РЕД База Данных

Вариант исполнения РС.РБД.5

Руководство администратора

46.29926343.502120-01 92 2

Содержание

Глава 1

Общие сведения о программе

1.1 Назначение программы

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

1.2 Функциональное назначение

СУБД РЕД База Данных предоставляет пользователям следующие возможности:

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

1.3 Минимальный состав аппаратных средств

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

- Процессор с поддержкой архитектуры x86_64, 1.6ГГц, 2 ядра.
- Оперативная память от 4Гб.
- Запоминающее устройство объемом не менее 21Гб.
- Клавиатура 101/102-х клавишная рус/лат.
- Сетевая карта с поддержкой Ethernet.

1.4 Минимальный состав программных средств

Для установки и эксплуатации СУБД в ОС семейства Windows не требуется установки дополнительного программного обеспечения. Для ОС семейства Linux необходимы библиотеки: glibc 2.17 и старше, libstdc++ от gcc 10.2 и старше.

1.5 Требования к персоналу, среде эксплуатации и внешним мерам безопасности

Администратор СУБД должен иметь:

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

Безопасная эксплуатация СУБД предполагает, что:

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

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

1.6 Режим работы СУБД РЕД База Данных

СУБД РЕД База Данных поддерживает единственный режим работы — штатный.

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

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

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

1.7 Настройка функций безопасности среды функционирования средства

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

Для обеспечения безопасной работы СУБД в ОС семейства Linux необходимо:

- Установить разрешения на файлы: все файлы и каталоги, используемые СУБД, должны иметь соответствующие разрешения. Владельцем файлов должен быть системный пользователь `Firebird`. Серверный процесс СУБД должен иметь полный доступ к чтению и записи файлов базы данных. Рекомендуемые разрешения: 600 для файлов базы данных и 700 для каталогов. В процессе эксплуатации разрешения и права доступа рекомендуется устанавливать

через делегирование прав пользователям ОС добавлением/исключением в группу `Firebird` хостовой ОС.

- Настроить брендмауэр: должны быть разрешены входящие соединения с сервером СУБД только из надежных источников. Порт, используемые СУБД по умолчанию (3050), должен быть открыт и доступен.
- Аутентифицировать пользователей: не используйте для подключения к серверу СУБД учетную запись суперадминистратора `root` хостовой ОС. Каждый пользователь СУБД должен пройти успешную авторизацию в хостовой ОС с использованием встроенных механизмов ОС, а все его действия должны логироваться.

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

1.8 Лимиты СУБД РЕД База Данных 5.0

Текущие лимиты СУБД РЕД База Данных 5.0:

- Размер базы данных без табличных пространств - 128ТБ;
- Максимальное количество табличных пространства - 254;
- Максимальный размер внешнего файла - неограничен;
- Максимальный размер страницы - 32Кб;
- Максимальное количество таблиц - 32768;
- Максимальный размер одной таблицы - $\sim$ 18ТБ;
- Максимальное число строк в одной таблице - 2^{40};
- Максимальный размер записи без `BLOB` - 64Кб;
- Максимальный размер `BLOB` - 32Гб;
- Максимальный размер поля - 32Гб;
- Максимальное количество столбцов в таблице - 16384;
- Максимальное число индексов на таблицу - 800;
- Максимальный размер ключа индекса - $\sim$ 8Кб (1/4 от размера страницы);
- Максимальный размер запроса - 10Мб;
- Максимально допустимое число транзакций - $2.8 * 10^{14}$.

Глава 2

Миграции

2.1 Переход с РЕД Базы Данных 3 на 5.0

1. Обновите версию РЕД Базы Данных 3 до последней.
2. Сделайте бэкап баз данных на РЕД Базе Данных 3:

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

12. Если в `firebird.conf` в параметре `UserManager` использовался плагин `Multifactor_Manager`, то в новой версии его надо заменить на `GostPassword_Manager`.
13. Перенести параметры конфигурации из файла `fbjava.yaml` в `databases.conf`.
14. Если используются UDF - задать параметр конфигурации:

```
UDFAccess = Restrict UDF
```

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

2.2 Проблемы совместимости РЕД Базы Данных 3 и 5

1. *Устаревание UDF функций*

   При восстановлении базы данных с версии 3.0 на версию 5.0 может возникнуть ошибка, если в базе данных использовались устаревшие UDF функции:

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

   Поддержка внешних UDF функций (реализованных в динамических библиотеках и определенных в базе данных с помощью функций оператора `DECLARE FUNCTION`) в версии 5.0 устарела. Теперь по умолчанию для параметра `UdfAccess` в `firebird.conf` задано значение `NONE`. UDF-библиотеки `ib_udf` и `fbudf` изъяты из дистрибутива. Конечно, вы можете реализовать свою собственную динамическую библиотеку (или использовать библиотеки из предыдущей версии РЕД Базы Данных), настроить конфигурацию и работать с устаревшими функциями. Но этого делать не рекомендуется — вам лучше переключиться на использование функций UDR или PSQL. Чтобы переход стал проще, в дистрибутив включен скрипт `udf_replace.sql`, выполняющий соответствующую замену устаревших функций. К этим функциям относятся:

ADDDAY	ADDDAY2	ADDHOUR
ADDMILLISECOND	ADDMINUTE	ADDMONTH
ADDSECOND	ADDWEEK	ADDYEAR
DIV	DNULLIF	DNVL
DOW	DPOWER	GETEXACTTIMESTAMP
GETEXACTTIMESTAMPUTC	I64NULLIF	I64NVL

(разрыв таблицы)

(разрыв таблицы)

I64ROUND	I64TRUNCATE	INULLIF
INVL	ISLEAPYEAR	LTRIM
ROUND	RTRIM	SDOW
SNULLIF	SNVL	SRIGHT
STRING2BLOB	STRLEN	SUBSTR
SUBSTRLEN	TRUNCATE	UDF_FRAC или FRAC

Просто запустите скрипт:

```
isql -user sysdba -pas masterkey -i udf_replace.sql <база данных>
```

и UDF, распространяемые в `ib_udf` и `fbudf`, будут заменены соответствующими новыми функциями.

2. *Изменения в DDL и DML из-за поддержки часовых поясов*

   Поддержка часовых поясов вносит некоторые изменения в DDL и DML, которые могут повлиять на совместимость с существующими базами данных и приложениями.

   - Синтаксис для объявления типов данных `TIMESTAMP` и `TIME` был расширен, чтобы включить аргументы, определяющие, должны ли столбец, домен, параметр или переменная быть определены с установкой часового пояса или без него:

     ```
     TIME [ {WITHOUT|WITH} TIME ZONE ]
     TIMESTAMP [ {WITHOUT|WITH} TIME ZONE ]
     ```

     По умолчанию используется аргумент `WITHOUT TIME ZONE`.

   - В версии 5.0 переменные `CURRENT_TIME` и `CURRENT_TIMESTAMP` изменены: теперь они возвращают значения типа `TIME WITH TIME ZONE` и `TIMESTAMP WITH TIME ZONE` с часовым поясом, установленным часовым поясом сеанса. В предыдущих версиях `CURRENT_TIME` и `CURRENT_TIMESTAMP` возвращали соответствующие типы в соответствии с системными часами, то есть без какого-либо часового пояса.

     Переменные `LOCALTIMESTAMP` и `LOCALTIME` теперь заменяют прежние функциональные возможности `CURRENT_TIMESTAMP` и `CURRENT_TIME` соответственно.

3. *Сокращенное преобразование неявных литералов даты/времени не поддерживается*

   Синтаксис сокращенного преобразования типов, используемый вместе с неявными литералами даты/времени, такой как, например:

   ```
   TIMESTAMP 'NOW'
   DATE 'TODAY'
   DATE 'YESTERDAY'
   ```

   мог привести к неожиданным результатам:

   - в хранимых процедурах и функциях вычисление этих выражений будет происходить во время компиляции, а не во время вызова процедуры или функции, сохраняя результат в BLR и извлекая это устаревшее значение во время выполнения;
   - в DSQL вычисление этих выражений происходит во время подготовки запросов, а не на каждой итерации оператора, как можно было бы ожидать при правильном использовании неявных литералов даты/времени. Разница во времени между подготовкой и выполнением оператора может быть слишком мала, чтобы обнаружить проблему. Пользователи могли быть введены в заблуждение, полагая, что выражение вычисляется на каждой итерации оператора во время выполнения, хотя на самом деле это происходит во время подготовки.

Если что-то вроде `TIMESTAMP NOW` использовалось в SQL запросах в коде приложения или в PSQL, возникнет проблема совместимости с Ред Базой Данных 5.0. Теперь движок будет выдавать ошибку.

В тоже время сокращенное преобразование явных литералов, таких как `DATE 2019.02.20` допустимо. Также преобразование `CAST(NOW AS TIMESTAMP)` продолжает работать как раньше.

4. *Стартовое значение последовательности*

   До версии 5.0 последовательности (`generator/sequence`) создавались с текущим значением равным стартовому значению (или 0 по умолчанию).

   Следующим значением последовательности со стартовым значением 0 и приращением 1 было 1.

   В версии 5.0 последовательности создаются с текущим значением равным стартовому минус инкремент. И начальным значением по умолчанию является 1 (а не 0).

   То есть результат оператора `NEXT VALUES FOR` для последовательности со стартовым значением 100 и приращением 10 будет 100 (а не 110, как было раньше). Аналогично функция `GEN_ID(SEQ, 1)` возвратит 91 (а не 101, как было раньше).

5. *Совместимость с новыми типами данных*

   Параметр `DataTypeCompatibility` задает уровень совместимости, определяющий, какие типы данных доступны клиентскому API. В настоящее время доступны два варианта: 3.0 и 2.5. Режим эмуляции 3.0 скрывает типы данных, появившиеся после версии 3.0, а именно `DECIMAL` и `NUMERIC` с точностью 19 и выше, `DECFLOAT`, `TIME WITH TIME ZONE`, `TIMESTAMP WITH TIME ZONE`. Соответствующие значения возвращаются через типы данных, поддерживаемые версией 3.0. Режим эмуляции 2.5 преобразует ещё и тип данных `BOOLEAN`. Этот параметр позволяет устаревшим клиентским приложениям работать с версией 5.0 без перекомпиляции для обработки новых типов данных.

6. *Параметры репликации*

   Следующие параметры репликации в РЕД Базе Данных 5 были переименованы. Для обеспечения совместимости можно использовать названия из версии 3. В РЕД Базе Данных 6 нельзя будет использовать старые названия.

Таблица 2.2 — Совместимость параметров репликации

Название опции в 5.0	**Название опции в 3.0**
sync_replica	replica_database
journal_segment_size	log_segment_size
journal_segment_count	log_segment_count
journal_directory	log_directory
journal_file_prefix	log_file_prefix
journal_group_flush_delay	log_group_flush_delay
journal_archive_directory	log_archive_directory
journal_archive_command	log_archive_command
journal_archive_timeout	log_archive_timeout
journal_source_directory	log_directory
verbose_logging	verbose

7. *Изменилось имя пользователя, от которого запускается сервер*

   Сервер РЕД Базы Данных 5 работает от пользователя `reddatabase`.

2.3 Переход с РЕД Базы Данных 5 на 3

1. Прекратите работы с базой данных и сделайте её копию.
2. Запустите миграцию:

```
rdbsvcmgr service_mgr -user SYSDBA -action_migrate -mig_version rdb3.0 -dbname
<база данных>
```

3. Сделайте резервную копию базы данных на версии 5 утилитой gbak от версии 3. Для этого скопируйте файлы РЕД Базы Данных 3 в каталог `/opt/rdb_3.0`. Затем выполните:

```
export FIREBIRD=/opt/rdb_3.0
export LD_LIBRARY_PATH=/opt/rdb_3.0/lib
gbak -user sysdba -password masterkey -b -v -g 127.0.0.1:<база данных> <файл
бэкапа> -y <путь к логу бэкапа> -ignore
```

4. Сохраните копии конфигурационных файлов:

```
cp /opt/RedDatabase/*.conf /home/firebird
```

5. Удалите РЕД Базу Данных 5.
6. Установите РЕД Базу Данных 3.
7. Восстановите базу данных из резервной копии:

```
gbak -user sysdba -password masterkey -c -v <файл бэкапа> <база данных 3.0> -y
<путь к логу рестора>
```

   При этом могут возникнуть сообщения `-bad debug info` и информация об ошибках `blr`. Эти предупреждения нужно игнорировать.
8. Выгрузите скрипты с несовместимыми объектами (`/home/firebird/prefix_permanent.sql`) и объектами, требующими перекомпиляции (`/home/firebird/prefix_restore.sql`):

```
isql -user SYSDBA -password masterkey -ch <кодировка> <база данных 3.0> -extract -
incompatible /home/firebird/prefix -MIGVER 30 -nodbtrig
```

   Каталог /home/firebird должен существовать. Если его нет, то создайте его предварительно.
9. Необходимо исправить `SQL`, содержащийся в `prefix_permanent.sql`, чтобы он был совместим с версией 3.0. После исправления выполните скрипт:

```
isql -user SYSDBA -password masterkey -ch <кодировка> -i /home/firebird/prefix_
permantnt.sql <база данных 3.0> -nodbtrig
```

10. Перекомпилируйте объекты:

```
isql -user SYSDBA -password masterkey -ch <кодировка> -i /home/firebird/prefix_
restore.sql <база данных 3.0> -nodbtrig
```

    При этом могут возникнуть сообщения `bad debug info`. Это предупреждения и они не влияют на результат.
11. Восстановите параметры конфигурации РЕД Базы Данных 5. Необходимо для каждого файла конфигурации, сохранённого на шаге 4, перенести активные опции в соответствующий файл конфигурации версии 3. Если в РЕД Базе Данных 3 нет параметра, который присутствовал в версии

5, необходимо либо найти подходящий по смыслу (т.е. случай, когда изменилось имя параметра), либо не переносить его (такого параметра в версии 3 просто нет).

12. Перезапустите сервер.

Глава 3

Редакции СУБД РЕД База Данных 5.0

СУБД РЕД База Данных выпускается стандартной и промышленной. От выбора редакции зависит содержимое дистрибутива.

Стандартная редакция предназначена для использования в бизнес-приложениях на предприятиях среднего и малого бизнеса. Она входит в реестр российского ПО и имеет сертификат ФСТЭК. В стандартную редакцию включены:

- Поддержка стандарта SQL-2016
- Возможность хранения базы данных в единственном файле
- Инкрементальный бэкап
- Параллельное резервное копирование и восстановление
- Многопоточное создание индексов
- Масштабируемая SMP архитектура SuperServer
- 64-битный счетчик транзакций
- Онлайн валидация БД
- Оконные функции
- Триггеры для DDL операций
- Максимальная длина SQL-запроса 10Мб
- Пул подготовленных запросов
- Синхронная и асинхронная репликация
- Профайлер SQL и PSQL
- Поддержка пакетных операций для вставки данных
- Параллельный сбор статистики
- Аудит изменения прав доступа
- Аутентификация по ГОСТ алгоритмам
- Поддержка отказоустойчивого кластера
- Аутентификация через LDAP/AD/GSS API
- Возможность работы с JSON
- Хранение временных блобов вне основной базы данных

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

- Поддержка адаптивной репликации
- Полнотекстовый поиск
- Коннектор CDC
- Табличные пространства
- Сравнение мастера и реплики онлайн

Глава 4

Установка, обновление и запуск сервера СУБД РЕД База Данных

4.1 Поддерживаемые ОС

СУБД РЕД База Данных может функционировать на следующих ОС:

- Microsoft Windows (x64);
- Linux x86_64 дистрибутивы, использующие:
  - glibc 2.17 и старше;
  - libstdc++ от gcc 10.2 и старше;
- Linux дистрибутивы, поддерживающие Linux Standard Base ISO/IEC 23360, начиная с версии 3.0

Рекомендуемые ОС семейства Linux:

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

4.2 Приемка поставленного сервера СУБД РЕД База Данных

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

4.3 Установка в ОС Windows

Скачать дистрибутив РЕД Базы Данных можно с официального сайта СУБД - RedDatabase. Загрузка доступна только авторизованному пользователю.

Запустите установку СУБД РЕД База Данных с помощью файла `RedDatabase-OE-5.0.X.X-windows-X.exe`, определив разрядность используемой операционной системы.

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

> Для установки РЕД База Данных 5.0 необходимы права администратора.

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

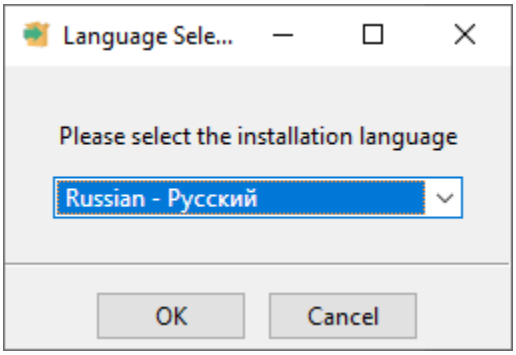
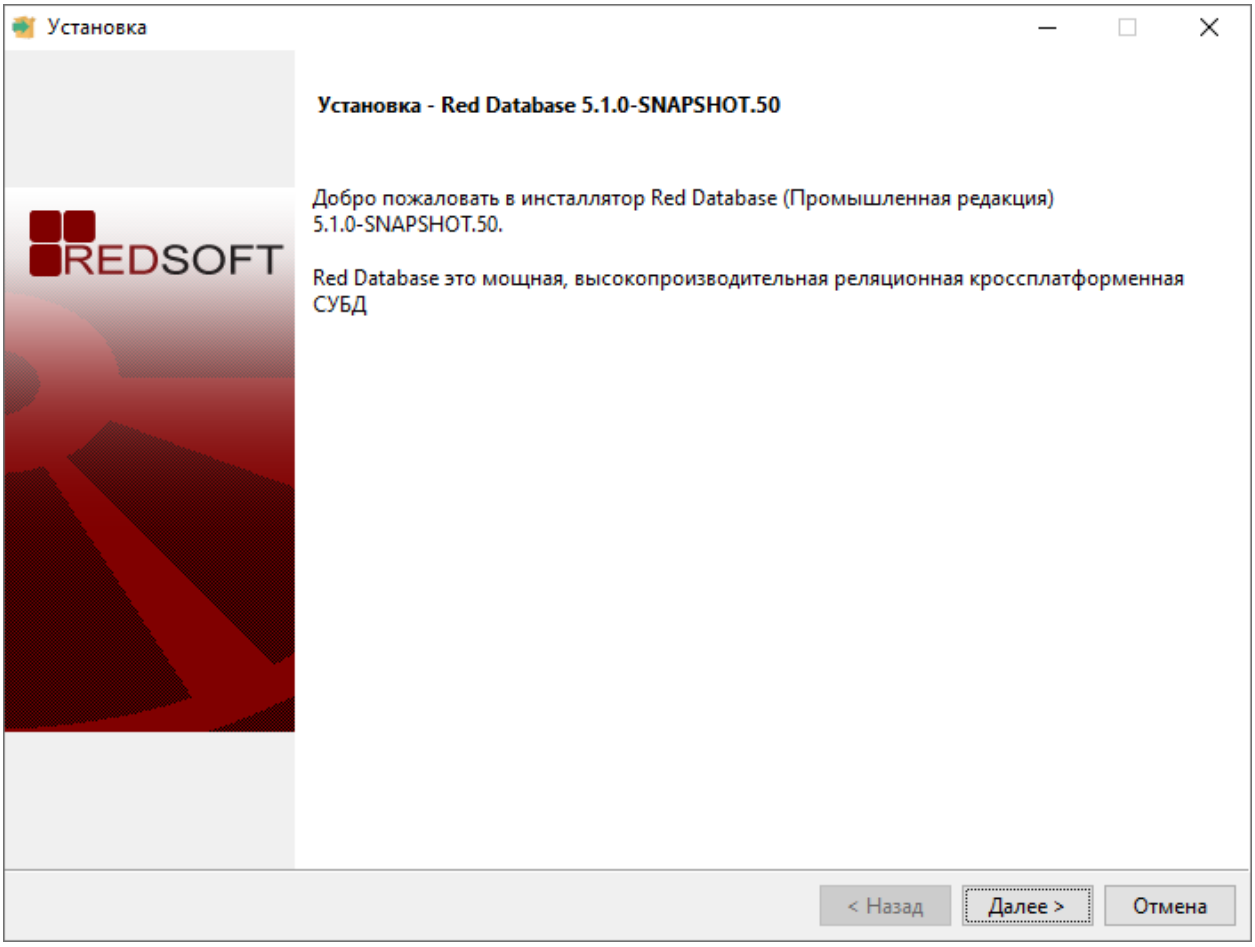
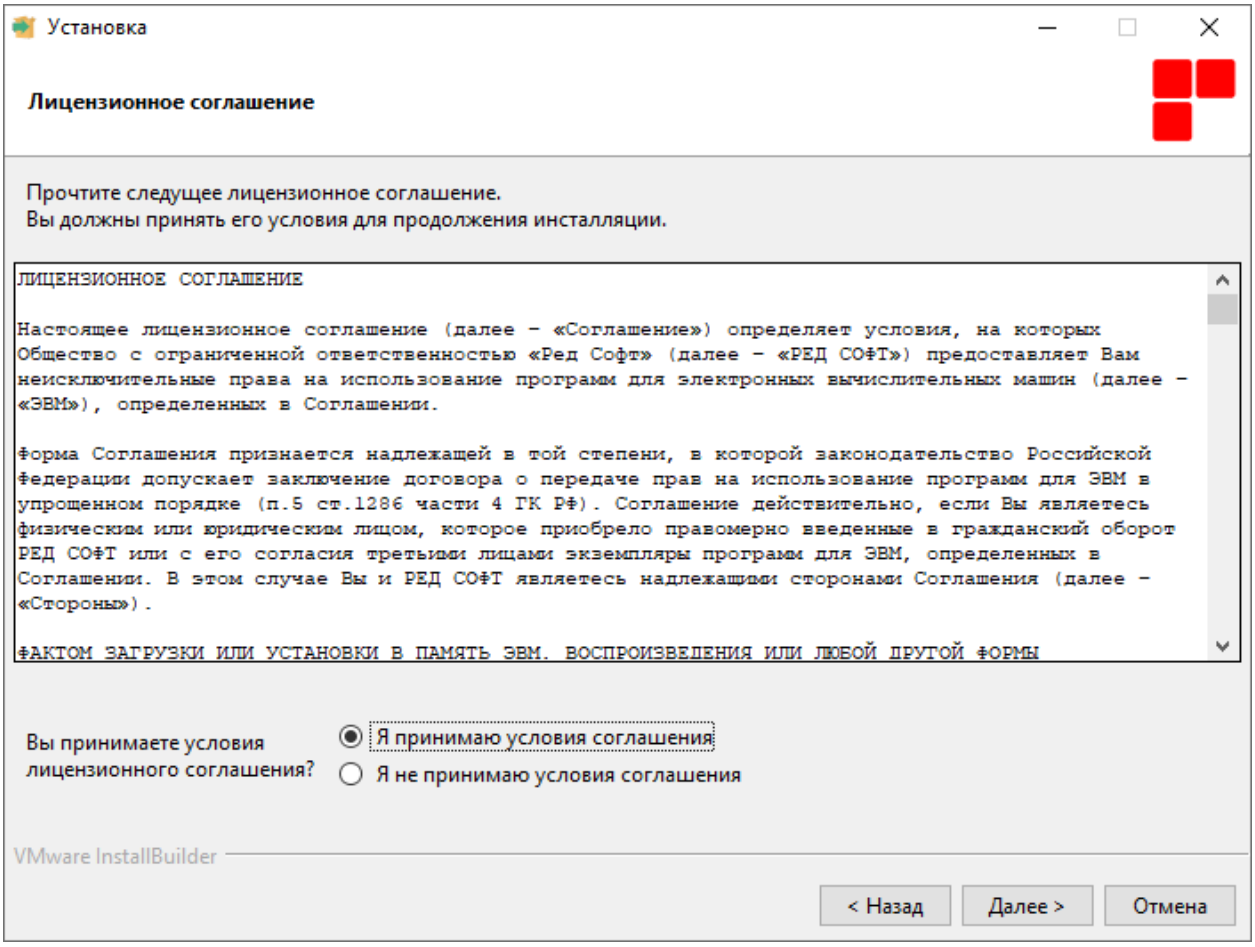


Рисунок 4.1 — Выбор языка





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

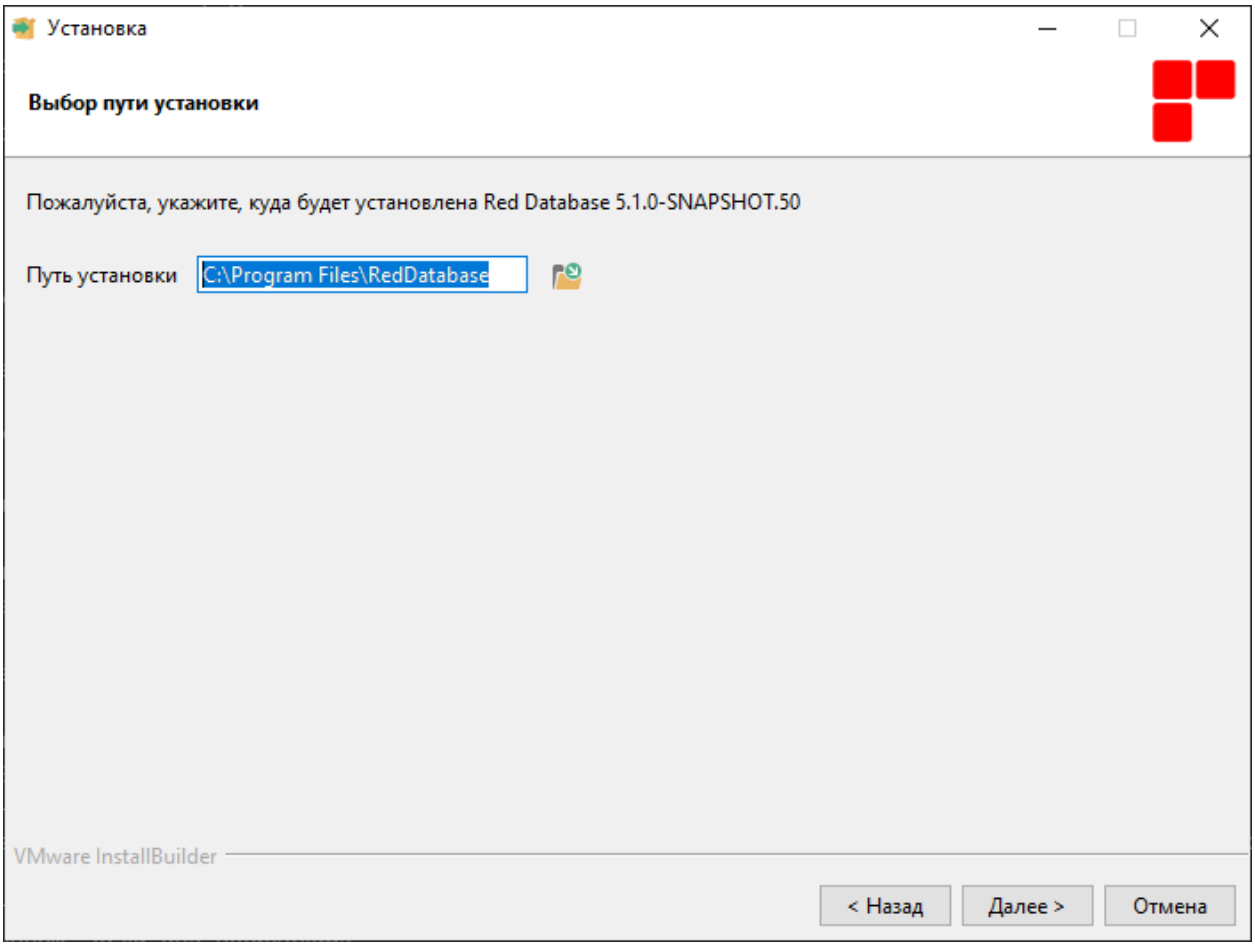
- **Classic**
  - использует отдельный процесс на каждое пользовательское соединение;
  - каждый процесс содержит в себе все что нужно для работы с базой данных: область памяти для метаданных, кэш данных для минимизации повторных чтений из файла БД; память для сортировок;
  - если происходит сбой, другие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

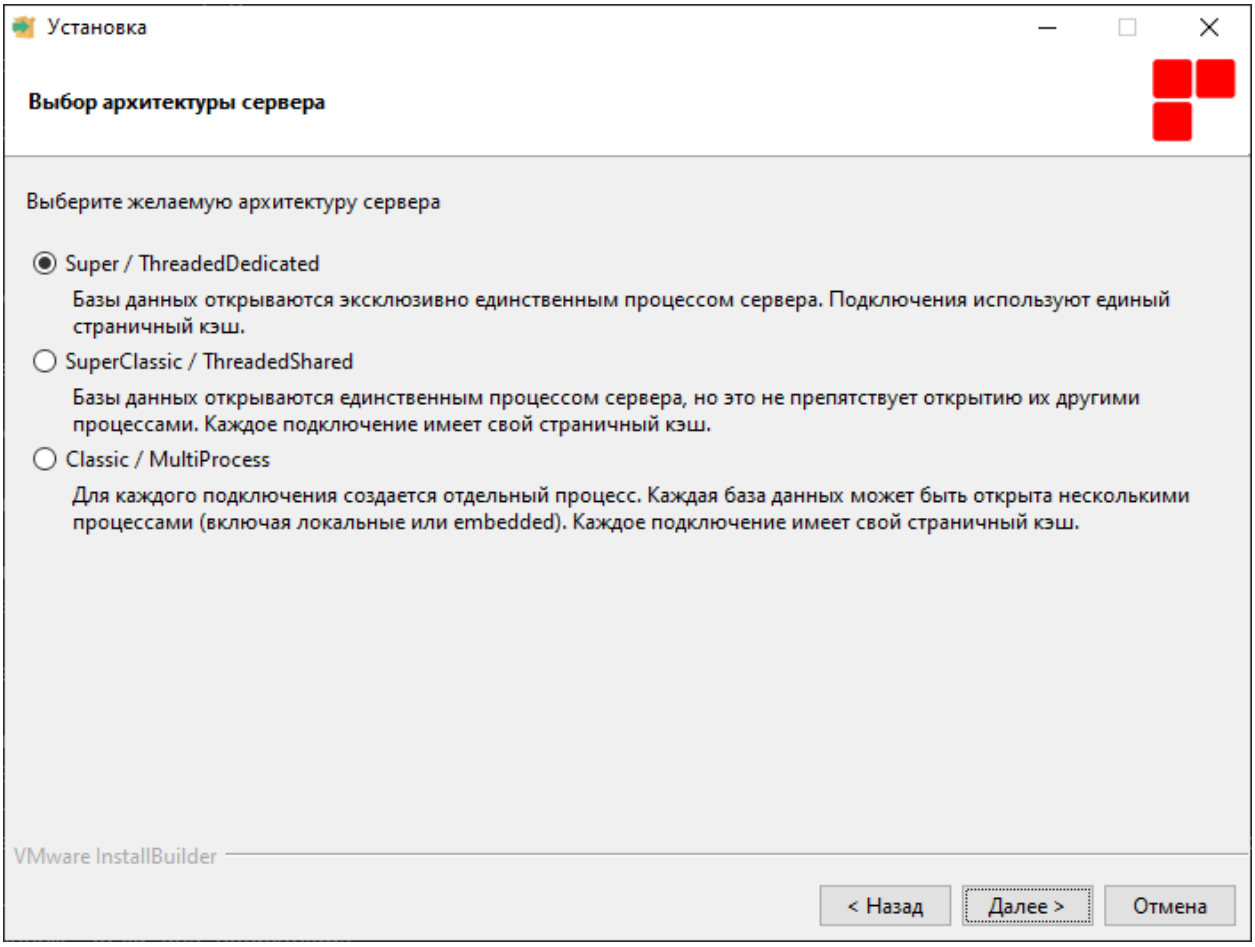
соединения не отъедают ресурсы потоков;

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

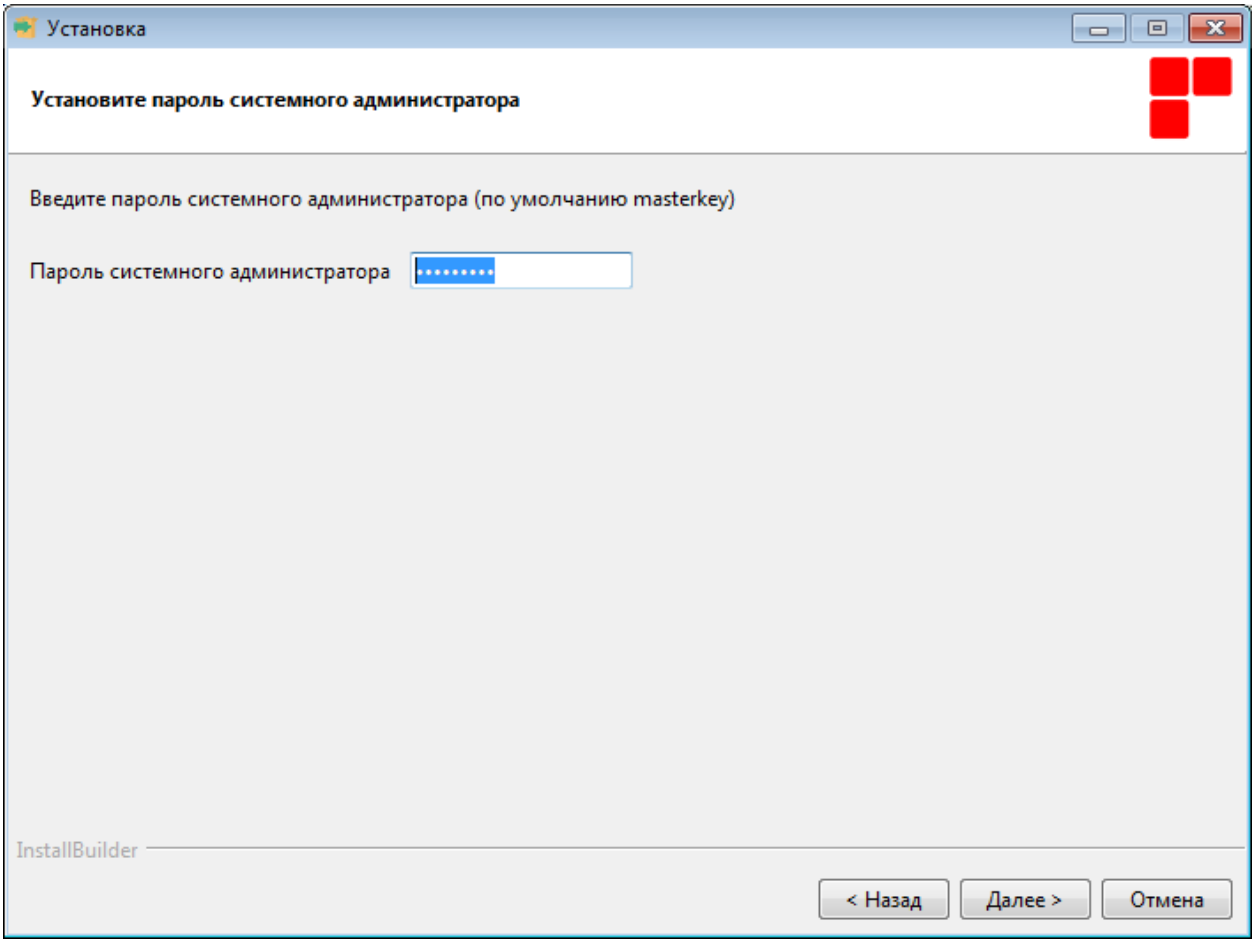
Каждый из режимов стабилен, и нет причин полностью отдавать предпочтение какому то одному. Конечно, у вас могут быть свои собственные конкретные соображения. Если Вы сомневаетесь, просто следуйте за установкой по умолчанию. Позже вы можете изменить архитектуру через файл конфигурации `firebird.conf`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

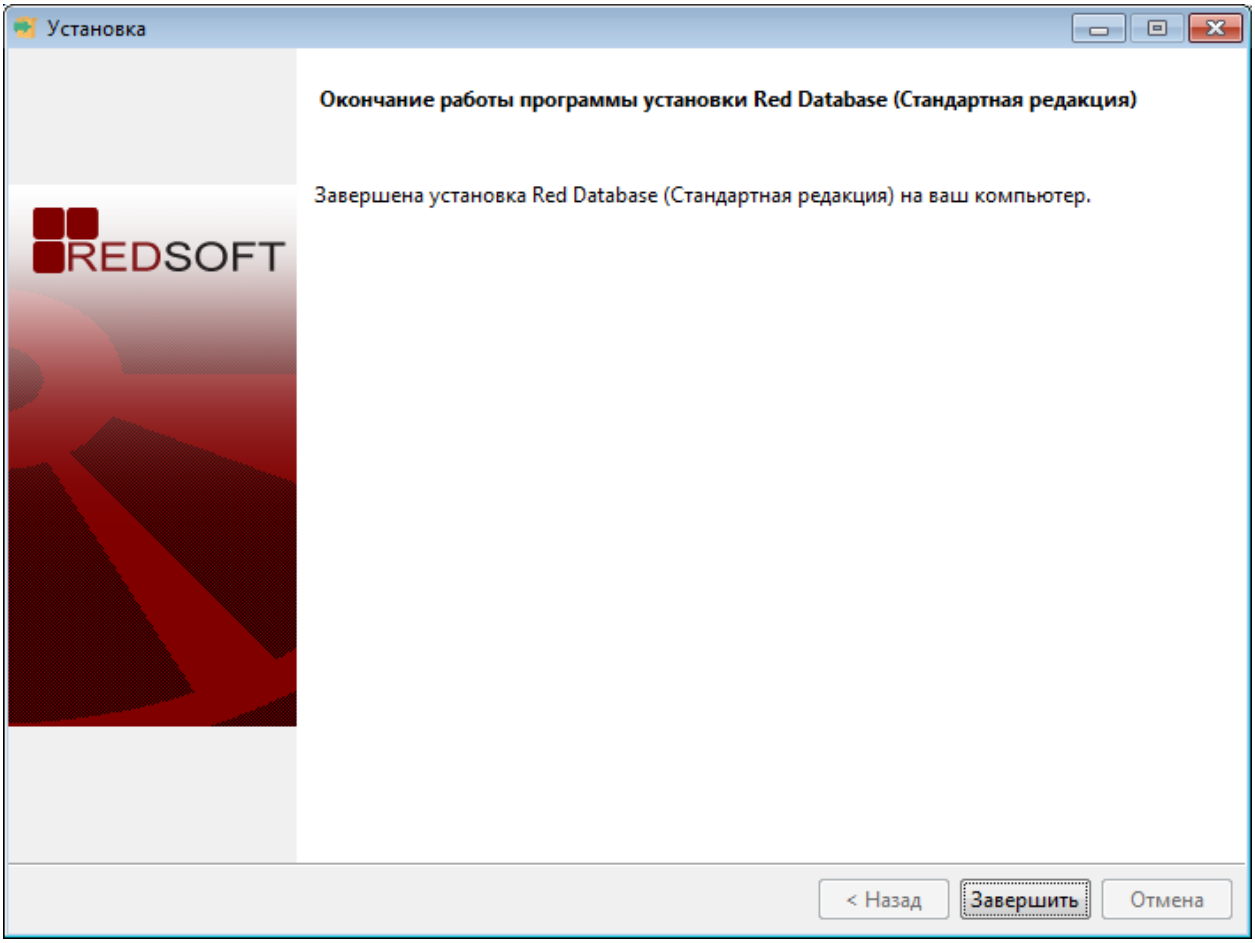
Режим сервера может быть настроен для каждой базы данных отдельно.





Изначально в системе существует только один пользователь – администратор сервера `SYSDBA`. Этот пользователь обладает полными правами на выполнение всех функций по управлению работой сервера и работе с базами данных. В процессе инсталляции необходимо указать пароль для данного пользователя.


Установка
Установите пароль системного администратора
Введите пароль системного администратора (по умолчанию masterkey)
Пароль системного администратора
InstallBuilder
< Назад
Далее >
Отмена



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

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

```
rdbserver -a
```

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

Остановка выполняется через иконку в системном трее (`Shutdown`).

На официальном сайте СУБД РЕД Базы Данных - RedDatabase - можно загрузить исходные коды сервера. Эти исходные коды можно впоследствии использовать для ручной сборки и компиляции бинарных файлов сервера с помощью команды:

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

Для этого необходимо иметь компилятор `C++ Visual Studio` не ниже 2017-й версии.

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

4.4 Установка в Unix-системах

4.4.1 Установка в графическом режиме

Файлы РЕД Базы Данных 5.0 поставляются в виде бинарного пакета. При запуске его из любой графической системы (например, KDE) будет вызван мастер установки, который произведет сбор всей необходимой информации и установит СУБД РЕД База Данных 5.0 на Ваш компьютер.

Для установки СУБД РЕД База Данных необходимо скопировать дистрибутивный файл `rdb50-linux-X-<редакция>-5.X.X.bin` на жесткий диск, а в операционной системе назначить в правах этого файла разрешение на исполнение:

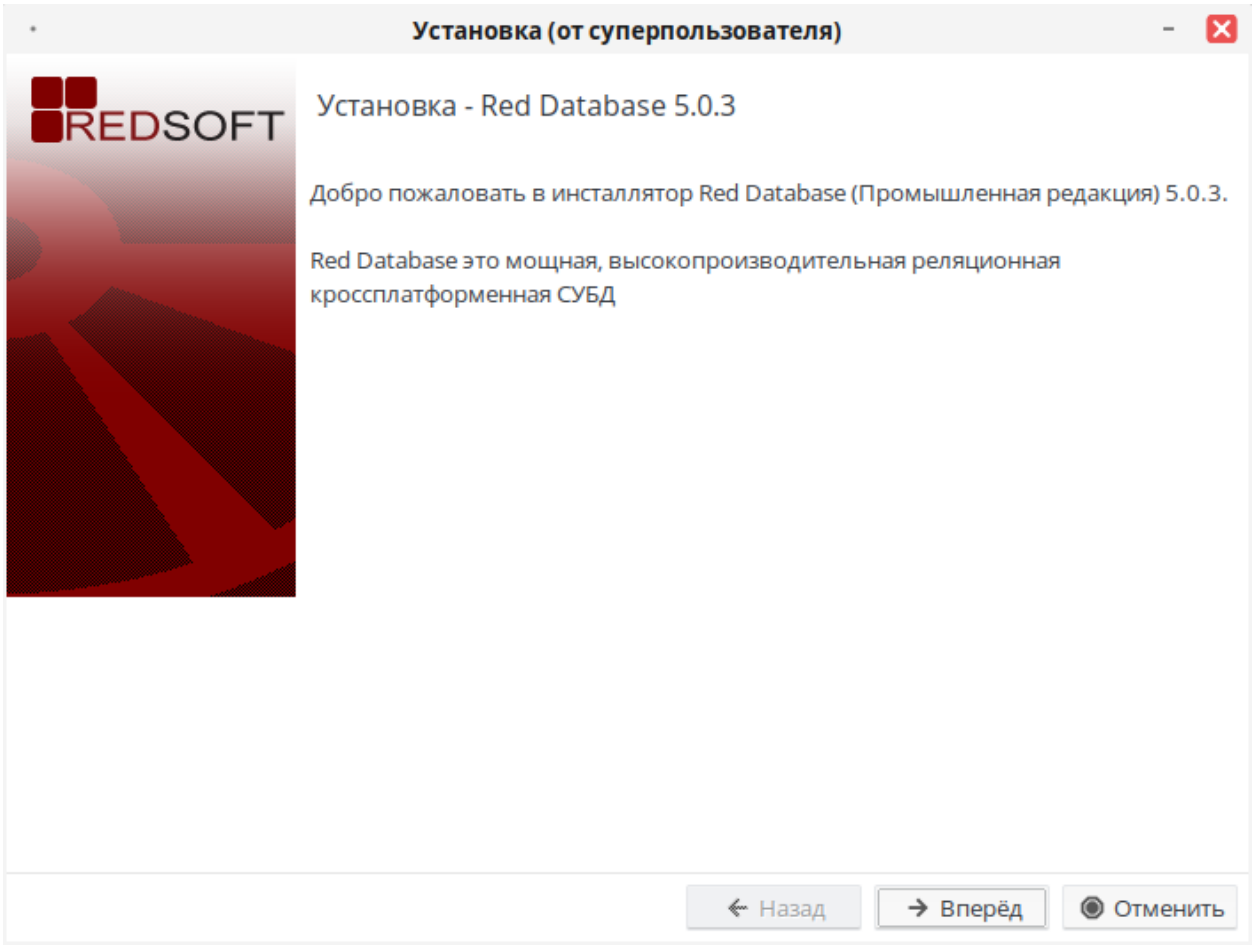
```
# chmod +x RedDatabase-5.0.X.X-X-linux-X.bin
```

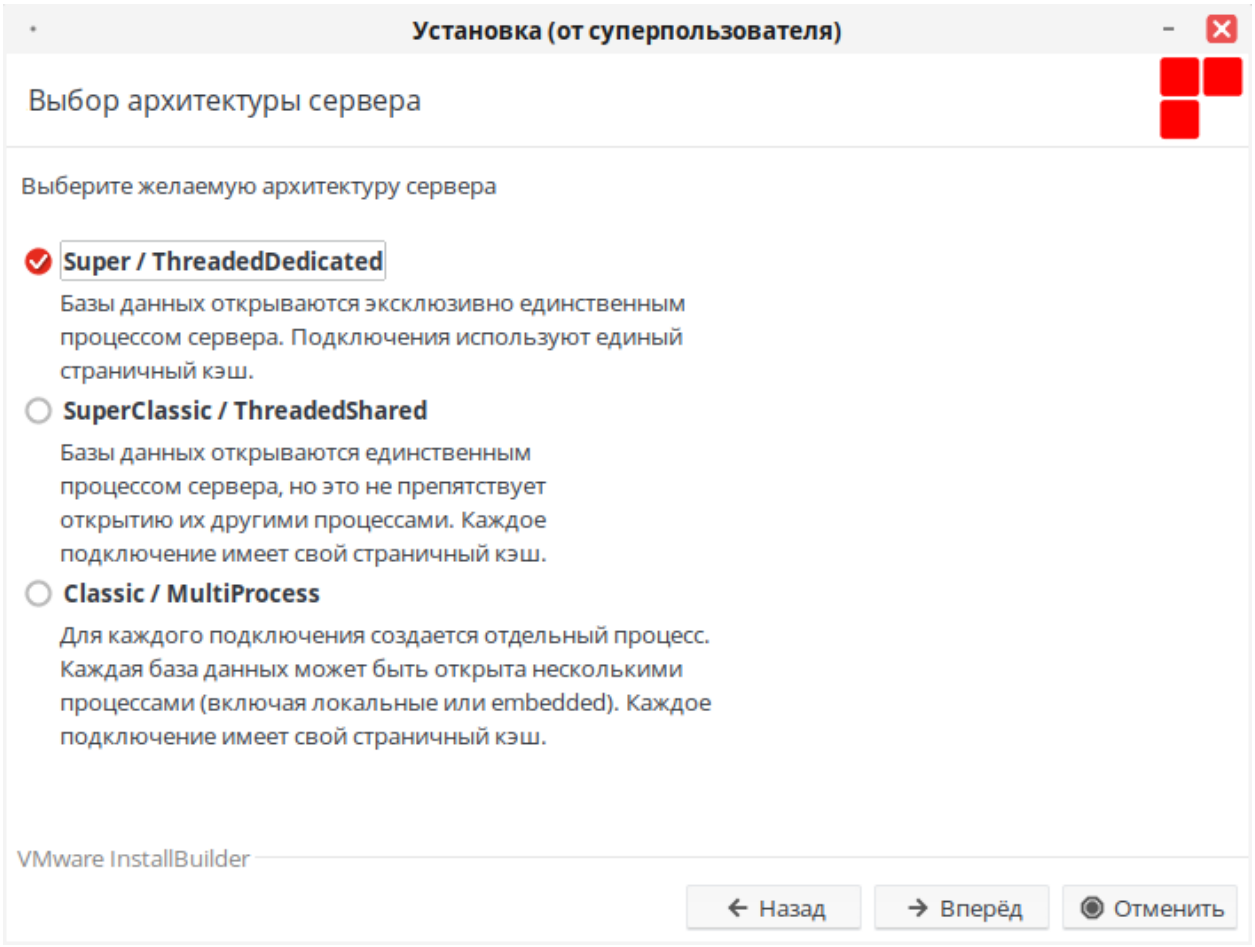
После этого запустить установку СУБД РЕД База Данных:

```
# ./RedDatabase-5.0.X.X-X-linux-X.bin
```

Для установки сервера РЕД База Данных 5.0 необходимы права суперпользователя (root).

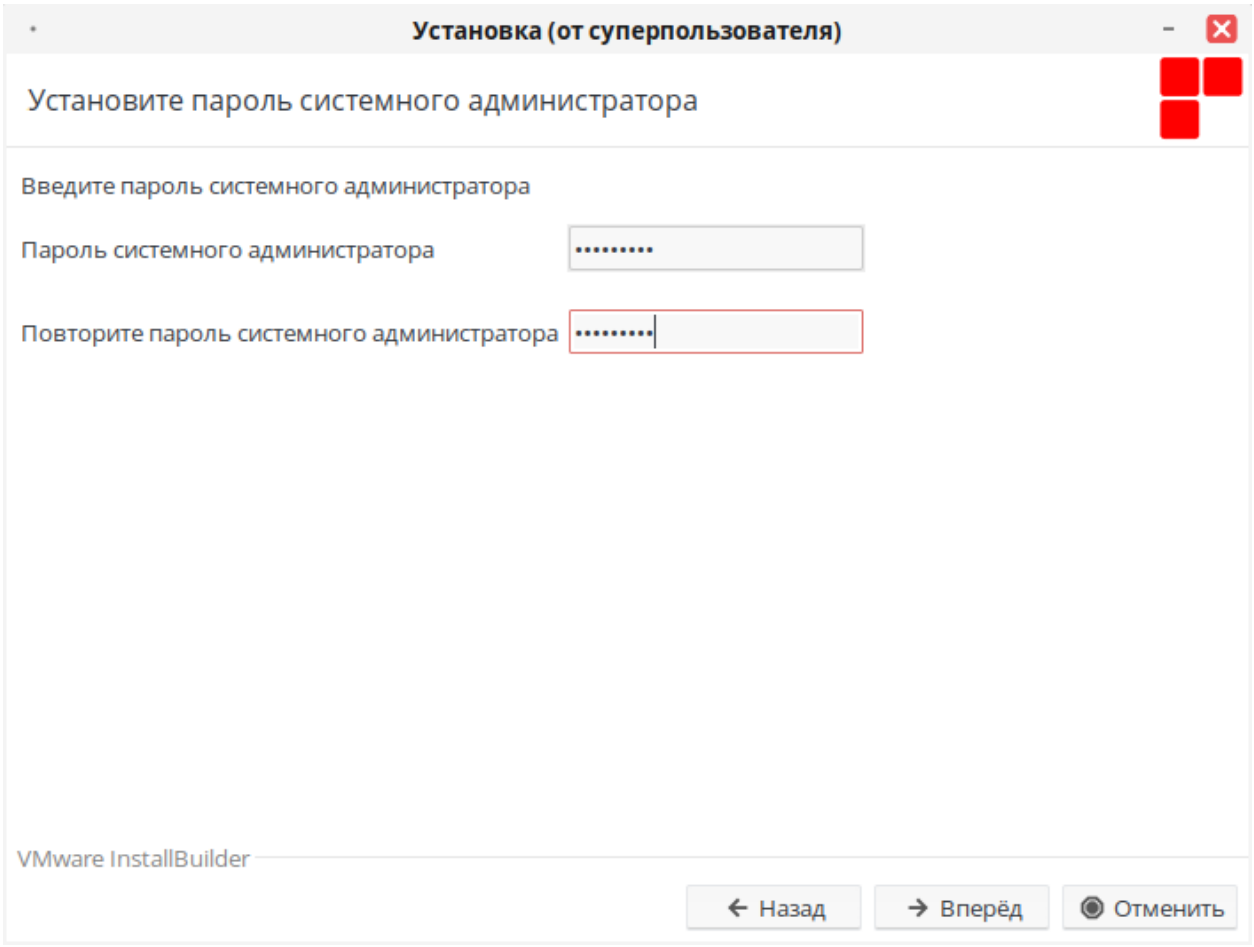
Инсталляция СУБД РЕД База Данных осуществляется с помощью стандартного мастера установки программ. Прежде всего предлагается выбрать язык установки. Предусмотрена инсталляция на русском и английском языках.

