнение не является чувствительным к конечным пробелам и к акцентированным буквам.
RDB$SYSTEM_FLAG	SMALLINT	Признак: определен пользователем — значение 0; определен в системе — значение 1.
RDB$DESCRIPTION	BLOB TEXT	Произвольное текстовое описание порядка сортировки.
RDB$FUNCTION_NAME	CHAR(31)	В настоящий момент не используется.
RDB$BASE_COLLATION_NAME	CHAR(31)	Имя базового порядка сортировки для данного порядка сортировки.

Идентификатор столбца	Тип данных	Описание
RDB$SPECIFIC_ATTRIBUTES	BLOB TEXT	Описание особых атрибутов.
RDB$SECURITY_CLASS	CHAR(31)	Может ссылаться на класс безопасности, определённый в таблице RDB$SECURITY_CLASSES для применения ограничений управления доступом для всех пользователей этой сортировки.
RDB$OWNER_NAME	CHAR(31)	Имя пользователя — владельца (создателя) сортировки.

RDB$DATABASE

Основные данные о базе данных. Содержит только одну запись.

Идентификатор столбца	Тип данных	Описание
RDB$DESCRIPTION	BLOB TEXT	Текст примечания базы данных.
RDB$RELATION_ID	SMALLINT	Количество таблиц и представлений в базе данных.
RDB$SECURITY_CLASS	CHAR(31)	Класс безопасности, определенный в RDB$SECURITY_CLASSES, для обращения к общим для базы данных ограничениям доступа.
RDB$CHARACTER_SET_NAME	CHAR(31)	Имя набора символов по умолчанию для базы данных, установленного в предложении DEFAULT CHARACTER SET при создании базы данных. NULL — набор символов NONE.
RDB$LINGER	INTEGER	Количество секунд "задержки" (установленной оператором alter database set linger) до закрытия последнего соединения базы данных (в SuperServer). Если задержка не установлена, то содержит NULL.
RDB$SQL_SECURITY	BOOLEAN	Определяет в контексте какого пользователя будет выполняться объект базы данных (процедура, функция, пакет, триггер, таблица) по умолчанию. Если установлен в FALSE, то объект выполняется с правами вызвавшего его пользователя. Иначе объект выполняется с правами его владельца (создателя)

RDB$DB_CREATORS

Содержит сведения о пользователях имеющих права на создание базы данных. Используется только в том случае, если текущая база данных назначена как база данных безопасности.

Идентификатор столбца	Тип данных	Описание
RDB$USER	CHAR(31)	Имя пользователя или роли, которому даны полномочия на создание базы данных.

Идентификатор столбца	Тип данных	Описание
RDB$USER_TYPE	SMALLINT	Тип пользователя: 8 — пользователь; 13 — роль.

RDB$DEPENDENCIES

Сведения о зависимостях между объектами базы данных.

Идентификатор столбца	Тип данных	Описание
RDB$DEPENDENT_NAME	CHAR(31)	Имя представления, процедуры, триггера, ограничения CHECK или вычисляемого столбца, для которого описывается зависимость.
RDB$DEPENDED_ON_NAME	CHAR(31)	Объект, зависящий от описываемого объекта — таблица, на которую ссылается представление, процедура, триггер, ограничение CHECK или вычисляемый столбец.
RDB$FIELD_NAME	CHAR(31)	Имя столбца в зависимой таблице, на который ссылается представление, процедура, триггер, ограничение CHECK или вычисляемый столбец.
RDB$DEPENDENT_TYPE	SMALLINT	Идентифицирует тип описываемого объекта: 0 – таблица, 1 – представление, 2 – триггер, 3 – вычисляемый столбец, 4 – ограничение CHECK, 5 – процедура, 6 – выражение для индекса, 7 – исключение, 8 – пользователь, 9 –столбец, 10 – индекс, 15 — хранимая функция, 18 — заголовок пакета, 19 — тело пакета.

Идентификатор столбца	Тип данных	Описание
RDB$DEPENDED_ON_TYPE	SMALLINT	Идентифицирует тип зависимого объекта: 0 – таблица (или её столбец);, 1 – представление, 2 – триггер, 3 – вычисляемый столбец, 4 – ограничение CHECK, 5 – процедура, 6 – выражение для индекса, 7 – исключение, 8 – пользователь, 9 –столбец, 10 – индекс, 14 — генератор (последовательность), 15 — UDF или хранимая функция, 17 — сортировка, 18 — заголовок пакета, 19 — тело пакета.
RDB$PACKAGE_NAME	CHAR(31)	Пакет процедуры или функции, для которой описывается зависимость.

RDB$EXCEPTIONS

Пользовательские исключения базы данных.

Идентификатор столбца	Тип данных	Описание
RDB$EXCEPTION_NAME	CHAR(31)	Имя пользовательского исключения.
RDB$EXCEPTION_NUMBER	INTEGER	Назначенный системой уникальный номер исключения.
RDB$MESSAGE	VARCHAR(1023)	Текст сообщения в исключении.
RDB$DESCRIPTION	BLOB TEXT	Произвольное текстовое описание исключения.
RDB$SYSTEM_FLAG	SMALLINT	Признак: определено пользователем = 0; определено системой = 1 или выше.
RDB$SECURITY_CLASS	CHAR(31)	Может ссылаться на класс безопасности, определённый в таблице RDB$SECURITY_CLASSES для применения ограничений управления доступом для всех пользователей этого исключения.
RDB$OWNER_NAME	CHAR(31)	Имя пользователя — владельца (создателя) исключения.

RDB$FIELDS

Характеристики столбцов и доменов, как системных, так и созданных пользователем.

Идентификатор столбца	Тип данных	Описание
RDB$FIELD_NAME	CHAR(31)	Уникальное имя домена, созданного пользователем, или домена, автоматически построенного системой для столбца таблицы. Во втором случае имя будет начинаться с символов 'RDB$'.
RDB$QUERY_NAME	CHAR(31)	Не используется.
RDB$VALIDATION_BLR	BLOB BLR	Двоичное представление (BLR) выражения SQL, задающее проверку значения CHECK у домена.
RDB$VALIDATION_SOURCE	BLOB TEXT	Оригинальный исходный текст на языке SQL, задающий проверку значения CHECK.
RDB$COMPUTED_BLR	BLOB BLR	Двоичное представление (BLR) выражения SQL, которое используется сервером базы данных для вычисления при обращении к столбцу COMPUTED BY.
RDB$COMPUTED_SOURCE	BLOB TEXT	Оригинальный исходный текст выражения, которое определяет столбец COMPUTED BY.
RDB$DEFAULT_VALUE	BLOB BLR	Значение по умолчанию в двоичном виде BLR.
RDB$DEFAULT_SOURCE	BLOB TEXT	Значение по умолчанию в исходном виде на языке SQL.
RDB$FIELD_LENGTH	SMALLINT	Размер столбца в байтах. FLOAT, DATE, TIME, INTEGER занимают 4 байта. DOUBLE PRECISION, BIGINT, TIMESTAMP и идентификатор BLOB — 8 байтов. Для типов данных CHAR и VARCHAR столбец задает максимальное количество байтов, указанное при объявлении строкового домена (столбца).
RDB$FIELD_SCALE	SMALLINT	Отрицательное число задает масштаб для столбцов DECIMAL и NUMERIC — количество дробных знаков после десятичной точки.
RDB$FIELD_TYPE	SMALLINT	Код типа данных для столбца: 7 = SMALLINT, 8 = INTEGER, 10 = FLOAT, 12 = DATE, 13 = TIME, 14 = CHAR, 16 = BIGINT, 27 = DOUBLE PRECISION, 35 = TIMESTAMP, 37 = VARCHAR, 261 = BLOB. Коды для DECIMAL и NUMERIC имеют тот же размер, что и целые типы, используемые для их хранения.

Идентификатор столбца	Тип данных	Описание
RDB$FIELD_SUB_TYPE	SMALLINT	Для типа данных BLOB задает подтип: 0 – не определен, 1 – текст, 2 – BLR, 3 – список управления доступом, 4 – резервируется для дальнейшего использования, 5 – кодированное описание метаданных таблицы, 6 – описание транзакции к нескольким базам данных, которая не завершилась нормально. Для типа данных CHAR задает: 0 – неопределенные данные, 1 – фиксированные двоичные данные. Для целочисленных типов данных (SMALLINT, INTEGER, BIGINT) и чисел с фиксированной точкой (NUMERIC, DECIMAL) задает конкретный тип данных: 0 или NULL – тип данных соответствует значению в поле RDB$FIELD_TYPE, 1 – NUMERIC, 2 – DECIMAL.
RDB$MISSING_VALUE	BLOB BLR	Не используется.
RDB$MISSING_SOURCE	BLOB TEXT	Не используется.
RDB$DESCRIPTION	BLOB TEXT	Произвольный текст комментария для домена (столбца таблицы).
RDB$SYSTEM_FLAG	SMALLINT	Признак: значение 1 — домен, автоматически созданный системой, значение 0 — домен определен пользователем.
RDB$QUERY_HEADER	BLOB TEXT	Не используется.
RDB$SEGMENT_LENGTH	SMALLINT	Для столбцов BLOB задает длину буфера BLOB в байтах. Для остальных типов данных содержит NULL.
RDB$EDIT_STRING	VARCHAR(127)	Не используется.
RDB$EXTERNAL_LENGTH	SMALLINT	Длина столбца в байтах, если он входит в состав внешней таблицы. Всегда NULL для обычных таблиц.
RDB$EXTERNAL_SCALE	SMALLINT	Показатель степени для столбца целого типа данных во внешней таблице; задается степенью 10, на которую умножается целое.

Идентификатор столбца	Тип данных	Описание
RDB$EXTERNAL_TYPE	SMALLINT	Тип данных поля, как он представляется во внешней таблице. 7 = SMALLINT, 8 = INTEGER, 10 = FLOAT, 12 = DATE, 13 = TIME, 14 = CHAR, 16 = BIGINT, 23 = BOOLEAN, 27 = DOUBLE PRECISION, 35 = TIMESTAMP, 37 = VARCHAR. Коды для DECIMAL и NUMERIC имеют тот же размер, что и целые типы, используемые для их хранения.
RDB$DIMENSIONS	SMALLINT	Задает количество размерностей массива, если столбец был определен как массив. Для столбцов, не являющихся массивами, всегда NULL.
RDB$NULL_FLAG	SMALLINT	Указывает, может ли столбец принимать пустое значение (в поле будет значение NULL) или не может (в поле будет содержаться значение 1).
RDB$CHARACTER_LENGTH	SMALLINT	Длина столбцов CHAR или VARCHAR в символах (не в байтах).
RDB$COLLATION_ID	SMALLINT	Идентификатор порядка сортировки для символьного столбца или домена. Если не задан, значением поля будет 0.
RDB$CHARACTER_SET_ID	SMALLINT	Идентификатора набора символов для символьного столбца, столбца BLOB или домена.
RDB$FIELD_PRECISION	SMALLINT	Указывает общее количество цифр для числового типа данных с фиксированной точкой (DECIMAL и NUMERIC). Для целочисленных типов данных значением является 0, для всех остальных типов данных — NULL.
RDB$SECURITY_CLASS	CHAR(31)	Может ссылаться на класс безопасности, определённый в таблице RDB$SECURITY_CLASSES для применения ограничений управления доступом для всех пользователей этого домена.
RDB$OWNER_NAME	CHAR(31)	Имя пользователя – владельца (создателя) домена.

RDB$FIELD_DIMENSIONS

Размерности столбцов, являющихся массивами.

Идентификатор столбца	Тип данных	Описание
RDB$FIELD_NAME	CHAR(31)	Имя столбца, являющегося массивом. Должно содержаться в поле RDB$FIELD_NAME таблицы RDB$FIELDS.
RDB$DIMENSION	SMALLINT	Определяет одну размерность столбца массива. Нумерация размерностей начинается с 0.
RDB$LOWER_BOUND	INTEGER	Нижняя граница этой размерности.
RDB$UPPER_BOUND	INTEGER	Верхняя граница описываемой размерности.

RDB$FILES

Сведения о вторичных файлах и файлах оперативных копий.

Идентификатор столбца	Тип данных	Описание
RDB$FILE_NAME	VARCHAR(255)	Полный путь к файлу и имя вторичного файла базы данных в многофайловой базе данных или файла оперативной копии.
RDB$FILE_SEQUENCE	SMALLINT	Порядковый номер вторичного файла в последовательности или номер файла копии в наборе оперативных копий.
RDB$FILE_START	INTEGER	Начальный номер страницы вторичного файла или файла оперативной копии.
RDB$FILE_LENGTH	INTEGER	Длина файла в страницах базы данных.
RDB$FILE_FLAGS	SMALLINT	Для внутреннего использования.
RDB$SHADOW_NUMBER	SMALLINT	Номер набора оперативных копий. Если строка описывает вторичный файл базы данных, то значением поля будет NULL или 0.

RDB$FILTERS

Содержит данные о BLOB -фильтрах.

Идентификатор столбца	Тип данных	Описание
RDB$FUNCTION_NAME	CHAR(31)	Уникальное имя фильтра BLOB.
RDB$DESCRIPTION	BLOB TEXT	Написанная пользователем документация о фильтре BLOB и используемых двух подтипах.
RDB$MODULE_NAME	VARCHAR(255)	Имя динамической библиотеки / совместно используемого объекта, где расположен код фильтра BLOB.
RDB$ENTRYPOINT	CHAR(255)	Точка входа в библиотеке фильтров для этого фильтра BLOB.
RDB$INPUT_SUB_TYPE	SMALLINT	Подтип BLOB для преобразуемых данных.
RDB$OUTPUT_SUB_TYPE	SMALLINT	Подтип BLOB, в который преобразуются входные данные.

Идентификатор столбца	Тип данных	Описание
RDB$SYSTEM_FLAG	SMALLINT	Признак: внешне определенный фильтр (т.е. определенный пользователем = значение 0, внутренне определенный = значение 1 или более)
RDB$SECURITY_CLASS	CHAR(31)	Может ссылаться на класс безопасности, определённый в таблице RDB$SECURITY_CLASSES для применения ограничений управления доступом для всех пользователей этого BLOB фильтра.
RDB$OWNER_NAME	CHAR(31)	Имя пользователя – владельца (создателя) BLOB фильтра.

RDB$FORMATS

Данные об изменениях таблиц. Каждый раз, когда таблица изменяется, таблица получает новый номер формата. Когда номер формата любой таблицы достигает 255, вся база данных становится недоступной для работы с ней. Тогда нужно выполнить резервное копирование, восстановить эту копию и продолжить работу с заново созданной базой данных.

