Ред База Данных

Версия 2.5

Руководство администратора

Содержание

1 Установка сервера СУБД «Ред База Данных»

1.1 Поддерживаемые ОС

СУБД «Ред База Данных» может функционировать на следующих ОС:

- Microsoft Windows (32-bit);
- Microsoft Windows (x64);
- Linux x86, Linux x86_64 дистрибутивы, использующие:
  - glibc 2.4 и старше;
  - libstdc++ от gcc 3.4.3 и старше;
- Linux дистрибутивы, поддерживающие Linux Standard Base ISO/IEC 23360, начиная с версии 3.0.

Рекомендуемые ОС семейства Linux:

- Red Hat Enterprise Linux 5.4 и старше;
- CentOS 6.4 и старше;
- Альт Линукс;
- Astra Linux;
- Mandriva Linux 2010 и старше;
- SUSE 11 и старше;
- ROSA 2011 и старше.

1.2 Установка в ОС Windows

Перед установкой СУБД «Ред База Данных» необходимо определить разрядность используемой операционной системы, и на каком языке будет отображаться процесс установки (предусмотрена установка на русском и английском языках). После этого необходимо перейти в соответствующую папку (win32 или win64) и запустить установку СУБД «Ред База Данных» с помощью файла setup_rbd_2.5.x.x.en.msi (для отображения процесса установки на английском языке) или setup_rbd_2.5.x.x.ru.msi (для отображения процесса установки на русском языке).

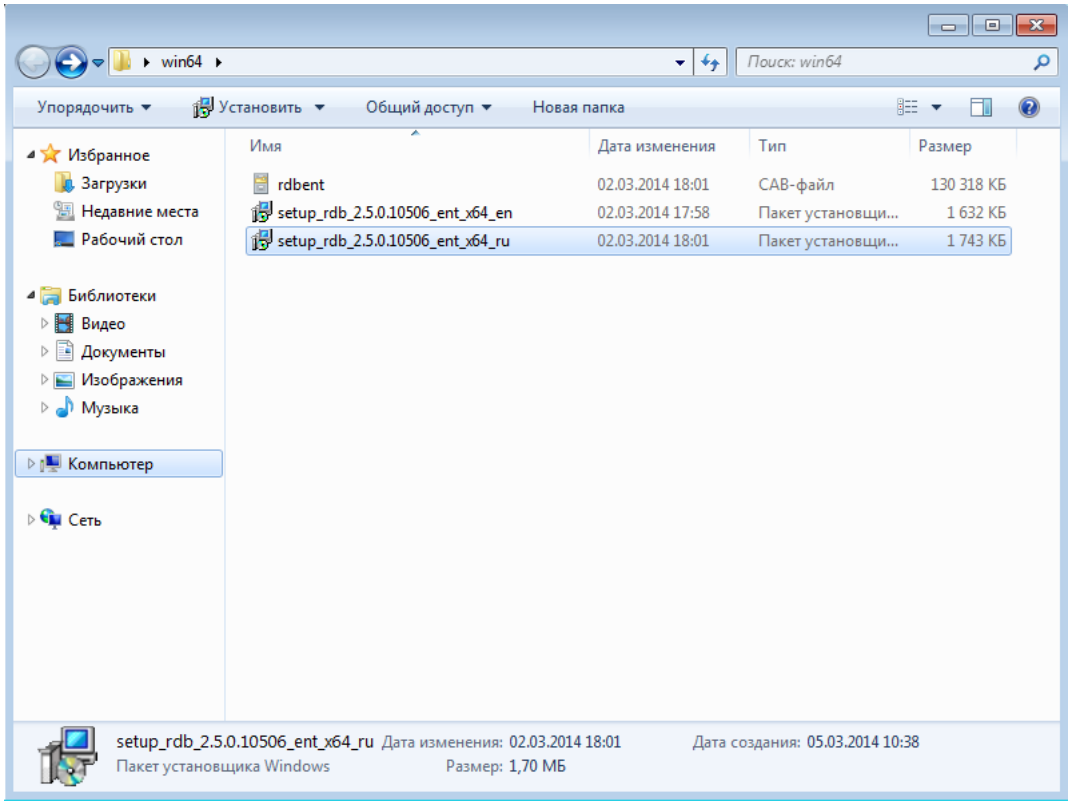


Рисунок 1 — Запуск установки СУБД «Ред База Данных»

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.

Внимание! Для установки сервера «Ред База Данных» 2.5 необходимы права администратора.

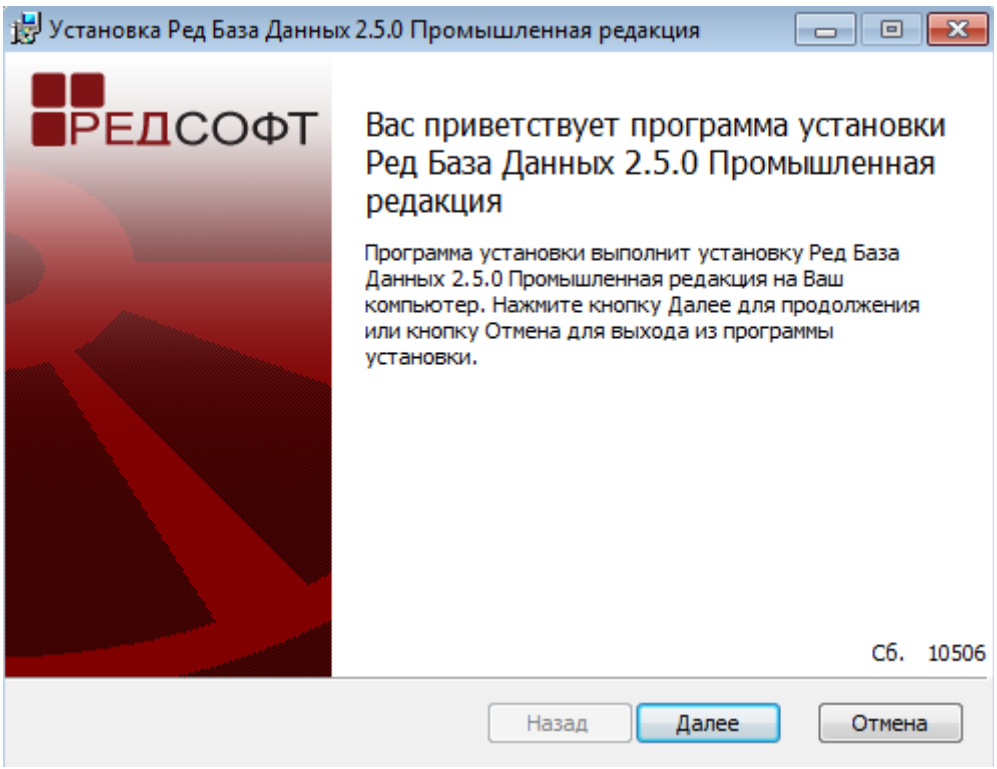


Рисунок 2 — Начало установки

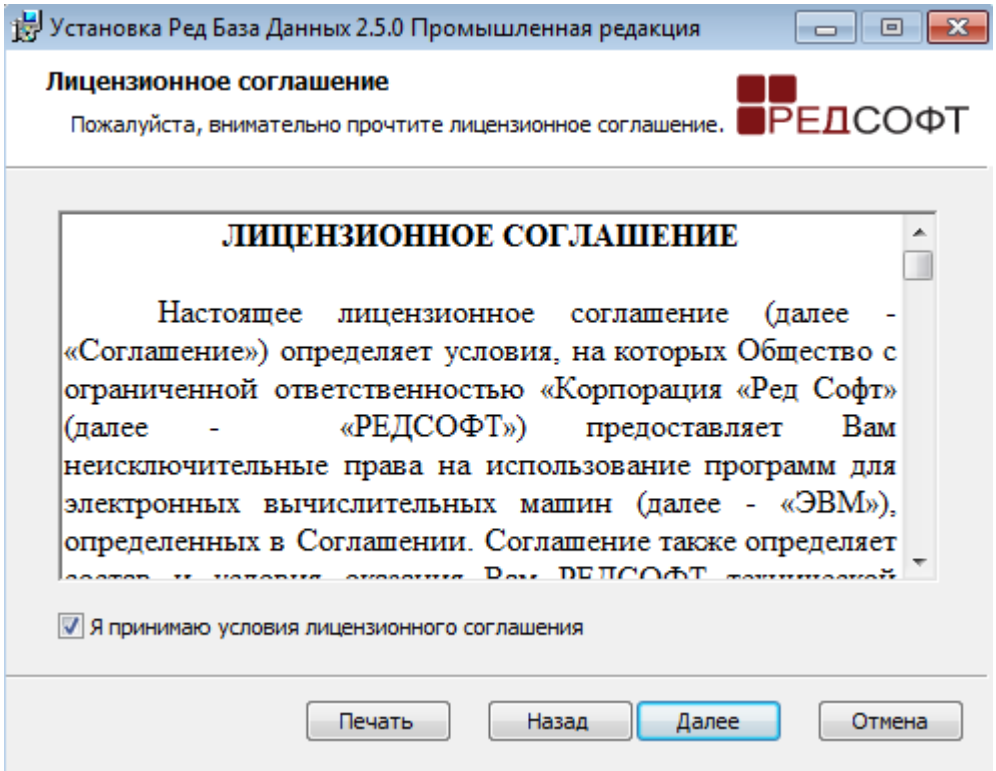


Рисунок 3 — Экран лицензионного соглашения

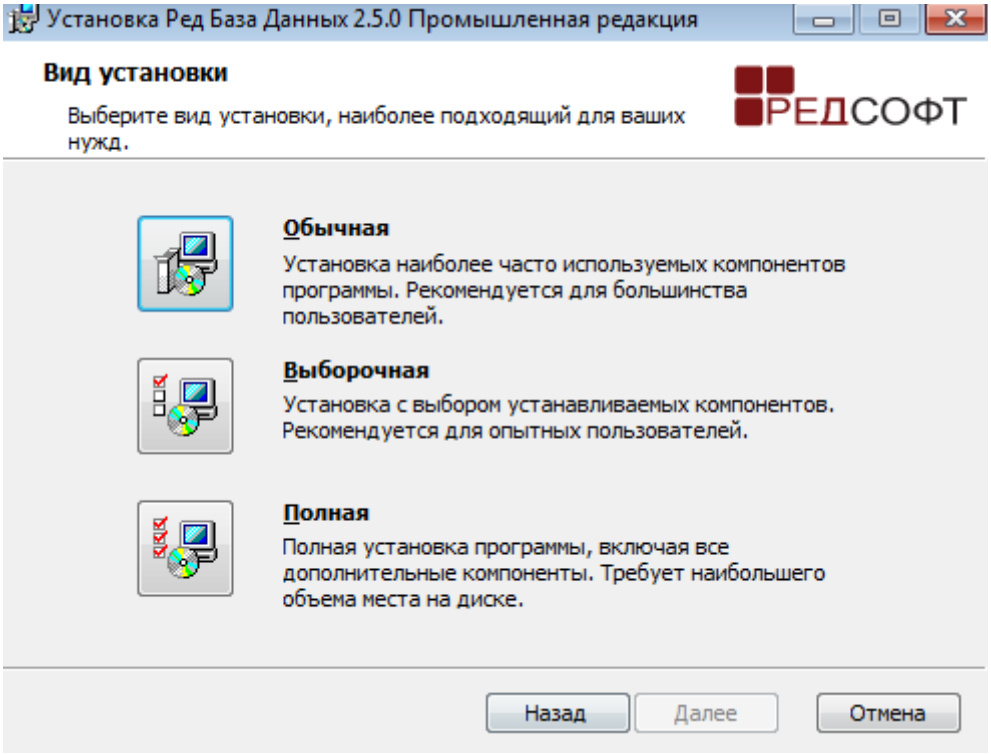


Рисунок 4 — Выбор варианта установки

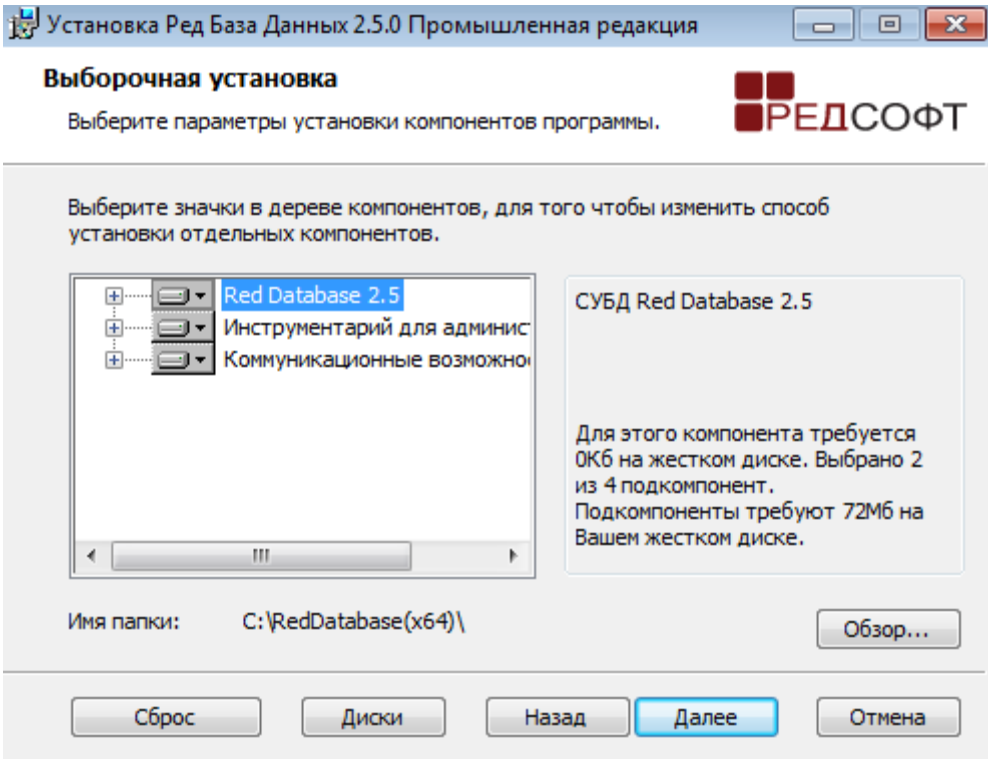


Рисунок 5 — Детальный выбор компонентов для установки

Данное окно появляется в случае выбора на предыдущем шаге варианта «Выборочная установка» Здесь можно указать путь к каталогу установки сервера, а также выбрать необходимый набор компонентов и утилит.

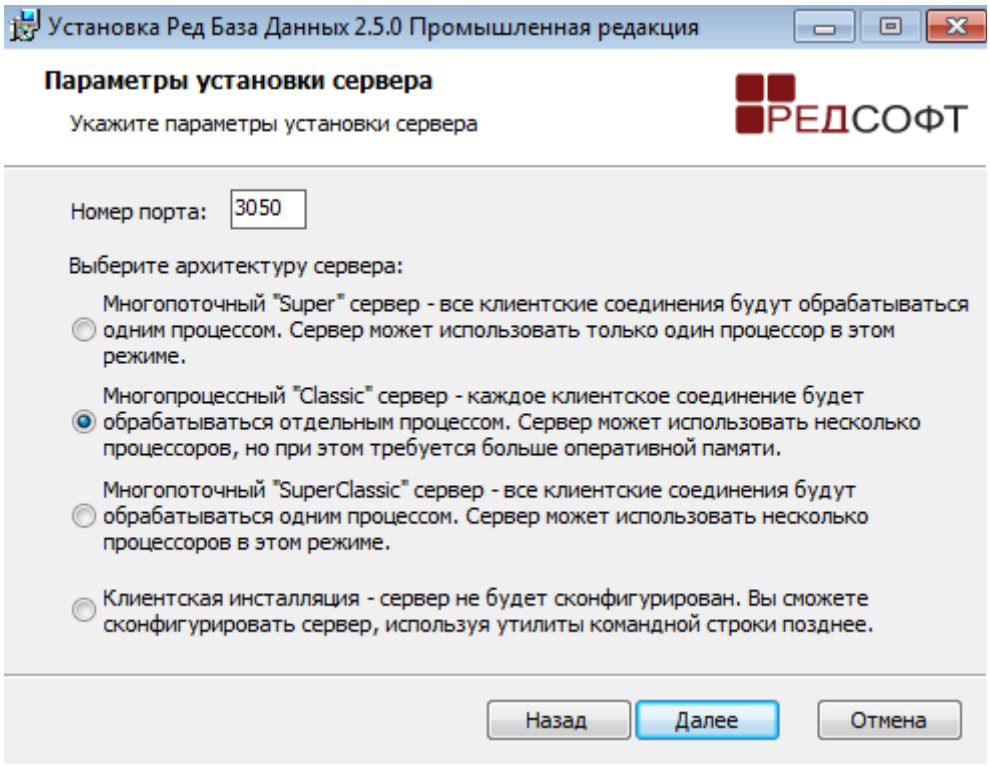


Рисунок 6 — Конфигурация сервера

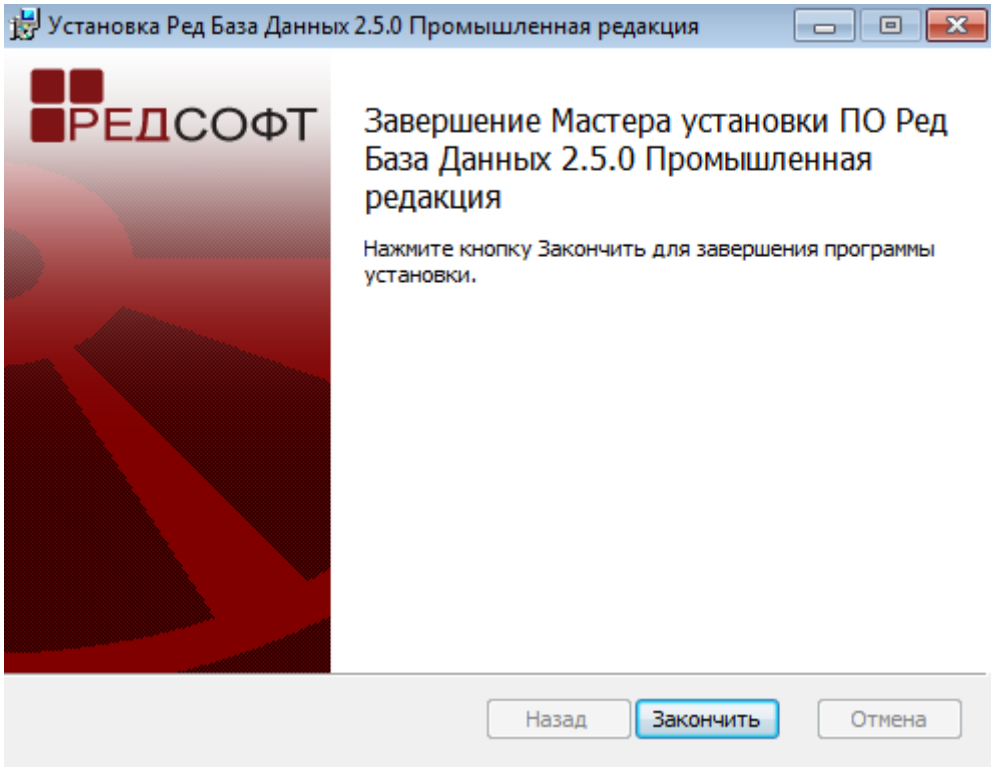


Рисунок 7 — Завершение установки

Настройка работы сервера осуществляется через файл конфигурации firebird.conf, расположенный в каталоге установки сервера.

При установке сервера «Ред База Данных» в системе регистрируется и запускается служба сервера, которая называется Red Database server. Однако если выбран вариант клиентской инсталляции, то в этом случае служба сервера не будет сконфигурирована и запущена. Кроме того, при любом варианте установки создает-

ся ветвь в системном реестре (HKLM\SOFTWARE\Firebird Project), в которой хранятся параметры установки, необходимые для корректной работы сервера.

Примечание: Изменить архитектуру сервера (например, суперсервер на классик сервер) а также зарегистрировать/разрегистрировать службу в системе можно с помощью соответствующих bat-файлов в каталоге bin\ сервера.

При установке «Ред База Данных» 2.5 можно, выбрав соответствующую опцию в инсталляторе, скопировать на диск исходные коды сервера СУБД «Ред База Данных». Эти исходные коды можно впоследствии использовать для ручной сборки и компиляции бинарных файлов сервера. Для этого необходимо иметь компилятор C++ 8-й версии.

Компиляция 32-х разрядного ядра СУБД производится запуском файла:

```
<Каталог Ред База Данных
2.5>\\Source\builds\win32\RUN_ALL.BAT
```

1.3 Установка в Unix-системах

1.3.1 Установка в графическом режиме

Файлы «Ред База Данных» 2.5 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» 2.5 на Ваш компьютер.

Для установки СУБД «Ред База Данных» необходимо скопировать дистрибутивный файл RedDatabase-2.5.X.X-X-linux-x86_64.bin на жесткий диск, а в операционной системе назначить в правах этого файла разрешение на исполнение:

```
# chmod +x RedDatabase-2.5.X.X-X-linux-x86_64.bin
```

После этого запустить установку СУБД «Ред База Данных»:

```
# ./RedDatabase-2.5.X.X-X-linux-x86_64.bin
```

Внимание! Для установки сервера «Ред База Данных» 2.5 необходимы права суперпользователя (root).

Внимание! Существуют различные виды пакетов для 32-х и 64-х разрядных Unix-систем.

Инсталляция СУБД «Ред База Данных» осуществляется с помощью стандартного мастера установки программ.