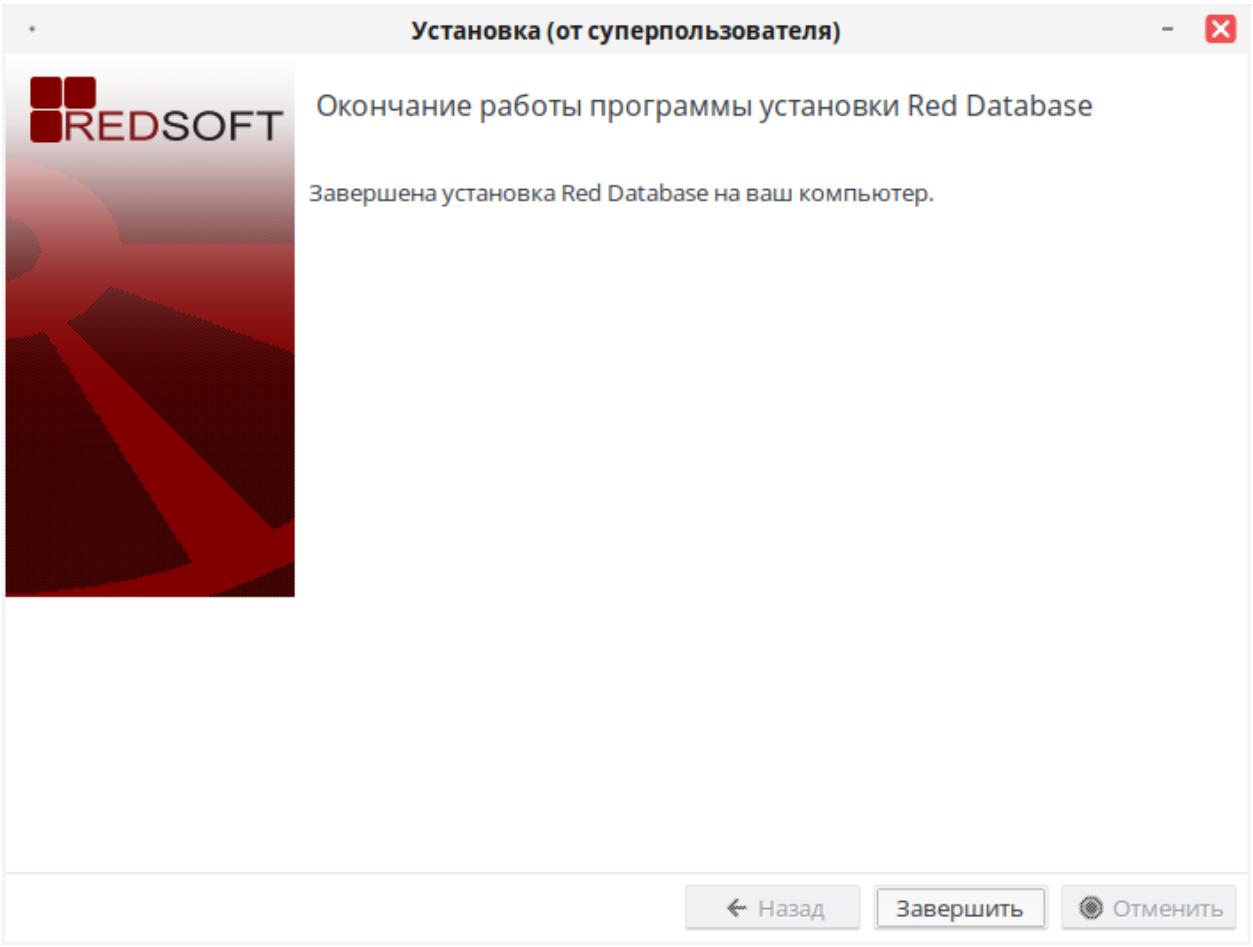

Установка (от суперпользователя)
REDSOFT
Установка - Red Database 5.0.3
Добро пожаловать в инсталлятор Red Database (Промышленная редакция) 5.0.3.
Red Database это мощная, высокопроизводительная реляционная кроссплатформенная СУБД
Назад
Вперёд
Отменить



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

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


Установка (от суперпользователя)
Установите пароль системного администратора
Введите пароль системного администратора
Пароль системного администратора
Повторите пароль системного администратора
VMware InstallBuilder
Назад
Вперёд
Отменить



После установки сервер автоматически не запускается. Вам понадобится сделать это вручную в зависимости от типа системы Linux и вашего установочного пакета. Управление запуском и остановкой сервера осуществляет демон инициализации `systemd`

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

или `init` скрипт:

```
/etc/init.d/firebird start
/etc/init.d/firebird stop
```

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

4.4.2 Установка в текстовом режиме

Если программа инсталляции запущена в текстовом режиме (с ключом `--mode text`):

```
# ./RedDatabase-5.0.X.X-X-linux-X.bin --mode text
```

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

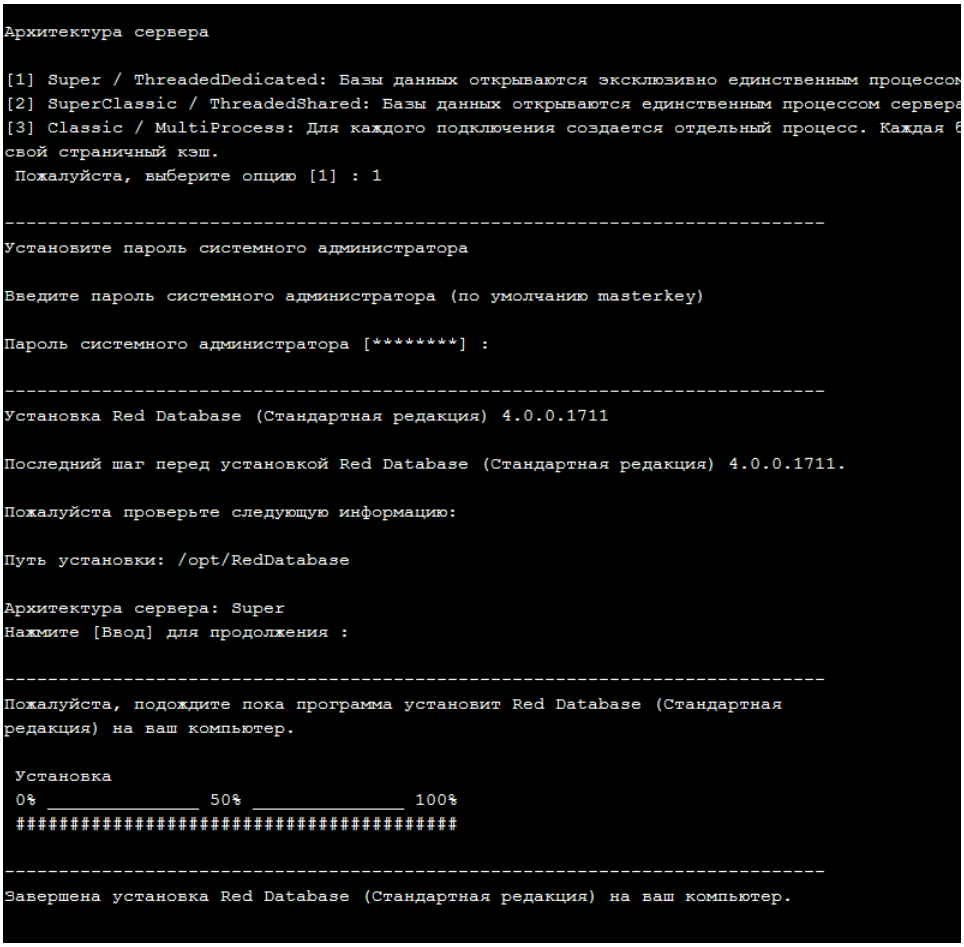


Рисунок 4.2 — Пример установки РЕД База Данных 5.0 в текстовом режиме

Удаление сервера осуществляется, по аналогии с графическим режимом, запуском программы `uninstall` из каталога установки РЕД База Данных.

4.4.3 Установка отладочных символов

1. Скачайте архив с отладочными символами с официального сайта СУБД — RedDatabase . Версия отладочных символов должна совпадать с версией установленной РЕД Базы Данных. Загрузка доступна только авторизованному пользователю.

2. Распакуйте содержимое архива в папку `/opt/RedDatabase`.

```
tar -xvf RedDatabase-5.0.X.X-X-dbg-linux-X.tar.gz -strip-components 1
```

4.5 Процедуры после установки

После установки необходимо настроить политику безопасности по умолчанию, выполнив следующую команду:

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

Политика безопасности по умолчанию:

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

Для включения политик в конфигурационном файле `firebird.conf` укажите плагин `Policy` в параметре `PolicyPlugin`:

```
PolicyPlugin = Policy
```

Для обеспечения контроля целостности выполните следующее:

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

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

   Загрузите изменения службы:

```
systemctl daemon-reload
```

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

```
crontab -e

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

Для настройки регистрации событий безопасности в `firebird.conf` необходимо указать параметр `AuditTraceConfigFiles = fbtrace_sec.conf`:

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

4.6 Обновление сервера

Обновление сервера между минорными- и патч-версиями РЕД Базы Данных происходит с помощью стандартного мастера установки программ. Перед установкой новой версии не требуется вручную удалять старую.

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

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

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

можно закрыть все подключения и запретить последующие.

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

- `databases.conf`
- `directories.conf`
- `fbtrace.conf`
- `firebird.conf`
- `plugins.conf`
- `java-security.fdb`
- `security5.fdb`
- `fbtrace_dba.conf`
- `fbtrace_sec.conf`
- `jvm.args`
- `firebird.log` (только на Linux)

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

4.7 Запуск и остановка сервера

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

4.7.1 На Linux

Используйте команду `top` командной строки для проверки запущенных процессов в интерактивном режиме. Если сервер РЕД Базы Данных 5.0 запущен, вы должны увидеть процесс с именем `rdbserver` и, возможно, также `rdbguard` (процесс Guardian).

На картинке показан вывод `top`, ограниченный `grep`, чтобы отображать только строки, содержащие строку `rdb`:

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

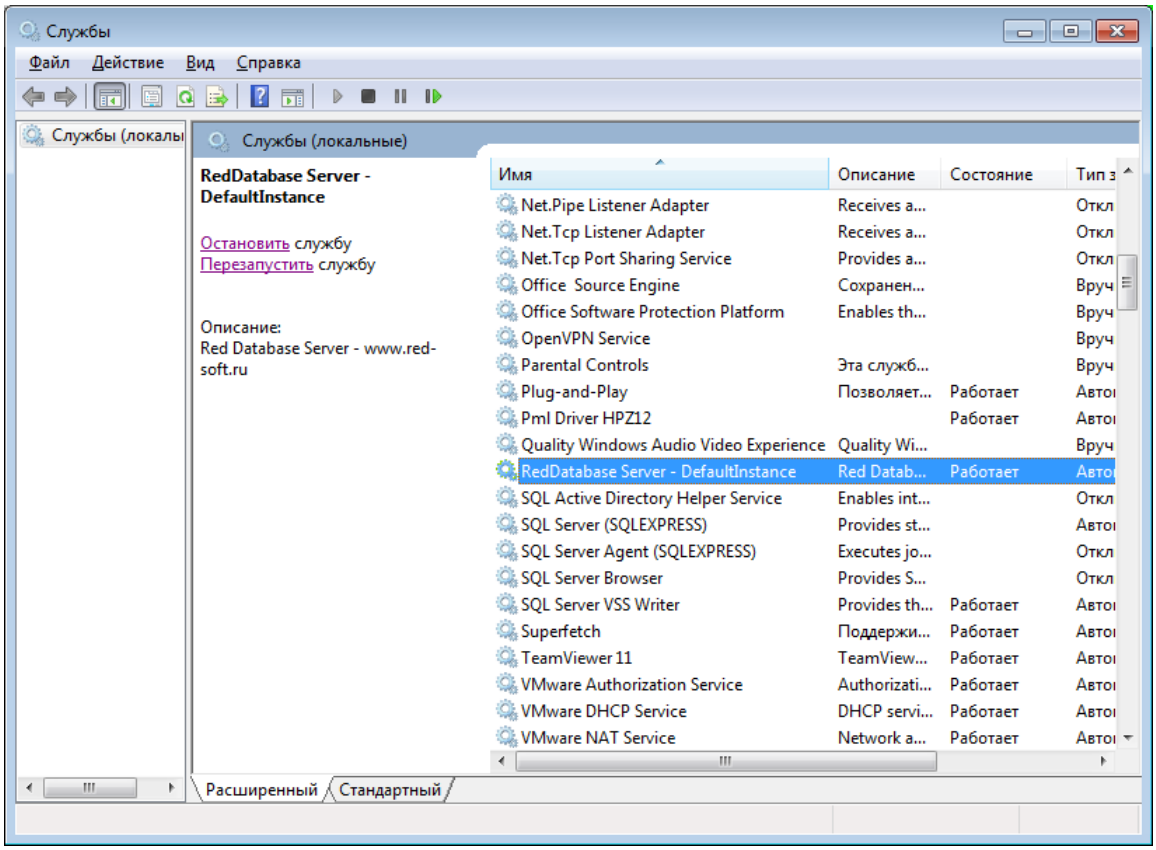
Как альтернатива, вместо команды `top`, Вы можете использовать `ps -ax` или `ps -aux`, при необходимости перенаправив вывод `grep`.

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

Если окажется, что среди запущенных процессов не окажется сервиса `rdbserver`, запустите сервер вручную (см. *выше*).

4.7.2 На Windows

Откройте *Панель управления* → *Администрирование* → *Службы*. На картинке представлен вид апплета Services (Службы) на Windows 7. Внешний вид может изменяться в зависимости от версии Windows.



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

4.8 Сборка сервера РЕД База Данных 5.0 из исходных файлов

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

В комплект поставки СУБД РЕД База Данных входят исходные файлы сервера. При сборке сервера вручную можно изменить каталог установки сервера (по умолчанию `/opt/RedDatabase`), 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. При сборке сервера вручную можно изменить каталог установки сервера (по умолчанию `/opt/RedDatabase`), 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.

Для сборки сервера необходимо выполнить следующие шаги:

1. Установить библиотеки `libtommath` и `ICU`;
2. Установить JRE 11;
3. Установить `g++, autools, make`;
4. Установить переменную среды `JAVA_HOME` (прописать путь до файлов JDK);
5. Выполнить:

```
./autogen.sh -enable-java -enable-binreloc
```

6. Выполнить `make`.

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

Для работы архитектуры Classic необходимо, чтобы сервис `xinetd` был установлен и запущен.

Глава 5

Состав файлов сервера

5.1 Файлы конфигурации, лог-файлы и базы данных

databases.conf

В этом текстовом файле можно сопоставить конкретный путь к БД и псевдоним, чтобы затем в прикладных кодах использовать более короткий и удобный псевдоним для обращения к нужной базе данных. Также здесь указываются индивидуальные настройки для каждой конкретной базы данных. Настройки считываются из файла при каждом соединении с базой данных.

directories.conf

В данном файле задаются алиасы каталогов, в которых хранятся:

- файлы, заполненные BLOB-данными, к которым обращаются функции CREATE_FILE, READ_FILE и DELETE_FILE;
- файлы табличных пространств.

Структура файла:

```
database [= </путь/к/бд> | <псевдоним базы данных>]
{
  blobs
  {
    <алиас_каталога_для_BLOB> = <существующий/путь/к/каталогу>
    ...
  }

  tablespaces
  {
    <алиас_каталога_для_tablespace> = <существующий/путь/к/каталогу>
    ...
  }
}
```

Секции database состоят из двух подсекций: blobs и tablespaces. В секции blobs указываются псевдонимы директорий для хранения BLOB. В секции tablespaces указываются псевдонимы директорий для хранения табличных пространств.

Алиасы каталогов можно настроить для конкретной базы данных отдельно. Для этого нужно указать путь к нужной базе данных. При этом в качестве имени базы данных можно указать регулярное выражение на основе SIMILAR TO. Синтаксис SIMILAR TO см. в Руководстве по SQL.

В старых версиях РЕД Базы Данных файл directories.conf имел другую структуру:

```
database
{
   <алиас_каталога_для_BLOB> = <существующий/путь/к/каталогу>
   ...
}
```

где в секций database перечислялись алиасы и пути к каталогам для хранения файлов с BLOB (но не файлов табличных пространств). Записи такого типа поддерживаются и сейчас. Таким

образом, пути к каталогам с `BLOB` файлами можно задавать как в подсекции `blobs`, так и вне её.

Настройки перечитывается для каждого подключения к базе данных.

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

Настройки перечитываются при старте сессии аудита. Сессии системного аудита запускаются сервером в соответствии с файлами конфигураций, перечисленными в параметре `AuditTraceConfigFile` в `firebird.conf`. Пользовательскую сессию аудита необходимо запускать явно. Перед записью каждого события файл `fbtrace.conf` проверяется на наличие изменений. Если они есть, то настройки будут перечитаны.

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

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

firebird.conf
Файл содержит параметры настройки сервера. Параметры, распространяющиеся на соединение (`per-connection`) перечитываются при каждом соединении. Все остальные параметры считываются при запуске сервера.

plugins.conf
Файл используется для настройки различных плагинов. Если в файле не указана конфигурация для плагина, то для него будут действовать настройки по умолчанию. Настройки считаются один раз при старте сервера.

replication.conf
Используется для настройки системы репликации. На мастере настройки перечитываются при каждом подключении к мастер-базе. На реплике настройки считываются один раз при старте сервера.

scheduler.conf
Файл с настройками планировщика заданий.

scheduler.fdb
База данных планировщика заданий. В ней хранятся сами задания и регистрируемые события, связанные с ними. Настройки считаются один раз при старте сервера.

java-security.fdb
База данных безопасности для пользователей Java. Здесь хранятся права доступа пользователей, использующих код Java.

Механизм `java-security` устарел и будет удален в РЕД Базе Данных 6.

security5.fdb
База данных безопасности. В этой базе хранятся параметры пользователей системы и политики доступа.

jvm.args
Список аргументов для JVM.

firebird.msg
Файл с сообщениями сервера (в основном об ошибках).

firebird.log
Лог-файл сервера.

5.2 Инструменты администрирования и сервисы РЕД Базы Данных

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

rdbrepldiff [.exe]
Утилита сравнения мастер-базы и реплики.

rdbreplmgr [.exe]
Утилита для применения журналов к реплике в ручном режиме. Также выводит информацию о состоянии асинхронной репликации, создает копию мастер-базы, если определен параметр конфигурации `db_copy_command`.

gbak [.exe]
Утилита предназначена для создания резервных копий баз данных и восстановления из резервных копий. Позволяет создать переносимую резервную копию.

gfix [.exe]
Утилита администрирования базы данных. Утилита выполняет проверку базы данных и исправляет некоторые ошибки, позволяет запустить принудительную сборку мусора, управлять зависшими (`limbo`) транзакциями, создавать и удалять теневые копии, переводить базу данных в режим `shutdown` и т.д.

gpre [.exe]
Это препроцессор, который конвертирует исходный код, написанный на различных языках, в корректный отформатированный вызов функций `Firebird API`.

gsec [.exe]
Этот инструмент поддержки списка пользователей и их паролей является интерфейсом командной строки для базы данных `security5.fdb`; он управляет записями пользователей на сервере. Утилита `GSEC` устарела. Вместо неё лучше использовать `SQl-команды` для управления пользователями или `Services API`.

gstat [.exe]
Этот инструмент получения статистики собирает и отображает статистические сведения по индексам и данным базы данных.

hashgen [.exe]
Используется для проверки целостности компонентов СУБД РЕД База Данных на внешних накопителях и в оперативной памяти во время загрузки и динамически.

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

- позволяет установить клиентскую часть РЕД Базы Данных одной командой;
- позволяет установить клиентскую часть как `fdbclient.dll`, либо как `gds32.dll`;
- позволяет проверить наличие установленной библиотеки `fdbclient` или `gds32`;

- позволяет удалить уже установленный в системе `fdbclient` или `gds32`.

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

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

isql [.exe]
Интерактивный инструмент, который позволяет выполнять запросы к базе данных.

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

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

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

rdbserver [.exe]
Исполняемый файл в случае архитектуры Classic, SuperClassic или SuperServer.

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

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

rdbtracemgr [.exe]
Утилита для работы в интерактивном режиме с трассировкой.

5.3 Плагины РЕД Базы Данных

aggtrace.dll
Плагин агрегатного аудита.

certificate.dll
Плагин аутентификации по сертификату.

chacha.dll
Плагин для шифрования данных, передаваемых по сети, алгоритмом `ChaCha`.

crypto_api.dll
Плагин криптографического менеджера.

default_profiler.dll
Плагин PSQL-профайлера.

engine13.dll
Ядро СУБД

extauth.dll
Плагин доверенной аутентификации `ExtAuth`.

fbtrace.dll
Плагин трейса.

gostpassword.dll
Плагин аутентификации с шифрованием по ГОСТ.

gostpassword_manager.dll
Плагин управления пользователями с паролем с шифрованием по ГОСТ.

gss.dll
Плагин аутентификации по протоколу `GSSAPI`.

ldap.dll
Плагин для взаимодействия с сервером `LDAP`.

legacy_auth.dll
Плагин аутентификации по традиционному протоколу (`Legacy`).

legacy_usermanager.dll
Плагин управления пользователями с традиционным паролем.

license.dll
Плагин лицензии для промышленной редакции.

license_open.dll
Плагин лицензии для открытой редакции.

license_std.dll
Плагин лицензии для стандартной редакции.

policy.dll
Плагин, реализующий политики доступа пользователей.

rdbcrypt.dll
Плагин шифрования по ГОСТ-алгоритму.

srp.dll
Плагин аутентификации по протоколу `SRP`.

udr_engine.dll
Плагин вызова пользовательских процедур (`UDR`).

verifyserver.dll
Плагин проверки подлинности сервера со стороны клиента при многофакторной аутентификации.

wire_wincrypt.dll
Плагин шифрования трафика по ГОСТ-алгоритму.

zlibcompressor.dll
Плагин сжатия по алгоритму `deflate`.

zstd.dll
Плагин сжатия по алгоритму `Zstandard`.

Глава 6

Настройка сервера РЕД Базы Данных

Для настройки сервера РЕД База Данных используется файл `firebird.conf`. Параметры, распространяющиеся на соединение (`per-connection`) перечитываются при каждом соединении. Все остальные параметры считываются при запуске сервера.

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

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

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

В файле конфигурации присутствуют параметры трех типов:

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

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

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

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

- `$(root)` – корневой каталог
- `$(install)` – директория, куда установлена СУБД. Изначально `$(root)` и `$(install)` одинаковые. `$(root)` может быть переопределена установкой или изменением переменной окружения `FIREBIRD`, в таком случае эта переменная отлична от `$(install)`.
- `$(this)` – каталог, в котором находится текущий файл конфигурации.
- `$(dir_conf)` – директория, где располагается `firebird.conf` и `databases.conf`.
- `$(dir_secdb)` – директория, где располагается база данных безопасности по-умолчанию.
- `$(dir_plugins)` – каталог расположения плагинов (`plugins`).
- `$(dir_udf)` – директория, где располагаются функции UDF по-умолчанию (`udf`).
- `$(dir_sample)` – каталог с примерами (`examples`).
- `$(dir_sampledb)` – директория, где лежит пример базы данных (`examples/empbuild`).
- `$(dir_intl)` – директория, в которой расположены международные модули (`intl`).
- `$(dir_msg)` – каталог, где находится файл с сообщениями сервера `firebird.msg`. Обычно он совпадает с `$(root)`, но может быть переопределен переменной окружения `FIREBIRD_MSG`.
- `$(dir_schedulerDb)` – директория, где находится база данных планировщика заданий

scheduler.fdb.

Один конфигурационный файл может включать другой с помощью директивы include:

```
include some_file.conf
```

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

Директива include поддерживает групповые символы * и ?.

6.1 Общие настройки

DatabaseAccess

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

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

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

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

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

RemoteAccess *(per-database)*

Параметр предоставляет или отменяет удаленный доступ к базам данных.

```
RemoteAccess = true
```

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

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

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

ExternalFileAccess *(per-database)*

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

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

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

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

BackupAccess

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

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

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

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

UdfAccess

В РЕД Базе Данных 5 UDF объявлены устаревшими и будут удалены в РЕД Базе Данных 6.

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

Параметр `UdfAccess` имеет строковый тип; значение по умолчанию равно `None` (в предыдущих версиях значение по умолчанию было `Restrict UDF`).

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

```
UdfAccess = Restrict UDF
```

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

TempDirectories

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

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

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

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

```
TempDirectories = c:\temp
TempDirectories = c:\temp;d:\temp
TempDirectories = /home/test
TempDirectories = /home/test;/home/test1
```

TempTableDirectory

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

BlobTempSpace *(per-database)*

Параметр `BlobTempSpace` устанавливает место, в котором будут сохраняться данные `BLOB`. Параметр имеет логический тип. По умолчанию его значение равно `false`. Если значении равно `true`, то `BLOB`, для которых явно не задано хранилище, будут создаваться во временном пространстве (таком же,

как для данных GTT). Если значение false, то BLOB, для которых явно не задано хранилище, будут создаются в файле базы данных. BLOB, созданные пользовательскими приложениями, не затрагиваются, даже если хранилище не указано в BPB.

```
BlobTempSpace = false
```

SeparateTempTableFiles *(per-database)*

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

```
SeparateTempTableFiles = 1
```

AuditTraceConfigFiles

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

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

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

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

MaxUserTraceLogSize

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

```
MaxUserTraceLogSize = 10
```

DefaultDbCachePages *(per-database)*

Параметр DefaultDbCachePages определяет количество страниц одной базы данных, которые могут одновременно находиться в кэше. По умолчанию SuperServer выделяет 32768 страниц для каждой базы данных, а Classic и SuperClassic по умолчанию выделяют 1024 страницы на каждое клиентское соединение для каждой базы данных.

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

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

```
DefaultDbCachePages = 32768
```

DatabaseGrowthIncrement *(per-database)*

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

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

```
DatabaseGrowthIncrement = 134217728
```

Пространство под теневые копии баз данных не резервируется.

FileSystemCacheThreshold *(per-database)*

Параметр `FileSystemCacheThreshold` устанавливает порог использования системного кэша сервером «РЕД База Данных» для архитектуры Суперсервер. Значение параметра `FileSystemCacheThreshold` определяет максимально допустимое количество страниц, которые могут находиться в кэш-памяти одновременно. Системный кэш будет использоваться до тех пор, пока количество закэшированных страниц меньше, чем значение параметра `FileSystemCacheThreshold`.

Параметр `FileSystemCacheThreshold` учитывается, только если не задан параметр `UseFileSystemCache`.

Параметр имеет целочисленный тип. Единица измерения – страница базы данных (определяется при создании БД, может иметь размер от 4 до 32 Кб). По умолчанию параметр имеет значение - 65536 страниц. Максимально допустимое значение параметра – 2147483647. Минимальное значение параметра – 0. Если значение параметра `FileSystemCacheThreshold` равно 0, то сервер не будет использовать системный кэш.

```
FileSystemCacheThreshold = 65536
```

ValidatePageChecksums *(per-database)*

Параметр поддерживается начиная с версии РЕД Базы Данных 5.1.

Параметр `ValidatePageChecksums` определяет проверять ли физическую согласованность базы

данных с помощью вычисления контрольных сумм на уровне страниц. В случае несоответствия контрольных сумм в лог-файл будет записана ошибка, но работа продолжится.

Параметр имеет логический тип. По умолчанию значение `true`, то есть контрольные суммы на уровне страниц вычисляются.

```
ValidatePageChecksums = true
```

UseFileSystemCache *(per-database)*

Параметр `UseFileSystemCache` определяет использовать ли кэш файловой системы сервером «РЕД База Данных». Параметр имеет логический тип. Параметр `FileSystemCacheThreshold` учитывается, только если не задан параметр `UseFileSystemCache`.

```
UseFileSystemCache = true
```

InlineSortThreshold

Параметр `InlineSortThreshold` определяет, как обрабатываются неключевые поля во время сортировки: сохраняются внутри блока сортировки или повторно извлекаются со страниц после сортировки. Параметр имеет целочисленный тип. Значение параметра `InlineSortThreshold` определяет максимальный размер записей (в байтах), который может быть сохранен внутри блока сортировки. Если значение параметра `InlineSortThreshold` равно 0, то неключевые поля будут извлекаются со страниц после сортировки.

```
InlineSortThreshold = 1000
```

FileSystemCacheSize

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

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

```
FileSystemCacheSize = 0
```

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

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

RemoteFileOpenAbility

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

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

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

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

```
RemoteFileOpenAbility = 0
```

TempBlockSize

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

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

```
TempBlockSize = 1048576
```

TempCacheLimit *(per-database)*

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

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

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

```
TempCacheLimit = 67108864
```

MaxIdentifierByteLength *(per-database)*

Максимально допустимая длина имени идентификатора в байтах.

Установка этого значения для всех баз данных (включая базу данных безопасности) может вызвать проблемы.

```
MaxIdentifierByteLength = 252
```

MaxIdentifierCharLength *(per-database)*

Максимально допустимая длина имени идентификатора в символах.

Установка этого значения для всех баз данных (включая базу данных безопасности) может вызвать проблемы.

```
MaxIdentifierCharLength = 63
```

AuthServer/AuthClient *(per-database/per-connection, per-database)*

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

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

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

СУБД РЕД База Данных 5.0 поддерживает следующие методы аутентификации:

- Безопасная парольная аутентификация использующая алгоритм хэширования `SHA` для передачи данных: `Srp, Srp224, Srp256, Srp384, Srp512`. По умолчанию используется `Srp256`;
- Традиционная (`Legacy_Auth`) аутентификация;
- Доверительная (`Win_Sspi`) аутентификация для ОС Windows;
- Метод `GostPassword` обеспечивает аутентификацию с использованием алгоритмов шифрования из криптографического плагина (Crypto_API);
- Плагин `Certificate` позволяет аутентифицировать пользователей по сертификатам X509;
- Плагин `VerifyServer` позволяет клиенту проверить сертификат сервера;
- Доверенная аутентификация через механизм GSSAPI (`Gss`);
- Доверенная аутентификации для выполнения Execute Statement On External без указания логина и пароля (`ExtAuth`).

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

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

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

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

PolicyPlugin *(per-database/per-connection, per-database)*

Параметр `PolicyPlugin` определяет, использовать ли политики безопасности. По умолчанию параметр не указан, то есть политики не используются.

Для включения политик безопасности укажите `PolicyPlugin = Policy`, тогда попытка пройти аутентификацию будет осуществляться для каждого плагина, указанного в `AuthServer`, а решение о предоставлении доступа будет принято плагином `Policy`.

```
PolicyPlugin =
```

UserManager *(per-database)*

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

- Srp;
- Legacy_UserManager;
- GostPassword_Manager;
- Ldap.

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

Для создания многофакторных пользователей и управления ими следует установить значение параметра `GostPassword_Manager`.

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

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

```
UserManager = Srp
```

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

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

DefaultUserManagers *(per-database)*

Позволяет указать набор стандартных плагинов.

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

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

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

При удалении пользователя он также удалится из всех стандартных плагинов.

Пример использования:

```
UserManager = Srp Legacy_UserManager GostPassword_Manager
DefaultUserManagers = Legacy_UserManager GostPassword_Manager
```

Значение по умолчанию:

```
DefaultUserManagers =
```

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

TrustedUser

Логин доверенного пользователя для аутентификации по паролю с именем другого пользователя. При коннекте кроме имени пользователя (`isc_dpb_user_name`) можно указать эффективный логин (`isc_dpb_effective_login`). В `isql` для этого добавлен ключ `-l`. Если задан этот логин, то после успешной аутентификации проверяется задан ли параметр конфигурации

- Если не задан - ошибка аутентификации: попытка подмены логина с отключенной опцией.
- Если параметр задан, но пользователь там указан другой - тоже ошибка: попытка подмены логина не доверенным пользователем.
- Если пользователь в конфигурации совпадает с текущим, то есть подключается доверенный пользователь, то при подключении к БД его имя заменяется на указанный им эффективный логин. Для ядра СУБД это подключение будет выглядеть как обычное, информация о подмене логина до него не доходит.

Работает для всех плагинов аутентификации.

TracePlugin

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

```
TracePlugin = fbtrace
```

WireCryptPlugin *(per-connection)*

Параметр определяет плагин, который будет использоваться для шифрования данных, передаваемых по сети.

```
WireCryptPlugin = ChaCha64, ChaCha, Arc4
```

По умолчанию данные шифруются с помощью `ChaCha` или `Alleged RC4`. Ключ должен быть сгенерирован плагином аутентификации. Для `ChaCha` используется 16 или 32 битный ключ, в зависимости от того, что предоставляет плагин аутентификации.

Указанный плагин может быть переопределен в `API` для конкретного соединения через `DPB` или `SPB`.

KeyHolderPlugin

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

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

LdapPlugin

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

LdapLibrary

Общий каталог `LDAP` (только для `Linux`).

```
LdapLibrary = libldap-2.4.so.2
```

AllowEncryptedSecurityDatabase *(per-database)*

Этот параметр позволяет использовать зашифрованную базу данных безопасности.

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

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

```
AllowEncryptedSecurityDatabase = false
```

Providers *(per-database, per-connection)*

Провайдеры - это практически то, что мы подразумеваем под способами, используемыми для соединения клиента с сервером, т.е. через интернет; на том же компьютере через localhost; или через прямое соединение в локальной сети.

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

```
Providers = Remote,Engine13,Loopback
```

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

Провайдеры, реализованные в РЕД Базе Данных 5.0, позволяют поддерживать удаленные соединения, базы данных с разными ODS, а также замыкание (chaining) провайдеров. Замыкание - это термин для ситуации, когда провайдер использует обратный вызов стандартного API при выполнении операции над базой данных.

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

Рассмотрим пример работы `y-valve`, когда он выбирает подходящего провайдера при подключении к базе данных. По-умолчанию имеется три провайдера: `Remote, Engine13, Loopback`.

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

Когда `<имя базы данных>` не содержит сетевого протокола, а только имя базы данных, провайдер `Remote` отклоняет его, а провайдер `Engine13` выходит на первый план и пытается открыть файл с именованной базой данных. Если это проходит успешно, создается подключение к базе данных.

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

- Если файл базы данных, к которому нужно подключиться, не существует, то в этом нет интереса.
- Встроенное соединение может не работать, если пользователь, подключившийся к нему, не имеет достаточных прав для открытия файла базы данных. Это было бы обычной ситуацией, если бы база данных не была создана этим пользователем во встроенном режиме или если ему явно не были предоставлены права ОС на встроенный доступ к базам данных.
- После отказа провайдера `Engine13` в получении доступа к базе данных, пытается подключиться провайдер `Loopback`. Он не очень отличается от `Remote`, за исключением того, что он пытается получить доступ к именованной базе данных `<dbname>` на сервере с сетевым интерфейсом «внутренней петли» (loopback) в сетевом протоколе TCP/IP.

Архитектура провайдеров делает возможным доступ к старым базам данных при переходе на более высокую версию РЕД Базы Данных.

DeadlockTimeout *(per-database)*

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

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

Слишком большое значение параметра может ухудшить производительность системы.

```
DeadlockTimeout = 10
```

StatementTimeout *(per-database)*

Это количество секунд, по истечении которых выполнение оператора будет автоматически прекращено движком. Ноль означает, что время ожидания не установлено. Значение по умолчанию равно 0.

```
StatementTimeout = 0
```

ConnectionIdleTimeout *(per-database)*

Это количество минут, по истечении которых бездействующее соединение будет отключено движком. Ноль означает, что время ожидания не установлено. Значение по умолчанию равно 0.

```
ConnectionIdleTimeout = 0
```

ConnectionSuspendTimeout *(per-database)*

Начиная с версии РЕД Базы Данных 5.0.3 параметр недоступен.

Параметр `ConnectionSuspendTimeout` устанавливает количество секунд, по истечении которых сессия бездействующего пользователя будет заблокирована. Если задан параметр `ConnectionIdleTimeout`, то значение `ConnectionSuspendTimeout` не должно его превышать. По умолчанию период блокировки отключен, то есть `ConnectionSuspendTimeout = 0`.

```
ConnectionSuspendTimeout = 0
```

MaxUnflushedWrites *(per-database)*

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

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

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

```
MaxUnflushedWrites = 100
```

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

MaxUnflushedWriteTime *(per-database)*

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

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

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

```
MaxUnflushedWriteTime = 5
```

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

BugcheckAbort

Опция `BugcheckAbort` определяет, прерывать ли работу сервера при возникновении внутренней ошибки (значение 0) или снимать дамп ядра для последующего анализа (значение 1).

Параметр имеет логический тип. Возможные значения 0 и 1. Значение по умолчанию равно 1.

```
BugcheckAbort = 0
```

ClearGTTAtRetaining *(per-database)*

Операторы `COMMIT RETAINING` и `ROLLBACK RETAINING` сохраняют данные в глобальных временных таблицах объявленных как `ON COMMIT DELETE ROWS`. В версиях 2.x была ошибка: `COMMIT RETAINING` и `ROLLBACK RETAINING` делали записи не видимыми для текущей транзакции. Для возврата поведения 2.x установите параметр `ClearGTTAtRetaining` равным 1.

RelaxedAliasChecking

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

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

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

```
RelaxedAliasChecking = 0
```

ReadConsistency

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

для чтения данных. В этом режиме флаги транзакций `rec_version / no_rec_version` не действуют: любой `READ COMMITTED` транзакции по умолчанию назначаются режимы `read consistency record version`; значение `no_rec_version` молча игнорируется.

```
ReadConsistency = 1
```

DefaultIsolationLevel *(per-database)*

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

```
DefaultIsolationLevel = SNAPSHOT
```

Таблица 6.1 — Совместимость параметров `DefaultIsolationLevel` и `ReadConsistency`

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

ConnectionTimeout *(per-connection)*

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

Параметр `ConnectionTimeout` имеет целочисленный тип и измеряется в секундах. Значение по умолчанию равно 180 секунд. Минимальное значение равно 0. Максимально допустимое значение равно 2147483647.

```
ConnectionTimeout = 180
```

WireCrypt *(per-connection)*

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

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

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

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

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

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

Таблица 6.2 — Совместимость параметров `WireCrypt` на клиенте и на сервере

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

WireCompression *(per-connection)*

Параметр включает или отключает сжатие сетевого трафика.

По-умолчанию параметр отключен.

```
WireCompression = false
```

Параметр настраивается только для клиента - сервер должен следовать настройкам клиента, если он подключен по правильному протоколу ($> = 13$)

DummyPacketInterval *(per-connection)*

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

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

```
DummyPacketInterval = 0
```

В Windows это - единственный способ обнаружить и разъединить неактивных клиентов при использовании `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\_*`.

DefaultTimeZone

Часовой пояс сеанса или клиента.

Если параметр не установлен, то часовой пояс сеанса будет тем же, что используется операционной системой.

При установке значения на сервере он определяет часовой пояс сеанса по умолчанию для подключений.

Когда он установлен на клиенте, он определяет часовой пояс по умолчанию, используемый функциями API на стороне клиента.

RemoteServiceName/RemoteServicePort *(per-connection)*

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

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

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

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

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

RemoteAuxPort *(per-connection)*

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

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

```
RemoteAuxPort = 0
```

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

TcpRemoteBufferSize

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

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

```
TcpRemoteBufferSize = 8192
```

TcpNoNagle *(per-connection)*

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 Nagles Algorithm. Он был разработан, для того, чтобы избежать проблем с маленькими пакетами в медленных сетях.

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

```
TcpNoNagle = 1
```

IPv6V6Only

Позволяет установить параметр сокета `IPV6_V6ONLY`. Включение параметра ограничивает использование сокета исключительно протоколом IPv6; для IPv4 и IPv6 должны использоваться отдельные сокеты. По умолчанию установлено значение `false`.

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

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

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

RemoteBindAddress

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

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

```
RemoteBindAddress =
```

LockMemSize *(per-database)*

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

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

Максимально допустимое значение равно 2147483647.

```
LockMemSize = 1048576
```

Размер таблицы блокировок влияет на:

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

LockAcquireSpins *(per-database)*

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

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

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

```
LockAcquireSpins = 0
```

LockHashSlots *(per-database)*

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

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

```
LockHashSlots = 65521
```

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

Запускаем утилиту

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

В группе заголовка блока (LOCK_HEADER BLOCK), которая описывает основную конфигурацию и состояние таблицы блокировок, смотрим значение элемента Hash lengths (длина цепочки хэширования). Этот элемент сообщает минимальную, среднюю и максимальную длину цепочки слотов. Чем длиннее будут цепочки, тем медленнее будет работать менеджер блокировок. Если среднее значение больше 3 или максимальное больше 10, то это означает, что слотов недостаточно. Поэтому следует увеличить параметр LockHashSlots в 2-3 раза (при этом взять простое число).

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

EventMemSize *(per-database)*

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

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

```
EventMemSize = 65536
```

OptimizationStrategy

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

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

```
OptimizationStrategy = default
```

SnapshotsMemSize *(per-database)*

Кол-во памяти для хранения снапшотов. Оно будет расти автоматически, если вы не используете экзотическую платформу, которая не является ни Windows, ни поддерживает системный вызов mmap. Каждый активный shapshot использует 16 байтов памяти.

```
SnapshotsMemSize = 64K
```

ClientBatchBuffer

Определяет размер буфера, используемого клиентским подключением для пакетной передачи на сервер (при использовании пакетного API).

```
ClientBatchBuffer = 131072
```

DataTypeCompatibility

Задает уровень совместимости, определяющий, какие типы данных доступны клиентскому API. В настоящее время доступны два варианта: 3.0 и 2.5. Режим эмуляции 3.0 скрывает типы данных, появившиеся после версии 3.0, а именно DECIMAL и NUMERIC с точностью 19 и выше, DECFLOAT, TIME WITH TIME ZONE, TIMESTAMP WITH TIME ZONE). Соответствующие значения возвращаются через типы данных, поддерживаемые версией 3.0. Режим эмуляции 2.5 преобразует ещё и тип данных BOOLEAN. Этот параметр позволяет устаревшим клиентским приложениям работать с версией 5.0 без перекомпиляции для обработки новых типов данных.

```
DataTypeCompatibility =
```

OutputRedirectionFile

Позволяет перенаправлять потоки сервера stdout/stderr в пользовательский файл. По умолчанию эти потоки открываются сервером, но выходные данные отбрасываются.

```
OutputRedirectionFile = " "
```

TipCacheBlockSize *(per-database)*

Количество выделяемой памяти для кэша каждого блока TIP. Понизьте это значения, если у вас небольшой TIP и вы хотите сберечь память. Увеличьте это значение, если вам нужен очень большой кэш и ограничения доступа к объектам ядра, выделенным для каждого блока (файлы, мьютексы и т. д.). Каждая кэшированная транзакция использует 8 байтов памяти.

```
TipCacheBlockSize = 4M
```

OnDisconnectTriggerTimeout *(per-database)*

Параметр OnDisconnectTriggerTimeout устанавливает число секунд, по истечении которых выполнение триггера на событие DISCONNECT будет прекращено. Параметр имеет целочисленный тип. Значение по умолчанию равно 180. Значение 0 означает, что время выполнения триггера на событие DISCONNECT не ограничено.

```
OnDisconnectTriggerTimeout = 180
```

SecurityLog

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

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

```
SecurityLog = local
```

MaxStatementCacheSize

Определяет максимальный объем памяти, используемый для кэширования неиспользуемых скомпилированных операторов DSQL. Значение ноль означает, что кэширование не используется. Значение по умолчанию - 2 мегабайта.

```
MaxStatementCacheSize = 2M6
```

DefaultProfilerPlugin

Определяет плагин профайлера по умолчанию, используемый для профилирования с помощью пакета `RDB$PROFILER`.

```
DefaultProfilerPlugin = Default_Profiler
```

OptimizeForFirstRows

Определяет стратегию оптимизации запросов: ускорить получение первых строк или всех.

Значение `false` соответствует стратегии `ALL ROWS`, а `true` соответствует стратегии `FIRST ROWS`.

По умолчанию оптимизатор направлен на получение всех строк.

```
OptimizeForFirstRows = false
```

OuterJoinConversion

Определяет, может ли оптимизатор преобразовывать `OUTER-соединения` в `INNER-соединения` при условии, что такое преобразование возможно с точки зрения результатов запроса.

По умолчанию включено. Может быть отключено для упрощения процесса миграции, если `OUTER-соединения` намеренно используются в SQL-запросах (например, в качестве подсказок оптимизатора), даже если известно, что они семантически эквивалентны `INNER-соединениям`.

```
OuterJoinConversion = true
```

SubQueryConversion

Параметр определяет, следует ли объединять подзапросы `IN/ANY/EXISTS` с внешним запросом с помощью алгоритма `semi-join`, если такое преобразование возможно.

Включив эту опцию можно улучшить производительность за счет того, что подзапрос обрабатывается только один раз и затем кэшируется. Однако в некоторых случаях это может привести к ухудшению производительности, например, когда подзапрос возвращает много строк. По умолчанию включено.

```
SubQueryConversion = true
```

LoginLockoutTime

Параметр `LoginLockoutTime` устанавливает время в минутах, на которое будет заблокирован пользователь, если количество неудачных попыток аутентификации превышает значение `MAX_FAILED_COUNT`, заданное в политике безопасности.

Параметр `LoginLockoutTime` работает, только если настроено использование политик безопасности.

По умолчанию `LoginLockoutTime = 0`, что означает постоянную блокировку. Разблокировать пользователя можно с помощью оператора `RESET USER`. Выполнить этот оператор может только пользователь, обладающий административными привилегиями в БД безопасности.

```
LoginLockoutTime = 0
```

6.2 Настройки ядра

CpuAffinityMask

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

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

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

При наличии в системе производительных (`Performance`) и энергоэффективных (`Efficient`) ядер, значение параметра по умолчанию означает использование сервером только производительных ядер (для `Windows 10` и выше).

```
CpuAffinityMask = 0
```

GCPolicy *(per-database)*

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

- `background` - сборщик мусора работает как фоновый, собирая мусор в отдельном потоке;
- `cooperative` - сборщик мусора работает в оперативном режиме, собирая мусор немедленно при чтении мусорных версий;
- `combined` - сборщик мусора работает в оперативном режиме, но если мусор собрать не удается, то о замусоренных страницах сигнализируется фоновому сборщику мусора.

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

```
GCPolicy = combined
```

SecurityDatabase *(per-database)*

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

По-умолчанию в `firebird.conf`:

```
SecurityDatabase = $(dir_secDb)/security5.fdb
```

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

6.3 Настройки для многопоточной работы

MaxParallelWorkers

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

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

```
MaxParallelWorkers = 8
```

ParallelWorkers

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

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

```
ParallelWorkers = 4
```

6.4 Настройки для Windows-систем

GuardianOption

Параметр определяет должен ли `Guardian` перезапускать сервер после того, как его работа была завершена неправильно.

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

```
GuardianOption = 1
```

ProcessPriorityLevel

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

- 0 – нормальный приоритет;
- положительное значение – повышенный приоритет;

- отрицательное значение – пониженный приоритет.

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

```
ProcessPriorityLevel = 0
```

IpcName

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

```
IpcName = FIREBIRD
```

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

6.5 Настройки для Unix/Linux систем

Redirection

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

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

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

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

```
Redirection = 0
```

6.6 Настройки архитектуры

ServerMode

Определяет архитектуру сервера.

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

```
ServerMode = Super
```

6.7 Настройки пула внешних подключений

ExtConnPoolSize

Устанавливает максимальное количество бездействующих соединений в пуле внешних соединений. Допустимые значения от 0 до 1000. Нулевое значение обозначает что пул выключен.

```
ExtConnPoolSize = 0
```

ExtConnPoolLifeTime

Устанавливает время жизни бездействующих соединений в пуле внешних соединений. Допустимые значения от 1 секунды до 24 часов (86400 секунд).

```
ExtConnPoolLifeTime = 7200
```

MemoryWipePasses

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

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

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

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

```
MemoryWipePasses = 0
```

MaxOpenFileBlobs

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

```
MaxOpenFileBlobs = 0
```

6.8 Настройки LDAP

LDAPServer

Задаёт адрес сервера `LDAP`, используемый для хранения учётной информации пользователей. Можно указать несколько серверов, используя `";"` в качестве разделителя. При неуспешном подключении будет выполнена попытка соединиться со следующим сервером в списке. По умолчанию задано пустое значение, т.е. сервер `LDAP` не используется, и учётная информация ищется только в БД безопасности.

```
LDAPServer = 192.168.1.1
```

LDAPConnectionRetries

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

```
LDAPConnectionRetries = 3
```

LDAPDomain

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

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

```
LDAPDomain = example.com
```

LDAPEncryption

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

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

По умолчанию используется незащищённое подключение.

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

```
LDAPEncryption = None
```

VerifyLdapServer

Включает или отключает проверку сертификата сервера LDAP при включении LDAPEncryption.