Идентификатор столбца	Тип данных	Описание
RDB$RELATION_ID	SMALLINT	Идентификатор таблицы или представления.
RDB$FORMAT	SMALLINT	Идентификатор формата таблицы. Форматов может быть до 255.
RDB$DESCRIPTOR	BLOB FORMAT	Отображение в виде BLOB столбцов и характеристик данных на момент, когда была создана запись формата.

RDB$FUNCTIONS

Сведения о внешних или хранимых функциях.

Идентификатор столбца	Тип данных	Описание
RDB$FUNCTION_NAME	CHAR(31)	Уникальное имя внешней функции.
RDB$FUNCTION_TYPE	SMALLINT	В настоящий момент не используется.
RDB$QUERY_NAME	CHAR(31)	В настоящий момент не используется.
RDB$DESCRIPTION	BLOB TEXT	Произвольный текст комментария к внешней функции.
RDB$MODULE_NAME	VARCHAR(255)	Имя модуля (библиотеки DLL), где расположен код функции.
RDB$ENTRYPOINT	CHAR(255)	Имя точки входа в библиотеке, где находится эта функция.
RDB$RETURN_ARGUMENT	SMALLINT	Номер позиции возвращаемого аргумента в списке параметров, соответствующем входным аргументам.

Идентификатор столбца	Тип данных	Описание
RDB$SYSTEM_FLAG	SMALLINT	Признак определения функции: определенная пользователем = 1, определенная системой = 0.
RDB$ENGINE_NAME	CHAR(31)	Имя движка для использования внешних функций. Обычно UDR.
RDB$PACKAGE_NAME	CHAR(31)	Имя пакета, если функция является упакованной.
RDB$PRIVATE_FLAG	SMALLINT	Для неупакованных хранимых функций всегда NULL, для упакованных 0 — если функция описана в заголовке пакета и 1 — если функция описана или реализована только в теле пакета (не описана в заголовке).
RDB$FUNCTION_SOURCE	BLOB TEXT	Исходный код функции на языке SQL.
RDB$FUNCTION_ID	SMALLINT	Уникальный идентификатор функции.
RDB$FUNCTION_BLR	BLOB BLR	Двоичное представление (BLR) кода функции.
RDB$VALID_BLR	SMALLINT	Указывает, остается ли текст хранимой функции корректным после последнего изменения функции при помощи оператора ALTER FUNCTION.
RDB$DEBUG_INFO	BLOB	Содержит отладочную информацию о переменных, используемых в хранимой функции.
RDB$SECURITY_CLASS	CHAR(31)	Может ссылаться на класс безопасности, определённый в таблице RDB$SECURITY_CLASSES для применения ограничений управления доступом.
RDB$OWNER_NAME	CHAR(31)	Имя пользователя – владельца (создателя) функции.
RDB$LEGACY_FLAG	SMALLINT	Признак legacy стиля функции. 1 — если функция описана в legacy стиле (DECLARE EXTERNAL FUNCTION), в противном случае 0 (CREATE FUNCTION).
RDB$DETERMINISTIC_FLAG	SMALLINT	Флаг детерминистической функции. 1 - если функция детерминистическая (DETERMINISTIC), в противном случае - 0.
RDB$SQL_SECURITY	BOOLEAN	Определяет в контексте какого пользователя будет выполняться функция. Если установлен в FALSE, то функция выполняется с правами вызвавшего его пользователя. Иначе функция выполняется с правами его владельца (создателя)

RDB$FUNCTION_ARGUMENTS

Характеристики параметров внешних или хранимых функций.

Идентификатор столбца	Тип данных	Описание
RDB$FUNCTION_NAME	CHAR(31)	Уникальное имя внешней функции.
RDB$ARGUMENT_POSITION	SMALLINT	Позиция аргумента в списке аргументов.
RDB$MECHANISM	SMALLINT	Механизм передачи параметра для Legacy функций: передается ли аргумент по значению (значение столбца 0), по ссылке (значение 1), через дескриптор (значение 2) или через дескриптор BLOB (значение 3).
RDB$FIELD_TYPE	SMALLINT	Число, задающее код типа данных, определенного для столбца: 7 = SMALLINT, 8 = INTEGER, 12 = DATE, 13 = TIME, 14 = CHAR, 16 = BIGINT, 23 = BOOLEAN; 27 = DOUBLE PRECISION, 35 = TIMESTAMP, 37 = VARCHAR, 40 = CSTRING (строка, завершаемая нулем), 45 = blob_id, 261 = BLOB.
RDB$FIELD_SCALE	SMALLINT	Масштаб для целого числа или аргумента с фиксированной точкой. Это показатель числа 10.
RDB$FIELD_LENGTH	SMALLINT	Длина аргумента в байтах. BOOLEAN = 1, SMALLINT = 2, INTEGER = 4, DATE = 4, TIME = 4, BIGINT = 8, DOUBLE PRECISION = 8, TIMESTAMP = 8, blob_id = 8.
RDB$FIELD_SUB_TYPE	SMALLINT	Для аргумента типа данных BLOB задает подтип BLOB.
RDB$CHARACTER_SET_ID	SMALLINT	Идентификатор набора символов для символьного аргумента.
RDB$FIELD_PRECISION	SMALLINT	Количество цифр точности, допустимой для типа данных аргумента.
RDB$CHARACTER_LENGTH	SMALLINT	Длина аргумента CHAR или VARCHAR в символах (но не в байтах).
RDB$PACKAGE_NAME	CHAR(31)	Имя пакета функции (если функция упакованная), в которой используется параметр.
RDB$ARGUMENT_NAME	CHAR(31)	Имя параметра.

Идентификатор столбца	Тип данных	Описание
RDB$FIELD_SOURCE	CHAR(31)	Имя домена, созданного пользователем (при использовании ссылки на домен вместо типа), или домена, автоматически построенного системой для параметра функции. Во втором случае имя будет начинаться с символов 'RDB$'.
RDB$DEFAULT_VALUE	BLOB BLR	Значение по умолчанию на языке BLR.
RDB$DEFAULT_SOURCE	BLOB TEXT	Значение по умолчанию в исходном виде на языке SQL.
RDB$COLLATION_ID	SMALLINT	Идентификатор используемого порядка сортировки для символьного параметра.
RDB$NULL_FLAG	SMALLINT	Признак допустимости пустого значения NULL.
RDB$ARGUMENT_MECHANISM	SMALLINT	Механизм передачи параметра для не Legacy функций: передается ли аргумент по значению (значение столбца 0), по ссылке (значение 1), через дескриптор (значение 2) или через дескриптор BLOB (значение 3).
RDB$FIELD_NAME	CHAR(31)	Имя столбца, на которое ссылается параметр с помощью предложения TYPE OF COLUMN.
RDB$RELATION_NAME	CHAR(31)	Имя таблицы, на которую ссылается параметр с помощью предложения TYPE OF COLUMN.
RDB$SYSTEM_FLAG	SMALLINT	Указывает, является ли параметр определённым системой (значение 1 и выше) или пользователем (значение 0).
RDB$DESCRIPTION	BLOB TEXT	Текст произвольного примечания к параметру.

RDB$GENERATORS

Сведения о генераторах (последовательностях).

Идентификатор столбца	Тип данных	Описание
RDB$GENERATOR_NAME	CHAR(31)	Уникальное имя генератора.
RDB$GENERATOR_ID	SMALLINT	Назначаемый системой уникальный идентификатор для генератора.
RDB$SYSTEM_FLAG	SMALLINT	Признак: значение 0 — генератор определен пользователем, значение 1 или выше — определен системой, 6 — внутренний генератор для identity столбца.
RDB$DESCRIPTION	BLOB TEXT	Произвольный текст примечания к генератору.
RDB$SECURITY_CLASS	CHAR(31)	Может ссылаться на класс безопасности, определённый в таблице RDB$SECURITY_CLASSES для применения ограничений управления доступом.
RDB$OWNER_NAME	CHAR(31)	Имя пользователя – владельца (создателя) генератора.

Идентификатор столбца	Тип данных	Описание
RDB$INITIAL_VALUE	BIGINT	Хранит начальное значение генератора или значение генератора, установленное при предыдущем рестарте (WITH RESTART).
RDB$GENERATOR_INCREMENT	INTEGER	Шаг приращения генератора при использовании оператора NEXT VALUE FOR.

RDB$INDEX_SEGMENTS

Сегменты и позиции индексов. Таблица описывает все столбцы таблицы, входящие в состав конкретного индекса. Для каждого столбца индекса создается отдельная строка в данной таблице.

Идентификатор столбца	Тип данных	Описание
RDB$INDEX_NAME	CHAR(31)	Имя индекса, к которому относится данный сегмент. Должно соответствовать главной записи в системной таблице RDB$INDICES.
RDB$FIELD_NAME	CHAR(31)	Имя одного из столбцов, входящего в состав индекса. Должно соответствовать значению в столбце RDB$FIELD_NAME в таблице RDB$RELATION_FIELDS.
RDB$FIELD_POSITION	SMALLINT	Позиция столбца в индексе. Нумерация начинается с нуля.
RDB$STATISTICS	DOUBLE PRECISION	Селективность индекса по данному столбцу.

RDB$INDICES

Определение индексов базы данных (созданных пользователем или системой). Описывает каждый индекс, созданный пользователем или системой. Для каждого столбца таблицы, входящего в состав индекса, присутствует строка системной таблицы RDB$INDEX_SEGMENTS, где описываются характеристики столбца индекса.

Идентификатор столбца	Тип данных	Описание
RDB$INDEX_NAME	CHAR(31)	Уникальное имя индекса, заданное пользователем или автоматически сгенерированное системой.
RDB$RELATION_NAME	CHAR(31)	Имя таблицы, к которой применяется индекс. Соответствует RDB$RELATION_NAME в строке таблицы RDB$RELATIONS.
RDB$INDEX_ID	SMALLINT	Внутренний (системный) идентификатор индекса.
RDB$UNIQUE_FLAG	SMALLINT	Указывает, является ли индекс уникальным: 1 — уникальный, 0 — не уникальный.
RDB$DESCRIPTION	BLOB TEXT	Произвольный текст комментария к индексу.
RDB$SEGMENT_COUNT	SMALLINT	Количество сегментов (столбцов) в индексе.

Идентификатор столбца	Тип данных	Описание
RDB$INDEX_INACTIVE	SMALLINT	Указывает, является ли в настоящий момент индекс активным: 1 — неактивный, 0 — активный.
RDB$INDEX_TYPE	SMALLINT	Упорядоченность индекса: 1 - по убыванию значений столбцов, 0 - по возрастанию значений столбцов.
RDB$FOREIGN_KEY	VARCHAR(31)	Имя ассоциированного ограничения внешнего ключа, если существует.
RDB$SYSTEM_FLAG	SMALLINT	Указывает, является ли индекс определенным системой (значение 1 или выше) или пользователем (значение 0).
RDB$EXPRESSION_BLR	BLOB BLR	Выражение, записанное на языке двоичного представления (BLR). Будет использовано для вычисления во время выполнения, когда будут реализованы индексы выражений.
RDB$EXPRESSION_SOURCE	BLOB TEXT	Исходный текст выражения. Будет использовано, когда будут реализованы индексы выражений.
RDB$STATISTICS	DOUBLE PRECISION	Хранит самую последнюю селективность индекса, вычисленную при помощи оператора SET STATISTICS.

RDB$LOG_FILES

В настоящей версии не используется.

Идентификатор столбца	Тип данных	Описание
RDB$FILE_NAME	VARCHAR(255)	Не используется.
RDB$FILE_SEQUENCE	SMALLINT	Не используется.
RDB$FILE_LENGTH	INTEGER	Не используется.
RDB$FILE_PARTITIONS	SMALLINT	Не используется.
RDB$FILE_P_OFFSET	INTEGER	Не используется.
RDB$FILE_FLAGS	SMALLINT	Не используется.

RDB$PACKAGES

В таблице содержатся сведения о PSQL пакетах.

Идентификатор столбца	Тип данных	Описание
RDB$PACKAGE_NAME	CHAR(31)	Уникальное имя пакета.
RDB$PACKAGE_HEADER_SOURCE	BLOB TEXT	Исходный код заголовка пакета на языке SQL.
RDB$PACKAGE_BODY_SOURCE	BLOB TEXT	Исходный код тела пакета на языке SQL.

Идентификатор столбца	Тип данных	Описание
RDB$VALID_BODY_FLAG	SMALLINT	Указывает, остаётся ли текст тела пакета корректным после последнего изменения заголовка пакета или его пересоздания.
RDB$SECURITY_CLASS	CHAR(31)	Может указывать на класс безопасности, определённый в системной таблице RDB$SECURITY_CLASSES, для применения ограничений управления доступом.
RDB$OWNER_NAME	CHAR(31)	Имя пользователя – владельца (создателя) пакета.
RDB$SYSTEM_FLAG	SMALLINT	Указывает, что пакет определён пользователем (значение 0) или системой (значение 1 или выше).
RDB$DESCRIPTION	BLOB TEXT	Произвольный текст примечания к пакету.
RDB$SQL_SECURITY	BOOLEAN	Определяет в контексте какого пользователя будут выполняться процедуры и функции. Если установлен в FALSE, то они выполняется с правами вызвавшего его пользователя. Иначе они выполняется с правами его владельца (создателя)

RDB$PAGES

Сведения о страницах базы данных.

Идентификатор столбца	Тип данных	Описание
RDB$PAGE_NUMBER	INTEGER	Уникальный номер физически созданной страницы базы данных.
RDB$RELATION_ID	SMALLINT	Идентификатор таблицы, для которой выделена эта страница.
RDB$PAGE_SEQUENCE	INTEGER	Последовательный номер страницы по отношению к другим страницам, выделенным для данной таблицы.
RDB$PAGE_TYPE	SMALLINT	Описывает тип страницы. Для системного использования.

RDB$PROCEDURE_PARAMETERS

Описывает параметры хранимых процедур.

Идентификатор столбца	Тип данных	Описание
RDB$PARAMETER_NAME	CHAR(31)	Имя параметра.
RDB$PROCEDURE_NAME	CHAR(31)	Имя процедуры, в которой используется параметр.
RDB$PARAMETER_NUMBER	SMALLINT	Последовательный номер параметра.
RDB$PARAMETER_TYPE	SMALLINT	Указывает, является ли параметр входным (значение 0) или выходным (значение 1).