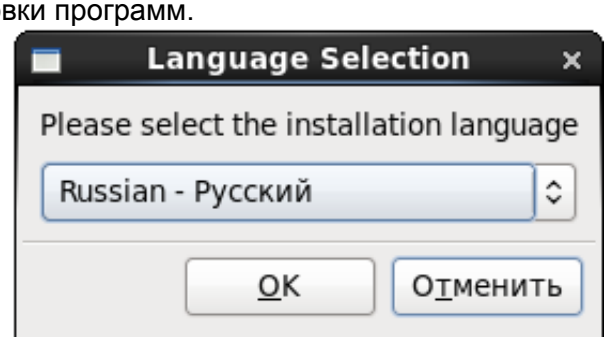


Рисунок 8 — Окно выбора языка, используемого в процессе установки

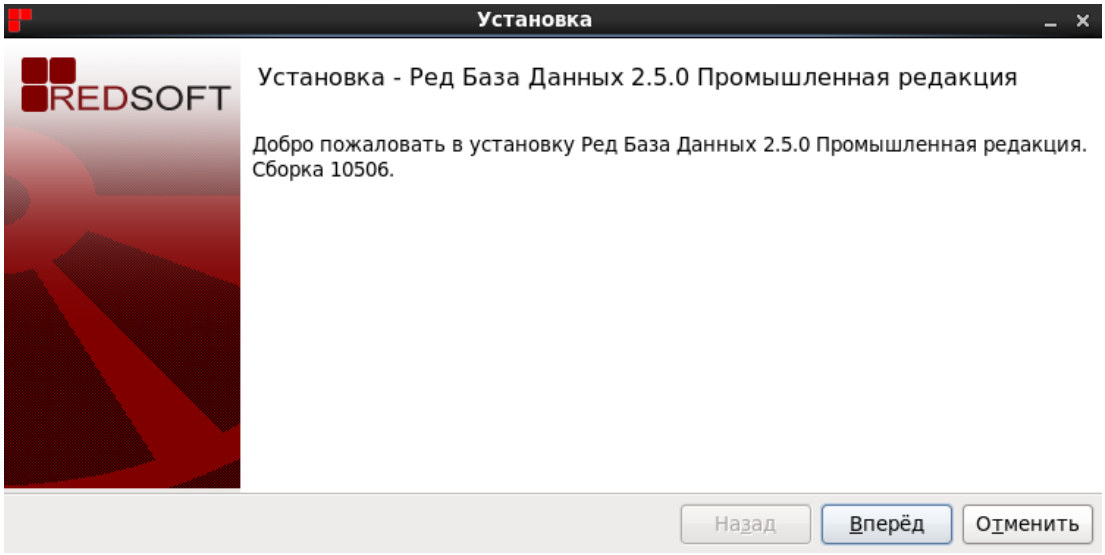


Рисунок 9 — Начало установки

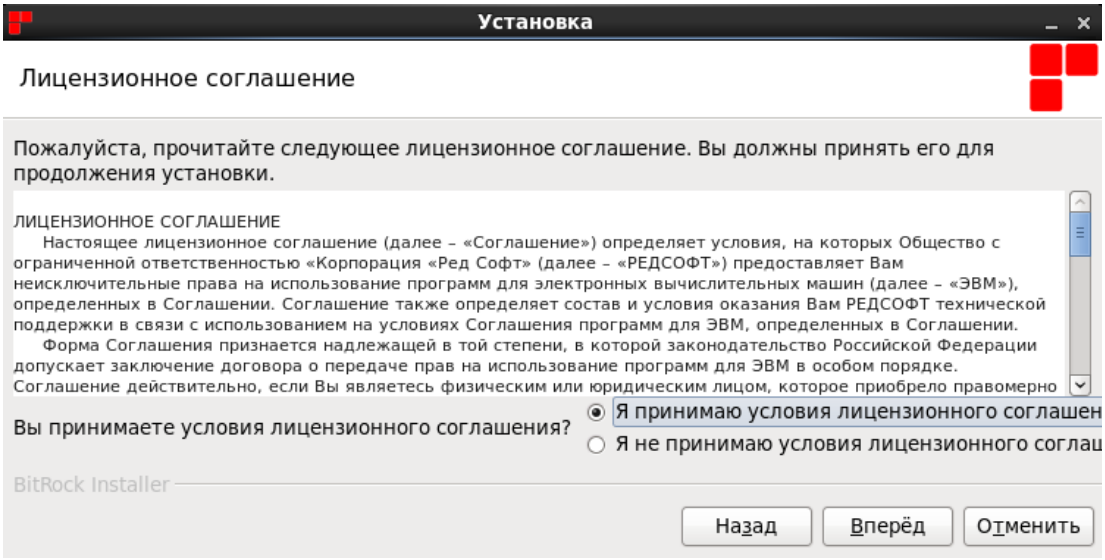


Рисунок 10 — Лицензионное соглашение

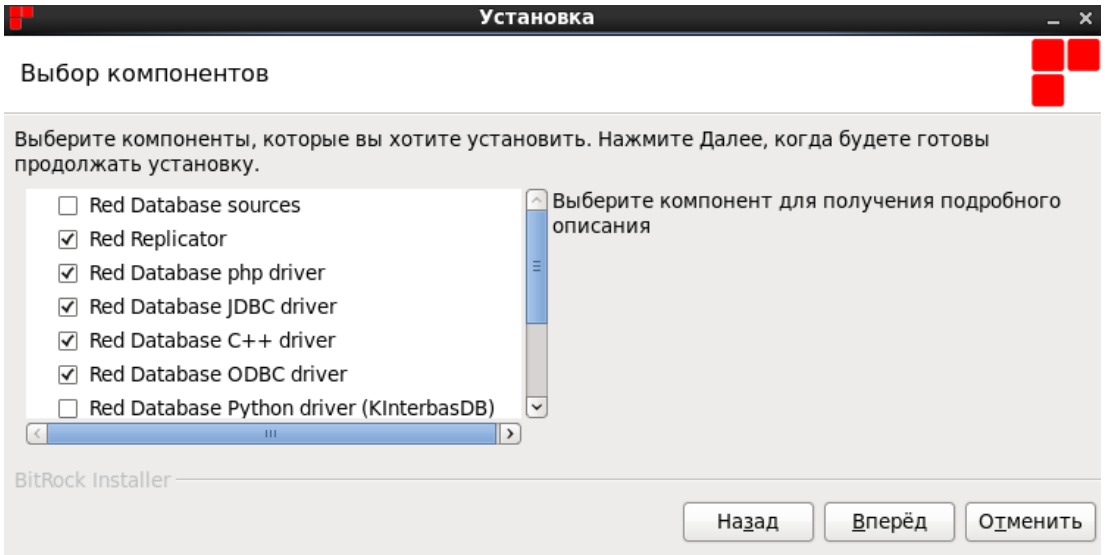


Рисунок 11 — Выбор компонентов для установки

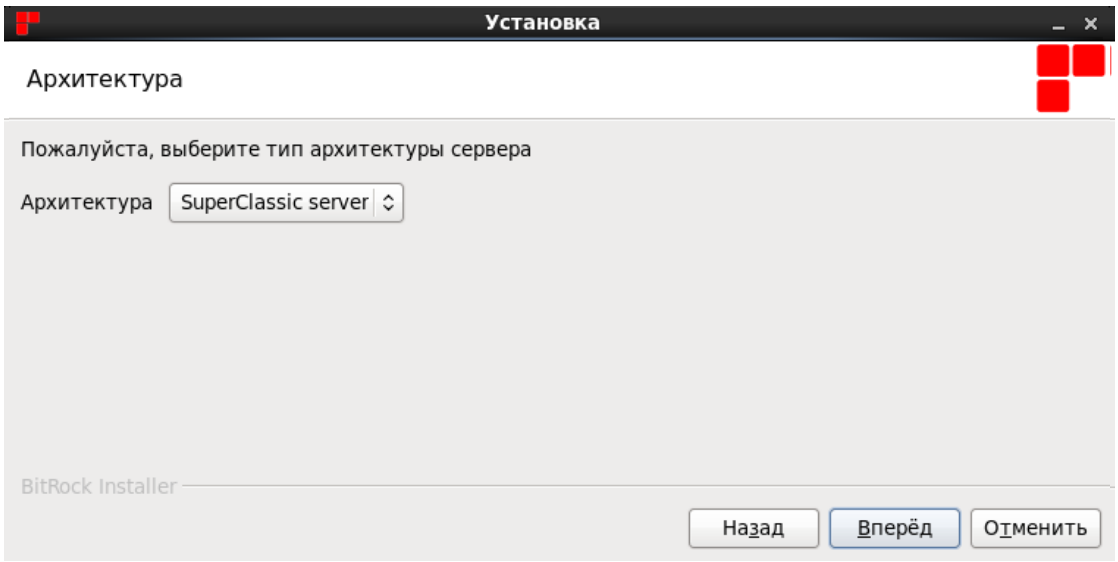


Рисунок 12— Выбор архитектуры сервера

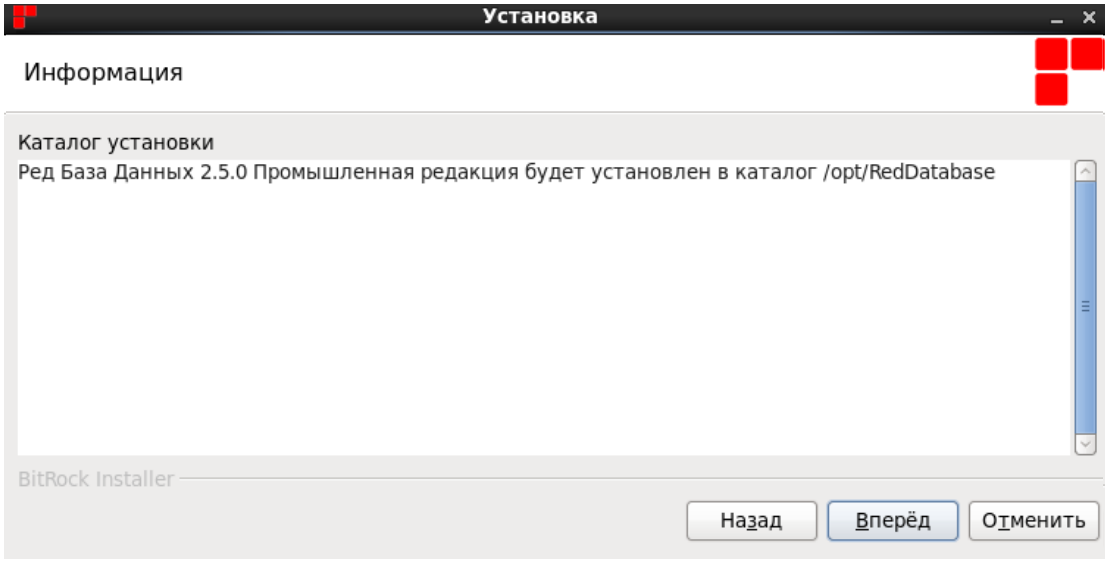


Рисунок 13— Информационное окно с указанием каталога установки

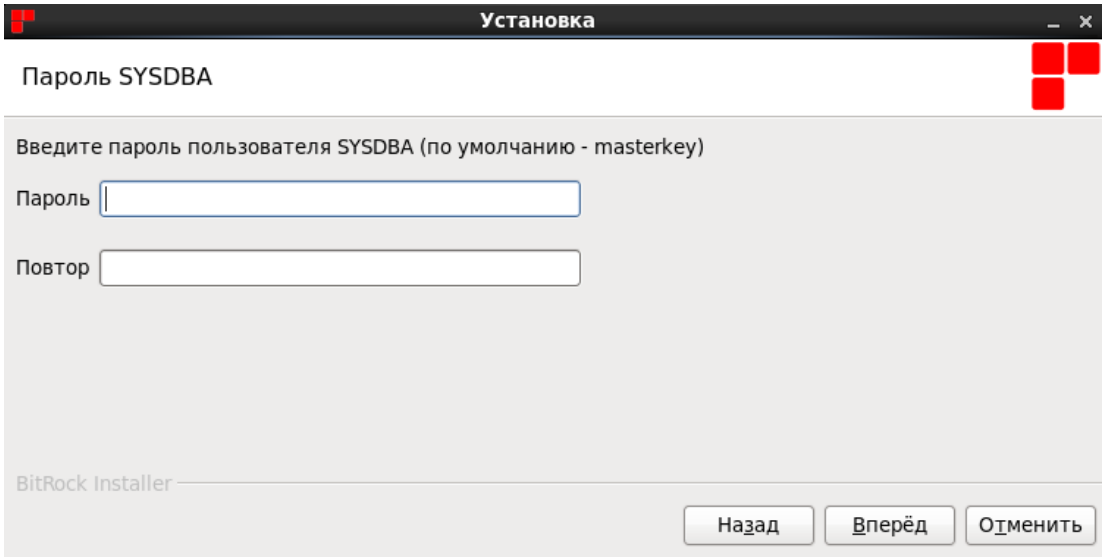


Рисунок 14 - Ввод пароля пользователя SYSDBA

Если оставить поля незаполненными, то для пользователя SYSDBA будет использован пароль по умолчанию — masterkey.

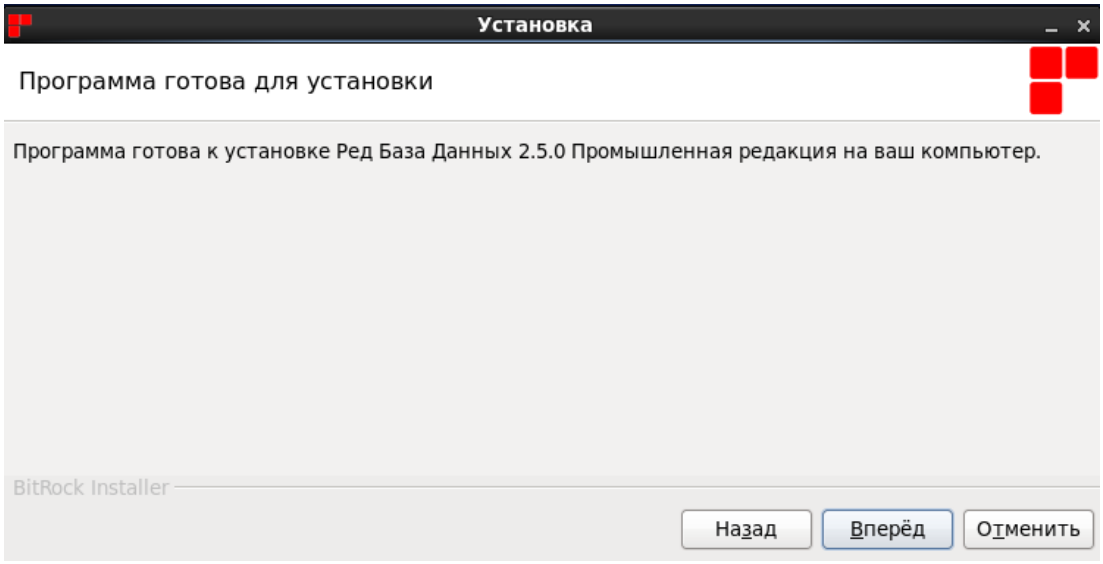


Рисунок 15— Информационное окно с сообщением о готовности к установке

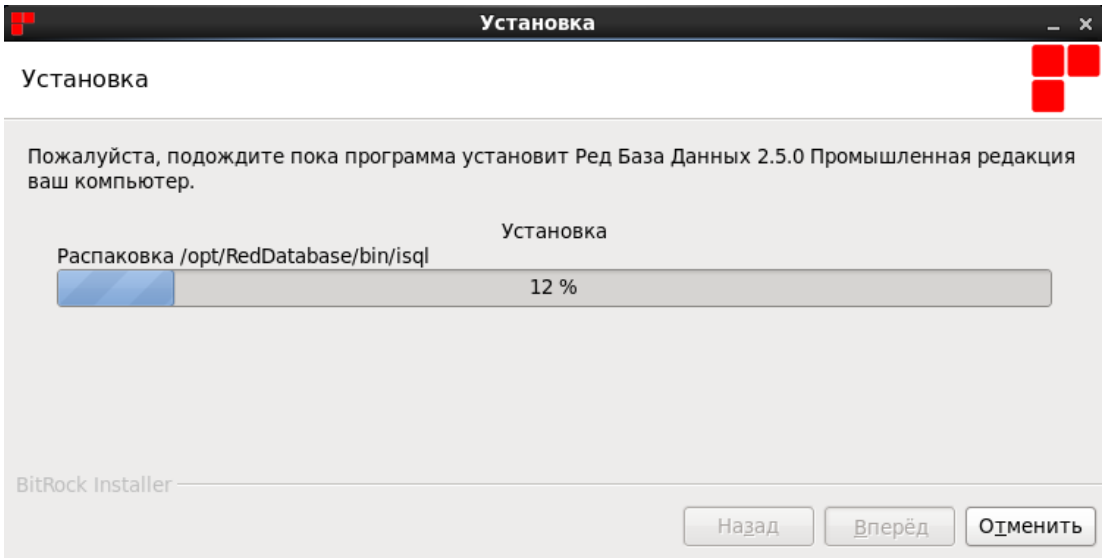


Рисунок 16 — Процесс установки

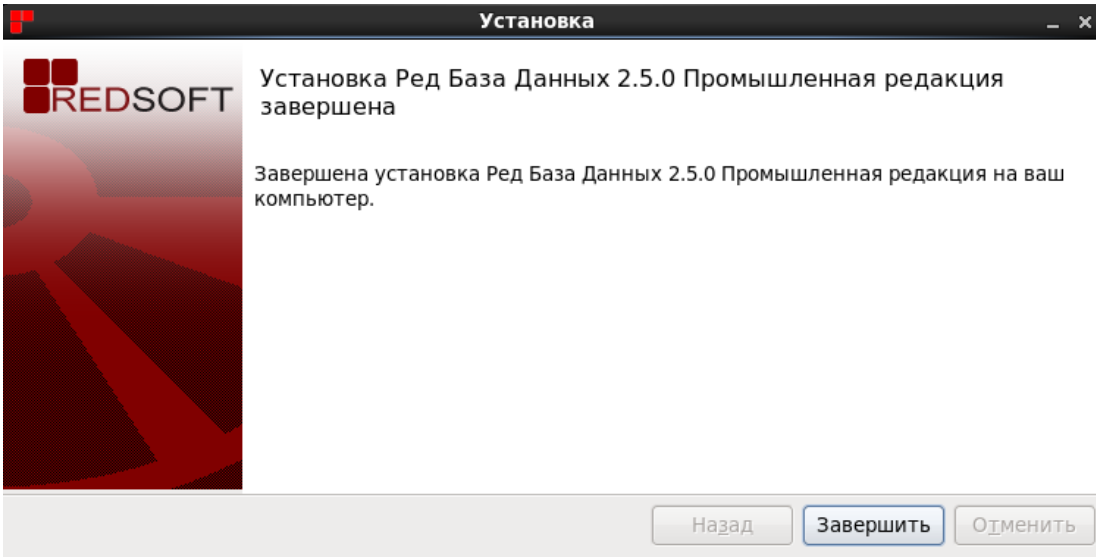


Рисунок 17 — Завершение установки

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 База Данных», после чего будет произведена деинсталляция сервера.

1.3.2 Установка в текстовом режиме

Если программа инсталляции запущена в текстовом 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).

```
Нажмите [Ввод] для продолжения :

Вы принимаете условия лицензионного соглашения? [y/n]: y

-------------------------------------------------------------------------------
Выберите компоненты, которые вы хотите установить. Нажмите Далее, когда будете
готовы продолжать установку.

SQL profiling support [Y/n] :

Debug information [y/N] : y

Red Database sources [y/N] : y

Red Replicator [Y/n] :

Red Database JDBC driver [Y/n] :

Red Database C++ driver [Y/n] :

Red Database ODBC driver [Y/n] :n

Red Database Python driver (KInterbasDB) [y/N] : y
```

Рисунок 13 — Пример установки «Ред База Данных» 2.5 в текстовом режиме

Удаление сервера осуществляется, по аналогии с графическим режимом, запуском программы uninstall из каталога установки «Ред База Данных».

1.3.3 Сборка сервера «Ред База Данных» 2.5 из исходных файлов

В состав «Ред База Данных» 2.5 входят исходные файлы сервера СУБД «Ред База Данных». Если при выборе компонентов установить флажок «Red Database sources» (рисунок 11), то после установки появляется возможность собрать сервер заново. Исходные коды будут скопированы в каталог /opt/RedDatabase/sources/. При сборке сервера вручную можно изменить каталог установки сервера (по умолчанию /usr/local/firebird), собрать отладочную («дебаговую») сборку и т.д. Во время сборки могут возникнуть ошибки, если в системе не установлены необходимые пакеты или их версии отличаются от требуемых. В этом случае сервер не будет установлен.

Ручная сборка и компиляция сервера «Ред База Данных» запускается скриптом autogen.sh. Если запустить скрипт с параметром —help, то будет выдана справка о доступных опциях сборки, в частности — архитектура сервера, путь для установки сервера и т. д.

Пример: ./autogen.sh --enable-superserver — будет собрана архитектура Супер. Если не задавать никаких опций, то будет произведена сборка по умолчанию (архитектура — Классик, папка установки /usr/local/firebird).

Дальнейшая компиляция осуществляется аналогично компиляции большинства Unix-приложений — вызовом команд make и make install. Для выполнения make install необходимы права суперпользователя.

Если устанавливается архитектура Классик, то будет настроен сервис xinetd, управляющий запуском процессов сервера. При установке архитектур Супер и Суперклассик будет запущен демон сервера. Кроме того, в ходе установки будет создан пользователь с именем firebird.

Внимание! Для работы архитектуры Классик необходимо, чтобы сервис xinetd был установлен и запущен.

2 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 = 2048 #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).

Внимание! Значение параметров, определяющих объем памяти, указываются в байтах. Для удобства таблица преобразования приводится в конце файла конфигурации.

2.1 Общие настройки

RootDirectory

Параметр RootDirectory указывает корневой каталог инсталляции сервера «Ред База Данных». Существует несколько способов сообщить серверу значение этого атрибута. Ниже приведено перечисление способов в том порядке, в котором сервер будет искать значение корневого каталога:

1. на любой платформе сервер сначала ищет переменную окружения FIREBIRD;
2. если переменная FIREBIRD не установлена, то для платформ Windows сервер отыскивает ключ в реестре;
3. если корневой каталог еще не найден, то предполагается, что она располагается на один уровень выше каталога запущенного процесса;
4. теперь процедура запуска ищет значение параметра RootDirectory в файле конфигурации.

Параметр имеет строковый тип. По умолчанию значение параметра равно пустой строке.

Пример:

```
RootDirectory = c:\RedDatabase
```

ServerName

Параметр ServerName может быть использован для того, чтобы задать имя сервера, отличное от имени сервера по умолчанию. Параметр имеет строковый тип. По умолчанию имя сервера равно Red Database 2.5.

Пример:

```
ServerName = Red Database 2.5
```

DatabaseAccess

Параметр DatabaseAccess позволяет обеспечить более четкое 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 каталога RootDirectory. В качестве разделителя директорий используется символ «;».

Пример:

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

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

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 каталога RootDirectory. В качестве разделителя директорий используется символ «;».

Пример:

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

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

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 Access 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 каталога RootDirectory. В качестве разделителя директорий используется символ «;».

Пример:

```
UdfAccess = Restrict UDF
```

Внимание! Неконтролируемая возможность использования внешних функций может быть использована для того, чтобы поставить под угрозу безопасность как баз данных, так и всей системы. Поэтому настоятельно рекомендуется использовать данный параметр для ограничения директорий размещения udf-библиотек.

TempDirectories

С помощью параметра TempDirectories можно задать временный каталог сервера «Ред База Данных». Временный каталог необходим для выгрузки данных во время сортировки, когда исчерпывается выделенная оперативная память.

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. В этом случае выгрузка временных данных при сортировке будет осуществляться в указанные каталоги. Для папок допускаются как абсолютные, так и относительные пути. Относительные пути берутся от каталога RootDirectory. 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)

Пример:

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

LegacyHash

Параметр LegacyHash делает возможным работу от пользователей, созданных до выполнения скрипта misc/upgrade/security_database.sql в базе данных безопасности.

LegacyHash имеет логический тип. Значение по умолчанию – 1 (истина). В этом случае пользователи со старым хэшем пароля смогут пройти авторизацию в обновленной security2.fdb. При необходимости запретить возможность авторизации пользователей со старым хэшем следует выставить значения параметра равным 0 (ложь).

Пример:

```
LegacyHash = 1
```

Authentication

Значение параметра Authentication определяет, какой из методов аутентификации будет использоваться сервером. Параметр Authentication имеет строковый тип и может принимать одно или несколько значений:

- традиционная (native) аутентификация;
- доверительная (trusted) аутентификация;
- многофакторная (multifactor) аутентификация;
- смешанная (mixed) аутентификация.

Для использования одновременно нескольких методов аутентификации необходимо перечислить их, разделяя запятыми. Более подробно каждый из режимов аутентификации рассмотрен в п. 4.4.2

По умолчанию используется смешанная аутентификация – значение параметра mixed.

Пример:

```
Authentication = native,multifactor
```

DefaultDbCachePages

Параметр DefaultDbCachePages используется для настройки количества страниц одной базы данных, находящихся в кэш-памяти одновременно. Суперсервер использует единый кэш (2048 страниц) для всех подключений и автоматически увеличивает его при необходимости. Классик создает отдельный кэш (по умолчанию 75 страниц) для каждого соединения.

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

Параметр имеет целочисленный тип. Единица измерения – страница базы данных. По умолчанию параметр имеет значение 2048. Максимальное значение 2147483647 страниц. Минимальное значение параметра – 0. Если значение параметра равно нулю, то сервер не будет выполнять кэширование страниц данных.

Пример:

```
DefaultDbCachePages = 2048
```

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.

Пример:

```
DatabaseGrowthIncrement = 134217728
```

Внимание! Пространство под теневые копии баз данных не резервируется.

MaxFileSystemCache

Параметр MaxFileSystemCache устанавливает порог использования системного кэша сервером «Ред База Данных» для архитектуры Суперсервер. Значение параметра MaxFileSystemCache определяет максимально допустимое количество страниц, которые могут находиться в кэш-памяти одновременно. Системный кэш бу-

дет использоваться до тех пор, пока количество закэшированных страниц меньше, чем значение параметра MaxFileSystemCache.

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 MaxFileSystemCache равно 0, то сервер не будет использовать системный кэш.

Пример:

```
MaxFileSystemCache = 65536
```

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.

Пример:

```
TempBlockSize = 1048576
```

TempCacheLimit

Параметр TempCacheLimit используется для ограничения объема оперативной памяти, выделяемой на одно соединение, он определяет максимальный объем оперативной памяти, выделяемой для одного соединения.

Параметр имеет целочисленный тип. Значение по умолчанию равно 67108864 байт. Максимально допустимое значение (2^64) - 1 байт. Минимальное значение параметра равно 0.

Пример:

```
TempCacheLimit = 67108864
```

OldParameterOrdering

Параметр OldParameterOrdering используется для обратной совместимости с приложениями, основанными на Firebird 1 и InterBase. В Firebird 1.5 и выше исправлена ошибка, заключавшаяся в особом порядке возвращаемых параметров, в XSQLDA структуре. Ошибка существовала на протяжении долгого времени, что привело к тому, что в ряд приложений, драйверов и компонентов было интегрировано решение этой проблемы на клиентской стороне.

Параметр имеет логический тип. Значение по умолчанию равно 0 (ложь). Для обратной совместимости следует выставить значение параметра равным 1 (истина).

Пример:

```
OldParameterOrdering = 0
```

Внимание! Изменение параметра затронет все базы данных на сервере, поэтому возможно возникновение ошибок при работе с приложениями, в которые не интегрировано решение данной ошибки.

CompleteBooleanEvaluation

Параметр CompleteBooleanEvaluation устанавливает метод вычисления логических выражений (полный или сокращенный). Значение по умолчанию равно 0 (ложь) используется сокращенный метод вычисления логических выражений. Сокращенный метод вычисления логических выражений позволяет, для предикатов AND или OR, возвращать результат, как только получен промежуточный результат, кото-

рый не может быть затронут дальнейшими вычислениями. Использование сокращенного режима вычисления логических выражения позволяет повысить производительность сервера.

Возможны очень редкие случаи, когда в режиме сокращенного вычисления операция внутри предикатов OR или AND остается не вычисленной. В этом случае существует возможность, что это повлияет на конечный результат выражения.

Если у вас существует такая проблема, то вы можете использовать полное вычисление логических выражений, для этого следует установить значение параметра равным 1 (истина).

Пример:

```
CompleteBooleanEvaluation = 0
```

DeadlockTimeout

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

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

Внимание! Слишком большое значение параметра может ухудшить производительность системы.

Пример:

```
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.

Пример:

```
MaxUnflushedWrites = 100
```

Внимание! 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.

Пример:

```
MaxUnflushedWriteTime = 5
```

Внимание! 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.

Пример:

```
BugcheckAbort = 0
```

OldColumnNaming

Параметр OldColumnNaming позволяет пользователям использовать старый (до версии 1.5) способ именования полей в выражениях SELECT.

Изменение значения этого параметра затронет псевдонимы, сгенерированные при использовании функций CONCATENATION, CURRENT_DATE, CURRENT_TIME, CURRENT_TIMESTAMP и EXTRACT.

Параметр OldColumnNaming имеет логический тип. Значение по умолчанию равно 0. Установка значения параметра равным 1 (истина) отключит автоматическую генерацию псевдонимов полей. Такой режим именования полей рекомендуется использовать только для совместимости со старыми скриптами.

Пример:

```
OldColumnNaming = 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.

Пример:

```
RelaxedAliasChecking = 0
```

ConnectionTimeout

С помощью параметра ConnectionTimeout устанавливается ограничение на время 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.

Пример:

```
ConnectionTimeout = 180
```

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.

Пример:

```
DummyPacketInterval = 0
```

Внимание! 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, как это описано здесь: http://support.microsoft.com/default.aspx? kbid=296265.

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_*.`

В Windows обратитесь к инструкциям следующей статьи:

`http://support.microsoft.com/default.aspx? kbid=140325`

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.

Пример:

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

Внимание! 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.

Пример:

```
RemoteAuxPort = 0
```

Внимание! 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.

Пример:

```
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.

Пример:

```
TcpNoNagle = 1
```

RemoteBindAddress

Параметр RemoteBindAddress позволяет привязать входящие соединения к определенному сетевому интерфейсу. 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 IP адреса).

Пример:

```
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.

Пример:

```
LockMemSize = 1048576
```

LockSemCount

Параметр LockSemCount используется для определения числа семафоров для менеджера блокировок. В системах, не поддерживающих многопоточность, этот параметр определяет число доступных семафоров.

Параметр LockSemCount имеет целочисленный тип. Значение по умолчанию равно 32.

Пример:

```
LockSemCount = 32
```

LockGrantOrder

Параметр LockGrantOrder определяет порядок получения блокировок. Если соединению требуется заблокировать объект, то оно получает блокировку, которая определяет блокируемый объект и требуемый уровень его блокировки. Каждому заблокированному объекту сопоставлена блокировка.

Параметр имеет логический тип. Значение параметра по умолчанию равно 1 (истина). В этом случае блокировки выдаются сервером в порядке очереди. Если выставить значение параметра равным 0 (ложь), то блокировки будут выдаваться сразу, после того как станут доступны. Это может привести к зависанию запроса блокировки.

Пример:

```
LockGrantOrder = 1
```

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.

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

Пример:

```
LockAcquireSpins = 0
```

LockHashSlots

Параметр LockHashSlots используется для настройки числа слотов кэширования блокировок. Чем больше слотов используется, тем короче получаются цепочки кэширования.

Параметр имеет целочисленный тип. По умолчанию значение параметра равно 1009. Увеличение параметра необходимо только при высокой загрузке. В качестве значения рекомендуется указывать простое число.

Пример:

```
LockHashSlots = 1009
```

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.

Пример:

```
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.

Пример:

```
OptimizationStrategy = default
```

2.2 Настройки только архитектуры Супер

CpuAffinityMask

Параметр CpuAffinityMask позволяет указать, какие процессоры будут ис-

пользоваться сервером. Параметр имеет эффект только в SMP (симметричных мультипроцессорных) системах, и архитектуре супер сервер.

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 1.

Пример:

```
CpuAffinityMask = 1
```

UsePriorityScheduler

Параметр UsePriorityScheduler имеет эффект только в системах Windows. Он позволяет управлять настройками планировщика потоков Windows.

Параметр имеет логический тип. Значение по умолчанию равно 1 (истина). Это означает, что планировщик потоков используется. Если возникает проблема со временем ответа компьютера при запуске сервера «Ред База Данных» следует отключить планировщик потоков. Для этого следует выставить значение параметра UsePriorityScheduler равным 0 (ложь).

Пример:

```
UsePriorityScheduler = 1
```

PrioritySwitchDelay

Параметр UsePriorityScheduler имеет эффект только в системах Windows. Он позволяет управлять настройками планировщика потоков Windows. Значение параметра определяет интервал времени, который должен пройти прежде, чем приоритет неактивного потока будет уменьшен до LOW, или приоритет активного потока будет увеличен до HIGH.

Параметр имеет целочисленный тип и измеряется в секундах. По умолчанию значение параметра равно 100 секунд. Минимально допустимое значение равно 0. Максимальное значение параметра равно 2147483647.

Пример:

```
PrioritySwitchDelay = 100
```

PriorityBoost

Параметр PriorityBoost используется для настройки механизма поднятия приоритетов в Windows. Значение параметра определяет количество дополнительных интервалов, выделяемых высокоприоритетному потоку.

Параметр имеет целочисленный тип. Значение по умолчанию равно 5. Минимально допустимое значение равно 0. Максимальное значение параметра равно 2147483647.

Пример:

```
PriorityBoost = 5
```

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. Пример:

```
GCPolicy = combined
```

2.3 Настройки только архитектуры Классик

2.3.1 *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).

Пример:

```
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.