```
VerifyLdapServer = 1
```

LDAPUserDN

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

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

LDAPPassword

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

```
LDAPPassword = 123qwe
```

LDAPUserBase

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

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

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

LDAPUserPrefix

Название атрибута, в котором хранится имя пользователя в его DN в LDAP. В основном используется, когда не задан LDAPUserDN. Если параметр не задан, клиентская библиотека fbclient будет шифровать пароль пользователя и для аутентификации будут использоваться пароли в атрибутах: "rdbPassword"; "rdbSrpVerifier"; "rdbSrpSalt"; "rdbSecurePassword"; "rdbPasswordAlgorithm".

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

LDAPUserFilter

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

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

LDAPGroupBase

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

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

LDAPMembershipFilter

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

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

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

LDAPUserCertificate

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

```
LDAPUserCertificate = userCertificate;
```

LDAPPasswordSync

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

- rdbPassword — традиционный пароль пользователя в СУБД «РЕД База Данных»;
- rdbSecurePassword — защищённый пароль пользователя в СУБД «Ред База Данных»;
- userPassword — пароль пользователя, используемый обычно в UNIX-системах;
- sambaLMPassword — пароль пользователя, используемый SAMBA-протоколом;
- rdbSrpVerifier — пароль пользователя для метода Srp;
- sambaNTPassword — пароль пользователя, используемый SAMBA-протоколом.

По умолчанию выполняется синхронизация всех паролей.

```
LDAPPasswordSync = rdbPassword;userPassword
```

LDAPReadOnly

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

```
LDAPReadOnly = 0
```

6.9 Настройки безопасности

LoginFailureDelay

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

```
LoginFailureDelay = 8
```

ServerCertificate

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

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

ServerPrivatePin

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

```
ServerPrivatePin = mypass
```

TrustedCertificate

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

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

CertUsernameDN

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

```
CertUsernameDN = CN
```

CertUsernamePattern

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

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

VerifyCertificateChain

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.

```
VerifyCertificateChain = 1
```

TraceAuthentication

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

```
TraceAuthentication = 0
```

HashesFile

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

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

Алгоритмы хэширования зависят от используемого криптоплагина.

```
HashesFile = hashes
```

IntegrityCheckInterval

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

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

```
IntegrityCheckInterval = 10
```

IntegrityShutdownAttempts

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

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

```
IntegrityShutdownAttempts = 2
```

GssServerKeyfile

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

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

GssServiceName

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

```
GssServiceName = rdb_server
```

GssHostName

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

```
GssHostName =
```

GSSLibrary

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

```
GSSLibrary = libgssapi_krb5.so
```

Classpath

В этом параметре указываются пути к jar-файлам, хранящим пользовательские классы с методами, которые будут использоваться в качестве тела внешних процедур, функций и триггеров или которые реализуют функции полнотекстового поиска.

```
Classpath = ["/path/to/jars", "myjar"]
```

Этот параметр можно настроить отдельно для каждой базы данных. Значения указанные в `Classpath` в `databases.conf` добавляются к значениям указанным в `firebird.conf`. Если список jar для какой-либо базы данных нужно полностью заменить, то нужно использовать опцию `OverrideClasspath` в `databases.conf`.

6.10 Настройка с учетом оборудования и нагрузки на СУБД

На производительность СУБД могут повлиять следующие параметры:

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

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

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

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

`TempCacheLimit = 4194304000`

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

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

`DeadlockTimeout = 100`

Если известно, что архитектура приложения не допускает дедлоков, можно увеличить этот параметр чтобы сервер не сканировал таблицу блокировок каждые 10 секунд при подозрении на потенциальный дедлок. Количество сканирований и найденных дедлоков отображается в заголовке `rdb_lock_print`. Если первый параметр значительно больше нуля, а второй - очень близок к нему, значит характер работы прикладной части предполагает ситуации блокировок, которые дедлоками не являются, но вынуждают СУБД выполнять их поиск. В этом случае можно увеличить значение `DeadlockTimeout` чтобы сервер не выполнял лишнюю работу.

`DefaultDbCachePages = 32768`

Количество страниц, используемых процессом в качестве кэша. `Classic` и `SuperClassic` выделяют 1024 страницы на каждое клиентское соединение для каждой базы данных. Увеличение позволяет до определенной степени сократить обмен данными с диском. Т.к. у каждого процесса классика будет собственный кэш, нужно учитывать общее потребление памяти. Значительное увеличение обычно не даёт эффекта.

По умолчанию `SuperServer` выделяет 32768 страниц для каждой базы данных. Для архитектуры суперсервер значение параметра рекомендуется рассчитать по формуле:

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

где `MemorySize` - общий объём памяти;

`PageSize`

Объём страницы. Для архитектуры классик значение параметра рекомендуется рассчитать по формуле:

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

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

`LockAcquireSpins = 100`

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

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

`WireCrypt = Disabled`

Отключает шифрование сетевого трафика.

`LockMemSize`

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

6.11 Настройка Linux

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

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

```
vm.max_map_count=256000
```

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

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

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

- дописать в `/etc/fstab` строку

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

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

- Настроить `Transparent Huge Pages`:

```
echo madvise >/sys/kernel/mm/transparent_hugepage/enabled
echo madvise >/sys/kernel/mm/transparent_hugepage/defrag
```

создать файл `/etc/rc.d/rc.local`, добавить в него:

```
echo madvise > /sys/kernel/mm/transparent_hugepage/enabled
echo madvise > /sys/kernel/mm/transparent_hugepage/defrag
```

после чего выполнить:

```
chmod u+x /etc/rc.d/rc.local
```

6.12 Настройка работы с Java методами

Механизм `java-security` устарел и будет удален в РЕД Базе Данных 6.

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

Для их использования необходимо установить JRE не ниже 11. А также настроить параметры взаимодействия сервера «РЕД База Данных» с виртуальной машиной Java.

В firebird.conf необходимо указать путь к `JAVA_HOME`:

```
JavaHome = /usr/lib/jvm/java-8-openjdk-amd64
```

В `plugins.conf`, который расположен в корневом каталоге установки сервера, необходимо раскомментировать секции относящиеся к `JavaEngine` и указать путь до каталога с jar-файлами (переменная `JarDirs`):

```
Plugin = JAVA {
   Module = $(dir_plugins)/javaengine
   Config = JAVA_config
}

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

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

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

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

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

6.13 Переменные окружения, служебные и временные файлы

6.13.1 Переменные окружения

Клиентские приложения РЕД Базы Данных могут использовать переменные среды для установки параметров программы. Эти переменные должны быть установлены таким образом, чтобы они были доступны приложению во время его работы.

Следующий список содержит краткое описание этих переменных и их использование:

`ISC_USER`

Имя пользователя базы данных по умолчанию. Если этот параметр установлен, имя пользователя можно не указывать при подключении к базе данных через клиентские приложения и при запуске различных утилит (`gfix`, `gbak`, `gstat` и др.).

Устанавливается в паре с `ISC_PASSWORD` (пользователь с таким именем и паролем должен быть заранее создан в базе данных).

```
set ISC_USER = Hermes
```

`ISC_PASSWORD`

Пароль пользователя базы данных по умолчанию. Если этот параметр установлен, пароль пользователя можно не указывать при подключении к базе данных через клиентские приложения и при запуске различных утилит (`gfix`, `gbak`, `gstat` и др.).

Устанавливается в паре с `ISC_USER` (пользователь с таким именем и паролем должен быть заранее создан в базе данных).

```
set ISC_PASSWORD = Ichneumon
```

FIREBIRD
: Корневой каталог сервера РЕД Базы Данных.

  Если РЕД База Данных устанавливается (или копируется) в каталог, отличный от каталога по умолчанию (в POSIX - `/opt/RedDatabase`), указание переменной среды `FIREBIRD` помогает клиентскому приложению связаться с правильным экземпляром сервера.

```
set FIREBIRD = C:\mydir\RedDatabase
```

```
export FIREBIRD = /mydir/RDB
```

FIREBIRD_LOCK
: Каталог с файлами таблицы блокировок [1]_. По умолчанию все служебные файлы размещаются в одном каталоге. Для Windows это каталог `C:\ProgramData\reddatabase`, а для POSIX – `/tmp/reddatabase`. Доступ к этому каталогу ограничен правами файловой системы.

> Все процессы, которые открывают один и тот же файл базы данных должны работать с одним и тем же `FIREBIRD_LOCK`. Несоблюдение этого правила гарантирует повреждение файла базы. По этой причине лучше никогда не менять местоположение по умолчанию разделяемых служебных файлов. Если вы хотите это сделать – воспользуйтесь символическими ссылками (утилиты `mklink` на Windows или `ln` для POSIX).

```
set FIREBIRD_LOCK = C:\mydir\RedDatabase\lockfiles
```

FIREBIRD_MSG
: Каталог, где находится файл с сообщениями сервера `firebird.msg`. Обычно он находится в корневом каталоге установки сервера (в POSIX - `/opt/RedDatabase`), но может быть переопределен этой переменной окружения.

```
set FIREBIRD_MSG = C:\mydir\RedDatabase\msg
```

FIREBIRD_TMP, TEMP, TMP
: Каталог с временными файлами [2]_. Если не заданы параметры конфигурации `TempDirectories` и `TempTableDirectory`, СУБД РЕД База Данных создаёт временные файлы в каталоге, задаваемом переменными окружения `FIREBIRD_TMP`, `TEMP`, `TMP` (в этом порядке), Если ни одна из этих переменных не установлена, то по умолчанию временные файлы создаются в каталоге `c:\temp` (Windows) или `/tmp` (POSIX). На Windows процессы сервиса РЕД Базы Данных используют, как правило, системный каталог временных файлов (`C:\Windows\TEMP`).

```
set FIREBIRD_TMP = C:\RDBTEMP
```

LC_CTYPE
: Устанавливает системную кодировку (см. UNIX-утилиту `locale`). Она считывается при запуске сервера и утилит и используется для конвертирования системной кодировки в UTF-8 (например различных путей).

```
export LC_CTYPE=ru_RU.UTF-8
```

ISC_MSGS

Задаёт полный путь к файлу `firebird.msg`. Обычно он находится в корневом каталоге установки сервера (в POSIX - `/opt/RedDatabase`).

```
set ISC_MSGS = C:\mydir\RedDatabase\msg\firebird.msg
```

`ICU_TIMEZONE_FILES_DIR`

Задаёт путь к папке с файлами, которые содержат актуальные часовые пояса (`timezone`). По умолчанию это каталог `tzdata` в корневом каталоге установки сервера (в POSIX - `/opt/RedDatabase/tzdata`).

```
export ICU_TIMEZONE_FILES_DIR = /mydir/RDB/timezone
```

`ISC_PATH`

Задаёт путь к каталогу базы данных по умолчанию для автоматического подключения на удаленном сервере.

В ней указывается полный путь к нужному каталогу, включая имя хоста.

Если переменная задана, можно открывать базу данных без указания полного пути.

```
set ISC_PATH = localhost:D:\Databases
```

`VISUAL, EDITOR`

Переменные для работы в isql.

В них указывается команда вызова редактора или полный путь к редактору, который открывается для команды `edit;` в isql.

По умолчанию - `vi` в POSIX и `Notepad` в Windows.

```
export EDITOR = kate
```

`ISQL_HISTSIZE`

Переменная для работы в isql.

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

```
set ISQL_HISTSIZE = 2000
```

`ENV_AUTH_CERT_ALIAS`

Алиас сертификата пользователя для аутентификации на клиенте.

Если алиас сертификата явно не задан при подключении к базе данных через клиентское приложение, то он может быть считан из этой переменной.

Устанавливается в паре с `ENV_AUTH_STORE_PIN` (если `PIN` необходим).

```
set ENV_AUTH_CERT_ALIAS = test,Test Center CRYPTO-PRO,143dd54900020002b231
```

`ENV_AUTH_STORE_PIN`

`PIN` (пароль закрытого ключа), *если он необходим* для получения закрытого ключа сертификата пользователя для аутентификации на клиенте.

Устанавливается в паре с `ENV_AUTH_CERT_ALIAS`.

```
set ENV_AUTH_STORE_PIN = 123456
```

`RDB_OID_TOKEN`
: Токен для аутентификации по протоколу `OpenIDConnect`.

  Если токен не задан при подключении к базе данных через клиентское приложение, то он может быть считан из этой переменной.

```
set RDB_OID_TOKEN = "<токен>"
```

6.13.2 Служебные файлы

Для работы РЕД Базы Данных может требоваться достаточно большое пространство на диске для размещения временных и служебных файлов.

В основном, суммарный размер разделяемых служебных файлов СУБД РЕД База Данных не превышает сотен мегабайт – единиц гигабайт. Достаточно надёжная оценка сверху – от пяти до десяти гигабайт.

Все служебные файлы размещаются в одном каталоге. Для Windows это каталог `C:\ProgramData\reddatabase`, а для POSIX – `/tmp/reddatabase`. Доступ к этому каталогу ограничен правами файловой системы.

Все служебные и временные файлы СУБД РЕД База Данных используют префиксы `rdb_` или `rdbЧИСЛО_`, где `ЧИСЛО` – сокращённый номер версии (`50`).

В категорию служебных входят файлы менеджера блокировок (lock manager), данные таблиц мониторинга, буферы событий и сессий трассировок, также некоторые другие файлы, включая специальные файлы-флаги нулевого размера.

Файлы менеджера блокировок
: Начальный размер файлов менеджера блокировок (префикс `rdb_lock_`) задаётся параметром `LockMemSize`. Архитектуре SuperServer требуется хранить в этом файле меньше данных, чем архитектурам Classic и SuperClassic. Типичный размер локфайлов находится в пределах 20-150 МБ, но при большом количестве подключений, в архитектурах Classic и SuperClassic, размер лок-файла может приближаться к двум гигабайтам.

  Лок-файл создаётся при открытии файла базы и удаляется при его закрытии. По умолчанию файл базы закрывается после отключении последнего клиента, но начиная с РЕД Базы Данных 3 можно установить «время удержания» (linger time) DDL-командой.

Файлы таблиц мониторинга
: Все таблицы мониторинга (виртуальные) заполняются при первом обращении к любой из них и сохранённые данные не меняются до завершения транзакции. Размер «снимков» таблиц мониторинга (префикс `rdb_monitor_`) зависит от приложения, его активности и от количества обращений к таблицам мониторинга из разных подключений и транзакций.

Файлы трассировки
: Файл `rdbЧИСЛО_trace` содержит «ссылку» на активный файл с информацией о сеансах трассировки.

  Файл с префиксом `rdb_trace_` содержит информацию о сеансах трассировки. Таких файлов может быть множество, но «активен» будет только один из них. Файлы с префиксом `rdb_trace.` – буферы сессий трассировки. Их максимальный размер ограничен параметром `MaxUserTraceLogSize`. Если клиент не успевает вычитывать данные, то по достижении предела (10 МБ по умолчанию), трассировка автоматически приостанавливается.

Файлы событий
: Файлы с префиксом `rdb_event_` представляют собой буферы, хранящие недоставленные события, если приложение на них подписано.

Файлы кэша сопоставления имен
: Файл `rdbЧИСЛО_user_mapping` используется для управления кэшем сопоставления имён в процес-

сах СУБД РЕД Базы Данных.

Файлы флаги

В POSIX-системах дополнительно используются специальные файлы-флаги нулевого размера – `fb_rename_guard`, `rdb_init` и `rdb_port_HOMEP`. Эти файлы требуются при запуске процессов СУБД РЕД Базы Данных.

6.13.3 Временные файлы

Местоположение временных объектов (кроме данных временных таблиц и блобов) управляется параметром конфигурации `TempDirectories`, а также параметром `TempTableDirectory` (для данных временных таблиц и блобов).

Если какой-то из этих параметров (или оба) не задан или указанный каталог недоступен, будет использован каталог из переменных окружения `FIREBIRD_TMP`, `TEMP`, `TMP` (в этом порядке).

Если ни одна из этих переменных не установлена, то по умолчанию временные файлы создаются в каталоге `C:\temp` (Windows) или `/tmp` (POSIX). На Windows процесс(ы) сервиса RedDatabase используют, как правило, системный каталог временных файлов (`C:\Windows\TEMP`).

Префиксы временных файлов:

- `rdb_table_` - данные временных таблиц;
- `rdb_blob_` - данные временных блобов;
- `rdb_undo_` - undo-логи;
- `rdb_recbuf_` - кэш курсоров;
- `rdb_merge_` - хэш-соединения;
- `rdb_sort_` - блоки сортировок;
- `rdb_tpc_` - новые режимы изоляции в транзакциях;
- `rdb_snap_` - снимки, согласованные по чтению для разных подключений.

Глава 7

Рекомендации по безопасной настройке СУБД

7.1 Общие рекомендации

- **Проверяйте актуальность версии СУБД**

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

- **Выбирайте правильную архитектуру СУБД**

  Существует четыре различных взаимозаменяемых архитектуры сервера:

  – **ClassicServer** — один процесс на одно соединение; поддержка многопроцессорных машин. Подходит для мощных систем с несколькими ЦПУ и большим количеством ОЗУ. Данная архитектура не вызывает отказ в обслуживании всех клиентов при сбое одного серверного процесса, и как следствие более надежна.

  – **SuperServer** — все соединения используют один процесс, меньшие требования к памяти при большем быстродействии; для многопроцессорных машин (до РЕД Базы Данных 3 для однопроцессорных). Это компактная и высокопроизводительная версия для встраивания в распространяемое ПО.

  – **SuperClassic Server** — один процесс, но свой поток на каждое соединение. Комбинация лучшего от SuperServer и Classic. Идеально подходит для виртуализации.

  – **Embedded** (встраиваемая) версия — весь движок содержится в одной библиотеке с именем клиентской библиотеки сервера, идеально подходит для однопользовательских систем, не требует инсталляции в Windows.

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

- **Защитите каталоги**

  Правильно настройте права доступа к каталогам, где размещается сама СУБД, базы данных, конфигурационные файлы и журналы аудита.

  Любой, кто имеет доступ к файловой системе на уровне чтения может скопировать БД и извлечь все данные из неё. Любой, кто имеет права на запись может уничтожить информацию, подменить ее или сделать БД нечитаемой. Как правило, только процесс сервера должен иметь доступ к файлам и каталогам. Пользователи не должны иметь доступа даже на уровне только для чтения. Они будут выполнять запросы к БД через сервер, а сервер гарантирует, что пользователи получают только разрешенный тип доступа к каким-либо объектам в базе данных.

  Права на каталог с СУБД и конфигурационные файлы настраиваются автоматически инсталлятором. Их владельцем становится `root`. Но файлы `firebird.log, *.fdb, rdb_guard` доступны для записи пользователю `firebird:firebird`.

  Владельцем каталога с базой данных и с журналами аудита должен быть пользователь, от имени которого запускается сервер (в Linux по умолчанию это пользователь `firebird:firebird`, в Windows - `System`). Только владельцу каталога должны быть предоставлены права на запись и чтение.

- **Делайте регулярные копии**

  Резервное копирование необходимо для возможности быстрого и недорогого восстановления информации (пользовательских баз данных, базы данных безопасности СУБД, конфигурационных файлов

настроек СУБД, журналов аудита СУБД) в случае утери рабочей копии информации по какой-либо причине.

Периодичность резервного копирования определяется исходя из важности хранимых данных. Для информации имеющей особенную ценность периодичность должна быть не реже чем 1 раз в сутки.

Утилита `Nbackup`, входящая в комплект поставки СУБД, позволяет создавать резервные копии, восстанавливать из резервных копий и дополнительно позволяет создавать инкрементные копии и восстанавливать из них БД. Инкрементная резервная копия содержит только изменения со времени создания определенной,ранее созданной резервной копии. `Nbackup` позволяет блокировать файл базы данных и может работать с активной базой данных, не мешая подключенным к ней пользователям. Созданная резервная копия базы данных всегда будет отображать состояние базы данных на момент начала создания резервной копии. Если вы используете инкрементное копирование не забывайте периодически делать полные копии БД.

Типичный порядок создания резервной копии: Блокировать базу данных с помощью параметра `-L (Lock)`:

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

Создать резервную копию, сжать файл базы данных, используя любые другие программы.

Простое копирование файла также допустимо.

Разблокировать базу данных с помощью параметра `-UN (UNlock)`:

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

Более подробно об использовании утилит резервного копирования можно прочитать в главе *Утилиты командной строки*.

- **Настройте репликацию**

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

- **Запускайте СУБД от имени несистемного пользователя**

  На Unix-подобных системах, СУБД обычно уже запущена от имени пользователя `firebird` по умолчанию, а не `root`.

  На Windows платформах также необходимо запустить сервис СУБД под заданной учетной записью (например, `firebird`). На практике запуск службы от имени пользователя `localsystem` представляет угрозу безопасности, тем более если ваша система подключена к сети Интернет.

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

  Пользователь `sysdba` - это аккаунт с полными правами доступа для всех ваших баз данных. Его пароль должен быть известен только нескольким доверенным администраторам. Не используйте этого супер-пользователя для регулярной работы с БД. Вместо этого, создайте учетные записи пользователей и наделите их необходимыми правами.

7.2 Рекомендации по настройке СУБД

- **Смените пароль SYSDBA**

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

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

- **Используйте псевдонимы для БД**

  В конфигурационном файле `databases.conf` можно сопоставить реальный путь к БД и специальный псевдоним, чтобы затем использовать более короткий и удобный псевдоним для обращения к нужной базе данных.

  Использование псевдонимов БД позволяет скрывать физическое расположение баз данных от конечных пользователей. Например, добавив строку в `databases.conf`:

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

  Пользователям для подключения к БД будет достаточно знать псевдоним БД - `personal`.

  Псевдонимы также позволят вам без проблем перемещать базы данных, ведь клиенты продолжат использовать свои существующие строки подключения.

  Псевдонимы начинают работать немедленно как только были добавлены и сохранены - нет необходимости в перезагрузке сервера.

- **Измените метод аутентификации**

  СУБД РЕД База Данных поддерживает следующие режимы аутентификации:

  – Безопасная парольная аутентификация использующая алгоритм хэширования SHA для передачи данных: `Srp, Srp224, Srp256, Srp384, Srp512`. По умолчанию используется `Srp256`;
  – Традиционная (`Legacy_Auth`) аутентификация, унаследованная от предыдущих версий;
  – Доверительная (`Win_Sspi`) аутентификация для ОС Windows;
  – Метод `GostPassword` обеспечивает аутентификацию с использованием алгоритмов шифрования из криптографического плагина (Crypto_API);
  – Плагин `Certificate` позволяет аутентифицировать пользователей по сертификатам X509;
  – Плагин `VerifyServer` позволяет клиенту проверить сертификат сервера;
  – Доверенная аутентификация через механизм GSSAPI (`Gss`).

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

- **Измените стандартный порт подключения**

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

  ```
  RemoteServicePort = 49156
  ```

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

- **Защитите сервер от атаки перебором паролей**

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

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

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

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

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

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

- **Контролируйте активность пользователей**

  Для простых учетных записей пользователей вполне достаточно одной открытой сессии. Для учетный записи администратора `sysdba` это значение может быть увеличено до 2.

  Компьютер, оставленный без присмотра пользователем, может быть использован злоумышленником. Задайте время бездействия пользователя, после которого сессия будет блокирована.

  Администраторы могут владеть неактуальной информацией об используемых учетных записях. Полезно блокировать те учетные записи пользователей, которые не используются продолжительное время.

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

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

- **Контролируйте целостность ваших файлов**

  Контроль за файлами сервера означает, что для всех критически важных файлов сервера (бинарные файлы, файлы конфигурации, база данных безопасности `security5.fdb` и т. д.) может быть вычислен и проверен хэш.

  Для генерации хэшей используется утилита `hashgen`, входящая в комплект поставки СУБД. Пример использования утилиты `hashgen`:

```
hashgen generate -S 32801 ../firebird.conf > hash.sign
```

Этой командой сгенерирована и сохранена в файл `hash.sign` контрольная сумма `firebird.conf`.

Для того, чтобы включить контроль целостности файлов сервера, необходимо указать имя файла, содержащего контрольные суммы (хэши) защищаемых файлов сервера в конфигурационном файле `firebird.conf`. Имя этого файла задается параметром `HashesFile`:

```
HashesFile = hash.sign
```

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

```
IntegrityCheckInterval = 10
```

Раз в 10 секунд сервер будет проверять БД `security5.fdb`. Если хэши при проверке не совпадут, то СУБД попытается прекратить работу безопасными средствами. Количество попыток безопасного прекращения работы задается параметром `IntegrityShutdownAttempts`:

```
IntegrityShutdownAttempts = 2
```

- **Обезличивайте память**

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

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

  Для быстрой очистки памяти установите `MemoryWipePasses = 1`.

- **Включите аудит событий**

  Аудит событий реализован на основе плагина `FBTrace`. Средства аудита позволяют серверу отслеживать и записывать в лог-файлы такие события: соединения и отсоединения от БД (создания и удаления БД), операции DML и DDL, выполнение хранимых процедур и т.д. Запись в лог для каждой конкретной БД начинает вестись с момента подключения к ней и до момента отсоединения от нее или ее удаления. Регистрируются события, завершившиеся как удачно, так и неудачно (с ошибкой).

  Сессию системного аудита запускает сам сервер. Это означает, что нет необходимости взаимодействия с пользователем. События, которые будут отслеживаться в этой сессии, задаются в конфигурационном файле и читаются при старте сессии.

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

  По умолчанию аудит выключен. Для включения аудита необходимо изменить в конфигурационном файле `fbtrace.conf` параметр `enabled`. Файл конфигурации перечитывается для каждого подключения.

- **Используйте шифрование**

  В СУБД РЕД База Данных существует возможность зашифровать данные хранимые в базе данных. Не весь файл базы данных шифруется: только страницы данных, индексов и blob.

  Чтобы сделать шифрование базы данных возможным, необходим плагин шифрования базы данных. РЕД База Данных не предоставляет такой плагин; его можно написать самому или получить у сторонних разработчиков. Существует возможность только его подключить и воспользоваться им.

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

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

  Также в СУБД РЕД База Данных есть защита канала передачи данных. Параметр `WireCrypt`

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

- **Изучите документацию**

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

Глава 8

Утилиты командной строки

8.1 Утилита ISQL

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

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

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

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

8.1.1 Запуск ISQL

Запуск утилиты производится следующим образом:

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

<спецификация сервера> ::= host[\port | service]:
                           | \\host[@port | service]\
                           | <protocol>://[ host[:port | service]/]
<protocol> ::= inet \| inet4 \| inet6 \| xnet
```

8.1.2 Соединение с базой данных

После запуска утилиты необходимо либо присоединиться уже к существующей БД, либо создать новую. Для создания базы используется оператор CREATE DATABASE, для соединения с уже существующей базой данных — оператор CONNECT (подробнее см. «Руководство по SQL», глава 6). Присоединиться к уже существующей БД можно непосредственно при запуске утилиты, указав имя базы, и, если необходимо, имя пользователя и пароль[1]. Необязательный переключатель -role задаёт имя роли, права которой будут учитываться при работе с базой данных.

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

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

8.1.3 Символ терминатора

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

```
SET TERM <строка>;
```

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

8.1.4 Транзакции в ISQL

ISQL может использоваться для выполнения операций трех типов:

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

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

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

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

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

8.1.5 Переключатели командной строки

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

Таблица 8.1 — Опции ISQL

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

(разрыв таблицы)

(разрыв таблицы)

Опция	Описание
-e(cho)	Включает или подавляет дублирование команд на указанное устройство вывода (монитор, в файл, и т. д.) (set echo on).
-ex(tract)	Извлечь метаданные.
-f(etch_password)	Извлечь пароль из файла.
-i(nput) <имя файла>	Задать файл с SQL-запросами для выполнения (set input).
-l(ogin)	Эффективный логин доверенного пользователя. Для аутентификации по паролю с именем другого пользователя. См. параметр конфигурации TrustedUser.
-m(erge)	Перенаправление ошибок на поток стандартного вывода.
-m2	Отправлять информацию о статистике и планах в выходной файл, который указан в переключателе -o(utput).
-n(oautocommit)	Отключить автоматическое подтверждение DDL-операций (set autoddl off).
-nod(btriggers)	Не запускать триггеры базы данных.
-now(arnings)	Не показывать предупреждения.
-o(utput) <имя файла>	Задать файл для вывода результата выполнения запросов. Без аргументов перенаправляет вывод на стандартное устройство вывода (монитор)(set output).
-pag(elength) <размер>	Размер страницы.
-p(assword) <пароль>	Пароль пользователя.
-pi(n) <PIN>	Задать PIN (пароль), если он необходим для получения закрытого ключа сертификата пользователя.
-q(uiet)	Не показывать сообщение "Use CONNECT...".
-r(ole) <роль>	Имя роли.
-r2 <роль>	Имя роли с учетом регистра.
-s(qldialect) <диалект>	SQL диалект (set sql dialect).
-t(erminator) <строка>	Команда терминатора (set term).
-ti(meout) <секунды>	Время в секундах, по истечении которого сессия бездействующего пользователя будет заблокирована. Подробнее см. *Блокирование сеанса пользователя* . Начиная с версии РЕД Базы Данных 5.0.3 опция недоступна.
-tr(usted)	Использовать доверительную аутентификацию.
-to(ken) <token>	Токен для аутентификации по протоколу OpenIDConnect.
-u(ser) <пользователь>	Имя пользователя.
-x	Извлечение метаданных.
-xn	Извлечь метаданные, опустив блоки SET TERM ^ ; ... SET TERM ; ^.
-z	Показать версии утилиты и сервера
-v(erify) <алиас сертификата>	Алиас сертификата, которым сервер будет удостоверять свою подлинность клиенту при аутентификации плагином VerifyServer.

8.1.6 Общие команды

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

Таблица 8.2 — Общие команды ISQL

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

(разрыв таблицы)

(разрыв таблицы)

Команда	Описание
RECONNECT [[PASSWORD '<пароль>'] | [CERTIFICATE '<алиас_сертификата>' [PIN <пароль_закрытого_ключа>]]]	Позволяет возобновить заблокированное подключение к базе данных. Авторизационные данные, которые необходимо указать, зависят от способа аутентификации во время первого подключения к базе данных. В случае использования доверенной аутентификации, указывать авторизационные данные не требуется.

8.1.7 Команды SHOW

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

```
SHOW <объект> [<имя объекта>|<шаблон имени объекта>]
```

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

Следующий пример покажет все таблицы, начинающиеся с tab.

```
show tables tab%
```

Таблица 8.3 — Команды SHOW

Команда	Описание
SHOW ALL	Покажет все метаданные.
SHOW CHECKs <имя таблицы>	Отображает имена и тексты всех определенных пользователем ограничений CHECK, заданных для указанной таблицы.
SHOW COMMENTs	Отображает все комментарии, которые были созданы для разных объектов текущей БД.
SHOW {COLLATIONs | COLLATION <имя>} SHOW {COLLATEs | COLLATE <имя>}	Выводит список всех определенных пользователем параметров сортировки текущей БД.
SHOW {DOMAINs|DOMAIN <имя>}	Отображает определение одного указанного домена <имя> или отображает список имен всех доменов, объявленных в базе данных (DOMAINs) .
SHOW {DATABASE|DB}	Отображает информацию о подключенной базе данных (имя файла, размер и количество выделенных страниц, интервал очистки, номера транзакций, статус Forced Writes, ODS, набор символов по умолчанию).
SHOW DEPENdencies <имя объекта> SHOW DEPENdency <имя объекта>	Отображает все зависимости для указанного имени объекта. На выходе - разделенный запятыми список других объектов БД, с которыми указанный объект зависим.

(разрыв таблицы)

(разрыв таблицы)

Команда	Описание
SHOW {EXCEPtions|EXCEPtion <имя>}	Отображает текст одного указанного исключения <имя> или отображает список имен и текстов всех исключений, объявленных в базе данных (EXCEPtions).
SHOW {FILTERs | FILTER <имя>}	Выдает список всех BLOB-фильтров, объявленных оператором declare filter или показывает подробную информацию о конкретном фильтре.
SHOW SYStem {FUNCtions|FUNCtion <имя>}	Отображает объявление указанной UDF функции <имя> или отображает список имен всех UDF функций, объявленных в базе данных (FUNCtions).
SHOW {FUNCtions|FUNCtion <имя>}	Отображает объявление указанной хранимой функции <имя> или отображает список имен всех хранимых функций, объявленных в базе данных (FUNCtions).
SHOW SYStem {GENERATORs|GENERATOR <имя>} SHOW SYStem {SEQuences|SEQuence <имя>}	Эти две команды идентичны. Отображают объявление указанного системного генератора <имя> вместе с его текущим значением или отображает список имен всех системных генераторов, объявленных в базе данных вместе с их текущими значениями (GENERATORs|SEQuences).
SHOW {GENERATORs|GENERATOR <имя>} SHOW {SEQuences|SEQuence <имя>}	Эти две команды идентичны. Отображают объявление указанного генератора <имя> вместе с его текущим значением или отображает список имен всех несистемных генераторов, объявленных в базе данных вместе с их текущими значениями (GENERATORs|SEQuences).
SHOW {GRANTs | GRANT {<имя объекта>|<имя роли>}}	Если команда содержит GRANTs, то она отображает список всех привилегий в текущей БД. Иначе отображает информацию привилегий и ролей по отношению к указанному объекту в подключенной базе данных или отображает членство пользователей в роли.
SHOW {INDexes|INDICES} SHOW {INDICES|INDexes} <имя таблицы> SHOW INDex <имя индекса>	Первая форма отображает информацию обо всех индексах для всех таблиц в подключенной базе данных. Вторая форма отображает информацию об индексах для указанной таблицы <имя таблицы>. Наконец третья форма отображает информацию об указанном индексе.
SHOW {MAPPING | MAPPING <имя>}	Показывает все созданные отображения объектов безопасности в подключенной базе данных или информацию о конкретном отображении.
SHOW {PROCedures | PROCedure <имя>}	Отображает все процедуры в подключенной базе данных с их зависимостями или отображает текст указанной процедуры с объявлениями и типами (входной/выходной) каждого аргумента.
SHOW {PACKAGES | PACKAGE <имя>}	Отображает все пакеты в подключенной базе данных с их зависимостями или отображает содержимое указанного пакета.
SHOW {ROLEs | ROLE <имя>}	Отображает имена ролей SQL в подключенной базе данных или отображает всех пользователь, получивших данную роль.

(разрыв таблицы)

(разрыв таблицы)

Команда	Описание
SHOW SECCLAsses <имя объекта>	Отображает информацию о классах безопасности для данного объекта.
SHOW SQL DIALECT	Отображает диалекты SQL клиента и подключенной базы данных.
SHOW SYStem [<имя таблицы>]	Команда без параметров отображает имена системных таблиц, функций и сортировок для конкретных наборов данных. Для уточнения деталей о конкретной таблице укажите имя таблицы.
SHOW {TABLEs | TABLE <имя>}	Отображает список имен всех таблиц в алфавитном порядке или показывает подробности указанной таблицы.
SHOW {TABLESPACEs | TABLESPACE <имя табличного пространства>}	Отображает список имен всех табличных пространств в алфавитном порядке или показывает информацию об указанном табличном пространстве.
SHOW {TRIGgers | TRIGger <имя>}	Отображает список имен всех таблиц вместе с именами их триггеров в алфавитном порядке или для заданного триггера указывает таблицу, к которой он принадлежит, отображает параметры заголовка, статус активности и исходный код PSQL тела триггера.
SHOW USERS	Выводит список пользователей, соединенных с БД в настоящее время.
SHOW VERsion	Отображает информацию о программной версии isql и серверной программы РЕД База Данных, а также номер структуры на диске (ODS) подключенной базы данных.
SHOW {VIEWs | VIEW <имя>}	Отображает все представления или информацию об указанном представлении.
SHOW {JOBs | JOB <имя>}	Отображает все задания и краткую информацию о них: имя задания, владелец, состояние. Или показывает полную информацию о конкретном задании: имя задания, состояние, владелец, текст задания, расписание в формате cron.
SHOW WIRE_STATistics	Отображает сетевую статистику, собранную с момента установки соединения.

8.1.8 Команды SET

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

Таблица 8.4 — Команды SET

Команда	Описание
SET	Выводит на экран текущие установленные опции SET

(разрыв таблицы)

(разрыв таблицы)

Команда	Описание
`SET AUTO[DDL] [on|off]`	Задает, будут ли операторы DLL подтверждаться автоматически после их выполнения или будут подтверждаться после явного выполнения `COMMIT`. Оператор доступен в скриптах. Без аргументов - просто переключает `AUTODDL` между включено и выключено.
`SET BAIL [on|off]`	Задание данной опции определяет поручить или нет ошибку ОС, но только в пакетном режиме. Переключатель возвращает код ошибки операционной системе. Команда была добавлена для предотвращения выполнения скриптов после обнаружения ошибки.
`SET BLOB [n|all|off]`	Задает необходимость отображения подтипа `BLOB` и отображения данных `BLOB`. `n` - отображать `BLOB` заданного подтипа. Значение по умолчанию `n=1` (текст). Положительные числа определены в системе; отрицательные числа определяются пользователем. `all` - отображать данные `BLOB` любого подтипа. `off` - отключает отображение данных `BLOB`. Вывод показывает только идентификатор `BLOB` (два шестнадцатеричных числа, разделенных двоеточием).
`SET BULK_INSERT <подготовленный insert запрос>`	Примитивная обработка подготовленных запросов на вставку. После ввода этой команды пользователь входит в интерактивный режим (`BULK>`), где требуется ввести параметры в виде кортежа `(val1, ..., valN)`. Для выхода из интерактивного режима введите пустую строку или любой символ, отличный от `'('` и `'--'` (можно явно выйти, введя слово `stop`). Кортежи должны располагаться на одной строке, за исключением параметров в кавычках. Однако, если добавить `'+++'`} после запятой, параметры кортежа можно переносить на следующую строку. Одиночные комментарии (`'--'`) можно добавлять только между кортежами. Команды `commit` или `commit work` вводятся с первого символа строки и располагаются на одной строке. Любые символы, идущие за кортежем, кроме `')'` и `'+++'`, будут игнорироваться.
`SET COUNT [on|off]`	Включает/выключает отображение количества строк, найденных по запросам.
`SET DECFLOAT ROUND <режим>`	Изменение режима округления для типа `DECFLOAT`. Подробнее о типе `DECFLOAT` см. Руководство по SQL. Поддерживаются следующие режимы округления: • `CEILING` • `HALF_UP` • `HALF_DOWN` • `REROUND` • `UP` • `HALF_EVEN` • `FLOOR` • `DOWN`

(разрыв таблицы)

(разрыв таблицы)

Команда	Описание
SET DECFLOAT TRAPS TO <trap>[, <trap> ...]	Изменение обработки ошибок для типа DECFLOAT. Подробнее о типе DECFLOAT см. Руководство по SQL. Исключения могут генерироваться для следующих ситуаций: • Division_by_zero • Inexact • Invalid_operation • Overflow • Underflow
SET DECFLOAT BIND <bind-type>	Изменение отображения значений DECFLOAT на другие доступные типы данных. Подробнее о типе DECFLOAT см. Руководство по SQL. Допустимые типы для привязки: • NATIVE • DOUBLE PRECISION • CHAR[ACTER] • BIGINT[,<точность>]
SET ECHO [on|off]	Включает/выключает отображение команд до их выполнения.
SET EXPLAIN [on|off]	Включает/выключает расширенного подробного плана запроса. Этот план выводит более подробную информацию о методах доступа используемых оптимизатором, однако его нельзя включить в запрос.
SET GENERATOR <имя генератора> TO <значение>	Устанавливает значение последовательности или генератора в заданное значение.
SET HEADING [on|off]	Включает/выключает отображение заголовков столбцов.
SET KEEP_TRAN_PARAMS [on|off]	Включает/выключает сохранение параметров транзакции. Если установлено значение ON, то будет сохранён полный текст SQL следующего успешного оператора SET TRANSACTION, и новые транзакции запускаются с использованием того же текста SQL (вместо режима CONCURRENCY WAIT по умолчанию). Если установлено значение OFF, то новые транзакции будут запускаться с параметрами по умолчанию. Имя KEEP_TRAN можно использовать в качестве сокращения KEEP_TRAN_PARAMS.
SET LIST [on|off]	Задает формат отображения результатов запросов SELECT. По умолчанию данные на экране выводятся в табличном варианте, где сверху названия столбцов, а под ними все строки, удовлетворяющие запросу SELECT. Значение ON позволяет выводить результат в другом виде, где для каждой строки (результата запроса) выводится своя отдельная таблица (с заголовками столбцов слева).
SET LOCAL_TIMEOUT <значение>	Позволяет установить тайм-аут выполнения оператора (в милисекундах) для следующего оператора. После выполнения SQL оператора он автоматически сбрасывается в ноль.
SET NAMES <csname>	Задает набор символов, который будет активным в транзакциях базы данных.

(разрыв таблицы)

(разрыв таблицы)

Команда	Описание
`SET PLAN [on|off]`	Задает, нужно ли отображать план запроса оптимизатора.
`SET PLANONLY`	Задает только подготовку запросов `SELECT` и отображение плана без выполнения самого запроса.
`SET [TRUSTED] ROLE`	Изменяет текущую роль. Позволяет установить контекстной переменной `CURRENT_ROLE` одну из назначенных ролей для пользователя `CURRENT_USER` или роль, полученную в результате доверительной аутентификации (`SET TRUSTED ROLE`).
`SET {ROWCOUNT|MAXROWS} [<n>]`	Указывает какое максимальное количество строк будет выводится при выполнении оператора `SELECT`. Значение 0 (стоит по умолчанию) говорит о том, что ограничений нет.
`SET SQLDA_Display [on|off]`	Отображает или скрывает внутренние подробности об SQL-операторах, которые были выполнены isql.
`SET SQL DIALECT <n>`	Устанавливает SQL-диалект в то значение, которое было задано для сессии клиента. $n \in \{1, 2, 3\}$.
`SET STATs [on|off]`	Определяет, отображать ли статистику выполнения, которая будет следовать за выходными данными запроса. Формат вывода статистики: • `Current memory` - Объём памяти, используемой сервером (в байтах) • `Delta memory` - Показывает, на сколько байт изменилось использование памяти (значение `Current memory`) за время выполнения запроса • `Max memory` - Максимальное использование памяти, которое было зафиксировано с момента запуска сервера • `Elapsed time` - Время выполнения запроса • `Cpu` - Время использования CPU • `Buffers` - Количество кэшированных страниц • `Reads` - Количество страниц, считанных с диска • `Writes` - Страницы, записанные из кэша на диск (или в кэш операционной системы). `Insert`, `update`, `delete` - влияют на этот показатель, также влияет сборка мусора при `select`. • `Fetches` - Индикатор операций с памятью (включая `CPU`). Сколько было обращений к странице в кэше, чтение записей, чтение ключей, чтение страниц указателей, и т.п.
`SET TIME [on|off]`	Задает, отображать ли время в значении `DATE` (только диалект 1).
`SET TERM <строка>`	Задает символ, который будет использоваться в качестве терминатора команды или оператора. Он доступен в скриптах.
`SET TRANSACTION`	Запускает на выполнение транзакцию с заданными характеристиками.
`SET {WARNINGS|WNG} [on|off]`	Задает, выводить ли предупреждающие сообщения.

(разрыв таблицы)

(разрыв таблицы)

Команда	Описание
SET WIDTH <столбец> [<n>]	Используя команду set width, пользователь может регулировать ширину столбца на экране. Но только для character - столбцов. <n> - количество символов.
SET WIRE_stats [on|off]	Включает/выключает отображение сетевой статистики, которая отображается после выполнения запроса По умолчанию установлено значение OFF. Счетчики статистики разделены на две группы: логические и физические. Логические счетчики показывают количество пакетов в рамках сетевого протокола и количество отправленных (до сжатия) и полученных (после декомпрессии) байт. Физические счетчики показывают количество пакетов и байт, отправленных и полученных по сети. На количество байт может повлиять сжатие сетевого трафика, если оно включено. Также показывается количество передач сетевых пакетов: это число изменений сетевых вызовов с "отправки" на "получение" Сетевая статистика собирается только удаленным провайдером, то есть для встроенных соединений она всегда равна нулю. Так как сетевая статистика собирается на клиентской стороне, то направление сетевого ввода-вывода (отправлено, получено) отображается относительно клиента.
SET EXEC_PATH_DISPLAY {BLR | OFF}	Отладочная команда, показывающая BLR (скомпилированную форму) оператора.

8.2 Утилита GBAK

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

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

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

8.2.1 Права на запуск gbak

Для того, чтобы выполнить любую операцию над базой данных с помощью gbak, необходимо пройти процедуру идентификации и аутентификации — указать имя и пароль пользователя, который имеет право на запуск сервиса gbak. Это могут сделать SYSDBA, владелец базы, администратор с ролью rdb$admin или любой другой пользователь с привилегией USE_GBAK_UTILITY (подробнее см. *Распределение системных привилегий*). Для указания имени и пароля пользователя используются

переключатели `-user` и `-pa[ssword]`, соответственно, например:

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

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

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

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

8.2.2 Имена файлов

Имя базы

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

В следующем примере использован локальный коннект:

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

подразумевающий, что `gbak` выполняется на сервере.

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

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

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

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

> Если требуется осуществить резервное копирование в файл на стороне сервера, лучше использовать *сервисы*.

Имя резервной копии

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

8.2.3 Опции утилиты

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

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

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

Опция	Описание
-crypt <имя плагина>	Имя плагина для шифрования файла резервной копии незашифрованной базы данных или для шифрования файла базы данных, восстановленной из незашифрованной резервной копии. Его также можно использовать в сочетании с ключом -KEYNAME. Опцию -KEYHOLDER нужно указывать обязательно.
-d[irect_io]	Прямой ввод-вывод для файла резервной копии.
-fe[tch_password]	Получить пароль из файла.
-ig[nore]	Игнорировать ошибки контрольных сумм. Если указана эта опция, то в суперсервере другие подключения к базе данных будут запрещены на время выполнения бэкапа.
-keyholder <имя плагина>	Это плагин хранилища ключей шифрования, необходимый gbak для доступа к зашифрованной базе данных.
-keyname <имя ключа>	Явное присвоение имени ключу вместо ключа по умолчанию, указанного в исходной базе данных (при резервном копировании) или в файле резервной копии (при восстановлении).
-m[eta_data]	Производит резервное копирование или восстановление только метаданных.
-pas[sword]	Пароль пользователя, выполняющего операцию gbak (см. Права на запуск gbak).
-ro[le]	Подсоединиться с использованием роли (см. Права на запуск gbak).
-se[rvice]	Создаёт резервную копию на том же компьютере, где находится база данных-источник. Для этого вызывается Service Manager на компьютере-сервере.
-skip_d[ata] <шаблон>	Не сохранять в бэкап (или рестор) данные таблиц, имена которых соответствуют указанному здесь регулярному выражению в SQL-синтаксисе (см. оператор SIMILAR TO)[3].
-include[_data] <шаблон>	Включать только данные таблиц, имена которых соответствуют указанному здесь регулярному выражению в SQL-синтаксисе (см. оператор SIMILAR TO). Причем, если <шаблон> для -INCLUDE_DATA и -SKIP_DATA соответствует одной и той же таблице, то таблица пропускается.
-st[atistics] TDRW	Вывести статистику в процессе работы (работает вместе с -v(erbose) и -verbi(nt)). T — время от начала работы gbak; D — время прошедшее с последнего вывода на экран; R — число страниц прочитанных с последнего вывода на экран; W — число страниц записанных с последнего вывода на экран.
-tru[sted]	Использовать доверительную аутентификацию (Win_Sspi).
-user	Имя пользователя, выполняющего операцию gbak (см. Права на запуск gbak).
-v[erify]	Отчет о каждом выполненом действии
-verbi[nt] <n>	Подробный вывод лога действии с заданным интервалом обработанных записей. Минимальное значение - 100.

(разрыв таблицы)

(разрыв таблицы)

Опция	Описание
-y {<путь к файлу> | SUPPRESS}	Вывод лога в файл. Указывается совместно с -v(erify) или -verbi(nt). Если указан параметр suppress, не будет выведено никакой информации даже с опцией -v(erify).
-z	Показать версию gbak и версию сервера «РЕД База Данных».

Таблица 8.6 — Backup опции GBAK

Опция	Описание
-B[ACKUP_DATABASE] [O[VERWRITE]]	Основная опция при создании резервной копии. Если файл резервной копии уже существует, то резервное копирование не будет выполнено. Но если указать опцию O[VERWRITE], файл бэкапа будет заменен.
-co[nvert]	Преобразование внешних таблиц (external tables) во внутренние таблицы.
-e[xpand]	Не производить сжатие резервной копии. По умолчанию резервная копия записывается в "сжатом" виде.
-fa[ctor] n	Использовать блокирующий фактор n для ленточного накопителя. Устарел.
-g[arbage_collect]	Не собирать мусор во время резервного копирования[4].
-l[imbo]	Игнорировать зависшие двухфазные транзакции (limbo).
-nod[btriggers]	Предотвращает срабатывание триггеров базы данных во время резервного копирования.
-nt	Создаёт резервную копию в непереносимой формате между аппаратными платформами.
-ol[d_descriptions]	Производит резервное копирование метаданных в формате старого стиля, т.е. в режиме совместимости со старыми базами данных. Устарел.
-par[allel] <n>	Количество параллельных рабочих потоков для бэкапа. Включает многопоточное выполнение резервного копирования.
-t[ransportable]	Создаёт транспортабельную (переносимую) резервную копию. Этот параметр включен по умолчанию, поэтому указывать его нет никакой необходимости.
-zip	Сжать файл резервной копии перед его шифрованием. После шифрования файл почти не сжимается. Для сжатия используется плагин zstd с 5 уровнем сжатия в 1 поток. Такой сжатый файл может быть распакован только восстановлением gbak. Ключ -zip не нужен для восстановления, потому что формат определяется автоматически.

(разрыв таблицы)

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

(разрыв таблицы)

Опция	Описание
-com[pressor] [<имя плагина> [<уровень сжатия> [<кол-во потоков>]]]	Сжать файл резервной копии перед его шифрованием. После шифрования файл почти не сжимается. С РЕД Базой Данных поставляются плагины zstd и zlibcompressor. Сжатие плагином zstd выполняется быстрее, итоговый размер резервной копии будет меньше, чем при использовании zlibcompressor. Плагин zstd поддерживает 22 уровня сжатия, а zlibcompressor 9. Плагин zlibcompressor выполняет сжатие в 1 поток и проигнорирует указанное количество потоков. По умолчанию используется плагин zstd с 5 уровнем сжатия в 1 поток. Такой сжатый файл может быть распакован только восстановлением gbak. Ключ -com не нужен для восстановления, потому что формат определяется автоматически.

Таблица 8.7 — Restore опции GBAK

Опция	Описание
-C[REATE_DATABASE]	Восстанавливает (создаёт) базу данных из бэкапа в новый файл.
-R[ECREATE_DATABASE] [O[VERWRITE]]	Создаёт новую базу данных или заменяет (если указана опция O) существующую базу данных из бэкапа.
-REP[LACE_DATABASE]	Восстанавливает файл базы данных, заменяя имеющуюся базу данных.
-bu[ffers]	Задать размер страничного кэша для БД.
-fix_fss_d[ata] <кодировка>	Исправить кодировку данных
-fix_fss_m[etadata] <кодировка>	Исправить кодировку метаданных
-i[nactive]	Деактивировать индексы во время восстановления. При использовании опции в логе восстановления будет сообщение "gbak: WARNING:Database is not online due to inactive indexes (option -I)". Перед возвращением базы данных в режим "online" не забудьте активировать индексы.
-k[ill]	Восстановить без создания теневых копий
-mo[de] <режим доступа>	Режим доступа "read_only" или "read_write"
-n[o_validity]	Отключение всех ограничений проверки данных (check) и ограничений NOT NULL. При использовании опции в логе восстановления будет сообщение "gbak: WARNING:Database is not online due to skipped restoring of validity conditions (option -N)". Перед возвращением базы данных в режим "online" не забудьте пересоздать отключенные ограничения.
-o[ne_at_a_time]	Подтверждать транзакцию после восстановления каждой таблицы.
-p[age_size]	Установить размер страницы
-replica <режим>	Настраивает режим реплики восстанавливаемой базы данных: "none", "read_only" или "read_write".

(разрыв таблицы)

[4] «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**.

(разрыв таблицы)

Опция	Описание
-par[allel] <n>	Количество параллельных рабочих потоков для рестора. Включает многопоточное выполнение восстановления БД, если MaxParallelWorkers > 1 (см. firebird.conf). Вместо этого параметра утилиты можно указать параметр конфигурации ParallelWorkers.
-use_[all_space]	Не резервировать место под версии записей. Максимально заполнять страницы данных (для read-only баз).
-rdb_map[ping_file] <файл с отображением>	Для корректного восстановления БД с версии 2.X, если ее бэкап содержит внешние Java функции или процедуры. Необходимость в этой опции возникла по причине изменения синтаксиса: в версии 2.X при объявлении внешней функции(процедуры) не указывались ее аргументы, в отличии от версии 3.0 и выше. Поэтому, если функции(процедуры) имеют аргументы, следует создать текстовый файл <файл с отображением>, где первый столбец - имя функции(процедуры) в java, а во втором - типы аргументов: package.class.Function1 arg1,arg2,arg3 package.class.Function2 arg1,arg2,arg3 В этом случае нужная java функция (процедура) будет подбираться на лету просто по имени.
-ts_map[ping] <файл с отображением>	Для корректного восстановления БД, если ее бэкап содержит таблицы или индексы, сохраненные в табличных пространствах. Текстовый файл <файл с отображением> должен состоять из строк с двумя значениями: первый столбец - имя табличного пространства, второй - новое расположение табличного пространства для восстановленной базы данных. Можно указывать как абсолютный путь, так и относительный, в том числе с использованием псевдонима директории, заданного в directories.conf в секции tablespaces. tablespace1 d:\db\restore\ts1.dat tablespace2 d:\db\restore\ts2.dat В этом случае табличные пространства копируются с новыми путями.
-ts <tablespace> <path>	Позволяет задать путь для табличного пространства. Можно указывать как абсолютный путь, так и относительный, в том числе с использованием псевдонима директории, заданного в directories.conf в секции tablespaces. Опцию можно использовать столько раз, сколько требуется. Также ее можно использовать совместно с -ts_map.
-ts_orig[inal_paths]	Для восстановления табличных пространств по первоначальным путям, по которым они располагались в момент создания бэкапа. При этом все еще можно переопределять пути для некоторых табличных пространств с помощью опций -ts и -ts_map.

8.2.4 Создание резервной копии

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

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

произведены приложениями во время работы gbak, разумеется не будут сохранены в резервную копию БД.

Формат командной строки для создания резервной копии:

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

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

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

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

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

Теперь поподробнее остановимся на других опциях бэкапа.

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

Полный вывод лога действий.

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

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

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

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

Так выглядит команда с ключом -v:

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

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

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

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

Так выглядит команда с ключом -verbi:

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

-y { <имя файла> | SUPPRESS }

Перенаправление лога в файл или полное подавление записи лога.

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

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

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

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

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

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

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

gbak:cannot open status and error output file e_bak.txt
gbak:Exiting before completion due to errors
```

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

-g[arbage_collect]

Отключает сборку мусора

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

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

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

-t[ransportable]

Создание транспортабельной (переносимой) резервной копии.