Идентификатор столбца	Тип данных	Описание
RDB$FIELD_SOURCE	CHAR(31)	Сгенерированное системой уникальное глобальное имя столбца.
RDB$DESCRIPTION	BLOB TEXT	Текст произвольного примечания к параметру.
RDB$SYSTEM_FLAG	SMALLINT	Указывает, является ли параметр определенным системой (значение 1 и выше) или пользователем (значение 0).
RDB$DEFAULT_VALUE	BLOB BLR	Значение по умолчанию на языке BLR.
RDB$DEFAULT_SOURCE	BLOB TEXT	Значение по умолчанию в исходном виде на языке SQL.
RDB$COLLATION_ID	SMALLINT	Идентификатор используемого порядка сортировки для символьного параметра.
RDB$NULL_FLAG	SMALLINT	Признак допустимости пустого значения NULL.
RDB$PARAMETER_MECHANISM	SMALLINT	Признак — передается ли параметр по значению (значение столбца 0), по ссылке (значение 1), через дескриптор (значение 2) или через дескриптор BLOB (значение 3).
RDB$FIELD_NAME	CHAR(31)	Имя столбца, на которое ссылается параметр с помощью предложения TYPE OF COLUMN.
RDB$RELATION_NAME	CHAR(31)	Имя таблицы, на которую ссылается параметр с помощью предложения TYPE OF COLUMN.
RDB$PACKAGE_NAME	CHAR(31)	Имя пакета процедуры (если процедура упакованная), в которой используется параметр.

RDB$PROCEDURES

Описывает хранимые процедуры.

Идентификатор столбца	Тип данных	Описание
RDB$PROCEDURE_NAME	CHAR(31)	Имя хранимой процедуры.
RDB$PROCEDURE_ID	SMALLINT	Уникальный идентификатор процедуры.
RDB$PROCEDURE_INPUTS	SMALLINT	Указывает количество входных параметров или их отсутствие (значение NULL).
RDB$PROCEDURE_OUTPUTS	SMALLINT	Указывает количество выходных параметров или их отсутствие (значение NULL).
RDB$DESCRIPTION	BLOB TEXT	Произвольный текст примечания к процедуре.
RDB$PROCEDURE_SOURCE	BLOB TEXT	Исходный код процедуры на языке SQL.
RDB$PROCEDURE_BLR	BLOB BLR	Двоичное представление (BLR) кода процедуры.
RDB$SECURITY_CLASS	CHAR(31)	Может указывать на класс безопасности, определенный в системной таблице RDB$SECURITY_CLASSES, для применения ограничений управления доступом.
RDB$OWNER_NAME	CHAR(31)	Имя пользователя — владельца (создателя) процедуры.

Идентификатор столбца	Тип данных	Описание
RDB$RUNTIME	BLOB	Описание метаданных процедуры. Внутреннее использование для оптимизации.
RDB$SYSTEM_FLAG	SMALLINT	Указывает, процедура определена пользователем (значение 0) или системой (значение 1 или выше).
RDB$PROCEDURE_TYPE	SMALLINT	Тип процедуры: 1 — хранимая процедура выбора (содержит в своем составе оператор SUSPEND), 2 — выполняемая хранимая процедура.
RDB$VALID_BLR	SMALLINT	Указывает, остается ли текст хранимой процедуры корректным после последнего изменения процедуры при помощи оператора ALTER PROCEDURE.
RDB$DEBUG_INFO	BLOB	Содержит отладочную информацию о переменных, используемых в хранимой процедуре.
RDB$ENGINE_NAME	CHAR(31)	Имя движка для использования внешних процедур. Обычно UDR.
RDB$ENTRYPOINT	CHAR(255)	Имя точки входа в библиотеке, где находится эта процедура.
RDB$PACKAGE_NAME	CHAR(31)	Имя пакета, если процедура является упакованной.
RDB$PRIVATE_FLAG	SMALLINT	Для неупакованных хранимых процедур всегда NULL, для упакованных 0 — если процедура описана в заголовке пакета и 1 — если процедура описана или реализована только в теле пакета (не описана в заголовке)
RDB$SQL_SECURITY	BOOLEAN	Определяет в контексте какого пользователя будет выполняться процедура. Если установлен в FALSE, то процедура выполняется с правами вызвавшего его пользователя. Иначе процедура выполняется с правами его владельца (создателя)

RDB$REF_CONSTRAINTS

Описания именованных ограничений базы данных (внешних ключей).

Идентификатор столбца	Тип данных	Описание
RDB$CONSTRAINT_NAME	CHAR(31)	Имя ограничения внешнего ключа. Задается пользователем или автоматически генерируется системой.
RDB$CONST_NAME_UQ	CHAR(31)	Имя ограничения первичного или уникального ключа, на которое ссылается предложение REFERENCES в данном ограничении.
RDB$MATCH_OPTION	CHAR(7)	Не используется. Текущим значением является FULL во всех случаях.

Идентификатор столбца	Тип данных	Описание
RDB$UPDATE_RULE	CHAR(11)	Действия по ссылочной целостности, применимые к данному внешнему ключу, когда изменяется первичный (уникальный) ключ родительской таблицы: RESTRICT, NO ACTION, CASCADE, SET NULL, SET DEFAULT.
RDB$DELETE_RULE	CHAR(11)	Действия по ссылочной целостности, применимые к данному внешнему ключу, когда удаляется первичный (уникальный) ключ родительской таблицы: RESTRICT, NO ACTION, CASCADE, SET NULL, SET DEFAULT.

RDB$RELATION_CONSTRAINTS

Описание всех ограничений на уровне таблиц: первичного, уникального, внешнего ключей, ограничений CHECK, NOT NULL.

Идентификатор столбца	Тип данных	Описание
RDB$CONSTRAINT_NAME	CHAR(31)	Имя ограничения на уровне таблицы, заданное пользователем или автоматически присвоенное системой.
RDB$CONSTRAINT_TYPE	CHAR(11)	Содержит название ограничения: PRIMARY KEY, UNIQUE, FOREIGN KEY, CHECK, NOT NULL.
RDB$RELATION_NAME	CHAR(31)	Имя таблицы, к которой применяется это ограничение.
RDB$DEFERRABLE	CHAR(3)	В настоящий момент во всех случаях NO.
RDB$INITIALLY_DEFERRED	CHAR(3)	В настоящий момент во всех случаях NO.
RDB$INDEX_NAME	CHAR(31)	Имя индекса, который поддерживает это ограничение (содержит NULL, если ограничением является CHECK или NOT NULL).

RDB$RELATION_FIELDS

Характеристики столбцов таблиц и представлений.

Идентификатор столбца	Тип данных	Описание
RDB$FIELD_NAME	CHAR(31)	Имя столбца.
RDB$RELATION_NAME	CHAR(31)	Имя таблицы (представления), где присутствует описываемый столбец.
RDB$FIELD_SOURCE	CHAR(31)	Содержит имя домена (определенного пользователем или созданного автоматически системой), на котором основывается данный столбец.
RDB$QUERY_NAME	CHAR(31)	В настоящей версии системы не используется.
RDB$BASE_FIELD	CHAR(31)	Только для представления. Имя столбца из базовой таблицы.

Идентификатор столбца	Тип данных	Описание
RDB$EDIT_STRING	VARCHAR(127)	Не используется.
RDB$FIELD_POSITION	SMALLINT	Позиция столбца в таблице или представлении. Нумерация начинается с 0.
RDB$QUERY_HEADER	BLOB TEXT	Не используется.
RDB$UPDATE_FLAG	SMALLINT	Указывает, является ли столбец обычным столбцом (значение 1) или вычисляемым (значение 0).
RDB$FIELD_ID	SMALLINT	В настоящей версии системы в точности соответствует значению в столбце RDB$FIELD_POSITION.
RDB$VIEW_CONTEXT	SMALLINT	Для столбца представления это внутренний идентификатор базовой таблицы, откуда приходит это поле.
RDB$DESCRIPTION	BLOB TEXT	Примечание к столбцу таблицы или представления.
RDB$DEFAULT_VALUE	BLOB BLR	Записанное в двоичном виде (BLR) значение по умолчанию — предложение DEFAULT, если оно присутствует при описании столбца таблицы (представления).
RDB$SYSTEM_FLAG	SMALLINT	Указывает, определено пользователем (значение 0) или системой (значение 1 или выше).
RDB$SECURITY_CLASS	CHAR(31)	Может ссылаться на класс безопасности, определенный в RDB$SECURITY_CLASSES для применения ограничений управления доступом для всех пользователей этого столбца.
RDB$COMPLEX_NAME	CHAR(31)	Не используется.
RDB$NULL_FLAG	SMALLINT	Указывает, допускает ли столбец значения NULL (значение NULL) или не допускает (значение 1).
RDB$DEFAULT_SOURCE	BLOB TEXT	Исходный текст предложения DEFAULT, если присутствует.
RDB$COLLATION_ID	SMALLINT	Идентификатор последовательности сортировки в составе набора символов для столбца не по умолчанию.
RDB$GENERATOR_NAME	CHAR(31)	Имя внутреннего генератора для реализации identity столбца.
RDB$IDENTITY_TYPE	SMALLINT	В настоящее время для identity столбца всегда хранит значение 0 (GENERATED BY DEFAULT) и NULL для других типов столбцов. В будущем этот столбец будет иметь возможность хранить значение 1 (GENERATED ALWAYS), когда этот тип идентификационных столбцов будет поддерживаться Ред базой данных.

RDB$RELATIONS

Хранит некоторые характеристики таблиц и представлений.

Идентификатор столбца	Тип данных	Описание
RDB$VIEW_BLR	BLOB BLR	Для представления содержит на языке BLR спецификации запроса. Для таблицы в поле содержится NULL.
RDB$VIEW_SOURCE	BLOB TEXT	Для представления содержит оригинальный исходный текст запроса на языке SQL (включая пользовательские комментарии). Для таблицы в поле содержится NULL.
RDB$DESCRIPTION	BLOB TEXT	Произвольный текст примечания к таблице (представлению).
RDB$RELATION_ID	SMALLINT	Внутренний идентификатор таблицы (представления).
RDB$SYSTEM_FLAG	SMALLINT	Указывает, создана ли таблица (представление) пользователем (значение 0) или системой (значение 1 или выше).
RDB$DBKEY_LENGTH	SMALLINT	Общая длина ключа. Для таблицы это 8 байтов. Для представления это 8, умноженное на количество таблиц, на которые ссылается представление.
RDB$FORMAT	SMALLINT	Внутреннее использование.
RDB$FIELD_ID	SMALLINT	Количество столбцов в таблице (представлении).
RDB$RELATION_NAME	CHAR(31)	Имя таблицы или представления.
RDB$SECURITY_CLASS	CHAR(31)	Может ссылаться на класс безопасности, определенный в таблице RDB$SECURITY_CLASSES для применения ограничений управления доступом для всех пользователей этой таблицы (представления).
RDB$EXTERNAL_FILE	VARCHAR(255)	Полный путь к внешнему файлу данных, если таблица описана с предложением EXTERNAL FILE.
RDB$RUNTIME	BLOB	Описание метаданных таблицы. Внутреннее использование для оптимизации.
RDB$EXTERNAL_DESCRIPTION	BLOB	Произвольное примечание к внешнему файлу таблицы.
RDB$OWNER_NAME	CHAR(31)	Имя пользователя — владельца (создателя) таблицы или представления.
RDB$DEFAULT_CLASS	CHAR(31)	Класс безопасности по умолчанию. Применяется, когда новый столбец добавляется в таблицу.
RDB$FLAGS	SMALLINT	Внутренние флаги.

Идентификатор столбца	Тип данных	Описание
RDB$RELATION_TYPE	SMALLINT	Тип описываемого объекта: 0 – системная таблица или таблица, созданная пользователем, 1 – представление, 2 – внешняя таблица, 3 – виртуальная таблица (таблица мониторинга MON$, псевдотаблицы безопасности SEC$), 4 – GTT уровня соединения (PRESERVE ROWS), 5 – GTT уровня транзакции (DELETE ROWS).
RDB$SQL_SECURITY	BOOLEAN	Определяет в контексте какого пользователя будет использоваться таблица. Если установлен в FALSE, то таблица будет использоваться с правами вызвавшего его пользователя. Иначе таблица будет использоваться с правами его владельца (создателя)
RDB$ADAPTER	CHAR(31)	Адаптер для внешней таблицы для чтения двоичных трейсов

RDB$ROLES

Определение ролей.

Идентификатор столбца	Тип данных	Описание
RDB$ROLE_NAME	CHAR(31)	Имя роли.
RDB$OWNER_NAME	CHAR(31)	Имя пользователя-владельца роли.
RDB$DESCRIPTION	BLOB TEXT	Произвольный текст примечания к роли.
RDB$SYSTEM_FLAG	SMALLINT	Системный флаг.
RDB$SECURITY_CLASS	CHAR(31)	Может ссылаться на класс безопасности, определённый в таблице RDB$SECURITY_CLASSES для применения ограничений управления доступом для всех пользователей этой роли

RDB$SECURITY_CLASSES

Списки управления доступом.

Идентификатор столбца	Тип данных	Описание
RDB$SECURITY_CLASS	CHAR(31)	Имя класса безопасности.
RDB$ACL	BLOB ACL	Список управления доступом, связанный с классом безопасности. Перечисляет пользователей и их полномочия.
RDB$DESCRIPTION	BLOB TEXT	Произвольный текст примечания к классу безопасности.

RDB$TRANSACTIONS

Состояние транзакций при обращении к нескольким базам данных.

Идентификатор столбца	Тип данных	Описание
RDB$TRANSACTION_ID	INTEGER	Уникальный идентификатор отслеживаемой транзакции.
RDB$TRANSACTION_STATE	SMALLINT	Состояние транзакции: зависшая (значение 0), подтвержденная (значение 1), отмененная (значение 2).
RDB$TIMESTAMP	TIMESTAMP	Не используется.
RDB$TRANSACTION_DESCRIPTION	BLOB	Описывает подготовленную транзакцию к нескольким базам данных. Используется в случае потери соединения, которое не может быть восстановлено.

RDB$TRIGGER_MESSAGES

Сообщения триггеров.

Идентификатор столбца	Тип данных	Описание
RDB$TRIGGER_NAME	CHAR(31)	Имя триггера, с которым связано данное сообщение.
RDB$MESSAGE_NUMBER	SMALLINT	Номер сообщения в пределах одного триггера (от 1 до максимум 32,767).
RDB$MESSAGE	VARCHAR(1023)	Текст сообщения триггера.

RDB$TRIGGERS

Описания триггеров.