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

Пример:

```
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.

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

Пример:

```
IpcName = FIREBIRD
```

Внимание! 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.

Пример:

```
RemotePipeName = interbas
```

CreateInternalWindow

Параметр CreateInternalWindow определяет, должен ли сервер создать невидимое окно, используемое для IPC (межпроцессная коммуникация).

Параметр имеет логический тип. Значение по умолчанию рано 1 (истина). В этом случае возможен запуск нескольких копий сервера на одной машине под управлением Windows. Если значение параметра выставить равным 0 (ложь), то сервер будет запускаться без скрытого окна и, следовательно, без поддержки локального протокола.

Пример:

```
CreateInternalWindow = 1
```

2.3.2 *Настройки для Unix/Linux систем*

LockSignal

Параметр LockSignal определяет сигнал, который сервер будет использовать в межпроцессных блокировках. Актуально только для архитектуры сервера Классик.

Параметр имеет целочисленный тип. Значение по умолчанию равно 16. Минимальное допустимое значение равно 0. Максимальное значение параметра равно 2147483647.

Пример:

```
LockSignal = 16
```

RemoteFileOpenAbility

Этот параметр позволяет отключить важную опцию безопасности сервера «Ред База Данных» и может вызвать неисправимое повреждение базы данных. Не используйте эту опцию, если вы не понимаете рисков потери данных.

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).

Внимание! 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

Пример:

```
RemoteFileOpenAbility = 0
```

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 корпораци-

ей 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/databas
e.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).

Пример:

```
Redirection = 0
```

2.4 Настройки безопасности

LoginFailureDelay

Задаёт время в секундах, на которое задерживается подключение к БД безопасности security2.fdb при четырёх неверных вводах пароля для конкретного пользователя или при 16 неверных вводах паролей для различных пользователей. Счётчик неверных вводов пароля отключается, если вводится правильный пароль или если не было попыток ввода пароля за указанный в данном параметре промежуток времени.

Пример:

```
LoginFailureDelay = 8
```

BlobAccessRights

Указывает, нужно ли контролировать доступ к BLOB-полям. Если этот параметр включен (по умолчанию), то доступ к BLOB (чтение, запись) через его дескриптор можно получить только в той же транзакции, которая открыла этот BLOB. Такое ограничение нужно потому, что права доступа к BLOB проверяются только при его открытии.

Пример:

```
BlobAccessRights = 1
```

CryptoPluginName

Параметр CryptoPluginName определяет имя криптоплагина, который будет использоваться сервером. Параметр имеет строковый тип. По умолчанию используется криптоплагин wincrypt_plugin. Для того, чтобы подключить криптоплагин, следует в качестве значения параметра указать имя библиотеки, размещенной в каталоге plugins, относительно корневой директории сервера. На текущий момент доступен один криптоплагин – wincrypt_plugin.

Пример:

```
CryptoPluginName = wincrypt_plugin
```

SymmetricMethod

Задаёт название или идентификатор алгоритма симметричного шифрования,

используемого сервером. Должен поддерживаться криптоплагином. Если задаётся именем алгоритма с национальными (русскими) символами, они должны быть указаны в URL-encoding. По умолчанию используется алгоритм шифрования, заданный в ГОСТ 28147-89.

Пример:

```
SymmetricMethod = 26142
```

HashMethod

Задаёт название или идентификатор алгоритма хеширования, используемого сервером. Должен поддерживаться криптоплагином. Если задаётся именем алгоритма с национальными (русскими) символами, они должны быть указаны в URL-encoding. По умолчанию используется алгоритм хеширования, заданный в ГОСТ Р 34.11-94.

Пример:

```
HashMethod = 32798
```

ProviderName

Задаёт название или идентификатор используемого сервером криптопровайдера. Должен поддерживаться криптоплагином. Если задаётся именем алгоритма с национальными (русскими) символами, они должны быть указаны в URL-encoding. По умолчанию используется провайдер «GOST R 34.10-2001 Signature with Diffie-Hellman Key Exchange».

Пример:

```
ProviderName = 75
```

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.

Внимание! Алгоритмы хеширования зависят от используемого криптоплагина, подробнее см. пункт 4.10

Пример

```
HashesFile = hashes
```

KeyRepository

Параметр KeyRepository определяет имя контейнера, в котором хранится ключевая пара для сервера. Параметр имеет строковый тип. По умолчанию значение параметра равно пустой строке.

Внимание! Значение параметра KeyRepository зависит от параметра CryptoPluginName. Если криптоплагин не подгружается сервером, то данный параметр не будет обрабатываться.

Внимание! Владельцем контейнера должен быть пользователь, от имени которого запускается сервер.

Пример:

```
KeyRepository = srv_kr
```

PrivateKeyPin

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

Пример:

```
PrivateKeyPin = mypass
```

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.

Пример:

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

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.

Пример:

```
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.

Пример:

```
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.

Пример:

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

VerifyCertChain

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.

Пример:

```
VerifyCertChain = 1
```

MemoryWipePasses

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

Пример:

```
MemoryWipePasses = 0
```

UseRecordFilter

Параметр UseRecordFilter используется для включения режима фильтрации записей. Параметр имеет логический тип, по умолчанию фильтрация записей отключена, значение параметра равно 0 (ложь). Чтобы включить фильтрацию записей следует выставить значение параметра равным 1 (истина). Подробнее о режиме фильтрации записей см. пункт 4.8.1.

Пример:

```
UseRecordFilter = 0
```

InitScript

Параметр InitScript определяет имена скриптов инициализации, которые будет выполняться при создании новой базы данных.

Параметр имеет строковый тип, значение по умолчанию равно пустой строке. В этом случае инициализирующий скрипт не проливается при создании новой БД. В качестве значения параметра следует указывать полные пути к sql-файлам (подробнее см. п. 4.8.1).

Для использования функционала полнотекстового поиска и фильтрации записей следует установить значение параметра равным init.sql.

Пример:

```
InitScript = init.sql
```

Внимание! Если указан относительный путь, то он берется от корневой директории сервера.

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.

Пример:

```
LDAPServer = 192.168.1.1
```

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.

Пример:

```
LDAPEncryption = None
```

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 Базы Данных.

Пример:

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

LDAPPassword

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

Пример:

```
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.

Пример:

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

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.

Пример:

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

LDAPMembershipFilter

Фильтр, который будет использован при поиске в LDAP групп, к которым принадлежит пользователь. Существует две основные схемы, используемые в LDAP для указания членства в группах. В соответствии с используемой схемой нужно формировать фильтр. Если указано пустое значение, принадлежность пользователя к группам не определяется.

Примеры:

```
LDAPMembershipFilter = memberUid=%u
LDAPMembershipFilter = member=uid=
%u,ou=people,dc=example,dc=com
```

LDAPUserCertificate

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

Пример:

```
LDAPUserCertificate = userCertificate;binary
```

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 — традиционный пароль пользователя в Ред Базе Данных;

rdbSecurePassword — защищённый пароль пользователя в Ред Базе Данных;

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.

Пример:

```
LDAPPasswordSync = rdbPassword;userPassword
```

TraceAuthentication

Включает/отключает запись отладочных сообщений о процессе многофакторной аутентификации в firebird.log.

Пример:

```
TraceAuthentication = 0
```

2.5 Настройки репликации

Репликация предназначена для обеспечения повышенной отказоустойчивости в случае повреждения физической структуры файла базы данных, вызванной либо техническим сбоем оборудования, либо программным сбоем операционной системы или самой СУБД. 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.

Резервные базы данных переключаются в специальный режим с помощью утилиты GFIX. Этот режим не позволяет подключаться к резервной базе данных с помощью обычных соединений и работать с ней как с основной. Узнать, в каком режиме находится база данных, можно с помощью утилиты gstat -h. Узнать, работает ли репликация на основном сервере, можно в виртуальной таблице MON$DATABASE. Также в таблице MON$ATTACHMENTS можно узнать, сколько было передано пакетов на резервные сервера, и сколько по времени основной сервер был в состоянии ожидания ответа о подтверждении выполнения операций синхронизации.

Однозначное соответствие строк на основном и резервных серверах определяется первичным ключом. Реплицируются только те таблицы, которые содержат первичный ключ. В случае отсутствия или неактивности первичного ключа в зависимости от настроек допускается либо остановка репликации, либо игнорирование изменений в данной таблице в целях репликации.

Для настройки системы репликации используется файл replication.conf, находящийся в корневом каталоге «Ред База Данных». 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; 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 корневом каталоге «Ред База Данных» и получает имя replication.log.

Файл журнала создается только в случае, когда возникает ошибка или предупреждение.

Файл конфигурации разбит на блоки:

- <database> - основной блок с параметрами по умолчанию;
- <database [path_to_database] > - блок с параметрами для базы, требующей репликации.

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

В файле конфигурации присутствуют параметры трех типов:

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

delay_buffer_size

Указывает размер накопляемого буфера операций на мастер-базе перед отправкой их на слейв-базы. 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.

Пример:

```
delay_buffer_size 1048576 # 1MB
```

detach_slave_on_error

Если установлен в true, то при возникновении ошибки, связанной с репликацией, мастер отключается от слейва и продолжает работать без репликации.

Пример:

```
detach_slave_on_error true
```

compress_records

Если установлен в true, то перед отправкой записи будут сжиматься алгоритмом RLE. Необходим для сжатия трафика, передаваемого по сети.

Пример:

```
compress_records true
```

master_priority

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.

Пример:

```
compress_records true
```

include_filter

Строковый параметр в формате регулярного выражения в синтаксисе SQL, который задает, какие сущности базы данных необходимо включить в репликацию. По умолчанию реплицируются все сущности.

Пример:

```
include_filter '(W$|ORD$)%'
```

exclude_filter

Строковый параметр в формате регулярного выражения в синтаксисе SQL, который задает, какие сущности базы данных необходимо исключить из репликации. По умолчанию реплицируются все сущности.

Пример:

```
exclude_filter '(TEMP$|LOG$)%|(SYS_DB_LOG)'
```

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.

Пример:

```
alert_command crash_replication.bat
```

3 Утилиты командной строки

3.1 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;).

Запуск утилиты производится следующим образом:

```
isql [-u[ser] <user_name>] [-p[assword] <password>]
[database_name]
```

где:

- `user_name` – имя пользователя «Ред Базы Данных»;
- `password` – пароль пользователя;
- `database_name` – спецификация базы данных в формате `<адрес сервера>:<путь к БД >`

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 2). Присоединиться к уже существующей БД можно непосредственно при запуске утилиты, указав имя базы, и, если необходимо, имя пользователя и пароль[1].

Например:

```
isql 127.0.0.1:c:\temp\base.fdb -user sysdba -password
masterkey;
```

Здесь параметры -user и -password задают имя пользователя и пароль для соединения с БД.

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

```
SET TERM <symbol>
```

где symbol может быть как одним, так и группой символов.

ISQL может использоваться для выполнения операций трех типов:

1 Имя пользователя и пароль можно не указывать если в системе установлены контекстные переменные ISC_USER и ISC_PASSWORD, а также в том случае, если включен режим доверительной (trusted) аутентификации, и пользователь системы, от имени которого запускается утилита ISQL, является членом группы администраторов. Эти опции также действуют для всех описываемых ниже утилит.

- 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:

```
SHOW <object|ALL> [<name>|<mask>]
```

Ключевое слово ALL будет соответствовать всем типам объектов. Задание имени объекта <name> 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.

Пример:

show tables tab%

- покажет все таблицы, начинающиеся с tab. Команда show all – покажет все метаданные.

Таблица 3.1 - Опции ISQL

Опция	Описание
-a(ll)	Извлечение всех метаданных, включая не-SQL объекты (например внешние функции). Используется совместно с командой extract
-c(ache) <num>	Задать число страниц, которые будут кэшироваться при соединении с БД
-ch(arset) <charset>	Задать набор кодировку для текущего соединения
-d(atabase) <database>	Задать имя и путь к БД, которое будет записано в выходной поток
-e(cho)	Включает или подавляет дублирование команд на указанное устройство вывода (монитор, в файл, и т. д.)
-ex(tract)	Извлечь метаданные
-i(nput) <file>	Задать файл с SQL-запросами для выполнения.
-m(erge)	Перенаправление ошибок на поток стандартного вывода
-n(oautocommit)	Отключить автоматическое подтверждение DDL-операций
-nod(btriggers)	Не запускать триггеры базы данных
-now(arnings)	Не показывать предупреждения
-o(utput) <file>	Задать файл для вывода результата выполнения запросов. Без аргументов перенаправляет вывод на стандартное устройство вывода (монитор)
-pag(elength) <size>	Размер страницы
-p(assword) <password>	Пароль пользователя
-q(uiet)	Не показывать сообщение "Use CONNECT..."
-r(ole) <role>	Имя роли
-s(qldialect) <dialect>	SQL диалект (set sql dialect)
-t(erminator) <term>	Команда терминатора (разделитель строк)

Опция	Описание
-tr(usted)	Использовать доверительную аутентификацию
-mf	Использовать многофакторную аутентификацию
-u(ser) <user>	Имя пользователя
-x	Извлечение метаданных
-z	Показать версии утилиты и сервера
-ce(rtificate) <alias>	Использовать сертификат для проверки подлинности пользователя при многофакторной аутентификации
-pi(n) <PIN>	Задать PIN (пароль), если он необходим для получения закрытого ключа сертификата пользователя
-v(erify)	Проверить сертификат сервера при многофакторной аутентификации

3.2 Утилита GBAK

Наиболее универсальным инструментом, позволяющим осуществить резервное копирование базы данных на любой платформе, является gbak - утилита командной строки, входящая в поставку «Ред База Данных». С помощью gbak можно обратиться к серверу и произвести считывание данных и получение на их основе резервной копии, а также восстановить базу данных из резервной копии.

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.

Для того, чтобы создать резервную копию базы данных, необходимо выполнить следующую команду:

```
gbak [-B] [options] <база_данных-источник> <файл
резервной копии>
```

Ключ -B означает, что необходимо выполнить резервное копирование базы данных, путь к которой указан как <база_данных-источник>, а результаты резервного копирования сохранить в файл, указанный как <файл резервной копии>.

Набор дополнительных опций, представлен ниже.

Таблица 3.2 - Опции GBAK

Опция	Описание
-b(ackup_database)	Осуществляет резервное копирование базы данных
-co(nvert)	Преобразование внешних файлов во внутренние таблицы
-e(xpand)	Не производить сжатие резервной копии
-fa(ctor)	Использовать блокирующий фактор n
-g(arbage_collect)	Не собирать мусор во время резервного копирования[2]
-ig(nore)	Игнорировать контрольные суммы[3]
-l(imbo)	Игнорировать зависшие двухфазные транзакции (limbo)
-m(eta_data)	Производит резервное копирование только метаданных
-nt	Создаёт резервную копию в нетранспортабельном формате
-nod(btriggers)	Не запускать триггеры уровня базы данных
-ol(d_descriptions)	Производит резервное копирование метаданных в формате старого стиля, т. е. в режиме совместимости со старыми базами данных
-pas(sword)	Пароль пользователя, подключающегося к базе данных для резервного копирования
-ro(le)	Подсоединиться с использованием роли

2 «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.

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

Опция	Описание
-se(rvice)	Создаёт резервную копию на том же компьютере, где находится база данных-источник. Для этого вызывается Service Manager на компьютере-сервере.
-t(ransportable)	Создаёт транспортабельную (переносимую) резервную копию - этот параметр включен по умолчанию
-user	Имя пользователя, который подключается к базе данных для резервного копирования
-skip_d(ata) <pattern>	Пропускать данные таблиц, имена которых соответствуют указанному здесь регулярному выражению в SQL-синтаксисе (см. оператор SIMILAR TO) [4]
-z	Показать версию gbak и версию сервера «Ред База Данных»
-bu(ffers)	Задать размер кэша для БД (количество кэшируемых страниц)
-c(reate_database)	Восстановить БД из резервной копии
-i(nactive)	Деактивировать индексы во время восстановления
-k(ill)	Восстановить без создания теневых копий
-mo(de) <ACCESS>	Режим доступа "read_only" или "read_write"
-o(ne_at_a_time)	Подтверждать после восстановления каждой таблицы
-p(age_size)	Установить размер страницы
-r(ecreate_database) (o(verwrite))	Восстановить БД или восстановить ее поверх старой (если используется параметр overwrite)
-rep(lace_database)	Заменить базу данных из резервной копии
-tru(sted)	Использовать доверительную аутентификацию
-use(_all_space)	Не резервировать место под версии записей

3.3 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.

Nbackup позволяет блокировать файл базы данных. Таким образом, Вы после этого сможете сами создавать обычные копии или резервные копии с помощью утилит по Вашему выбору.

Таблица 3.3 - Опции NBACKUP

Опция	Описание
-L <database>	Блокировка базы данных для копирования
-N <database>	Разблокировка ранее заблокированной базы данных[5]
-F <database>	Разблокировка ранее заблокированной базы данных[6]
-B <level>|<GUID> <database> [<filename>]	Создает инкрементную копию[7]
-R <database> [<file0> [<file1>...]]	Восстанавливает базу данных
-U <user>	Имя пользователя
-P <password>	Пароль пользователя
-FE <file>	Считывает пароль из файла
-T	Не запускать триггеры базы данных
-S	Вывести количество страниц в базе после блокировки БД

4 Внешние ключи, ссылающиеся на эти таблицы, будут деактивированы

5 При использовании параметра `-N` 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.

6 При использовании параметра `-F` только изменяется в «нормальное» значение флага состояния самостоятельно восстановленной базы данных.

7 GUID базы данных можно узнать из вывода утилиты gstat с параметром -h

-D [ON | OFF]	Включает/выключает небуферизованный ввод/вывод при чтении БД
-I	Восстанавливает инкрементную копию в существующую базу данных [8]
-Z	Печатает версию СУБД

Внимание! 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".

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:

```
nbackup [-U <пользователь> -P <пароль>] -B 0
<база_данных> [<резервный_файл>]
```

Например:

```
nbackup -user sysdba -password masterkey -b 0 base_01.fdb
base_01_10_04_08.nbk
```

Параметр -B означает создание резервной копии. 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 aliases.conf).

Резервная копия всей базы данных восстанавливается следующим образом:

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

Например:

```
nbackup -user sysdba -password masterkey -r base_01.fdb
base_01_10_04_08.nbk
```

Уровень не указывается при восстановлении.

Внимание! 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.

3.3.1 Инкрементные резервные копии

Для создания инкрементной резервной копии необходимо указать уровень резервной копии больше 0. 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 -user sysdba -password masterkey -B 1 base.fdb
base_5-Mar-2008.nbk
```

Эта резервная копия будет содержать только изменения базы данных за последний день. Если через день вновь создать резервную копию уровня 1:

```
nbackup -user sysdba -password masterkey -B 1 base.fdb
```

8 Работает только с опцией -R

```
base_6-Mar-2008.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:

```
nbackup -user sysdba -password masterkey -B 2 base.fdb
base_6-Mar-2008_2.nbk
```

Эта резервная копия будет содержать изменения только с момента создания последней резервной копии уровня 1.

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>]-R <база_данных>
[<резервная_копия0> [<резервная_копия1> [...] ] ]
```

Таким образом, восстановление для предыдущего примера до уровня 2 будет выглядеть так:

```
nbackup -user sysdba -password masterkey -R base.fdb
base_5-Mar-2008.nbk base_6-Mar-2008.nbk base_6-Mar-
2008_2.nbk
```

Внимание! 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.

Внимание! Не существует формального ограничения на уровень резервной копии, однако на практике редко имеет смысл создавать копии уровней больше 3 или 4.

3.3.2 Блокировка базы данных и самостоятельное резервное копирование

Типичными действиями являются следующие:

1. Блокировать базу данных с помощью параметра -L (Lock):

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

2. Создать резервную копию, сжать файл базы данных, используя любые другие программы. Простое копирование файла также допустимо.
3. Разблокировать базу данных с помощью параметра -N (uNlock):

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

3.4 Утилита GFIX

Утилита GFIX предназначена для проверки базы данных и восстановления повреждённых баз данных. Помимо этого, GFIX также может выполнять различные действия по управлению базой данных: менять диалект базы данных, устанавливать и снимать режим работы "только чтение", устанавливать размер кеша для конкретной базы данных и др. Запуск GFIX осуществляется следующим образом:

```
gfix <операция> [<опции>] <db_name>
```

- <операция> - операция, которую необходимо выполнить над базой данных;
- <опции> - набор опций и параметров для выполнения операции;
- <db_name> - имя базы данных, над которой будут производиться операции[9].

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. пункт 4.3.6). Для указания имени и пароля пользователя используются переключатели -user и -pa[ssword], соответственно, например:

```
gfix -user SYSDBA -pa masterkey [опции] <имя базы>
```

Также в ОС Windows возможно использование режима доверительной аутентификации — с использованием переключателя -trusted (если доверительная аутентификация разрешена в настройках сервера «Ред База Данных»). В этом случае указание имени пользователя и пароля необязательно.

3.4.1 Активация теневой (оперативной копии)

Для активации теневой копии используется команда -ac[tivate]:

```
gfix -activate <db_name>
```

Удаление теневых копий производиться с помощью команды -k[ill]:

```
gfix -kill <db_name>
```

Внимание! 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. пункт 4.3.4.

3.4.2 Закрытие (блокировка) базы данных

Закрытие базы данных означает, что база данных переводится в особый режим, в котором к ней запрещены все подключения, кроме администратора сервера — пользователя SYSDBA. 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 до полной блокировки:

```
gfix -sh[ut] {-at[tach] n | -tr[an] n | -f[orce] n}
```

Режим -at[tach] n означает, что все новые соединения к базе данных запрещены. Существующие соединения при этом не разрываются. Число n 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 -tr[an] n сервер заблокирует запуск новых транзакций в закрываемой базе данных. Если по истечении n секунд к базе не останется ни одного активного подключения, то база данных будет закрыта. В противном случае за-

9 Если база данных состоит из нескольких файлов, то при запуске gfix необходимо указать первичный файл.

крытие БД будет отменено.

В режим -f[orce] n сервер будет ждать завершения всех активных соединений к базе данных. По истечении времени n 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 -on[line]
```

3.4.3 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 старейшей заинтересованной транзакцией (заинтересованной называется транзакция, которая находится в любом состоянии, кроме подтвержденного) и старейшей **snapshot-**транзакцией. По умолчанию этот интервал равен 20000 транзакций. Изменить этот параметр для конкретной базы данных можно с помощью опции -h[ouskeeping]:

```
gfix -h[ouskeeping] n <db_name>
```

Здесь n — разница между старейшей заинтересованной транзакцией и старейшей snapshot транзакцией;

db_name — спецификация базы данных (адрес сервера, путь и имя БД).

Если выставить значение n равным 0, то в этом случае автоматическая чистка будет отменена. Вручную чистку можно произвести с помощью команды -sweep:

```
gfix -sweep n <db_name>
```

3.4.4 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. 3.4.2.

Проверка базы данных выполняется с помощью команды -v[alidate]:

```
gfix -v[alidate]
```

По умолчанию при проверке базы данных gfix освобождает неиспользуемые

страницы в базе данных[10], а также обнаруживает разрушенные структуры данных. 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]
```

Для того, чтобы gfix не проверял ошибки контрольных сумм, используется опция -i[gnore][11] команды -v[alidate]:

```
gfix -v[alidate] -i[gnore]
```

Результаты работы утилиты gfix сохраняются в лог-файле сервера — firebird.log.

После проверки базы данных, если были найдены ошибки, можно попытаться исправить базу данных.

Если в базе данных остались зависшие транзакции, то, в первую очередь, необходимо разрешить все такие транзакции. Зависшие (in limbo) транзакции могут образоваться при использовании транзакций с двухфазным подтверждением (2PC). При этом, если возникнет какой-либо сбой, и связь с одной из баз данных будет потеряна до того, как произойдет вторая фаза подтверждения, транзакция станет «зависшей», сервер не сможет определить, должна ли она быть подтверждена или отменена. Для поиска и разрешения таких зависших транзакций с помощью gfix используется команда -l[ist]. Она выводит список всех зависших транзакций в базе данных. Команда может использоваться с ключом -p[rompt] — в этом случае gfix будет выдавать запрос о действиях с каждой обнаруженной зависшей транзакцией по очереди.

Поскольку серверу известно, что должно было произойти с каждой из подтранзакций во время второй фазы двухфазной транзакции (подтверждение или откат), сервер может автоматически разрешить зависшие двухфазные транзакции. Для этого используется опция -t[wo_phase] {ID | All}. Здесь можно указать идентификатор конкретной транзакции или разрешить все зависшие транзакции опцией All.

Аналогично можно завершить все зависшие транзакции подтверждением (или откатом) вне зависимости от того, чем они должны были завершиться изначально. Для этого используются опции -c[ommit] {ID|All} и -r[ollback] {ID|All}:

```
gfix -t[wo_phase] {ID|All}
gfix -c[ommit] {ID|All}
gfix -r[ollback] {ID|All}
```

Внимание! Так как при запуске gfix можно указать только одну пару имя пользователя/пароль, то при восстановлении зависших двухфазных транзакций, которые работали с базами данных на разных серверах, логин и пароль пользователя, выполняющего восстановление, должны совпадать на всех серверах.

3.4.5 Изменение установок БД

С помощью gfix можно изменить установки базы данных, такие как используемый диалект, количество кэшируемых страниц, режим записи (синхронный/асинхронный) и т.д. Команды для изменения настроек базы данных сведены в таблицу 3.4:

10 Страницы, которые были назначены какой-либо структуре данных но небыли использованы вследствие аппаратных или программных сбоев

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

Таблица 3.4 - Опции GFIX

Опция	Описание опции
-m[ode] {read_write|read_only}	Устанавливает режим записи для базы данных - только для чтения или чтение/запись. Этот параметр может принимать два значения
-buffers <n>	Устанавливает количество кэшируемых страниц
-w[rite] {sync|async}	Устанавливает режим записи (Forced Writes on/off)
-sql_dialect <n>	Установка диалекта базы данных
-use {reserve|full}	Включает или отключает использование свободного пространства на страницах с данными. По умолчанию сервер «Ред База Данных» заполняет страницы с данными не более чем на 80% (режим reserve). Включать режим заполнения на 100% (-use full) имеет смысл, например для баз данных, предназначенных только для чтения[12].

Примеры использования gfix:

```
gfix -w sync firstbase.fdb
```

В этом примере для базы данных firstbase.fdb устанавливается режим синхронной записи (FW ON).

```
gfix -v -full firstbase.fdb
```

В этом примере запускается проверка тестовой базы данных (опция -v), причем указывается, что необходимо проверить также фрагменты записей (-full).

3.4.6 Активация режима репликации

Для перевода базы данных в состояние резервной используется команда -re[plica] on:

```
gfix -replica on <db_name>
```

Возврат в обычную базу данных производится с помощью команды -re[plica] off:

```
gfix -replica off <db_name>
```

3.5 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.

Синтаксис gstat:

```
gstat [-U <пользователь> -P <пароль>] <переключатель>
<база_данных>
```

Таблица 3.5 - Опции GSTAT

Переключатель	Описание переключателя
-a	Статистика страниц данных и индексов
-d	Статистика страниц данных
-h	Статистика заголовочной страницы
-i	Статистика индексов
-s	Статистика системных таблиц
-u	Имя Пользователя
-p	Пароль пользователя
-t	Название таблицы
-z	Показать версию сервера и утилиты
-tr	Использовать доверительную аутентификацию

12 Изменять режим заполнения страниц с данными можно только если база находится в режиме read_write

3.6 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.

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 security2.fdb.

GSEC может быть запущена как в интерактивном, так и в командном режиме режиме, и может отображать подсказки с перечислением всех опций.

Синтаксис GSEC:

```
gsec [ <опции> ... ] <команда> [ <параметры> ... ]
```

Таблица 3.6 - Опции GSEC

Опция	Описание опции
-user <имя>	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
-mf	Использовать многофакторную аутентификацию
-certificate <алиас>	Сертификат проверки подлинности пользователя для многофакторной аутентификации
-pin <PIN>	Пароль для закрытого ключа сертификата пользователя
-v	Проверка сертификата сервера при подключении
-database <БД_безопасности>	Путь к файлу базы данных безопасности
-z	Отобразить версию GSEC и сервера «Ред Базы Данных»
-help	Отобразить помощь по всем опциям GSEC

Таблица 3.7 - Команды GSEC[13]

Команда	Описание команды
add <имя> [<параметр> ...]	Добавление нового пользователя
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
quit	Выход из интерактивного режима

Таблица 3.3 - 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

Пример запуска утилиты GSEC в интерактивном режиме:

```
gsec -user sysdba -password masterkey
```

13 В не интерактивном режиме команды вводятся с префиксом «-», например -add

Права на внесение изменений в базу данных безопасности имеет только sysdba или пользователь, которому назначена роль SECADMIN. Остальные пользователи имеют права только на изменение своей учетной записи.

Для выхода из интерактивного режима утилиты GSEC используется команда q[uit]:

```
GSEC> quit
```

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:C:\RedDatabase\security2.fdb
-user sysdba -password masterkey
```

4 Администрирование подсистемы безопасности

4.1 Основные термины и определения

Сервер баз данных (далее сервер) — установленная 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 security2.fdb, расположенная в корневой директории каталога установки сервера. В этой базе хранятся параметры пользователей системы, политики доступа, глобальные роли.

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.

Криптопровайдер — внешнее программное обеспечение, осуществляющее функции хеширования, шифрования, криптографической защиты.

Криптоплагин — библиотека, которая обеспечивает взаимодействие между криптопровайдером и сервером или утилитами сервера. Каждый криптоплагин предназначен для взаимодействия с определенным криптопровайдером.

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. Роли могут быть глобальными – будучи назначенными пользователям, действуют в пределах всего сервера, и локальными – действуют в пределах одной базы данных. Глобальные роли хранятся в базе данных безопасности security2.fdb. Локальные роли хранятся в той базе данных, к которой они относятся.

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.

Администратор безопасности — пользователь, которому назначена роль SECADMIN. Предназначен для управления пользователями. Может быть глобальным — в пределах всего сервера и локальным — в пределах определенной базы данных. Таких пользователей может быть несколько.

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

4.2 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 База Данных» являются:

- хранящиеся в «Ред База 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 подсистем (механизмов безопасности):

- идентификация и аутентификация;
- разграничение доступа;
- регистрация событий;
- очистка освобождаемых ресурсов;
- контроль целостности информационных объектов.

4.2.1 Подсистема идентификации и аутентификации

В «Ред База Данных» подсистема идентификации и аутентификации 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 аутентифика-

ции.