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

Он означает, что полученный файл резервной копии можно восстановить на альтернативной аппаратной платформе, где порядок байт в целых числах отличается. То есть, например, между Intel и Sparc, или HP-UX и Intel, и так далее. Но между Windows и Linux (или другой ОС) на Intel файл резервной копии будет и так переносимым, даже при указании ключа `-nt` (non-transportable). Так что, про `-t`, как и про `-nt` можно забыть, и никогда не указывать их в командной строке `gbak`.

-e[xpand]

Сжатие (низкой степени) файла резервной копии

По умолчанию `gbak -b` производит некую легкую компрессию данных. Отключить ее можно параметром `-expand`.

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

Для лучшего сжатия резервной копии используйте опцию `-ZIP`.

-co[nvert]

Преобразование внешних таблиц в обычные таблицы.

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

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

-ig[nore]

Игнорирование ошибок контрольных сумм страниц

Если произошло повреждение базы данных, и данные на страницах БД искажены (нули или другая произвольная информация), то сервер при чтении такой страницы будет писать информацию об ошибке контрольной суммы в лог.

Параметр `-ig` позволяет игнорировать ошибки контрольных сумм страниц. Если во время резервного копирования возникла ошибка при чтении блоба, то блоб будет пропущен, если установлен ключ `-ig`.

> Поэтому, в обычной командной строке для создания резервной копии БД категорически не рекомендуется указывать параметр `-ig`. Если база данных вдруг окажется повреждена, то с этим параметром вы можете "не увидеть" повреждение. Так что, `-ig` для `gbak` используется только в крайнем случае – когда поврежденная база починена утилитой `gfix`, но обычный backup не проходит. Только тогда имеет смысл повторить бэкап с опцией `-ig`.

Если указана эта опция, то в суперсервере другие подключения к базе данных будут запрещены на время выполнения бэкапа.

-m[eta_data]

Выполняется резервное копирование или восстановление только метаданных.

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

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

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

-l[imbo]

Игнорирование limbo-транзакций

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

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

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

-crypt <имя плагина>

Имя плагина для шифрования файла резервной копии или восстановленной базы данных

Опцию `-crypt` указывают совместно с опцией `-KEYHOLDER` (обязательно). Также может указываться опция `-KEYNAME`.

При создании резервной копии *незашифрованной базы данных* в опции `-crypt` указывается плагин шифрования, который будет использоваться для шифрования файла резервной копии.

При создании резервной копии *зашифрованной базы данных* указывать `-crypt` не нужно, поскольку по умолчанию будет использоваться тот же плагин и ключ, что и для самой базы данных. Невозможно создать резервную копию зашифрованной базы данных в незашифрованный файл резервной копии.

При восстановлении *незашифрованной резервной копии* опция `-crypt` зашифрует новую базу данных с использованием указанного плагина.

При восстановлении базы данных из *зашифрованной резервной копии* указывать `-crypt` не нужно, поскольку по умолчанию будет использоваться тот же плагин и ключ, что и для самой резервной копии. Невозможно восстановить зашифрованный файл резервной копии в незашифрованную базу данных.

Пример

Чтобы сделать зашифрованную резервную копию незашифрованной базы данных, выполните:

```
gbak -b -keyholder MyKeyHolderPlugin -crypt MyDbCryptPlugin -keyname
SomeKey host:dbname encrypted_backup_file
```

-keyholder <имя плагина>

Имя плагина хранилища ключей шифрования базы данных.

Опцию обычно используют в сочетании с опциями `-CRYPT` и `-KEYNAME` (необязательно).

Опция `-keyholder` должна указываться:

- при создании резервной копии зашифрованной базы данных (соответственно, без опции `-crypt`);
- при создании зашифрованной резервной копии незашифрованной базы данных (совместно с опцией `-crypt`);
- при восстановлении из зашифрованной резервной копии в зашифрованную базу данных (без опции `-crypt`);
- при восстановлении из незашифрованной резервной копии в зашифрованную базу данных (совместно с опцией `-crypt`).

Пример 1

Чтобы сделать бэкап зашифрованной базы данных, выполните подобную команду:

```
gbak -b -keyholder MyKeyHolderPlugin host:dbname backup_file_name
```

Файл резервной копии будет зашифрован с использованием того же плагина и ключа шифрования, которые используются для шифрования базы данных. Это гарантирует, что украсть данные из файла резервной копии будет не легче, чем из базы данных.

Пример 2

Чтобы создать сжатую зашифрованную резервную копию, выполните:

```
gbak -b -keyholder MyKeyHolderPlugin -zip host:dbname backup_file_name
```

Файл резервной копии будет сжат до того, как он будет зашифрован с использованием того же плагина шифрования и того же ключа, который используется для шифрования базы данных.

Пример 3

Чтобы восстановить базу данных, которая была ранее зашифрована, выполните следующую команду:

```
gbak -c -keyholder MyKeyHolderPlugin backup_file_name host:dbname
```

Восстановленная база данных будет зашифрована с использованием того же плагина и ключа, что и файл резервной копии. В данном примере это тот же плагин и ключ, что и в исходной базе данных.

-keyname <имя ключа>

Имя ключа, который будет использоваться для шифрования.

Опция `-keyname` может указываться (необязательно, если этого требует плагин шифрования):

- при создании зашифрованной резервной копии незашифрованной базы данных (совместно с опцией `-crypt` и `-keyholder`);
- при восстановлении из зашифрованной резервной копии в зашифрованную базу данных с именем ключа, отличным от значения по умолчанию (без опции `-crypt`, но с опцией `-keyholder`);
- при восстановлении из незашифрованной резервной копии в зашифрованную базу данных (совместно с опцией `-crypt` и `-keyholder`).

Пример

С помощью следующей команды можно:

- либо восстановить базу данных из файла резервной копии, созданного с использованием нестандартных плагинов шифрования Crypt и Keyholder, используя тот же Keyname, который использовался для резервного копирования;
- либо восстановить незашифрованную резервную копию как зашифрованную базу данных.

```
gbak -c -keyholder MyKeyHolderPlugin -crypt MyDbCryptPlugin -keyname
SomeKey non_encrypted_backup_file host:dbname
```

-d[irect_io]

Прямая запись-чтение файлов резервной копии.

При включении этой опции `gbak` создает (при бэкапе) или открывает (при восстановлении), файл резервной копии в режиме прямой записи-чтения (без буферизации). В этом режиме не задействуется кэш файловой системы.

Обычно резервная копия записывается (при бэкапе) или считывается (при восстановлении) всего один раз, и нет реальной выгоды от кэширования ее содержимого. Производительность не должна падать, так как `gbak` читает файл или записывает в него последовательно, используя относительно большие порции данных.

Режим прямой записи-чтения игнорируется, если файл резервной копии записывается в стандартный вывод или считывается из стандартного ввода (т.е. если имя файла резервной копии - `stdout` или `stdin`).

-fe[tch_password]

Считать пароль из файла

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

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

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

-par[allel] <n>

Количество потоков, используемых для создания резервной копии.

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

По умолчанию используется один поток.

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

Для включения распараллеливания в `gbak` выполните команду:

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

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

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

-skip_d[ata] <шаблон>

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

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

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

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

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

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

-include[_data] <шаблон>

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

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

Противоположная опция -SKIP_D[ATA].

Если <шаблон> для -INCLUDE_DATA и -SKIP_DATA соответствует одной и той же таблице, то таблица пропускается.

Выборочное включение данных таблиц в резервную копию или восстановленную базу данных может вызвать ошибки во время восстановления, если существует внешний ключ в таблице, которая была включена, связанный с таблицей, которая не была включена.

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

Замечания

Резервное копирование может быть выполнено в любой момент, во время работы пользователей с базой данных. Эту операцию можно и нужно автоматизировать, сделав резервное копирование регулярным. Без резервных копий есть риск потерять данные, если база данных окажется повреждённой по какой-либо причине.

Категорически нельзя делать резервные копии на тот же самый логический диск, где находится база данных. Еще лучше делать резервные копии на другой физический диск, поскольку чтение и запись будут разделены, и это даст как минимум 30% ускорение процесса резервного копирования.

8.2.5 Восстановление базы данных из резервной копии

Для восстановления базы из резервной копии действует следующая команда:

```
gbak -C[REATE_DATABASE] [restore опции] [общие опции] <файл резервной копии> <база данных>
```

Ключ -C означает, что необходимо восстановить базу данных из резервной копии, путь к которой указан как <файл резервной копии>, а результаты восстановления сохранить в файл, указанный как <база_данных>. Недостающие каталоги, если такие имеются, в пути к файлу будут автоматически созданы. Файл <база_данных> не должен существовать, иначе произойдёт ошибка.

Вместо данной команды можно использовать -R[ECREATE_DATABASE] [O[VERWRITE]] (восстанавливает БД в новый файл или восстанавливает ее вместо старой, если используется параметр o) и -REP[LACE_DATABASE] (восстанавливает БД, заменяя имеющуюся базу данных).

```
gbak -c emp.fbk emp.fdb
```

База emp.fdb будет восстановлена из резервной копии emp.fbk. Процесс восстановления базы данных из резервной копии происходит следующим образом:

1. Сервер создает пустую базу данных emp.fdb. ODS базы данных будет определяться версией сервера, а не версией gbak, т. к. только сервер создает базу данных, а gbak при restore всего лишь ее наполняет метаданными и данными.

   Пустая база данных будет содержать все таблицы rdb$ (пока пустые), и будет иметь ряд

параметров, например, такие как размер страницы, forced write и т. д., которые или взяты из резервной копии, или установлены в командной строке `gbak`.

2. Сервер считывает метаданные (описания таблиц и индексов, процедур) из резервной копии и переносит их в базу данных.
3. Сервер считывает данные из резервной копии и переносит их в базу данных.
4. Сервер считывает остальные метаданные (триггеры, гранты, check constraints и т. п.) из резервной копии и переносит их в базу данных
5. Сервер создает (активирует) все индексы таблиц (которые были активны в момент создания резервной копии).

То есть, восстановленная из резервной копии база данных будет на самом деле не копией старой базы, а совершенно новой базой данных, наполненной старыми данными.

Если в процессе восстановления из резервной копии возникает ошибка в `BLR` процедуры, функции или триггера, то рестор продолжится, а в лог будет добавлено предупреждение. `BLR` такого объекта будет отмечен, как невалидный (поле `RDB$VALID_BLR` устанавливается в 0). После завершения рестора невалидные объекты необходимо перекомпилировать.

Помните, что если вы делаете восстановление на новой версии сервера, например, резервную копию делали на РЕД Базе Данных 3 (формат БД ODS 12.3), а восстанавливаете на РЕД Базе Данных 5 (формат БД ODS 13.1), то база будет создана в формате, поддерживаемом по умолчанию РЕД Базой Данных 5, и РЕД База Данных 3 с этой базой работать не сможет.

Узнать версию ODS поможет команда:

```
gstat -h <база_данных>
```

Резервные копии тоже имеют свой формат, и резервная копия базы данных, сделанная, например, в РЕД Базе Данных 5 утилитой `gbak` этой же версии, не может быть восстановлена утилитой `gbak` от РЕД Базы Данных 3.

Теперь поподробнее остановимся на опциях бэкапа. О некоторых общих опциях (для бэкапа и рестора) : -crypt, -keyholder , -keyname, -d[irect_io], -fe[tch_password], -ig[nore], -m[eta_data], -skip_d[ata], -include[_data], -v[erify], -verbi[nt], -y уже шла речь в предыдущем разделе. О дополнительных параметрах restore рассказано ниже.

-R[ECREATE_DATABASE] [O[VERWRITE]]

Восстановление в новую БД, при необходимости позволяя перезаписать существующую БД.

Эта команда аналогична команде `-C[REATE_DATABASE]`, которая выполняет восстановление базы данных в новый файл. Но если указанный файл уже существует, будет выдано сообщение об ошибке.

Избежать такую ошибку поможет опция `o[verwrite]`. В таком случае файл с существующей базой данных будет молча удален и вместо него создан новый с тем же именем. Этого допускать нельзя, потому что по разным причинам восстановление из резервной копии может не состояться, и тогда вы останетесь без оригинальной базы данных и с невосстановимой резервной копией.

Если вы замените открытую и работающую базу данных, есть большая вероятность, что вы ее испортите. Для достижения наилучших результатов и минимальной вероятности повреждения базы данных следует закрыть ее перед заменой. Чтобы закрыть базу данных, используйте `gfix` следующим образом:

```
gfix -shut -tran 60 employee.fdb
```

В приведенном выше примере предотвращается запуск любой новой транзакции, что предотвращает выполнение новых запросов или новых сеансов подключения к базе данных. Прежде чем завершить работу базы данных, `gfix` будет ждать до 60 секунд, пока все выйдут из системы и завершится все текущие транзакции. Если какие-либо длительные транзакции не завершатся по истечении 60 секунд, время завершения работы истечет, а база данных останется открытой.

> Не рекомендуется использовать команду `-r[ecreate_database] o[verwrite]` с целью избежания потери данных.

Использование `-r[ecreate_database] o[verwrite]` фактически аналогично использованию `-rep[lace_database]`.

-REP[LACE_DATABASE]

Восстановление в базу данных, позволяя перезаписать существующую базу данных.

Если база данных уже существует, то она будет удалена перед восстановлением. Этого допускать нельзя, потому что по разным причинам восстановление из резервной копии может не состояться, и тогда вы останетесь без оригинальной базы данных и с невосстановимой резервной копией.

Команда `-rep` полностью аналогична `-r o`.

> Не рекомендуется использовать команду `-rep` с целью избежания потери данных.

-bu[ffers] <n>

Изменяет размер кэша базы данных.

По умолчанию кэш базы данных задан в файле конфигурации сервера (параметр `DefaultDbCachePages` в `firebird.conf`), и равен 1024 страниц. Это значение действует для всех баз данных, у которых размер кэша задан неявно. Если вы используете на сервере несколько баз данных, то может потребоваться указать для них разный размер кэша, в зависимости от назначения этих баз.

Узнать текущий размер кэша поможет команда `gstat -h <база данных>`.

Параметр `-bu <n>` позволяет при восстановлении базы данных задать или изменить этот размер. К сожалению, у `gbak` можно указать значение `-bu` только больше 0. Если вы обнаружили, что в бэкапе уже определено значение кэша, то "сбросить" его при восстановлении не получится. Для убирания размера кэша в БД придется использовать `gfix`.

Если размер кэша вообще задан в БД (а в большинстве случаев это не делают), то этот параметр используется обычно при переносе базы данных, например, со старого сервера на новый, или при переносе БД между архитектурами Classic и SuperServer.

Эквивалентно команде `gfix -bu <n>`.

```
gstat -h employee.fdb | grep -i buffer   # Проверить текущий размер кэша
  Page buffers            0

gbak -c -buffer 200 /backups/employee.fbk employee.fdb
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
gstat -h employee.fdb | grep -i buffer
  Page buffers            200
```

-fix_fss_d[ata], -fix_fss_m[etadata]

Изменение набора символов метаданных (и данных)

Опции `-fix_fss_metadata` и `-fix_fss_data` предназначены для исправления набора символов (charset) метаданных (и данных), которые были записаны в некорректной кодировке. Например, при редактировании процедур или триггеров в код могли попасть комментарии или константы в кодировке `none`, в то время когда они на самом деле являются символами `Windows-1251`.

В штатной ситуации указывать эти опции не требуется, но если при восстановлении `gbak` выводит сообщение об ошибке:

```
Malformed string
```

то нужно принудительно указать нужную кодировку при помощи указанных опций. После этого в столбцах Unicode данные будут записаны корректно.

`gbak` также может выдавать сообщение:

```
gbak:Invalid metadata detected. Use -FIX_FSS_METADATA option.
```

что сигнализирует об аналогичной ситуации, и требует явного указания данной опции с нужной кодировкой.

> Указывать опции `-fix...` можно **только один раз**. Если вы сделаете еще раз backup/restore этой же базы данных с этими опциями, то исходные тексты процедур и триггеров будут испорчены.

Указание неправильного имени набора символов может привести к логическому повреждению ваших данных.

> Не используйте эту опцию без четкого понимания того, что она делает. Неправильное использование может повредить ваши данные вместо того, чтобы что-то исправить. Всегда сохраняйте копию исходной базы данных и ее резервную копию.

-i[nactive]

Не выполнять создание (активацию) индексов (последняя фаза восстановления).

После восстановления базы данных все индексы в ней останутся отключены (неактивны). Фактически с такой базой данных работать нельзя, т. к. при отключенных индексах `Primary key`, `Foreign key` и `Unique` возможны нарушения целостности данных в таблицах (дублирование первичных ключей и т. д.).

Данный параметр имеет смысл использовать разве что в специфических целях – например, восстановить только данные из бэкапа за максимально быстрое время, или определить длительность создания всех индексов (замерить время `gbak -c`, затем замерить время `gbak -c -i`, после чего вычесть одно время из другого – получите длительность активации всех индексов), чтобы определить или текущую производительность каталога `temp`, или сравнить ее с явно заданным в конфигурации расположении `temp` на другом физическом диске.

Если вы восстановили бэкап с опцией `-i`, индексы можно активировать командой `alter`

`index <имя индекса> active` (для каждого индекса).

-k[ill]

Восстановить базу данных без создания теневый копий

Эта опция восстанавливает базу данных, но не воссоздает ранее существовавшие теневые копии. На самом деле этот параметр не только "не создает" shadow, но и еще удаляет существующие, например в варианте с backup/restore с промежуточным переименованием базы данных.

```
gbak -b -kill /backups/employee.fbk employee.fdb
```

Эквивалентно команде `gfix -kill`.

-mo[de] <режим доступа>

Восстанавливает базу данных в режиме read_ write или read_ only

```
-MO[DE] { READ_ONLY | READ_WRITE }
```

По умолчанию режим берется от базы данных, для которой была создана резервная копия.

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

Эквивалентно команде `gfix -mode read_write/read_only`.

> Эту опцию не следует путать с режимом реплики, настроенным с помощью `-REPLICA`. Например, база данных, созданная с параметром `-REPLICA READ_ONLY`, по-прежнему доступна для записи при подключении репликатора, а база данных, созданная с параметром `-MODE READ_ONLY`, вообще не доступна для записи.

-n[o_validity]

Отключает ограничения CHECK

Эта опция аналогична `-i[nactive]`, за исключением того, что она отключает ограничения `CHECK` в востанавливаемой базе данных.

-o[ne_at_a_time]

Подтверждать транзакцию после восстановления каждой таблицы.

Эта опция восстанавливает данные потаблично, используя транзакцию для каждой таблицы. Она может быть полезна, если предыдущее восстановление не удалось из-за ошибок данных. Обычно восстановление выполняется за одну транзакцию с одним коммитом в конце восстановления. Если восстановление по какой-либо причине прерывается, конечным результатом является пустая база данных. С опцией `-o[ne_at_a_time]` запускается транзакция для каждой таблицы и после восстановления каждой таблицы выполняется коммит.

-p[age_size] <размер страниц>

Позволяет задать новый размер страницы базы данных.

По умолчанию база данных восстанавливается с тем же размером страницы базы данных, что и исходная база данных (как записано в файле резервной копии).

Посмотреть размер страницы базы данных можно с помощью команды `gstat -h`.

```
gstat -h employee | grep -i "page size"
    Page size               4096

gbak -b -page_size 8192 /backups/employee.fbk employee.fdb

gstat -h employee | grep -i "page size"
    Page size               8192
```

Сделать рестор с параметром `-p` - это единственный способ, которым можно изменить размер страницы базы данных. В зависимости от версии допустимые размеры страниц: 1024, 2048, 4096, 8192, 16384 и 32768 байт. Если у базы данных размер страницы 1024 или 2048 байт – рекомендуется сделать backup и restore с указанием размера страницы 4096, 8192 или 16384 байт (если ваша версия сервера поддерживает страницы в 16к). Потому что даже небольшие базы данных с таким небольшим размером страницы имеют явно худшую производительность, чем базы со страницей 4 килобайта и выше.

-par[allel] <n>

Количество потоков, используемых для восстановления из резервной копии.

РЕД База Данных (начиная с версии 3.0) поддерживает многопоточный restore. Он показал себя до 2-х раз быстрее обычного рестора в один поток. Реальное ускорение зависит от процессора, дисковой подсистемы и структуры базы данных.

По умолчанию используется один поток. Но для рестора это не идентично явному указанию `-PARALLEL 1`.

Для включения распараллеливания в `gbak` появилась опция `-par <n>`:

```
gbak -c backup database -par 8 ...
```

а также два параметра конфигурации `ParallelWorkers` и `MaxParallelWorkers`.

Для рестора параллельная обработка реализована только при построении индексов. При этом внутри ядра создаются `N` потоков, где `N` - либо значение, переданное из `gbak`, либо значение параметра конфига `ParallelWorkers` (если при ресторе не задан ключ `-PAR`). `N` должен быть меньше значения параметра конфигурации `MaxParallelWorkers` (максимальное допустимое значение - 64), иначе операция завершится с ошибкой. По умолчанию `MaxParallelWorkers = 1`, т.е. параллельное создание индексов отключено даже если в ресторе задан ключ `-PAR`.

Параллельные потоки одновременно читают таблицу и сортируют её записи. Затем один основной поток строит дерево сортировки (B+tree) и создаёт индекс.

> Т.к. при параллельном создании индексов рабочие потоки внутри ядра создают собственные подключения к базе, она не может быть восстановлена в режиме `single shutdown` как это было раньше. Поэтому при использовании опции `-par` при ресторе база создаётся в режиме `multi shutdown`, т.е. в процессе рестора к ней может подключиться `SYSDBA` или владелец базы и сделать с ней что хочет.

-replica

Настраивает режим реплики восстанавливаемой базы данных.

```
-REPLICA { NONE | READ_ONLY | READ_WRITE }
```

Режим реплики базы данных хранится в файле резервной копии. При восстановлении по умолчанию этот режим реплики устанавливается для вновь созданной базы данных.

Опция `-REPLICA` явно устанавливает режим реплики, переопределяя значение, унаследованное от резервной копии. Например, значение `NONE` сделает базу данных основной (или обычной), `READ_ONLY` помечает базу данных как реплику, доступную только для чтения, а `READ_WRITE` — как реплику для чтения/записи.

После восстановления режим реплики базы данных можно изменить с помощью `gfix -replica { NONE | READ_ONLY | READ_WRITE } <база данных>`.

-use_[all_space]

Максимально заполнять страницы данных (для read-only баз).

По умолчанию базы данных РЕД Базы Данных резервируют примерно 30% пространства на страницах данных, для размещения версий при будущих вставках, удалениях или обновлениях записей. Если предполагается запись базы данных на диск, то лучше базу данных несколько "сжать", указав параметр `-use_` при restore (одновременно указав `-mode read_only`).

Эквивалентно команде `gfix -use full`. Обратная команда (снятие режима максимального заполнения страниц) – `gfix -use reserve`.

Замечания

Восстановление (например, проверочное) на тот же самый диск, где находится резервная копия, не так опасно, как в случае backup (когда свободное место может кончится, и резервная копия будет неполной, да еще и база данных может быть повреждена). Но разумеется имеет те же самые проблемы с производительностью, когда восстановление проводится на тот же самый физический диск (поочередные чтение и запись с разных участков диска). Поэтому при восстановлении базы данных из резевной копии рекомендуется использовать разные физические диски.

8.2.6 Работа с GBAK через Services API

Как уже было сказано, `gbak` - это программа, которая сама выполняет команду резервного копирования, "прокачивая" данные через себя. Но утилита может работать и в другом режиме, используя Services API для выполнения сервером по команде действий, аналогичных `gbak`. Утилита `gbak`, использующая Services API, отправляет серверу команду, а сервер ее выполняет сам (то есть не `gbak`, а сам сервер будет выполнять резервное копирование). В этом случае программа, отдавшая команду, может получать лог выполнения от сервера и выдавать все, что сообщит о процессе сервер. Чтобы включить работу `gbak` в этом режиме, используйте опцию `-se`:

```
gbak -b -g -se server:service_mgr c:\db\e.fdb d:\bak\e.fbk ...
```

```
gbak -c -se server:service_mgr d:\e.fbk c:\db\e.fdb
```

`server` – имя компьютера, где находится сервер РЕД Базы Данных (если на этом же, то можно указать `localhost`). `server` можно не указывать, если сервер "локальный", и работает локальный протокол:

```
gbak -b -g -se service_mgr c:\db\e.fdb d:\bak\e.fbk ...
```

`service_mgr` – имя интерфейса Services API, оно обязательно, неизменно, и пока только одно (может быть в дальнейшем появится что-то еще).

Поскольку не `gbak`, а сам сервер теперь выполняет резервное копирование, то есть несколько требований:

- пути к базам (или алиасы) должны быть указаны только серверные, как для базы так и для файла резервной копии.
- имя сервера к имени базы данных добавлять не нужно, т. к. оно уже должно быть указано как опция команды `-se`.
- у сервера должны быть права на запись туда, куда сохраняется резервная копия. На Windows сервер по умолчанию стартует под учетной записью LocalSystem, которая не имеет и не может иметь прав на внешние ресурсы (например, папки с общим доступом). Поэтому, если вы хотите сохранять резервные копии БД на другой компьютер сразу, а не путем копирования получившегося на сервере файла backup – нужно создать пользователя (например, `reddatabase`), дать этому пользователю права на папки установки сервера, папки с базами данных и папки с резервными копиями, и затем остановить сервер и запустить его указав в параметрах сервиса новое имя пользователя.

По результатам тестирования резервное копирование через Services API является самым быстрым, по сравнению с локальным протоколом или tcp/ip (результаты тестирования). Однако, если потребуется прекратить процесс backup или restore, то:

- для `gbak -b/-c` достаточно принудительно снять (завершить) процесс `gbak.exe`, или если он запущен интерактивно, нажать в окне cmd Ctrl-C. Поскольку backup или restore выполняется не сервером, а утилитой `gbak`, этот процесс будет прекращен.
- для backup и restore, выполняемых через Services API, это возможно только остановкой сервера, т. к. именно он выполняет этот процесс.

8.3 Утилита NBACKUP

`Nbackup` позволяет создавать резервные копии и восстанавливать из резервных копий так же, как `gbak`, и дополнительно позволяет создавать инкрементные копии и восстанавливать из них БД. Инкрементная резервная копия содержит только изменения со времени создания определенной, ранее созданной резервной копии.

Второе назначение `Nbackup` – блокировать файл базы данных. Таким образом, Вы после этого сможете сами создавать обычные копии или резервные копии с помощью утилит по Вашему выбору. В этом режиме `Nbackup` ничего не резервирует, а лишь создает подходящие условия, чтобы Вы могли без каких бы то ни было проблем создавать резервные копии.

Таблица 8.8 — Опции основных задач утилиты `NBACKUP`

Опция	Описание
`-L(OCK) <база данных>`	Блокировка базы данных для резервирования. Это означает, что основной файл базы данных временно замораживается с возможностью внесения изменений. Как и в режиме резервирования, изменения фиксируются во временном файле дельты.

(разрыв таблицы)

(разрыв таблицы)

Опция	Описание
`-UN(LOCK) <база данных>`	Разблокировка ранее заблокированной базы данных. Сначала определяется наличие любых изменений с момента блокирования базы данных и производится объединение временного файла дельты и основного файла базы данных. После этого база данных переводится в нормальный режим чтения/записи, а временный файл удаляется.
`-F(IXUP) <база данных>`	Разблокировка базы данных после самостоятельного восстановления из блокированной резервной копии. Так как копия блокированной базы данных является так же блокированной, поэтому не получится использовать копию как рабочую базу данных. Поэтому, если исходная база данных повреждена или утеряна, то нужно восстановить-/разархивировать/скопировать базу данных из копии и разблокировать ее с параметром `-F` (но не `-UN`).
`-B(ACKUP) <уровень>|<GUID> <база данных> [<резервный файл>]`	Создание резервной копии всей базы данных или инкрементных резервных копий. Вместо уровня инкрементной резервной копии можно указать GUID бэкапа из таблицы в `RDB$BACKUP_HISTORY`. Он формируется при каждом бэкапе. В этом случае инкрементный бэкап можно делать относительно любой копии (а не только относительно предыдущего уровня)[6]
`-R(ESTORE) <база данных> [<резервный файл 0> [<резервный файл 1>...]]`	Восстановление из резервной копии всего файла базы данных или из инкрементных резервных копий
`-SEQ[UENCE]`	Сохраняет существующий `GUID` и последовательность репликации исходной базы данных (в противном случае они сбрасываются). Эта опция должна использоваться при создании реплики, чтобы асинхронная репликация могла быть автоматически продолжена с того момента, когда было выполнено физическое резервное копирование на основной базе

Таблица 8.9 — Специальные опции утилиты `NBACKUP`

Опция	Описание
`-D(IRECT) [ON | OFF]`	Включает/выключает небуферизованный ввод/вывод при чтении БД
`-I(NPLACE)`	Восстанавливает инкрементную копию в существующую базу данных[7] 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".
`-S(IZE)`	Вывести количество страниц в базе после блокировки БД[8].
`-DE(COMPRESS) <команда>`	Команда для разархивирования бэкапа во время восстановления. Символ `@` в команде соответствует файлу бэкапа.

[6] GUID базы данных можно узнать из вывода утилиты `gstat` с параметром `-h`
[7] Работает только с опцией `-R`
[8] Работает только с опцией `-L(OCK)`

Таблица 8.10 — Общие опции утилиты NBACKUP

Опция	Описание
-NOD(BTRIGGERS)	Не запускать триггеры базы данных
-U(SER) <имя пользователя>	Имя пользователя
-P(ASSWORD) <пароль пользователя>	Пароль пользователя
-RO(LE) <роль>	Роль пользователя
-FETCH_PASSWORD <файл>	Считывает пароль из файла
-Z	Выдает версию СУБД
-CLEAN_HISTORY	Очистить таблицу RDB$HISTORY
-KEEP_DAYS <число>	Определяет сколько дней, начиная с сегодняшнего дня, должно храниться в истории
-KEEP_ROWS <число>	Определяет сколько записей должно храниться в истории

8.3.1 Создание резервной копии всей базы данных

Nbackup может работать с активной базой данных, не мешая подключенным к ней пользователям. Созданная резервная копия базы данных всегда будет отображать состояние базы данных на момент начала создания резервной копии.

Синтаксис nbackup для создания резервной копии всей базы данных:

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

Например:

```
nbackup -user testuser -password pass -role rdb$admin -b 0 base.fdb
base_10_04_17.nbk
```

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

Необязательный переключатель -role задаёт имя роли, права которой будут учитываться при выполнении каких-либо действий утилиты. Поэтому если пользователь не указывает роль, то он получает права только тех ролей, которые ему назначены с DEFAULT. Например, если в команде выше не указать роль пользователя (предположим роль rdb$admin не была назначена ему с DEFAULT), то выдастся сообщение об ошибке:

```
-ALTER DATABASE failed
-no permission for ALTER access to DATABASE
```

Уровень резервной копии 0 означает создание резервной копии всей базы данных. Уровни резервных копий больше 0 используются для создания инкрементных резервных копий.

Вместо имени файла базы данных можно указать псевдоним (alias, из файла databases.conf).

Вместо имени файла резервной копии можно указать stdout. Это перенаправит резервную копию в стандартный поток вывода, откуда Вы сможете перенаправить ее, например, на ленточный накопитель или на вход утилиты для сжатия получаемой резервной копии.

8.3.2 Восстановление из резервной копии всего файла базы данных

Резервная копия всей базы данных восстанавливается следующим образом:

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

Например:

```
nbackup -user sysdba -password masterkey -r base.fdb base_10_04_17.nbk
```

Уровень не указывается при восстановлении. Параметр `-R` должен быть указан последним.

Если указанная база данных уже существует и нет активных соединений, она будет перезаписана без предупреждения! Если есть активные соединения, восстановление не состоится, и Вы получите сообщение об ошибке.

8.3.3 Создание инкрементных резервных копий

Инкрементная резервная копия базы данных содержит только изменения с момента создания последней резервной копии. Существует два подхода к создания инкрементной резервной копии:

- По уровню резервной копии (больше 0);
- С помощью `GUID` последней созданной резервной копии базы данных.

Создание инкрементных резервных копий для многофайловых баз данных еще не поддерживается.

Инкрементная резервная копия уровня N содержит изменения базы данных с момента создания последней резервной копии уровня $N - 1$.

Примеры

Через день после создания резервной копии всей базы данных (уровня 0) создается резервная копия уровня 1:

```
nbackup -B 1 base.fdb base_11_04_17.nbk
```

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

```
nbackup -B 1 base.fdb base_12_04_17.nbk
```

Эта копия будет содержать изменения за последние два дня, то есть с момента создания резервной копии всей базы данных, а не только с момента создания предыдущей инкрементной копии уровня 1.

Далее, при создании резервной копии уровня 2 (допустим, в тот же день):

```
nbackup -B 2 base.fdb base_12_04_17_2.nbk
```

Эта резервная копия будет содержать изменения только с момента создания последней резервной копии уровня 1 (то есть за несколько часов).

Создать инкрементную резервную копию базы данных можно используя `GUID`. Такой способ не

требует знания уровня резервной копии для создания инкремента. Резервная копия будет содержать только изменения с момента создания последней резервной копии.

После создания резервной копии всей базы данных (уровня 0) можно создать инкрементную резервную копию. Для этого сначала необходимо узнать GUID последней резервной копии (Database backup GUID). Это можно сделать с помощью утилиты gstat с параметром -h.

Синтаксис для создания инкрементной резервной копии базы данных:

```
nbackup [-U <пользователь> -P <пароль> [-RO <роль>]] -B <GUID>
<база_данных> [<резервный_файл>]
```

Например:

После создания резервной копии всей базы данных (уровня 0) узнаём её GUID:

```
gstat -h base.fdb
```

В выводе будет GUID последней резервной копии (Database backup GUID):

```
Variable header data:
  Database backup GUID: {DEE5ECBE-3731-4FB4-4693-2D96439FD746}
  Database GUID: {6A03998A-33B5-2BCC-B8CB12CAA7ED}
```

Далее создаём инкрементную резервную копию:

```
nbackup -B {DEE5ECBE-3731-4FB4-4693-} base.fdb base_11_04_17.nbk
```

После этого GUID в выводе утилиты gstat изменится и можно будет создать инкрементную резервную копию, которая будет содержать изменения, внесённые с момента создания последней резервной копии.

8.3.4 Восстановление из инкрементных резервных копии

При восстановлении базы данных из инкрементных резервных копий необходимо обеспечить наличие полной цепочки инкрементных резервных копий, начиная с уровня 0 и до уровня, которым необходимо завершить восстановление.

Синтаксис:

```
nbackup [-U <пользователь> -P <пароль> -RO <роль>] -R <база_данных>
[<резервная_копия0> [<резервная_копия1> [...] ] ]
```

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

```
nbackup -R base.fdb base_10_04_17.nbk base_12_04_17.nbk base_11_04_17_2.nbk
```

Nbackup считает все аргументы после параметра именами файлов с резервными копиями. По этой причине никакие другие параметры (-U или -P) не могут следовать за списком файлов параметра -R.

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

8.3.5 Блокировка база данных и самостоятельное резервное копирование

Если Вы предпочитаете использовать какие-то другие утилиты для создания резервных копий базы данных или просто делать обычную копию базы данных как резервную, то для этого существует режим блокировки/разблокировки программы `nbackup`. «Блокировка» в данном случае означает, что основной файл базы данных временно замораживается, а не невозможность внесения изменений в базу данных. Как и в режиме резервирования, изменения фиксируются во временном файле дельты; при разблокировании файл дельты объединяется с основным файлом базы данных.

Самостоятельное резервное копирование обычно происходит по следующему сценарию:

1. Блокировка базы данных с помощью параметра `-L(OCK)`:

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

2. Создание резервной копии, сжатие файла базы данных, используя любые

другие программы или простое копирование файла с базой.

3. Разблокировка базы данных с помощью параметра `-UN(LOCK)`:

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

8.3.6 Восстановление из резервной копии,сделанной после блокировки

Копия блокированной базы данных является так же блокированной, поэтому Вы не сможете просто использовать копию как рабочую базу данных. В случае, если Ваша исходная база данных повреждена или утеряна, и нужно восстановить базу данных из копии, сделанной Вами самостоятельно, действуйте следующим образом:

1. Разархивируйте/скопируйте/восстановите файл базы данных с помощью

используемых Вами утилит.

1. Теперь разблокируйте базу данных, но не с параметром `-UN(LOCK)`, а с параметром `-F(IXUP)`:

```
nbackup -F <база_данных>
```

При использовании параметра `-UN` сначала определяется наличие любых изменений с момен- та блокирования базы данных (после использования параметра `-L`) и производится объединение временного файла дельты и основного файла базы данных. После этого база данных переводится в нормальный режим чтения/записи, а временный файл удаляется. При использовании параметра `-F` только изменяется в «нормальное» значение флага состояния самостоятельно восстановленной базы данных.

8.4 Утилита GFIX

Эта утилита является одним из основных инструментов администратора БД. Утилита `GFIX` позволяет:

- Выполнять принудительную сборку мусора (`sweep`);
- Изменять интервал автоматической сборки мусора;
- Закрывать базу данных для получения монопольного доступа, и затем снова открывать ее;
- Переводить базу в режимы «чтение/запись» или «только чтение»;
- Переключаться между синхронным и асинхронным вводом (Forced Writes);

- Изменять диалект БД;
- Устанавливать размер кэша;
- Изменять GUID базы данных;
- Искать зависшие транзакции и отменять или подтверждать их;
- Активировать или удалять теневые копии;
- Производить ремонт поврежденных баз данных.

Запуск GFIX осуществляется следующим образом:

```
gfix [<опции>] <имя базы данных>
```

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

Для того, чтобы выполнить любую операцию над базой данных с помощью gfix, необходимо пройти процедуру идентификации и аутентификации — указать имя и пароль пользователя, который имеет право на запуск сервиса gfix (подробнее см. *Распределение системных привилегий*). Для указания имени и пароля пользователя используются переключатели -user и -pa[ssword], соответственно, например:

```
gfix -user testuser -pas pass -role rdb$admin [опции] <имя базы данных>
```

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

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

Набор всех возможных опций утилиты GFIX, представлен ниже.

Таблица 8.11 — Опции GFIX

Опция	Описание
-ac(tivate_shadow) <теневая копия>	Параметр предназначен для активации теневой копии. <теневая копия> - адрес и имя файла теневой копии (или первого из файлов).
-at(tach) <n>	Дополнительный параметр к -shut. Предназначен для запрета новых соединений с БД. <n> указывает количество секунд, через которое произойдет отключение БД. Отключение отменится, если к этому времени еще останутся активные соединения.
-b(uffers) <n>	Устанавливает новый размер страничного кэша БД в страницах. <n> - количество страниц.
-co(mmit) {<ID> | all}	Завершает подтверждением зависшую транзакцию с идентификатором <ID>, или все зависшие транзакции (all).
-ca(che)	Параметр не используется.

(разрыв таблицы)

(разрыв таблицы)

Опция	Описание
-ch(ecksums) {on | off}	Включает/отключает вычисление контрольных сумм на уровне страниц базы данных. После включения контрольные суммы начнут пересчитываться при последующей записи страниц базы данных на диск. В случае несоответствия контрольных сумм в лог-файл будет записана ошибка, но работа продолжится.
-conf(ig_validate) <firebird.conf>	Производит валидацию конфигурационного файла firebird.conf. Во время валидации проверяется корректность указанных параметров и регулярных выражений, отсутствие дубликатов, права доступа к файлам и директориям, перечисленным в значениях параметров, в том числе с использованием wildcards.
-fu(ll)	Используется вместе с -validate для более глубокой проверки на уровне записей; освобождает неназначенные фрагменты записей.
-fo(rce_shutdown) <n>	Дополнительный параметр к -shut. Предназначен для принудительного закрытия базы данных. <n> указывает количество секунд, через которое произойдет закрытие. Если остались активные пользователи, они отключатся, последние результаты их работы будут потеряны. Такое средство нужно применять с осторожностью, как последнюю возможность.
-fe(tch_password)	Извлекает пароль из файла.
-g(uid)	Изменяет GUID базы данных. GUID генерируется случайно.
-h(ousekeeping) <n>	Изменяет интервал транзакций для автоматической чистки sweep. <n> устанавливает новый интервал. Если n = 0, автоматическая чистка запрещена.
-i(gnore)	Игнорировать ошибки контрольных сумм при проверке или чистке.
-icu	Исправляет базу данных для работы с текущей версией ICU.
-k(ill_shadow) <база данных>	Удаляет все неиспользуемые теневые копии базы данных.
-l(ist)	Показывает некоторые сведения (в том числе ID) о всех зависших транзакциях.
-me(nd)	Подготавливает повреждённую базу данных для резервного копирования. Помечает повреждённые записи как неиспользуемые.
-mo(de) {read_write | read_only}	Устанавливает режим записи для базы данных - только для чтения или чтение/запись. Этот параметр может принимать два значения.
-nol(inger)	Останавливает указанную базу данных, после того как последнего соединения не стало, независимо от установок LINGER в базе данных.
-n(o_update)	Используется вместе с -v(alidate) для проверки разрушенных или неразмещенных структур. Если таковые есть, они отобразятся в сообщении, но не будут исправлены.

(разрыв таблицы)

(разрыв таблицы)

Опция	Описание
-o(nline) {single|multi|normal}	Возвращает в режим "online" базу, остановленную с помощью -shut. Опция normal позволяет устанавливать соединение с БД любым авторизованным пользователям, а не только SYSDBA и владельцем БД. Опция multi позволяет устанавливать соединение с БД только SYSDBA и владельцем БД. Опция single похожа на multi, за тем исключением, что только один из пользователей - SYSDBA или владелец БД - может соединиться.
-pr(ompt)	Используется вместе с -l(ist). Выводит подсказки при восстановлении транзакций.
-pa(ssword) <пароль>	Пароль пользователя для работы с gfix.
-par(allel) <n>	Количество параллельных рабочих потоков для сборки мусора. Включает многопоточную сборку мусора, если MaxParallelWorkers > 1 (см. firebird.conf). Вместо этого параметра утилиты можно указать параметр конфигурации ParallelWorkers.
-replica {none|read_only|read_write}	Выбрать режим реплики - только для чтения (read_only) или для чтения и записи (read_write). Режим none отключает репликацию и переводит реплику в режим read_write. Как правило, этот режим должен использоваться после сбоя в базе данных мастера, когда эта реплика переходит в статус мастера. Или если нужно сделать физические резервные копии из реплики.
-role	Роль пользователя для работы с gfix.
-r(ollback) {<ID> | all}	Завершает откатом зависшую транзакцию с идентификатором <ID>, или все зависшие транзакции (all).
-sq(l_dialect) <n>	Изменяет диалект базы данных. <n> может быть 1 или 3.
-sw(eep)	Запускает сборку мусора в БД.
-sh(utdown) {single|multi|full}	Останавливает базу данных. Используется с одним из дополнительных параметров -attach, -force или -tran. Опция multi позволяет устанавливать соединение с БД только SYSDBA и владельцем БД. Опция single похожа на multi, за тем исключением, что только один из пользователей - SYSDBA или владелец БД - может соединиться. Опция full не позволяет соединиться с БД никому, даже SYSDBA и владельцу БД.
-tw(o_phase) {<ID> | all}	Автоматическое двухфазное восстановление limbo транзакции с номером <ID>, или всех транзакций (all).
-tra(nsaction) <n>	Дополнительный параметр к -shut. Предназначен для запрета запуска новых транзакций. <n> указывает количество секунд, через которое произойдет отключение БД. Отключение отменится, если к этому времени еще останутся активные транзакции.
-tru(sted)	Используется trusted авторизация.
-upgrade	Обновляет базу данных до последней минорной версии ODS в пределах поддерживаемой мажорной версии.

(разрыв таблицы)

(разрыв таблицы)

Опция	Описание
`-u(se) {full | reserve}`	Включает 100% заполнение страниц БД (`full`) или 80% заполнение по умолчанию (`reserve`). 100%-е заполнение имеет смысл для баз только для чтения.
`-user`	Имя пользователя для работы с `gfix`.
`-v(alidate)`	Проверяет базу данных на уровне страниц и освобождает страницы, помеченные как используемые, но никому не принадлежащие (`orphans`). Также выполняет проверку контрольных сумм базы данных. В случае их несоответствия в лог файл будет записана ошибки и, если не указана опция `no_update`, контрольная сумма будет исправлена.
`-w(rite) {sync|async}`	Переключает режимы синхронной/асинхронной записи `Forced Writes`.
`-z`	Выводит версию РЕД Базы Данных и утилиты `gfix`.

8.4.1 Активация теневой (оперативной) копии

Файл теневой копии является дополнительной копией первичного файла(ов) базы данных. Для любой данной базы данных может существовать несколько теневых копий, которые могут быть активированы и деактивированы по умолчанию с помощью утилиты `gfix`.

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

```
gfix -activate <файл теневой копии>
```

Это делает файл теневой копии новым файлом базы данных, и пользователи могут продолжать нормальную обработку данных и без потерь.

В случае, если ваш основной файл(ы) базы данных поврежден или нечитабелен, администратор базы данных может активировать теневой файл. После активации файл больше не является теневым файлом, и для его замены необходимо создать новый. Кроме того, теневой файл должен быть переименован на имя старого файла базы данных, который он заменяет.

Следует отметить, что активация теневой копии, в то время как сама база данных активна, может привести к повреждению тени. Перед ее активацией убедитесь, что файл базы данных действительно недоступен.

Администратор базы данных может настроить базу данных для автоматического создания нового теневого файла в случае активации текущей тени. Это позволяет обеспечить непрерывную поставку теневых файлов и предотвращает работу базы данных без нее.

8.4.2 Удаление теневых копий

Удаление всех недоступных теневых копий конкретной базы данных производится командой:

```
gfix -kill <база данных>
```

В случае, если база данных, работающая с теневыми файлами, теряет тень, или по какой-либо причине тень становится непригодной, база данных перестанет принимать новые соединения до тех

пор, пока администратор базы не уничтожит поврежденную тень и, в идеале, создаст новую тень для замены сломанной.