Идентификатор столбца	Тип данных	Описание
RDB$TRIGGER_NAME	CHAR(31)	Имя триггера.
RDB$RELATION_NAME	CHAR(31)	Имя таблицы или представления, для которого используется триггер. Если триггер применяется не к событию таблицы, а к событию базы данных, то в этом поле находится NULL.
RDB$TRIGGER_SEQUENCE	SMALLINT	Последовательность (позиция) триггера. Ноль обычно означает, что последовательность не задана.

Идентификатор столбца	Тип данных	Описание
RDB$TRIGGER_TYPE	BIGINT	Событие, на которое вызывается триггер. Значение высчитывается как битовая маска. Наиболее распростаненные значения: 1 — before insert, 2 — after insert, 3 — before update, 4 — after update, 5 — before delete, 6 — after delete, 17 — before insert or update, 18 — after insert or update, 25 — before insert or delete, 26 — after insert or delete, 27 — before update or delete, 28 — after update or delete, 113 — before insert or update or delete, 114 — after insert or update or delete, 8192 — on connect, 8193 — on disconnect, 8194 — on transaction start, 8195 — on transaction commit, 8196 — on transaction rollback. Для DDL триггеров тип триггера получается путём побитового ИЛИ над фазой события (0 - BEFORE, 1 - AFTER) и всех перечисленных типов событий: CREATE TABLE - 0x0000000000004002; ALTER TABLE - 0x0000000000004004; DROP TABLE - 0x0000000000004008; CREATE PROCEDURE - 0x0000000000004010; ALTER PROCEDURE - 0x0000000000004020; DROP PROCEDURE - 0x0000000000004040; CREATE FUNCTION - 0x0000000000004080; ALTER FUNCTION - 0x0000000000004100; DROP FUNCTION - 0x0000000000004200;

Идентификатор столбца	Тип данных	Описание
		CREATE TRIGGER - 0x0000000000004400; ALTER TRIGGER - 0x0000000000004800; DROP TRIGGER - 0x0000000000005000; CREATE EXCEPTION - 0x0000000000014000; ALTER EXCEPTION - 0x0000000000024000; DROP EXCEPTION - 0x0000000000044000; CREATE VIEW - 0x0000000000084000; ALTER VIEW - 0x0000000000104000; DROP VIEW - 0x0000000000204000; CREATE DOMAIN - 0x0000000000404000; ALTER DOMAIN - 0x0000000000804000; DROP DOMAIN - 0x0000000001004000; CREATE ROLE - 0x0000000002004000; ALTER ROLE - 0x0000000004004000; DROP ROLE - 0x0000000008004000; CREATE INDEX - 0x0000000010004000; ALTER INDEX - 0x0000000020004000; DROP INDEX - 0x0000000040004000; CREATE SEQUENCE - 0x0000000080004000; ALTER SEQUENCE - 0x0000000100004000; DROP SEQUENCE - 0x0000000200004000; CREATE USER - 0x0000000400004000; ALTER USER - 0x0000000800004000; DROP USER - 0x0000001000004000; CREATE COLLATION - 0x0000002000004000; DROP COLLATION - 0x0000004000004000; ALTER CHARACTER SET - 0x0000008000004000; CREATE PACKAGE - 0x0000010000004000; ALTER PACKAGE - 0x0000020000004000; DROP PACKAGE - 0x0000040000004000; CREATE PACKAGE BODY - 0x0000080000004000; DROP PACKAGE BODY - 0x0000100000004000; CREATE MAPPING - 0x0000200000004000; ALTER MAPPING - 0x0000400000004000; DROP MAPPING - 0x0000800000004000; ANY DDL STATEMENT - 0x7FFFFFFFFFFFDFFE.
RDB$TRIGGER_SOURCE	BLOB TEXT	Хранит исходный код триггера в PSQL.
RDB$TRIGGER_BLR	BLOB BLR	Хранит триггер в двоичном коде BLR.
RDB$DESCRIPTION	BLOB TEXT	Текст примечания триггера.
RDB$TRIGGER_INACTIVE	SMALLINT	Указывает, является ли триггер в настоящее время неактивным (1) или активным (0).
RDB$SYSTEM_FLAG	SMALLINT	Признак — триггер определен пользователем (0) или системой (1 или выше).
RDB$FLAGS	SMALLINT	Внутреннее использование.
RDB$VALID_BLR	SMALLINT	Указывает, остается ли текст триггера корректным после последнего изменения триггера при помощи оператора ALTER TRIGGER.

Идентификатор столбца	Тип данных	Описание
RDB$DEBUG_INFO	BLOB	Содержит отладочную информацию о переменных, используемых в триггере.
RDB$ENGINE_NAME	CHAR(31)	Имя движка для использования внешних триггеров. Обычно UDR.
RDB$ENTRYPOINT	CHAR(255)	Имя точки входа в библиотеке, где находится этот триггер.
RDB$SQL_SECURITY	BOOLEAN	Определяет в контексте какого пользователя будет выполняться триггер. Если установлен в FALSE, то триггер выполняется с правами вызвавшего его пользователя. Иначе триггер выполняется с правами его владельца (создателя)

RDB$TYPES

Описание перечислимых типов данных.

Идентификатор столбца	Тип данных	Описание
RDB$FIELD_NAME	CHAR(31)	Имя столбца, для которого определен данный перечислимый тип.
RDB$TYPE	SMALLINT	Задает идентификатор для типа. Последовательность чисел является уникальной для каждого отдельного перечислимого типа: 0 — таблица, 1 — представление, 2 — триггер, 3 — вычисляемый столбец, 4 — проверка, 5 — процедура.
RDB$TYPE_NAME	CHAR(31)	Текстовое представление для перечислимого типа.
RDB$DESCRIPTION	BLOB TEXT	Произвольный текст примечания к перечислимому типу.
RDB$SYSTEM_FLAG	SMALLINT	Признак: определен пользователем (значение 0) или системой (значение 1 или выше).

RDB$USER_PRIVILEGES

Полномочия пользователей системы.

Идентификатор столбца	Тип данных	Описание
RDB$USER	CHAR(31)	Объект, которому предоставляется данное полномочие.
RDB$GRANTOR	CHAR(31)	Имя пользователя, предоставляющего полномочие.

Идентификатор столбца	Тип данных	Описание
RDB$PRIVILEGE	CHAR(6)	Привилегия, предоставляемая в полномочии: A - all (все привилегии), S - select (выборка данных), I - insert (добавление данных), D - delete (удаление строк), R - reference (внешний ключ), U - update (изменение данных), X - execute (выполнение), G - usage (использование), C - create (создание), L - alter (изменение), O - drop (удаление), M - membership (членство).
RDB$GRANT_OPTION	SMALLINT	Содержит ли полномочие авторизацию WITH GRANT OPTION: 1 — содержит, 0 — не содержит.
RDB$RELATION_NAME	CHAR(31)	Имя объекта (таблица или роль), которому предоставляется полномочие.
RDB$FIELD_NAME	CHAR(31)	Имя столбца, к которому применяется привилегия на уровне столбца (только привилегии UPDATE и REFERENCES).
RDB$USER_TYPE	SMALLINT	Идентифицирует тип пользователя, которому предоставляется привилегия: 1 – представление; 2 – триггер; 5 – процедура; 8 – пользователь; 13 – роль; 15 – функция; 18 – пакет.

Идентификатор столбца	Тип данных	Описание
RDB$OBJECT_TYPE	SMALLINT	Идентифицирует тип объекта, к которому предоставляется привилегия: 0 – таблица, представление; 2 – триггер; 3 - вычисляемый столбец; 4 - проверочное ограничение; 5 – процедура; 6 - выражение, используемое для индекса; 7 – исключение; 8 – пользователь; 9 - столбец; 10 - индекс; 11 – набор символов; 12 - группа пользователя; 13 – роль; 14 – генератор (последовательность); 15 – функция; 16 – BLOB фильтр; 17 – порядок сортировки; 18 - заголовок пакета; 19 - тело пакета. Идентифицирует тип объекта, к которому предоставляется DDL-привилегия: 20 - база данных; 21 - таблица; 22 - представление; 23 - процедура; 24 - фунцкия; 25 - пакет; 26 - генератор; 27 - домен; 28 - исключение; 29 - роль; 30 - набор символов; 31 - порядок сортировки; 32 - фильтр.

RDB$VIEW_RELATIONS

Описывает представления. Не используется в настоящей версии.

Идентификатор столбца	Тип данных	Описание
RDB$VIEW_NAME	CHAR(31)	Имя представления.
RDB$RELATION_NAME	CHAR(31)	Имя таблицы, на которое ссылается данное представление.
RDB$VIEW_CONTEXT	SMALLINT	Псевдоним, используемый для ссылки на столбец представления. Имеет то же значение, что и псевдоним, используемый в самом тексте представления на языке BLR в операторе запроса этого представления.

Идентификатор столбца	Тип данных	Описание
RDB$CONTEXT_NAME	CHAR(255)	Текстовый вариант псевдонима, указанного в столбце RDB$VIEW_CONTEXT.
RDB$CONTEXT_TYPE	SMALLINT	Тип контекста: 0 – таблица; 1 – представление; 2 – хранимая процедура
RDB$PACKAGE_NAME	CHAR(31)	Имя пакета для упакованной хранимой процедуры.

Приложение B Псевдотаблицы безопасности

Псевдотаблицы безопасности имеют префикс имени SEC$. Они отображают данные из текущей базы данных безопасности. Эти таблицы являются виртуальными в том смысле, что до обращения к ним со стороны пользователя, никаких данных в них не записано. Они заполняются данными только в момент запроса пользователя. При этом описания этих таблиц в базе данных присутствуют постоянно. В этом смысле эти псевдотаблицы подобны таблицам семейства MON$, используемых для мониторинга сервера.

Полный доступ ко всей информации, предоставляемой таблицами безопасности, имеют SYSDBA и владелец базы данных. Обычные пользователи ограничены информацией о самих себе, другие пользователи невидимы для них.

Список псевдотаблиц представлен в таблице B.1.

Таблица B.1 — Список псевдотаблиц «Ред База Данных»

Таблица	Содержание
SEC$GLOBAL_AUTH_MAPPING	Сведения о глобальных отображениях
SEC$USERS	Список пользователей в текущей базе данных безопасности
SEC$USER_ATTRIBUTES	Сведения о дополнительных атрибутах пользователей

SEC$GLOBAL_AUTH_MAPPING

Таблица хранит сведения о глобальных отображениях.

Идентификатор столбца	Тип данных	Описание
SEC$MAP_NAME	CHAR(31)	Имя глобального отображения.
SEC$MAP_USING	CHAR(1)	Является ли аутентификация общесерверной (S) или обычной (P).
SEC$MAP_PLUGIN	CHAR(31)	Имя плагина аутентификации, из которого происходит отображение.
SEC$MAP_DB	CHAR(31)	Имя базы данных, в которой прошла аутентификация. Из неё происходит отображение.
SEC$MAP_FROM_TYPE	CHAR(31)	Тип объекта, который будет отображён.
SEC$MAP_FROM	CHAR(255)	Имя объекта, из которого будет произведено отображение.
SEC$MAP_TO_TYPE	SMALLINT	Тип объекта, в который будет произведено отображение: 0 — USER; 1 — ROLE.
SEC$MAP_TO	CHAR(31)	Наименование объекта, в который будет произведено отображение (имя пользователя или роли).
RDB$SYSTEM_FLAG	SMALLINT	Является ли отображение системным: 0 — определено пользователем; 1 — определено системой.
RDB$DESCRIPTION	BLOB TEXT	Произвольное текстовое описание порядка сортировки.

SEC$USERS

Таблица хранит список пользователей в текущей базе данных безопасности.

Идентификатор столбца	Тип данных	Описание
SEC$USER_NAME	CHAR(31)	Имя пользователя.
SEC$FIRST_NAME	VARCHAR(32)	Имя.
SEC$MIDDLE_NAME	VARCHAR(32)	Отчество.
SEC$LAST_NAME	VARCHAR(32)	Фамилия.
SEC$ACTIVE	BOOLEAN	Флаг активности пользователя.
SEC$ADMIN	BOOLEAN	Отражает, имеет ли пользователь права RDB$ADMIN в базе данных безопасности.
SEC$DESCRIPTION	BLOB TEXT	Комментарий к пользователю.
SEC$PLUGIN	CHAR(31)	Имя плагина управления пользователями, с помощью которого был создан данный пользователь.

SEC$USER_ATTRIBUTES

Таблица хранит сведения о дополнительных атрибутах пользователей.

Идентификатор столбца	Тип данных	Описание
SEC$USER_NAME	CHAR(31)	Имя пользователя.
SEC$KEY	VARCHAR(31)	Имя атрибута.
SEC$VALUE	VARCHAR(255)	Значение атрибута.
SEC$PLUGIN	CHAR(31)	Имя плагина управления пользователями, с помощью которого был создан данный пользователь.

Приложение Г Схема LDAP

OpenLDAP

```
attributetype ( 1.2.643.2.63.1.1.1
        NAME 'rdbPassword'
        DESC 'Native RDB password'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.2
        NAME 'rdbSecurePassword'
        DESC 'Secure RDB password'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.3
        NAME 'rdbPasswordAlgorithm'
        DESC 'RDB password hashing algorithm'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.4
        NAME 'rdbPasswordHistory'
        DESC 'RDB password change history'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.5
        NAME 'rdbPolicy'
        DESC 'User policy in security database'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.6
        NAME 'rdbPasswordTime'
        DESC 'Date/time of last password change'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.7
        NAME 'rdbFailedCount'
        DESC 'Count of failed authentication events'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.8
        NAME 'rdbAccessTime'
        DESC 'Date/time when ends user ban'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
```

```
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.9
        NAME 'rdbMlsLevel'
        DESC 'Security level'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.10
        NAME 'rdbMlsCompartment'
        DESC 'User compartment'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.11
        NAME 'rdbSrpVerifier'
        DESC 'RDB password for SRP protocol'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.12
        NAME 'rdbSrpSalt'
        DESC 'Salt for SRP'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.13
        NAME 'rdbActive'
        DESC 'Blocking flag'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
        SINGLE-VALUE )

objectclass ( 1.2.643.2.63.2.1
        NAME 'rdbAuth'
        SUP top
        AUXILIARY
        DESC 'RDB authentication data'
        MAY ( rdbPassword $ rdbSecurePassword $
                rdbPasswordAlgorithm $ rdbPasswordHistory $
                rdbPolicy $ rdbPasswordTime $ rdbFailedCount $ rdbAccessTime $
                rdbMlsLevel $ rdbMlsCompartment $
                rdbSrpVerifier $ rdbSrpSalt $ rdbActive
        )
)
```

Приложение Д База данных безопасности

Информация обо всех пользователях баз данных СУБД «Ред База Данных» хранится в общей базе данных безопасности, которая расположена в корневой директории каталога установки сервера и называется `security3.fdb`. В этой базе хранятся параметры пользователей системы, созданных с помощью различных плагинов управления пользователями, политики безопасности, хэши старых паролей. Эти данные располагаются в системных таблицах, которые представлены в таблице Д.1.