Первичным фактором для прохождения процедуры аутентификации является пароль пользователя. Параметры безопасности для каждого пользователя задаются с помощью политик безопасности.

В «Ред База Данных» реализованы политики безопасности учётных записей, которые позволяют задать:

- требования к сложности пароля;
- количество предыдущих паролей, которые не должен повторять вновь заданный;
- срок действия пароля;
- количество допустимых неудачных попыток аутентификации;
- требования к дополнительным факторам для прохождения аутентификации (цифровые сертификаты, биометрические данные).

«Ред База Данных» использует собственный алгоритм аутентификации RS CHAP на основе MS CHAP, который позволяет избежать передачи аутентификационных данных по каналу связи между клиентом и сервером «Ред База Данных» в открытом виде. Подробное описание алгоритма приведено в п. 4.4.1.

Все необходимые настройки подсистемы идентификации и аутентификации задаются администратором «Ред База Данных» в файле конфигурации сервера. Кроме того, требуемые факторы аутентификации и другие параметры политик определяются через политики безопасности для каждого конкретного субъекта безопасности (в данном случае — пользователя).

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 (телефон, адрес, email и т.д.), и на основе этих данных заполняются контекстные переменные на сервере БД.

4.2.2 Подсистема разграничения доступа

В «Ред База Данных» доступ субъектов безопасности к защищаемым объектам может быть разграничен двумя способами:

- пользователям могут быть назначены персональные права;
- пользователю 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 его права определяются совокупностью прав назначенных ролей (действует принцип кумулятивного действия ролей). Возможно также назначение одной роли другой.

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.

Внимание! Во время установки сервера или сразу после нее рекомендуется как можно быстрее сменить пароль по умолчанию пользователя SYSDBA.

Кроме того, существует ряд предопределённых ролей, предназначенных для выполнения функций поддержки и администрирования СУБД. Роли и их назначение приведены в следующей таблице:

Таблица 4.1 - Предопределенные роли

Имя роли	Назначение роли
RDB$ADMIN	Распределение прав пользователей, а также выполнение операций обслуживания баз данных (резервное копирование, восстановление и т. д.).
SECADMIN	Управление пользователями (создание, изменение,удаление) и политиками безопасности
PUBLIC	Роль по умолчанию для вновь создаваемых пользователей. Не имеет никаких прав. Не существует в базе данных в явном виде.

Существуют два вида ролей: локальные и глобальные. Локальные роли создаются администратором сервера или администратором безопасности. Они хранятся непосредственно в той базе данных, для которой они создаются. Для создания глобальных ролей необходимо напрямую соединиться с базой данных безопасности security2.fdb. Такое соединение может выполнить только администратор сервера (SYSDBA) или пользователь с ролью администратора безопасности (SECADMIN). После этого необходимо создать роль и дать ей права по аналогии с локальными ролями. Роль сохраняется в базе данных безопасности и, таким образом, автоматически станет глобальной. Глобальной роли нельзя добавить право на какую-либо операцию сразу для всех баз сервера[14] — в каждой базе данных такое назначение необходимо производить заново, так как права ролей хранятся непосредственно в каждой базе. Основное же назначение глобальных ролей - выполнение операций с системными сервисами (см. п. 4.3.6).

Управление ролями производится администратором безопасности (пользователь с ролью SECADMIN) или пользователем SYSDBA.

Подробнее о назначении прав на конкретные операции см. п. 4.3

Используя набор доступных средств администрирования, администратор безопасности может управлять правами доступа до уровня полей записей в таблицах БД. Доступом к хранимой в БД информации можно также управлять с использованием различных представлений, давая пользователю права на соответствующие представления.

Непротиворечивость прав доступа обеспечивается тем, что они носят разрешительный характер, и суммарные права пользователя определяются как объединение прав пользователя, полученных им путём назначения ему различных ролей.

Неконтролируемого распространения прав не происходит, поскольку они назначаются только администратором сервера или администратором безопасности. Кроме того, права на какую-либо операцию может делегировать в пределах той же базы данных пользователь, получивший это право с опцией WITH GRANT OPTION.

4.2.3 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 пользова-

14 Исключение составляет роль RDB$ADMIN, для которой такие права предопределены (см. таблицу 4.3)

телей (GSEC);

- получение свойств БД, анализ и восстановление поврежденной БД (GFIX);
- получение статистики БД (GSTAT);

Право на запуск сервсисов в общем случае является глобальным для сервера, поэтому права на эти операции хранятся в базе данных безопасности security2.fdb. Назначить право на запуск сервиса можно только пользователям и глобальным ролям.

4.2.4 Подсистема регистрации событий (аудит)

Настройка регистрации событий в «Ред База Данных» производится с помощью изменения параметров в конфигурационном файле подсистемы регистрации и учета – fbtrace.conf. Задаются следующие параметры:

- включение/отключение регистрации событий;
- формат журнала (текстовый, бинарный);
- 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. По умолчанию используется текстовый формат. Подробнее о типах регистрируемых событий и параметрах, сохраняемых для каждого типа события, см. п. 4.6.

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. 4.7). 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.

4.2.5 Подсистема очистки 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.

4.2.6 Подсистема контроля целостности

Подсистема контроля целостности программных и информационных ресурсов встроенной системы защиты в «Ред База Данных» реализована на основе расчёта и последующей проверки хеш значений контролируемых объектов.

При сборке дистрибутива, после того, как сформированы все компоненты системы, запускается утилита формирования хешей. Она хеширует файлы, указанные в параметрах, и помещает результат в файл хешей.

При инсталляции и в процессе эксплуатации администратор может модифицировать список файлов, подлежащих контролю, и сгенерировать для них файл хешей-эталонов (например, добавить к контролируемым файлам файл конфигурации). Для этого используется утилита hashgen, входящая в состав дистрибутива «Ред База Данных» подробнее см. п. 4.10

При старте сервера, после того, как инициализируется криптопровайдер, производится считывание содержимого файла с хешами. Далее, для каждого подконтрольного файла генерируется хеш с определенным для него алгоритмом и сверяется с эталонным значением хеша. При несоответствии сгенерированного и хранимого хешей в журнал заносится запись с именем файла, имеющего неверную контрольную сумму и сервер прекращает работу.

Включение режима контроля целостности файлов сервера выполняется путем задания имени файла с хешами в конфигурационном файле «Ред База Данных» (параметр HashesFile)

Кроме того, аналогично описанному выше, может также контролироваться целостность метаданных в конкретной базе данных. Для этого используется утилита mint, также входящая в состав дистрибутива «Ред База Данных» (см. п. 4.9).

4.3 Дискреционный принцип контроля доступа

4.3.1 Общие сведения

Субъектами доступа в СУБД «Ред База Данных» являются пользователи, а также процедуры и процессы, запущенные от имени пользователей.

Объекты доступа - это сами базы данных и объекты баз данных – таблицы, представления, процедуры, триггеры, генераторы, домены, индексы, роли 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 4.2:

Таблица 4.2 - Объекты и операции над ними

№ п/п	Объект	Операция
1	База данных	Создание, изменение, удаление, резервное копирование/восстановление, получение статистики
2	Теневая (оперативная) копия	Создание, удаление
3	Таблица	Создание, изменение, удаление
4	Представление	Создание, изменение, удаление
5	Процедура	Создание, изменение, удаление, запуск (вызов)
6	Триггер	Создание, изменение, удаление
7	Генератор	Создание, изменение, удаление, установка значения, получение значения
8	Индекс	Создание, изменение, удаление
9	Домен	Создание, изменение, удаление
10	Исключение	Создание, изменение, удаление
11	Роль	Создание, изменение прав, удаление, назначение пользователю/другой роли
12	Запись	Добавление, изменение, удаление, выборка

Примечание: Права на DDL-операции с триггерами определяются правами пользователя на таблицу.

Примечание: Если пользователь получает права на изменение/удаление таблицы, процедуры, триггера, генератора, домена, индекса, то он может производить эти действия только над теми объектами, для которых он является владельцем. Пользователь автоматически становится

владельцем объекта, который он создал.

Основным механизмом реализации принципа дискреционного доступа в СУБД «Ред База Данных» является индивидуальное назначение прав пользователям непосредственно или назначение прав ролям с последующим присвоением пользователям необходимых им ролей.

Права пользователя назначаются ему на конкретные операции над конкретными объектами непосредственно или делегируются ему назначением ролей.

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.

Добавление, удаление и изменение пользователей системы могут осуществлять только следующие субъекты безопасности:

- администратор сервера СУБД – пользователь SYSDBA;
- администратор безопасности – пользователь, которому назначена глобальная роль SECADMIN.

Примечание: Назначить пользователю роль SECADMIN пользователю может только администратор сервера – пользователь SYSDBA или пользователь, уже имеющий роль SECADMIN с опцией WITH ADMIN OPTION.

Добавление глобальных ролей могут также производить только пользователь SYSDBA и пользователь с ролью SECADMIN.

Назначение прав пользователям и ролям могут производить SYSDBA и пользователи с ролями SECADMIN, RDB$ADMIN.

Пользователь может изменять только свои учетные данные (пароль, описание)

Делегирование прав осуществляться администратором сервера (пользователем SYSDBA), администратором безопасности сервера или администратором безопасности конкретной базы данных.

4.3.2 Работа с ролями

Администратор сервера баз данных «Ред База Данных» может назначить или лишить других пользователей права назначать права на операции с объектами БД. (операторы GRANT и REVOKE)

Рассмотрим механизмы реализации дискреционного доступа в СУБД «Ред База Данных»

Права пользователям системы назначаются пользователем SYSDBA или пользователями с ролями SECADMIN или RDB$ADMIN. Кроме того, если пользователь получил права на какую-либо операцию совместно с опцией WITH GRANT OPTION, то он может передать права на эту операцию другим пользователям.

Контроль доступа субъектов к защищаемым ресурсам осуществляется через делегирование прав на доступ к объектам БД пользователям непосредственно или через делегирование прав ролям с последующим присвоением этих ролей пользователям.

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[15]. Для назначения роли пользователю используется оператор:

```
GRANT <имя_роли> TO <имя_пользователя>|<имя_роли> [WITH
ADMIN OPTION];
```

Для того, чтобы отнять роль у пользователя, используется оператор:

```
REVOKE <имя_роли> FROM <имя_пользователя>|<имя_роли>;
```

Для отбора у пользователя прав делегировать свои роли, которые ему были выданы с опцией WITH ADMIN OPTION, используется оператор:

```
REVOKE ADMIN OPTION FROM <имя_пользователя>|<имя_роли>;
```

Для удаления роли используется оператор:

```
DROP ROLE <имя_роли>;
```

Роль, под которой пользователь соединяется с базой данных, задается в операторе CONNECT:

```
CONNECT <имя_базы_данных> USER <имя_пользователя>
PASSWORD <пароль> ROLE <имя_роли>;
```

4.3.3 Предопределенные роли

В СУБД «Ред База Данных» существуют предопределенные роли:

- SECADMIN – администратор безопасности;
- RDB$ADMIN – администратор системы;
- PUBLIC – роль по умолчанию для вновь создаваемых пользователей.

Права этих ролей сведены в таблицу 1:

Таблица 4.3 - Права доступа (по умолчанию) предопределенных ролей

Объект / Роль	RDB$ADMIN	SECADMIN	PUBLIC
Операции DML (изменение данных)	+	+/- (Всегда может назначить необходимые привилегии)	-
Операции DDL изменение структуры базы, метаданных	+	Только операции с ролями	-
Операции GRANT/REVOKE	+	+	-
Резервное копирование/ восстановление (GBAK)	+	-	-
Операции с пользователями (GSEC)	-	+	-
Восстановление БД, получение статистики по БД Операции GFIX и GSTAT	+	-	-

То есть роль RDB$ADMIN имеет полные права на базу данных, но не может

15 Подробнее о назначении роли на роль см. п. 4.3.7

производить операции с пользователями и назначать роли пользователям. Роль SECADMIN имеет полные права на операции назначения/отбора (GRANT/REVOKE) прав и поэтому имеет больше полномочий, чем RDB$ADMIN. Роль PUBLIC по умолчанию не имеет никаких прав на операции с данными, операции с метаданными для этой роли определяются ее правами на изменение системного каталога (см. далее). Кроме этих ролей, существует глобальный администратор сервера СУБД «Ред База данных» - пользователь SYSDBA. Этот пользователь имеет все права на все базы данных сервера. Его нельзя лишить этих прав.

Для того, чтобы создать или изменить права глобальной роли, или назначить роль пользователю глобально, необходимо прямое подключение к базе данных безопасности – security2.fdb. Право напрямую подключаться к этой базе данных имеют только пользователь SYSDBA и пользователи, которым назначена глобальная роль администратора безопасности SECADMIN.

4.3.4 Распределение прав на операции управления данными

Для назначения прав пользователям или ролям используется оператор GRANT. В СУБД «Ред База Данных» пользователь может получить права на выполнение операций изменения структуры базы данных (DDL-операции), на операции манипулирования данными (DML-операции). Также пользователь может получить право делегировать свои права другим пользователям (предложение WITH GRANT OPTION в операторе GRANT).

В общем случае GRANT, REVOKE определены для объектов следующих типов: PROCEDURE, FUNCTION, ROLE, TABLE, VIEW, EXCEPTION, GENERATOR, DOMAIN, SHADOW, POLICY. Права на DDL-операции с объектами INDEX не изменяются. Права на создание объектов (CREATE) определяются следующим выражением:

```
GRANT CREATE <объект_БД> TO <имя_пользователя>|<имя_роли>
[WITH GRANT OPTION];
```

и, соответственно,

```
REVOKE CREATE <объект_БД> FROM <имя_пользователя>|
<имя_роли>;
```

Права на операции изменения и удаления объектов (ALTER и DROP) определяются следующими выражениями:

```
GRANT ALTER|DROP [ANY] OBJECT TO <имя_пользователя>|
<имя_роли> [WITH GRANT OPTION];
```

и, соответственно,

```
REVOKE ALTER|DROP [ANY] OBJECT FROM <имя_пользователя>|
<имя_роли>;
```

В данном случае, ANY имеет следующий смысл: если слово ANY указано, то пользователь может удалять и модифицировать любой объект указанного типа, иначе только тот, владельцем которого он является. Если при назначении прав использовалось ключевое слово ANY, то и в операторе REVOKE ANY необходимо указать для аналогичных объектов.

Для отбора у пользователя или роли привилегии передавать свои права на работу с объектами базы данных (если они были получены с опцией WITH GRANT OPTION) используется следующий оператор:

```
REVOKE GRANT OPTION FOR CREATE|ALTER|DROP <объект_БД>
FROM <имя_пользователя>|<имя_роли>;
```

В таблице 4.4 представлены языковые конструкции для выполнения DDL опе-

раций над объектами СУБД.

Таблица 4.4 - Назначение прав на DDL операции

Операции	Объект	Назначение прав
CREATE, ALTER, DROP	TABLE	GRANT CREATE | ALTER | DROP [ANY] TABLE TO USER|ROLE [WITH GRANT OPTION]
CREATE, ALTER, DROP	TRIGGER	Права определяются исходя из прав на таблицу/представление
CREATE, ALTER, DROP	PROCEDURE	GRANT CREATE | ALTER | DROP [ANY] PROCEDURE TO USER|ROLE [WITH GRANT OPTION]
CREATE, ALTER, DROP	VIEW	GRANT CREATE | ALTER | DROP [ANY] VIEW TO USER|ROLE [WITH GRANT OPTION]
CREATE, ALTER, DROP	DOMAIN	GRANT CREATE | ALTER | DROP [ANY] DOMAIN TO USER|ROLE [WITH GRANT OPTION]
CREATE, ALTER, DROP	ROLE	GRANT CREATE | ALTER | DROP [ANY] ROLE TO USER|ROLE [WITH GRANT OPTION]
CREATE, ALTER, DROP	GENERATOR	GRANT CREATE | ALTER | DROP [ANY] GENERATOR TO USER|ROLE [WITH GRANT OPTION]
CREATE, ALTER, DROP	SEQUENCE	GRANT CREATE | ALTER | DROP [ANY] SEQUENCE TO USER|ROLE [WITH GRANT OPTION]
CREATE, ALTER, DROP	EXCEPTION	GRANT CREATE | DROP [ANY] EXCEPTION TO USER|ROLE [WITH GRANT OPTION]
CREATE, DROP	SHADOW	GRANT CREATE | DROP [ANY] SHADOW TO USER| ROLE [WITH GRANT OPTION]
DECLARE, ALTER, DROP	FUNCTION	GRANT CREATE | ALTER | DROP [ANY] FUNCTION TO USER|ROLE [WITH GRANT OPTION]
CREATE, ALTER, DROP	INDEX	Права определяются исходя из прав на таблицу
CREATE, DROP	POLICY	GRANT SECADMIN TO USER|ROLE[16]

Для того, чтобы дать пользователю TestUser возможность создавать таблицы, необходимо выполнить следующую команду:

```
GRANT CRATE 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, триггеров 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
```

16 DDL-операции с политиками может осуществлять только пользователь SYSDBA и пользователь с ролью глобального администратора безопасности — SECADMIN (см. п. 4.3.3).

```
There is no privilege for this operation.
```

4.3.5 Распределение прав на операции манипулирования данными

По аналогии с распределением прав на DDL-операции, администратор системы, владелец базы данных или пользователи с ролями RDB$ADMIN или SECADMIN могут распределять права и на операции доступа к данным (добавление, изменение, выборка, удаление). Синтаксис запросов по доступу к DML-операциям сведен в таблицу 4.5:

Таблица 4.5 - Назначение прав на DML операции

Операция	Объект	Назначение прав
SELECT, INSERT, UPDATE	TABLE [(<column>[,...])] VIEW [(<column>[,...])]	GRANT SELECT|INSERT|UPDATE [(column [, ...])] ON [TABLE] <имя_таблицы> TO <имя_пользователя>| <имя_роли> [WITH GRANT OPTION] GRANT SELECT|INSERT|UPDATE [(column [, ...])] ON [TABLE] <имя_таблицы> TO PROCEDURE <имя_процедуры>|TRIGGER <имя_триггера> REVOKE SELECT|INSERT|UPDATE [(column [, ...])] ON [TABLE] <имя_таблицы> FROM <имя_пользователя>| <имя_роли>|PROCEDURE <имя_проседуры>|TRIGGER <имя_триггера> REVOKE GRANT OPTION FOR SELECT|INSERT|UPDATE [(column [, ...])] ON [TABLE] <имя_таблицы> FROM <имя_пользователя>|<имя_роли>
DELETE	TABLE,VIEW	GRANT DELETE ON [TABLE] <имя_таблицы> TO <имя_пользователя>|<имя_роли> [WITH GRANT OPTION] GRANT DELETE ON [TABLE] <имя_таблицы> TO PROCEDURE <имя_процедуры>|TRIGGER <имя_триггера> REVOKE DELETE ON [TABLE] <имя_таблицы> FROM <имя_пользователя>|<имя_роли>|PROCEDURE <имя_процедуры>|TRIGGER <имя_триггера> REVOKE GRANT OPTION FOR DELETE ON [TABLE] <имя_таблицы> FROM <имя_пользователя>|<имя_роли>
SET GENERATOR, функция GEN_ID[17]	GENERATOR	GRANT SELECT|ALTER ON GENERATOR <имя_генератора> TO <имя_пользователя>|<имя_роли> [WITH GRANT OPTION] GRANT SELECT|ALTER ON GENERATOR <имя_генератора> TO PROCEDURE <имя_процедуры>| TRIGGER <имя_триггера> REVOKE SELECT|ALTER ON GENERATOR <имя_генератора> FROM <имя_пользователя>|<имя_роли>| PROCEDURE <имя_процедуры>|TRIGGER <имя_триггера> REVOKE GRANT OPTION FOR SELECT|ALTER ON GENERATOR <имя_генератора> FROM <имя_пользователя>|<имя_роли>

17 Привилегия SELECT на генератор даёт пользователю право вызывать для него функцию GEN_ID() со значениями 0 и 1. Для использования других значений нужна также привилегия на ALTER.

Операция	Объект	Назначение прав
EXECUTE	PROCEDURE	GRANT EXECUTE ON PROCEDURE <имя_процедуры> TO <имя_пользователя>|<имя_роли> [WITH GRANT OPTION] GRANT EXECUTE ON PROCEDURE <имя_процедуры> TO PROCEDURE <имя_процедуры>|TRIGGER <имя_триггера> REVOKE EXECUTE ON PROCEDURE <имя_процедуры> FROM <имя_пользователя>|<имя_роли>|PROCEDURE <имя_процедуры>|TRIGGER <имя_триггера> REVOKE GRANT OPTION FOR EXECUTE ON PROCEDURE <имя_процедуры> FROM {<имя_пользователя>|<имя_роли>

Таким образом, для того, чтобы дать пользователю, например, 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 TestTable;
```

```
Statement failed, SQLCODE = -551
no permission for read/select access to TABLE TESTTABLE
```

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 TestTable пользователь получит сообщение об ошибке:

```
Statement failed, SQLCODE = -551
no permission for insert/write access to TABLE TestTable
```

Аналогично пользователю можно дать или отнять права на добавление/изменение/удаление/просмотр данных, запуск процедур. Причем права на изменение, добавление и просмотр 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 TestTable TO TestUser;
COMMIT;
CONNECT 'TestDB.fdb' USER TESTUSER PASSWORD TestPass123;
UPDATE TestTable 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 TestTable Name='Tom';
```

выполнится без ошибок.

Исключение составляет оператор SELECT. Если в списке выбора полей этого оператора указать * (все поля), а пользователь имеет право выбирать только часть полей, то он получит только разрешенные ему поля[18]

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.

Кроме назначения прав непосредственно пользователям возможно назначение прав ролям. 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.

4.3.6 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).

Права на доступ к административным функциям могут назначаться только непосредственно пользователям или глобальным ролям, поэтому запросы на делегирование прав на системные сервисы должны выполняться при прямом подключении к базе данных безопасности security2.fdb. Попытка выполнения такого запроса при подключении к пользовательской базе данных приведет к выводу сообщения об ошибке назначения прав на сервис.

Назначение права на запуск сервиса осушествляется с помощью следующего оператора SQL :

```
GRANT EXECUTE ON SERVICE <имя_сервиса> TO
<имя_пользователя>|<имя_роли>
```

18 Для этого необходимо включить и проинициализировать режим фильтрации в файле firebird.conf — установить параметр UseRecordFilter = 1 и инициализировать систему фильтрации столбцов и записей с помощью скрипта filtering.sql (подробнее см. п. 4.8.1)

и, соответственно, для лишения пользователя или глобальной роли прав на работу с сервисом используется оператор

```
REVOKE EXECUTE ON SERVICE <имя_сервиса> FROM
<имя_пользователя>|<имя_роли>
```

Здесь параметр <service_name> может иметь следующие значения:

- BACKUP;
- RESTORE;
- GFIX;
- GSTAT.

Операции с сервисом GSEC не регламентируются этим оператором. Любой пользователь может производить такие операции, но его права при этом ограничены возможностью изменения своего собственного пароля. Полные права на сервис GSEC имеет только пользователь SYSDBA и пользователь с глобальной ролью администратора безопасности SECADMIN.

4.3.7 Кумулятивное действие ролей

В СУБД «Ред База Данных» действует принцип кумулятивного действия ролей. Это значит, что, если пользователь при подключении к базе данных не укажет конкретную роль, то он получит права всех назначенных ему ролей. Кроме этого, также возможно назначение прав одной роли другой.

Правила кумулятивного действия ролей:

- если пользователь не указывает роль при подключении к серверу, то он получает права всех ролей, которые ему назначены;
- если пользователь при подключении указал конкретную роль, то он получает только её привилегии;
- 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.

4.3.8 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 AUTHID OWNER|CALLER. По умолчанию этот параметр равен OWNER, т.е. процедура выполняется с правами её владельца. Синтаксис объявления процедуры следующий:

```
CREATE PROCEDURE <имя_процедуры>
    [AUTHID OWNER|CALLER]
    [(param <datatype> [, param <datatype> …])]
    [RETURNS <datatype> [, param <datatype> …])]
    AS <procedure_body> [terminator]
    <procedure_body> =
    [<variable_declaration_list>]
    <block>
    <variable_declaration_list> =
    DECLARE VARIABLE var <datatype>;
    [DECLARE VARIABLE var <datatype>; …]
    <block> =
    BEGIN
    <compound_statement>[<compound_statement> …]
    END
```

4.4 Идентификация и аутентификация[19]

4.4.1 Основные понятия

Идентификация и аутентификация пользователей являются основой дискреционного доступа субъектов безопасности в систему. Идентификация — это предъявление пользователем своего имени. Идентификация позволяет однозначно определить, какие именно права назначены данному пользователю.

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. «Ред База Данных» 2.5 поддерживает следующие режимы аутентификации:

- традиционная (Native) аутентификация;
- доверительная (Trusted) аутентификация;
- многофакторная (Multifactor) аутентификация;
- смешанная (Mixed) аутентификация.

При первичной аутентификации сервер проверяет первичные факторы. В случае успешной проверки сервер и клиент вырабатывают сеансовые ключи.

Проверка фактора пароля производится по протоколу RS CHAP:

Условные обозначения:

- h(x) – функция хеширования строки x. Разрядность определяется криптопровайдером.
- enc(x, k) – функция симметричного шифрования строки x, ключом k.
- dec(x, k) – функция симметричного дешифрования строки x, ключом k.
- g() – генератор случайного числа необходимого размера.
- p – введенный пароль пользователя в открытом виде.
- L – логин пользователя.
- hs1=h(L + p) – хеш суммы имени и пароля пользователя. Хранится на сервере.

Протокол аутентификации:

1. Клиент вычисляет хеш клиента 1 как сумму логина и пароля клиента: hc1 = h(L + p)
2. Клиент посылает challenge-запрос серверу на начало процедуры авторизации с указанием имени пользователя, под которым он хочет авторизоваться.
3. Сервер генерирует случайное число s=g() длиной 32 байта
4. Сервер шифрует сгенерированное случайное число с помощью своего

19 Для работы данной функции необходимо наличие криптопровайдера и криптоплагина (последний входит в поставку СУБД «Ред База Данных»). Криптопровайдер реализует функции шифрования и хеширования, а криптоплагин предоставляет унифицированный интерфейс между криптопровайдером и сервером СУБД «Ред База Данных». В качестве криптопровайдера в настоящее время используется КриптоПро CSP 3.0. Подробнее о настройке КриптоПро CSP 3.0 см. приложение В

ключа s' = enc(s, hs1)
5. Сервер передает s' и название алгоритма хеширования клиенту
6. Клиент дешифрует полученное число своим ключом s = dec(s', hc1)
7. Клиент вычисляет хеш клиента 2 как сумму хеша клиента 1 и полученного от сервера случайного числа hc2 = h(s + hc1)
8. Клиент передает вычисленный на предыдущем этапе хеш клиента 2 hc2 серверу
9. Сервер вычисляет hs2 = h(s + hs1)
10. Сервер сравнивает: если вычисленный hs2 равен полученному hc2, то пользователь прошел аутентификацию, если нет, то сервер направляет клиенту отказ.

Таким образом, ни пароль, ни случайная составляющая никогда не передается в открытом виде.

Проверка клиента по сертификату производится по следующему алгоритму:

1. Клиент посылает свой сертификат серверу.
2. Сервер проверяет его подлинность; генерирует сессионный ключ; на основе открытого ключа пользователя и своей ключевой пары, предназначенной для обмена, вырабатывает ключ парной связи; шифрует сессионный ключ ключом парной связи; посылает зашифрованный сессионный ключ и свой открытый ключ клиенту.
3. Клиент вырабатывает ключ парной связи на основе своего закрытого ключа и открытого ключа сервера, которым дешифрует сессионный ключ и шифрует им ключевое слово «success», которое отправляет серверу.
4. Сервер дешифрует слово и проверяет. В случае совпадения считается, что пользователь, на чье имя выписан сертификат, успешно прошел аутентификацию по этому фактору.

Первичное назначение пароля происходит при установке СУБД под суперпользователем SYSDBA. Далее пользователь SYSDBA может создать требуемое количество пользователей, со всеми необходимыми правами. Смена пароля происходит в рамках уже установленного соединения с использованием сессионных ключей, с помощью которых симметричным шифрованием шифруется новый пароль. На стороне сервера он дешифруется, проверяется на соответствие политике безопасности и сохраняется.

4.4.2 Режимы аутентификации

Традиционная (Native) аутентификация

Традиционная аутентификация подразумевает использование пароля в качестве единственного фактора аутентификации. Для включения режима традиционной аутентификации следует в файле конфигурации firebird.conf выставить значение параметра Authentication = native.