При соединении с локальной базой данных пользователь может не обладать правом на запуск сервиса `gfix`. Однако он сможет выполнить ту или иную операцию, только если у него есть соответствующие права. Например, для удаления теневой копии пользователь должен обладать правом на DDL-операцию `DROP SHADOW`— подробнее см. *Распределение прав на DDL-операции* .

8.4.3 Установка размера кэша базы данных

Кэш (или буфер) базы данных - это оперативная память, выделяемая сервером для работы с базой данных. Операции в оперативной памяти происходят гораздо быстрее, чем если данные постоянно считываются с диска. Размер кэша указывается в страницах БД. Если размер страницы установлен 8192, то кэш в 5000 страниц займет примерно 40 мегабайт ОЗУ. Если сразу много пользователей одновременно обращаются к базе данных, может случиться, что серверу не хватит выделенной оперативной памяти. В этом случае он начнет работать с диском, что замедлит производительность БД. Изменить размер кэша для базы данных можно командой:

```
gfix -b[uffers] <кол-во страниц> <база данных>
```

Это действие применяется только к указанной вами базе данных. На другие базы данных, работающие на том же сервере, это изменение не повлияет.

Другим способом установить размер кэша по умолчанию для всех вновь создаваемых БД, является изменение конфигурационного файла `firebird.conf`, а именно параметра `DefaultDbCachePages`.

Однако более предпочтительным способом для этих целей является утилита `gfix`, так как она позволяет установить собственный размер кэша для каждой базы данных. Если какой-то базой данных пользуются реже, размер кэша для нее можно оставить по умолчанию, или даже уменьшить.

8.4.4 Управление limbo транзакциями

«Застрявшей» («зависшей») транзакцией (limbo) называют транзакцию, которая работает одновременно с двумя или более базами данных. При завершении такой транзакции, РЕД База Данных совершает двухфазное подтверждение Commit, гарантируя, что изменения будут внесены либо во все БД, либо ни в одну. При этом подтверждения в базах данных будут даваться по очереди. Если в это время возникнет сбой системы, то может получиться, что в каких то БД изменения были сделаны, а в каких то нет. При этом транзакция переходит в неопределенное состояние, когда сервер не знает, следует ли подтвердить эту транзакцию, или откатить.

`Gfix` предоставляет несколько команд для управления транзакциями limbo.

Команда `gfix -l[ist]` будет отображать сведения о limbo транзакциях. Если команда не выдала результата, то зависших транзакций нет, и дальнейшие действия не требуются:

```
gfix -l[ist] <база данных>
```

Эту команду можно дополнить ключом `-pr[ompt]`, и тогда вам будет предложено выполнить `Commit` или `Rollback` для каждой обнаруженной транзакции limbo. В этом случае команда будет такой:

```
gfix -l[ist] -pr[ompt] <база данных>
```

Если обнаружено больше одной транзакции в состоянии `in-limbo`, то администратор может подтвердить (или откатить) все limbo транзакции или какую-то одну, указав идентификатор:

```
gfix -commit {all | <ID>} <база данных>
gfix -rollback {all | <ID>} <база данных>
```

После срабатывания этих команд следует заново запустить команду `-list`, чтобы убедиться, что limbo транзакций больше не осталось.

`Gfix` можно использовать для автоматического двухфазного восстановления limbo транзакции с идентификатором, или всех транзакций:

```
gfix -tw[o_phase] {all | <ID>} <база данных>
```

Эти три команды можно также использовать вместе с ключом `-pr[ompt]`.

Так как при запуске `gfix` можно указать только одну пару 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.

8.4.5 Установка режима доступа для базы данных

База данных может работать в одном из двух режимов доступа: только для чтения, или для чтения / записи (по умолчанию). Если вам понадобилось поменять режим, выполните команду:

```
gfix -mo {read_only | read_write} <база данных>
```

Например, если вы хотите разместить базу данных на CD диске, у вас не получится это сделать в режиме «чтения-записи». После того, как база данных будет заполнена данными, ее следует изменить в режим только для чтения, а затем использовать на CD диске (или других файловых системах только для чтения) без проблем.

8.4.6 Сборка мусора

Вследствие того, что СУБД «РЕД База Данных» имеет версионную архитектуру, со временем в ней могут накапливаться устаревшие версии записей, которые не нужны ни одной активной транзакции. В обычных условиях такие записи успешно удаляются «сборщиком мусора», но при возникновении ошибок в работе сервера «РЕД База Данных» (например, из-за аппаратного сбоя), в базе данных могут остаться зависшие транзакции или фрагменты записей, которые не могут быть удалены обычным «сборщиком мусора». Большое число таких «мусорных» записей может привести к значительному росту размера БД и падению производительности. Для удаления таких записей применяется процедура чистки (`sweep`). Чистка может производится в ручном и автоматическом режиме. В автоматическом режиме администратор задает интервал чистки — разницу между Oldest Transaction (или Oldest Interesting Transaction, OIT – находится в любом состоянии, кроме подтвержденного) и Oldest Snapshot Transaction (OST). По умолчанию этот интервал равен 20000 транзакций. Изменить этот параметр для конкретной базы данных можно с помощью опции `-h[ouskeeping]`:

```
gfix -h[ouskeeping] <интервал> <база данных>
```

Если интервал равен 0, то в этом случае автоматическая чистка будет отменена.

Вручную чистку можно произвести с помощью команды `-sweep`:

```
gfix -sweep [-i[gnore]] <база данных>
```

Ключ `-i[gnore]` заставляет РЕД Базу Данных игнорировать ошибки контрольной суммы на страницах базы данных. Лучше не использовать эту опцию, однако, если в вашей базе данных возникли некоторые проблемы, возможно, эта опция станет необходимой.

8.4.7 Закрытие (блокировка) базы данных

Закрытие базы данных означает, что база данных переводится в особый режим, в котором к ней запрещены некоторые подключения, в зависимости от указанного состояния. Полноценная работа с закрытой базой данных возможна только после обратного ее перевода в открытый режим.

Закрытие базы данных может быть необходимо для получения монопольного доступа к ней и проведения действий по восстановлению структуры базы и/или хранящихся в ней данных.

Для закрытия БД используется команда `-sh[ut]`. Совместно с этой командой указывается режим отключения существующих соединений и время ожидания (в секундах) до полной блокировки:

```
gfix -sh[ut] {full|single|multi} {-at[tach]|-tr[an]|-f[orce]} <таймаут> <база
данных>
```

Состояние `full` означает, что запрещены любые подключения, даже `SYSDBA` и владельца базы данных.

Состояние `multi` означает, что пользователи `SYSDBA` и владелец базы данных могут неограниченно подключаться к базе данных. Все остальные соединения запрещены. Это состояние используется по умолчанию при блокировке базы данных.

Состояние `single` означает, что пользователю `SYSDBA` или владельцу базы данных позволено одно подключение к базе данных. Все остальные соединения запрещены.

Режим `-at[tach]` означает, что все новые соединения к базе данных запрещены. Существующие соединения при этом не разрываются. Число `<таймаут>` определяет количество секунд, которое сервер будет ждать до завершения всех активных соединений к базе данных. Если после этого не останется ни одного активного соединения, то база данных будет закрыта. В противном случае, закрытие БД будет отменено.

При указании режима `-tra[nsaction]` сервер заблокирует запуск новых транзакций в закрываемой базе данных. Если по истечении `<таймаут>` секунд к базе не останется ни одного активного подключения, то база данных будет закрыта. В противном случае закрытие БД будет отменено.

В режим `-fo[rce_shutdown]` сервер будет ждать завершения всех активных соединений к базе данных. По истечении времени `<таймаут>` база будет закрыта вне зависимости от того, есть ли еще к ней активные соединения. В этом режиме возможна потеря данных, но он гарантирует, в отличие от двух предыдущих, что база будет переведена в закрытое состояние.

Перевести закрытую базу снова в открытое состояние можно только с помощью команды `-on[line]`

```
gfix -o[nline] {single|multi|normal} <база данных>
```

Состояние `normal` означает, что к базе данных могут подключаться любые авторизованные пользователи. Этот режим используется по умолчанию при включении базы данных.

8.4.8 Использование пространства страниц базы данных

> Изменять режим заполнения страниц с данными можно только, если база находится в режиме `read_write`.

Когда пишется страница базы данных, РЕД База Данных резервирует 20% страницы для буду-

щего использования.

Для использования всего доступного пространства страницы базы данных вы можете использовать команду `-use full`. Если впоследствии вы захотите вернуться к режиму по умолчанию, введите команду `-use reserve`, чтобы использовать только 80% каждой страницы.

```
gfix -use {full|reserve} <база данных>
```

8.4.9 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 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 эксклюзивный доступ к ней, как это описано в подразделе 8.4.7.

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

```
gfix -v[alidate] <база данных>
```

По умолчанию при проверке базы данных `gfix` освобождает неиспользуемые страницы в базе данных[9], а также обнаруживает разрушенные структуры данных.

По умолчанию проверка работает на уровне страниц. Для проверки и на уровне страниц и на уровне записей используйте команду:

```
gfix -v[alidate] -full <база данных>
```

которая освобождает неиспользуемые фрагменты записей.

Для того, чтобы `gfix` производил только проверку базы данных без освобождения неиспользуемых страниц, необходимо указать опцию `-n[o_update]`:

```
gfix -v[alidate] -n[o_update] <база данных>
```

Для того, чтобы `gfix` не проверял ошибки контрольных сумм, используется опция `-i[gnore]`[10] команды `-v[alidate]`:

```
gfix -v[alidate] -i[gnore] <база данных>
```

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

После проверки базы данных, если были найдены ошибки, можно попытаться исправить базу данных. Для этого существует опция `-mend`. Однако и она не в состоянии исправить все ошибки и может привести к потере данных (поврежденных записей).

```
gfix -mend <база данных>
```

[9] Страницы, которые были назначены какой-либо структуре данных, но не были использованы вследствие аппаратных или программных сбоев.

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

Лучший способ избежать потери данных - регулярно делать резервное копирование и проверять копии на возможность восстановления. При попытке исправить поврежденную базу данных всегда работайте с копией основного файла, а не с оригиналом. Использование опции `-mend` может привести к «бесшумному» удалению данных.

8.4.10 Изменение режима записи на диск

Многие ОС применяют кэширование операций ввода-вывода. При этом для буферизации используется некоторый объем быстрой памяти (который может быть как частью оперативной памяти сервера, так и специальной памятью, встроенной в сам диск). Это повышает производительность приложений, их интенсивность записи, но пользователь не будет уверен, когда его данные будут занесены на физический диск.

При работе с базой данных крайне желательно, чтобы данные были безопасно сохранены. В РЕД Базе Данных можно указать должны ли данные сразу записываться на физический диск по `commit` или же доверить запись ОС.

```
gfix -write {sync|async} <база данных>
```

По умолчанию, все БД работают с включенным Forced Writes (`sync`) и отключать этот режим не рекомендуется. Если все же вы не удовлетворены производительностью БД, и при этом стопроцентно уверены в своем серверном оборудовании, можете попробовать отключить Forced Writes.

8.4.11 Активация режима репликаци

Для активации режима репликации для базы данных используется команда:

```
gfix -replica { NONE | READ_ONLY | READ_WRITE } <база данных>
```

Здесь указывается GUID мастер-базы. GUID это уникальный идентификатор БД, используемый для её опознавания в системе репликации. Узнать его можно с помощью утилиты `gstat -h`. Он генерируется при создании базы (либо при первом коннекте, если еще не задан). Он будет автоматически пересоздан при восстановлении базы из бэкапа и при выполнении команды `nbackup -f` (выход из режима блокировки базы без влития изменений из дельта-файла). Также его можно сменить с помощью команды `gfix -g`, которая должна выполняться в эксклюзивном режиме с правами администратора. Это может быть полезно при копировании БД средствами файловой системы.

Удаляется `GUID` для слейв-базы в момент отключения режима реплики.

Снятие режима реплики производится с помощью команды:

```
gfix -replica {} <имя базы данных>
```

8.5 Утилита GSTAT

Утилита `gstat` предназначена для получения полной информации о базе данных. `Gstat` даёт информацию о дате создания базы данных, размере страниц базы данных, количестве теневых копий, таблицах и другую.

Для того, чтобы работать с `gstat`, необходимо пройти процедуру идентификации и аутентификации — указать имя и пароль пользователя, который имеет право на запуск сервиса `gstat` (подробнее см. *Распределение системных привилегий*). Для указания имени и пароля пользователя используются переключатели `-user` и `-password`, соответственно, например:

```
gstat -u testuser -p pass -role rdb$admin [<опции>] <база данных>
```

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

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

Запуск `GSTAT` осуществляется одним из следующих способов:

```
gstat [<опции>] <база_данных>
gstat <база_данных> [<опции>]
```

Таблица 8.12 — Опции `GSTAT`

Переключатель	Описание переключателя
`-a(ll)`	Это значение по умолчанию, если вы не запросили `-index, -data` или `-all`. Отыскивает и отображает статистику по `-index` и `-data`.
`-b(lob) TABLE.COLUMN [TABLE.COLUMN] ...`	За переключателем `-b` следует список имен таблиц и столбцов, где хранятся ссылки на файловые блобы. Список должен быть в верхнем регистре, и каждая `TABLE.COLUMN` разделяется пробелом. Выводится информация для каждого элемента `TABLE.COLUMN`: общее количество ссылок, количество отсутствующих файлов (ссылка есть, файла нет), общий размер файлов по этим ссылкам, некорректные ссылки (ссылки, для которых не удалось установить путь к файлу). Если указано несколько таблиц — выводится также суммарная статистика по всем указанным таблицам.
`-d(ata)`	Статистика страниц данных – информация о таблицах содержащихся в базе данных. Пользовательские и системные индексы, системные таблицы не анализируются.
`-e(ncryption)`	Статистика по зашифрованным и незашифрованным страницам БД.
`-h(eader)`	Статистика заголовочной страницы - это информация о глобальных свойствах всей базы данных, хранящаяся на заголовочной страницы каждой базы данных. Эта информация также выводится, если использовать любой другой переключатель `gstat`.
`-i(ndex)`	Статистика индексов – информация об индексах в базе данных. Пользовательские и системные таблицы, системные индексы не анализируются.
`-s(ystem)`	Эквивалент переключателям `-a[ll] -s[ystem]`. Это модификатор, который дополняет выводы `-data` или `-index` анализом системных таблиц (индексов).
`-u(sername)`	Имя пользователя.
`-p(assword)`	Пароль пользователя.
`-par(allel) <n>`	Количество параллельных рабочих потоков для сбора статистики. Включает многопоточность, если `MaxParallelWorkers > 1` (см. `firebird.conf`). Вместо этого параметра утилиты можно указать параметр конфигурации `ParallelWorkers`.

(разрыв таблицы)

(разрыв таблицы)

Переключатель	Описание переключателя
`-fetch <имя файла>|stdin|/dev/tty`	Файл, из которого будет считан пароль пользователя.
`-r(ecord)`	Модификатор для переключателей `-data` и `system`. Добавляет статистику о средних длинах записей, количестве версий и информацию о `BLOB`.
`-t(able) <ТАБЛИЦА> [<ТАБЛИЦА>] ...`	Анализ таблицы или списка таблиц и любых индексов, принадлежащих указанным таблицам, а также страниц данных этих таблиц. За этим параметром следует список имен таблиц, которые вы хотите проанализировать. Список должен быть в верхнем регистре, и каждая таблица разделяется пробелом. Если использовать опцию `-t` совместно с `-i`, то будет отображена статистика по индексам, но не будет информации о страницах данных. Если использовать опцию `-t` совместно с `-d`, то будет отображена статистика по страницам данных, но не будет информации об индексах.
`-ts [<ИМЯ ТАБЛИЧНОГО ПРОСТРАНСТВА> [<ИМЯ ТАБЛИЧНОГО ПРОСТРАНСТВА>...] | <PRIMARY>]`	Вывод информации об объектах, находящихся в табличных пространствах. При указании списка имён табличных простанств или `PRIMARY` (основной файл базы данных) будет отображена информация об объектах, находящихся в заданных табличных пространствах. Список должен быть в верхнем регистре, а имена табличных пространств должны разделяться пробелом. Если не указывать имена табличных пространств или `PRIMARY`, то будет отображена информация обо всех объектах, находящихся в пользовательских табличных пространствах.
`-role`	Роль.
`-tr`	Использовать доверительную аутентификацию.
`-z`	Показать версию сервера и утилиты.

8.5.1 Статистика заголовочной страницы

Запустив утилиту `gstat` с ключом `-header` можно получить статистические данные о нашей базе данных. Часть информации является статичной, часть – меняется в зависимости от происходящих в базе данных изменений. Пример информации с заголовочной страницы приводится ниже:

```
Database "d:\RedDatadase\testdb.fdb"
Database header page information:
      Flags                0
      Checksum             65417
      Generation           103
      System Change Number 3
      Page size            8192
      Server               RedDatabase
      ODS version          12.0
      Oldest transaction   91
      Oldest active        92
      Oldest snapshot      92
      Next transaction     92
      Autosweep gap        1
      Sequence number      0
      Next attachment ID   65
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
        Implementation       HW=AMD/Intel/x64 little-endian OS=WindowsCC=MSVC
        Shadow count         0
        Page buffers         0
        Next header page     0
        Database dialect     3
        Creation date        Sep 6, 2017 16:58:58
        Attributes           force write, checksums enabled
  Variable header data:
        Database backup GUID: {49E55285-939F-4960-94A3-3681659D2341}
        Database GUID: {F87421F8-92F4-49C0-3D8F-3B88FDA010C1}
        *END*
```

`Flags`:

Не используеся в заголовочной странице базы данных.

`Checksum`:

Контрольная сумма заголовочной страницы базы данных.

`Generation`:

Это счетчик, который увеличивается на единицу всякий раз, когда заголовочная страница записывается на диск.

`Page size`:

Размер страницы базы данных, исчисляется в байтах. Все файлы одной базы данных состоят из страниц одинакового размера, который устанавливается при создании базы данных и при восстановлении базы данных из резервной копии. Множество важнейших параметров сервера зависят от размера страницы - например, кэш базы данных. Рекомендуют создавать базу данных с размером страниц не менее 4096 байт, а лучше 8192 или 16384 байта.

`System Change Number`:

Маркер (число от 0 до 3), которым помечаются страницы базы данных при создании инкрементных копий с помощью утилиты `nbackup`. `Nbackup` выполняет резервное копирование страниц базы данных, копируя страницы, которые были изменены с момента последней резервной копии непосредственно предшествующего уровня. Если делается резервная копия уровня 0, копируются все страницы, а если уровня 1 – копируются только те страницы, которые были изменены после последнего уровня 0. Чтобы иметь возможность находить измененные страницы, РЕД База Данных использует маркер, который называется `SCN`. Это число увеличивается при каждом изменении состояния резервной копии и не зависит от уровня резервного копирования.

0 – страницы перед резервным копированием;

1 – страницы, записанные в дельта-файл во время резервного копирования;

2 – страницы, записанные во время слияния дельта-файла с основной резервной копией;

3 – страницы, записанные после окончания первого бэкапа и слияния.

Резервное копирование и восстановление с помощью `gbak` не восстанавливает содержимое таблицы `RDB$BACKUP_HISTORY` и сбрасывает `SCN` всех страниц обратно на 0. Таким образом, восстановление с помощью `gbak` перепишет всю базу данных (и может даже изменить размер страницы). Это делает предыдущие резервные копии с помощью nbackup бессмысленными в качестве отправной точки для последующих резервных копий: нужно начать заново с уровня 0.

`Server`:

Чьим сервером была создана база данных: `RedDatabase` или `Firebird`.

`ODS version`:

Версия структуры базы данных на диске (`On-Disk Structure`). Представляет собой два числа,

разделенные точкой. "Целая" часть - это основная версия ODS, которая зависит от версии сервера, создавшего данную базу данных. Главная версия определяет основные возможности работы с базой данных, и ее значение присваивается при создании (восстановлении) базы данных.

"Дробная" часть (после точки) - это минорная версия ODS, которая может меняться (точнее, увеличиваться) в течение жизни базы данных в зависимости от того, под управлением какой версии сервера работают с этой базой данных.

`Oldest transaction`:

Идентификатор старейшей заинтересованной транзакции в базе данных (`Oldest Interesting Transaction` или `OIT`). Первая транзакция с состоянием, отличным от `commit`. На практике это:

- `Oldest Active Transaction`;
- номер самой старой транзакции, завершенной "настоящим" `rollback` (выполняется если пользователь в одной транзакции модифицировал более 100000 записей);
- транзакция в состоянии `in-limbo` (полузавершенная) двухфазного `commit`.

Значение этого параметра часто сравнивается с `Next transaction`. Разница этих параметров показывает количество мусора в базе данных и можно судить о целесообразность выполнения резервного копирования.

`Oldest active`:

Старейшая транзакция, которую пользователь стартовал, и до сих пор не завершил по `commit` или `rollback` (`Oldest Active Transaction` или `OAT`).

`Oldest snapshot`:

Номер последней транзакции , которая влияет на процесс сборки мусора (`Oldest Snapshot Transaction` или `OST`). Дело в том, что только транзакции с уровнем изоляции `snapshot` вызывают появление мусорных версий записей в базе данных. `OST` – это старейший «номер snapshot» для всех активных транзакций. Транзакция `read-only`, `read committed` не имеет `«номера snapshot»`. Транзакция `read-write`, `read committed` имеет `«номер snapshot»` равный своему собственному номеру. Для транзакции `snapshot` `«номер snapshot»` – это номер `Oldest Active Transaction` на момент ее старта. Номер `Oldest Snapshot Transactio` обновляется, когда стартует новая транзакция (или выполняется `commit retaining`) или когда стартует `sweep`.

`Autosweep gap`:

Разница между `Oldest transaction` и `Oldest snapshot`. Показывает необходимость сборки мусора. Когда значение достигает `Sweep interval` (по умолчанию значение `Sweep interval` равно 20000) запускается автоматическая сборка мусора.Значение может быть отрицательным при долгоживущих транзакциях `snapshot`.

`Next transaction`:

Это просто номер, который будет присвоен следующей транзакции при ее старте.

Если вы заметили, что разница между `OIT` и `Next transaction` все увеличивается, значит какие-то транзакции не подтверждаются должным образом и следовательно может накапливаться все большее число мусорных записей. В конце концов вы увидите, что время запуска базы данных увеличивается, а производительность падает. В таком случае возможно следует запустить `gfix` для сборки мусора вручную.

`Sequence number`:

Порядковый номер файла базы данных. Для базы данных, состоящей из одного файла, он всегда равен нулю. Второй файл в базе данных будет иметь номер 1 и т. д.

`Next attachment ID`:

Номер следующего соединения к этой базе данных. Каждый раз, когда приложение подключается к базе данных, это число увеличивается на один. Запуск и завершение работы базы данных также увеличивает этот номер. Запуск `gstat` не изменяет идентификатор.

`Implementation`:

Архитектура аппаратуры, на которой была создана база данных.

`Shadow count`:

Число файлов оперативных копий, которые определены для данной базы данных.

`Page buffers`:

Размер кэша базы данныхв страницах. Ноль означает, что база данных использует значение по умолчанию сервера (`DefaultDbCachePages` в `firebird.conf`).

`Next header page`:

Номер следующей заголовочной страницы. Всегда равен нулю. Собственно говоря, любая страница в базе данных имеет ссылку на номер следующей страницы такого же типа, но так как заголовочная страница всегда единственная в каждом файле базы данных, то у нее эта ссылка обнулена.

`Database dialect`:

Диалект SQL базы данных.

`Creation date`:

Дата создания базы данных или последнего восстановления из резервной копии.

`Attributes`:

Это набор флагов, определяющий важные особенности поведения базы данных. Флаги устанавливаются только с помощью специальных инструментов, например, `gfix`. Значения флагов можно посмотреть в таблице 8.13.

`Variable header data`:

Переменные данные заголовочной страницы. Например, интервал очистки (`sweep interval`), `GUID` базы даных, `GUID` последней резервной копии и другое.

Таблица 8.13 — Атрибуты и флаги файла базы данных

Атрибут	Значение флага	Расшифровка
`active shadow`	`0x1 1`	Файл является активным Shadow-файлом.
`checksums enabled`	`0x8000 32768`	Включено вычисление контрольных сумм на уровне страниц базы данных.
`force write`	`0x2 2`	Режим принудительной записи включен (`forced write on`).
`crypt process, plugin <имя плагина>`	`0x4 4`	Идет процесс шифрования в фоновом режиме указанным плагином.
`no reserve`	`0x8 8`	Указывает, что на страницах не резервируется место для старых версий данных. Это позволяет более плотно упаковывать данные на каждой странице, в силу чего база данных занимает меньше дискового пространства. Это идеал для баз данных только для чтения.
`read only`	`0x20 32`	База данных работает в режиме `Read Only`. Если флаг не установлен, то допустимы как чтение, так и запись.
`encrypted, plugin <имя плагина>`	`0x40 64`	База данных зашифрована указанным плагином.
`replica`	`0x100 256`	База данных является объектом репликации (слейвом).
`multi-user maintenance`	`0x80 128`	База данных блокирована в режиме `multi`.
`single-user maintenance`	`0x1080 4224`	База данных блокирована в режиме `single`.

(разрыв таблицы)

(разрыв таблицы)

Атрибут	Значение флага	Расшифровка
`full shutdown`	`0x1000 4096`	База данных блокирована в режиме `full`.
`backup lock`	`0x400 1024`	Основной файл базы данных временно заблокирован. Изменения фиксируются во временном файле дельты.
`backup merge`	`0x800 2048`	Объединение файла дельты с основным файлом базы данных.

8.5.2 Статистика по табличным пространствам

Для вывода информации только об объектах, находящихся в табличных пространствах, выполните команду:

```
gstat <база данных> -ts
```

Данные о табличных пространствах в общем выводе отображаются так:

```
Tablespaces:
  PRIMARY (1)
    Full path: /home/test/employee.fdb
    Tables:
      COUNTRY
      JOB
    Indices:
      RDB$PRIMARY1 (COUNTRY)
      RDB$FOREIGN23 (CUSTOMER)
      RDB$PRIMARY22 (CUSTOMER)
  TS1 (2)
    Full path: /home/test/ts.ts
    Tables:
      EMPLOYEE
      PROJECT
      SALES
    Indices:
      CUSTNAMEX (CUSTOMER)
      MAXSALX (JOB)
      NEEDX (SALES)
      QTYX (SALES)
  TS2 (3)
    Full path: /home/test/ts2.ts
    Tables:
      CUSTOMER
    Indices:
      CUSTREGION (CUSTOMER)
  TS3 (4)
    Full path: /home/test/ts3.ts
    Tablespace is empty
```

8.5.3 Анализ всей базы данных

Если не заданы никакие опции утилиты `gstat` по умолчанию выполняется анализ всей базы данных. Будет собрана информация о таблицах и индексах, содержащихся в базе данных. Поскольку вывод, скорее всего, будет очень большим, рекомендуется передать вывод в файл:

```
gstat d:\RedDataBases\testdb.fdb > d:\RedDataBases\testdb.txt
```

8.5.4 Статистика страниц данных

Следующая команда

```
gstat -data <база данных>
```

просматривает в базе данных таблицу за таблицей, отображая итоговую информацию о страницах данных. Для включения в отчет системных таблиц добавьте переключатель `-system`.

При указании опции `-d` совместно с `-t` будет сформирована статистика по страницам данных только для указанной таблицы. Вы можете записать разделенный пробелам и список имен таблиц для получения результатов более чем по одной таблице.

Вывод о каждой таблице будет примерно следующий:

```
TEST_TABLE (338)
     Primary pointer page: 734, Index root page: 735
     Pointer pages: 7, data page slots: 20448
     Data pages: 20448, average fill: 85%
     Primary pages: 13998, secondary pages: 6450, swept pages: 13998
     Empty pages: 0, full pages: 20446
     Big record pages: 8
     Blobs: 463, total length: 248371310, blob pages: 15410
         Level 0: 463, Level 1: 0, Level 2: 0
     Fill distribution:
          0 - 19% = 0
         20 - 39% = 0
         40 - 59% = 0
         60 - 79% = 0
         80 - 99% = 0
```

Таблица 8.14 — Вывод `gstat -d[ata]`

Элемент	Описание
`Tablespace`	Табличное пространство, в котором находится объект. Если указано `PRIMARY`, то объект расположен в основном файле базы данных.
`Primary pointer page`	Номер первой страницы косвенных указателей на страницы, хранящие данные таблицы
`Index root page`	Номер страницы, которая является первой страницей указателей на индексы таблицы
`Pointer pages`	Общее количество страниц косвенных указателей на страницы, хранящие данные таблицы

(разрыв таблицы)

(разрыв таблицы)

Элемент	Описание
`data page slots`	Количество указателей на страницы базы данных, содержащихся на страницах указателей. Должно равняться числу страниц данных
`Data pages`	Общее количество страниц, в которых хранятся данные таблицы. Этот счетчик включает страницы, хранящие неподтвержденные версии записей и мусор, потому что `gstat` не может их отличить друг от друга
`Average fill`	Обобщающая гистограмма распределения использования памяти для всех страниц, выделенных в таблице.
`Primary pages`	Количество страниц, равное (`Data pages - Secondary pages`)
`Secondary pages`	Количество страниц, на которых не хранятся первичные версии записей.
`Swept pages`	Количество страниц, которые имеют только первичные версии записей, и все они были созданы подтвержденными транзакциями. Такие страницы данных должны быть пропущены процедурой `sweep`.
`Empty pages`	Количество страниц, на которых нет записей
`Full pages`	Количество полностью заполненных страниц
`Fill distribution`	Это гистограмма из пяти 20-процентных "полос", каждая из которых показывает количество страниц данных, чье среднее заполнение попадает в этот диапазон. Процент заполнения определяется соотношением пространства каждой страницы, содержащей данные. Сумма этих чисел дает общее количество страниц, содержащих данные.

8.5.5 Анализ индексов

Следующая команда

```
gstat -i[ndex] <база данных>
```

отыскивает и отображает статистику по индексам в базе данных: средняя длина ключа (в байтах), общее количество дубликатов и максимальное количество дубликатов одного ключа и другое. Вы можете добавить переключатель `-system`, чтобы включить сведения о системных индексах в отчет.

К сожалению, не существует способа получить статистику по одному индексу. Однако вы можете ограничить результат одной таблицей, используя переключатель `-t`, за которым следует имя таблицы. Вы можете записать разделенный пробелам и список имен таблиц для получения результатов более чем по одной таблице.

Данные индексной страницы отображаются так:

```
TEST_TABLE (128)
     Index IND1 (0)
         Root page: 756595, depth: 4, leaf buckets: 232528, nodes: 50422773
         Average node length: 11.90, total dup: 0, max dup: 0
         Average key length: 8.11, compression ratio: 1.11
         Average prefix length: 3.89, average data length: 5.11
         Clustering factor: 6485823, ratio: 0.13
         Fill distribution:
             0 - 19% = 1
             20 - 39% = 0
             40 - 59% = 0
             60 - 79% = 0
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
                80 - 99% = 232527
```

Таблица 8.15 — Вывод `gstat -i[ndex]`

Элемент	Описание
`Tablespace`	Табличное пространство, в котором находится объект. Если указано `PRIMARY`, то объект расположен в основном файле базы данных.
`Root page`	Номер корневой страницы индекса
`Depth`	Количество уровней в странице индексного дерева. Если глубина дерева индексной страницы превышает 3, то доступ к записям через индекс не будет максимально эффективным. Для уменьшения глубины дерева индексной страницы увеличьте размер страницы. Если увеличение размера страницы не уменьшает глубины, снова увеличьте размер страницы.
`Leaf buckets`	Количество страниц самого низкого уровня (листовых) в дереве индекса. Это страницы, которые содержат указатели на записи. Страницы высокого уровня содержат косвенные связи.
`Nodes`	Общее количество записей, индексированных в дереве. Должно быть равно количеству индексированных строк в дереве, хотя отчет `gstat` может включать узлы, которые были удалены, но не вычищены в процессе сборки мусора. Может также включать множество элементов для записей, у которых был изменен индексный ключ.
`Average node length`	Средний размер узлов в байтах
`Total dup`	Общее количество строк дубликатов индекса
`Max dup`	Количество дублирующих узлов в "цепочке", имеющих наибольшее количество дубликатов. Всегда будет нулем для уникальных индексов. Если число велико по сравнению с числом в `Total dup`, то это признак плохой селективности.
`Average key length`	Средний размер ключа в байтах с учетом сжатия. К длине каждого ключа прибавляется от 1 до 5 байт в зависимости от размера ключа и префикса. Затем высчитывается средний размер ключа.
`Compression ratio`	Отношение средней длины ключа без учета сжатия (`Average prefix length + Average data length`) к средней длине ключа с учетом сжатия (`Average key length`).
`Average prefix length`	Средний размер, занимаемый префиксами узлов, в байтах.
`Average data length`	Средняя длина каждого ключа в байтах. Она, скорее всего, меньше, чем фактическая сумма размеров столбцов, поскольку РЕД База Данных использует индексное сжатие для уменьшения объема данных, хранящихся на странице листа индекса.

(разрыв таблицы)

(разрыв таблицы)

Элемент	Описание
`Clustering factor`	Это мера того, насколько много операций ввода-вывода будет осуществлять база данных, если бы ей пришлось читать каждую строку таблицы посредством индекса, в порядке индекса. То есть она показывает, насколько упорядочены строки в таблице по значениям индекса. Если значение близко к общему количеству страниц, значит таблица очень хорошо упорядочена. В этом случае записи индекса на одной странице листа индекса обычно указывают на строки, находящиеся в одних и тех же страницах данных. Если значение близко к общему количеству строк, значит, таблица весьма неупорядочена. В этом случае маловероятно, что записи индекса на одной странице листа индекса указывают на те же страницы данных.
`Ratio`	Отношение `Clustering factor` к общему количеству узлов в индексе.
`Fill distribution`	Это гистограмма с пятью 20-процентными полосами, каждая из которых показывает количество индексных страниц, чей средний процент заполнения находится в указанном диапазоне. Процент заполнения определяется соотношением пространства каждой страницы, содержащей данные. Сумма таких чисел дает общее количество страниц, содержащих индексные данные

8.5.6 Статистика по размерам и версиям записей

Следующая команда:

```
gstat -r[ecord] [-d] <база данных>
```

отображает статистику по размерам и версиям записей.

Таблица 8.16 — Вывод `gstat -r[ecord]`

Элемент	Описание
`Total formats`	Общее количество форматов в таблице `RDB$FORMATS`
`Used formats`	Количество используемых форматов
`Average record length`	Средний размер сжатой записи в байтах
`Total records`	Общее количество строк в таблице.
`Average version length`	Среднее значение длины старых версий в байтах
`Total versions`	Общее количество старых версий в таблице
`Max versions`	Максимальная цепочка старых версий для записи
`Average fragment length`	Средний размер фрагмента в байтах
`Total fragments`	Количество всех фрагментов во всех записях
`Max fragments`	Максимальное количество фрагментов в одной записи
`Average unpacked length`	Средний размер записи в байтах (без сжатия)
`Compression ratio`	Отношение несжатых данных к сжатым
`Big record pages`	Количество страниц, которые полностью заняты только одной записью
`Blobs`	Количество всех блобов (0, 1 и 2 уровня)
`Total length`	Размер, занимаемый блобами, в байтах

(разрыв таблицы)

(разрыв таблицы)

Элемент	Описание
Blob pages	Количество страниц с блобами
Level <n>	Количество блобов каждого уровня

8.5.7 Статистика по файловым блобам

В утилите gstat можно воспользоваться опцией, в которой указываются таблицы, где хранятся ссылки на файловые блобы и которая будет подсчитывать для каждой указанной таблицы количество и общий размер файлов по ссылкам, количество ненайденных файлов (ссылка есть, файла нет), некорректные ссылки (не удалось установить путь к файлу). Null, пустые строки, а также строки, состоящие только из пробелов, ссылками не считаются и не увеличивают счётчик ссылок.

```
gstat <база данных> -b <таблица>.<столбец> [<таблица>.<столбец>] ...
```

Если указано несколько столбцов одной таблицы, то при выводе произойдёт группировка по таблице. Если передано больше одного поля, то выведется суммарная статистика.

Файловые блобы хранятся в каталогах, настраиваемых в файле directories.conf. Они управляются тремя системными функциями: CREATE_FILE, READ_FILE, DELETE_FILE.

CREATE_FILE создаёт файл из указанного блоба и возвращает ссылку на него. Эта ссылка может быть сохранена в таблице в обычном текстовом поле. READ_FILE может по этой ссылке прочитать файл и вернуть его содержимое в виде блоба.

Опция -b при вызове gstat указывается после пути к базе данных. Опцию -b можно совмещать со всеми параметрами, кроме -e и -b.

```
gstat d:/work/test.fdb -b BAT.VCHAR BAT.BL LINKS.B1 LINKS.B2 LINKS.B3
================================================================================
Database file sequence:
File D:\WORK\TEST.FDB is the only file

Analyzing file blobs ...
BAT (130)
  'VCHAR' field:
    Links'count: 2
    Missing files'count: 0
    Unresolved links'count: 0
    Blob files'size: 26 bytes

  'BL' field:
    Links'count: 2
    Missing files'count: 0
    Unresolved links'count: 0
    Blob files'size: 22 bytes

LINKS (128)
  'B1' field:
    Links'count: 2
    Missing files'count: 0
    Unresolved links'count: 1
    Blob files'size: 7 bytes
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
    'B2' field:
      Links'count: 2
      Missing files'count: 1
      Unresolved links'count: 1
      Blob files'size: 0 bytes

    'B3' field:
      Links'count: 2
      Missing files'count: 0
      Unresolved links'count: 1
      Blob files'size: 13 bytes

  Total links'count: 10
  Total missing files'count: 1
  Total unresolved links'count: 3
  Total blob files'size: 68 bytes
```

8.6 Утилита GSEC

GSEC — это утилита для работы с базой данных безопасности (содержащей информацию о пользователях СУБД). Она позволяет привилегированному пользователю управлять учетными записями пользователей для различных баз данных. Используя различные опции, можно добавлять, изменять или удалять учетные записи пользователей из базы данных безопасности.

Утилита GSEC устарела. Вместо неё лучше использовать SQl-команды для управления пользователями или Services API.

Информация о всех пользователях хранится в обычной базе данных, называемой базой данных безопасности. По умолчанию база данных безопасности располагается в директории «РЕД Базы Данных» и называется security_version (см. app:securitydb).

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

Синтаксис GSEC:

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

Таблица 8.17 — Опции GSEC

Опция	Описание
-user <имя пользователя>	Имя пользователя
-password <пароль>	Пароль пользователя
-fetch_password <файл>	Файл, из которого будет считан пароль пользователя
-role <имя роли>	Название роли, чьи права будет использовать указанный пользователь
-trusted	Использовать доверительную аутентификацию
-database <БД_безопасности>	Путь к файлу базы данных безопасности

(разрыв таблицы)

(разрыв таблицы)

Опция	Описание
-z	Отобразить версию GSEC и сервера РЕД Базы Данных.

Таблица 8.18 — Команды GSEC[11]

Команда	Описание
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
mapping {set | drop}	Включает или выключает флаг AUTO ADMIN MAPPING для автоматического предоставления роли RDB$ADMIN администраторам Windows в текущей базе данных, если используется доверительная авторизация
{? | help}	Отображает все опции и команды GSEC
z	Отобразить версию GSEC и сервера РЕД Базы Данных
quit	Выход из интерактивного режима.

Таблица 8.19 — Параметры GSEC

Команда	Описание
-pw <пароль>	Пароль пользователя
-uid <uid>	Идентификатор пользователя
-gid <uid>	Идентификатор группы
-fname <имя>	Полное имя пользователя
-mname <отчество>	Отчество
-lname <фамилия>	Фамилия
-admin {yes | no}	Установка флага, имеет ли пользователь роль RDB$ADMIN

Для того, чтобы выполнить любую операцию с помощью gsec, необходимо пройти процедуру идентификации и аутентификации — указать имя и пароль пользователя, который имеет право на запуск сервиса gsec (подробнее см. *Распределение системных привилегий*). Для указания имени и пароля пользователя используются переключатели -user и -pa[ssword], соответственно, например:

```
gsec -user testuser -password pass -role rdb$admin <команда> [<параметры>...]
```

Необязательный переключатель -role задаёт имя роли, права которой будут учитываться при выполнении каких-либо действий утилиты. Поэтому если пользователь не указывает роль, то он получает права только тех ролей, которые ему назначены с DEFAULT. Права на внесение изменений в базу данных безопасности имеют пользователи с административными привилегиями. При этом роль RDB$ADMIN должна быть указана в переключателе -role. Остальные пользователи имеют права только на изменение своей учетной записи.

[11] В не интерактивном режиме команды вводятся с префиксом «-», например -add.

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

Пример запуска утилиты GSEC в интерактивном режиме:

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

Пример вывода информации о пользователях:

```
GSEC> display
user name uid gid full name
------------------------------------------------------------
SYSDBA 0 0 Sql Server Administrator
TEST_USER 0 0 John Smith
```

GSEC можно использовать для управления базой данных безопасности на удаленном сервере. Для этого необходимо указать в командной строке имя:

```
gsec -database 192.168.10.1:C:\RedDatabase\security5.fdb -user sysdba -password
masterkey
```

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

```
GSEC> quit
```

8.7 Утилита rdb_lock_print

Утилита rdb_lock_print является инструментом, формирующим статистические данные файла блокировок, что помогает при решении сложных проблем взаимных блокировок.

Исполняемый модуль rdb_lock_print находится в каталоге bin установки СУБД РЕД База Данных. Синтаксис вызова утилиты может выглядеть одним из следующих способов:

```
rdb_lock_print -d <database_name> [<опции>...]
rdb_lock_print -f <lock_file> [<опции>...]
```

Где параметр <database_name> задает имя и путь к БД. Параметр lock_file задает имя и путь к lock - файлу. Путь к файлу таблицы блокировок проходит через /tmp/firebird в ОС Linux и C:/ProgramData/firebird в ОС Windows.

С помощью утилиты rdb_lock_print может быть выведена таблица блокировок в удобном пользователю формате. Отчет вывода разбит на блоки в следующей последовательности:

1. LOCK_HEADER BLOCK: заголовок блока;
2. OWNER BLOCK: группы владельцев;
3. REQUEST BLOCK: группы запросов. Каждый владелец выводится с его запросами;
4. LOCK BLOCK: группы блокировок;
5. History: история событий.

Утилита rdb_lock_print имеет несколько опций, приведенных в таблице 8.20. Если никаких опций не задано, то выводится заголовок блока LOCK_HEADER BLOCK, в котором описано основная конфигурация и состояние таблицы блокировок.

Таблица 8.20 — Опции `rdb_lock_print`

Опция	Описание
`-a`	Выводит содержимое таблицы блокировок: заголовок блока (`LOCK_HEADER BLOCK`), группы блоков (`LOCK BLOCK`), группы владельцев (`OWNER BLOCK`), группы запросов (`REQUEST BLOCK`) и историю событий (`History`).
`-c`	Запрашивает мьютекс (mutex) менеджера блокировок распечатать цельное последовательное представление таблицы блокировок. При этом останавливаются все процессы доступа к базе данных.
`-h`	Выводит недавнюю историю событий
`-i [a,o,t,w] [<N> [<M>]]`	Выдает интерактивный отчет, отслеживающий текущую деятельность таблицы блокировок. Выводит выбранные параметры менеджера блокировок в течение n секунд с интервалом m секунд. Значения по умолчанию 1 сек для обоих значений. Если указано только `-i` - выводится вся информация.
`-l`	Выводит группы блоков
`-m`	Вывод в HTML формате
`-n`	Выводит только ожидающих завершения владельцев (если указано `-o`) или ожидающих завершения групп блоков (если указано `-l`)
`-o | -p`	Выводит группы владельцев
`-q <N>`	Находит владельцев взаимных блокировок и печатает их PID (`N` - интервал сканирования)
`-r`	Выводит информацию о группе блока запросов
`-s <номер>`	Выводит группы блокировок ресурсов заданной серии (действует только если указан спецификатор `-l`)
`-w`	Выводит временный список блокировок (групп запросов), блокирующих другие запросы на блокировку, для каждого владельца (действует только с указанием `-o`)

Результаты использования утилиты могут оказаться достаточно большими, поэтому удобнее отчет выводить в файл, указав: `> <путь к файлу>`.

8.7.1 Блок LOCK_HEADER

Первая группа всех отчетов `rdb_lock_print`. Каждый отчет выводит только одну группу заголовка. На рисунке 8.1 представлено, как может выглядеть блок `LOCK_HEADER`.

`Version:`
Номер версии менеджера блокировок.

`Active owner:`
Владелец, который управляет таблицей блокировок в настоящий момент. Если в таблицу блокировок не пишет ни один процесс, то активным владельцем будет 0.

`Length:`
Общий объем памяти, выделенный таблице блокировок (в байтах).

`Used:`
Наибольшая величина смещения в таблице блокировок, которая используется в настоящий момент.

```
LOCK_HEADER BLOCK
    Version: 145, Active owner:      0, Length: 1048576, Used:  23128
    Flags: 0x0001
    Enqs:     18, Converts:     40, Rejects:      0, Blocks:      0
    Deadlock scans:      0, Deadlocks:      0, Scan interval:  10
    Acquires:    109, Acquire blocks:      0, Spin count:   0
    Mutex wait: 0.0%
    Hash slots: 1009, Hash lengths (min/avg/max):    0/   0/   3
    Remove node:      0, Insert queue:      0, Insert prior:      0
    Owners (1): forward:  20840, backward:  20840
    Free owners: *empty*
    Free locks (2): forward:  22808, backward:  23064
    Free requests (2):  forward:  22744, backward:  23000
    Lock Ordering: Enabled
```

Рисунок 8.1 — Пример группы заголовка блока

`Flags:`
Определены два битовых флага:

- `LHB_shut_manager` — показывает, что БД остановлена;
- `LHB_lock_ordering` — блокировки назначаются в порядке запросов FIFO.

`Enqs:`
Число запросов, полученных на блокировку (не включает запросы, которые пришли и ушли).

`Converts:`
Запросы на повышение уровня блокировки.

`Rejects:`
Запросы, которые не могут быть удовлетворены.

`Blocks:`
Запросы, которые не могут быть удовлетворены немедленно.

`Deadlock scans:`
Показывает число просмотров менеджером блокировок цепочки блокировок и владельцев для поиска взаимных блокировок. Менеджер приступает к просмотру, когда процесс ожидает блокировки в течение `Scan interval` секунд.

`Deadlocks:`
Число найденных взаимных блокировок.

`Scan interval:`
Время (в секундах), которое ожидает менеджер блокировок до того как запустить к поиску взаимных блокировок.

`Acquires:`
Сколько раз владелец запрашивает исключительное управление таблицей блокировок, чтобы выполнить изменения.

`Acquire blocks:`
Сколько раз владелец находился в состоянии ожидания при запросе исключительного управления таблицей блокировок.

`Spin count:`
Режим ожидания взаимной блокировки, когда повторяется запрос к таблице блокировок. По умолчанию отключено(равно 0).

`Mutex wait:`
Процент попыток, которые были заблокированы, когда владелец старался обратиться к таблице блокировок. Сколько раз (в процентах) владелец находился в состоянии ожидания, когда пытался

обратиться к таблице блокировок.

`Hash slots:`
Число слотов кэширования блокировок.

`Hash lengths:`
Длина цепочки кэширования.

`Remove node:`
Владелец сообщает о намерении удалить узел из таблицы, когда зависает при запросе к таблице блокировок.

`Insert queue:`
Владелец сообщает о намерении добавить узел в таблицу, когда зависает при запросе к таблице блокировок.

`Insert prior:`
Указывает, где размещается ошибочное добавление узла.

`Owners:`
Количество владельцев, соединенных с таблицей блокировок.

`Free owners:`
Количество блоков владельцев, выделенных владельцам, которые завершили их соединения, оставив блоки неиспользованными.

`Free locks:`
Количество групп блокировок, которые были освобождены и не использованы повторно.

`Free requests:`
Количество групп запросов, которые были освобождены и не использованы повторно.

`Lock ordering:`
Определяет порядок получение запросов на блокировку (в порядке их поступления). Включено, если установлен флаг `LHB_lock_ordering`.

`Creation timestamp:`
Время создания.

`Hash lengths distribution:`
Распределение длин хэшей.

`Pending request count:`
Количество ожидающих запросов.

8.7.2 Блок OWNER

Идентифицирует конкретного владельца. Владельцы делятся на несколько типов, которым сопоставляются числа: 1 – процесс, 2 – база данных, 3 – клиентское соединение, 4 – транзакция, 255 – фиктивный процесс.

Заголовок блока содержит число, которое используется в качестве идентификатора владельца в таблице блокировок.

```
OWNER BLOCK  20840
    Owner id: 26834955665412, type: 1, pending:      0
    Process id:   6248 (Alive), thread id:   8928
    Flags: 0x00
    Requests (15):  forward:  20952, backward:  22872
    Blocks: *empty*
```

Рисунок 8.2 — Пример группы владельцев

`Owner id::`
Идентификатор владельца.

`Type:`
Тип владельца.

`Pending:`
Ожидание завершения блокировки, запрошенную владельцем, но не полученную. Показывает смещение группы запроса блокировки.

`Process id:`
Идентификатор процесса.

`Thread id:`
Идентификатор потока.

`Flags:`
Биты, которые определяют состояние. Процесс в одно и то же время может находиться более чем в одном состоянии.

`Requests:`
Запросы на блокировку (обработанные или ожидающие завершения) связанные с этим процессом.

`Forward:`
Ссылается на последующий элемент в очереди запросов, принадлежащих этому процессу. Число задает смещение.

`Backward:`
Ссылается на предыдущий элемент в очереди запросов, принадлежащих этому процессу. Число задает смещение.

`Blocks:`
Временный список блокировок (групп запросов), блокирующих другие запросы на блокировку, которыми владеет этот процесс.

8.7.3 Блок REQUEST

Выводит все запросы владельца. На рисунке 8.3 показано как выглядит блок `REQUEST BLOCK`.

```
REQUEST BLOCK  23000
    Owner:  20840, Lock:  23064, State: 6, Mode: 6, Flags: 0x00
    AST: 0x0000000000000000, argument: 0x000000000483A2F0
    lrq_own_requests: *empty*
    lrq_lbl_requests:   forward:     24, backward:  22744
    lrq_own_blocks  : *empty*
```

Рисунок 8.3 — Пример блока конкретного запроса

`Owner:`
Смещение группы владельца в таблице блокировок, выполнившего запрос.

`Lock:`
Смещение группы блокировки, которая описывает блокируемый ресурс.

`State:`
Состояние блокировки, которое назначено этому ресурсу.

`Mode:`
Режим блокировки – состояние, которое было запрошено для блокировки.