Таблица Д.1 — Список системных таблиц `security3.fdb`

Таблица	Содержание
`PLG$USERS`	Хранит параметры пользователей системы, созданных с помощью плагина управления пользователями `Legacy_UserManager`
`PLG$SRP`	Хранит параметры пользователей системы, созданных с помощью плагина управления пользователями `Srp`
`PLG$MF`	Хранит параметры пользователей системы, созданных с помощью плагина управления пользователями `Multifactor_Manager`
`PLG$POLICIES`	Хранит политики учетных записей пользователей
`PLG$PASSWD_HISTORY`	Хранит хэши старых паролей

PLG$USERS

Хранит параметры пользователей системы, созданных с помощью плагина управления пользователями `Legacy_UserManager`.

Идентификатор столбца	Тип данных	Описание
`PLG$USER_NAME`	`VARCHAR(31)`	Уникальное имя пользователя (логин).
`PLG$GROUP_NAME`	`VARCHAR(31)`	Название группы.
`PLG$UID`	`INTEGER`	ID пользователя.
`PLG$GID`	`INTEGER`	ID группы.
`PLG$PASSWD`	`VARCHAR(64)`	Пароль.
`PLG$COMMENT`	`BLOB SUB_TYPE TEXT SEGMENT 80`	Комментарий.
`PLG$FIRST_NAME`	`VARCHAR(32)`	Имя пользователя.
`PLG$MIDDLE_NAME`	`VARCHAR(32)`	Отчество пользователя.
`PLG$LAST_NAME`	`VARCHAR(32)`	Фамилия пользователя.

PLG$SRP

Идентификатор столбца	Тип данных	Описание
`PLG$USER_NAME`	`VARCHAR(31)`	Уникальное имя пользователя (логин).
`PLG$VERIFIER`	`VARCHAR(128)`	Зашифрованный пароль.

Идентификатор столбца	Тип данных	Описание
PLG$SALT	VARCHAR(32)	Некоторая случайная последовательность для формирования VERIFIER
PLG$COMMENT	BLOB SUB_TYPE TEXT SEGMENT 80	Комментарий.
PLG$FIRST	VARCHAR(32)	Имя пользователя.
PLG$MIDDLE	VARCHAR(32)	Отчество пользователя.
PLG$LAST	VARCHAR(32)	Фамилия пользователя.
PLG$ATTRIBUTES	BLOB SUB_TYPE TEXT SEGMENT 80	Пользовательские атрибуты
PLG$ACTIVE	BOOLEAN	Состояние пользователя: активное или неактивное

PLG$MF

Идентификатор столбца	Тип данных	Описание
PLG$USER_NAME	VARCHAR(31)	Уникальное имя пользователя (логин).
PLG$HASH_ALG	VARCHAR(32)	Название алгоритма хэширования. Этот алгоритм будет использован для проверки пароля. Если поле не указано, то используется старый алгоритм хэширования.
PLG$MF_PASSWD	VARCHAR(64)	хэш пароля многофакторногопользователя.
PLG$COMMENT	BLOB SUB_TYPE TEXT SEGMENT 80	Комментарий.
PLG$FIRST	VARCHAR(32)	Имя пользователя.
PLG$MIDDLE	VARCHAR(32)	Отчество пользователя.
PLG$LAST	VARCHAR(32)	Фамилия пользователя.
PLG$POLICY_NAME	VARCHAR(32)	Ссылается на имя политики безопасности, определенное в системной таблице PLG$POLICIES
PLG$PASSWD_TIME	TIMESTAMP	Дата установки пароля.
PLG$FAILED_COUNT	INTEGER	Число неудачных попыток авторизации.
PLG$ACCESS_TIME	TIMESTAMP	Время, по истечении которого пользователь может получить доступ к БД
PLG$LAST_ONLINE	TIMESTAMP	Времени неактивности учетных записей пользователя.
PLG$ATTRIBUTES	BLOB SUB_TYPE TEXT SEGMENT 80	Пользовательские атрибуты.
PLG$ACTIVE	BOOLEAN	Состояние пользователя: активное или неактивное.

PLG$POLICIES

Используется для хранения политик учетных записей пользователей.

Идентификатор столбца	Тип данных	Описание
PLG$POLICY_NAME	VARCHAR(32)	Имя политики безопасности. Политика по умолчанию имеет значение поля равное DEFAULT.
PLG$PSWD_NEED_CHAR	INTEGER	Минимально допустимое число буквенных символов в пароле.
PLG$PSWD_NEED_DIGIT	INTEGER	Минимально допустимое число цифровых символов в пароле
PLG$PSWD_NEED_DIFF_CASE	INTEGER	Необходимость наличия в пароле буквенных символов в различных регистрах
PLG$PSWD_MIN_LEN	INTEGER	Минимально допустимая длина пароля.
PLG$PSWD_VALID_DAYS	INTEGER	Срок действия пароля в днях.
PLG$PSWD_UNIQUE_COUNT	INTEGER	Количество не повторяющихся подряд паролей.
PLG$MAX_FAILED_COUNT	INTEGER	Максимальное число ошибок при прохождении аутентификации.
PLG$AUTH_FACTORS	VARCHAR(64)	Факторы аутентификации: PASSWORD, CERT_X509. Задаются условным выражением из символических обозначений (P и C соответственно). Выражение является объединением возможных наборов необходимых факторов аутентификации и позволяет задавать различные варианты требуемых факторов с помощью операций «И», «ИЛИ». Например, (P) | (PC) | (C).
PLG$MAX_UNUSED_DAYS	INTEGER	Максимальное время неактивности учетных записей пользователя в днях.

PLG$PASSWD_HISTORY

Хранит хэши старых паролей.

Идентификатор столбца	Тип данных	Описание
PLG$KEY_ID	INTEGER	Суррогатный первичный ключ
PLG$USER_NAME	VARCHAR(31)	Ссылается на логин пользователя, определенный в системной таблице RDB$USERS.
PLG$MF_PASSWD	VARCHAR(64)	Хэш пароля многофакторногопользователя.
PLG$HASH_ALG	VARCHAR(32)	Название алгоритма хэширования. Этот алгоритм будет использован для проверки пароля. Если поле не указано, то используется старый алгоритм хэширования.

Приложение E Тестирование системы безопасности

Утилита тестирования

Утилита `fbtest-portable`, входящая в состав дистрибутива Ред Базы Данных 3.0, предназначена для тестирования средств защиты информации. Она проверяет все требования, которым должна удовлетворять система соответствующая 5 классу НСД. Она представляет собой совокупность тестов, моделирующих попытки несанкционированного доступа к защищенным объектам.

Утилита `fbtest-portable` – это инструмент командной строки, написанный на Python.

Распакуйте архив с утилитой и запустите ее. Для запуска введите в консоли имя исполняемого файла и основные параметры.

```
fbtest-portable [-h] [-b BIN_DIR] [-d DB_DIR] [-archive] [-rerun] [-untested] [-v]
[-verbosity {0,1,2}] [-q] [-x] [-e FILENAME] [-u] [-w PASSWORD] [-o HOST] [-p
PERSON] -a ARCH [-s SEQUENCE] [-k SKIP [SKIP...]] [-url URL] [-c CLIENT] [names
[names...]]
```

Список всех параметров утилиты представлен в таблице:

Таблица E.1 — Параметры утилиты `fbtest-portable`

Аргумент	Описание
`-h`, `--help`	Показать все опции утилиты
`-b <bin_каталог>`, `--bin-dir <bin_каталог>`	Указать путь к каталогу с бинарными файлами сервера. Рекомендуем указать.
`-d <db_каталог>`, `--db-dir <db_каталог>`	Директория для хранения тестовых баз данных
`--archive`	Сохранить результаты запуска в файл с расширением `.trf` по пути `fbt-repository/archive`
`--rerun`	Выполнить только те тесты, которые не пройдены при последнем запуске
`--untested`	Выполнить только те тесты, которые имели статус `UNTESTED` при последнем запуске
`-v`, `--verbose`	Подробный вывод
`--verbosity {0|1|2}`	Уровень детальности вывода. `--verbosity 2` соответствует параметру `-v`. По умолчанию значение 1.
`-q`, `--quiet`	Краткий вывод
`-x`, `--xunit`	Сохраняет результаты тестов в стандартном формате XUnit XML
`-e <имя файла>`, `--expect <имя файла>`	Указать файл с расширением `.trf` с результатами тестов для использования в качестве ожидаемых результатов.

Аргумент	Описание
-u, --update	Обновить результаты последнего запуска с результатами повторного запуска
-w <пароль>, --password <пароль>	Пароль SYSDBA. По умолчанию masterkey
-o <хост>, --host <хост>	Имя хоста машины с сервером Ред Базы Данных. По умолчанию localhost
-p <имя тестировщика>, --person <имя тестировщика>	Для отчетности задать имя тестировщика
-a {SS, CS, SC}, --arch {SS, CS, SC}	Архитектура сервера Ред Базы Данных: SS - Super , CS - Classic, SC - SuperClassic (см. firebird.conf, параметр ServerMode). Обязательный параметр
-s <номер>, --sequence <номер>	Указать порядковый номер запуска тестов. Он запишется в архив results.trf
-k <пропуск> [<пропуск> ...], --skip <пропуск> [<пропуск> ...]	Указать имя теста или набор тестов, или имя файла с именами тестов, которые нужно пропустить
--url <URL>	URL адрес к репозиторию с тестами. Если этот параметр установлен, отчет будет содержать ссылку на тестовый файл.
-c <клиентская библиотека>, --client <клиентская библиотека>	Использовать указанную клиентскую библиотеку Ред Базы Данных.

Здесь <names> – это название теста. Все тесты утилиты лежат в директории /fbt-repository/tests/stest. В качестве <names> указывается путь от папки stest до самого файла с тестом, где в качестве разделителя используется точка. Например: stest.Connect_Testing.correctloginpass. Если параметр <names> не указан, то автоматически будут прогоняться все тесты из папки tests.

Пример запуска отдельного теста на Linux:

```
./fbtest-portable -a SS -v -x stest.DDL_Testing.altrdmnaftergrant
```

Пример запуска всех тестов на Windows:

```
fbtest-portable.exe -b "C:\RedDatabase 3.0" -a SC -v -x
```

В процессе тестирования напротив каждого теста выводится результат его выполнения – OK или FAIL (если включен подробный вывод -v).

Для прохождения некоторых тестов необходимо:

- **установить и настроить криптопровайдер КриптоПро CSP 4.0;**
- **на ОС Windows установить переменную окружения FIREBIRD, содержащую путь до каталога установки Ред Базы Данных:**

```
set FIREBIRD=c:\Program Files\RedDatabase
```

Во избежание потерь данных не рекомендуется запускать тесты на рабочей Ред Базе Данных , т.к. в тестах модифицируется база данных безопасности и подменяются файлы конфигурации.

Описание тестов

Все тесты логически разделены и хранятся в разных подпапках папки `stest` (см. таблицу E.2).

Таблица E.2

Директория	Назначение
`Audit_Testing`	Тестирование аудита
`Blob_Testing`	Тестирование доступа к BLOB-полям
`Connect_Testing`	Тестирование разных методов аутентификации
`DDL_Testing`	Тестирование DDL-операций
`DML_Testing`	Тестирование DML-операций
`Policy_Testing`	Тестирование политик безопасности
`Roles_Testing`	Тестирование операций с ролями
`Service_Testing`	Тестирование доступа к сервисам
`WipeMemory_Testing`	Тестирование обезличивания освобождаемой сервером оперативной памяти и дискового пространства.

В таблице E.3 приведены все тесты и их уникальные имена, используемые для запуска конкретного теста.

Таблица E.3 — Описание тестов утилиты `fbtest-portable`

Audit_Testing	
Уникальное имя	**Описание теста**
`attach_database`	Логирование события успешной аутентификации
`attach_database_failed`	Логирование события подключения к несуществующей БД
`attach_database_unauth`	Логирование события неуспешной аутентификации
`attach_service`	Логирование события присоединения к сервису
`attach_service_unauth`	Логирование события неуспешного присоединения к сервису
`commit_transaction`	Логирование события подтверждения транзакции
`connection_id`	Тестирование работы параметра `connection_id`
`create_database`	Логирование события успешного создания БД
`create_database_failed`	Логирование события неуспешного создания БД
`create_database_unauth`	Логирование события создания БД без необходимых привилегий
`dbname_full`	Логирование событий для конкретной БД
`dbname_regexpr`	Логирование событий для БД, соответствующих регулярному выражению
`detach_database`	Логирование события отключения от БД
`detach_service`	Логирование события отсоединения от сервиса
`disabled`	Проверка параметра `enabled` на значение `false`
`drop_database`	Логирование события удаления БД

enabled_invalid	Проверка параметра enabled на недопустимое значение
enabled_true	Проверка параметра enabled на значение true
exclude_and_include_same_gdscode	Тест на исключение из лога и включение в лог одинаковых GDS-кодов ошибок. В лог такие ошибки включены не будут.
exclude_filter_boolean	Проверка параметра exclude_filter на булево значение
exclude_filter_empty	Проверка параметра exclude_filter на пустое значение
exclude_filter_select	Тест на исключение из лога запросов на выборку (SELECT)
exclude_filter_select_join	Тест на исключение из лога запросов на выборку из соединенных таблиц (SELECT%JOIN)
exclude_filter_system_data	Тест на исключение из лога системных данных(RDB$, MON$)
exclude_filter_table	Тест на исключение из лога запросов связанных с определенной таблицей
exclude_gds_codes	Проверка параметра exclude_filter
exclude_process_filter	Проверка параметра exclude_process_filter
exclude_user_filter	Проверка параметра exclude_user_filter
execute_ddl_statement	Логирование события выполнения запроса на создание таблицы
execute_delete_statement	Логирование события выполнения запроса на удаления из таблицы
execute_function_finish	Логирование события завершения выполнения хранимой функции
execute_function_finish_failed	Логирование события завершения неуспешного выполнения хранимой функции
execute_function_start	Логирование события старта выполнения хранимой функции
execute_insert_statement	Логирование события выполнения запроса на вставку
execute_procedure_finish	Логирование события завершения выполнения хранимой процедуры
execute_procedure_finish_failed	Логирование события завершения неуспешного выполнения хранимой процедуры
execute_procedure_start	Логирование события старта выполнения хранимой процедуры
execute_select_statement	Логирование события выполнения запроса на выборку
execute_statement_finish_failed	Логирование события завершения неуспешного выполнения запроса
execute_statement_finish_unauth	Логирование события завершения выполнения запроса без соответствующих прав