Для аутентификации в традиционном режиме необходимо предъявить имя пользователя и пароль, например:

```
isql test.fdb -user testuser -password testpass
```

При создании пользователя в режиме традиционной аутентификации его многофакторный пароль будет равен NULL (см. п. «Многофакторная (Multifactor) аутентификация»

Доверительная (Trusted) аутентификация

В доверительном режиме аутентификации используется система безопасности операционной системы. Для включения режима доверительной аутентификации следует в файле конфигурации firebird.conf выставить значение параметра Authentication = trusted.

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. Если пользователь входит в группу администраторов, то он может получить роль RDB$ADMIN командой:

```
ALTER ROLE RDB$ADMIN ADD OS_NAME 'Domain Admins'
```

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.

Администратор сервера может решить, какую роль должен получить пользователь домена при подключении к серверу (по умолчанию дается роль public). Для этого существует команда:

```
ALTER role role_name ADD|DEL OS_NAME 'OS_Secure_Object'
```

role_name – имя роли (локальной или глобальной RDB$ADMIN|SECADMIN), OS_Secure_Object – имя группы или пользователя домена,

ADD|DEL – действия добавления и удаления роли для группы или пользователя домена.

В данный момент ролью может быть только RDB$ADMIN, OS_Secure_Object – ‘Domain Admins’. Таким образом, простые пользователи домена всегда получают роль public, члены группы администраторов домена могут получить роль RDB$ADMIN, если выполнена команда “ALTER ROLE RDB$ADMIN ADD OS_NAME ‘Domain Admins’”, или public.

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.

Примечание: 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».

Многофакторная (Multifactor) аутентификация

В зависимости от числа предъявляемых при аутентификации факторов различают однофакторную и многофакторную аутентификацию. В «Ред База Данных» 2.5 реализована возможность использования многофакторной аутентификации. Требуемые для аутентификации пользователя факторы определяются политикой безопасности, назначенной пользователю. Подробно политики безопасности рассмотрены в пункте 4.5

Для использования только режима многофакторной аутентификации следует

в файле конфигурации firebird.conf выставить значение параметра Authentication = multifactor.

Также для использования этого режима в конфигурационном файле сервера и клиента firebird.conf необходимо указать используемый криптоплагин и серверный набор ключей.

Для того, чтобы пользователь «Ред База Данных» смог пройти аутентификацию, он должен быть предварительно создан с указанием ключа –mf. Например:

```
gsec –user sysdba –password masterkey -database
localhost:c:\RDB\security2.fdb –add test –pw test –mf
```

Внимание! При создании многофакторного пользователя необходимо указывать полный сетевой путь до базы security2.fdb.

Для совместимости многофакторного режима аутентификации с традиционными пользователями «Ред База Данных» в файл конфигурации был добавлен параметр LegacyHash. Чтобы традиционные пользователи «Ред База Данных» могли пройти аутентификацию в многофакторном режиме, нужно установить значение этого параметра равным единице. Если значение параметра LegacyHash равно 0, то традиционные пользователи (в том числе и SYSDBA) не смогут аутентифицироваться в многофакторном режиме (то есть если Authentication=multifactor).

Примечание: Для того, чтобы сделать SYSDBA многофакторным, необходимо изменить его пароль в многофакторном режиме.

Хеш пароля многофакторного пользователя хранится в поле RDB$MF_PASSWD таблицы RDB$USERS. При добавлении нового пользователя или изменении его пароля в многофакторном режиме новый пароль устанавливается как для многофакторной, так и для традиционной аутентификации. При добавлении нового пользователя или изменении его пароля в режиме традиционной аутентификации, многофакторный пароль не создаётся (RDB$MF_PASSWD=NULL), и пользователь не сможет соединяться с БД в многофакторном режиме.

Для подключения к серверу «Ред База Данных» в режиме многофакторной аутентификации необходимо указать ключ –mf и все требуемые политикой безопасности факторы. Например, если политика безопасности пользователя требует аутентификации по факторам «пароль», «сертификат» и «доверительная аутентификация», 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> -tr –mf
```

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 (псевдонима). 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.

- ключ –tr указывает на аутентификацию по доверительному фактору. Ключ –mf указывают на то, что пользователь аутентифицируется в многофакторном режиме.

Контейнер с набором ключей и сертификат создаются заранее.

Примечание: Для того, чтобы пройти аутентификацию по доверительному фактору в многофакторном режиме, нужно создать пользователя, имя которого совпадает с доменным именем пользователя системы, который будет проходить доверительную аутентификацию. Например, вы вошли в домен “domain" под пользователем administrator, соответственно, имя пользователя будет domain\administrator.

Назначить политику такому пользователю можно будет так:

```
grant policy <имя_политики> to “DOMAIN\ADMINISTRATOR”;
```

Смешанная (Mixed) аутентификация

В режиме смешанной аутентификации допускаются все вышеописанные режимы аутентификации. В этом режиме пользователь может аутентифицироваться традиционным способом, используя доверительную аутентификацию и многофакторно. Для использования режима смешанной аутентификации следует в файле конфигурации firebird.conf выставить значение параметра Authentication = mixed.

В режиме смешанной аутентификации возможны, например, следующие варианты соединения с сервером «Ред База Данных»:

Традиционная аутентификация:

```
isql test.fdb –user testuser –password testpass
```

Доверительная аутентификация:

```
isql test.fdb
isql test.fdb –tr
isql test.fdb –tr –mf
```

Многофакторная аутентификация:

```
isql test.fdb –user testuser –password testpass –mf –tr
```

Особенности многофакторной аутентификации

- возможность использования результата аутентификации в ОС АРМ или на контроллере домена;
- Доступ к БД определяется политикой безопасности, в которой определены факторы, которые должны быть предъявлены пользователем для прохождения процедуры аутентификации;
- при аутентификации все данные пользователя, кроме имени, передаются только в зашифрованном виде;
- в настоящее время используются следующие факторы: пароль, сертификат и результат аутентификации в ОС.

Вторичные факторы 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. Ведутся доработки для использования следующих вторичных факторов аутентификации: отпечатки пальцев, сетчатки глаза и т.д.

При первичной аутентификации сервер проверяет первичные факторы, и в

результате аутентификации сервер и клиент вырабатывают сеансовые ключи. Далее клиент имеет возможность предъявить вторичные факторы, зашифрованные сеансовыми ключами.

Политики безопасности, которые определяют требуемые факторы аутентификации, хранятся в базе данных безопасности security2.fdb. Такие политики назначаются на пользователя. Аутентификация клиента считается успешной, если предъявленные им факторы аутентификации удовлетворяют политике безопасности.

В базе данных безопасности хранится не только хеш пароля, но и название алгоритма, с помощью которого этот хеш получен.

Смена пароля происходит в рамках уже установленного соединения с использованием сессионных ключей, с помощью которых симметричным шифрованием шифруется новый пароль. На стороне сервера он дешифруется, проверяется на соответствие политике безопасности и сохраняется.

Таблица 4.6 - Параметры конфигурационного файла, имеющие отношение к механизму многофакторной аутентификации

Параметр	Возможное значение	Комментарий
Authentication	native|trusted|mixed|multifactor	native — режим совместимости с более ранними версиями trusted — разрешена только trusted аутентификация (аутентификация Windows) multifactor — только многофакторная аутентификация mixed — любой из трех предыдущих вариантов аутентификации
LegacyHash	0 | 1	Позволяет пользователям, которые созданы как немногофакторные, соединяться с базой при включенном режиме Autentication=multifactor
CryptoPluginName	Значение по умолчанию - wincrypt_plugin	Имя библиотеки криптопровайдера (расположена в каталоге plugins сервера)
KeyRepository	<имя контейнера>	Имя контейнера с ключами сервера для установки защищенного соединения
ServerCertificate	<путь к сертификату>	Путь к файлу сертификата сервера. Он будет передан для проверки клиенту, если тот потребует этого.
CertUsernameDN	<атрибут>	Атрибут сертификата, из которого будет извлекаться имя его владельца. По умолчанию - CN.
CertUsernamePattern	<атрибут>	Регулярное выражение, применяемое к атрибуту сертификата с именем пользователя для извлечения самого этого имени. Использует синтаксис SQL, по умолчанию не задано.
VerifyCertChain	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.

4.4.3 Аутентификация по протоколу LDAP

Для аутентификации по протоколу LDAP в firebird.conf должны быть заданы настройки подключения к серверу каталогов в параметрах вида «LDAP…».

Адрес сервера LDAP (IP-адрес или символьное имя) указывается в параметре firebird.conf «LDAPServer».

Тип шифрования, используемый при подключении к LDAP, задаётся параметром «LDAPEncryption». Он может принимать три значения:

- None – шифрование отсутствует, коннект к серверу по порту 389 (по умолчанию);
- SSL – подключение по протоколу LDAPS по порту 636.
- TLS – подключение с командой START_TLS к порту 389.

Сервер LDAP, к которому выполняется подключение, должен быть настроен соответствующим образом.

Если задан параметр «VerifyCertChain» (по умолчанию), то при SSL/TLS-соединениях будет выполняться проверка сертификата LDAP-сервера со стороны сервера «Ред База Данных». Если сертификат не проходит проверку, соединение разрывается. Верификация сертификата выполняется аналогично проверке пользовательского сертификата.

Если параметр «VerifyCertChain» отключен, проверка сертификата LDAP-сервера не выполняется.

Имя пользователя, которое будет использоваться для подключения к серверу LDAP, указывается в параметре «LDAPUserDN» в виде DN, например:

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

Пароль пользователя для подключения к серверу указывается в параметре «LDAPPassword».

Ветвь в каталоге, относительно которой будут искаться пользователи, указывается в параметре «LDAPUserBase» в виде DN. Поиск пользователей по данной базе выполняется рекурсивно по всем вложенным веткам, т.е. внутри этой ветви можно создавать другие ветви с учётной информацией пользователей – они будут найдены.

Ветвь в каталоге, относительно которой будут искаться группы, указывается в параметре «LDAPGroupBase» в виде DN.

Для определения списка групп, к которым принадлежит пользователь в LDAP, могут использоваться различные схемы. Указать конкретную схему можно в параметре «LDAPMembershipFilter». В нём в качестве имени предполагаемого пользователя указывается шаблон %u. Две основные схемы определения групп выглядят следующим образом:

```
LDAPMembershipFilter = memberUid=%u
LDAPMembershipFilter = member=uid=
%u,ou=people,dc=example,dc=com
```

Для проверки сертификата пользователя через LDAP нужно указать название атрибута, в котором сертификат пользователя будет храниться на сервере LDAP. За это отвечает параметр «LDAPUserCertificate». Если этот параметр не задан, сертификат пользователя не будет проверяться на соответствие его сертификату из LDAP. Если в названии атрибута есть подстрока «binary», считается, что сертификат сохранён в двоичном формате, иначе – в текстовом. По стандарту сертификат сохраняется в текстовом виде в атрибуте «userCertificate», в двоичном – в атрибуте «userCertificate;binary».

При native-аутентификации сервер сначала пытается проверить пользователя через security2.fdb. Если в ней пользователь не найден или он найден, но его пароль не совпадает, а в конфигурации задан параметр LDAPServer, выполняется попытка найти пользователя в LDAP. Если проверка через security2.fdb не прошла, и не задан адрес LDAP-сервера, или пользователь не найден в LDAP, сообщается об ошибке аутентификации.

Пароль пользователя для native-аутентификации хранится в в LDAP в атрибуте «rdbPassword» в том же виде, в котором он сохраняется в security2.fdb. Алгоритм шифрования пароля «NATIVE» описан ниже. Пароль, используемый при многофакторной аутентификации, хранится в атрибуте «rdbSecurePassword». Алгоритм хэширования, использованный для получения многофакторного пароля, сохраняется в атрибуте «rdbPasswordAlgorithm».

При многофакторной аутентификации по паролю сначала выполняется попытка найти пользователя в базе security2.fdb. Если он там найден, аутентификация выполняется по считанным из неё данным (пароль, алгоритм его шифрования). Если пользователь в базе не найден, и в конфигурации заданы параметры подключения к LDAP-серверу, выполняется попытка считать аналогичные данные из LDAP. Если пользователь не найден в security2.fdb и не найден в LDAP (или подключение к LDAP не настроено), проверка пароля считается неуспешной.

При многофакторной аутентификации по сертификату после верификации сертификата 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 «VerifyCertChain» (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.

После аутентификации с сервера LDAP запрашивается дополнительная информация о пользователе, которая будет сохраняться в контекстных переменных в пространстве имён AUTHDATA. Если соответствующий атрибут у пользователя отсутствует, контекстная переменная будет возвращать NULL. Список запрашиваемых атрибутов и соответствующие им контекстные переменные указаны в таблице в конце документа.

Также заполняются переменные LDAP_ROLES (список ролей пользователя, полученных из каталога LDAP) и AUTH_TYPE (тип аутентификации – LDAP или SECURITY).

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. Назначаются как роли, найденные в базе, к которой выполняется подключение, так и глобальные роли, заданные в security2.fdb

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 security2.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. Сюда записывается хэш SHA1 в кодировке BASE64.
- sambaLMPassword – LMHash для Samba.
- sambaNTPassword – MD4 хэш для Samba.
- rdbPassword – пароль пользователя на сервере БД, зашифрованный алгоритмом NATIVE, 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.

Если меняется пароль у многофакторного пользователя (ключ –mf у GSEC, 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 име-

ется 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 security2, ни в LDAP, возвращается ошибка «record not found for user».

На клиентской стороне используемый способ аутентификации может задаваться переменной окружения RDB_AUTH_MODE в виде

RDB_AUTH_MODE=[native][,trusted][,multi_factor]

Клиентская библиотека будет пробовать выполнить аутентификацию в порядке, заданном в этой переменной. Сообщение об ошибке будет выдаваться если ни один из указанных методов не смог обеспечить аутентификацию клиента.

Переменная RDB_AUTH_MODE будет использоваться только в том случае, если клиентское приложение не указало явно тэги, отвечающие за аутентификацию, т.е.

isc_dpb_trusted_auth/isc_spb_trusted_auth – доверенная;

isc_dpb_multi_factor_auth/isc_spb_multi_factor_auth – многофакторная.

Если какой-либо из этих тэгов указан, переменная игнорируется.

При отсутствии тэгов и переменной окружения, будет использована native-аутентификация.

При использовании многофакторной аутентификации, инициированной переменной окружения RDB_AUTH_MODE, дополнительно будут просматриваться переменные окружения RDB_AUTH_CERT_STORE (название контейнера с клиентскими ключами) и RDB_AUTH_CERT_PATH (путь к файлу с клиентским сертификатом). Если они заданы, то указанный в них сертификат будет использован в качестве фактора аутентификации. Если клиент явно указал тэги с типом аутентификации (многофакторная или доверенная), значения переменных не используются.

4.5 Политики безопасности

4.5.1 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 одновременно открытых сессий пользователя;
- продолжительность простоя пользователя до отключения.

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.

Внимание! Политики безопасности применимы только для многофакторных пользователей при многофакторном подключении к БД.

4.5.2 Создание политик безопасности

Для создания, изменения или удаления политики безопасности администра-

тору необходимо соединиться с какой-либо базой данных для того, чтобы создать политику, с помощью оператора CREATE POLICY. Синтаксис этого оператора приведен ниже:

```
CREATE POLICY policy_name AS [param = value [, param =
value]];
```

Примечание: хотя сами политики хранятся в базе данных безопасности security2.fdb, создать их можно, соединившись с любой базой данных. Однако пользователь при этом должен иметь права на запись в базу данных безопасности security2.fdb — эти права есть только у пользователя SYSDBA и у пользователей с ролью SECADMIN.

Возможные значения параметров следующие:

- AUTH_FACTORS = (auth_factors_expr_list) - факторы аутентификации;
- PSWD_NEED_CHAR=long_integer - минимальное количество букв;
- PSWD_NEED_DIGIT=long_integer - минимальное количество цифр;
- PSWD_NEED_DIFF_CASE = bool_const - требование различности регистров;
- PSWD_MIN_LEN = long_integer - минимальная длинна;
- PSWD_VALID_DAYS = long_integer срок действия пароля;
- PSWD_UNIQUE_COUNT = long_integer - количество последних не повторяющихся паролей;
- MAX_FAILED_COUNT = long_integer – количество неудачных попыток входа;
- MAX_SESSIONS '=' long_integer – максимальное число одновременных подключений;
- MAX_IDLE_TIME = long_integer – время бездействия, в секундах.

Факторы аутентификации могут быть следующими:

Таблица 4.7 - Символическое обозначение факторов аутентификации

Символическое обозначение	Расшифровка
WINDOWS_NTLM	Результат аутентификации в Windows
CERT_X509	Сертификат пользователя
PASSWORD	Пароль
FINGERPRINT	Отпечатки пальцев
EYE	Снимок сетчатки глаза

Следующий пример демонстрирует создание политики:

```
CREATE POLICY TestPolicy AS PSWD_NEED_CHAR=5,
PSWD_NEED_DIGIT=3, PSWD_NEED_LEN=8,
PSWD_NEED_DIFF_CASE=true; PSWD_VALID_DAYS=15,
PSWD_UNOQUE_COUNT=5, MAX_FAILED_COUNT=5, MAX_SESSIONS=10,
MAX_IDLE_TIME=1800;
```

Назначение политики пользователям

Для того, чтобы назначить созданную политику пользователю, необходимо

выполнить следующую команду:

```
GRANT POLICY <имя_политики> TO <имя_пользователя>;
```

Политика безопасности применяется при смене пароля пользователем. Таким образом, после создания пользователей и назначения им политик, администратор должен сменить всем пользователям пароли, чтобы они удовлетворяли требованиям сопоставленных этим пользователям политик безопасности.

Политики назначаются только на пользователей (не на роли). Назначить политику несуществующему пользователю нельзя.

Для смены пароля пользователя администратор сервера (пользователь SYSDBA) или администратор безопасности (пользователь с ролью SECADMIN) должен выполнить следующую команду в утилите gsec :

```
gsec -user SYSDBA -password masterkey -modify
<имя_пользователя> -pw <новый пароль> -mf
```

Пользователь также может и самостоятельно изменять свой пароль:

```
gsec -user <имя_пользователя> -password <старый_пароль>
-modify <имя_пользователя> -pw <новый_пароль> -mf
```

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:

```
An error occurred while attempting to modify the user record.
```

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.
```

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;
- ограничить число одновременных подключений для пользователя;
- автоматически отключать пользователя при длительном бездействии и требовать прохождения им повторной аутентификации.

Кроме того, в случае, если пользователь не прошел процедуру аутентификации, он не получит информации о том, какой именно из предъявленных им факторов является неправильным.

4.6 Аудит

Аудит событий в «Ред База Данных» реализован на основе утилиты FBTrace. В зависимости от параметров, заданных в ее конфигурационном файле, эта утилита отслеживает события соединения и отсоединения от БД (создания и удаления БД), операции DML и DDL, выполнение хранимых процедур и т.д. Утилита стартует одновременно с сервером «Ред База Данных» и ведет лог-файлы для каждой из отслеживаемых баз данных. По умолчанию лог-файлы размещаются в том же каталоге что и отслеживаемая (логгируемая) 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).

4.6.1 Типы и параметры 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;
- проверка предъявленного фактора аутентификации;
- 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 триггера;

- установка значения контекстной переменной;
- начало и завершение транзакции.

Для каждого события, кроме предъявления фактора аутентификации, в журнал аудита сохраняется следующая информация:

- дата, время и тип события;
- идентификатор процесса, вызвавшего событие;
- результат (успешно, не успешно, не санкционировано);

Для каждого из регистрируемых событий записывается следующая информация:

1. Для присоединения к БД — сведения о соединении (путь к БД, идентификатор соединения, пользователь, от имени которого совершается соединение, протокол соединения, IP адрес или имя компьютера, с которого установлено соединение). Также указывается, происходит ли подключение к существующей БД, либо создается новая база. При отключении от базы определяется, произошло ли просто отключение, либо база была удалена.
2. Для фактора аутентификации указывается, дата, время и тип события; идентификатор процесса, вызвавшего событие; имя пользователя; тип фактора; сам предъявленный фактор (содержимое зависит от типа); результат предъявления (успешно, не успешно, не санкционировано).
3. Для присоединения к сервису, отсоединения от него, старта сервиса указываются сведения, общие для всех событий, кроме пути к БД и идентификатора соединения. Вместо них указывается имя сервиса.
4. Для события старта сервиса также указываются параметры старта (опции командной строки, с которыми он был запущен).
5. Для запроса к сервису также указывается имя сервиса, кроме того, сохраняется содержимое запроса, который был передан сервису.
6. Для установки значения контекстной переменной – идентификатор транзакции, пространство имен (namespace), для которого она устанавливается, имя переменной и ее значение.
7. Для хранимой процедуры – идентификатор транзакции, имя процедуры, входные параметры процедуры, время выполнения, статистика производительности.
8. Для транзакции – идентификатор транзакции и ее параметры (уровень изоляции, режим блокировки). При событии завершения транзакции также сохраняется результат выполнения – commit для подтвержденных транзакций и rollback для откаченных.
9. Для подготовки SQL-запроса – идентификатор транзакции, идентификатор запроса, содержимое запроса, время подготовки запроса (в мс), план выполнения.
10. Для выполнения SQL или BLR запроса – идентификатор транзакции, идентификатор запроса, время выполнения запроса (в мс), статистика производительности, параметры запроса, содержимое запроса (в случае текстового формата лога).
11. Для выполнения DYN-запроса – идентификатор транзакции, время выполнения запроса (в мс), содержимое запроса (в текстовом представлении для текстового лога, в бинарном – для бинарного).
12. Для компиляции BLR-запроса - идентификатор транзакции, идентификатор запроса, содержимое запроса (в текстовом представлении для

текстового лога, в бинарном – для бинарного), время подготовки запроса (в мс).

Примечание: По умолчанию система аудита выключена.

Примечание: 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).

Примечание: Сообщения о вызове сервисов и предъявлении факторов аутентификации записываются в лог-файл базы security2.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. 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.

4.6.2 Настройка аудита. Параметры конфигурационного файла

Настройка регистрации событий происходит с помощью изменения параметров в файле fbtrace.conf, расположенном в каталоге установки «Ред База Данных».

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 логгирование единственного типа событий — завершения выполнения SQL-запросов (если включен сам аудит). В двоичном режиме автоматически регистрируются события всех типов, а параметры, отвечающие за выбор определённых событий, игнорируются.

В конфигурационном файле настраиваются следующие параметры:

enabled true/false – вести аудит или нет. По умолчанию аудит выключен.

format 0/1 – формат лог-файла. 0 – текстовый (по умолчанию), 1 – бинарный.

time_threshold <число> – минимальный предел времени, при котором регистрируются события выполнения запросов. Если время выполнения меньше указанного, то запись об операции не помещается в лог. Значение по умолчанию – 100 мс.

max_sql_length <число> – максимальная длина одной записи SQL-запроса в лог-файле, в байтах. Значение по умолчанию – 0 (неограниченно), максимальное значение – 64К. Если длина запроса больше указанного здесь значения, запрос будет обрезан. Применяется только для текстового формата лог-файла[20].

max_blr_length <число> – максимальная длина BLR-запроса, сохраняемого в лог, в

20 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.

байтах. Значение по умолчанию – 500 байт. Максимальное значение – 64К. 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 – 64К. 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 – 64К. Если длина параметра больше указанного здесь значения, параметр будет обрезан. Применяется только для текстового формата лог-файла.

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.

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 – 50.

print_plan true/false – включает/отключает печать планов запросов. По умолчанию отключено.

print_perf true/false – включает/отключает печать статистики выполнения запросов. По умолчанию отключено.

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. Значение по умолчанию – false.

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. Значение по умолчанию – false.

print_stack_trace true/false – включает/отключает печать стека вызовов функций сервера при завершении запроса с ошибкой. По умолчанию отключено.

log_errors_only true/false – включает/отключает запись только тех событий, которые завершились с ошибкой. По умолчанию отключено.

log_changes_only true/false – включает/отключает запись только тех запросов, которые изменяли данные в базе. По умолчанию отключено.

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. При этом запись ведётся как в лог-файл, так и в системный журнал (Журнал событий в Windows, syslog в linux). По умолчанию отключено.

connection_id <число> – задаёт номер (идентификатор) подключения на сервере, которое будет отслеживаться. По умолчанию равно 0, т.е. отслеживаются все подключения.

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.

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.

log_auth_factors true/false – определяет, записывать ли события проверки предъявленных факторов аутентификации в лог-файл.

log_connections true/false – определяет, записывать ли события присоединения/отсоединения к БД в лог-файл.

log_init true/false – определяет, записывать ли события начала/окончания ведения аудита БД в лог-файл.

log_transactions true/false – определяет, записывать ли события начала и завершения транзакций в лог-файл.[21]

log_statement_prepare true/false – определяет, записывать ли события подготовки запросов к БД в лог-файл.

log_statement_free true/false – определяет, записывать ли события освобождения запросов к БД в лог-файл.

log_statement_start true/false – определяет, записывать ли события начала выполнения запросов к БД в лог-файл.

log_statement_finish true/false – определяет, записывать ли события окончания выполнения запросов к БД в лог-файл.

log_context true/false – определяет, записывать ли события изменений значений контекстных переменных в лог-файл.

log_procedure_start true/false – определяет, записывать ли события начала выполнения хранимых процедур.

21 При подтверждении транзакции записывается операция commit, при откате - rollback

log_procedure_finish true/false – определяет, записывать ли события завершения выполнения хранимых процедур.

log_trigger_start true/false – определяет, записывать ли события начала выполнения триггеров.

log_trigger_finish true/false – определяет, записывать ли события завершения выполнения триггеров.

log_blr_requests true/false – определяет, записывать ли события прямого выполнения откомпилированных запросов во внутреннем представлении сервера - BLR.

log_dyn_requests true/false – определяет, записывать ли события прямого выполнения откомпилированных запросов на изменение метаданных (DDL) во внутреннем представлении сервера - DYN.

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, log_filename=\1.log, означает использование файла лога с именем, совпадающим с полным именем аудируемой БД и с расширением log (данный пример будет корректно работать только в случае задания имени БД, для которой ведется аудит, в виде регулярного выражения). 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.

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 с именами  test.log, azk2.log,
rules.log соответственно
    log_filename \1.log
</database>
```

Пример 2:

```
#Для всех БД на диске C с расширением fdb — формат лога
текстовый
<database C:%.fdb>
    enabled true
    format 0
</database>
#Для всех БД с расширением fdb на диске D — формат лога
бинарный
<database D:%.fdb>
    enabled true
    format 1
</database>
```

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.

4.7 Адаптер для подключения бинарного файла аудита

Это инструмент анализа журнала аудита в бинарном формате, который позволяет:

- производить фильтрацию записей за период времени;
- производить поиск записей по указанным характеристикам.

Инструмент анализа разбирает бинарные лог-файлы системы аудита «Ред База Данных» и предоставляет возможность просмотра и фильтрации записей о событиях. Удаление и редактирование записей не позволяется. 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).

Для работы с файлами аудита используется следующий механизм:

- подключение файла журнала к базе «Ред База Данных» в качестве внешней таблицы;
- просмотр, поиск и фильтрация записей с использованием встроенных средств СУБД (построение и выполнение запросов к подключенной таблице).

Это позволяет применять средство анализа файлов аудита независимо от используемой операционной системы, а также наличия графической оболочки в ОС.

Подключение журнала производится с помощью SQL-запроса вида:

```
CREATE TABLE <table_name> EXTERNAL [FILE] '<filespec>'
ADAPTER 'fbtrace' [<col_defs>]
```

где:

- 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. табл 4.8).

Таблица 4.8 - Поля создаваемой таблицы аудита

Имя поля	Тип	Комментарий
event_time	TIMESTAMP	Время 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_att_id	INTEGER	Идентификатор соединения, в котором произошло событие
event_database	BLOB	База данных, с которой связано событие
event_user	BLOB	Пользователь, от имени которого произошло событие
event_protocol	BLOB	Протокол, по которому произошло подключение
event_hostname	BLOB	Имя хоста, с которого произошло подключение
event_type	CHAR(20)	Тип события
auth_factor_type	BLOB	Тип фактора, предъявленного при аутентификации
auth_factor_data	BLOB	Содержимое фактора, предъявленного при аутентификации
trans_id	INTEGER	Идентификатор транзакции
trans_opt	BLOB	Параметры транзакции
stmt_id	VARCHAR(16)	Идентификатор запроса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
stmt_sql	BLOB	Содержимое SQL-запроса
blr_data	BLOB SUB_TYPE BLR	Содержимое BLR-запроса
dyn_data	BLOB SUB_TYPE BLR	Содержимое DYN-запроса
stmt_access_path	BLOB	План выполнения запроса
stmt_params	BLOB	Параметры выполнения запроса
var_name	BLOB	Имя контекстной переменной
var_ns	BLOB	Пространство имен, для которого устанавливается контекстная переменная
var_value	BLOB	Значение контекстной переменной
svc_id	VARCHAR(16)	Идентификатор сервиса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
svc_name	BLOB	Имя вызываемого сервиса
svc_switches	BLOB	Параметры запуска сервиса
svc_query	BLOB	Тип запроса к сервису
proc_name	BLOB	Имя хранимой процедуры
proc_params	BLOB	Входные параметры хранимой процедуры
perf_time	INTEGER;	Время выполнения запроса
perf_info	BLOB	Статистика производительности запроса
row_fetched	INTEGER	Число выбранных строк
event_result	VARCHAR(12)	Результат события (успешно, не успешно, не санкционировано)

Примечание: 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(12)
);
```

4.8 Контроль доступа к системному каталогу

4.8.1 Фильтрация полей и записей

В «Ред База Данных» реализован механизм фильтрации полей и записей обычных таблиц, системного каталога и механизм предотвращения прямого изменения данных системного каталога. Для активизации режима фильтрации полей и записей необходимо установить параметр UseRecordFilter из файла конфигурации «Ред База Данных»:

```
UseRecordFilter = 0|1
```

Также необходимо добавить скрипт инициализации механизма фильтрации записей системного каталога filtering.sql в параметр InitScript:

```
InitScript = [FILE_1, .., FILE_N]
```

В этом параметре через запятую перечисляются файлы SQL-скриптов, которые будут выполнятся при создании базы данных (например, скрипт с метаданными для использования функционала полнотекстового поиска и др.).