`Flags:`
Флаг запроса содержит биты: 1 – блокирование, 2 – ожидание завершения, 4 – конвертирование,

8 – отмена. Они могут комбинироваться.

`AST:`
Адрес подпрограммы, которая вызывается, когда кто-либо еще хочет получить блокировку на ресурс, используемый настоящим запросом.

`Argument:`
Адрес аргумента, который может понадобиться подпрограмме AST.

8.7.4 Блок LOCK

Группы блокировок представляют блокируемые ресурсы различных типов, соответствующих определенным сериям:

Таблица 8.21 — Типы ресурсов

Серия	Символ	Тип ресурса
1	LCK_database	Сама база данных
2	LCK_relation	Отношение
3	LCK_bdb	Страница базы данных
4	LCK_tra	Транзакция
5	LCK_rel_exist	Существование отношения
6	LCK_idx_exist	Существование индекса
7	LCK_attachment	Подключение
8	LCK_shadow	Теневая копия
9	LCK_sweep	Сборка мусора
10	LCK_expression	Механизм кэширования индекса по выражению
11	LCK_prc_exist	Существование процедуры
12	LCK_update_shadow	Синхронное изменение теневой копии
13	LCK_backup_alloc	Страница размещения таблицы в резервном файле
14	LCK_backup_database	Защита записи в файл БД
15	LCK_backup_end	Защита согласованного удаления резервного файла
16	LCK_rel_partners	Связанные таблицы
17	LCK_page_space	Идентификатор страничного пространства
18	LCK_dsql_cache	DSQL кэш
19	LCK_monitor_state	Сбрасывание данных мониторинга
20	LCK_tt_exist	Существование TextType
21	LCK_cancel	Отмена операции
22	LCK_btr_dont_gc	Предотвращение удаления страницы B-дерева из индекса
23	LCK_rel_gc	Разрешение сборки мусора для отношения
24	LCK_tpc_init	Инициализация TPC
25	LCK_tpc_block	Существование файла tpc-блока памяти
26	LCK_fun_exist	Существование функции
27	LCK_rel_rescan	Принудительное повторное сканирование отношения

(разрыв таблицы)

(разрыв таблицы)

Серия	Символ	Тип ресурса
28	LCK_crypt	Однопоточное шифрование
29	LCK_crypt_status	Сообщает об изменении статуса шифрования базы данных
30	LCK_record_gc	Сборка мусора на уровне записи
31	LCK_alter_database	ALTER DATABASE
32	LCK_repl_state	Состояние репликации
33	LCK_repl_tables	Реплицированный набор
34	LCK_dsql_statement_cache	Кэш DSQL операторов
35	LCK_profiler_listener	Слушатель сетевого профайлера
36	LCK_tablespace_exist	Существование табличного пространства
37	LCK_repl_txn	Транзакция репликации

На рисунке 8.4 представлен вывод группы блокировки ресурса серии 8. Блок LOCK_BLOCK идентифицирует группу описания заблокированного ресурса.

```
LOCK BLOCK  22552
    Series: 20, Parent:  21016, State: 2, size: 8 length: 4 data: 0
    Key: 000000, Flags: 0x00, Pending request count:      0
    Hash que (3):   forward:    380, backward:  22424
    Requests (1):   forward:  22488, backward:  22488
        Request  22488, Owner:  20840, State: 2 (2), Flags: 0x00
```

Рисунок 8.4 — Пример группы блокировок

Series:
Тип ресурса.

Parent:
Родитель для всех блокировок, связанных с конкретным типом ресурса.

State:
Наивысшее текущее состояние блокировки.

Size:
Длина части группы блокировки, содержащей ключ (в байтах).

Length:
Фактическая длина ключа.

Data:
Данные. Являются целым числом.

Key:
Идентификатор заблокированного ресурса.

Hash queue:
Начало и конец очереди хэш для ключей ресурсов.

Requests:
Показывает число запросов на блокировку этого ресурса и указатели вперед и назад на группы блокировок.

Request:
Список запросов. Показывает идентификатор запрашиваемой группы, процесс, выполняющий запрос и фактическое состояние блокировки с запрашиваемым (в круглых скобках).

`Flags:`
Флаг запроса содержит биты: 1 – блокирование, 2 – ожидание завершения, 4 – конвертирование, 8 – отмена. Они могут комбинироваться.

8.7.5 Блок History

Менеджер блокировок запоминает действия, которые он выполнял для каждого владельца. Эти действия можно просмотреть в блоке `History` и блоке `Event log`.

```
History:
    ENQ:     owner =  20840, lock =      0, request =  20952
    GRANT:   owner =  20840, lock =  21016, request =  20952
    ENQ:     owner =  20840, lock =      0, request =  21088
    GRANT:   owner =  20840, lock =  21152, request =  21088
    ENQ:     owner =  20840, lock =      0, request =  21208
    GRANT:   owner =  20840, lock =  21272, request =  21208
    CONVERT:     owner =  20840, lock =  21272, request =  21208
    GRANT:   owner =  20840, lock =  21272, request =  21208
    CONVERT:     owner =  20840, lock =  21272, request =  21208
    GRANT:   owner =  20840, lock =  21272, request =  21208
```

Рисунок 8.5 — Пример вывода записей истории

`GRANT:`
Предоставление блокировки на ресурс.

`ENQ:`
Помещение в очередь запрос.

`DENY:`
Отказ в запросе на ресурс.

`POST:`
Отправка сообщения владельцу по поводу ресурса.

`DEQ:`
Снятие блокировки владельцем.

`SCAN:`
Сканирование взаимных блокировок.

`WAIT:`
Владелец в состоянии ожидания.

`CONVERT:`
Повышение уровня блокировки.

8.8 Утилита rdbsvcmgr

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

`Services API` - программный интерфейс для запуска служебных задач (например, бэкап, рестор, проверка базы данных, сбор статистики и т.д.) без использования утилит. СУБД выполняет эти задачи с помощью менеджера сервисов (`service manager`). Утилита `rdbsvcmgr` обеспечивает доступ к сервисам.

Синтаксис вызова утилиты:

```
rdbsvcmgr <имя сервиса> <опции>
```

Имя сервиса должно быть `service_mgr`, может иметь префикс с именем хоста в соответствии с общими правилами (`host:service_mgr`, `\\\host\service_mgr`).

Опции в точности соответствуют тегам SPB, используемым в сокращенном виде. Удалите части тега `isc_, spb_, svc_` и вы получите опцию. Например, `isc_action_svc_backup` нужно указать как `action_backup`, `isc_spb_dbname` как `dbname`, `isc_info_svc_implementation` как `info_implementation`, `isc_spb_prp_db_online` как `prp_db_online` и так далее.

В одном вызове `rdbsvcmgr` можно указать либо одно действие, либо несколько информацион- ных элементов.

8.8.1 Подключение к менеджеру сервисов

Чтобы подключиться к удалённому менеджеру сервисов нужно указать имя локальной или удаленной службы. Эта строка похожа на строки подключения к базе данных, поскольку синтаксис определяет сетевой протокол, используемый для подключения клиентского приложения к менеджеру сервисов на хосте сервера.

Таблица 8.22 — Подключение к менеджеру сервисов

Опция	Описание
`user | user_name <имя пользователя>`	Имя пользователя.
`role | sql_role_name <роль>`	Роль.
`password <пароль>`	Пароль.
`fetch_password <файл>`	Считать пароль из файла.
`trusted_auth`	Использовать доверительную аутентификацию.
`expected_db <база данных>`	База данных безопасности, если она отличается от базы по умолчанию (`security5.fdb`).
`key_holder <имя плагина>`	Ключ, необходимый для доступа к зашифрованной базе данных.

Запуск `rdbrepldiff` через сервис:

```
./rdbsvcmgr localhost:service_mgr -user SYSDBA -password masterkey -action_diff
-dbname localhost:/my/other/database.fdb -dif_simple -dif_replica
john:smith@/my/replica/database2.fdb
```

8.8.2 Информационные запросы

Вы можете использовать следующие опции для получения информации о конфигурации сервера.

Таблица 8.23 — Информационные запросы

Опция	Описание
`info_server_version`	Версия сервера.
`info_implementation`	Аппаратная платформа, на которой была создана база данных.
`info_user_dbpath`	Путь к базе данных безопасности сервера.

(разрыв таблицы)

(разрыв таблицы)

Опция	Описание
info_get_env	Расположение корневого каталога сервера.
info_get_env_lock	Расположение файла таблицы блокировок на сервере.
info_get_env_msg	Расположение файла сообщений на сервере.
info_svr_db_info	Количество подключений к базе данных и баз данных, активных в данный момент на сервере.
info_version	Версия Service Manager.
info_capabilities	Битовая маска возможностей, поддерживаемых сервером.

Отображение баз данных, используемых на сервере, а также версии СУБД:

```
./rdbsvcmgr yourserver:service_mgr user sysdba password masterkey
info_server_version info_svr_db_info
```

8.8.3 Действия

В одном вызове менеджера сервисов можно выполнять только одну задачу. Вы можете поддерживать несколько подключений к менеджеру сервисов и выполнять задачу в каждом подключении.

Действие action_backup позволяет выполнять резервное копирование базы данных. Аналогично gbak -b.

Таблица 8.24 — Опции action_backup

Опция	Описание
dbname <путь к бд>	Путь к основному файлу базы данных.
verbose	Подробный вывод о бэкапе.
bkp_file <файл резервной копии>	Путь к файлу резервной копии. Если файл резервной копии уже существует, то резервное копирование не будет выполнено, если не указана опция bkp_replace.
bkp_replace	Перезаписать файл резервной копии, если он уже существует.
bkp_length <n>	Длина файла резервной копии в байтах.
bkp_factor <n>	Использовать блокирующий фактор n.
bkp_ignore_checksums	Игнорировать контрольные суммы, а также поврежденные BLOB при бэкапе и BLR при ресторе.
bkp_ignore_limbo	Игнорировать зависшие двухфазные транзакции (limbo).
bkp_metadata_only	Производит резервное копирование только метаданных.
bkp_no_garbage_collect	Не собирать мусор во время резервного копирования.
bkp_old_descriptions	Производит резервное копирование метаданных в формате старого стиля, т.е. в режиме совместимости со старыми базами данных.
bkp_non_transportable	Создаёт резервную копию в нетранспортабельном формате.
bkp_convert	Преобразование внешних файлов во внутренние таблицы.
bkp_no_triggers	Не запускать триггеры уровня базы данных.

(разрыв таблицы)

(разрыв таблицы)

Опция	Описание
`verbint <n>`	Подробный вывод лога действии с заданным интервалом обработанных записей.
`bkp_skip_data <шаблон>`	Пропускать данные таблиц, имена которых соответствуют указанному здесь регулярному выражению в SQL-синтаксисе (см. оператор `SIMILAR TO`).
`bkp_include_data <шаблон>]`	Включать только данные таблиц, имена которых соответствуют указанному здесь регулярному выражению в SQL-синтаксисе (см. оператор `SIMILAR TO`).
`bkp_stat TDRW`	Вывести статистику в процессе работы (работает вместе с `-verbose`). T — время от начала работы `gbak`; D — время прошедшее с последнего вывода на экран; R — число страниц прочитанных с последнего вывода на экран; W — число страниц записанных с последнего вывода на экран.
`bkp_keyholder <имя плагина>`	Это ключ, необходимый для доступа к зашифрованной базе данных.
`bkp_keyname <имя ключа>`	Явное присвоение имени ключу вместо ключа по умолчанию, указанного в исходной базе данных (при резервном копировании) или в файле резервной копии (при восстановлении).
`bkp_crypt <имя плагина>`	Имя плагина для шифрования файла резервной копии или восстановленной базы данных вместо плагина по умолчанию. Его также можно использовать в сочетании с ключом `-KEYNAME` для шифрования резервной копии незашифрованной базы данных или для шифрования базы данных, восстановленной из незашифрованной резервной копии.
`bkp_zip`	Сжать файл резервной копии перед его шифрованием. После шифрования файл почти не сжимается. Для сжатия используется плагин `zstd` с 5 уровнем сжатия в 1 поток.
`-bkp_compressor [<имя плагина> [<уровень сжатия> [<кол-во потоков>]]]`	Сжать файл резервной копии перед его шифрованием. После шифрования файл почти не сжимается. С РЕД Базой Данных поставляются плагины `zstd` и `zlibcompressor`. Плагин `zstd` поддерживает 22 уровня сжатия, а `zlibcompressor` 9. Количество потоков для `zstd` и `zlibcompressor` не ограничено. По умолчанию используется плагин `zstd` с 5 уровнем сжатия в 1 поток.
`bkp_parallel_workers <n>`	Количество параллельных рабочих потоков для бэкапа. Включает многопоточное выполнение резервного копирования.
`bkp_direct_io`	Открытие файла бекапа в режиме прямого доступа (без буферизации).

Действие `action_restore` позволяет выполнить восстановление базы данных из резервной копии. Аналогично `gbak -c`.

Таблица 8.25 — Опции `action_restore`

Опция	Описание
`bkp_file <файл резервной копии>`	Путь к файлу резервной копии.
`dbname <база данных>`	Путь к файлу базы данных.
`res_length <n>`	Количество страниц в восстановленном файле базы данных.
`verbose`	Подробный вывод о ресторе.
`res_buffers <n>`	Задать размер страничного кэша для БД.
`res_page_size <n>`	Установить размер страницы.
`res_access_mode [prp_am_readonly | prp_am_readwrite]`	Режим доступа `"read_only"` или `"read_write"`.
`res_deactivate_idx`	Деактивировать индексы во время восстановления.
`res_no_shadow`	Восстановить без создания теневых копий.
`res_no_validity`	Отключение всех ограничений проверки данных (`check`) и ограничений `NOT NULL`.
`res_one_at_a_time`	Подтверждать транзакцию после восстановления каждой таблицы.
`res_replace`	Заменить базу данных, если она существует.
`res_create`	Восстановить, но не перезаписывать существующую базу данных.
`res_use_all_space`	Не резервировать место под версии записей.
`res_fix_fss_data <кодировка>`	Исправить кодировку данных.
`res_fix_fss_metadata <кодировка>`	Исправить кодировку метаданных.
`res_metadata_only`	Производит восстановление только метаданных.
`verbint <n>`	Подробный вывод лога действии с заданным интервалом обработанных записей.
`res_skip_data <шаблон>`	Пропускать данные таблиц, имена которых соответствуют указанному здесь регулярному выражению в SQL-синтаксисе (см. оператор `SIMILAR TO`).
`res_include_data <шаблон>`	Включать только данные таблиц, имена которых соответствуют указанному здесь регулярному выражению в SQL-синтаксисе (см. оператор `SIMILAR TO`).
`res_stat TDRW`	Вывести статистику в процессе работы (работает вместе с `-verbose`). T — время от начала работы `gbak`; D — время прошедшее с последнего вывода на экран; R — число страниц прочитанных с последнего вывода на экран; W — число страниц записанных с последнего вывода на экран
`res_keyholder <имя плагина>`	Это ключ, необходимый для доступа к зашифрованной базе данных.
`res_keyname <имя ключа>`	Явное присвоение имени ключу вместо ключа по умолчанию, указанного в исходной базе данных (при резервном копировании) или в файле резервной копии (при восстановлении).

(разрыв таблицы)

(разрыв таблицы)

Опция	Описание
res_crypt <имя плагина>	Имя плагина для шифрования файла резервной копии или восстановленной базы данных вместо плагина по умолчанию. Его также можно использовать в сочетании с ключом -KEYNAME для шифрования резервной копии незашифрованной базы данных или для шифрования базы данных, восстановленной из незашифрованной резервной копии.
res_replica_mode [prp_rm_none | prp_rm_readonly | prp_rm_readwrite]	Выбрать режим реплики - только для чтения (read_only) или для чтения и записи (read_write). Режим none отключает репликацию и переводит реплику в режим read_write. Как правило, этот режим должен использоваться после сбоя в базе данных мастера, когда эта реплика переходит в статус мастера. Или если нужно сделать физические резервные копии из реплики.
res_parallel_workers <n>	Количество параллельных рабочих потоков для рестора. Включает многопоточное выполнение восстановления БД, если MaxParallelWorkers > 1 (см. firebird.conf). Вместо этого параметра утилиты можно указать параметр конфигурации ParallelWorkers.
res_direct_io	Открытие файла бекапа в режиме прямого доступа (без буферизации).

Действие action_properties позволяет настроить параметры базы данных. Аналогично gfix.

Таблица 8.26 — Опции action_properties

Опция	Описание
dbname <база данных>	Путь к файлу базы данных.
prp_page_buffers <n>	Устанавливает новый размер страничного кэша БД в страницах. <n> - количество страниц.
prp_sweep_interval <n>	Изменяет интервал транзакций для автоматической сборки мусора (sweep). <n> устанавливает новый интервал. Если n = 0, автоматическая сборки мусора не выполняется.
prp_shutdown_db <n>	Закрывает базу данных, когда нет соединений с базой данных, или по истечении указанного таймаута.
prp_deny_new_transactions <n>	Завершает работу базы данных, если в конце указанного таймаута нет активных транзакций; запрещает новые транзакции в течение этого периода ожидания; завершается ошибкой, если в конце периода ожидания есть активные транзакции.
prp_deny_new_attachments <n>	Завершает работу базы данных, если в конце указанного таймаута нет активных подключений; запрещает новые подключения к базе данных в течение периода ожидания; завершится ошибкой, если в конце периода ожидания есть активные подключения.
prp_set_sql_dialect <n>	Изменяет диалект базы данных. <n> может быть 1 или 3.
prp_access_mode [prp_am_readonly | prp_am_readwrite]	Устанавливает режим записи для базы данных - только для чтения или чтение/запись. Этот параметр может принимать два значения.

(разрыв таблицы)

(разрыв таблицы)

Опция	Описание
prp_reserve_space [prp_res_use_full | prp_res]	Включает 100% заполнение страниц БД (full) или 80% заполнение по умолчанию (reserve). 100%-е заполнение имеет смысл для баз только для чтения.
prp_write_mode [prp_wm_async | prp_wm_sync]	Переключает режимы синхронной/асинхронной записи Forced Writes.
prp_activate <теневая копия>	Параметр предназначен для активации теневой копии. <теневая копия> - адрес и имя файла теневой копии (или первого из файлов).
prp_db_online	Возвращает в режим "online" базу, остановленную с помощью -shut
prp_force_shutdown <n>	Предназначен для принудительного закрытия базы данных. <n> указывает количество секунд, через которое произойдет закрытие. Если остались активные пользователи, они отключатся, последние результаты их работы будут потеряны. Такое средство нужно применять с осторожностью.
prp_attachments_shutdown <n>	Предназначен для запрета новых соединений с БД. <n> указывает количество секунд, через которое произойдет отключение БД. Отключение отменится, если к этому времени еще останутся активные соединения.
prp_transactions_shutdown <n>	Предназначен для запрета запуска новых транзакций. <n> указывает количество секунд, через которое произойдет отключение БД. Отключение отменится, если к этому времени еще останутся активные транзакции.
prp_shutdown_mode [prp_sm_normal | prp_sm_multi | prp_sm_single | prp_sm_full]	Останавливает базу данных. Используется с одним из дополнительных параметров -attach, -force или -tran. Опция multi позволяет устанавливать соединение с БД только SYSDBA и владельцем БД. Опция single похожа на multi, за тем исключением, что только один из пользователей – SYSDBA или владелец БД – может соединиться. Опция full не позволяет соединиться с БД никому, даже SYSDBA или владельцу БД.
prp_online_mode [prp_sm_normal | prp_sm_multi | prp_sm_single | prp_sm_full]	Возвращает в режим "online" базу, остановленную с помощью -shut. Опция normal позволяет устанавливать соединение с БД любым авторизованным пользователям, а не только SYSDBA и владельцем БД. Опция multi позволяет устанавливать соединение с БД только SYSDBA и владельцем БД. Опция single похожа на multi за тем исключением, что допускается только одно подключение от SYSDBA или владельца БД.
prp_nolinger	Останавливает указанную базу данных, после того как последнего соединения не стало, независимо от установок LINGER в базе данных.
prp_replica_mode [prp_rm_none | prp_rm_readonly | prp_rm_readwrite]	Выбрать режим реплики - только для чтения (read_only) или для чтения и записи (read_write). Режим none отключает репликацию и переводит реплику в режим read_write. Как правило, этот режим должен использоваться после сбоя в базе данных мастера, когда эта реплика переходит в статус мастера. Или если нужно сделать физические резервные копии из реплики.

(разрыв таблицы)

(разрыв таблицы)

Опция	Описание
`prp_checksums [true | false]`	Включает/отключает вычисление контрольных сумм на уровне страниц базы данных. После включения контрольные суммы начнут пересчитываться при последующей записи страниц базы данных на диск. В случае несоответствия контрольных сумм в лог-файл будет записана ошибка, но работа продолжится.

Действие `action_repair` позволяет инициировать проверку согласованности базы данных и исправление найденных ошибок. Аналогично `gfix` с опциями `-validate, -full` и `-mend`.

Таблица 8.27 — Опции `action_repair`

Опция	Описание
`dbname <база данных>`	Путь к файлу базы данных.
`rpr_commit_trans {<ID> | all}`	Завершает подтверждением зависшую (`in limbo`) двухфазную транзакцию с идентификатором `<ID>`, или все зависшие транзакции (`all`).
`rpr_rollback_trans {<ID> | all}`	Завершает откатом зависшую (`in limbo`) двухфазную транзакцию с идентификатором `<ID>`, или все зависшие транзакции (`all`).
`rpr_recover_two_phase {<ID> | all}`	Автоматическое двухфазное восстановление `limbo` транзакции с номером `<ID>`, или всех транзакций (`all`).
`rpr_commit_trans_64 {<ID> | all}`	Завершает подтверждением зависшую (`in limbo`) двухфазную транзакцию с идентификатором `<ID>`, или все зависшие транзакции (`all`).
`rpr_rollback_trans_64 {<ID> | all}`	Завершает откатом зависшую (`in limbo`) двухфазную транзакцию с идентификатором `<ID>`, или все зависшие транзакции (`all`).
`rpr_recover_two_phase_64 {<ID> | all}`	Автоматическое двухфазное восстановление `limbo` транзакции с номером `<ID>`, или всех транзакций (`all`).
`rpr_check_db`	Проверка базы данных без исправления каких-либо проблем.
`rpr_ignore_checksum`	Игнорировать ошибки контрольных сумм при проверке.
`rpr_kill_shadows`	Удаляет все неиспользуемые теневые копии базы данных.
`rpr_mend_db`	Подготавливает повреждённую базу данных для резервного копирования. Помечает повреждённые записи как неиспользуемые.
`rpr_validate_db`	Проверяет базу данных на уровне страниц и освобождает страницы, помеченные как используемые, но никому не принадлежащие (`orphans`).
`rpr_full`	Используется вместе с `-validate_db` для более глубокой проверки на уровне записей; освобождает неназначенные фрагменты записей.
`rpr_sweep_db`	Запускает принудительную сборку мусора в БД.
`rpr_list_limbo_trans`	Выводит список «зависших» транзакций (`limbo`).
`rpr_icu`	Исправляет базу данных для работы с текущей версией ICU.

(разрыв таблицы)

(разрыв таблицы)

Опция	Описание
`rpr_par_workers <n>`	Количество параллельных рабочих потоков для сборки мусора. Включает многопоточную сборку мусора, если `MaxParallelWorkers > 1` (см. `firebird.conf`). Вместо этого параметра утилиты можно указать параметр конфигурации `ParallelWorkers`.
`rpr_upgrade_db`	Обновляет базу данных до последней минорной версии `ODS` в пределах поддерживаемой мажорной версии.

Действие `action_db_stats` выводит статистику базы данных. Аналогично `gstat`.

Таблица 8.28 — Опции `action_db_stats`

Опция	Описание
`dbname <база данных>`	Путь к файлу базы данных.
`sts_record_versions`	Добавляет статистику о средних длинах записей, количестве версий и информацию о `BLOB`.
`sts_nocreation`	Не выводить дату создания.
`sts_table <таблица>`	Анализ таблицы или списка таблиц и любых индексов, принадлежащих указанным таблицам. За этим параметром следует список имен таблиц, которые вы хотите проанализировать. Список должен быть в верхнем регистре, и каждая таблица разделяется пробелом.
`sts_data_pages`	Статистика страниц данных – информация о таблицах содержащихся в базе данных. Пользовательские и системные индексы, системные таблицы не анализируются.
`sts_hdr_pages`	Статистика заголовочной страницы - это информация о глобальных свойствах всей базы данных, хранящаяся на заголовочной странице каждой базы данных.
`sts_idx_pages`	Статистика индексов – информация об индексах в базе данных. Пользовательские и системные таблицы, системные индексы не анализируются.
`sts_sys_relations`	Статистика для системных таблиц и индексов в дополнение к пользовательским таблицам и индексам.
`sts_encryption`	Статистика по зашифрованным и незашифрованным страницам БД.
`sts_par_workers <n>`	Количество параллельных рабочих потоков для сбора статистики. Включает многопоточность, если `MaxParallelWorkers > 1` (см. `firebird.conf`). Вместо этого параметра утилиты можно указать параметр конфигурации `ParallelWorkers`.

Отображение информации заголовка базы данных `employee` на локальном компьютере:

```
./rdbsvcmgr service_mgr user sysdba password masterkey action_db_stats dbname
employee sts_hdr_pages
```

Действие `action_diff` выполняет проверку целостности данных. Аналогично `rdbrepldiff`.

Таблица 8.29 — Опции `action_diff`

Опция	Описание
`dbname <база данных>`	Путь к файлу базы данных.
`verbose`	Подробный вывод о проверке.
`dif_replica`	Путь к реплике базы данных.
`dif_config`	Имя измененного файла конфигурации для возможности сравнивать не все сущности базы данных, а только необходимые. При этом процесс репликации не прерывается.
`dif_sync`	Ожидание синхронизации мастера и слейва в течение одной минуты или времени, указанного в параметре `dif_timeout`.
`dif_metaonly`	Производит проверку только метаданных.
`dif_timeout`	Таймаут ожидания синхронизации (в секундах).
`dif_nodbtriggersoff`	Включает ипользование триггеров при подключении к мастеру.
`dif_simple`	Сравнение двух нереплицируемых баз данных.
`dif_full`	Показать расширенный вывод о проверке. Расширенный вывод отображает: • Записи, которые отсутсвуют на слейве; • Поля, которые отличаются у записей; • Разницу в количестве записей на мастере и на слейве.
`dif_values`	Показывает значения, которые отличаются у записей. Можно использовать только совместно с опцией `dif_full`.

Можно использовать `Services API` для отображения, добавления, удаления и изменения пользователей аналогично `gsec`.

Действие `action_display_user` выводит информацию обо всех зарегистрированных пользователях. Действие `action_display_user_adm` дополнительно показывает являются ли они администраторами БД.

Таблица 8.30 — Опции `action_display_user` и `action_display_user_adm`

Опция	Описание
`dbname <БД_безопасности>`	Путь к файлу базы данных безопасности.
`sec_username <имя пользователя>`	Имя пользователя, для которого будет выполнена операция.
`sql_role_name <имя роли>`	Название роли, чьи права будет использовать указанный пользователь.

Действие `action_add_user` нужно для добавления нового пользователя. Аналогично `gsec -add`.

Таблица 8.31 — Опции `action_add_user`

Опция	Описание
`dbname <база данных>`	Путь к файлу базы данных
`sec_username <имя пользователя>`	Имя пользователя
`sql_role_name <имя роли>`	Название роли, чьи права будет использовать указанный пользователь.

(разрыв таблицы)

(разрыв таблицы)

Опция	Описание
`sec_password <пароль>`	Пароль.
`sec_groupname <имя группы>`	Группа.
`sec_firstname <имя>`	Имя.
`sec_middlename <отчество>`	Отчество.
`sec_lastname <фамилия>`	Фамилия.
`sec_userid <uid>`	Идентификатор пользователя.
`sec_groupid <gid>`	Идентификатор группы.
`sec_admin <число>`	Если значение не равно 0, пользователю будут назначены права администратора

Действие `action_delete_user` нужно для удаления нового пользователя. Аналогично `gsec -delete`.

Таблица 8.32 — Опции `action_delete_user`

Опция	Описание
`dbname <база данных>`	Путь к файлу базы данных.
`sec_username <имя пользователя>`	Имя пользователя.
`sql_role_name <роль>`	Роль.

Действие `action_modify_user` нужно для удаления изменения информации о пользователе. Аналогично `gsec -modify`.

Таблица 8.33 — Опции `action_modify_user`

Опция	Описание
`dbname <база данных>`	Путь к файлу базы данных.
`sec_username <имя пользователя>`	Имя пользователя.
`sql_role_name <роль>`	Роль.
`sec_password <пароль>`	Пароль.
`sec_groupname <имя группы>`	Группа.
`sec_firstname <имя>`	Имя.
`sec_middlename <отчество>`	Отчество.
`sec_lastname <фамилия>`	Фамилия.
`sec_userid <uid>`	Идентификатор пользователя.
`sec_groupid <gid>`	Идентификатор группы.
`sec_admin <число>`	Если значение не равно 0, пользователю будут назначены права администратора

Действие `action_nbak` позволяет создавать инкрементные резервные копии.

Таблица 8.34 — Опции `action_nbak`

Опция	Описание
`dbname <база данных>`	Путь к файлу базы данных.
`nbk_file <резервный файл>`	Путь к файлу резервной копии.
`nbk_level <уровень>`	Уровень резервной копии.
`nbk_guid <GUID>`	`GUID` предыдущего бекапа, относительно которого нужно создавать инкремент.
`nbk_no_triggers`	Не запускать триггеры базы данных.
`nbk_direct [ON | OFF]`	Включает/выключает небуферизованный ввод/вывод при чтении БД.
`nbk_clean_history`	Очистить таблицу `RDB$HISTORY`.
`nbk_keep_days <число>`	Определяет сколько дней, начиная с сегодняшнего дня, должно храниться в истории.
`nbk_keep_rows <число>`	Определяет сколько записей должно храниться в истории.

Действие `action_nrest` нужно для восстановления базы данных из инкрементных резервных копий.

Таблица 8.35 — Опции `action_nrest`

Опция	Описание
`dbname <база данных>`	Путь к файлу базы данных.
`nbk_file <резервный файл>`	Путь к файлу резервной копии.
`nbk_inplace`	Восстанавливать инкремент в существующую базу данных.
`nbk_sequence`	Сохраняет существующий `GUID` и последовательность репликации исходной базы данных (в противном случае они сбрасываются). Эта опция должна использоваться при создании реплики, чтобы асинхронная репликация могла быть автоматически продолжена с того момента, когда было выполнено физическое резервное копирование на основной базе.

Команда `action_nfix` нужна для разблокировки базы данных после самостоятельного восстановления из блокированной резервной копии. Так как копия блокированной базы данных является так же блокированной, поэтому не получится использовать копию как рабочую базу данных. Поэтому, если исходная база данных повреждена или утеряна, то нужно восстановить/разархивировать/скопировать базу данных из копии и разблокировать ее с параметром `-F` (но не `-UN`).

Таблица 8.36 — Опции `action_nfix`

Опция	Описание
`dbname <база данных>`	Путь к файлу базы данных.
`nbk_sequence`	Сохраняет существующий `GUID` и последовательность репликации исходной базы данных (в противном случае они сбрасываются). Эта опция должна использоваться при создании реплики, чтобы асинхронная репликация могла быть автоматически продолжена с того момента, когда было выполнено физическое резервное копирование на основной базе.

Таблица 8.37 — Опции `action_diff`

Опция	Описание
`dbname <база данных>`	Путь к файлу базы данных.
`verbose`	Подробный вывод о проверке.
`dif_replica <файл>`	Вторая база (реплика).
`dif_config <файл>`	Имя измененного файла конфигурации для возможности сравнивать не все сущности базы данных, а только необходимые. При этом процесс репликации не прерывается.
`dif_sync`	Ожидание синхронизации мастера и слейва в течение одной минуты или времени, указанного в параметре `-dif_timeout`.
`dif_metaonly`	Производит проверку только метаданных.
`dif_timeout <таймаут>`	Таймаут ожидания синхронизации (в секундах).
`dif_nodbtriggersoff`	Включает ипользование триггеров при подключении к мастеру
`dif_simple`	Сравнение двух нереплицируемых баз данных.

Таблица 8.38 — Действия для ведения аудита

Действие	Описание
`action_trace_start -trc_cfg <конфигурационный файл> -trc_name <имя сессии>`	Запускает сессию пользовательской трассировки.
`action_get_fb_log, action_get_ib_log`	Отображают лог СУБД.
`action_trace_suspend -trc_id <ID сессии>`	Приостановление сеанса трассировки.
`action_trace_resume -trc_id <ID сессии>`	Возобновление сеанса трассировки.
`action_trace_stop -trc_id <ID сессии>`	Завершение сеанса трассировки.
`action_trace_list`	Вывод списка существующих сеансов трассировки.

Действие `action_aggtrace` нужно для запуска агрегатного аудита.

Таблица 8.39 — Опции `action_aggtrace`

Опция	Описание
`atrc_get`	Запросить метрики новых и обновлённых запросов.
`atrc_get_new`	Запросить метрики только новых запросов.
`atrc_get_updated`	Запросить метрики только обновлённых запросов т.е тех, у которых изменились значения метрик.
`atrc_get_old`	Запросить метрики только старых запросов, то есть тех, которые уже запрашивались ранее, но значения метрик не изменились.
`atrc_get_all`	Запросить метрики по всем статусам.
`atrc_clear`	Очистить сохранённую статистику.

(разрыв таблицы)

(разрыв таблицы)

Опция	Описание
`atrc_with_query`	Добавлять текст запроса в вывод.
`atrc_with_plan`	Добавлять план запроса в вывод.
`atrc_get_version`	Вывести версию плагина агрегатного аудита.
`atrc_statement`	Запросить метрики событий. Если указана какая-либо `"get"` опция (`atrc_get_new`, `atrc_get_old`, `atrc_get_all`, `atrc_with_query`, `atrc_get` или `atrc_get_updated`), то `atrc_statement` будет применена автоматически.
`atrc_transaction`	Запросить метрики событий транзакций.
`atrc_text {<хэш запроса>|<хэш плана>}`	Вывести запрос или план запроса по указанному хэшу.

Действие `action_migrate` выполняет миграцию.

Таблица 8.40 — Опции `action_migrate`

Опция	Описание
`dbname []`	
`mig_version [string value]`	

Таблица 8.41 — Работа с отображениями

Действие	Описание
`action_set_mapping -dbname <база данных> -sql_role_name <роль>`	Предоставляет администраторам `Windows` роль `RDB$ADMIN` в указанной базе данных, если используется доверительная авторизация
`action_drop_mapping -dbname <база данных> -sql_role_name <роль>`	Снимает с администратора `Windows` роль `RDB$ADMIN` в указанной базе данных.

Действие `action_validate` позволяет запустить онлайн-проверку базы данных.

Таблица 8.42 — Опции `action_validate`

Опция	Описание
`-dbname <база данных> -sql_role_name <роль>`	Снимает с администратора `Windows` роль `RDB$ADMIN` в указанной базе данных.
`dbname <база данных>`	Путь к файлу базы данных.
`val_tab_incl <шаблон>`	Шаблон таблиц, включенных в проверку. По умолчанию все пользовательские таблицы включены, системные таблицы не проверяются.
`val_tab_excl <шаблон>`	Шаблон для таблиц, исключенных из проверки.
`val_idx_incl <шаблон>`	Шаблон для индексов, включенных в проверку. По умолчанию `%` – все индексы включены.
`val_idx_excl <шаблон>`	Шаблон списка индексов для исключения из процесса валидации.

(разрыв таблицы)

(разрыв таблицы)

Опция	Описание
`val_lock_timeout <число>`	Таймаут ожидания блокировки на таблицу (в секундах). По умолчанию 10 секунд, -1 - бесконечное ожидание.

8.9 Утилита rdbguard

Утилита `rdbguard` контролирует состояние сервера. Если сервер был нештатно остановлен, `Guardian` автоматически перезапускает его.

Таблица 8.43 — Опции `rdbguard`

Опция	Описание
`-s(ignore)`	Если запуск сервера завершается с ошибкой, выполнять повторные попытки запуска.
`-(o)netime`	Если запуск сервера завершается с ошибкой, не пытаться запустить его повторно.
`-(f)orever`	Если запуск или выполнение сервера завершается с ошибкой, выполнять его повторный запуск (режим по умолчанию).
`-(d)aemon`	Запускать сервер в режиме демона, отключаясь от родительского терминала.
`-(p)idfile путь_к_файлу`	Использовать указанный файл в качестве PID-файла сервера.
`-(g)uardpidfile путь_к_файлу`	Использовать указанный файл в качестве PID-файла `rdbguard`.
`-(t)imeout таймаут`	Время в секундах, которое `rdbguard` будет ожидать с момента начала остановки сервера, прежде чем убить его процесс. По умолчанию 1800 секунд (30 минут). Если сервер не успевает остановиться в отведенное время, то это может привести к неполной записи на диск измененных страниц из кэша. В этом случае целостность базы данных не нарушается, но может наблюдаться снижение производительности.

8.10 Коды возврата утилит

Для утилит `gfix`, `gbak`, `gstat`, `rdbsvcmgr`, `rdbtracemgr`, `rdblogmgr`, `rdbreplmgr`, `hashgen`, `mint`, `nbackup`, `rdb_lock_print`:

0 – успешно
1 – ошибка

Для утилиты `gbak`:

0 – успешно
1 – ошибка
2 – база данных не переведена в режим `online` из-за ошибки при восстановлении из резервной копии

Для утилиты `rdbrepldiff`:

0 – успешно (базы идентичны)
1 – базы не идентичны

2 – ошибка в процессе работы

Для утилиты `gsec`:

0 – успешно
15 – невозможно открыть базу данных
16 – ошибка в опции утилиты
17 – операция не указана
18 – имя пользователя не указано
19 – ошибка добавления записи
20 – ошибка изменения записи
21 – ошибка поиска/изменения записи
22 – запись не найдена для пользователя:
23 – ошибка удаления записи
24 – ошибка поиска/удаления записи
28 – ошибка поиска/отображения записи
29 – недопустимый параметр, опция не определена
30 – операция уже указана
31 – пароль уже указан
32 – идентификатор пользователя уже указан
33 – идентификатор группы уже указан
34 – проект уже указан
35 – организация уже указан
36 – имя уже указаноОткрытие файла бекапа в режиме прямого доступа
37 – отчество уже указано
38 – фамилия уже указана
40 – указана недопустимая опция
41 – указана неоднозначная опция
42 – не указана операция для параметров
43 – нет допустимых параметров для этой операции
44 – указаны несовместимые опции
74 – ошибка выделения памяти
75 – ошибка
76 – недопустимое имя пользователя (допускается не более 31 байта)
77 – недопустимый пароль (допускается не более 16 байт)
78 – база данных уже указана
79 – имя администратора базы данных уже указано
80 – пароль администратора базы данных уже указан
81 – роль уже указана
92 – указана недопустимая в интерактивном режиме опция
93 – ошибка при закрытии базы данных безопасности
94 – ошибка при выполнении запроса в базе данных безопасности
96 – ошибка при получении пароля из файла
97 – ошибка при изменении `AUTO ADMINS MAPPING` в базе данных безопасности
99 – недопустимый параметр для `-MAPPING`, допускается только `SET` или `DROP`
103 – недопустимый параметр для `-ADMIN`, принимается только `YES` или `NO`
104 – недостаточно прав для завершения операции
200 – пользователь не найден в `LDAP`

200 – ошибка при изменении пароля `LDAP`

Для утилиты `rdbserver`:

0 – успешно
1 – ошибка
2 – ошибка запуска

Для утилиты `isql`:

0 – успешно
1 – ошибка
101 – ошибка аутентификации
102 – ошибка ввода/вывода
103 – ошибка `DSQL`
104 – ошибка функции `SHOW`

Глава 9

Администрирование функций безопасности

9.1 Основные термины и определения

Сервер баз данных (далее сервер) — установленная СУБД РЕД База Данных. Для соединения с сервером по сети по умолчанию используется порт 3050 (настраивается через параметр `RemoteServicePort` в конфигурационном файле РЕД Базы 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, в которой хранится информация о пользователях, зарегистрированных для конкретного сервера РЕД Базы Данных - `security5.fdb` (расположена в корневой директории каталога установки сервера). Для каждой базы данных база данных безопасности может переопределена в файле `databases.conf` (параметр `SecurityDatabase`). Любая база данных может быть базой данных безопасности для самой себя. В этой базе хранятся параметры пользователей системы и политики доступа (см. Приложение Г).

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 — метод аутентификации из любого плагина.

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 хранятся в той базе данных, в которой они были созданы.

Администратор сервера БД — пользователь с именем `SYSDBA`. Создается при установке сервера. Обладает всеми правами по управлению работой сервера и полным доступом ко всем базам данных сервера. Пароль для `SYSDBA` по умолчанию – `masterkey`.

Системный администратор — пользователь, которому назначена роль `RDB$ADMIN`. Предназначен для распределения прав пользователей, а также для операций обслуживания баз данных (резервное копирование, восстановление и т.д.).

Системный каталог — совокупность системных таблиц, содержащая информацию обо всех объектах базы данных. Создается при создании БД и изменяется при изменении метаданных в БД.

9.2 Общая модель защиты

Модель защиты описывает совокупность объектов защиты, субъектов доступа к защищаемым объектам и используемые механизмы безопасности, обеспечивающие выполнение заданных требований к безопасности информации.

Объектами защиты в РЕД Базе Данных являются:

- хранящиеся в РЕД Базе Данных пользовательские данные (записи, поля);
- данные системного каталога (метаданные);

- операции над данными и метаданными.

Субъектами доступа являются пользователи системы, прошедшие процесс идентификации и аутентификации, а также запущенные от их имени процедуры и функции.

Система защиты состоит из следующих механизмов безопасности:

- идентификация и аутентификация;
- ролевое разграничение доступа;
- распределение системных привилегий;
- разграничение доступа к объектам базы данных;
- аудит событий;
- шифрование информации;
- очистка освобождаемых ресурсов;
- контроль целостности информационных объектов.

9.3 Специальные учетные записи

SYSDBA

Изначально в системе существует только один пользователь – администратор сервера `SYSDBA`. Этот пользователь обладает полными правами на выполнение всех функций по управлению работой сервера и работе с базами данных.

Пользователи POSIX

В POSIX системах, включая MacOSX, имя пользователя POSIX будет интерпретировано как имя пользователя РЕД Базы Данных Embedded, если имя пользователя не указано явно.

В POSIX системах пользователь `root` может выступать в роли `SYSDBA`. РЕД База Данных Embedded в этом случае будет трактовать имя пользователя `root` как `SYSDBA`, и он будет иметь доступ ко всем базам данных сервера.

Пользователи Windows

В операционных системах семейства Windows NT вы также можете пользоваться учётными записями ОС. Для этого необходимо, чтобы в файле конфигурации `firebird.conf` (параметр `AuthServer`) в списке плагинов присутствовал провайдер `Win_Sspi`. Кроме того, этот плагин должен присутствовать и в списке плагинов клиентской стороны (параметр `AuthClient`).

Администраторы операционной системы Windows автоматически не получают права `SYSDBA` при подключении к базе данных (если, конечно, разрешена доверенная авторизация). Имеют ли администраторы автоматические права `SYSDBA`, зависит от установки значения флага `AUTO ADMIN MAPPING`.

> До версии 3 при включенной доверительной аутентификации, пользователи прошедшие проверку по умолчанию автоматически отображались в `CURRENT_USER`. В версии 3 и выше отображение должно быть сделано явно для систем с несколькими базами данных безопасности и включенной доверительной аутентификацией.

Владельцы базы данных

Владелец базы данных — это либо текущий пользователь (`CURRENT_USER`), который был в момент создания, либо пользователь который был указан в параметрах `USER` и `PASSWORD` в операторе `CREATE DATABASE`.

Владелец базы данных является администратором в ней и имеет полный доступ ко всем объектам

этой базы данных, даже созданных другими пользователями.

Администраторы

Администратор — это пользователь, которые имеет достаточные права для чтения и записи, создания, изменения и удаления любого объекта в базе данных. В таблице показано, как привилегии «Суперпользователя» включены в различных контекстах безопасности.

Таблица 9.1 — Администраторы

Пользователь	Роль RDB$ADMIN	Замечание
`SYSDBA`	Автоматически	Существует автоматически на уровне сервера. Имеет полные привилегии ко всем объектам во всех базах данных. Может создавать, изменять и удалять пользователей.
Пользователь `root` или суперпользователь в POSIX	Автоматически	Также как `SYSDBA`. Только в РЕД Базе Данных Embedded.
Владелец базы данных	Автоматически	Также как `SYSDBA`, но только в этой базе данных.
Администраторы Windows	Устанавливается в `CURRENT_ROLE`, если вход успешен	Также как `SYSDBA`, если соблюдены следующие условия: • В файле конфигурации `firebird.conf` (параметр `AuthServer`) в списке плагинов присутствовал провайдер `Win_Sspi`. Кроме того, этот плагин должен присутствовать и в списке плагинов клиентской стороны (параметр `AuthClient`). • Во всех базах данных, где требуется полномочия суперпользователя должен быть включен `AUTO ADMIN MAPPING` или создано отображение предопределенной группы `DOMAIN_ANY_RID_ADMINS` на роль `RDB$ADMIN`. • При входе не указана роль.
Обычный пользователь	Должна быть предварительно выдана и должна быть указана при входе	Также как `SYSDBA`, но только в тех базах данных, где эта роль предоставлена.
Пользователь POSIX	Должна быть предварительно выдана и должна быть указана при входе	Также как `SYSDBA`, но только в тех базах данных, где эта роль предоставлена. Только в РЕД Базе Данных Embedded.
Пользователь Windows	Должна быть предварительно выдана и должна быть указана при входе	Также как `SYSDBA`, но только в тех базах данных, где эта роль предоставлена. Доступно только если в файле конфигурации `firebird.conf` (параметр `AuthServer`) в списке плагинов присутствовал провайдер `Win_Sspi`. Кроме того, этот плагин должен присутствовать и в списке плагинов клиентской стороны (параметр `AuthClient`).

9.4 Идентификация и аутентификация

В СУБД РЕД База Данных идентификация и аутентификация основана на имени пользователя, его пароле, а также других факторах аутентификации.

Для того, чтобы пройти процедуру идентификации, пользователь обязан предъявить свой логин (имя пользователя). Основываясь на этой информации, сервер в дальнейшем определит, как должна происходить аутентификация пользователя, и какие права сможет получить пользователь после прохождения аутентификации.

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 (как правило, пароля или сертификата).

Весь функционал, который относится к аутентификации, реализован в виде сторонних плагинов: `Legacy_Auth`, `Gss`, `Srp`, `Win_Sspi`, `GostPassword`, `Certificate` и `VerifyServer`. Они реализуют следующие методы аутентификации:

- Безопасная парольная аутентификация использующая алгоритм хэширования SHA для передачи данных: `Srp, Srp224, Srp256, Srp384, Srp512`. По умолчанию используется `Srp256`;
- Традиционная (`Legacy_Auth`) аутентификация, унаследованная от предыдущих версий;
- Доверительная (`Win_Sspi`) аутентификация для ОС Windows;
- Метод `GostPassword` обеспечивает аутентификацию с использованием алгоритмов шифрования из криптографического плагина (Crypto_API)[1];
- Плагин `Certificate` позволяет аутентифицировать пользователей по сертификатам X509;
- Плагин `VerifyServer` позволяет клиенту проверить сертификат сервера;
- Доверенная аутентификация через механизм GSSAPI (`Gss`);
- Доверенная аутентификации для выполнения `Execute Statement On External` без указания пароля (`ExtAuth`).

Плагины аутентификации состоят из трех частей и все три части должны быть настроены отдельно в `firebird.conf`:

- Клиентская часть (параметр `AuthClient`) - подготавливает данные на клиенте для отправки на сервер;
- Серверная часть (параметр `AuthServer`) - проверяет пароль на правильность;
- Менеджер пользователей (параметр `UserManager`) - добавляет, изменяет и удаляет пользователей на сервере. Это не требуется, если используется какой-либо внешний метод аутентификации, такой как доверенная аутентификация Windows.

Параметр `AuthClient` настраивается на клиентской машине и содержит список методов, поддерживаемых клиентом. Параметр `AuthServer` настраивается на сервере и содержит список доступных методов аутентификации, разрешенных сервером. Методы перечисляются в виде строковых символов (слов), разделенных запятыми, точками с запятой или пробелами.

Подлинность пользователя проверяется методом, указанным первым в списке. Если проверить подлинность с помощью него не удалось, то сервер переходит к следующему и т.д. Если ни один метод не подтвердил подлинность, то пользователь получает сообщение об ошибке.

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.

[1] Для работы данной функции необходимо наличие криптопровайдера и криптоплагина (последний входит в поставку СУБД РЕД База Данных - `crypto_api`). Криптопровайдер реализует функции шифрования и хэширования, а криптоплагин предоставляет унифицированный интерфейс между криптопровайдером и сервером СУБД РЕД База Данных.

9.4.1 Управление пользователями

Создать пользователя можно с помощью одного из плагинов управления пользователями, указанных в параметре `UserManager` (файла конфигурации `firebird.conf`): `Srp, GostPassword_Manager, Legacy_UserManager`. По умолчанию используется тот плагин, который был указан первым в списке.

Информация о пользователях, зарегистрированных для конкретного сервера Ред Базы Данных, хранится в особой базе данных безопасности — `security5.fdb`. Для каждой базы данных база данных безопасности может быть переопределена в файле `databases.conf` (параметр `SecurityDatabase`). Любая база данных может быть базой данных безопасности для самой себя.

> Следует учитывать, что одноименные пользователи, созданные с помощью разных плагинов управления пользователями — это разные пользователи.

Это связано с тем, что методы парольной аутентификации используют разные таблицы в базе данных безопасности для хранения данных пользователей. Поэтому для системы аутентификации пользователь, созданный менеджером `Legacy_UserManager`, никак не связан с пользователем, созданным `Srp`. У них разные пароли и другая пользовательская информация. Но с точки зрения движка эти пользователи одинаковые, так как движок идентифицирует пользователей по именам.

Можно управлять учётными записями пользователей средствами операторов SQL. Такая возможность предоставлена следующим пользователям:

- `SYSDBA`;
- Любому пользователю, имеющему права на роль `RDB$ADMIN` в базе данных безопасности и права на ту же роль для базы данных в активном подключении (пользователь должен подключаться к базе данных с ролью `RDB$ADMIN`);
- При включенном флаге `AUTO ADMIN MAPPING` в базе данных пользователей (`security5.fdb` или той, что установлена для вашей базы данных в файле `databases.conf`) — любой администратор операционной системы Windows (при условии использования сервером доверенной авторизации - `Win_Sspi`) без указания роли. При этом не важно, включен или выключен флаг `AUTO ADMIN MAPPING` в самой базе данных.