execute_trigger_finish	Логирование события завершения выполнения триггера
execute_trigger_finish_failed	Логирование события завершения неуспешного выполнения триггера
execute_trigger_start	Логирование события начала выполнения триггера
execute_update_statement	Логирование события выполнения запроса на обновление
explain_plan	Проверка параметра `explain_plan`
format_0	Проверка параметра `format` на значение 0
format_3	Проверка параметра `format` на значение 3
format_invalid	Проверка параметра `format` на некорректное значение
free_ddl_statement	Логирование события освобождения запроса на создание таблицы
free_delete_statement	Логирование события освобождения запроса на удаление
free_insert_statement	Логирование события освобождения запроса на вставку
free_select_statement	Логирование события освобождения запроса на выборку
free_update_statement	Логирование события освобождения запроса на обновление
grant_privileges	Логирование события назначения привилегий
grant_privileges_failed	Логирование события неуспешного назначения привилегий
grant_privileges_unauth	Логирование события назначения привилегий от имени пользователя без необходимых привилегий
include+exclude_filter	Тестирование параметров `include_filter` и `exclude_filter`
include=exclude_filter	Тестирование логирования при совпадающих значениях параметров `include_filter` и `exclude_filter`
include_filter_boolean	Тестирование параметров `include_filter` на булево значение
include_filter_select_join	Тест на включение в лог только запросов на выборку из соединенных таблиц (`SELECT%JOIN`)
include_gds_codes	Тестирование параметра `include_gds_codes`
include_only_ddl	Тест на включение в лог только DDL-запросов
include_only_insert	Тест на включение в лог только запросов на вставку
include_process_filter	Тестирование параметра `include_process_filter`
include_user_filter	Тестирование параметра `include_user_filter`
log_connections_false	Тестирование параметра `log_connections` на значение `false`
log_connections_invalid	Тестирование параметра `log_connections` на некорректное значение

log_context_false	Проверка параметра log_context на значение false
log_context_true	Логирование событие установки значения контекстной переменной
log_filename_1	Тестирование работы аудита, если значение параметра log_filename задано как путь к файлу журнала
log_filename_2	Тестирование работы аудита, если значение параметра log_filename не задано
log_filename_3	Тестирование работы аудита, если значение параметра log_filename использует Sed синтаксис
log_filename_4	Тестирование работы аудита, если значение параметра log_filename использует Sed синтаксис
log_initfini	Тестирование параметра log_initfini
log_procedure_finish_false	Проверка параметра log_procedure_finish на значение false
log_procedure_finish_invalid	Проверка параметра log_procedure_finish на некорректное значение
log_procedure_start_false	Проверка параметра log_procedure_start на значение false
log_procedure_start_invalid	Проверка параметра log_procedure_start на некорректное значение
log_security_incidents	Логирование событий, связанных с нарушением безопасности сервера (инциденты безопасности)
log_service_query_false	Проверка параметра log_service_query на значение false
log_service_query_invalid	Проверка параметра log_service_query на некорректное значение
log_services_false	Проверка параметра log_services на значение false
log_services_invalid	Проверка параметра log_services на некорректное значение
log_statement_finish_false	Проверка параметра log_statement_finish на значение false
log_statement_finish_invalid	Проверка параметра log_statement_finish на некорректное значение
log_statement_free_false	Проверка параметра log_statement_free на значение false
log_statement_free_invalid	Проверка параметра log_statement_free на некорректное значение
log_statement_prepare_false	Проверка параметра log_statement_prepare на значение false
log_statement_prepare_invalid	Проверка параметра log_statement_prepare на некорректное значение
log_statement_start_false	Проверка параметра log_statement_start на значение false
log_statement_start_invalid	Проверка параметра log_statement_start на некорректное значение

log_sweep	Логирование события процесса сборки мусора
log_transactions+deadlock	Логирование события начала и завершения транзакций при deadlock
log_transactions_false	Проверка параметра log_transactions на значение false
log_transactions_invalid	Проверка параметра log_transactions на некорректное значение
log_transactions_true	Логирование события начала и завершения транзакций
max_arg_count=-1	Тестирование параметра max_arg_count на значение меньше нуля
max_arg_count=0	Тестирование параметра max_arg_count на значение ноль
max_arg_count=2	Тестирование параметра max_arg_count на значение 2
max_arg_count=3	Тестирование параметра max_arg_count на значение 3
max_arg_count_invalid	Тестирование параметра max_arg_count на некорректное значение
max_arg_length=-1	Тестирование параметра max_arg_length на значение меньше нуля
max_arg_length=0	Тестирование параметра max_arg_length на значение ноль
max_arg_length=5	Тестирование параметра max_arg_length на значение 5
max_arg_length=10	Тестирование параметра max_arg_length на значение 10
max_arg_length_invalid	Тестирование параметра max_arg_length на некорректное значение
max_log_size=-1	Тестирование параметра max_log_size на значение меньше нуля
max_log_size=0	Тестирование параметра max_log_size на значение ноль
max_log_size=1	Тестирование параметра max_log_size на значение 1
max_log_size_invalid	Тестирование параметра max_log_size на некорректное значение
max_sql_length=-1	Тестирование параметра max_sql_length на значение меньше нуля
max_sql_length=0	Тестирование параметра max_sql_length на значение ноль
max_sql_length=30	Тестирование параметра max_sql_length на значение 30
max_sql_length=100	Тестирование параметра max_sql_length на значение 100
max_sql_length_invalid	Тестирование параметра max_sql_length на некорректное значение

prepare_ddl_statement	Логирование события подготовки запроса на создание таблицы
prepare_delete_statement	Логирование события подготовки запроса на удаление из таблицы
prepare_insert_statement	Логирование события подготовки запроса на вставку
prepare_select_statement	Логирование события подготовки запроса на выборку
prepare_statement_failed	Логирование события неудачной подготовки запроса
prepare_statement_unauth	Логирование события подготовки запроса непривилегированным пользователем
prepare_update_statement	Логирование события подготовки запроса на обновление
print_perf	Тестирование параметра print_perf
print_plan	Тестирование параметра print_plan
query_service_1	Логирование события запросов к сервису
query_service_2	Логирование события запросов к сервису
revoke_privileges	Логирование события отзыва привилегий
revoke_privileges_failed	Логирование события неудачного отзыва привилегий
revoke_privileges_unauth	Логирование события отзыва привилегий непривилегированным пользователем
rollback_transaction	Логирование отката транзакции
start_service	Логирование события запуска сервиса
start_service_failed	Логирование события неудачного запуска сервиса
start_transaction	Логирование запуска транзакции
start_transaction_2	Логирование запуска транзакции
time_threshold=-1	Тестирование параметра time_threshold на значение меньше нуля
time_threshold=0	Тестирование параметра time_threshold на значение ноль
time_threshold=50	Тестирование параметра time_threshold на значение 50
time_threshold_invalid	Тестирование параметра time_threshold на некорректное значение

Blob_Testing	
Уникальное имя	**Описание теста**
insertblobidintotable	Вставка идентификатора blob в таблицу
insertselectblobquery	Попытка вставки blob с использованием запроса insert ...select
openblobafterselect	Доступ к blob после выборки
openblobwithoutselect	Доступ к blob без выборки
updateblobidintable	Обновление идентификатора blob в таблице

Connect_Testing	
Уникальное имя	**Описание теста**
legacyauth_correctloginpass	Тест на legacy аутентификацию с правильным логином и паролем
legacyauth_incorrectlogin	Попытка legacy аутентификации с неправильным логином
legacyauth_incorrectpass	Попытка legacy аутентификации с неправильным паролем
multifactorauth_correctloginpass	Тест на многофакторнуюаутентификацию с правильным логином и паролем
multifactorauth_incorrectlogin	Попытка многофакторнойаутентификации с неправильным логином
multifactorauth_incorrectpass	Попытка многофакторнойаутентификации с неправильным паролем
srpauth_correctloginpass	Тест на srp аутентификацию с правильным логином и паролем
srpauth_incorrectlogin	Попытка srp аутентификации с неправильным логином
srpauth_incorrectpass	Попытка srp аутентификации с неправильным паролем
trustedauth	Тест на доверительную аутентификацию

DDL_Testing[17]	
Уникальное имя	**Описание теста**
altrcharsetaftergrant	Установка сортировки по умолчанию для набора символов после назначения прав
altrcharsetafterrevoke	Установка сортировки по умолчанию для набора символов после отмены прав
altrcharsetwithoutgrant	Установка сортировки по умолчанию для набора символов без прав
altrcolaftergrant	Добавление комментария к сортировке после назначения прав
altrcolafterrevoke	Добавление комментария к сортировке после отмены прав
altrcolwithoutgrant	Добавление комментария к сортировке без прав
altrdbaftergrant	Изменение БД после назначения прав
altrdbafterrevoke	Изменение БД после отмены прав
altrdbwithoutgrant	Изменение БД без прав
altrddltrgaftergrant	Изменение ddl-триггера после назначения прав
altrddltrgafterrevoke	Изменение ddl-триггера после отмены прав
altrddltrgwithoutgrant	Изменение ddl-триггера без прав

[17] В каталоге /WGO находятся аналогичные тесты, но привилегии назначаются пользователем или ролью с опцией WITH GRANT OPTION

altrdmltrgaftergrant	Изменение dml-триггера после назначения прав
altrdmltrgafterrevoke	Изменение dml-триггера после отмены прав
altrdmltrgwithoutgrant	Изменение dml-триггера без прав
altrdmnaftergrant	Изменение домена после назначения прав
altrdmnafterrevoke	Изменение домена после отмены прав
altrdmnwithoutgrant	Изменение домена без прав
altrexcaftergrant	Изменение текста сообщения пользовательского исключения после назначения прав
altrexcafterrevoke	Изменение текста сообщения пользовательского исключения после отмены прав
altrexcwithoutgrant	Изменение текста сообщения пользовательского исключения без прав
altrfiltaftergrant	Добавление комментария к BLOB фильтру после назначения прав
altrfiltafterrevoke	Добавление комментария к BLOB фильтру после отмены прав
altrfiltwithoutgrant	Добавление комментария к BLOB фильтру без прав
altrfuncaftergrant	Изменение хранимой функции после назначения прав
altrfuncafterrevoke	Изменение хранимой функции после отмены прав
altrfuncwithoutgrant	Изменение хранимой функции без прав
altrgenaftergrant	Изменение генератора после назначения прав
altrgenafterrevoke	Изменение генератора после отмены прав
altrgenwithoutgrant	Изменение генератора без прав
altrindaftergrant	Изменение активности индекса после назначения прав
altrindafterrevoke	Изменение активности индекса после отмены прав
altrindwithoutgrant	Изменение активности индекса без прав
altrpackaftergrant	Изменение пакета после назначения прав
altrpackafterrevoke	Изменение пакета после отмены прав
altrpackwithoutgrant	Изменение пакета без прав
altrprcaftergrant	Изменение хранимой процедуры после назначения прав
altrprcafterrevoke	Изменение хранимой процедуры после отмены прав
altrprcwithoutgrant	Изменение хранимой процедуры без прав
altrrolaftergrant	Изменение роли после назначения прав
altrrolafterrevoke	Изменение роли после отмены прав
altrrolwithoutgrant	Изменение роли без прав
altrsqcaftergrant	Изменение последовательности после назначения прав
altrsqcafterrevoke	Изменение последовательности после отмены прав
altrsqcwithoutgrant	Изменение последовательности без прав

altrtblaftergrant	Изменение таблицы после назначения прав
altrtblafterrevoke	Изменение таблицы после отмены прав
altrtblwithoutgrant	Изменение таблицы без прав
altrviewaftergrant	Изменение представления после назначения прав
altrviewafterrevoke	Изменение представления после отмены прав
altrviewwithoutgrant	Изменение представления без прав
crtcolaftergrant	Добавление новой сортировки после назначения прав
crtcolafterrevoke	Добавление новой сортировки после отмены прав
crtcolwithoutgrant	Добавление новой сортировки без прав
crtdbaftergrant	Создание БД после назначения прав
crtdbafterrevoke	Создание БД после отмены прав
crtdbwithoutgrant	Создание БД без прав
crtddltrgaftergrant	Создание ddl-триггера после назначения прав
crtddltrgafterrevoke	Создание ddl-триггера после отмены прав
crtddltrgwithoutgrant	Создание ddl-триггера без прав
crtdmltrgaftergrant	Создание dml-триггера после назначения прав
crtdmltrgafterrevoke	Создание dml-триггера после отмены прав
crtdmltrgwithoutgrant	Создание dml-триггерav без прав
crtdmnaftergrant	Создание домена после назначения прав
crtdmnafterrevoke	Создание домена после отмены прав
crtdmnwithoutgrant	Создание домена без прав
crtexcaftergrant	Создание исключения после назначения прав
crtexcafterrevoke	Создание исключения после отмены прав
crtexcwithoutgrant	Создание исключения без прав
crtfiltaftergrant	Объявление BLOB фильтра после назначения прав
crtfiltafterrevoke	Объявление BLOB фильтра после отмены прав
crtfiltwithoutgrant	Объявление BLOB фильтра без прав
crtfuncaftergrant	Создание хранимой функции после назначения прав
crtfuncafterrevoke	Создание хранимой функции после отмены прав
crtfuncwithoutgrant	Создание хранимой функции без прав
crtgenaftergrant	Создание генератора после назначения прав
crtgenafterrevoke	Создание генератора после отмены прав
crtgenwithoutgrant	Создание генератора без прав
crtindaftergrant	Создание индекса после назначения прав
crtindafterrevoke	Создание индекса после отмены прав
crtindwithoutgrant	Создание индекса без прав
crtoraltrexistddltrgaftergrant	Изменение существующего ddl-триггера оператором `CREATE OR ALTER` после назначения прав