При получении записи из системного каталога срабатывают соответствующие триггеры, в которых проверяются условия и возвращаются только те записи, которые удовлетворяют этому условию.

Запрещается прямое изменение таблиц системного каталога (разрешается только для внутренних запросов).

Возможно задание условий фильтрации полей и записей и для пользовательских таблиц.

Условие для фильтрации записей отношения задается с помощью ключевого слова RECFILTER после списка столбцов. <condition> представляет собой логическое выражение (Например sum = 10).

Условие для фильтрации по полям задается с помощью ключевого слова COLFILTER в списке столбцов.

Внимание! Фильтрация полей и записей не распространяется на поль-

зователя SYSDBA.

Задание условий для фильтрации полей и столбцов осуществляется при создании или изменении таблицы:

```
CREATE TABLE <имя таблицы>
[EXTERNAL [FILE] '<спецификация файла>']
(<определение столбца>
[, <определение столбца> |, <ограничение таблицы>] ...
[, COLFILTER <имя столбца> (<условие>), ...])
[, RECFILTER (<условие>)];
```

Установка условия фильтрации записей <condition> для таблицы:

```
ALTER TABLE <имя таблицы> SET RECFILTER (<условие>);
```

Пример:

```
ALTER TABLE TEST_TABLE SET RECFILTER (NEW.TEST_FIELD>5)
```

Здесь TEST_TABLE – таблица, содержащая поле TEST_FIELD типа INTEGER. Таким образом, пользователь после выполнения запроса:

```
SELECT * FROM TEST_TABLE
```

увидит только те записи, в которых значение поля TEST_FIELD1>5.

Внимание! Фильтрация записей осуществляется с помощью автоматически создаваемого триггера. Поэтому, если в условии фильтрации присутствует имя поля, то необходимо использовать оператор NEW.<имя поля>

Удаление условия фильтрации записей:

```
ALTER TABLE <имя таблицы> DROP RECFILTER;
```

Установка условия фильтрации полей <condition> для таблицы:

```
ALTER TABLE <имя таблицы> SET COLFILTER <имя столбца>
(<условие>);
```

Пример:

```
ALTER TABLE TEST_TABLE SET COLFILTER TEST_FIELD
(CURRENT_USER='TEST_USER');
```

В результате последующей выборки данных пользователями из таблицы TEST_TABLE, пользователю TEST_USER будут доступны данные из поля TEST_FIELD этой таблицы, а другим пользователям нет.

Удаление условия фильтрации полей для таблицы:

```
ALTER TABLE <имя таблицы> DROP COLFILTER <имя столбца>;
```

4.8.2 Правила доступа к системному каталогу

Таблица 4.9 - Права Доступа к системным таблицам

Отношение	Доступ к таблице	Доступ к записям	Скрытость полей
RDB$BACKUP_HISTORY	Доступна только SYSDBA	Не фильтровать	RDB$BACKUP_ID - не скрывать RDB$TIMESTAMP - не скрывать RDB$BACKUP_LEVEL - не скрывать RDB$GUID - не скрывать

Отношение	Доступ к таблице	Доступ к записям	Скрытость полей
			RDB$SCN - не скрывать RDB$FILE_NAME - не скрывать
RDB$CHARACTER_SETS	Доступна всем	Не фильтровать	RDB$CHARACTER_SET_NAME - не скрывать RDB$FORM_OF_USE - не скрывать RDB$NUMBER_OF_CHARACTERS - не скрывать RDB$DEFAULT_COLLATE_NAME - не скрывать RDB$CHARACTER_SET_ID - не скрывать RDB$SYSTEM_FLAG - не скрывать RDB$DESCRIPTION - не скрывать RDB$FUNCTION_NAME - не скрывать RDB$BYTES_PER_CHARACTER - не скрывать
RDB$CHECK_CONSTRAINTS	Доступна всем	Возвращать ограничения связанные с отношениями доступными пользователю (любые права на отношение DML или DDL).	RDB$CONSTRAINT_NAME - не скрывать RDB$TRIGGER_NAME - не скрывать
RDB$COLLATIONS	Доступна всем	Не фильтровать	RDB$COLLATION_NAME - не скрывать RDB$COLLATION_ID - не скрывать RDB$CHARACTER_SET_ID - не скрывать RDB$COLLATION_ATTRIBUTES - не скрывать RDB$SYSTEM_FLAG - не скрывать RDB$DESCRIPTION - не скрывать RDB$FUNCTION_NAME - не скрывать RDB$BASE_COLLATION_NAME - не скрывать RDB$SPECIFIC_ATTRIBUTES - не скрывать
RDB$DATABASE	Доступна всем	Не фильтровать	RDB$DESCRIPTION - не скрывать RDB$RELATION_ID - не скрывать RDB$SECURITY_CLASS - не скрывать RDB$CHARACTER_SET_NAME - не скрывать
RDB$DEPENDENCIES	Доступна всем	Возвращать все, где есть DML права на поле RDB$FIELD_NAME	RDB$DEPENDENT_NAME - не скрывать RDB$DEPENDED_ON_NAME - не скрывать RDB$FIELD_NAME - не скрывать RDB$DEPENDENT_TYPE - не скрывать RDB$DEPENDED_ON_TYPE - не скрывать
RDB$EXCEPTIONS	Доступна	Добавить поле	RDB$EXCEPTION_NAME - не

Отношение	Доступ к таблице	Доступ к записям	Скрытость полей
	всем	RDB$OWNER. В DDL будет доработана модель прав и выбирать записи, если пользователь имеет права на исключение.	скрывать RDB$EXCEPTION_NUMBER - не скрывать RDB$MESSAGE - не скрывать RDB$DESCRIPTION - не скрывать RDB$SYSTEM_FLAG - не скрывать
RDB$FIELDS	Доступна всем	Возвращать все, на которые есть права у пользователя.	RDB$FIELD_NAME - не скрывать RDB$QUERY_NAME - не скрывать RDB$VALIDATION_BLR – скрыть RDB$VALIDATION_SOURCE - не скрывать RDB$COMPUTED_BLR - скрыть RDB$COMPUTED_SOURCE - скрыть RDB$DEFAULT_VALUE - не скрывать RDB$DEFAULT_SOURCE - скрыть RDB$FIELD_LENGTH- не скрывать RDB$FIELD_SCALE - не скрывать RDB$FIELD_TYPE - не скрывать RDB$FIELD_SUB_TYPE - не скрывать RDB$MISSING_VALUE - не скрывать RDB$MISSING_SOURCE - не скрывать RDB$DESCRIPTION - не скрывать RDB$SYSTEM_FLAG - не скрывать RDB$QUERY_HEADER - не скрывать RDB$SEGMENT_LENGTH - не скрывать RDB$EDIT_STRING - не скрывать RDB$EXTERNAL_LENGTH- не скрывать RDB$EXTERNAL_SCALE - не скрывать RDB$EXTERNAL_TYPE- не скрывать RDB$DIMENSIONS- не скрывать RDB$NULL_FLAG - не скрывать RDB$CHARACTER_LENGTH - не скрывать RDB$COLLATION_ID - не скрывать RDB$CHARACTER_SET_ID - не скрывать RDB$FIELD_PRECISION - не скрывать
RDB$FIELD_DIMENSIONS	Доступна всем	Возвращать все связанные с отношениями на которые есть права у пользователя.	RDB$FIELD_NAME - не скрывать RDB$DIMENSION - не скрывать RDB$LOWER_BOUND - не скрывать RDB$UPPER_BOUND - не скрывать
RDB$FILES	Доступна всем	Возвращать по DDL правам на SHADOW.	RDB$FILE_NAME - не скрывать RDB$FILE_SEQUENCE - не

Отношение	Доступ к таблице	Доступ к записям	Скрытость полей
			скрывать RDB$FILE_START - не скрывать RDB$FILE_LENGTH - не скрывать RDB$FILE_FLAGS - не скрывать RDB$SHADOW_NUMBER - не скрывать
RDB$FILTERS	Доступна всем	Не фильтровать	RDB$FUNCTION_NAME - не скрывать RDB$DESCRIPTION - не скрывать RDB$MODULE_NAME - не скрывать RDB$ENTRYPOINT - не скрывать RDB$INPUT_SUB_TYPE - не скрывать RDB$OUTPUT_SUB_TYPE - не скрывать RDB$SYSTEM_FLAG - не скрывать
RDB$FORMATS	Доступна всем	Возвращать все связанные с отношениями, на которые у пользователя есть права (DDL или DML).	RDB$RELATION_ID - не скрывать RDB$FORMAT - не скрывать RDB$DESCRIPTOR - не скрывать
RDB$FUNCTIONS	Доступна всем	Добавить поле RDB$OWNER. В DDL будет доработана модель прав и выбирать записи, если пользователь имеет права на функции.	RDB$FUNCTION_NAME - не скрывать RDB$FUNCTION_TYPE - не скрывать RDB$QUERY_NAME - не скрывать RDB$DESCRIPTION - не скрывать RDB$MODULE_NAME - не скрывать RDB$ENTRYPOINT - не скрывать RDB$RETURN_ARGUMENT - не скрывать RDB$SYSTEM_FLAG - не скрывать
RDB$FUNCTION_ARGUMENTS	Доступна всем	Возвращать все связанные с функциями на которые у пользователя есть права.	RDB$FUNCTION_NAME - не скрывать RDB$ARGUMENT_POSITION - не скрывать RDB$MECHANISM - не скрывать RDB$FIELD_TYPE - не скрывать RDB$FIELD_SCALE - не скрывать RDB$FIELD_LENGTH - не скрывать RDB$FIELD_SUB_TYPE - не скрывать RDB$CHARACTER_SET_ID - не скрывать RDB$FIELD_PRECISION - не скрывать RDB$CHARACTER_LENGTH - не скрывать
RDB$GENERATORS	Доступна всем	Добавить поле RDB$OWNER. В DDL будет доработана модель прав и выбирать записи, если пользователь имеет	RDB$GENERATOR_NAME - не скрывать RDB$GENERATOR_ID - не скрывать RDB$SYSTEM_FLAG - не скрывать

Отношение	Доступ к таблице	Доступ к записям	Скрытость полей
		права на генератор.	RDB$DESCRIPTION - не скрывать
RDB$INDEX_SEGMENTS	Доступна всем	Возвращать все которые связаны с отношениями доступными текущему пользователю.	RDB$INDEX_NAME - не скрывать RDB$FIELD_NAME - не скрывать RDB$FIELD_POSITION - не скрывать RDB$STATISTICS - не скрывать
RDB$INDICES	Доступна всем	Возвращать все которые связаны с отношениями доступными текущему пользователю (по DDL и DML правам).	RDB$INDEX_NAME - не скрывать RDB$RELATION_NAME - не скрывать RDB$INDEX_ID - не скрывать RDB$UNIQUE_FLAG - не скрывать RDB$DESCRIPTION - не скрывать RDB$SEGMENT_COUNT - не скрывать RDB$INDEX_INACTIVE - не скрывать RDB$INDEX_TYPE - не скрывать RDB$FOREIGN_KEY - не скрывать RDB$SYSTEM_FLAG - не скрывать RDB$EXPRESSION_BLR - скрыть RDB$EXPRESSION_SOURCE - скрыть RDB$STATISTICS - не скрывать
RDB$LOG_FILES	Доступна только SYSDBA	Не фильтровать	RDB$FILE_NAME - не скрывать RDB$FILE_SEQUENCE - не скрывать RDB$FILE_LENGTH - не скрывать RDB$FILE_PARTITIONS - не скрывать RDB$FILE_P_OFFSET - не скрывать RDB$FILE_FLAGS - не скрывать
RDB$PAGES	Доступна только SYSDBA	Не фильтровать.	RDB$PAGE_NUMBER - не скрывать RDB$RELATION_ID - не скрывать RDB$PAGE_SEQUENCE - не скрывать RDB$PAGE_TYPE - не скрывать
RDB$PROCEDURES	Доступна всем	Возвращать все на которые есть права (DML и DDL).	RDB$PROCEDURE_NAME - не скрывать RDB$PROCEDURE_ID - не скрывать RDB$PROCEDURE_INPUTS - не скрывать RDB$PROCEDURE_OUTPUTS - не скрывать RDB$DESCRIPTION - не скрывать RDB$PROCEDURE_SOURCE - скрыть RDB$PROCEDURE_BLR - скрыть RDB$SECURITY_CLASS - не скрывать RDB$OWNER_NAME - не скрывать RDB$RUNTIME - не скрывать RDB$SYSTEM_FLAG - не скрывать RDB$PROCEDURE_TYPE - не скрывать

Отношение	Доступ к таблице	Доступ к записям	Скрытость полей
			RDB$VALID_BLR - не скрывать RDB$DEBUG_INFO - не скрывать
RDB$PROCEDURE_PARAMETERS	Доступна всем	Возвращать все которые связаны с процедурами доступными текущему пользователю.	RDB$PARAMETER_NAME - не скрывать RDB$PROCEDURE_NAME - не скрывать RDB$PARAMETER_NUMBER - не скрывать RDB$PARAMETER_TYPE - не скрывать RDB$FIELD_SOURCE - не скрывать RDB$DESCRIPTION - не скрывать RDB$SYSTEM_FLAG - недоступно пользователю RDB$DEFAULT_VALUE - не скрывать RDB$DEFAULT_SOURCE - скрыть RDB$COLLATION_ID - не скрывать RDB$NULL_FLAG - не скрывать RDB$PARAMETER_MECHANISM - не скрывать
RDB$REF_CONSTRAINTS	Доступна всем	Возвращать все связанные с отношениями доступными пользователю (по DDL и DML правам).	RDB$CONSTRAINT_NAME - не скрывать RDB$CONST_NAME_UQ - не скрывать RDB$MATCH_OPTION - не скрывать RDB$UPDATE_RULE - не скрывать RDB$DELETE_RULE - не скрывать
RDB$RELATIONS	Доступна всем	Возвращать все доступные текущему пользователю (по DDL и DML правам).	RDB$VIEW_BLR - скрыть RDB$VIEW_SOURCE - скрыть RDB$DESCRIPTION - не скрывать RDB$RELATION_ID - не скрывать RDB$SYSTEM_FLAG - не скрывать RDB$DBKEY_LENGTH - не скрывать RDB$FORMAT - не скрывать RDB$FIELD_ID - не скрывать RDB$RELATION_NAME - не скрывать RDB$SECURITY_CLASS - не скрывать RDB$EXTERNAL_FILE - не скрывать RDB$RUNTIME - не скрывать RDB$EXTERNAL_DESCRIPTION - не скрывать RDB$OWNER_NAME - не скрывать RDB$DEFAULT_CLASS - не скрывать RDB$FLAGS - не скрывать RDB$RELATION_TYPE - не скрывать
RDB$RELATION_CONSTRAINTS	Доступна всем	Возвращать все связанные с отношениями доступными	RDB$CONSTRAINT_NAME - не скрывать RDB$CONSTRAINT_TYPE - не скрывать

Отношение	Доступ к таблице	Доступ к записям	Скрытость полей
		пользователю (по DDL).	RDB$RELATION_NAME - не скрывать RDB$DEFERRABLE - не скрывать RDB$INITIALLY_DEFERRED - не скрывать RDB$INDEX_NAME - не скрывать
RDB$RELATION_FIELDS	Доступна всем	Возвращать все на которые есть права у пользователя.	RDB$FIELD_NAME - не скрывать RDB$RELATION_NAME - не скрывать RDB$FIELD_SOURCE - скрыть RDB$QUERY_NAME - не скрывать RDB$BASE_FIELD - не скрывать RDB$EDIT_STRING - не скрывать RDB$FIELD_POSITION - не скрывать RDB$QUERY_HEADER - не скрывать RDB$UPDATE_FLAG - не скрывать RDB$FIELD_ID - не скрывать RDB$VIEW_CONTEXT - не скрывать RDB$DESCRIPTION - не скрывать RDB$DEFAULT_VALUE - не скрывать RDB$SYSTEM_FLAG - не скрывать RDB$SECURITY_CLASS - не скрывать RDB$COMPLEX_NAME - не скрывать RDB$NULL_FLAG - не скрывать RDB$DEFAULT_SOURCE - не скрывать RDB$COLLATION_ID - не скрывать
RDB$ROLES	Доступна всем	Возвращать те которые назначены пользователю	RDB$ROLE_NAME - не скрывать RDB$OWNER_NAME - не скрывать RDB$DESCRIPTION - не скрывать RDB$SYSTEM_FLAG - не скрывать
RDB$SECURITY_CLASSES	Доступна только SYSDBA	Не фильтровать.	RDB$SECURITY_CLASS - не скрывать RDB$ACL - не скрывать RDB$DESCRIPTION - не скрывать
RDB$TRANSACTIONS	Доступна всем	Не фильтровать	RDB$TRANSACTION_ID — не скрывать RDB$TRANSACTION_STATE — не скрывать RDB$TIMESTAMP — не скрывать RDB$TRANSACTION_DESCRIPTION — не скрывать
RDB$TRIGGERS	Доступна всем	Возвращать только если пользователь имеет, имеет право на вставку, изменение и удаление записей отношения, а также изменение структуры таблицы.	RDB$TRIGGER_NAME - не скрывать RDB$RELATION_NAME - не скрывать RDB$TRIGGER_SEQUENCE - не скрывать RDB$TRIGGER_TYPE - не скрывать

Отношение	Доступ к таблице	Доступ к записям	Скрытость полей
			RDB$TRIGGER_SOURCE - скрыть RDB$TRIGGER_BLR - скрыть RDB$DESCRIPTION - не скрывать RDB$TRIGGER_INACTIVE - не скрывать RDB$SYSTEM_FLAG — не скрывать RDB$FLAGS - не скрывать RDB$VALID_BLR — не скрывать RDB$DEBUG_INFO - не скрывать
RDB$TRIGGER_MESSAGES	Доступна всем	Возвращать связанные с триггерами, на которые пользователь имеет права.	RDB$TRIGGER_NAME - не скрывать RDB$MESSAGE_NUMBER - не скрывать RDB$MESSAGE - не скрывать
RDB$TYPES	Доступна всем	Не фильтровать	RDB$FIELD_NAME - не скрывать RDB$TYPE - не скрывать RDB$TYPE_NAME - не скрывать RDB$DESCRIPTION - не скрывать RDB$SYSTEM_FLAG — не скрывать
RDB$USER_PRIVILEGES	Доступна всем	Возвращать все относящиеся к текущему пользователю	RDB$USER - не скрывать RDB$GRANTOR - не скрывать RDB$PRIVILEGE - не скрывать RDB$GRANT_OPTION - не скрывать RDB$RELATION_NAME - не скрывать RDB$FIELD_NAME - не скрывать RDB$USER_TYPE - не скрывать RDB$OBJECT_TYPE - не скрывать
RDB$VIEW_RELATIONS	Доступна всем	Возвращать только связанные с отношениями на которые есть права у пользователя.	RDB$VIEW_NAME — не скрывать RDB$RELATION_NAME - не скрывать RDB$VIEW_CONTEXT - не скрывать RDB$CONTEXT_NAME - не скрывать

Скрытие полей не производится только вне внутренних запросов (пользователь не имеет право изменять процедуру, соответственно, не видит ее исходный код и blr, но должен получить blr во время исполнения).

Скрытие полей производится на основании DDL-прав, например, пользователь имеет право на выборку из таблицы, но не имеет прав на ее изменение, соответственно, он сможет выбрать запись с вычисляемым полем, но посмотреть на его исходный код и blr — нет.

Таблицы мониторинга не фильтруются средствами Record Access, а возвращают только данные текущего подключения к базе.

4.9 Контроль целостности метаданных[22]

Контроль за целостностью метаданных в БД осуществляется с помощью утилиты mint (находится в подкаталоге bin/ каталога установки сервера). Эта утилита

22 См. примечание 11 (4.4.Идентификация и аутентификация13)

предназначена для извлечения и хеширования метаданных из баз данных, а также для проверки ранее полученного хеша метаданных. Таким образом, администратор может защитить структуру базы данных от изменений.

Утилита mint позволяет выбрать все метаданные из базы данных или только их часть, по заданной маске, хешировать их и сохранить в файл. Также утилита может проводить проверку текущего состояния метаданных в базе путем повторной выборки данных и сравнения результата с ранее сохраненным.

Утилита имеет следующие команды и опции:

- M <extract|check> - определяет режим работы утилиты;
- d - имя базы данных, из которой будет производиться извлечение данных
- u - имя пользователя для присоединения к БД;
- p - пароль пользователя для присоединения к БД;
- m - определяет маску для извлечения метаданных из БД;
- s - извлекать метаданные;
- r - хранилище с ключами шифрования;
- R - алгоритм шифрования для хранилища ключей;
- i - файл для сохранения|проверки хеша метаданных;
- I - название алгоритма цифровой подписи.

Существуют два режима работы утилиты – генерация контрольной суммы (extract) и проверка (check).

Если маска не задана, то выбираются все метаданные из базы. В маске можно использовать символ % - заменяет собой любое количество любых символов.

Например, генерация подписи для всех системных объектов (начинающихся с префикса RDB$) в базе данных security2.fdb:

```
mint -M extract -m RDB$% -d ../security2.fdb -u sysdba -p
masterkey -r test -R "Crypto Pro" -i sign -I AT_SIGNATURE
```

проверка подписи для этих объектов:

```
mint -M check -m RDB$% -d ../security2.fdb -u sysdba -p
masterkey -r test -R "Crypto Pro" -i sign -I AT_SIGNATURE
Signature verification complete successfully
```

Запуск утилиты без параметров, или с ключом -h выдает краткую справку о командах и опциях работы утилиты.

4.10 Контроль целостности файлов сервера[23]

Контроль за файлами сервера означает, что для всех критически важных файлов сервера (бинарные файлы, файлы конфигурации, база данных безопасности security2.fdb и т. д.) может быть вычислен и проверен хеш.

Файл с контрольными суммами (хеш-функциями) всех защищаемых файлов, а также файл конфигурации «Ред База Данных» должны быть защищены с помощью

23 См. примечание 13 (п. 4.4 «Идентификация и аутентификация»)

организационно-технических мер (например, запись на носителе только для чтения и т. д.). Файл с контрольными суммами поставляется вместе с дистрибутивом СУБД «Ред База Данных»

Для того, чтобы включить контроль целостности файлов сервера, необходимо указать имя файла, содержащего контрольные суммы (хеши) защищаемых файлов сервера в конфигурационном файле «Ред База Данных» firebird.conf. Имя этого файла задается параметром HashesFile. Если задано значение этого параметра, то каждый раз при запуске сервера происходит проверка целостности файлов сервера.

При загрузке сервер загружает все строки из файла хешей и проверяет содержащиеся в этих строках хеши. Файл хешей должен содержать строки вида:

```
<хеш> <алгоритм хеширования> <имя файла>
```

Если какой-либо из хешей не совпал, сервер делает соответствующую запись в журнале аудита и завершает работу.

В состав дистрибутива входит утилита hashgen (находиться в каталоге bin/ установки сервера). Администратор с помощью этой утилиты может пересоздать хеш отдельно взятого файла. Входные параметры утилиты – имя хешируемого файла, имя файла с хешами, алгоритм хеширования. Запуск утилиты без параметров выводит краткую справку по ее параметрам и доступным алгоритмам шифрования.

Также этой утилитой можно производить проверку ранее созданных хешей. Периодичность проверки определяется администратором.

Пример использования утилиты hashgen:

```
hashgen.exe generate CALG_GR3411 ../security2.fdb
>test.sign
```

- сгенерирована и сохранена в файл test.sign контрольная сумма для файла security2.fdb.

```
hashgen.exe check test.sign
../security2.fdb: Success
```

- контрольная сумма в файле test.sign совпадает с контрольной суммой исходного файла.

Приложение А. Описание таблиц мониторинга «Ред База Данных»

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$DATABASE	Сведения о базе данных, с которой выполнено соединение
MON$ATTACHMENTS	Сведения о текущих соединениях с базой данных
MON$TRANSACTIONS	Запущенные на выполнение транзакции
MON$STATEMENTS	Подготовленные к выполнению операторы
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$DATABASE

Таблица А.2 - Сведения о базе данных, с которой выполнено соединение

Идентификатор столбца	Тип данных	Описание
MON$DATABASE_NAME	VARCHAR(253)	Полный путь и имя первичного файла базы данных или псевдоним базы данных.
MON$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$OLDEST_TRANSACTION	INTEGER	Номер старейшей заинтересованной транзакции — OIT, Oldest Interesting Transaction.
MON$OLDEST_ACTIVE	INTEGER	Номер старейшей активной транзакции — OAT, Oldest Active Transaction.
MON$OLDEST_SNAPSHOT	INTEGER	Номер транзакции, которая была активной на момент старта транзакции OAT, — транзакция OST, Oldest Snapshot Transaction.
MON$NEXT_TRANSACTION	INTEGER	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 транзакцией только для чтения, read only, (значение 1) или для чтения и записи, read-write (0).

Идентификатор столбца	Тип данных	Описание
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 резервирование пространства.
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 — нормальное, 1 — заблокированное, 2 — слияние (объединение).
MON$REPLICATION_STATUS	SMALLINT	Текущее состояние репликации: 0 — основная база без репликации, 1 — основная база с репликацией.

MON$ATTACHMENTS

Таблица А.3 - Сведения о текущих соединениях с базой данных

Идентификатор столбца	Тип данных	Описание
MON$ATTACHMENT_ID	INTEGER	Идентификатор соединения.
MON$SERVER_PID	INTEGER	Идентификатор серверного процесса.
MON$STATE	SMALLINT	Состояние соединения: 0 — бездействующее, 1 — активное.
MON$ATTACHMENT_NAME	VARCHAR(253)	Строка соединения — 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$REMOTE_PROTOCOL	VARCHAR(8)	Имя удаленного протокола.
MON$REMOTE_ADDRESS	VARCHAR(253)	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	Флаг сборки мусора.
MON$REMOTE_PROCESS	VARCHAR(253)	Полный путь к файлу и имя программного файла, выполнившего данное соединение.
MON$STAT_ID	INTEGER	Идентификатор статистики.
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$TRANSACTIONS

Таблица А.4 - Описывает запущенные на выполнение транзакции

Идентификатор столбца	Тип данных	Описание
MON$TRANSACTION_ID	INTEGER	Идентификатор (номер) транзакции.

Идентификатор столбца	Тип данных	Описание
MON$ATTACHMENT_ID	INTEGER	Идентификатор соединения.
MON$STATE	SMALLINT	Состояние транзакции: 0 — бездействующая, 1 — активная.
MON$TIMESTAMP	TIMESTAMP	Дата и время старта транзакции.
MON$TOP_TRANSACTION	INTEGER	Идентификатор (номер) транзакции верхнего уровня.
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.5 - Подготовленные к выполнению операторы

Идентификатор столбца	Тип данных	Описание
MON$STATEMENT_ID	INTEGER	Идентификатор оператора.
MON$ATTACHMENT_ID	INTEGER	Идентификатор соединения.
MON$TRANSACTION_ID	INTEGER	Идентификатор транзакции.
MON$STATE	SMALLINT	Состояние оператора: 0 — бездействующий, 1 — активный.
MON$TIMESTAMP	TIMESTAMP	Дата и время старта оператора.
MON$SQL_TEXT	BLOB TEXT	Текст оператора на языке SQL.
MON$STAT_ID	INTEGER	Идентификатор статистики.