Непривилегированные пользователи могут использовать только оператор `ALTER USER` для изменения собственной учётной записи.

Для создания новой учетной записи пользователя используется следующий синтаксис:

```
CREATE USER <логин> PASSWORD '<пароль>'
    [FIRSTNAME '<имя пользователя>']
    [MIDDLENAME '<отчество пользователя>']
    [LASTNAME '<фамилия пользователя>']
    [ACTIVE | INACTIVE]
    [USING PLUGIN 'имя плагина']
    [TAGS (<атрибут> [, <атрибут> ...] )]
    [GRANT ADMIN ROLE]

<атрибут> ::= <имя атрибута> = 'строковое значение'
```

Пользователь должен отсутствовать в текущей базе данных безопасности РЕД Базы Данных иначе будет выдано соответствующее сообщение об ошибке.

Имя пользователя может состоять максимум из 63 символов. Начиная с РЕД Базы Данных 3 имена пользователей подчиняются общему правилу наименования идентификаторов объектов метаданных. Таким образом, пользователь с именем `"Alex"` и с именем `"ALEX"` будут разными пользователями.

Предложение `PASSWORD` задаёт пароль пользователя. Максимальная длина пароля зависит от плагина проверки подлинности (`AuthServer`) и плагина управления пользователями (`UserManager`). Для

плагина SRP эффективная длина пароля ограничена 20 байтами. Для плагина `Legacy_UserManager` максимальная длина пароля равна 8 байт.

Необязательные предложения `FIRSTNAME, MIDDLENAME` и `LASTNAME` задают дополнительные атрибуты пользователя, такие как имя пользователя, отчество и фамилия соответственно.

Кроме того можно задать неограниченное количество пользовательских атрибутов с помощью необязательного предложения `TAGS`.

Если при создании учётной записи будет указан атрибут `INACTIVE`, то пользователь будет создан в "неактивном состоянии", т.е. подключиться с его учётной записью будет невозможно. При указании атрибута `ACTIVE` пользователь будет создан в активном состоянии (по умолчанию).

С опцией `GRANT ADMIN ROLE` создаётся новый пользователь с правами роли `RDB$ADMIN` в базе данных пользователей (`security5.fdb`). Это позволяет ему управлять учётными записями пользователей, но не дает ему специальных полномочий в обычных базах данных.

Необязательное предложение `USING PLUGIN` позволяет явно указывать какой плагин управления пользователями будет использован. Допустимыми являются только значения, перечисленные в параметре `UserManager`.

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

Для изменения существующей учетной записи пользователя используется следующий синтаксис:

```
ALTER {USER <логин> | CURRENT USER}
{
    [SET]
    [PASSWORD '<пароль>']
    [FIRSTNAME '<имя пользователя>']
    [MIDDLENAME '<отчество пользователя>']
    [LASTNAME '<фамилия пользователя>']
    [ACTIVE | INACTIVE]
    [TAGS (<атрибут>|DROP <имя атрибута> [, <атрибут>|DROP <имя атрибута>...] )]
}
[USING PLUGIN 'имя плагина']
[{GRANT | REVOKE} ADMIN ROLE];

<атрибут> ::= <имя атрибута> = 'строковое значение'
```

В операторе `ALTER USER` должно присутствовать хотя бы одно из необязательных предложений.

Это единственный оператор управления учётными записями, который может также использоваться непривилегированными пользователями для изменения их собственных учетных записей, однако это не относится к опциям `GRANT/REVOKE ADMIN ROLE` и атрибуту `ACTIVE/INACTIVE` для изменения которых, необходимы административные привилегии.

> Если предложение `USING PLUGIN` не указано, то при изменении атрибутов пользователя они сами сменяется у соответствующих пользователей в плагинах из списка параметра `DefaultUserManagers`.
>
> Если в каком-либо плагине из списка нет пользователя, то он добавляется, но только если среди изменяемых атрибутов есть пароль.

Для удаления существующей учетной записи пользователя используется следующий синтаксис:

```
DROP USER <логин>
[USING PLUGIN 'имя плагина'];
```

Если предложение USING PLUGIN не указано, то при удалении пользователя он сам удаляется из всех плагинов из списка параметра DefaultUserManagers.

9.4.2 Блокирование сеанса пользователя

Начиная с версии РЕД Базы Данных 5.0.3 оператор недоступен.

Оператор SUSPEND позволяет заблокировать сеансы пользователей и ролей:

```
SUSPEND { [USER <список пользователей>] | [ROLE <список ролей>] }
[DISCONNECT] | [PERMANENT];

<список пользователей> ::= "<имя пользователя 1>" [, "<имя пользователя 2>" ...]
<список ролей> ::= "<имя роли 1>" [, "<имя роли 2>" ...]
```

Заблокировать сеанс для списка пользователей или ролей может только SYSDBA, владелец базы данных, пользователь с ролью RDB$ADMIN. Заблокировать собственный сеанс может любой пользователь.

Заблокированный пользователь будет получать ошибку "Connection suspended" на все запросы, кроме повторного подключения к базе данных, отключения от базы данных или создания нового подключения к базе данных.

```
SUSPEND USER TEST_USER
```

Параметр конфигурации ConnectionSuspendTimeout позволяет установить для всего сервера или отдельно для базы данных количество секунд, по истечении которых сессия бездействующего пользователя будет заблокирована. Изменить значение параметра для текущего подключения можно с помощью isc_dbp_suspend_timeout. Установить таймаут больше определенного конфигурацией может только SYSDBA, владелец базы данных, пользователь с ролью RDB$ADMIN.

Для возобновления доступа необходимо использовать API интерфейса IAttachment функцию reconnect() или isc_reconnect(), передавая все необходимые для повторной авторизации данные. В случае использования доверенной аутентификации, указывать авторизационные данные не требуется.

Пример использования функции при возобновлении доступа в утилите ISQL:

```
RECONNECT [ [PASSWORD '<пароль>']
          | [CERTIFICATE '<алиас_сертификата>' [PIN <пароль_закрытого_ключа>] ]
        ];
```

Для возобновления выполнения транзакций/запросов заблокированный пользователь должен повторно ввести свои авторизационные данные. Это можно сделать с помощью оператора RECONNECT. В случае успешного ввода пароля, все незавершенные до момента блокировки транзакции будут продолженны.

Если пользовательское подключение закрыто вместо блокировки или после неё, то все незавершенные транзакции отменятся. Для возобновления работы, пользователю необходимо выполнить подключение к базе данных со всеми своими авторизационными данными. Это можно сделать с помощью

оператора CONNECT.

9.4.3 Безопасная парольная аутентификация (SRP)

РЕД База Данных 5.0 поддерживает метод аутентификации пользователей, реализованный в качестве плагина по умолчанию - Secure Remote Password (SRP) Protocol.

В результате работы данного протокола обе стороны получают длинный секретный ключ, проверяемый на соответствие между сторонами после получения. В случаях, когда помимо аутентификации необходимо шифрование данных, SRP предоставляет более надёжные, чем SSH, и более быстрые, чем протокол Диффи-Хеллмана, средства для достижения этой цели. Он также не зависит от третьих лиц, в отличие от Kerberos.

SSH протокол требует предварительного обмена ключами между сервером и клиентом, когда открытый ключ располагается на сервере. SRP не нуждается в этом. От клиента требуется только логин и пароль. Все обмены происходят, когда соединение установлено.

Кроме того, SRP устойчив к атаке посредника (man-in-the-middle).

Для того, чтобы пользователь РЕД Базы Данных смог пройти парольную аутентификацию, он должен быть предварительно создан с помощью плагина управления пользователями Srp:

```
CREATE USER test PASSWORD 'pass'
USING PLUGIN Srp;
```

Для этого в файле конфигурации firebird.conf параметр UserManager должен содержать значение Srp в списке:

```
UserManager = Srp, Legacy_UserManager
```

Данные о пользователях, созданных с помощью Srp плагина, хранятся в базе данных безопасности (security5.fdb) в таблице PLG$SRP (см. %s).

Менеджер пользователей Srp хэширует пароль алгоритмом SHA-1, при этом эффективная длина пароля ограничена 20 байтами. На длину пароля нет ограничения в 20 байт и он может быть использован. хэши различных паролей, длина которых более 20 байт, тоже различны. Предел эффективности наступает из-за ограниченной длины хэша в SHA1 равном 20 байт или 160 бит. Рано или поздно найдётся более короткий пароль с тем же хэшем с помощью атаки Brute Force. Именно поэтому часто говорят, что эффективная длина пароля для алгоритма SHA1 составляет 20 байт.

Для работы данного метода необходимо, чтобы в файле конфигурации firebird.conf параметр AuthServer в списке значений содержал метод аутентификации Srp. Кроме того, этот метод должен присутствовать и в списке методов клиентской стороны - в параметре AuthClient.

```
AuthServer = Srp
AuthClient = Srp
```

Для аутентификации в режиме Srp необходимо предъявить имя пользователя и пароль:

```
isql localhost:d:\test.fdb -user test -password pass
```

9.4.4 Традиционная (Legacy_Auth) аутентификация

Традиционная аутентификация подразумевает использование пароля в качестве единственного фактора аутентификации. Пароль может передаваться серверу в виде хэша или в открытом виде. Шифрование пароля происходит по алгоритму LEGACY.

Если вы собираетесь использовать данный метод аутентификации, в файле конфигурации `firebird.conf` выставите значения следующих параметров:

```
UserManager = Legacy_UserManager
WireCrypt = Enabled
AuthServer = Legacy_Auth, Srp, Win_Sspi
AuthClient = Legacy_Auth, Srp, Win_Sspi
```

Плагин, отвечающий за `Legacy_Auth` аутентификацию, не предоставляет ключа шифрования трафика. Поэтому следует отключить обязательное (`Required`) шифрование сессий через параметр `WireCrypt` в конфигурации сервера, т.е. выставить значение `Enabled` или `Disabled`.

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

```
isql localhost:d:\test.fdb -user testuser -password testpass
```

В этом методе аутентификации учитывается только первые 8 символов любого пароля.

Для того, чтобы пользователь РЕД Базы Данных смог пройти традиционную аутентификацию, он должен быть предварительно создан с помощью плагина управления пользователями `Legacy_UserManager`:

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

Данные о пользователях, созданных в традиционном режиме, хранятся в базе данных безопасности (`security5.fdb`) в таблице `PLG$USERS` (см. `%s`).

9.4.5 Доверительная (Win_Sspi) аутентификация

В доверительном режиме аутентификации используется система безопасности операционной системы. В операционных системах семейства Windows NT можно пользоваться учётными записями ОС. Для этого необходимо, чтобы в файле конфигурации `firebird.conf` параметр `AuthServer` в списке значений содержал метод аутентификации `Win_Sspi`. Кроме того, этот метод должен присутствовать и в списке методов клиентской стороны - в параметре `AuthClient`. Также для использования доверительной аутентификации следует отключить обязательное (`Required`) шифрование соединений (параметр `WireCrypt`), поскольку плагин, реализующий `Win_Sspi`, не предоставляет ключ шифрования. Для этого достаточно выставить значение `Enabled` или `Disabled`.

```
AuthServer = Win_Sspi, Srp
AuthClient = Win_Sspi, Srp
WireCrypt = Enabled
```

До РЕД Базы Данных 3 при включенной доверительной аутентификации, пользователи прошедшие проверку по умолчанию автоматически отображались в `CURRENT_USER`. В версии 5.0 отображение должно быть сделано явно для систем с несколькими базами данных безопасности и включенной доверительной аутентификацией.

Отображение для включения использования доверительной аутентификации Windows во всех базах данных, которые используют текущую базу данных безопасности:

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

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
TO USER;
```

В этом режиме при подключении к серверу РЕД Базы Данных не требуется предъявлять имя пользователя и пароль. Если пользователь локального компьютера подключается к серверу, работающему на том же компьютере, то он получает роль PUBLIC.

При сетевом соединении происходит проверка принадлежности пользователя домену, в состав которого входит компьютер с работающим сервером БД. Если пользователь не является доменным, то он не имеет прав на подключение к серверу.

Для того, чтобы узнать, с каким именем пользователя и паролем вы подключились в режиме доверительной аутентификации, можно выполнить следующий запрос:

```
select CURRENT_USER from rdb$database;
-------------------------------------
domain\administrator
```

То есть подключился пользователь с именем administrator, который является членом домена domain.

Чтобы при подключении доверенного пользователя не указывать никакой дополнительной информации о роли, существует оператор SET TRUSTED ROLE, который включает доступ доверенной роли.

Администраторы операционной системы Windows автоматически не получают права SYSDBA при подключении к базе данных. Имеют ли администраторы автоматические права SYSDBA зависит от установки значения флага AUTO ADMIN MAPPING. После успешного "auto admin" подключения текущей ролью будет являться RDB$ADMIN.

Оператор ALTER ROLE разрешает или запрещает автоматическое предоставление роли RDB$ADMIN администраторам Windows в текущей базе данных, если используется доверительная авторизация. По умолчанию автоматическое предоставление роли RDB$ADMIN отключено.

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

Оператор ALTER ROLE является упрощённым видом оператора создания отображения предопределённой группы DOMAIN_ANY_RID_ADMINS на роль RDB$ADMIN.

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

В обычных базах данных статус AUTO ADMIN MAPPING проверяется только во время подключения. Если Администратор имеет роль RDB$ADMIN потому, что произошло автоматическое отображение во время входа, то он будет удерживать эту роль на протяжении всей сессии, даже если он или кто-то другой в это же время выключает автоматическое отображение. Точно также, включение AUTO ADMIN MAPPING не изменит текущую роль в RDB$ADMIN для администраторов, которые уже подключились.

Рекомендуется явно указывать на то, что ожидается доверительная аутентификация, используя для этого ключ – tr. Например, если в системе установлены соответствующие значения переменных окружения ISC_USER и ISC_PASSWORD, и не будет указан ключ – tr при аутентификации, то вместо контекста безопасности пользователя на сервер будут переданы имя пользователя и пароль, соответствующие переменным окружения, как при традиционной аутентификации, что приведет к ошибке, так как на сервере ожидается доверительная аутентификация.

При доверительной аутентификации права на доступ и операции над объектами баз данных могут быть назначены пользователю операционной системы, как обычному пользователю РЕД Базы Данных.

При доверительной аутентификации возможен следующий вариант соединение с БД:

```
isql localhost:d:\test.fdb
```

9.4.6 GostPassword метод аутентификации

Метод GostPassword обеспечивает аутентификацию с использованием алгоритмов шифрования из криптографического плагина. При аутентификации все данные пользователя, кроме имени, передаются только в зашифрованном виде. Для работы данного метода необходимо наличие криптопровайдера и криптоплагина. Криптоплагин входит в поставку СУБД и называется Crypto_API.

Для использования этого режима аутентификации следует в файле конфигурации firebird.conf добавить в список значений параметра AuthServer метод аутентификации GostPassword. Кроме того, этот метод должен присутствовать и в списке методов клиентской стороны - в параметре AuthClient. Также в конфигурационном файле сервера и клиента необходимо указать используемый криптоплагин CryptoPlugin = Crypto_API (по умолчанию).

```
AuthServer = GostPassword, Srp
AuthClient = GostPassword, Srp
UserManager = GostPassword_Manager
CryptoPlugin = Crypto_API
```

Для того, чтобы пользователь РЕД Базы Данных смог пройти GostPassword-аутентификацию, он должен быть предварительно создан с помощью плагина управления пользователями GostPassword_Manager:

```
CREATE USER test PASSWORD 'pass'
USING PLUGIN GostPassword_Manager;
```

Для того, чтобы SYSDBA мог пройти GostPassword-аутентификацию, необходимо создать нового пользователя с именем SYSDBA, используя плагин GostPassword_Manager.

Данные о пользователях, созданных этим плагином, хранятся в базе данных безопасности (security5.fdb) в таблице PLG$MF (см. %s). Там хранится не только хэш пароля, но и название алгоритма, с помощью которого этот хэш получен.

Для GostPassword-аутентификации необходимо предъявить имя пользователя и пароль, например:

```
isql localhost:d:\test.fdb -user test -password pass;
```

9.4.7 Аутентификация по сертификату (Certificate)

РЕД База Данных 5.0 поддерживает метод аутентификации по сертификату с использованием алгоритмов шифрования из криптографического плагина. Для работы данного метода необходимо наличие криптопровайдера и криптоплагина. Криптоплагин входит в поставку СУБД и называется Crypto_API.

Для того, чтобы пользователь РЕД Базы Данных смог пройти аутентификацию по сертификату, в файле конфигурации firebird.conf параметр AuthServer должен содержать значение Certificate

в списке. Кроме того, этот метод должен присутствовать и в списке методов клиентской стороны - в параметре `AuthClient`. Также в конфигурационном файле сервера и клиента необходимо указать используемый криптоплагин `CryptoPlugin = Crypto_API` (по умолчанию).

```
AuthServer = Certificate, Srp
AuthClient = Certificate, Srp
WireCrypt = Disabled
CryptoPlugin = Crypto_API
```

Следует отключить обязательное (`Required`) шифрование соединений, поскольку плагин, реализующий `Certificate`, не предоставляет ключ шифрования. Для этого достаточно выставить значение параметра `WireCrypt` равным `Enabled` или `Disabled`.

Данные о таком пользователе не хранятся в базе данных безопасности (`security5.fdb`).

При подключении необходимо предъявить алиас сертификата и пароль закрытого ключа - PIN (если установлен). Контейнер с набором ключей и сертификат создаются заранее. Например:

```
isql localhost:d:\test.fdb -certificate "TESTUSER,Федеральная служба судебных
приставов,120025bec5106c2a6b1143bac900000025bec5" -pin 123456
```

Алиас сертификата представляет собой строку следующего вида:

```
<алиас сертификата> ::= "SubjectCN,IssuerCN,SerialNumber"
```

где `SubjectCN` – имя владельца сертификата, `IssuerCN` – название издателя сертификата, `SerialNumber` – серийный номер сертификата в шестнадцатеричном виде.

Таблица 9.2 — Параметры конфигурационного файла, имеющие отношение к аутентификации по сертификату

Параметр	Возможное значение	Комментарий
`AuthServer`, `AuthClient`	`Srp, Win_Sspi, Legacy_Auth, Gss, GostPassword, Certificate`	Параметр `AuthServer` - набор методов аутентификации, разрешенных на сервере. Параметр `AuthClient` - набор методов аутентификации, поддерживаемых клиентом.
`CryptoPlugin`	`Crypto_API`	Имя библиотеки криптопровайдера (расположена в каталоге `plugins` сервера)
`ServerCertificate`	`<алиас сертификата>`	Задаёт алиас сертификата, которым сервер будет удостоверять свою подлинность клиенту.
`CertUsernameDN`	`<атрибут>`	Атрибут сертификата, из которого будет извлекаться имя его владельца. По умолчанию `CN`.
`CertUsernamePattern`	`<регулярное выражение>`	Регулярное выражение, применяемое к атрибуту сертификата с именем пользователя для извлечения самого этого имени. Использует синтаксис SQL, по умолчанию не задано.
`VerifyCertificateChain`	`0 | 1`	Задаёт / отключает проверку цепочки сертификации пользовательского сертификата.

(разрыв таблицы)

(разрыв таблицы)

Параметр	Возможное значение	Комментарий
TrustedCertificate	<алиас сертификата>	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.

9.4.8 Проверка сертификата сервера (VerifyServer)

РЕД База Данных 5.0 поддерживает плагин проверки сертификата сервера с использованием алгоритмов шифрования из криптографического плагина. Для работы данного плагина необходимо наличие криптопровайдера и криптоплагина. Криптоплагин входит в поставку СУБД и называется Crypto_API.

Данный плагин не имеет смысла использовать без политик безопасности (п. 9.4.14), который позволяет не завершать процесс аутентификации на первом успешном методе, а проверять все методы аутентификации и только потом принимать решение о том, давать ли доступ пользователю или нет.

Для того, чтобы пользователь РЕД Базы Данных смог проверить сертификат сервера, в файле конфигурации firebird.conf должны быть указаны параметры ServerCertificate и ServerPrivatePin, а параметр AuthServer должен содержать значение VerifyServer в списке. Кроме того, этот метод должен присутствовать и в списке методов клиентской стороны - в параметре AuthClient. Также в конфигурационном файле сервера и клиента необходимо указать используемый криптоплагин CryptoPlugin = Crypto_API (по умолчанию).

```
AuthServer = Srp, VerifyServer
AuthClient = Srp, VerifyServer
WireCrypt = Disabled
CryptoPlugin = Crypto_API
```

Данные плагина не хранятся в базе данных безопасности (security5.fdb).

При подключении необходимо указывать параметр для проверки сертификата сервера. Например:

```
isql localhost:d:\test.fdb -verify <алиас сертификата>
```

9.4.9 Аутентификация доверенным пользователем

В firebird.conf добавлен параметр конфигурации TrustedUser для аутентификации по паролю с именем другого пользователя. По умолчанию пустой.

При соединении с базой данных кроме имени пользователя (isc_dpb_user_name) можно указать эффективный логин (isc_dpb_effective_login). В isql для этого добавлен ключ -l.

Если задан эффективный логин, то после успешной аутентификации любого плагина проверяется задан ли параметр TrustedUser:

- Если не задан - ошибка аутентификации: попытка подмены логина с отключенной опцией.
- Если параметр задан, но логин пользователя не совпадает с доверенным - тоже ошибка: попытка подмены логина не доверенным пользователем.
- Если параметр задан и логин пользователя совпадает с доверенным, то при подключении к базе данных его имя заменяется на указанный им эффективный логин.

Для ядра СУБД это подключение будет выглядеть как обычное, информация о подмене логина

до него не доходит.

Работает для всех плагинов аутентификации.

9.4.10 Доверенная аутентификация через механизм GSSAPI (Gss)

Доверенная аутентификация через механизм `GSSAPI` доступна только в промышленной редакции СУБД РЕД База Данных. Подробнее различия функционала редакций описаны в разделе *Редакции СУБД РЕД База Данных 5.0*.

РЕД База Данных 5.0 поддерживает Kerberos аутентификацию через GSSAPI (Generic Security Services Application Programming Interface). GSSAPI - это абстрактный уровень над Kerberos 5, предназначенный для решения проблемы несовместимости схожих сервисов безопасности. GSSAPI обеспечивает автоматическую аутентификацию (single sign-on), для систем, которые её поддерживают.

Принцип работы:

- Для аутентификации клиента используется контекст безопасности, полученный при входе в систему или из других источников типа kinit.
- Из контекста безопасности клиент извлекает токен, который отправляется на сервер СУБД.
- Сервер СУБД проверяет токен через GSSAPI (по умолчанию для проверки используется механизм Kerberos).
- Если токен проходит проверку, сервер извлекает из него имя пользователя, которое будет использовано для подключения к базе.

К достоинствам данной аутентификации можно отнести преимущества технологии единого входа (SSO):

- ввод пользователем своих учётных данных только один раз;
- отсутствие ввода пароля;
- безопасность;
- централизованное хранение учётных данных пользователей.

При работе с Kerberos GSSAPI использует стандартные учётные записи в формате `servicename/hostname@realm`.

Имя пользователя при такой аутентификации формируется из имени учетной записи в базе данных `KDC`.

Такой вид аутентификации работает только при передаче данных через сетевой протокол.

Если пользователь указал логин, то данный метод аутентификации будет проигнорирован сервером.

Для работы данного метода аутентификации необходимо, чтобы в файле конфигурации `firebird.conf` параметр `AuthServer` в списке значений содержал метод аутентификации `Gss`. Кроме того, этот метод должен присутствовать и в списке методов клиентской стороны - в параметре `AuthClient`. Так как плагин, отвечающий за `GSSAPI` аутентификацию, не предоставляет ключа шифрования трафика, следует отключить обязательное (`Required`) шифрование сессий через параметр `WireCrypt` в конфигурации сервера, т.е. выставить значение `Enabled` или `Disabled`.

Если вы собираетесь использовать данный метод аутентификации, в файле конфигурации

`firebird.conf` выставите значения следующих параметров:

```
AuthServer = Gss, Srp
AuthClient = Gss, Srp
WireCrypt = Disabled
GssServerKeyfile =
GssServiceName =
GssHostName =
GSSLibrary =
```

где

- `GssServerKeyfile` - путь до файла, содержащий долговременный ключ сервиса, который будет использовать СУБД для аутентификации в Kerberos. Этот ключевой файл создаётся в центре распределения ключей (KDC), например командой утилитой в Active Directory;
- `GssServiceName` - имя сервиса (по умолчанию «`rdb_server`»), созданное на сервере Kerberos для аутентификации СУБД;
- `GssHostName` - DNS-адрес сервера СУБД (например «`rdb.example.com`»);
- `GSSLibrary` - динамическая библиотека GSSAPI (`libgssapi_krb5.so`). Поддерживается также библиотека `libvas-gssapi.so` от One Identity Authentication Services. При её использовании СУБД после аутентификации определяет группы, назначенные пользователю в домене, и назначает ему одноимённые роли, существующие в базе данных.

В РЕД Базе Данных 3 при включенной доверенной аутентификации пользователи, прошедшие проверку, по умолчанию автоматически отображались в `CURRENT_USER`. В РЕД Базе Данных 5 отображение должно быть создано явно для систем с несколькими базами данных безопасности и включенной доверенной аутентификацией.

Отображение для использования доверительной аутентификации во всех базах данных, которые используют текущую базу данных безопасности:

```
CREATE GLOBAL MAPPING GSS_AUTH
USING PLUGIN GSS
FROM ANY USER
TO USER;
```

В этом режиме при подключении к серверу РЕД Базы Данных не требуется предъявлять имя пользователя и пароль. Если пользователь локального компьютера подключается к серверу, работающему на том же компьютере, то он получает роль `PUBLIC`.

При доверенной аутентификации возможен следующий вариант соединение с БД:

```
isql localhost:d:\test.fdb
```

9.4.11 Доверенная аутентификации для выполнения Execute Statement On External без указания логина и пароля (ExtAuth)

Доверенная аутентификации для выполнения `Execute Statement On External` без указания логина и пароля доступна только в промышленной редакции СУБД РЕД База Данных. Подробнее различия функционала редакций описаны в разделе *Редакции СУБД РЕД База Данных 5.0*.

Для выполнения оператора EXECUTE STATEMENT ON EXTERNAL, если внешний источник данных находится на другом сервере, предложения AS USER <имя пользователя> и PASSWORD <пароль> являются обязательными.

```
EXECUTE STATEMENT 'SELECT * FROM RDB$DATABASE'
ON EXTERNAL 'server:db1' AS USER 'MYUSER' PASSWORD 'mypassword'
```

Значения имени пользователя и пароля передаются в открытой форме, что небезопасно. Например, если ESOE (сокр. от EXECUTE STATEMENT ON EXTERNAL) вызывается из кода хранимой процедуры, подключенные пользователи могут видеть пароль.

Для безопасного подключения в РЕД Базе Данных был разработан плагин аутентификации ExtAuth специально для ESOE, который устанавливает доверительную связь между серверами РЕД Базы Данных и выполняет аутентификацию ESOE без логина и пароля:

```
EXECUTE STATEMENT 'SELECT * FROM RDB$DATABASE'
ON EXTERNAL 'server:db1';
```

Для использования данной возможности следует в файле конфигурации firebird.conf добавить в список значений параметров AuthServer и AuthClient метод аутентификации ExtAuth на всех серверах, которые будут "доверять" друг другу.

```
AuthServer = Srp, ExtAuth
AuthClient = Srp, ExtAuth
```

Затем необходимо сгенерировать ключевой файл для плагина. Этот ключ должен быть размещен на всех серверах РЕД Базы Данных, которые должны доверять друг другу.

Чтобы сгенерировать ключевой файл ExtAuth.conf, запустите исполняемый файл extauth_keygen. Ключевой файл содержит три параметра:

- Key - сам ключ;
- IgnoreLogin - игнорировать явное указание логина в операторе ESOE (по умолчанию логин не игнорируется). Если значение параметра No и указан логин, то плагин ExtAuth перестает действовать.
- IgnorePassword - игнорировать явное указание пароля в операторе ESOE (по умолчанию пароль не игнорируется). Если значение параметра No и указан пароль, то плагин ExtAuth перестает действовать.

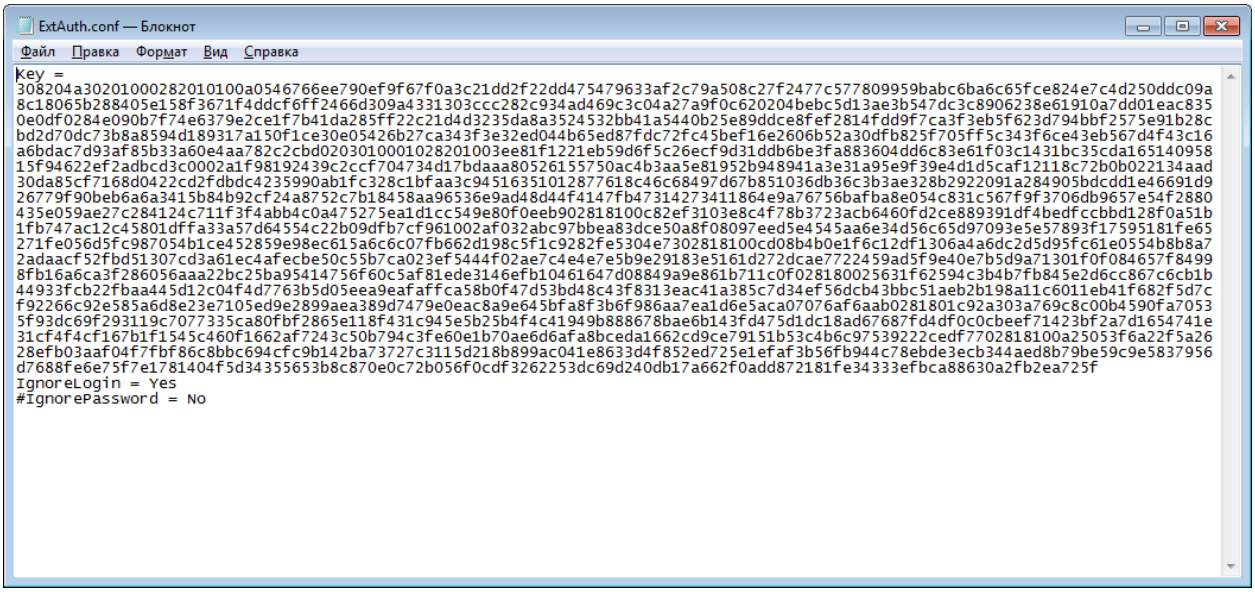


Рисунок 9.1 — Содержимое файла `ExtAuth.conf`

Потом скопируйте файл ключа на все доверенные сервера РЕД Базы Данных в папку `plugins`. Ключ, созданный на Windows, можно использовать в Linux, и наоборот.

> Другое имя ключевого файла и его расположение не допускается.

Чтобы использовать плагин аутентификации внутри конкретной базы данных, необходимо создать отображение между пользователями плагина `ExtAuth` и обычными пользователями РЕД Базы Данных.

Например, чтобы запустить ESOE от имени пользователя `MYUSER`:

```
EXECUTE STATEMENT 'SELECT * FROM RDB$DATABASE'
ON EXTERNAL 'server:db1';
```

Нужно сопоставить пользователя `MYUSER` с фактическим пользователем в целевой базе данных `db1`. Давайте предположим, что есть пользователь `MYUSER2` в целевой базе данных, в этом случае нужно создать следующее отображение в целевой базе данных `db1`:

```
CREATE MAPPING cluster_auth USING PLUGIN extauth
FROM USER MYUSER TO user MYUSER2;
```

Оба пользователя должны существовать.

Можно создать отображение между пользователями для всех баз данных на сервере.

9.4.12 Аутентификация по протоколу LDAP

> Аутентификация по протоколу LDAP доступна в стандартной редакции СУБД РЕД База Данных. Подробнее различия функционала редакций описаны в разделе *Редакции СУБД РЕД База Данных 5.0*.

РЕД База Данных поддерживает возможность аутентификации пользователей сервера баз дан-

ных с использованием службы каталогов через протокол LDAP. Как известно, учётные данные пользователей хранятся в БД безопасности – security5.fdb. Также существует возможность добавить дополнительный источник учётной информации – службу каталогов на основе сервера OpenLDAP и Active Directory. При этом LDAP используется именно как дополнение к традиционной схеме безопасности сервера. При проверке пользовательских учётных данных, если заданы параметры подключения к LDAP, сервер для всех пользователей, кроме SYSDBA, сначала пытается проверить наличие учетной записи пользователя в каталоге LDAP и если он там не найден, то выполняется также поиск в БД безопасности security5.fdb. Для SYSDBA поиск выполняется только в БД безопасности.

Если пользователь не найден в LDAP-сервере(не задан адрес LDAP-сервера) и проверка в базе данных безопасности прошла неуспешно, выдается сообщение об ошибке аутентификации.

С точки зрения конечного пользователя всё это работает совершенно прозрачно и ему не нужно выполнять никаких дополнительных действий.

Параметры конфигурации LDAP

Для аутентификации по протоколу LDAP в firebird.conf должны быть заданы настройки подключения к серверу каталогов.

Таблица 9.3 — Параметры конфигурационного файла, имеющие отношение к LDAP

Параметр	Возможное значение	Комментарий
LDAPServer		Адрес сервера LDAP (IP-адрес или символьное имя)
LDAPEncryption	None|SSL|TLS	Тип шифрования, используемый при подключении к LDAP. Он может принимать три значения: • None – шифрование отсутствует, подключение 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. Сертификат сервера LDAP будет проверяться, если параметр VerifyLdapServer не отключен.
VerifyLdapServer	true|false	Если параметр включен, то при SSL/TLS-соединениях будет выполняться проверка сертификата LDAP-сервера со стороны сервера РЕД Базы Данных. Если сертификат не проходит проверку, соединение разрывается. Верификация сертификата выполняется аналогично проверке пользовательского сертификата. Если параметр отключен, проверка сертификата LDAP-сервера не выполняется.
LDAPUserDN	uid=rdb, ou=people, dc=example, dc=com	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. Если параметр не задан, сервер будет делать bind к LDAP с именем и паролем пользователя, указанным клиентом, чтобы пройти аутентификацию.

(разрыв таблицы)

(разрыв таблицы)

Параметр	Возможное значение	Комментарий
LDAPPassword		Пароль пользователя, определенного в атрибуте LDAPUserDN, от имени которого сервер будет подключаться к LDAP для аутентификации других пользователей.
LDAPUserBase	ou=people, dc=example, dc=com	Ветка в LDAP каталоге в виде DN, 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.
LDAPUserPrefix	uid	Название атрибута, в котором хранится имя пользователя в его DN в LDAP. Для OpenLDAP обычно uid, для AD - cn. В основном используется, когда не задан LDAPUserDN. Если параметр LDAPUserPrefix не задан, клиентская библиотека fbclient будет шифровать пароль пользователя и для аутентификации будут использоваться пароли из атрибутов: rdbPassword; rdbSrpVerifier; rdbSrpSalt; rdbSecurePassword; rdbPasswordAlgorithm. Если параметр LDAPUserPrefix задан, то пароль не шифруется и отправляется на сервер открытым, сервер "биндится" к LDAP с этим именем пользователя и этим паролем.
LDAPUserFilter	uid=%u	Фильтр для поиска учетных записей пользователей. В значении этого параметра используется шаблон %u, который при поиске будет заменен на имя пользователя, предъявленное при аутентификации. По умолчанию параметр имеет значение uid=%u, что типично для схем каталога на базе OpenLDAP. В схеме, поддерживаемой Active Directory, обычно используется фильтр -(objectClass=user)(cn=%u).
LDAPGroupBase	ou=group, dc=example, dc=com	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. Указывается в виде DN.
LDAPMembershipFilter	member=%d	Фильтр, который будет использован при поиске в LDAP групп, к которым принадлежит пользователь. Существует три основные схемы, используемые в LDAP для указания членства в группах. В соответствии с используемой схемой нужно формировать фильтр. В нём в качестве имени предполагаемого пользователя указывается шаблон %u, а в качестве DN пользователя указывается %d. Если указано пустое значение, принадлежность пользователя к группам не определяется.

(разрыв таблицы)

(разрыв таблицы)

Параметр	Возможное значение	Комментарий
LDAPUserCertificate		Атрибут LDAP, в котором будет храниться сертификат пользователя. Если данный параметр задан, то сертификат, предъявленный пользователем, должен соответствовать его сертификату в LDAP (если настроены параметры подключения к LDAP-серверу). Если этот параметр не задан, сертификат пользователя не будет проверяться на соответствие его сертификату из LDAP. Сертификат должен храниться в двоичном формате (DER) в атрибуте «userCertificate».
LDAPPasswordSync	rdbSrpVerifier; userPassword	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 СУБД РЕД База Данных; • rdbSrpVerifier — пароль (верификатор) SRP; • rdbSecurePassword — пароль по ГОСТ; • 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.
LDAPReadOnly	0|1	Заданный параметр может перевести LDAP-сервер в режим только для чтения (значение 1). Тогда параметр LDAPPasswordSync будет игнорироваться. По умолчанию используется значение 0.

Процесс аутентификации при передаче пароля в открытом виде

Пользователь при подключении к БД передаёт пароль в *открытом виде* в тэге isc_dpb_password. В файле конфигурации задан параметр LDAPServer.

Сервер для всех пользователей, кроме SYSDBA, сначала пытается проверить наличие учетной записи пользователя в каталоге LDAP. Пользователь будет аутентифицироваться в LDAP по его реальному логину методом bind одним из следующих способов:

1. Если задан параметр LDAPUserDN, то сначала сервер подключается к LDAP от этого общего пользователя. Затем он ищет DN реального пользователя и отключается от LDAP. Далее сервер делает bind с полным именем (DN) и паролем реального пользователя. Это позволяет распределять учетные записи пользователей по разным веткам LDAP.
2. Если LDAPUserDN не задан, то сервер сразу делает bind с именем и паролем реального пользователя.
3. Если задан параметр LDAPUserBase, то DN пользователя формируется, как LDAPUserPrefix + <имя пользователя> + LDAPUserBase. В этом случае пользователи должны находиться в одной ветке LDAP, указанной в LDAPUserBase;

4. Если параметр LDAPUserBase не задан, то сервер сначала попытается сделать bind с именем пользователя в том виде, в котором его передал клиент.
5. Если аутентифицироваться не получилось и задан параметр LDAPDomain, то будет выполнена попытка сделать bind используя UPN (UserPrincipalName) в виде <имя пользователя>@<значение LDAPDomain>. Такой способ работает для входа в ActiveDirectory.

Пример настройки для ActiveDirectory:

```
#Адрес сервера LDAP
LDAPServer=10.81.1.1

#Шифрование по SSL
LDAPEncryption=SSL

#Атрибут LDAP, по которому будет выполняться поиск пользователя, имя которого передает
клиент
LDAPUserPrefix=sAMAccountName

#Фильтр для поиска пользователя в каталоге LDAP
LDAPUserFilter=&(objectClass=user)(sAMAccountName=%u)

#Стартовая ветка для поиска групп пользователя (поиск выполняется рекурсивно по всем
вложенным веткам)
LDAPGroupBase=ou=groups,dc=example,dc=com

#Фильтр для поиска групп пользователя
LDAPMembershipFilter=member=%d

#Имя домена
LDAPDomain = example.com

#Отключение проверки сертификата сервера LDAP
VerifyLdapServer=0
```

Если пользователь не найден в LDAP, сервер пытается проверить пользователя через security5.fdb. Если он там найден, аутентификация выполняется по считанным из неё данным. Если не найден или найден, но его пароль не совпадает, то сообщается об ошибке аутентификации.

Если в firebird.conf заданы параметры LDAPServer и LDAPUserPrefix, клиентская библиотека fbclient шифровать пароль не будет.

Для SYSDBA поиск выполняется только в БД безопасности.

Процесс аутентификации при передаче пароля в зашифрованном виде

Пользователь при подключении к БД передаёт пароль в *зашифрованном виде* в тэге isc_dpb_password_enc. В файле конфигурации задан параметр LDAPServer.

Сервер для всех пользователей, кроме SYSDBA, сначала пытается проверить наличие учетной записи пользователя в каталоге LDAP. Сначала выполняется подключение к LDAP-серверу от имени общего пользователя LDAPUserDN (указывается обязательно). Затем на сервере происходит поиск нужного пользователя относительно указанной ветви LDAPUserBase. Если пользователь найден, то сверяются хэш его пароля в LDAP с хэшем предоставленного им пароля.

Пароль пользователя для Legasy/Srp/GostPassword-аутентификации хранится в LDAP в атри-

буте `rdbPassword/rdbSrpVerifier/rdbSecurePassword` в том же виде, в котором он сохраняется в `security5.fdb`.

Если пользователь не найден в `LDAP`, сервер пытается проверить пользователя через `security5.fdb`. Если он там найден, аутентификация выполняется по считанным из неё данным.

Если пользователь не найден в `LDAP` и не найден в `security5.fdb`, проверка пароля считается неуспешной.

Процесс аутентификации по сертификату

При аутентификации по сертификату сначала проверяется является ли пользовательский сертификат доверенным. Для этого он сравнивается с сертификатом, заданным в параметре конфигурации `TrustedCertificate`. Если сертификат не доверенный, то он верифицируется и после этого проверяется, задан ли параметр конфигурации `LDAPUserCertificate`. Если он не задан, проверка сертификата считается успешной. Если параметр задан, то выполняется подключение к `LDAP`-серверу и скачивание оттуда сертификата пользователя. Затем полученный сертификат сравнивается с сертификатом, предоставленным пользователем. Если они одинаковы, проверка сертификата считается успешной. Если не удалось подключиться к `LDAP`, получить оттуда сертификат пользователя или сертификаты не совпадают, проверка сертификата считается неуспешной.

Если проверка сертификата через `LDAP` прошла успешно, имя пользователя, извлечённое из сертификата, может отличаться от имён пользователей, которые заданы другими факторами аутентификации.

Имя пользователя из сертификата (из секции `Subject`) извлекается с помощью двух параметров конфигурации.

- `CertUsernameDN` – содержит `DN` атрибута пользователя внутри сертификата. По умолчанию используется атрибут `CN`.
- `CertUsernamePattern` – задаёт регулярное выражение в синтаксисе `SQL`, которое извлекает подстроку из содержимого найденного атрибута. По умолчанию используется пустой шаблон, что означает использование содержимого атрибута целиком.

Если указанный атрибут не найден в сертификате или результатом применения к нему регулярного выражения оказалась пустая строка, возвращается ошибка.

Верификация сертификата пользователя выполняется в два этапа. На первом этапе проверяется наличие у пользователя доступа к закрытому ключу, соответствующему предъявленному сертификату. Это выполняется генерацией сессионного ключа по алгоритму Диффи-Хеллмана, в которой участвуют пары открытых и закрытых ключей клиента и сервера. На втором этапе выполняется проверка самого сертификата и его цепочки сертификации. Верификация может быть отключена параметром конфигурации `VerifyLdapServer`. Верификация считается неуспешной в следующих случаях:

- не удалось построить цепочку сертификации;
- любой сертификат из цепочки отозван;
- любой сертификат из цепочки просрочен;
- любой сертификат из цепочки не прошёл проверку подписи;
- любой сертификат из цепочки используется не по назначению;
- цепочка основана на недоверенном корневом центре сертификации;
- цепочка сертификации содержит цикл;
- цепочка сертификации построена не полностью;
- не удалось проверить статус отзыва для любого сертификата из цепочки.

При неудачной верификации пользователю сообщается об ошибке проверки фактора аутентификации, а в `firebird.log` записывается сообщение с информацией о владельце сертификата и типе произошедшей ошибки.

Информация о пользователях, получаемая из LDAP

После аутентификации через LDAP можно запросить информацию о пользователе с помощью функций LDAP_ATTR:

```
LDAP_ATTR(<имя атрибута> [, <имя пользователя> ])
```

Важно учитывать, что получить атрибуты пользователя, указанного в необязательном аргументе функции LDAP_ATTR, может только SYSDBA, владелец базы данных, пользователь с ролью RDB$ADMIN или пользователь, принадлежащий группе LDAP_ADMIN каталога LDAP.

Можно узнать где пользователь прошел проверку подлинности - через базу данных безопасности или в LDAP:

```
select RDB$GET_CONTEXT('AUTHDATA', 'AUTH_TYPE') from rdb$database;
```

и с помощью какого плагина аутентификации:

```
select RDB$GET_CONTEXT('AUTHDATA', 'AUTH_PLUGIN') from rdb$database;
```

Также можно узнать ФИО пользователя. При аутентификации через LDAP эта информация считывается из атрибута пользователя "CN":

```
select RDB$GET_CONTEXT('AUTHDATA', 'USER_FIRST_NAME') from rdb$database;
select RDB$GET_CONTEXT('AUTHDATA', 'USER_MIDDLE_NAME') from rdb$database;
select RDB$GET_CONTEXT('AUTHDATA', 'USER_LAST_NAME') from rdb$database;
```

Если заданы параметры LDAPGroupBase и LDAPMembershipFilter, то осуществляется поиск групп, к которым принадлежит пользователь в LDAP. После этого ему назначаются роли, соответствующие найденным группам пользователя из LDAP. Заполняются переменные LDAP_ROLES и LDAP_ROLES_DN - список ролей пользователя и DN ролей пользователя, полученных из каталога LDAP.

Список всех групп, которым принадлежит пользователь, можно получить с помощью функции LDAP_USER_GROUPS:

```
LDAP_USER_GROUPS(<имя пользователя> [, <фильтр списка групп> ])
```

Для получения полного списка групп пользователей каталога LDAP можно использовать функцию LDAP_GROUPS:

```
LDAP_GROUPS([ <фильтр списка групп> ])
```

Получение полного списка групп и списка групп для конкретного пользователя доступно только для SYSDBA, владельца базы данных, пользователю с ролью RDB$ADMIN или пользователю, принадлежащему группе LDAP_ADMIN каталога LDAP.

Результирующие списки групп можно отфильтровать во время запроса с помощью необязательных аргументов функций LDAP_GROUPS и LDAP_USER_GROUPS:

```
select LDAP_GROUPS('objectClass=posixGroup') from rdb$database;
select LDAP_USER_GROUPS('TEST-USER', 'objectClass=posixGroup') from rdb$database;
```

Изменение пароля в LDAP

Чтобы сменить или задать пароль пользователя в LDAP используется параметр конфигурации LDAPPasswordSync. В нём через «;» указываются пароли, которые необходимо сменить (по умолчанию – все возможные). При изменении пароля указанный пользователь сначала ищется в базе данных безопасности и, если он там найден, его пароль меняется. Затем пользователь ищется в LDAP (если задан адрес LDAP-сервера). В LDAP меняются следующие атрибуты:

- userPassword – пароль пользователя в Linux (записывается хэш SHA1 в кодировке BASE64).
- sambaLMPassword – LMHash для Samba.
- sambaNTPassword – MD4 хэш для Samba.
- rdbPassword – пароль пользователя на сервере БД, зашифрованный алгоритмом LEGACY, используемым при традиционной аутентификации.
- rdbLegacyHistory - история изменения паролей Legacy.
- rdbLegacyPasswordTime - время последнего изменения пароля Legacy
- rdbSecurePassword – пароль по ГОСТ пользователя на сервере БД, зашифрованный каким-либо алгоритмом из криптоплагина.
- rdbPasswordAlgorithm – алгоритм шифрования пароля по ГОСТ на сервере БД (в кодировке UTF-8).
- rdbPasswordHistory – история смены ГОСТ паролей.
- rdbPasswordTime – время последнего изменения пароля ГОСТ.
- rdbSrpVerifier - пароль (верификатор) SRP.
- rdbSrpSalt - соль SRP.
- rdbSrpHistory - история изменения паролей SRP.
- rdbSrpPasswordTime - время последнего изменения пароля SRP.

Если сменить какой-либо пароль не получилось, в firebird.log записывается сообщение об ошибке с указанием имени пользователя, атрибута и описания ошибки. Два связанных с паролем атрибута тоже не меняются. Например, пароль не получится сменить, если в схеме LDAP нет соответствующего ему атрибута. При ошибке смены одного из паролей выполняется попытка сменить остальные пароли.

Если атрибут, соответствующий паролю, у пользователя не задан, но он имеется в схеме LDAP, нужный атрибут создаётся.

Если пользователь найден, но хотя бы один из паролей не получилось задать, пользователю возвращается ошибка «error changing ldap password».

Если пользователь не найден ни в security5.fdb, ни в LDAP, возвращается ошибка «record not found for user».

Схемы LDAP

Так как сервер при работе с LDAP использует ряд атрибутов, не входящих в стандартные схемы, то для полноценной работы сервера необходимо добавить к конфигурации LDAP схему, аналогичную приведенной в Приложение B.

Чтобы сервер мог использовать эти атрибуты, администратор LDAP должен назначить пользователю класс rdbAuth. У пользователя, от имени которого сервер выполняет подключение к LDAP (LDAPUserDN) должны быть права на чтение всех этих атрибутов и на запись в атрибуты rdbPassword, rdbSrpVerifier, rdbSrpSalt, rdbSecurePassword, rdbPasswordAlgorithm, rdbLegacyHistory, rdbSrpHistory, rdbPasswordHistory, rdbLegacyPasswordTime, rdbSrpPasswordTime, rdbPasswordTime, rdbActive, rdbPolicy, rdbFailedCount, rdbAccessTime. При этом сервер сможет создавать нужные атрибуты запросами к LDAP.

Таблица 9.4 — Атрибуты пользователя LDAP

Атрибут	Описание
userPassword	Пароль пользователя, используемый обычно в UNIX-системах.
sambaLMPassword	Пароль пользователя, используемый SAMBA-протоколом.
sambaNTPassword	Пароль пользователя, используемый SAMBA-протоколом.
sambaPwdLastSet	Время последнего изменения атрибутов sambaLMPassword и sambaNTPassword в формате UNIX.
rdbPassword	Пароль для пользователя метода Legacy_Auth.
rdbSrpVerifier	Верификатор пользователя для протокола SRP.
rdbLegacyHistory	История изменения паролей Legacy.
rdbSrpHistory	История изменения паролей SRP.
rdbLegacyPasswordTime	Время последнего изменения пароля Legacy.
rdbSrpPasswordTime	Время последнего изменения пароля SRP.
rdbSrpSalt	Соль для аутентификации по протоколу SRP.
rdbSecurePassword	Пароль для пользователя метода Multifactor.
rdbPasswordAlgorithm	Алгоритм шифрования многофакторного пароля на сервере БД (в одировке UTF-8).
rdbPasswordHistory	История смены многофакторных паролей.
rdbActive	Атрибут показывает активен пользователь или заблокирован. Значение true, если пользователь активен и false, если пользователь заблокирован.
rdbPolicy	Название политики безопасности. Если у пользователя этот атрибут не найден, для него применяется политика безопасности по умолчанию (DEFAULT).
rdbPasswordTime	Время последней смены ГОСТ-пароля пользователем.
rdbFailedCount	Число неудачных попыток аутентификации с момента последнего успешного входа.
rdbAccessTime	Время, до которого вход пользователя на сервер запрещён.

9.4.13 Аутентификация по протоколу OpenIDConnect

Для аутентификации по протоколу OpenIDConnect в firebird.conf необходимо указать следующие настройки:

```
AuthServer = OpenIDConnect, Srp -- на стороне сервера
AuthClient = OpenIDConnect, Srp -- на стороне клиента
```

Для дальнейшей настройки необходимо получить публичные данные ключа провайдера (JSON Web Key - JWK). Это можно сделать по ссылке https://<провайдер>/.well-known/openid-configuration. Публичные ключи провайдера хранятся в jwks_uri.

В файле plugins.conf нужно указать использование OpenIDConnect и публичные данные ключа провайдера (или в plugins/OpenIDConnect.conf).