crtoraltrexistddltrgafterrevoke	Изменение существующего ddl-триггера оператором CREATE OR ALTER после отмены прав
crtoraltrexistddltrgwithoutgrant	Изменение существующего ddl-триггера оператором CREATE OR ALTER без прав
crtoraltrexistdmltrgaftergrant	Изменение существующего dml-триггера оператором CREATE OR ALTER после назначения прав
crtoraltrexistdmltrgafterrevoke	Изменение существующего dml-триггера оператором CREATE OR ALTER после отмены прав
crtoraltrexistdmltrgwithoutgrant	Изменение существующего dml-триггера оператором CREATE OR ALTER без прав
crtoraltrexistexcaftergrant	Изменение существующего исключения оператором CREATE OR ALTER после назначения прав ALTER ANY
crtoraltrexistexcaftergrantcreate	Изменение существующего исключения оператором CREATE OR ALTER после назначения прав на создание исключения
crtoraltrexistexcafterrevoke	Изменение существующего исключения оператором CREATE OR ALTER после отмены прав
crtoraltrexistexcwithoutgrant	Изменение существующего исключения оператором CREATE OR ALTER без прав
crtoraltrexistfuncaftergrant	Изменение существующей функции оператором CREATE OR ALTER после назначения прав ALTER ANY
crtoraltrexistfuncaftergrantcreate	Изменение существующей функции оператором CREATE OR ALTER после назначения прав на создание функции
crtoraltrexistfuncafterrevoke	Изменение существующей функции оператором CREATE OR ALTER после отмены прав
crtoraltrexistfuncwithoutgrant	Изменение существующей функции оператором CREATE OR ALTER без прав
crtoraltrexistgenaftergrant	Изменение существующего генератора оператором CREATE OR ALTER после назначения прав ALTER ANY
crtoraltrexistgenaftergrantcreate	Изменение существующего генератора оператором CREATE OR ALTER после назначения прав на создание генератора
crtoraltrexistgenafterrevoke	Изменение существующего генератора оператором CREATE OR ALTER после отмены прав
crtoraltrexistgenwithoutgrant	Изменение существующего генератора оператором CREATE OR ALTER без прав
crtoraltrexistpackaftergrant	Изменение существующего пакета оператором CREATE OR ALTER после назначения прав ALTER ANY
crtoraltrexistpackaftergrantcreate	Изменение существующего пакета оператором CREATE OR ALTER после назначения прав на создание пакета
crtoraltrexistpackafterrevoke	Изменение существующего пакета оператором CREATE OR ALTER после отмены прав
crtoraltrexistpackwithoutgrant	Изменение существующего пакета оператором CREATE OR ALTER без прав

crtoraltrexistprcaftergrant	Изменение существующей процедуры оператором CREATE OR ALTER после назначения прав ALTER ANY
crtoraltrexistprcaftergrantcreate	Изменение существующей процедуры оператором CREATE OR ALTER после назначения прав на создание процедуры
crtoraltrexistprcafterrevoke	Изменение существующей процедуры оператором CREATE OR ALTER после отмены прав
crtoraltrexistprcwithoutgrant	Изменение существующей процедуры оператором CREATE OR ALTER без прав
crtoraltrexistsqcaftergrant	Изменение существующей последовательности оператором CREATE OR ALTER после назначения прав ALTER ANY
crtoraltrexistsqcaftergrantcreate	Изменение существующей последовательности оператором CREATE OR ALTER после назначения прав на создание последовательности
crtoraltrexistsqcafterrevoke	Изменение существующей последовательности оператором CREATE OR ALTER после отмены прав
crtoraltrexistsqcwithoutgrant	Изменение существующей последовательности оператором CREATE OR ALTER без прав
crtoraltrexistviewaftergrant	Изменение существующего представления оператором CREATE OR ALTER после назначения прав ALTER ANY
crtoraltrexistviewaftergrantcreate	Изменение существующего представления оператором CREATE OR ALTER на создание
crtoraltrexistviewafterrevoke	Изменение существующего представления оператором CREATE OR ALTER после отмены прав
crtoraltrexistviewwithoutgrant	Изменение существующего представления оператором CREATE OR ALTER без прав
crtoraltrnewddltrgaftergrant	Создание нового ddl-триггера оператором CREATE OR ALTER после назначения прав
crtoraltrnewddltrgafterrevoke	Создание нового ddl-триггера оператором CREATE OR ALTER после отмены прав
crtoraltrnewddltrgwithoutgrant	Создание нового ddl-триггера оператором CREATE OR ALTER без прав
crtoraltrnewdmltrgaftergrant	Создание нового dml-триггера оператором CREATE OR ALTER после назначения прав
crtoraltrnewdmltrgafterrevoke	Создание нового dml-триггера оператором CREATE OR ALTER после отмены прав
crtoraltrnewdmltrgwithoutgrant	Создание нового dml-триггера оператором CREATE OR ALTER без прав
crtoraltrnewexcaftergrant	Создание нового исключения оператором CREATE OR ALTER после назначения прав на создание
crtoraltrnewexcaftergrantalterany	Создание нового исключения оператором CREATE OR ALTER после назначения прав ALTER ANY
crtoraltrnewexcafterrevoke	Создание нового исключения оператором CREATE OR ALTER после отмены прав

crtoraltrnewexcwithoutgrant	Создание нового исключения оператором CREATE OR ALTER без прав
crtoraltrnewfuncaftergrant	Создание новой функции оператором CREATE OR ALTER после назначения прав на создание
crtoraltrnewfuncaftergrantalterany	Создание новой функции оператором CREATE OR ALTER после назначения прав ALTER ANY
crtoraltrnewfuncafterrevoke	Создание новой функции оператором CREATE OR ALTER после отмены прав
crtoraltrnewfuncwithoutgrant	Создание новой функции оператором CREATE OR ALTER без прав
crtoraltrnewgenaftergrant	Создание нового генератора оператором CREATE OR ALTER после назначения прав на создание
crtoraltrnewgenaftergrantalterany	Создание нового генератора оператором CREATE OR ALTER после назначения прав ALTER ANY
crtoraltrnewgenafterrevoke	Создание нового генератора оператором CREATE OR ALTER после отмены прав
crtoraltrnewgenwithoutgrant	Создание нового генератора оператором CREATE OR ALTER без прав
crtoraltrnewpackaftergrant	Создание нового пакета оператором CREATE OR ALTER после назначения прав на создание
crtoraltrnewpackaftergrantalterany	Создание нового пакета оператором CREATE OR ALTER после назначения прав ALTER ANY
crtoraltrnewpackafterrevoke	Создание нового пакета оператором CREATE OR ALTER после отмены прав
crtoraltrnewpackwithoutgrant	Создание нового пакета оператором CREATE OR ALTER без прав
crtoraltrnewprcaftergrant	Создание новой процедуры оператором CREATE OR ALTER после назначения прав на создание
crtoraltrnewprcaftergrantalterany	Создание новой процедуры оператором CREATE OR ALTER после назначения прав ALTER ANY
crtoraltrnewprcafterrevoke	Создание новой процедуры оператором CREATE OR ALTER после отмены прав
crtoraltrnewprcwithoutgrant	Создание новой процедуры оператором CREATE OR ALTER без прав
crtoraltrnewsqcaftergrant	Создание новой последовательности оператором CREATE OR ALTER после назначения прав на создание
crtoraltrnewsqcaftergrantalterany	Создание новой последовательности оператором CREATE OR ALTER после назначения прав ALTER ANY
crtoraltrnewsqcafterrevoke	Создание новой последовательности оператором CREATE OR ALTER после отмены прав
crtoraltrnewsqcwithoutgrant	Создание новой последовательности оператором CREATE OR ALTER без прав
crtoraltrnewviewaftergrant	Создание нового представления оператором CREATE OR ALTER после назначения прав
crtoraltrnewviewaftergrantalterany	Создание нового представления оператором CREATE OR ALTER после назначения прав

crtoraltrnewviewafterrevoke	Создание нового представления оператором CREATE OR ALTER после отмены прав
crtoraltrnewviewwithoutgrant	Создание нового представления оператором CREATE OR ALTER без прав
crtpackaftergrant	Создание пакета после назначения прав
crtpackafterrevoke	Создание пакета после отмены прав
crtpackwithoutgrant	Создание пакета без прав
crtprcaftergrant	Создание процедуры после назначения прав
crtprcafterrevoke	Создание процедуры после отмены прав
crtprcwithoutgrant	Создание процедуры без прав
crtrolaftergrant	Создание роли после назначения прав
crtrolafterrevoke	Создание роли после отмены прав
crtrolwithoutgrant	Создание роли без прав
crtsdwaftergrant	Создание теневой копии после назначения прав
crtsdwafterrevoke	Создание теневой копии после отмены прав
crtsdwwithoutgrant	Создание теневой копии без прав
crtsqcaftergrant	Создание последовательности после назначения прав
crtsqcafterrevoke	Создание последовательности после отмены прав
crtsqcwithoutgrant	Создание последовательности без прав
crttblaftergrant	Создание таблицы после назначения прав
crttblafterrevoke	Создание таблицы после отмены прав
crttblwithoutgrant	Создание таблицы без прав
crtviewaftergrant	Создание представления после назначения прав
crtviewafterrevoke	Создание представления после отмены прав
crtviewwithoutgrant	Создание представления без прав
dropcolaftergrant	Удаление сортировки после назначения прав
dropcolafterrevoke	Удаление сортировки после отмены прав
dropcolwithoutgrant	Удаление сортировки без прав
dropdbaftergrant	Удаление БД после назначения прав
dropdbafterrevoke	Удаление БД после отмены прав
dropdbwithoutgrant	Удаление БД без прав
dropddltrgaftergrant	Удаление ddl-триггера после назначения прав
dropddltrgafterrevoke	Удаление ddl-триггера после отмены прав
dropddltrgwithoutgrant	Удаление ddl-триггера без прав
dropdmltrgaftergrant	Удаление dml-триггера после назначения прав
dropdmltrgafterrevoke	Удаление dml-триггера после отмены прав
dropdmltrgwithoutgrant	Удаление dml-триггера без прав
dropdmnaftergrant	Удаление домена после назначения прав
dropdmnafterrevoke	Удаление домена после отмены прав

dropdmnwithoutgrant	Удаление домена без прав
dropexcaftergrant	Удаление исключения после назначения прав
dropexcafterrevoke	Удаление исключения после отмены прав
dropexcwithoutgrant	Удаление исключения без прав
dropfiltaftergrant	Удаление BLOB фильтра после назначения прав
dropfiltafterrevoke	Удаление BLOB фильтра после отмены прав
dropfiltwithoutgrant	Удаление BLOB фильтра без прав
dropfuncaftergrant	Удаление функции после назначения прав
dropfuncafterrevoke	Удаление функции после отмены прав
dropfuncwithoutgrant	Удаление функции без прав
dropgenaftergrant	Удаление генератора после назначения прав
dropgenafterrevoke	Удаление генератора после отмены прав
dropgenwithoutgrant	Удаление генератора без прав
dropindaftergrant	Удаление индекса после назначения прав
dropindafterrevoke	Удаление индекса после отмены прав
dropindwithoutgrant	Удаление индекса без прав
droppackaftergrant	Удаление пакета после назначения прав
droppackafterrevoke	Удаление пакета после отмены прав
droppackwithoutgrant	Удаление пакета без прав
dropprcaftergrant	Удаление процедуры после назначения прав
dropprcafterrevoke	Удаление процедуры после отмены прав
dropprcwithoutgrant	Удаление процедуры без прав
droprolaftergrant	Удаление роли после назначения прав
droprolafterrevoke	Удаление роли после отмены прав
droprolwithoutgrant	Удаление роли без прав
dropsdwaftergrant	Удаление теневой копии после назначения прав
dropsdwafterrevoke	Удаление теневой копии после отмены прав
dropsdwwithoutgrant	Удаление теневой копии без прав
dropsqcaftergrant	Удаление последовательности после назначения прав
dropsqcafterrevoke	Удаление последовательности после отмены прав
dropsqcwithoutgrant	Удаление последовательности без прав
droptblaftergrant	Удаление таблицы после назначения прав
droptblafterrevoke	Удаление таблицы после отмены прав
droptblwithoutgrant	Удаление таблицы без прав
dropviewaftergrant	Удаление представления после назначения прав
dropviewafterrevoke	Удаление представления после отмены прав
dropviewwithoutgrant	Удаление представления без прав

DML_Testing

Уникальное имя	Описание теста
deleteaftergrant	Удаление данных таблицы после назначения прав
deleteafterrevoke	Удаление данных таблицы после отмены прав
deletewithoutgrant	Удаление данных таблицы без прав
executefuncaftergrant_definer	Выполнение функции с опцией DEFINER после назначения прав EXECUTE
executefuncaftergrant_invoker	Выполнение функции с опцией INVOKER после назначения прав EXECUTE и SELECT
executefuncaftergrantexecute_invoker	Выполнение функции с опцией INVOKER после назначения прав EXECUTE
executefuncaftergrantselect_invoker	Выполнение функции с опцией INVOKER после назначения прав SELECT
executefuncafterrevoke_definer	Выполнение функции с опцией DEFINER после отмены прав EXECUTE
executefuncafterrevoke_invoker	Выполнение функции с опцией INVOKER после отмены прав EXECUTE и SELECT
executefuncafterrevokeexecute_invoker	Выполнение функции с опцией INVOKER после отмены прав EXECUTE
executefuncafterrevokeselect_invoker	Выполнение функции с опцией INVOKER после отмены прав SELECT
executefuncwithoutgrant_definer	Выполнение функции с опцией DEFINER без прав
executefuncwithoutgrant_invoker	Выполнение функции с опцией INVOKER без прав
executepackaftergrant_definer	Выполнение пакета с опцией DEFINER после назначения прав EXECUTE
executepackaftergrant_invoker	Выполнение пакета с опцией INVOKER после назначения прав EXECUTE и SELECT
executepackaftergrantexecute_invoker	Выполнение пакета с опцией INVOKER после назначения прав EXECUTE
executepackaftergrantselect_invoker	Выполнение пакета с опцией INVOKER после назначения прав SELECT
executepackafterrevoke_definer	Выполнение пакета с опцией DEFINER после отмены прав EXECUTE
executepackafterrevoke_invoker	Выполнение пакета с опцией INVOKER после отмены прав EXECUTE и SELECT
executepackafterrevokeexecute_invoker	Выполнение пакета с опцией INVOKER после отмены прав EXECUTE
executepackafterrevokeselect_invoker	Выполнение пакета с опцией INVOKER после отмены прав SELECT
executepackwithoutgrant_definer	Выполнение пакета с опцией DEFINER без прав
executepackwithoutgrant_invoker	Выполнение пакета с опцией INVOKER без прав
executeprocaftergrant_definer	Выполнение процедуры с опцией DEFINER после назначения прав EXECUTE
executeprocaftergrant_invoker	Выполнение процедуры с опцией INVOKER после назначения прав EXECUTE и SELECT