MON$CALL_STACK

Таблица А.6 - Обращения к стеку запросами хранимых процедур и триггеров

Идентификатор столбца	Тип данных	Описание
MON$CALL_ID	INTEGER	Идентификатор обращения.
MON$STATEMENT_ID	INTEGER	Идентификатор верхнего уровня оператора SQL— оператора, инициировавшего цепочку обращений.

Идентификатор столбца	Тип данных	Описание
MON$CALLER_ID	INTEGER	Идентификатор обращающегося триггера или хранимой процедуры.
MON$OBJECT_NAME	CHAR(31)	Имя объекта PSQL.
MON$OBJECT_TYPE	SMALLINT	Тип объекта PSQL (триггер или хранимая процедура).
MON$TIMESTAMP	TIMESTAMP	Дата и время старта обращения.
MON$SOURCE_LINE	INTEGER	Номер исходной строки оператора SQL, выполняющегося в настоящий момент.
MON$SOURCE_COLUMN	INTEGER	Номер исходного столбца оператора SQL, выполняющегося в настоящий момент.
MON$STAT_ID	INTEGER	Идентификатор статистики.

MON$IO_STATS

Таблица А.7 - 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 — оператор (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$PAGE_FETCHES	BIGINT	Количество загруженных в память (fetch) страниц базы данных.
MON$PAGE_MARKS	BIGINT	Количество отмеченных (mark) страниц базы данных.

MON$RECORD_STATS

Таблица А.8 - 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 — оператор (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$RECORD_DELETES	BIGINT	Количество удаленных записей (deleted records).
MON$RECORD_BACKOUTS	BIGINT	Количество возвращенных в базу данных записей (backed out records).

Идентификатор столбца	Тип данных	Описание
MON$RECORD_PURGES	BIGINT	Количество удаленных ненужных записей (purged records).
MON$RECORD_EXPUNGES	BIGINT	Количество вычищенных средствами сборки мусора записей (expunged records).

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 В.1.

Таблица В.1- Список системных таблиц «Ред База Данных»

Таблица	Содержание
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$DATABASE	Основные данные о базе данных
RDB$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$FUNCTIONS	Сведения о внешних функциях
RDB$FUNCTION_ARGUMENTS	Характеристики параметров внешних 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$PAGES	Сведения о страницах базы данных
RDB$PROCEDURE_PARAMETERS	Параметры хранимых процедур
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$BACKUP_HISTORY

Таблица хранит историю копирования базы данных при помощи утилиты nbackup.

Идентификатор столбца	Тип данных	Описание
RDB$BACKUP_ID	INTEGER	Присваиваемый системой

Идентификатор столбца	Тип данных	Описание
		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(253)	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$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.

Идентификатор столбца	Тип данных	Описание
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$COLLATION_ATTRIBUTES	SMALLINT	Указывает, дополнять ли при сравнении строки справа пробелами до максимальной величины или нет.
RDB$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$BASE_COLLATION_NAME	CHAR(31)	Имя базового порядка сортировки для данного порядка сортировки.
RDB$SPECIFIC_ATTRIBUTES	BLOB TEXT	Описание особых атрибутов.

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$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.

Идентификатор столбца	Тип данных	Описание
RDB$DEPENDENT_TYPE	SMALLINT	Идентифицирует тип описываемого объекта: 0 – таблица, 1 – представление, 2 – триггер, 3 – вычисляемый столбец, 4 – ограничение CHECK, 5 – процедура, 6 – выражение для индекса, 7 – исключение, 8 – пользователь, 9 –столбец, 10 – индекс.
RDB$DEPENDED_ON_TYPE	SMALLINT	Идентифицирует тип зависимого 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.

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$MESSAGE	VARCHAR(1021)	Текст сообщения в исключении.
RDB$DESCRIPTION	BLOB TEXT	Произвольное текстовое описание исключения.
RDB$SYSTEM_FLAG	SMALLINT	Признак: определено пользователем = 0; определено системой = 1 или выше.

RDB$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

Идентификатор столбца	Тип данных	Описание
		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$FIELD_TYPE	SMALLINT	Код типа данных для столбца: • 7 = SMALLINT, • 8 = INTEGER, • 12 = DATE, • 13 = TIME, • 14 = CHAR, • 16 = BIGINT, • 27 = DOUBLE PRECISION, • 35 = TIMESTAMP, • 37 = VARCHAR, • 261 = BLOB. Коды для DECIMAL и NUMERIC имеют тот же размер, что и целые типы, используемые для их хранения.
RDB$FIELD_SUB_TYPE	SMALLINT	Для типа данных BLOB задает подтип: • 0 – не определен,

Идентификатор столбца	Тип данных	Описание
		• 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(125)	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.
RDB$EXTERNAL_TYPE	SMALLINT	Тип данных поля, как он представляется во внешней таблице. • 7 = SMALLINT, • 8 = INTEGER,

Идентификатор столбца	Тип данных	Описание
		• 12 = DATE, • 13 = TIME, • 14 = CHAR, • 16 = BIGINT, • 27 = DOUBLE PRECISION, • 35 = TIMESTAMP, • 37 = VARCHAR, • 261 = BLOB, • 40 = CSTRING (завершаемый нулем текст).
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$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$FILE_NAME	VARCHAR(253)	Полный путь к файлу и имя вторичного файла базы данных в многофайловой базе данных или файла оперативной копии.
RDB$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$MODULE_NAME	VARCHAR(253)	Имя динамической библиотеки / совместно используемого объекта, где расположен код фильтра BLOB.
RDB$ENTRYPOINT	CHAR(31)	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.
RDB$SYSTEM_FLAG	SMALLINT	Признак: внешне определенный фильтр (т.е. определенный пользователем = значение 0, внутренне определенный = значение 1 или более)

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 TEXT	Отображение в виде BLOB столбцов и характеристик данных на момент, когда была создана запись формата.

RDB$FUNCTIONS

Сведения о внешних 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$MODULE_NAME	VARCHAR(253)	Имя модуля (библиотеки DLL), где расположен код функции.
RDB$ENTRYPOINT	CHAR(31)	Имя точки входа в библиотеке, где находится эта функция.
RDB$RETURN_ARGUMENT	SMALLINT	Номер позиции возвращаемого аргумента в списке параметров, соответствующем входным аргументам.
RDB$SYSTEM_FLAG	SMALLINT	Признак определения функции: определенная пользователем = 1, определенная системой = 0.

RDB$FUNCTION_ARGUMENTS

Характеристики параметров внешних 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$MECHANISM	SMALLINT	Признак — передается ли аргумент по значению (значение столбца 0), по ссылке (значение 1), через дескриптор (значение 2) или через дескриптор BLOB (значение 3).
RDB$FIELD_TYPE	SMALLINT	Число, задающее код типа данных, определенного для столбца: • 7 = smallint, • 8 = integer, • 12 = date, • 13 = time, • 14 = char, • 16 = bigint, • 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.

Идентификатор столбца	Тип данных	Описание
RDB$FIELD_LENGTH	SMALLINT	Длина аргумента в байтах. • smallint = 2, • integer = 4, • date = 4, • time = 4, • bigint = 8, • double precision = 8, • timestamp = 8, • blob_id = 8.
RDB$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$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$SYSTEM_FLAG	SMALLINT	Признак: значение 0 — генератор определен пользователем, значение 1 или выше — определен системой.
RDB$DESCRIPTION	BLOB	Произвольный текст примечания к генератору.

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.
RDB$INDEX_INACTIVE	SMALLINT	Указывает, является ли в настоящий момент индекс активным: 1 — неактивный, 0 — активный.
RDB$INDEX_TYPE	SMALLINT	В настоящий момент не используется.
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.

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

Идентификатор столбца	Тип данных	Описание
		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).
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	Значение по умолчанию на языке BLR.
RDB$DEFAULT_SOURCE	BLOB	Значение по умолчанию в исходном виде на языке SQL.
RDB$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$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

Идентификатор столбца	Тип данных	Описание
		системной таблице RDB$SECURITY_CLASSES, для применения ограничений управления доступом.
RDB$OWNER_NAME	VARCHAR(31)	Имя пользователя — владельца (создателя) процедуры.
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$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$MATCH_OPTION	CHAR(7)	Не используется. Текущим значением является FULL во всех случаях.
RDB$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$BASE_FIELD	CHAR(31)	Только для представления. Имя столбца из базовой таблицы.
RDB$EDIT_STRING	VARCHAR(125)	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

Идентификатор столбца	Тип данных	Описание
		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$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(253)	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 EFD	Произвольное примечание к внешнему файлу таблицы.
RDB$OWNER_NAME	VARCHAR(31)	Имя пользователя — владельца (создателя) таблицы или

Идентификатор столбца	Тип данных	Описание
		представления.
RDB$DEFAULT_CLASS	CHAR(31)	Класс безопасности по умолчанию. Применяется, когда новый столбец добавляется в таблицу.
RDB$FLAGS	SMALLINT	Внутренние флаги.
RDB$RELATION_TYPE	SMALLINT	Тип описываемого объекта: 0 — системная таблица или таблица, созданная пользователем, 1 — представление, 3 — таблица мониторинга.

RDB$ROLES

Определение ролей.

Идентификатор столбца	Тип данных	Описание
RDB$ROLE_NAME	VARCHAR(31)	Имя роли.
RDB$OWNER_NAME	VARCHAR(31)	Имя пользователя-владельца роли.
RDB$DESCRIPTION	BLOB TEXT	Произвольный текст примечания к роли.
RDB$SYSTEM_FLAG	SMALLINT	Системный флаг.

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 TEXT	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(1021)	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$TRIGGER_SEQUENCE	SMALLINT	Последовательность (позиция) триггера. Ноль обычно означает, что последовательность не задана.
RDB$TRIGGER_TYPE	SMALLINT	Событие, на которое вызывается триггер: • 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.
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

Идентификатор столбца	Тип данных	Описание
		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.
RDB$DEBUG_INFO	BLOB	Содержит отладочную информацию о переменных, используемых в триггере.

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$USER	CHAR(31)	Пользователь, которому предоставляется данное полномочие.
RDB$GRANTOR	CHAR(31)	Имя пользователя, предоставляющего полномочие.
RDB$PRIVILEGE	CHAR(6)	Привилегия, предоставляемая в полномочии: A — all (все привилегии), S — select (выборка данных), I — insert (добавление данных), D — delete (удаление строк), R — reference (внешний ключ), U — update (изменение данных).
RDB$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),

Идентификатор столбца	Тип данных	Описание
		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 (пользователь, процедура, представление и т.д.).
RDB$OBJECT_TYPE	SMALLINT	Идентифицирует тип объекта, к которому предоставляется привилегия.

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$VIEW_CONTEXT	SMALLINT	Псевдоним, используемый для ссылки на столбец представления. Имеет то же значение, что и псевдоним, используемый в самом тексте представления на языке BLR в операторе запроса этого представления.
RDB$CONTEXT_NAME	CHAR(31)	Текстовый вариант псевдонима, указанного в столбце RDB$VIEW_CONTEXT.

Приложение В. Настройка КриптоПро (на примере версии 3.0)

Настройка в операционной системе Windows.

Общие настройки

Необходимо установить дистрибутив КриптоПро. Далее настроить провайдер можно через Панель управления->КриптоПро CSP.

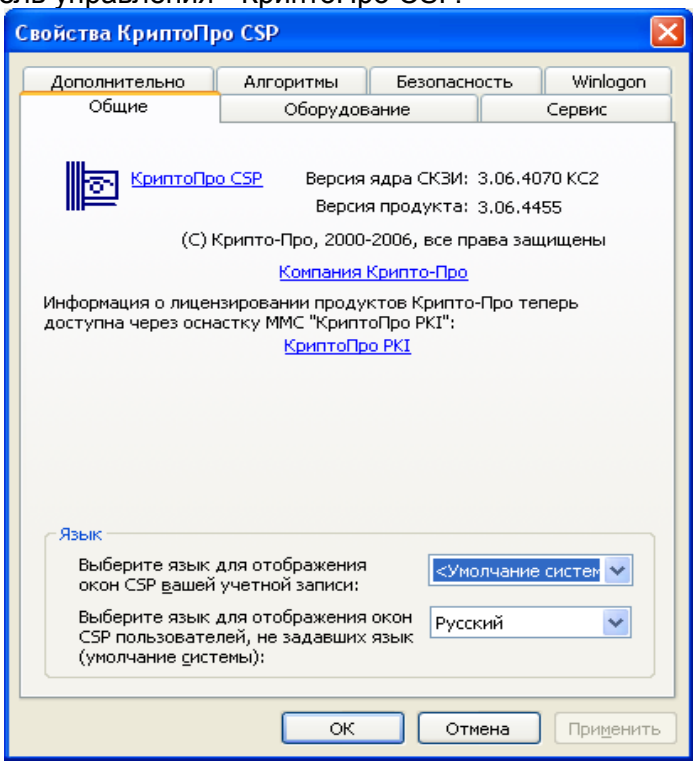


Рисунок 14 - Настройка КриптоПро через панель управления Windows

На вкладке алгоритмы настраиваются параметры для криптопровайдера (рисунок 15).

- GOST R 34.10-2001 Signature with Diffie_Hellman Key Exchange – название криптопровайдера (используется как параметр в утилитах hashgen, mint, а также для аутентификации на сервере и клиенте[24]);
- параметры алгоритма хеширования – название алгоритма хеширования по умолчанию, который используется, например, для создания/проверки ЭЦП.
- параметры алгоритма шифрования – название алгоритма шифрования по умолчанию;
- параметры алгоритма подписи – название алгоритма ЭЦП по умолчанию, аналогично использование псевдонима AT_SIGNATURE как значение параметра для утилиты mint;параметры алгоритма Диффи-Хелмана – название алгоритма, который будет использоваться по умолчанию для обмена сессионными ключами, аналогично использованию псевдонима AT_KEYEXCHANGE.

24 Планируется, что этот параметр будет вынесен отдельной настройкой в конфигурационный файл firebird.conf

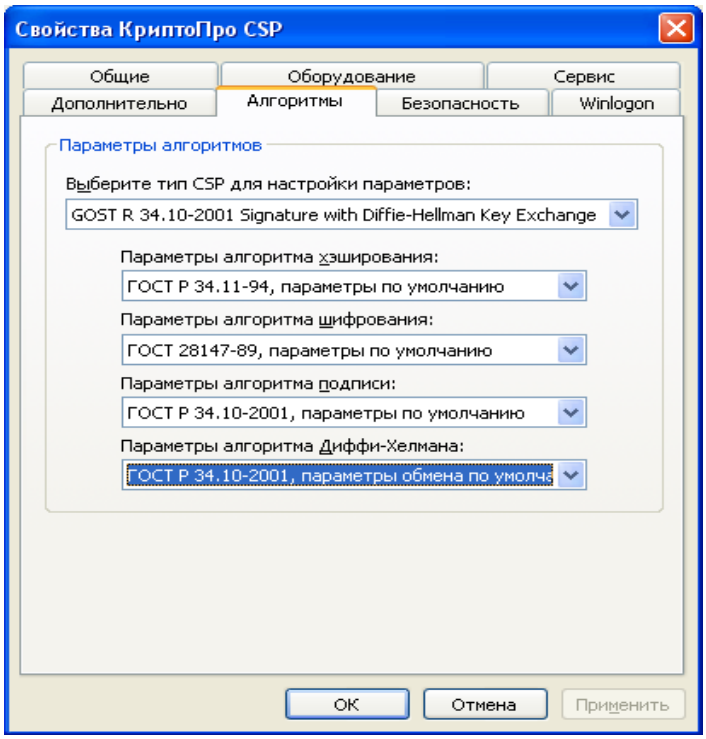


Рисунок 15 — Настройка алгоритмов

Генерирование ключей

Сессионные ключи генерируются плагином автоматически на время сессии. Для более безопасного обмена ими и аутентификации с использованием сертификата необходимо использовать секретные ключи.

Для генерирования ключей секретных можно воспользоваться средством КриптоПро <каталог_установки_КриптоПро\csptext.exe:

```
csptest -keyset -newkeyset -container <имя контейнера,
содержащего секретный ключ>
```

Контейнеры сохраняются на считывателях, соответственно, необходимо настроить считыватели на вкладке оборудование (рисунок 16.а).

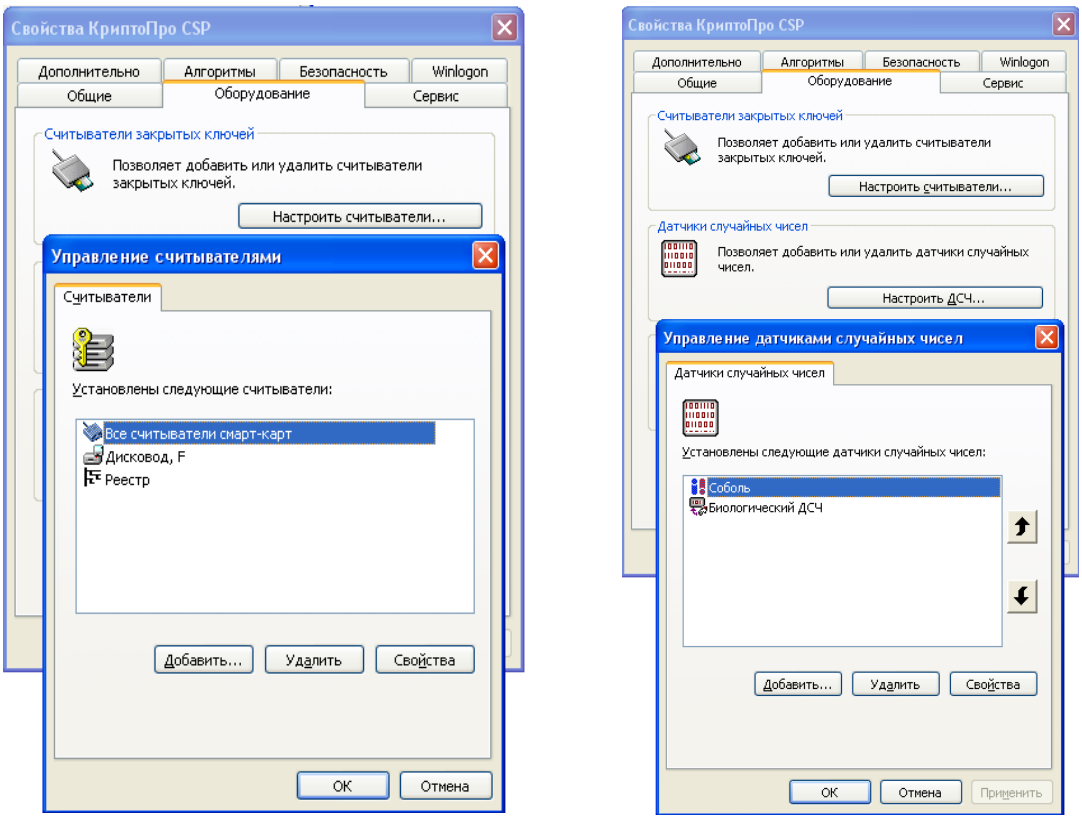
Кроме того, необходимо установить биологический датчик случайных чисел на вкладке оборудование (рисунок 16.б).

Необходимо создавать ключи без паролей, так как при аутентификации используется режим CRYPT_SILENT (чтобы не нарушать работу сервера).

При создании ключа следует учитывать разграничение доступа среди пользователей Windows, то есть, ключ созданный одним пользователем, не будет доступен другому. Если сервер работает в режиме службы (то есть от имени системного пользователя), то необходимо, чтобы владельцем контейнера с серверными ключами (параметр KeyRepository в firebird.conf) была сама операционная система. Для этого необходимо при создании серверного контейнера указать опцию -machinekeys:

```
csptest -keyset -newkeyset -container <имя контейнера,
содержащего секретный ключ> -machinekeys
```

HKEY_LOCAL_MACHINE\SOFTWARE\CryptoPro\Settings\USERS\SID_пользователя\Keys\имя_контейнера.



а *б*

Рисунок 16 - Настройка считывателей и датчиков случайных чисел

Пример создания ключевой пары:

```
csptest -keyset -newkeyset -container test_keyq
CSP (Type:75) v3.6.4070 KC2 Release Ver:3.06.4455
OS:Windows CPU:IA32 FastCode:R
EADY,ENABLED.
AcquireContext: OK. HCRYPTPROV: 1410480
GetProvParam(PP_NAME): Crypto-Pro GOST R 34.10-2001
Cryptographic Service Provider
Container name: "test_keyq"
Signature key is not available.
Attempting to create a signature key...
a signature key created.
Exchange key is not available.
Attempting to create an exchange key...
an exchange key created.
Everything is OK.
Keys in container:
  signature key
  exchange key
Total: SYS: 0.063 sec USR: 0.000 sec UTC: 24.968 sec
```

```
[ErrorCode: 0x00000000]
C:\Program Files\Crypto Pro\CSP>
```

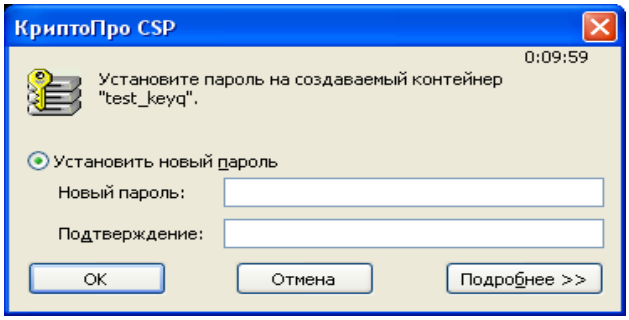


Рисунок 17 - Запрос установки пароля (желательно оставить пустым)

Получение сертификата

Для тестирования работы с пользовательскими сертификатами можно воспользоваться сертификационным центром КриптоПро (http://www.cryptopro.ru/certsrv/).