executeprocaftergrantexecute_invoker	Выполнение процедуры с опцией INVOKER после назначения прав EXECUTE
executeprocaftergrantselect_invoker	Выполнение процедуры с опцией INVOKER после назначения прав SELECT
executeprocafterrevoke_definer	Выполнение процедуры с опцией DEFINER после отмены прав EXECUTE
executeprocafterrevoke_invoker	Выполнение процедуры с опцией INVOKER после отмены прав EXECUTE и SELECT
executeprocafterrevokeexecute_invoker	Выполнение процедуры с опцией INVOKER после отмены прав EXECUTE
executeprocafterrevokeselect_invoker	Выполнение процедуры с опцией INVOKER после отмены прав SELECT
executeprocwithoutgrant_definer	Выполнение процедуры с опцией DEFINER без прав
executeprocwithoutgrant_invoker	Выполнение процедуры с опцией INVOKER без прав
insertaftergrant	Вставка данных в таблицу после назначения прав
insertafterrevoke	Вставка данных в таблицу после отмены прав
insertwithoutgrant	Вставка данных в таблицу без прав
referencesaftergrant	Ссылка на столбцы внешним ключом после назначения прав
referencesafterrevoke	Ссылка на столбцы внешним ключом после отмены прав
referencescolaftergrant	Ссылка на столбцы внешним ключом после назначения прав
referencescolaftergrant2	Ссылка на столбцы внешним ключом после назначения прав
referencescolafterrevoke	Ссылка на столбцы внешним ключом после отмены прав
referencescolwithoutgrant	Ссылка на столбцы внешним ключом без прав
referencesowntabaftergrant	Ссылка на столбцы той же таблицы внешним ключом после назначения прав
referencesowntabaftergrant2	Ссылка на столбцы той же таблицы внешним ключом после назначения прав
referencesowntabafterrevoke	Ссылка на столбцы той же таблицы внешним ключом после отмены прав
referencesowntabwithoutgrant	Ссылка на столбцы той же таблицы внешним ключом без прав
referenceswithoutgrant	Ссылка на столбцы внешним ключом без прав
selectaftergrant	Выборка из таблицы после назначения прав
selectafterrevoke	Выборка из таблицы после отмены прав
selectallaftergrant	Выборка всех полей из таблицы после назначения прав
selectwithoutgrant	Выборка из таблицы без прав
updateaftergrant	Обновление данных таблицы после назначения прав
updateafterrevoke	Обновление данных таблицы после отмены прав

updateafterrevoke2	Обновление данных таблицы после отмены прав
updatenotgrantfield	Обновление полей таблицы после назначения прав на обновление других полей
updatewithoutgrant	Обновление данных таблицы без прав
usageexcaftergrant	Использование исключения в процедуре после назначения прав
usageexcafterrevoke	Использование исключения в процедуре после отмены прав
usageexcwithoutgrant	Использование исключения в процедуре без прав
usagegenaftergrant	Приращение генератора после назначения прав
usagegenafterrevoke	Приращение генератора после отмены прав
usagegenwithoutgrant	Приращение генератора без прав
usagesqcaftergrant	Приращение последовательности после назначения прав
usagesqcafterrevoke	Приращение последовательности после отмены прав
usagesqcwithoutgrant	Приращение последовательности без прав

Policy_Testing	
Уникальное имя	**Описание теста**
compliantdiffcase	Тест на удовлетворяющую политике сложность пароля (регистр)
compliantneedchar	Тест на удовлетворяющую политике сложность пароля (буквы)
compliantneeddigit	Тест на удовлетворяющую политике сложность пароля (цифры)
compliantpaslength	Тест на удовлетворяющую политике длину пароля
failedcount_reset01	Тест на проверку максимального количества ошибок при аутентификации с последующим успешным выполнение `RESET USER` пользователем `SYSDBA`
failedcount_reset02	Тест на проверку максимального количества ошибок при аутентификации с последующим успешным выполнение `RESET USER` администратором БД безопасности
failedcount_reset03	Тест на проверку максимального количества ошибок при аутентификации с последующим неуспешным выполнение `RESET USER` администратором БД
failedcount_reset04	Тест на проверку максимального количества ошибок при аутентификации с последующим неуспешным выполнение `RESET USER` обычным пользователем
failedcountcontrol	Тест на проверку максимального количества ошибок при аутентификации
pswuniquecountcontrol	Тест на удовлетворяющей политике разрыв между повторяющимися паролями

pswuniquecountfailed	Тест на неудовлетворяющей политике разрыв между повторяющимися паролями
uncompliantdiffcase	Тест на неудовлетворяющую политике сложность пароля (регистр)
uncompliantneedchar	Тест на неудовлетворяющую политике сложность пароля (буквы)
uncompliantneeddigit	Тест на неудовлетворяющую политике сложность пароля (цифры)
uncompliantpaslength	Тест на неудовлетворяющую политике длину пароля

Roles_Testing	
Уникальное имя	**Описание теста**
connectwithroleusing	Подключение с указанием роли
connrolegrantupdate	Тест на проверку прав роли на Update
connwithnongrantrole	Подключение с ролью, которая не была назначена пользователю
equalroleandusernames	Создание роли и пользователя с одинаковыми именами
grantdefaultallroles	Установка пользователю всех ролей по умолчанию
grantdefaultoneofroles	Установка пользователю только одной роли по умолчанию
grantdefaultoneofroles2	Установка пользователю только одной роли по умолчанию
grantinsroletoupdrole	Тест на проверку передачи прав от роли к роли
revokedefaultroles	Отмена прав роли по умолчанию
revokedefaultroles2	Отмена прав роли по умолчанию
revokerolefromrole	Ограничение прав роли, отмена прав роли от роли
rolecrosslinking	Тест на перекрестную ссылочность между ролями
rolesgrantscumulating	Тест на кумулятивное действие ролей
setroles	Тестирование оператора `SET ROLE`

Service_Testing	
Уникальное имя	**Описание теста**
adduseraftergrant	Добавление пользователя после назначения прав
adduserafterrevoke	Добавление пользователя после ограничения прав
adduserwithoutgrant	Добавление пользователя без прав
backingaftergrant	Создание резервной копии БД после назначения прав
backingafterrevoke	Создание резервной копии БД после отмены прав
backingwithoutgrant	Создание резервной копии БД без прав
deleteusraftergrant	Удаление пользователя после назначения прав
deleteusrafterrevoke	Удаление пользователя после отмены прав

deleteusrwithoutgrant	Удаление пользователя без прав
getdbstataftergrant	Получение статистики базы данных после назначения прав
getdbstatafterrevoke	Получение статистики базы данных после отмены прав
getdbstatwithoutgrant	Получение статистики базы данных без прав
getusrlstaftergrant	Получение списка пользователей после назначения прав
getusrlstafterrevoke	Получение списка пользователей после отмены прав
getusrlstwithoutgrant	Получение списка пользователей без прав
modifyusraftergrant	Изменение пользователя после назначения прав
modifyusrafterrevoke	Изменение пользователя после отмены прав
modifyusrwithoutgrant	Изменение пользователя без прав
repairdbaftergrant	Исправление БД после назначения прав
repairdbafterrevoke	Исправление БД после отмены прав
repairdbwithoutgrant	Исправление БД без прав
restoreaftergrant	Восстановление базы данных после назначения прав
restoreafterrevoke	Восстановление базы данных после отмены прав
restorewithoutgrant	Восстановление базы данных без прав
setdbpropaftergrant	Изменение свойств базы данных после назначения прав
setdbpropafterrevoke	Изменение свойств базы данных после отмены прав
setdbpropwithoutgrant	Изменение свойств базы данных без прав

WipeMemory_Testing	
Уникальное имя	**Описание теста**
wipedbondisc	Очистка данных с диска после удаления БД
wiperecordinmemory	Очистка данных из памяти после удаления записи из таблицы
wiperecordondisc	Очистка данных с диска после удаления записи из таблицы
wipetableinmemory	Очистка данных из памяти после удаления таблицы
wipetableondisc	Очистка данных с диска после удаления таблицы

Приложение Ж Скрипты для обновления полнотекстового поиска

Скрипт pre_update_fts.sql

```
CREATE TABLE FTS$INDICES_PREV (
  FTS$INDEX_NAME    CHAR(31) CHARACTER SET UNICODE_FSS NOT NULL PRIMARY KEY,
  FTS$DESCRIPTION   BLOB SUB_TYPE 1 SEGMENT SIZE 80 CHARACTER SET UNICODE_FSS,
  FTS$ANALYZER      VARCHAR(255) CHARACTER SET UNICODE_FSS
);

INSERT INTO FTS$INDICES_PREV (FTS$INDEX_NAME, FTS$DESCRIPTION, FTS$ANALYZER)
  SELECT fi.FTS$INDEX_NAME, fi.FTS$DESCRIPTION, fis.FTS$ANALYZER
  FROM FTS$INDICES fi LEFT JOIN FTS$INDEX_SEGMENTS fis
                       ON fi.FTS$INDEX_NAME = fis.FTS$INDEX_NAME;

CREATE TABLE FTS$INDEX_SEGMENTS_PREV (
  FTS$INDEX_NAME     CHAR(31) CHARACTER SET UNICODE_FSS NOT NULL,
  FTS$RELATION_NAME  CHAR(31) CHARACTER SET UNICODE_FSS NOT NULL,
  FTS$FIELD_NAME     CHAR(31) CHARACTER SET UNICODE_FSS NOT NULL
);

INSERT INTO FTS$INDEX_SEGMENTS_PREV (FTS$INDEX_NAME, FTS$RELATION_NAME,
                                     FTS$FIELD_NAME)
  SELECT FTS$INDEX_NAME, FTS$RELATION_NAME, FTS$FIELD_NAME
  FROM FTS$INDEX_SEGMENTS;

COMMIT;

DROP PROCEDURE fts$reindex;
DROP PROCEDURE fts$full_reindex;
DROP PROCEDURE fts$stopdaemon;
DROP PROCEDURE fts$startdaemon;
DROP PROCEDURE fts$search;
DROP PROCEDURE fts$apply_metadata_changes;
DROP PROCEDURE fts$drop_field_from_index;
DROP PROCEDURE fts$add_field_to_index;
DROP PROCEDURE fts$drop_index;
DROP PROCEDURE fts$create_index;
DROP FUNCTION fts$fillpool;
COMMIT;
DROP TABLE fts$pool;
DROP TABLE fts$lucene_file_system;
DROP TABLE fts$index_segments;
DROP TABLE fts$indices;
DROP GENERATOR fts$trigger_id;
COMMIT;

SET TERM ^ ;
```

```
EXECUTE BLOCK
RETURNS (trig_name VARCHAR(255))
AS
BEGIN
  FOR SELECT RDB$TRIGGER_NAME FROM rdb$triggers
    WHERE RDB$TRIGGER_NAME STARTS \rr{FTS$TRIG_}
    INTO :trig_name
  DO
  BEGIN
    EXECUTE STATEMENT \rr{DROP TRIGGER} || :trig_name;
    SUSPEND;
  END
END^

SET TERM ; ^

COMMIT;
```

Скрипт post_update_fts.sql

```
SET TERM ^ ;

EXECUTE BLOCK
RETURNS (idx_name FTS$OBJECT_NAME)
AS
  DECLARE VARIABLE description BLOB SUB_TYPE 1 CHARACTER SET UNICODE_FSS;
  DECLARE VARIABLE analyzer FTS$NAME;
BEGIN
  FOR SELECT FTS$INDEX_NAME, FTS$DESCRIPTION, FTS$ANALYZER
    FROM fts$indices_prev
    INTO :idx_name, :description, :analyzer
  DO
  BEGIN
    EXECUTE PROCEDURE FTS$CREATE_INDEX(:idx_name, :analyzer, :description);
    SUSPEND;
  END
END^

EXECUTE BLOCK
RETURNS (idx_name FTS$OBJECT_NAME)
AS
  DECLARE VARIABLE rel_name FTS$OBJECT_NAME;
  DECLARE VARIABLE field_name FTS$OBJECT_NAME;
BEGIN
  FOR SELECT FTS$INDEX_NAME, FTS$RELATION_NAME, FTS$FIELD_NAME
    FROM fts$index_segments_prev
    INTO :idx_name, :rel_name, :field_name
  DO
  BEGIN
    EXECUTE PROCEDURE FTS$ADD_FIELD_TO_INDEX(:idx_name, :rel_name, :field_name);
    SUSPEND;
```

```
  END
END^

SET TERM ; ^

COMMIT;

DROP TABLE fts$index_segments_prev;
DROP TABLE fts$indices_prev;
DROP GENERATOR fts$trigger_id;

COMMIT;

EXECUTE PROCEDURE FTS$FULL_REINDEX;

COMMIT;
```

Приложение З Производительность СУБД Ред База Данных 3.0

Время выполнения операции резервного копирования файла базы данных размером не более 1ТБ составляет менее 6 часов при условии, что файл базы данных размещается на серверном SSD-диске, подключенном по интерфейсу SATA III, а файл резервной копии базы данных размещается на отдельном HDD-диске.

Время выполнения операции восстановления БД размером не более 1ТБ из резервной копии составляет менее 15 часов при условии, что файл базы данных размещается на серверном SSD-диске, подключенном по интерфейсу SATA III, а файл резервной копии базы данных размещается на отдельном HDD-диске.

Время сборки устаревших версий записей для файла БД размером 1ТБ составляет менее 15 минут при условии, что файл базы данных размещается на серверном SSD-диске, подключенном по интерфейсу SATA III.

Потребление оперативной памяти СУБД при каждом новом подключении к базе данных, увеличивается не пропорционально количеству подключений, а за вычетом объема кэша общих данных. В таком режиме при установлении нового соединения память используется только на нужды процесса и хранение метаданных, но не на хранение страниц из базы данных. Обрабатывается большее количество подключений, чем при использовании раздельных кэша в разных процессах. При обработке подключений пользователей в рамках одного процесса доступ к файлу базы данных блокируется для обеспечения его эксклюзивного использования и предотвращения открытия этого файла другими процессами для недопущения его некорректного изменения.

Для исключения риска невозможности восстановления базы данных из резервной копии, восстановление выполняется в несколько этапов: создание таблиц, восстановление данных, создания хранимых процедур и функций.

Каждый этап восстановления БД из резервной копии выполняется в рамках своей транзакции. Хранимые процедуры и функции, восстановление которых завершается ошибкой, получают соответствующую отметку (с формированием соответствующей записи в журнале восстановления) для обеспечения возможности их последующего восстановления с сохранением всех ранее восстановленных данных.