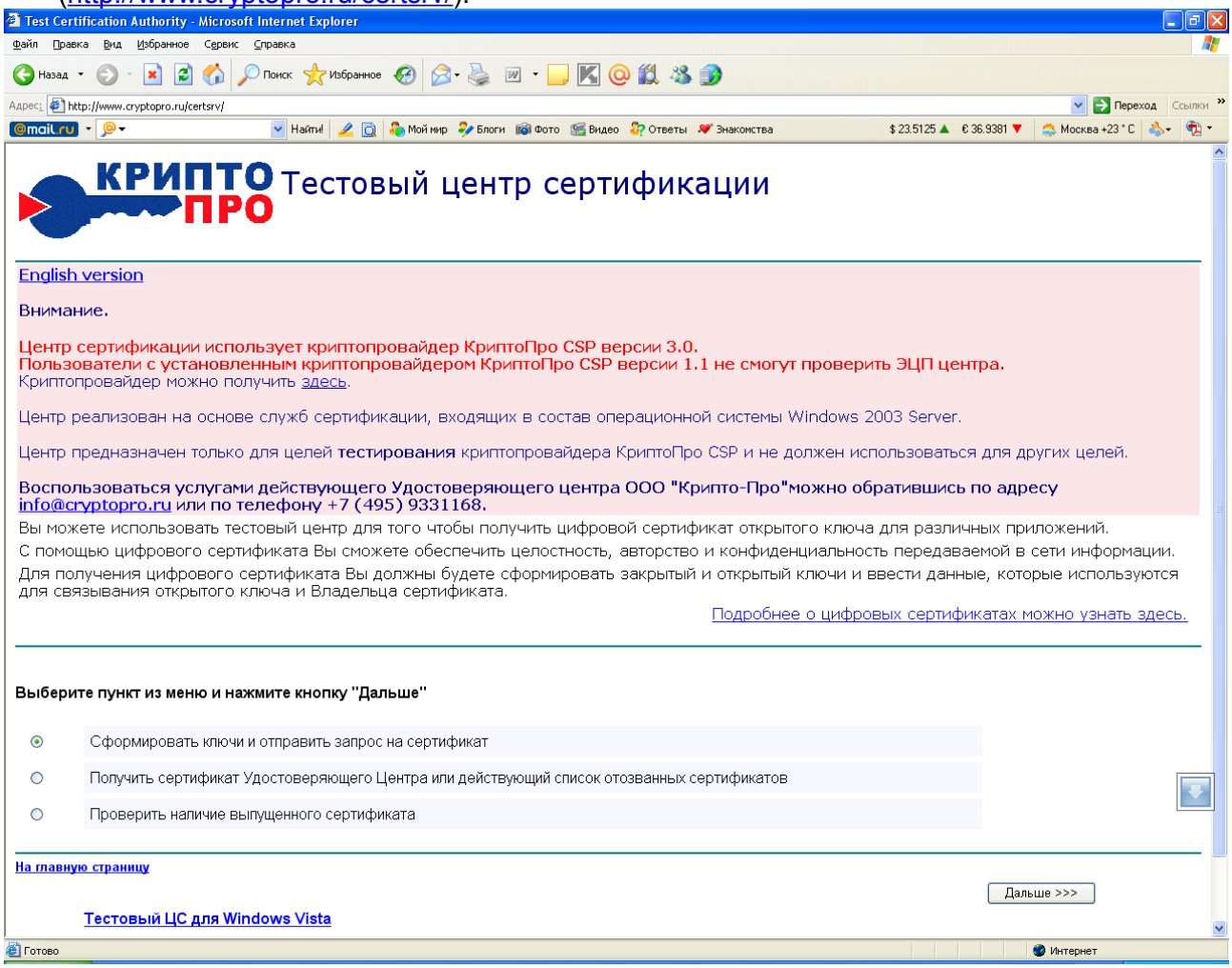


Рисунок 18 - Главная страница

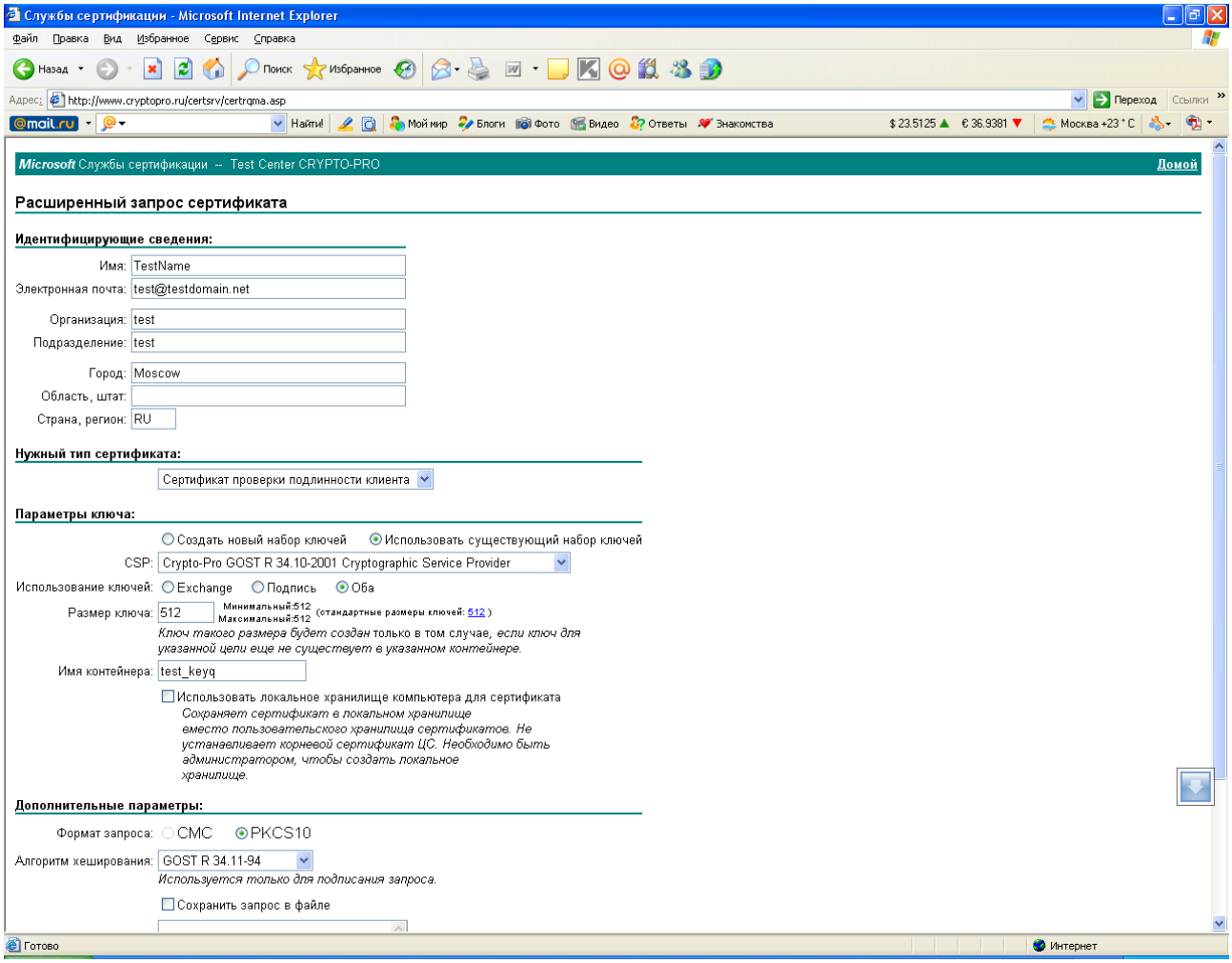


Рисунок 19 - Заполнение формы

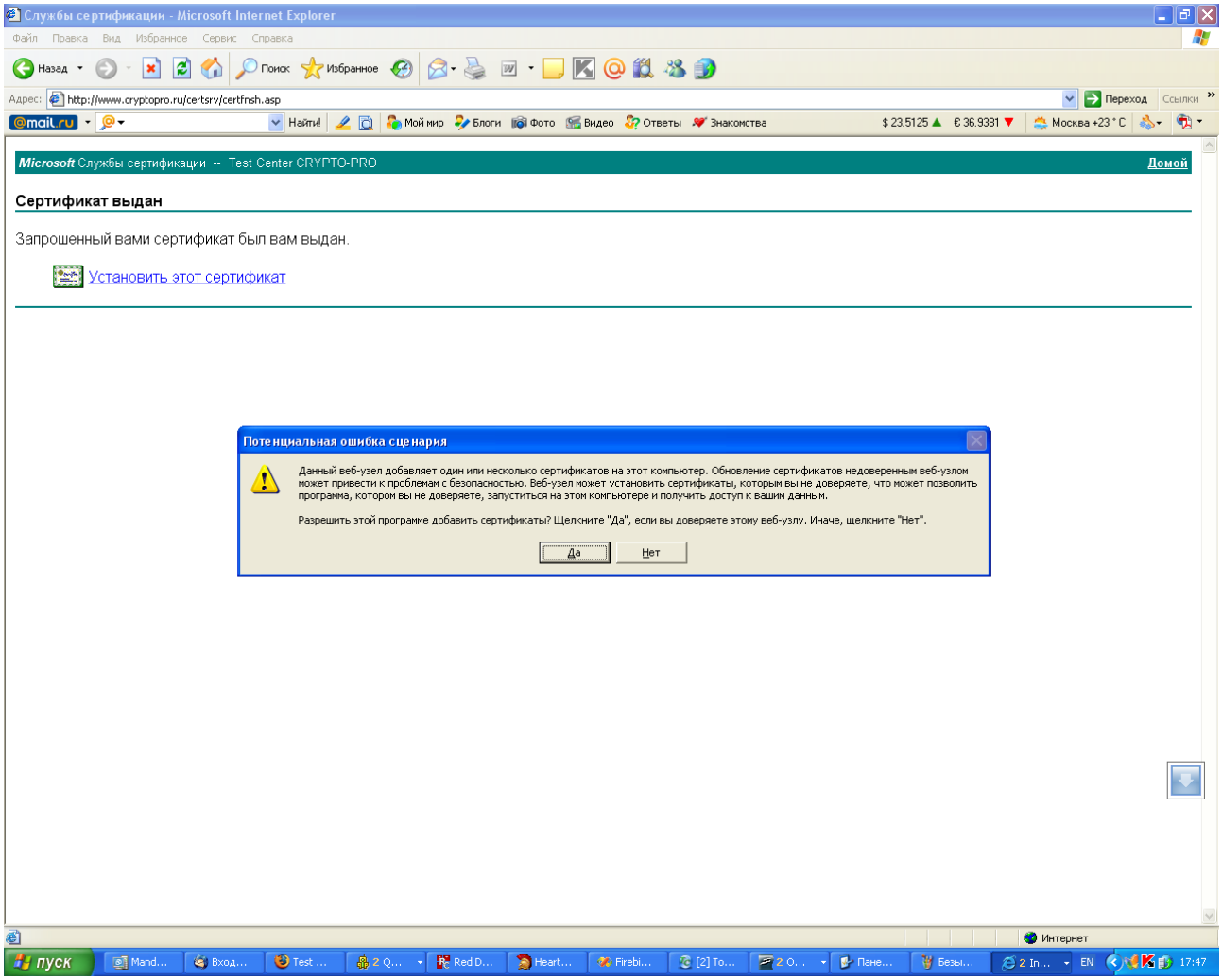


Рисунок 20 - Установка сертификата

После установки сертификат можно экспортировать его в кодировке BASE64: Свойства обозревателя->Содержание->Сертификаты->Имя сертификата

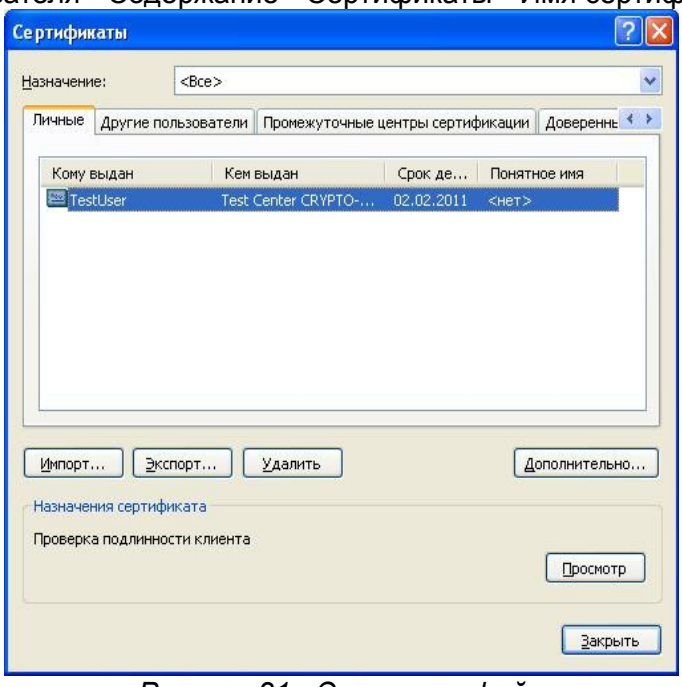


Рисунок 21 - Экспорт в файл

Настройка в операционной системе Linux.

Общие настройки

Для работы с КриптоПро версии 3.0. необходимо установить следующие пакеты:

- cpcsp-rdr-3.0.4-9.i386.rpm
- cpcsp-3.0.4-9.i386.rpm
- cpcsp-capilite-3.0.4-9.i386.rpm

По мере удовлетворения зависимостей, возможно. потребуется установить другие пакеты (из папки extra_pkg, optional). Также требуется установить библиотеку curl из папки extra_pkg, или же создать ссылку на уже установленную.

КриптоПро устанавливается в /opt/cprocsp.

В linux поддерживается два типа провайдеров Crypto-Pro GOST R 34.10-2001 KC1 CSP и Crypto-Pro GOST R 34.10-94 KC1 CSP.

После установки и настройки криптопровайдера необходимо настроить сервер «Ред База Данных» 2.5 на работу с криптоплагином. Для этого в конфигурационном файле firebird.conf указываются следующие параметры:

- CryptoPluginName — имя криптоплагина (значение по умолчанию wincrypt_plugin)
- KeyRepository — контейнер, содержащий ключевую пару серверных ключей. Эти ключи используются сервером для генерации сессионных ключей и других нужд сервера.

  Внимание! Контейнер, указанный в этом параметре, должен принадлежать тому пользователю, от имени которого запущен сервер.

- SymmetricMethod — алгоритм, используемый для симметричного шифрования. Можно указывать название алгоритма или его идентификатор в криптопровайдере (AlgId). Идентификатор алгоритма можно узнать утилитами mint и hashgen при запуске без параметров (параметр CryptoPluginName в конфигурационном файле должен быть задан) или утилитой КриптоПро csptest с параметрами -enum -info. Если используется неанглоязычные названия алгоритмов, их необходимо перекодировать их в 2-х байтный юникод, например

```
ГОСТ Р 34.11-94
примет вид:
%D0%93%D0%9E%D0%A1%D0%A2+%D0%A0+34.11-94
```

- HashMethod — название или идентификатор алгоритма хэширования. Способ представления аналогичен SymmetricMethod.
- ProviderName — название или идентификатор алгоритма криптопровайдера. Способ представления аналогичен SymmetricMethod.

Генерирование ключей

Для генерации ключей следует запустить утилиту:

```
/opt/cprocsp/bin/ia32/csptest -keyset -newkeyset
-container <контейнер>
```

Для создания ключей на жестком диске в ОС Linux контейнер должен указываться следующим образом: \\.\HDIMAGE\<имя контейнера>

Например, создание ключевой пары ”test”:

```
/opt/cprocsp/bin/ia32/csptest -keyset -newkeyset
-container '\\.\HDIMAGE\test'
```

Ключи в Linux хранятся в

```
 /var/opt/cprocsp/keys/user_name/container_name
```

Для использования ключей другим пользователем можно скопировать каталог с контейнером в папку пользователя, изменить права на этот каталог и файлы в нем, а также проверить файл /var/opt/cprocsp/users/user_name/local.ini на содержание там информации о ключе.

Например, если есть ключ /var/opt/cprocsp/keys/firebird/rdb.key, то необходимо, чтобы файл /var/opt/cprocsp/users/firebird/local.ini содержал, как минимум, следующие данные:

```
[Random]
RootRandomSeed = hex:
a6,16,3e,dd,a4,43,ca,46,2d,fe,a4,95,eb,bb,cb,95,7e,
1e,18,65,9a,fb,22,51,21,3a,43,73,13,2f,c3,1f,07,b8,60,1a,
72,6a,3d,46,bd,ee,f9,1d,c8,a9,83,9c

[KeyDevices]
LastShowDialogDate =
"00000009MBT92VP8G4E7L2BXX02DYDLBNAPE2D4F285PW8NP99" \
"H7AV9G89W4E1K90"

[KeyDevices\passwords\rdb.key]
shortcut = "HDIMAGE\\\\rdb.key.000\\0B32"
```

где последний блок показывает расположение ключа.

Создание сертификата

Для получения сертификата в ОС Linux с помощью тестового центра КриптоПро необходимо сформировать запрос на получение сертификата с помощью утилиты cryptcp из состава дистрибутива CryptoPro.

```
-creatrqst -dn <RDN> [-provtype <N>] [-provname <CSP>] [-
SMIME] [-nokeygen|-exprt] [-ex|-sg|-both] [-ku|-km] [-
cont <имя>] [-silent] [-askpin] [-certusage <OIDs>] [-
der] <имя файла>
```

При генерации запроса для клиентского сертификата необходимо указать заранее созданный клиентский контейнер с ключевой парой. Для этого используется опция -nokeygen. Контейнер может быть указан без спецификатора HDIMAGE. Например:

```
/opt/cprocsp/bin/ia32/cryptcp -creatrqst -cont
client_keys -nokeygen /home/tester/test.tmp -dn
'CN=tester'
```

Внимание: контейнер с ключевой парой, который используется для генерации запроса на получение клиентского сертификата, должен принадлежать тому же пользователю, для которого запрашивается сертификат.

Подробнее о командах и входных параметрах этой утилиты см. руководство по приложению командной строки CryptCP по адресу:

http://www.cryptopro.ru/CryptoPro/products/cryptcp/3-16/CryptCP.pdf

С помощью приведённой выше команды будет сформирован и сохранен в файл запрос на получение сертификата. Далее полученный запрос необходимо передать в тестовый центр КриптоПро по адресу:

http://www.cryptopro.ru/certsrv/certrqxt.asp

Примечание: Не все браузеры в ОС Linux могут поддерживать выгрузку сертификатов. Это зависит от настроек сервера центра сертификации.

После получения сертификата его можно выгрузить в файл средствами браузера.

Примечание: Выгрузку сертификата необходимо проводить в формате Base64

Приложение Г. Использование средства Userdump

Средство Userdump можно использовать для создания пользовательского дампа процесса, который завершается с исключением или перестает отвечать на запросы (зависает).

Создание файла дампа для сервера «Ред База Данных»

Когда сервер перестанет отвечать на запросы, в окне командной строки запустите userdump.exe (находиться в каталоге bin\userdump.exe установки сервера) и введите следующую команду:

```
userdump PID
```

В этой команде *PID* — это идентификатор процесса, который не отвечает. Для получения PID программы откройте диспетчер задач и перейдите на вкладку **Процесс**. На вкладке **Процесс** найдите приложение с названием rdbserver (в случае архитектуры SuperServer) или rdb_inet_server (в случае архитектуры Classic или SuperClassic). В случае архитекуры Classic, процессов с названием rdb_inet_server может быть несколько (в зависимости от количества подключений), поэтому необходимо сделать дампы всех запущенных процессов rdb_inet_server, указав разные имена дампов для каждого.

При запуске команды userdump *PID* создается DMP-файл, который может быть использован для выполнения отладки после неустранимого сбоя с помощью таких программ, как Windbg.exe.

Приложение Д. Тестирование системы безопасности

Утилита Stest, входящая в состав Ред Базы Данных 2.5, предназначена для тестирования средств защиты информации, которая проверяет все требования, которым должна удовлетворять система уровня 1Г. Она представляет собой совокупность тестов, моделирующих попытки несанкционированного доступа к защищенным объектам.

Консольное приложение STest имеет следующие входные параметры:

- –d <путь> - путь к каталогу, в котором будет размещаться тестовая база данных (по умолчанию предусмотреть для платформы Windows - создание базы данных в каталоге, определяемом переменной окружения TEMP, для Linux – в каталоге /tmp).

- –g <имя тест-кейса> - имя запускаемого тест-кейса, например: -g DML. По умолчанию производится запуск всех тест-кейсов. Возможные значения параметра приведены ниже, в таблице Д.1;

- –t <имя теста> - имя теста (опция). При указании опции производится запуск отдельного теста, например: -t IncorrectLogin. Имена всех тестов утилиты приведены ниже, в таблице Д.2;

- –h - опция вывод на консоль справочной информации по использованию консольной утилиты stest;

- –s <CryptoPluginName> <ServerRepository> - имя криптоплагина, реализующего интерфейс к криптографическим функциям, и ключевого контейнера сервера, содержащего криптографические ключи. Эти параметры являются опциональными, если они не указаны, то тесты на многофакторную аутентификацию и политики безопасности не будут запущены;

- –c <UserName> <Certificate> - имя пользователя и алиас сертификата. Имя пользователя, и владелец сертификата должны быть одинаковыми. Эти параметры являются опциональными, если они не указаны, то тесты на многофакторную аутентификацию не будут запущены;

- –p - пароль пользователя SYSDBA. При запуске утилиты потребуется ввести пароль для пользователя SYSDBA;

Если запустить утилиту без параметров, то в тестах будут использованы значения по умолчанию, при этом тесты на многофакторную аутентификацию и политики безопасности не будут запущены.

В процессе тестирования напротив каждого теста выводится результат его выполнения (OK или FAIL). Суммарные результаты выводятся после завершения тестирования в виде:

```
Error Details:
Summary:
        Executed Tests:         11
        Passed Tests:           9
      Failed Tests:         2
```

Раздел Error Details содержит подробную информацию по каждому тесту с результатом FAIL. Она включает ожидаемый и актуальный результаты теста, чтобы пользователь мог определить причину ошибки.

В таблице перечислены все тест-кейсы и значения параметра <-g> для их отдельного запуска.

Таблица Д.1: Тест-кейсы утилиты STest

Назначение тест-кейса	Значение параметра <-g>
тестирование DML-операций	DML
тестирование DDL-операций	DDL
тестирование доступа к сервисам	SERVICES
тестирование доступа к процедурам	PROCEDURES
тестирование доступа к BLOB-полям	BLOB
тестирование политик безопасности	POLICY
тестирование многофакторной аутентификации	FACTORS
тестирование операций с ролями	ROLES
тестирование подсистемы аудита	AUDIT

В таблице Д.2 приведены все тесты и их уникальные имена, используемые для запуска конкретного теста.

Таблица Д.2: Описание тестов утилиты STest

DML	
Описание теста	**Уникальное имя**
Соединение с неправильным логином	IncorrectLogin
Соединение с неправильным паролем	IncorrectPassword
Соединение с верными параметрами	CorrectLoginPass
Select * без прав	SelectAllFields
Select после назначения прав	SelectAfterGRANT
Select * после назначения прав	SelectAllAfterGRANT
Select после ограничения прав	SelectAfterREVOKE
Update полей таблицы без прав	UpdateTableFields
Update полей таблицы после выдачи прав	UpdateAfterGRANT
Update поля таблицы без прав	UpdateNotGRANTField
Update полей таблицы после отмены прав	UpdateAfterREVOKE

DDL	
Описание теста	**Уникальное имя**
Создание таблицы без прав	CrtTblWithoutGRANT
Изменение таблицы без прав	AltrTblWithoutGRANT
Удаление таблицы без прав	DropTblWithoutGRANT
Создание таблицы после назначения прав	CrtTblAfterGRANT
Изменение таблицы после назначения прав	AltrTblAfterGRANT
Удаление таблицы после назначения прав	DropTblAfterGRANT
Создание таблицы после отмены прав	CrtTblAfterREVOKE
Изменение таблицы после отмены прав	AltrTblAfterREVOKE
Удаление таблицы после отмены прав	DropTblAfterREVOKE
Создание процедуры без прав	CrtPrcWithoutGRANT
Изменение процедуры без прав	AltrPrcWithoutGRANT
Удаление процедуры без прав	DropPrcWithoutGRANT
Создание процедуры после назначения прав	CrtPrcAfterGRANT
Изменение процедуры после назначения прав	AltrPrcAfterGRANT
Удаление процедуры после назначения прав	DropPrcAfterGRANT
Создание процедуры после отмены прав	CrtPrcAfterREVOKE
Изменение процедуры после отмены прав	AltrPrcAfterREVOKE
Удаление процедуры после отмены прав	DropPrcAfterREVOKE
Создание роли без прав	CrtRolWithoutGRANT
Изменение роли без прав	AltrRolWithoutGRANT
Удаление роли без прав	DropRolWithoutGRANT
Создание роли после назначения прав	CrtRolAfterGRANT
Изменение роли после назначения прав	AltrRolAfterGRANT

Удаление роли после назначения прав	DropRolAfterGRANT
Создание роли после ограничения прав	CrtRolAfterREVOKE
Изменение роли после ограничения прав	AltrRolAfterREVOKE
Удаление роли после ограничения прав	DropRolAfterREVOKE
Создание генератора без назначения прав	CrtGenWithoutGRANT
Изменение генератора без назначения прав	AltrGenWithoutGRANT
Удаление генератора без назначения прав	DropGenWithoutGRANT
Создание генератора после назначения прав	CrtGenAfterGRANT
Изменение генератора после назначения прав	AltrGenAfterGRANT
Удаление генератора после назначения прав	DropGenAfterGRANT
Создание генератора после отмены прав	CrtGenAfterREVOKE
Изменение генератора после отмены прав	AltrGenAfterREVOKE
Удаление генератора после отмены прав	DropGenAfterREVOKE
Создание домена без назначения прав	CrtDmnWithoutGRANT
Изменение домена без назначения прав	AltrDmnWithoutGRANT
Удаление домена без назначения прав	DropDmnWithoutGRANT
Создание домена после назначения прав	CrtDmnAfterGRANT
Изменение домена после назначения прав	AltrDmnAfterGRANT
Удаление домена после назначения прав	DropDmnAfterGRANT
Создание домена после отмены прав	CrtDmnAfterREVOKE
Изменение домена после отмены прав	AltrDmnAfterREVOKE
Удаление домена после отмены прав	DropDmnAfterREVOKE
Создание исключения без прав	CrtExcWithoutGRANT
Изменение исключения без прав	AltrExcWithoutGRANT
Удаление исключения без прав	DropExcWithoutGRANT
Создание исключения после назначения прав	CrtExcAfterGRANT
Изменение исключения после назначения прав	AltrExcAfterGRANT
Удаление исключения после назначения прав	DropExcAfterGRANT
Создание исключения после отмены прав	CrtExcAfterREVOKE
Изменение исключения после отмены прав	AltrExcAfterREVOKE
Удаление исключения после отмены прав	DropExcAfterREVOKE
Создание последовательности без прав	CrtSqcWithoutGRANT
Изменение последовательности без прав	AltrSqcWithoutGRANT
Удаление последовательности без прав	DropSqcWithoutGRANT
Создание последовательности после назначения прав	CrtSqcAfterGRANT
Изменение последовательности после назначения прав	AltrSqcAfterGRANT
Удаление последовательности после назначения прав	DropSqcAfterGRANT
Создание последовательности после отмены прав	CrtSqcAfterREVOKE
Изменение последовательности после отмены прав	AltrSqcAfterREVOKE
Удаление последовательности после отмены прав	DropSqcAfterREVOKE
Создание теневой копии без прав	CrtSdwWithoutGRANT
Удаление теневой копии без прав	DropSdwWithoutGRANT
Создание теневой копии после назначения прав	CrtSdwAfterGRANT
Удаление теневой копии после назначения прав	DropSdwAfterGRANT
Создание теневой копии после отмены прав	CrtSdwAfterREVOKE
Удаление теневой копии после отмены прав	DropSdwAfterREVOKE
Создание представления без прав	CrtViewWithoutGRANT
Удаление представления без прав	DropViewWithoutGRANT
Создание представления после назначения прав	CrtViewAfterGRANT
Удаление представления после назначения прав	DropViewAfterGRANT
Создание представления после отмены прав	CrtViewAfterREVOKE
Удаление представления после отмены прав	DropViewAfterREVOKE

SERVICES	
Описание теста	**Уникальное имя**
Добавление пользователя без прав	AddUserWithoutGRANT
Добавление пользователя после назначения прав	AddUserAfterGRANT
Добавление пользователя после ограничения прав	AddUserAfterREVOKE

Изменение пользователя без назначения прав	ModifyUsrWithoutGRANT
Изменение пользователя после назначения прав	ModifyUsrWithSECADRol
Удаление пользователя без назначения прав	DeleteUsrWithoutGRANT
Удаление пользователя после назначения прав	DeleteUsrAfterGRANT
Удаления пользователя после ограничения прав	DeleteUsrAfterREVOKE
Получение списка пользователей без назначения прав	GetUsrLstWithoutGRANT
Получение списка пользователей после назначения прав	GetUsrLstAfterGRANT
Получение статистики базы данных без назначения прав	GetDBStatWithoutGRANT
Получение статистики базы данных после назначения прав	GetDBStatAfterGRANT
Изменение свойств базы данных без назначения прав	SetDBPropWithoutGRANT
Изменение свойств базы данных после назначения прав	SetDBPropAfterGRANT
Создание резервной копии БД без назначения прав	BackingWithoutGRANT
Создание резервной копии БД после назначения прав	BackingAfterGRANT
Восстановление базы данных без назначения прав	RestoreWithoutGRANT
Восстановление базы данных после назначения прав	RestoreAfterGRANT
Исправление БД без назначения прав	RepairDBWithoutGRANT
Исправление БД после назначения прав	RepairDBAfterGRANT

PROCEDURES	
Описание теста	**Уникальное имя**
Выполнение процедуры без назначения прав	ExecProcWithoutGRANT
Выполнение процедуры с правами пользователя-владельца	AuthidOwnerProcExec
Выполнение процедуры с правами вызывающего пользователя	AuthidCallerProcExec
Выполнение процедуры после назначения прав	ProcExecAfterGRANT
Создание процедуры без прав на вставку	CrtPrcWithOutInsGrnt
Создание процедуры с правами на вставку	ExecPrcAfterGRANTExec

BLOB	
Описание теста	**Уникальное имя**
Доступ к blob без выборки	OpenBlobWithoutSelect
Доступ к blob поле выборки	OpenBlobAfterSelect
Вставка идентификатора blob в таблицу	InsertBlobIDIntoTable
Обновление идентификатора blob в таблице	UpdateBlobIDInTable
Попытка вставки blob с использованием запроса insert … select	InsertSelectBlobQuery

POLICY	
Описание теста	**Уникальное имя**
Назначение политики DEFAULT	AssignDefaultPolicy
Проверка параметра LegacyHash	LegacyHashParameter
Тест на неудовлетворяющую политике длину пароля	UnCompliantPasLength
Тест на удовлетворяющую политике длину пароля	CompliantPasLength
Тест на неудовлетворяющую политике сложность пароля (регистр)	UnCompliantDiffCase
Тест на удовлетворяющую политике сложность пароля (регистр)	CompliantDiffCase
Тест на неудовлетворяющую политике сложность пароля (цифры)	UnCompliantNeedDigit
Тест на неудовлетворяющую политике сложность пароля (цифры)	CompliantNeedDigit
Тест на неудовлетворяющую политике сложность пароля (буквы)	UnCompliantNeedChar
Тест на удовлетворяющую политике сложность пароля (буквы)	CompliantNeedChar
Тест на проверку ограничения политикой количества	SessionsNumControl

одновременных сессий	
Тест на проверку требуемого политикой разрыва между повторяющимися паролями	PswUniqueCountControl
Тест на проверку максимального количества ошибок при аутентификации	`FailedCountControl`

FACTORS	
Описание теста	**Уникальное имя**
Попытка аутентификации по фактору пароль, требуемые: пароль и сертификат	MFConn_P_Policy_PC
Попытка аутентификации по факторам пароль и сертификат, требуемые: пароль и сертификат	MFConn_PC_Policy_PC
Попытка соединения с паролем в многофакторном режиме	MFConnectWithPW
Тест на доверительную аутентификацию	TrustedAuthentication
Попытка аутентификации в обычном режиме, при требуемом многофакторном	NotMFConnButNeedMF

ROLES	
Описание теста	**Уникальное имя**
Назначение прав на несуществующую роль	GrantToNotExistRole
Подключение с ролью, которая не была назначена пользователю	ConnWithNonGrantRole
Тест на перекрестную ссылочность между ролями	RoleCrossLinking
Тест на проверку прав роли на Update	ConnRoleGrantUpdate
Тест на проверку передачи прав от роли к роли	GrantInsRoleToUpdRole
Ограничение прав роли, отмена прав роли от роли	RevokeRoleFromRole
Тест на кумулятивное действие ролей	RolesGrantsCumulating
Подключение с указанием роли	ConnectWithRoleUsing
Создание роли и пользователя с одинаковыми именами	EqualRoleAndUserNames

AUDIT	
Описание теста	**Уникальное имя**
Логирование события успешной аутентификации	CorrectDataConnect
Логирование события неуспешной аутентификации	IncorrDataConnect
Логирование отсоединения от БД	DisconnectFromBase
Логирование выполнения хранимой процедуры	StoredProcExecute
Логирование запуска транзакции	StartTransaction
Логирование запуска транзакции	StartTransaction2
Логирование подтверждения транзакции	CommitTransaction
Логирование отката транзакции	RollBackTransaction
Логирование события присоединения к сервису	Attach_Service
Логирование события запуска сервиса	Start_Service
Логирование события отсоединения от сервиса	Detach_Service
Логирование события запроса к сервису	Query_Service1
Логирование события запроса к сервису	Query_Service2
Логирование события подготовки запроса на выборку	SQL_Select_Prepare
Логирование события выполнения запроса на выборку	SQL_Select_Execute
Логирование события освобождения запроса на выборку	SQL_Select_Free
Логирование события подготовки запроса на вставку	SQL_Insert_Prepare
Логирование события выполнения запроса на вставку	SQL_Insert_Execute
Логирование события подготовки запроса на вставку	SQL_Insert_Free
Логирование события подготовки запроса на удаление	SQL_Delete_Prepare
Логирование события выполнения запроса на удаление	SQL_Delete_Execute
Логирование события освобождения запроса на удаление	SQL_Delete_Free
Логирование события подготовки запроса на обновление	SQL_Update_Prepare
Логирование события выполнения запроса на	SQL_Update_Execute

обновление	
Логирование события освобождения запроса на обновление	SQL_Update_Free
Логирование события подготовки DDL-запроса	SQL_DDL_Prepare
Логирование события выполнения DDL-запроса	SQL_DDL_Execute
Логирование события освобождения DDL-запроса	SQL_DDL_Free
Логирование события выполнения запроса без прав	SQL_Without_GRANT
Логирование событие установки значения контекстной переменной	SetContextVariable
Логирование события компилирования BLR-запроса	BLR_Compilation
Логирование события выполнения BLR-запроса	BLR_Execution
Логирование события выполнения DYN-запроса	DYN_Execution
Тест на исключение из лога запросов на выборку (SELECT)	ExcludeSelect
Тест на исключение из лога системных данных(RDB$)	ExcludeSystemData
Тест на включение в лог только DDL-забросов	IncludeOnlyDDL
Тест на включение в лог только запросов на вставку (INSERT)	IncludeOnlyInsert