```
Plugin = OpenIDConnect {
  Module = $(dir_plugins)/OpenIDConnect
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
  Config = OpenIDConnect_config
}

Config = OpenIDConnect_config {
  jwks
  {
    jwk = <имя провайдера>
    {
      kid = "<идентификатор ключа>"
      alg = "<алгоритм шифрования>"
      kty = "<тип ключа>"
      use = "<использование открытого ключа>"
      e = "<экспонента RSA ключа>"
      n = "<модуль RSA ключа>"
      x5c = "<сертификат формата X.509>"
    }
  }
}
```

Описание параметров:

- `jwk` - имя провайдера, выдающего токен;
- `kid` - идентификатор ключа;
- `alg` - криптографический алгоритм;
- `kty` - тип ключа;
- `use` - использование открытого ключа;
- `e` - экспонента `RSA` ключа (обычно `AQAB (0x10001 = 65537)`); указывается в формате `base64url`;
- `n` - модуль `RSA` ключа; указывается в формате `base64url`;
- `x5c` - сертификат формата `X.509`, который можно указать вместо параметров `e` и `n`; указывается в формате `base64`, а не `base64url`.

Создайте отображение пользователей для аутентификация по протоколу `OpenIDConnect` во всех базах данных, которые используют текущую базу данных безопасности:

```
CREATE GLOBAL MAPPING OIDC_AUTH
USING PLUGIN OPENIDCONNECT
FROM ANY USER
TO USER;
```

Для аутентификация по протоколу `OpenIDConnect` необходимо предъявить токен. Токен представляет собой строку `base64url` в формате: `<header>.<payload>.<sign>`. `Header` содержит алгоритм токена. **На данный момент поддерживается только RS256**. `Payload` содержит следующие ключи: `exp` - срок действия токена в `UTC`; `sub` - пользователь, которому выдан токен; `iss` - провайдер, выдавший токен.

Передать токен можно через переменную окружения `RDB_OID_TOKEN` или с помощью утилиты `isql`, используя опцию `-token`:

```
isql <спецификация сервера> -user "<имя пользователя>" -token "<токен>";
```

Имя пользователя должно совпадать с пользователем, указанным в `sub` полезной части токена.

9.4.14 Политики безопасности

Общие сведения

Политики безопасности (политики учетных записей) позволяют контролировать следующие параметры безопасности системы:

- сложность пароля при его задании;
- количество предыдущих паролей, которые не должен повторять вновь заданный;
- срок действия пароля;
- количество допустимых неудачных попыток аутентификации;
- количество одновременно открытых сессий пользователя;
- продолжительность простоя пользователя до отключения;
- период времени неиспользования учетных записей пользователей;
- набор методов аутентификации, которые пользователь должен пройти.

Политики учетных записей создаются и хранятся в базе данных безопасности (`security5.fdb` или любой другой заданной) в таблице `PLG$POLICIES` (см. `%s`) или в LDAP. Информация о назначенных пользователям политиках хранится в базе данных безопасности в таблице `PLG$USER_POLICY`. Для контроля уникальности паролей хэши старых паролей хранятся в `PLG$PASSWD_HISTORY`. Дата и время создания пароля сохраняется в поле `PLG$PASSWD_TIME` в таблице соответствующего метода (`PLG$SRP, PLG$USERS, PLG$MF`).

Для удобного просмотра всех созданных политик существует псевдотаблица безопасности `SEC$POLICIES` (см. Приложение Б), аналогичная `PLG$POLICIES`.

Политики безопасности работают со всеми известными методами аутентификации.

Политики реализуется в виде плагина `Policy`.

Для включения политик плагин `Policy` должен быть указан в `PolicyPlugin`.

Если `PolicyPlugin = Policy`, то попытка пройти аутентификацию будет осуществляться для каждого плагина, указанного в `AuthServer`, а информация об успешном прохождении будет добавляться в `AuthBlock`. Затем плагин `Policy`, указанный в `PolicyPlugin`, проверяет `AuthBlock` и решает дать ли доступ пользователю.

Если `PolicyPlugin` не указан, то аутентификация завершается при первом успешном прохождении по методу, указанному в `AuthServer`.

Создание политик безопасности

Для создания, изменения или удаления политики безопасности администратору необходимо соединиться с какой-либо базой данных.

Хотя сами политики хранятся в базе данных безопасности `security5.fdb`, создать их можно, соединившись с любой базой данных. Однако пользователь при этом должен иметь права на запись в базу данных безопасности.

Для создание политики используется оператор `CREATE POLICY`. Синтаксис этого оператора приведен ниже:

```
CREATE POLICY <имя политики> [AS <параметр>=<значение> [,<параметр>=<значение>...]]

<параметр> ::= AUTH_FACTORS
             | PSWD_NEED_CHAR
             | PSWD_NEED_DIGIT
             | PSWD_NEED_DIFF_CASE
             | PSWD_MIN_LEN
             | PSWD_VALID_DAYS
             | PSWD_UNIQUE_COUNT
             | PSWD_NEED_SPECIAL
             | MAX_FAILED_COUNT
             | MAX_UNUSED_DAYS
```

Описание политик приведено в следующей таблице:

Таблица 9.5 — Параметры политик

Параметр	Тип	Описание
AUTH_FACTORS	VARCHAR(64)	Факторы аутентификации (задаются в круглых скобках через запятую): • SRP - пароль (верификатор) SRP • LEGACY - пароль Legacy • WIN_SSPI • GSS • CERTIFICATE - сертификат пользователя • VERIFYSERVER - проверка сервера • GOSTPASSWORD - пароль по ГОСТ
PSWD_NEED_CHAR	INTEGER	Минимальное количество букв в пароле
PSWD_NEED_DIGIT	INTEGER	Минимальное количество цифр в пароле
PSWD_NEED_DIFF_CASE	BOOLEAN	Требование использования различных регистров букв в пароле
PSWD_MIN_LEN	INTEGER	Минимальная длина пароля
PSWD_VALID_DAYS	INTEGER	Срок действия пароля
PSWD_UNIQUE_COUNT	INTEGER	Количество последних не повторяющихся паролей
PSWD_NEED_SPECIAL	INTEGER	Количество специальных символов, требуемых в пароле. Специальным считается любой печатный символ, за исключением букв, цифр и пробела.
MAX_FAILED_COUNT	INTEGER	Количество неудачных попыток входа
MAX_UNUSED_DAYS	INTEGER	Максимальное время неактивности учетных записей пользователя, в днях

Следующий пример демонстрирует создание политики:

```
CREATE POLICY TestPolicy AS
    AUTH_FACTORS = (SRP, LEGACY),
    PSWD_NEED_CHAR = 5,
    PSWD_NEED_DIGIT = 3,
    PSWD_MIN_LEN = 8,
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
    PSWD_NEED_DIFF_CASE = true,
    PSWD_VALID_DAYS = 15,
    PSWD_UNIQUE_COUNT = 5,
    MAX_FAILED_COUNT = 5,
    MAX_UNUSED_DAYS = 45;
```

Назначение политики пользователям

Назначение политики пользователю выполняется оператором `GRANT POLICY`:

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

Пользователь может иметь только одну политику. По умолчанию при создании пользователя ему назначается политика `DEFAULT`. В ней отсутствуют какие-либо требования к паролям или сессиям пользователей. Чтобы отменить предыдущую политику и назначить новую, нужно просто еще раз выполнить оператор `GRANT POLICY <новая_политика>`. То есть для того, чтобы отменить требования политики для определенного пользователя, ему необходимо назначить политику по умолчанию:

```
GRANT POLICY  "DEFAULT" TO <имя_пользователя>
```

Политики назначаются только пользователям, но не ролям. Назначить политику несуществующему пользователю нельзя.

После назначения пользователям политик, касающихся требований к паролям, администратор должен сменить этим пользователям пароли, чтобы они удовлетворяли требованиям сопоставленных этим пользователям политик безопасности. При этом ограничение `PSWD_UNIQUE_COUNT` действует в рамках используемого парольного плагина, т.е. если пользователь меняет `LEGACY`-пароль, и политикой задано ограничение на 3 неповторяющихся пароля, проверены будут текущий `LEGACY`-пароль и 2 предыдущих. Пароли остальных плагинов в проверке не участвуют.

Если при смене пароля пользователь введет пароль, который не удовлетворяет установленной для пользователя политике безопасности, то будет выдано сообщение:

```
Statement failed, SQLSTATE = HY000
modify record error
```

При неудачной попытке входа у пользователя увеличивается счётчик неудачных попыток и время доступа к БД устанавливается на текущее + 1 секунда. Когда достигается значение `MAX_FAILED_COUNT` при неудачных попытках соединиться с базой данных, пользователь блокируется. Для его разблокировки следует подключиться к базе безопасности `security5.fdb` и установить значение столбца `PLG$FAILED_COUNT` (см. таблицу `PLG$MF`), выполнив запрос:

```
UPDATE PLG$MF SET PLG$FAILED_COUNT=<значение> WHERE PLG$USER_NAME = '<имя_
пользователя>'
```

При установке значения параметра в 0 пользователь блокироваться не будет.

Есть и другой способ снятие блокировки пользователя. Сброс количества неудачных попыток и времени доступа к БД выполняется оператором `RESET USER`:

```
RESET USER <имя пользователя>;
```

Выполнить этот оператор может только пользователь, обладающий административными привилегиями в БД безопасности.

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.

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.

Работа с политиками безопасности через LDAP

Чтобы для LDAP-пользователей действовали политики безопасности, в LDAP для пользователя предусмотрены следующие атрибуты:

- `rdbPolicy` – название политики безопасности. Если у пользователя этот атрибут не найден, для него применяется политика безопасности по умолчанию (`DEFAULT`);
- `rdbLegacyPasswordTime` – время последней смены Legacy-пароля пользователем;
- `rdbSrpPasswordTime` – время последней смены SRP-пароля пользователем;
- `rdbPasswordTime` – время последней смены ГОСТ-пароля пользователем;
- `rdbFailedCount` – число проваленных попыток аутентификации с момента последнего успешного входа;
- `rdbAccessTime` – время, до которого вход пользователя на сервер запрещён.

Политика пользователя (`rdbPolicy`) задаётся при выполнении в базе оператора `GRANT <policy> TO <user>`, если в конфигурации задано использование `LDAP`. Это происходит после попытки назначить эту политику пользователю в `security5.fdb` (независимо от результата этой попытки). Если назначить политику не получилось ни в `security5.fdb`, ни в `LDAP`, выдаётся ошибка.

Время последней смены пароля (`rdbPasswordTime`) задаётся в `LDAP` после успешной синхронизации пароля.

Атрибуты `rdbFailedCount` и `rdbAccessTime` изменяются при попытке аутентификации пользователя на сервере. При успешной попытке они сбрасываются, т.е. устанавливаются в 0. При неудачном входе `rdbFailedCount` увеличивается на 1, `rdbAccessTime` устанавливается на секунду большим текущего времени.

9.4.15 Отображение объектов безопасности

С введением поддержки множества баз данных безопасности в РЕД Базе Данных появились новые проблемы, которые не могли произойти с единой глобальной базой данных безопасности. Кластеры баз данных, использующие одну и ту же базу данных безопасности, были эффективно разделены. Отображения предоставляют средства для достижения той же эффективности, когда множество баз данных используют каждая свою базу данных безопасности. В некоторых случаях требуется управление для ограничения взаимодействия между такими кластерами. Например:

- когда `EXECUTE STATEMENT ON EXTERNAL DATA SOURCE` требует обмена данными между кластерами;
- когда `SYSDBA` доступ к базам данных необходим от других кластеров, использующих службы;
- аналогичные проблемы существовали в РЕД Базе Данных 2.5 и 2.6 под Windows, из-за поддержки доверительной аутентификации: два отдельных списка пользователей — один в базе

данных безопасности, а другой в Windows, и необходимо связать их.

Единое решение для всех этих случаев является отображение информации о пользователе, входящего в систему, на внутренние объекты безопасности — CURRENT_USER и CURRENT_ROLE.

Создание отображения объекта безопасности выглядит следующим образом:

```
CREATE [GLOBAL] MAPPING <имя отображения>
USING {
      PLUGIN <имя плагина> [IN <имя базы данных>]
    | ANY PLUGIN [IN <имя базы данных> | SERVERWIDE]
    | MAPPING [IN <имя базы данных>]
    | '*' [IN <имя базы данных>] }
FROM { ANY <тип отоб-го объекта> |  <тип отоб-го объекта>   <имя отоб-го объекта> }
TO { USER | ROLE } [<имя объекта, на которое произведено отображение>]
```

Оператор CREATE MAPPING создаёт отображение объектов безопасности (пользователей, групп, ролей) одного или нескольких плагинов аутентификации на внутренние объекты безопасности – CURRENT_USER и CURRENT_ROLE.

Если присутствует опция GLOBAL, то отображение будет применено не только для текущей базы данных, но и для всех баз данных находящимся в том же кластере, в том числе и базы данных безопасности. Одноименные глобальные и локальные отображение являются разными объектами.

Предложение USING описывает источник отображения. Оно имеет весьма сложный набор опций:

- явное указание имени плагина (опция PLUGIN) означает, что оно будет работать только с этим плагином;
- оно может использовать любой доступный плагин (опция ANY PLUGIN), даже если источник является продуктом предыдущего отображения;
- оно может быть сделано так, чтобы работать только с обще серверными плагинами (опция SERVERWIDE);
- оно может быть сделано так, чтобы работать только с результатами предыдущего отображения (опция MAPPING);
- вы можете опустить использование любого из методов, используя звёздочку (*) в качестве аргумента;
- оно может содержать имя базы данных (опция IN), из которой происходит отображение объекта FROM.

Предложение FROM описывает отображаемый объект. Оно принимает обязательный аргумент — тип объекта. Особенности:

- при отображении имён из плагинов, тип определяется плагином;
- при отображении продукта предыдущего отображения, типом может быть только USER и ROLE;
- если имя объекта будет указано явно, то оно будет учитываться при отображении;
- при использовании ключевого слова ANY будут отображены объекты с любыми именами данного типа.

В предложении TO указывается пользователь или роль, на которого будет произведено отображение. NAME является не обязательным аргументом. Если он не указан, то в качестве имени объекта будет использовано оригинальное имя из отображаемого объекта.

Воспользоваться оператором создания отображений может SYSDBA, владелец базы данных (если отображение локальное), пользователь с ролью RDB$ADMIN, пользователь root (Linux).

Синтаксис позволяет изменять любые опции существующего отображения (ALTER MAPPING) и удалять отображение (DROP MAPPING).

Примеры

1. Включение доступа определённому пользователю из другой базы данных к текущей базе данных под другим именем.

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

2. Включение SYSDBA сервера (от основной базы данных безопасности) для доступа к текущей базе данных.

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

3. Обеспечение гарантирование, что у пользователей, которые подключаются традиционным плагином аутентификации не слишком много прав.

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

9.5 Ролевое разграничение доступа

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[2]:

```
CREATE ROLE <имя роли>
[SET SYSTEM PRIVILEGES TO <сис.привилегия> [,<сис.привилегия> ...]]
```

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.

Ролям могут предоставляться привилегии к объектам базы данных точно так же, как и пользователям. Для этого используется оператор GRANT. Одной роли может быть предоставлено произвольное количество привилегий. В дальнейшем роли могут назначаться отдельным пользователям, которые в результате получают все привилегии, предоставленные роли. Одна роль может быть назначена любому количеству пользователей или ролей. Для назначения роли пользователю используется оператор:

[2] Подробнее о назначении системных привилегий см. п. 9.6

```
GRANT [DEFAULT] <имя роли> [, [DEFAULT] <имя роли> ...]
TO [USER]|[ROLE] <имя польз-я/роли>  [, [USER]|[ROLE] <имя польз-я/роли>...]
[WITH ADMIN OPTION]
[{GRANTED BY | AS} [USER] <имя грантора>]
```

Пользователь, которому предоставлена роль, должен указать её при входе, для того чтобы получить её привилегии. Но помимо них пользователь получает привилегии всех ролей, назначенных ему с DEFAULT (если такие имеются). Поэтому если пользователь не указывает роль при подключении к серверу, то он получает права только тех ролей, которые ему назначены с DEFAULT. Вход в систему с несколькими ролями не поддерживается. Можно изменить текущую роль с помощью оператора SET ROLE.

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

```
REVOKE [ADMIN OPTION FOR] [DEFAULT] <имя роли> [, [DEFAULT] <имя роли> ...]
FROM [USER]|[ROLE] <имя польз-я/роли>  [, [USER]|[ROLE] <имя польз-я/роли>...]
[{GRANTED BY|AS} [USER] <имя грантора>]
```

Для удаления роли используется оператор:

```
DROP ROLE <имя роли>;
```

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

Таблица 9.6 — Предопределенные роли

Имя роли	Назначение роли
RDB$ADMIN	Дает полные права, но только в текущей базе данных.
PUBLIC	Роль по умолчанию для вновь создаваемых пользователей. Не имеет никаких прав. Не существует в базе данных в явном виде.
RDB$SYSADMIN	Даёт права управления пользователями, создания/изменения/удаления базы данных, а также возможность назначать права пользователя. Роль RDB$SYSADMIN должна назначаться в базе данных безопасности.
RDB$DBADMIN	Даёт права управления пользователями, выполнения резервного копирования и восстановления базы из бэкапа, управление настройками базы данных, а также возможность назначать права пользователя.
RDB$USER	Даёт права создавать объекты базы данных и манипулировать ими, выполнять хранимые процедуры.

9.5.1 Кумулятивное действие ролей

Роли могут быть грантованы другие роли. В СУБД РЕД База Данных действует принцип кумулятивного действия ролей. Это значит, что, привилегии конкретной роли - это объединение привилегий, выданных этой роли, и привилегий ролей, назначенных этой роли.

Правила кумулятивного действия ролей:

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

вилегии и привилегии ролей, которые ему назначены с DEFAULT;

- пользователь может с помощью оператора SET ROLE сменить роль, указанную при подключении. В этом случае привилегии пользователя CURRENT_USER будут складываться из привилегий роли, назначенной оператором SET ROLE и привилегии ролей, которые ему назначены с DEFAULT;
- при подключении происходит проверка, что данная роль существует и назначена данному пользователю;
- циклические ссылки ролей друг на друга недопустимы.

Назначение и отбор у ролей прав других ролей происходит аналогично назначению и отбору прав у пользователей или у ролей:

```
GRANT Role1 TO Role2;
REVOKE Role2 FROM Role1;
```

Попытка выполнить повторный GRANT одной роли на другую не даст ошибки – права обеих ролей не могут от этого измениться.

Попытка выполнить циклическое наследование прав между ролями:

```
GRANT Role1 TO Role2;
GRANT Role2 TO Role1;
```

Вернет ошибку при выполнении второго оператора:

```
Statement failed, SQLCODE = -607
unsuccessful metadata update
-role ROLE2 can not be granted to role ROLE1
```

При попытке повторного отбора прав одной роли у другой роли:

```
GRANT Role1 TO Role2;
REVOKE Role1 FROM Role2;
REVOKE Role1 FROM Role2;
```

будет выдано предупреждение:

```
Warning: privileges on ROLE1 is not granted to ROLE2.
```

Аналогичное предупреждение будет выдано и при попытке отнять у роли права той роли, которые не были ей назначены.

9.5.2 Роль RDB$ADMIN

Системная роль RDB$ADMIN, присутствует в каждой базе данных. Предоставление пользователю роли RDB$ADMIN в базе данных даёт ему права SYSDBA, но только в текущей базе данных.

Привилегии вступают в силу сразу после входа в обычную базу данных с указанием роли RDB$ADMIN, после чего пользователь получает полный контроль над всеми объектами базы данных.

Роль RDB$ADMIN может быть предоставлена с использованием ключевого слова DEFAULT. В этом случае пользователь автоматически будет получать административные привилегии даже если он не указал роль RDB$ADMIN при входе.

Предоставление в базе данных безопасности даёт возможность создавать, изменять и удалять учётные записи пользователей.

В обоих случаях пользователь с правами RDB$ADMIN роли может всегда передавать эту роль другим. Другими словами, WITH ADMIN OPTION уже встроен в эту роль и эту опцию можно не указывать.

Предоставление роли RDB$ADMIN в обычной базе данных

Для предоставления и удаления роли RDB$ADMIN в обычной базе данных используются операторы GRANT и REVOKE, как и для назначения и отмены остальных ролей.

```
GRANT [DEFAULT]  RDB$ADMIN TO <имя пользователя>
REVOKE [DEFAULT]  RDB$ADMIN FROM <имя пользователя>
```

Привилегии на роль RDB$ADMIN могут давать только пользователи с правами RDB$ADMIN.

Пользователь может указать роль не только при входе, но и позднее с помощью оператора SET ROLE. Если роль назначена с DEFAULT, то пользователь автоматически будет получать административные привилегии .

Предоставление роли RDB$ADMIN в базе данных безопасности

Предоставление роли RDB$ADMIN в базе данных безопасности даёт возможность создавать, изменять и удалять учётные записи пользователей. Для этого могут использоваться не только операторы GRANT и REVOKE, но и SQL команды управления пользователями: CREATE USER и ALTER USER, в которых указываются специальные опции GRANT ADMIN ROLE и REVOKE ADMIN ROLE.

```
CREATE USER <имя пользователя> PASSWORD '<пароль>'
GRANT ADMIN ROLE

ALTER USER <имя пользователя> REVOKE ADMIN ROLE
```

Привилегии на роль RDB$ADMIN могут давать только администраторы.

Для управления учётными записями пользователей пользователь, имеющий права на роль RDB$ADMIN, должен подключиться к базе данных безопасности с этой ролью. Чтобы управлять пользователями из обычной базы данных, у этого пользователя должны быть права на роль RDB$ADMIN в это базе данных. Он определяет роль при соединении и может в ней выполнить любой SQL запрос. Иначе управление учётными записями посредством SQL запросов недоступно.

То же самое можно сделать используя утилиту gsec указав параметр -admin для сохранения атрибута RDB$ADMIN учётной записи пользователя:

```
gsec -add <имя пользователя> -pw <пароль> -admin yes
gsec -mo <имя пользователя> -admin no
```

Для управления пользователями через утилиту gsec роль RDB$ADMIN должна быть указана в переключателе -role.

AUTO ADMIN MAPPING

Администраторы операционной системы Windows автоматически не получают права SYSDBA при подключении к базе данных (если, конечно, разрешена доверенная авторизация). Имеют ли администраторы автоматические права SYSDBA зависит от установки значения флага AUTO ADMIN MAPPING. Это флаг в каждой из баз данных, который по умолчанию выключен. Если флаг AUTO ADMIN MAPPING включен, то он действует, когда администратор Windows:

- подключается с помощью доверенной аутентификации
- не определяет никакой роли при подключении

После успешного подключения текущей ролью будет являться RDB$ADMIN.

Включение и выключение флага AUTO ADMIN MAPPING в обычной базе данных осуществляется следующим образом:

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

Эти операторы могут быть выполнены владельцами баз данных или администраторами.

Альтернативой включения такого флага служит создание отображения предопределённой группы DOMAIN_ANY_RID_ADMINS на роль RDB$ADMIN:

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

В обычных базах данных статус AUTO ADMIN MAPPING проверяется только во время подключения. Если администратор имеет роль RDB$ADMIN потому, что произошло автоматическое отображение во время входа, то он будет удерживать эту роль на протяжении всей сессии, даже если он или кто-то другой в это же время выключает автоматическое отображение. Точно также, включение AUTO ADMIN MAPPING не изменит текущую роль в RDB$ADMIN для администраторов, которые уже подключились.

Для включения AUTO ADMIN MAPPING в базе данных пользователей можно создать глобальное отображение предопределённой группы DOMAIN_ANY_RID_ADMINS на роль RDB$ADMIN следующим образом:

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

Также можно использовать утилиту командной строки gsec:

```
gsec -mapping set
gsec -mapping drop
```

Только SYSDBA может включить AUTO ADMIN MAPPING, если он выключен, но любой администратор может выключить его.

При выключении AUTO ADMIN MAPPING пользователь отключает сам механизм, который предоставлял ему доступ и, таким образом, он не сможет обратно включить AUTO ADMIN MAPPING. Даже в интерактивном gsec сеансе новая установка флага сразу вступает в силу.

9.6 Распределение системных привилегий

Пользователю можно назначить часть прав администратора БД, делегировав ему роль с системными привилегиями.

Для этого сначала нужно создать роль с системными привилегиями:

Листинг 9.1. Синтаксис оператора CREATE ROLE

```
CREATE ROLE <имя роли>
SET SYSTEM PRIVILEGES TO <сис.привилегия> [,<сис.привилегия> ...];
```

Таблица 9.7 — Системные привилегии

Привилегия	Описание
USER_MANAGEMENT	Управление пользователями.
READ_RAW_PAGES	Чтение страниц в сыром формате используя Attachment::getInfo()
CREATE_USER_TYPES	Создание, изменение и удаление не системных записей в таблице RDB$USER_TYPES.
USE_NBACKUP_UTILITY	Использование nbackup для создания резервных копий
CHANGE_SHUTDOWN_MODE	Закрытие базы данных (shutdown) и возвращение её в online.
TRACE_ANY_ATTACHMENT	Трассировка чужих пользовательских сессий.
MONITOR_ANY_ATTACHMENT	Мониторинг (MON$ таблицы) чужих пользовательских сессий.
CREATE_DATABASE	Создание новой базы данных (хранится в базе данных пользователей security.db).
DROP_DATABASE	Удаление текущей БД.
USE_GBAK_UTILITY	Использование утилиты или сервиса gbak.
USE_GSTAT_UTILITY	Использование утилиты или сервиса gstat
USE_GFIX_UTILITY	Использование утилиты или сервиса gfix
IGNORE_DB_TRIGGERS	Разрешает игнорировать триггеры на события БД
CHANGE_HEADER_SETTINGS	Изменение параметров на заголовочной странице БД.
SELECT_ANY_OBJECT_IN_DATABASE	Выполнение оператора SELECT из всех селективных объектов (таблиц, представлений, хранимых процедур выбора).
ACCESS_ANY_OBJECT_IN_DATABASE	Доступ (любым способом) к любому объекту БД.
MODIFY_ANY_OBJECT_IN_DATABASE	Изменение любого объекта БД.
CHANGE_MAPPING_RULES	Изменение правил отображения при аутентификации.
USE_GRANTED_BY_CLAUSE	Использование GRANTED BY в операторах GRANT и REVOKE.
GRANT_REVOKE_ON_ANY_OBJECT	Выполнение операторов GRANT и REVOKE для любого объекта БД.
GRANT_REVOKE_ANY_DDL_RIGHT	Выполнение операторов GRANT и REVOKE для выдачи DDL привилегий
CREATE_PRIVILEGED_ROLES	Создание привилегированных ролей (с использование SET SYSTEM PRIVILEGES)
GET_DBCRYPT_KEY_NAME	Получение имени ключа шифрования
MODIFY_EXT_CONN_POOL	Управление пулом внешних соединений
REPLICATE_INTO_DATABASE	Использование API репликации для загрузки наборов изменений в базу данных
PROFILE_ANY_ATTACHMENT	Профилирование подключений других пользователей.
EXECUTE_ANY_OBJECT_IN_DATABASE	Выполнение любого объекта в базе данных.
UPDATE_ANY_OBJECT_IN_DATABASE	Обновление любого объекта в базе данных.

Системные привилегии позволяют производить очень тонкую настройку, поэтому иногда вам нужно будет выдать более 1 системной привилегии для выполнения какой-либо задачи. Например, необходимо выдать IGNORE_DB_TRIGGERS совместно с USE_GSTAT_UTILITY, потому что

`gstat` должен игнорировать триггера на события БД.

Если вы хотите, чтобы конкретный пользователь получил системные привилегии, грантуйте ему соответствующую роль. Далее пользователю следует подключаться к сервисам или к базе данных с указанием этой роли.

Оператор `ALTER ROLE` изменяет список системных привилегий роли или удаляет их.

Листинг 9.2. Синтаксис оператора ALTER ROLE

```
ALTER ROLE <имя роли>
{     SET SYSTEM PRIVILEGES TO <сис.привилегия> [,<сис.привилегия>...]
    | DROP SYSTEM PRIVILEGES  }
```

При использовании предложения `SET SYSTEM PRIVILEGES TO` роли назначаются системные привилегии из списка. Для очистки списка системных привилегий установленных предыдущим оператором используйте оператор `ALTER ROLE` с предложением `DROP SYSTEM PRIVILEGES`.

Выполнить оператор могут администраторы, владелец роли или пользователи с привилегией `ALTER ANY ROLE`.

Для проверки имеет ли текущее подключение заданную системную привилегию можно воспользоваться встроенной функцией `RDB$SYSTEM_PRIVILEGE`.

9.7 Разграничение доступа к объектам базы данных

В этом разделе будет рассмотрено разграничение доступа в отношении операций над объектами конкретной базы данных.

После успешного входа в систему авторизованный пользователь получает доступ к серверу и ко всем базам данных этого сервера, но это не означает, что он имеет доступ к любым объектам в любой базе данных. После создания объекта, только пользователь создавший объект (его владелец) и администраторы имеют доступ к нему. Пользователю необходимы привилегии на каждый объект, к которому он должен получить доступ.

Контроль доступа субъектов к объектам доступа осуществляется через делегирование прав субъектам на различные действия над объектами.

Субъектами доступа в СУБД РЕД База Данных являются пользователи, роли, а также представления, процедуры, функции, пакеты и триггеры, запущенные от имени пользователей.

Объекты доступа - это сами базы данных и объекты баз данных – таблицы, представления, процедуры, функции, пакеты, генераторы, домены, исключения, роли, наборы символов, сортировки, `BLOB`-фильтры и записи.

Для каждой пары (субъект – объект) задается явное и недвусмысленное перечисление допустимых типов доступа (читать, писать и т.д.).

Для назначения прав субъектам доступа используется оператор `GRANT`. Для снятия этих прав используется оператор `REVOKE`. Полный синтаксис этих операторов представлен в Руководстве по SQL.

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.

Доступные для каждого объекта операции сведены в таблицу 9.8:

Таблица 9.8 — Объекты и операции над ними

Объект	Операция
База данных	Создание, изменение, удаление
Таблица	Создание, изменение, удаление, добавление строк, изменение строк, удаление строк, выборка строк, ссылка на указанные столбцы внешним ключом
Представление	Создание, изменение, удаление
Процедура/функция/пакет	Создание, изменение, удаление, запуск (вызов)
Триггер	Создание, изменение, удаление
Генератор	Создание, изменение, удаление, использование
Домен	Создание, изменение, удаление
Исключение	Создание, изменение, удаление, использование
Роль	Создание, удаление, назначение пользователю/другой роли, изменение системных привилегий
`BLOB` фильтр	Объявление и удаление объявления `BLOB` фильтра
Сортировка	Добавление сортировки и удаление существующей сортировки
Набор символов	Установка сортировки по умолчанию для набора символов
Записи	Добавление, изменение, удаление, выборка, ссылка на указанные столбцы внешним ключом

Пользователь, создавший объект базы данных, становится его владельцем. Только владелец объекта и пользователи с правами администратора в базе данных могут изменить или удалить объект базы данных. Владелец базы данных, то есть пользователь, который создал её, имеет все права на объекты, которые были созданы другими пользователями.

Администраторы или владелец объекта могут выдавать привилегии другим пользователям, в том числе и привилегии на право выдачи привилегий другим пользователям.

Все привилегии по доступу к объектам базы данных хранятся в самой базе, и не могут быть применены к любой другой базе данных.

9.7.1 Распределение прав на DDL-операции

В СУБД РЕД База Данных пользователь или роль могут получить права на выполнение операций изменения структуры базы данных (DDL-операции) и делегировать свои права другим пользователям или ролям (предложение `WITH GRANT OPTION` в операторе `GRANT`).

Права на создание, изменение и удаление объектов назначаются и отменяются следующими выражениями:

Таблица 9.9 — Назначение прав на DDL операции

Операция	Объект	Назначение прав
CREATE, ALTER ANY, DROP ANY, ALL [PRIVILEGES]	PROCEDURE, FUNCTION, PACKAGE, ROLE, TABLE, VIEW, EXCEPTION, GENERATOR, DOMAIN, SEQUENCE, CHARACTER SET, COLLATION, FILTER, JOB	GRANT <операция> <объект> TO <список получателей привилегий> [WITH GRANT OPTION] [{GRANTED BY|AS} [USER] <имя грантора>] REVOKE [GRANT OPTION FOR] <операция> <объект> FROM <список обладателей привилегий> [{GRANTED BY|AS} [USER] <имя грантора>]
CREATE	DATABASE	GRANT CREATE DATABASE TO <список пользователей и ролей> REVOKE CREATE DATABASE FROM <список пользователей и ролей>
ALTER, DROP, ALL [PRIVILEGES]	DATABASE	GRANT <операция> DATABASE TO <список получателей привилегий> [WITH GRANT OPTION] [{GRANTED BY|AS} [USER] <имя грантора>] REVOKE [GRANT OPTION FOR] <операция> DATABASE FROM <список обладателей привилегий> [{GRANTED BY|AS} [USER] <имя грантора>]

Предложение ALL [PRIVILEGES] объединяет привилегии CREATE, ALTER и DROP на указанный тип объекта.

При предоставлении привилегий пользователям можно указать предложение WITH GRANT OPTION, что позволяет свою очередь предоставлять другим пользователям эти привилегии.

Предложение GRANT OPTION FOR в операторе REVOKE позволяет отменить для соответствующего объекта право предоставления другим объектам эти привилегии.

С помощью предложение GRANTED BY можно предоставлять или отозвать права не от имени текущего пользователя, а от другого пользователя. При использовании оператора REVOKE после GRANTED BY права будут удалены только в том случае, если они были зарегистрированы от удаляющего пользователя. Предложение AS является синонимом GRANTED BY. Предложения GRANTED BY и AS могут использовать только владелец базы данных и администраторы. Даже владелец объекта не может использовать их, если он не имеет административных привилегий.

Кроме назначения прав непосредственно пользователям возможно назначение прав ролям, хранимым процедурам, функциям, пакетам, триггерам, представлениям и системный привилегиям.

Оператор назначения привилегий на создание, удаление и изменение базы данных имеет несколько отличную форму от оператора назначения DDL привилегий на другие объекты метаданных.

Привилегия CREATE DATABASE является особым видом привилегий, поскольку она сохраняется в базе данных безопасности. Список пользователей имеющих привилегию CREATE DATABASE можно посмотреть в виртуальной таблице SEC$DB_CREATORS. Привилегию на создание новой базы данных могут выдавать только администраторы в базе данных безопасности.

Привилегии ALTER DATABASE и DROP DATABASE относятся только к текущей базе данных. Привилегии на изменение и удаление текущей базы данных могут выдавать только администраторы.

Пример

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

```
GRANT CREATE TABLE TO TestUser;
```

Теперь пользователь сможет создавать таблицы, например:

```
CREATE TABLE TEST_TABLE (ID integer, Name: VARCAHR(50));
```

При попытке создать какой-либо другой объект базы данных пользователь получит сообщение об ошибке:

```
Statement failed, SQLCODE = -901
There is no privilege for this operation.
```

Аналогично пользователь может получить права на создание, изменение, удаление таблиц, представлений, процедур, функций и других объектов базы данных.

Для того, чтобы лишить пользователя права на изменение структуры базы данных, необходимо выполнить команду `REVOKE`, например:

```
REVOKE CRATE TABLE FROM TestUser;
```

Теперь при попытке выполнить операцию

```
CREATE TABLE TEST_TABLE_2 (ID integer, Name: VARCAHR(50))
```

пользователь получит сообщение об ошибке следующего вида:

```
Statement failed, SQLCODE = -901
There is no privilege for this operation.
```

9.7.2 Распределение прав на DML-операции

По аналогии с распределением прав на DDL-операции, администратор и владелец объекта могут распределять права и на операции доступа к данным (добавление, изменение, выборка, удаление). Пользователь или роль могут делегировать свои права другим пользователям или ролям (предложение `WITH GRANT OPTION` в операторе `GRANT`). Синтаксис запросов по доступу к DML-операциям сведен в таблицу 9.10:

Таблица 9.10 — Назначение прав на DML операции

Операция	Объект	Назначение прав
SELECT, DELETE, INSERT, UPDATE [(<имя столбца>[, <имя столбца>])], REFERENCES (<имя столбца>[, <имя столбца>...]), ALL [PRIVILEGES]	TABLE, VIEW	GRANT <операция> ON [TABLE] {<имя таблицы>|<имя view>} TO <список получателей привилегий> [WITH GRANT OPTION] [{GRANTED BY|AS} [USER] <имя грантора>] REVOKE [GRANT OPTION FOR] <операция> ON [TABLE] {<имя таблицы> | <имя представления>} FROM <список обладателей привилегий> [{GRANTED BY|AS} [USER] <имя грантора>]

(разрыв таблицы)

(разрыв таблицы)

Операция	Объект	Назначение прав
USAGE[3]	EXCEPTION, GENERATOR, SEQUENCE	GRANT USAGE ON {EXCEPTION|GENERATOR|SEQUENCE} <имя> TO <список получателей привилегий> [WITH GRANT OPTION] [{GRANTED BY|AS} [USER] <имя грантора>] REVOKE [GRANT OPTION FOR] USAGE ON {EXCEPTION|GENERATOR|SEQUENCE} <имя> FROM <список обладателей привилегий> [{GRANTED BY|AS} [USER] <имя грантора>]
EXECUTE	PROCEDURE, FUNCTION, PACKAGE	GRANT EXECUTE ON {PROCEDURE|FUNCTION|PACKAGE} <имя> TO <список получателей привилегий> [WITH GRANT OPTION] [{GRANTED BY|AS} [USER] <имя грантора>] REVOKE [GRANT OPTION FOR] EXECUTE ON {PROCEDURE | FUNCTION | PACKAGE} <имя> FROM <список обладателей привилегий> [{GRANTED BY|AS} [USER] <имя грантора>]

Предложение WITH GRANT OPTION означает, что пользователь (или роль), кроме права на ту или иную операцию, получит также возможность передать право на эту операцию другому пользователю или роли. Лишить пользователя возможности делегировать свои права можно с помощью оператора REVOKE GRANT OPTION.

С помощью предложение GRANTED BY можно предоставлять или отозвать права не от имени текущего пользователя, а от другого пользователя. При использовании оператора REVOKE после GRANTED BY права будут удалены только в том случае, если они были зарегистрированы от удаляющего пользователя. Предложение AS является синонимом GRANTED BY. Предложения GRANTED BY и AS могут использовать только владелец базы данных и администраторы. Даже владелец объекта не может использовать их, если он не имеет административных привилегий.

Кроме назначения прав непосредственно пользователям возможно назначение прав ролям, хранимым процедурам, функциям, пакетам, триггерам, представлениям и системный привилегиям.

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.

Примеры

Для того, чтобы дать пользователю, право на вставку данных в таблицу Test_Table необходимо выполнить команду:

```
GRANT INSERT ON TABLE Test_Table To TESTUSER;
```

Теперь пользователь TestUser сможет добавлять записи в таблицу Test_Table:

[3] Привилегия USAGE применяется для использования исключений и генераторов/последовательностей в пользовательских запросах.

```
INSERT INTO Test_Table (ID, Name) VALUES (1, 'Alex');
```

Попытка же выполнить другие операции манипулирования данными вернет ошибку. Например:

```
SELECT * FROM Test_Table;
```

вернет ошибку:

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

Для того, чтобы лишить пользователя права добавлять записи в таблицу, необходимо выполнить команду:

```
REVOKE INSERT ON TABLE Test_Table FROM TESTUSER;
```

Теперь при попытке вставить запись в таблицу `Test_Table` пользователь получит сообщение об ошибке:

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

При попытке доступа к неразрешенным столбцам таблиц пользователь получит сообщение об ошибке, например:

```
GRANT UPDATE (Name) ON Test_Table TO TestUser;
COMMIT;
CONNECT 'TestDB.fdb' USER TESTUSER PASSWORD TestPass123;
UPDATE Test_Table SET ID=2, Name='Tom';
```

вернет ошибку:

```
Statement failed, SQLCODE = -551
no permission for update/write access to COLUMN ID
```

так как пользователю `TESTUSER` разрешено изменять значение только столбца `Name`. А оператор

```
UPDATE Test_Table Name='Tom';
```

выполнится без ошибок.

9.7.3 Привилегии выполнения SQL кода

Все объекты метаданных содержащие DML или PSQL код могут выполнятся в одном из следующих режимов:

- с привилегиями вызывающего пользователя (привилегии `CURRENT_USER`);
- с привилегиями определяющего пользователя (владельца объекта метаданных).

В РЕД Базе Данных есть возможность указывать объектам метаданных с какими привилегиями они будут выполняться: вызывающего или определяющего пользователя. Для этого используется предложение `SQL SECURITY`, которое можно указать для таблицы, триггера, процедуры, функции или пакета. Если выбрана опция `INVOKER`, то объект метаданных будет выполняться с привилегиями вызывающего пользователя. Если выбрана опция `DEFINER`, то объект метаданных будет выполняться с привилегиями определяющего пользователя (владельца). Эти привилегии будут дополнены привилегиями выданные самому PSQL модулю с помощью оператора `GRANT`.

Привилегии выполнения, с которым по умолчанию создаётся любой PSQL модуль можно изменить с помощью оператора:

```
ALTER DATABASE SET DEFAULT SQL SECURITY {DEFINER | INVOKER}
```

Для сохранения обратной совместимости по умолчанию используется опция INVOKER.

- Представления (VIEWs) всегда выполняются с привилегиями определяющего пользователя (владельца);
- По умолчанию триггеры наследуют привилегии выполнения которые были указаны у таблицы. Привилегии выполнения могут быть переопределены в самом триггере;
- Процедуры и функции пакета всегда наследуют привилегии выполнения указанный при определении пакета. Привилегии выполнения не могут быть переопределены в самих процедурах и функция пакета;
- Анонимные PSQL блоки (EXECUTE BLOCK) всегда выполняются с правами вызывающего пользователя.

В хранимых процедурах, функциях и триггерах вы можете проверить действующего в настоящий момент пользователя, т.е. пользователя с привилегиями которого выполняется текущий модуль, с помощью системной контекстной переменной EFFECTIVE_USER из пространства имён SYSTEM.

```
select RDB$GET_CONTEXT('SYSTEM', 'EFFECTIVE_USER') from RDB$DATABASE
```

Один и тот же объект может вызываться в разных контекстах безопасности и требовать различных привилегий. Например у нас есть:

- хранимая процедура INV с SECURITY INVOKER, которая вставляет записи в таблицу T;
- хранимая процедура DEF с SQL SECURITY DEFINER, которая определена пользователем SYSDBA.

Если пользователь U вызывает процедуру INV, то для доступа к таблице T потребуется привилегия INSERT выданная пользователю U (и конечно привилегия EXECUTE для INV). В этом случае U является эффективным пользователем (EFFECTIVE_USER) во время выполнения INV.

Если пользователь U вызывает процедуру DEF, то для доступа к таблице T потребуется привилегия INSERT выданная пользователю SYSDBA (и EXECUTE на DEF для пользователя U). В этом случае SYSDBA является эффективным пользователем (EFFECTIVE_USER) во время выполнения DEF.

Если внутри DEF вызывается процедура INV, то эффективным пользователем по время выполнения INV будет так же SYSDBA.

9.8 Аудит

Аудит событий реализован на основе утилиты FBTrace. РЕД База Данных отличает пользовательскую трассировку и системный аудит, которые с помощью Services API позволяют отслеживать и анализировать все, что происходит в базе данных в режиме реального времени. Средства трассировки и аудита позволяют серверу отслеживать и записывать в лог-файлы такие события: соединения и отсоединения от БД (создания и удаления БД), операции DML и DDL, выполнение хранимых процедур и т.д. По умолчанию лог-файлы размещаются в том же каталоге, что и отслеживаемая (логируемая) БД, и имеют имя вида: <имя_базы.fbtrace_text> или <имя_базы.fbtrace_bin> для текстового и бинарного формата лога соответственно. Запись в лог для каждой конкретной БД начинает вестись с момента ее создания или присоединения к ней и до момента отсоединения от нее или ее удаления. Регистрируются события, завершившиеся как удачно, так и неудачно (с ошибкой).

Сессию системного аудита запускает сам сервер. Это означает, что нет необходимости взаимодействия с пользователем. События, которые будут отслеживаться в этой сессии, задаются в конфигура-

ционном файле и читаются при старте сессии.

Параметр `AuditTraceConfigFile` в файле конфигурации `firebird.conf` задает имя и расположение файла с настройками системного аудита. Этот параметр по умолчанию имеет значение `fbtrace.conf`. Но по умолчанию он не включен (`enabled false`), что означает отсутствие сессий системного аудита. Можно указать несколько конфигурационных файлов, тогда для каждого будет запущена отдельная сессия аудита. Файл с шаблоном настроек `fbtrace.conf` находится в корневом каталоге и содержит список отслеживаемых событий и указывает размещение логов трассировки для каждого события. Это позволяет достаточно гибко настроить параметры аудита различных событий для любой базы данных, при этом логирование будет осуществляться в отдельные файлы.

Что касается пользовательской трассировки, она нуждается в запуске пользователем явно. При запуске сессии пользовательской трассировки из приложения задаются ее конфигурация и имя (необязательный параметр). Конфигурация сессии представляет собой текстовый файл, составленный в соответствии с правилами и синтаксисом, приведенными в файле `fbtrace.conf`. Любой пользователь может инициировать и управлять сессией трассировки. Обычный пользователь может управлять сессиями только в своих соединениях и не может управлять сессиями, начатыми другими пользователями. Администраторы могут управлять любыми сессиями.

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

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

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

9.8.1 Типы событий аудита

В логе аудита могут быть зарегистрированы следующие события:

- начало и окончание ведения аудита для БД;
- присоединение к БД и отсоединение от нее;
- присоединение к сервису и отсоединение от него;
- старт сервиса, запрос к сервису;
- подготовка, выполнение и освобождение запроса к БД, а также выборка записей;
- компиляция и выполнение BLR- и DYN-запросов;
- начало и окончание выполнения хранимой процедуры;
- начало и окончание выполнения хранимой функции;
- начало и окончание выполнения триггера;
- установка значения контекстной переменной;
- начало и завершение транзакции;
- возникновение ошибок и предупреждений;
- сборка мусора.

По умолчанию система аудита выключена.

Несанкционированной попыткой выполнения действия считается такая, при которой не была пройдена аутентификация, либо не оказалось прав на выполнение действия. Неуспешной – любая другая неудачная попытка, закончившаяся ошибкой.

Сообщения о вызове сервисов и предъявлении факторов аутентификации записываются в лог-файл базы `security5.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, GZIP - в Linux-системах. После завершения сжатия исходный архивируемый файл удаляется, а его содержимое очищается если включена очистка памяти (`MemoryWipePasses > 0`). Директория, в которой будет создан архив, задаётся параметром `archive_directory`. После переименования рабочего лога, создается новый файл с именем переименованного. Этот новый файл используется в дальнейшем в качестве рабочего лога. Удаление старых лог-файлов не предусмотрено и может осуществляться средствами ОС и планировщиками задач.

9.8.2 Настройка аудита. Параметры конфигурационного файла

Настройка регистрации событий происходит с помощью изменения параметров в файле `fbtrace.conf`, расположенном в каталоге установки РЕД База Данных.

Файл конфигурации может состоять из двух секций:

```
database [= /путь/к/бд]
{

...

}
services
{

...

}
```

Различные `database ...` секции состоят из параметров, отвечающие за логирование событий на уровне запросов. `services` секция может быть только в единственном экземпляре, позволяет делать трейс для широкого круга вызовов Services API (таких как резервирование, восстановление данных и т.д.)

Настроить регистрацию событий можно для конкретной базы данных отдельно. Для этого нужно указать путь к нужной базе данных. При этом в качестве имени базы данных можно указать регулярное выражение на основе `SIMILAR TO`. Синтаксис `SIMILAR TO` см. в Руководстве по SQL.

Строка, следующая за символом #, считается комментарием. В параметрах, значения которых допускают использование регулярных выражений, используется синтаксис регулярных выражений SQL (аналогично оператору SIMILAR TO). Значение параметра true означает что параметр включен, false – что он выключен.

В текстовом режиме по умолчанию включено логирование единственного типа событий — завершения выполнения SQL-запросов (если включен сам аудит).

В бинарном режиме автоматически регистрируются события всех типов. Игнорируются все параметры, за исключением enabled, format, log_filename, max_log_size, rotate_log и archive_directory.

В конфигурационном файле настраиваются следующие параметры.

Общие параметры

enabled = true/false
: Вести аудит или нет. По умолчанию аудит выключен.

format = 0/1/2/3/text/binary/syslog/aggtrace
: Формат лог-файла.

  - 0/text – текстовый (по умолчанию);
  - 1/binary – бинарный;
  - 2/syslog - запись в системный лог;
  - 3/aggtrace - агрегатный.

  В Linux запись в системный лог осуществляется вызовом функции syslog, обычно события регистрируются в системный лог-файл (например, /var/log/messages). При этом используются следующие категории событий:

  - LOG_AUTH - для событий аутентификации;
  - LOG_AUTHPRIV - для событий безопасности;
  - LOG_DAEMON - для остальных событий.

  Приоритет событий:

  - LOG_ERR - для событий, завершившихся с ошибкой;
  - LOG_NOTICE - для событий безопасности;
  - LOG_INFO - для всех остальных событий.

  В Windows события регистрируюся в системный журнал событий. Для успешных событий используется тип сообщения EVENTLOG_INFORMATION_TYPE, для неуспешных - EVENTLOG_ERROR_TYPE. В качестве источника будет указан Red Database SQL Server.

  Параметр игнорируется при запуске трассировки через rdbtracemgr.

log_filename = <строка>
: Имя файла лога. Если этот параметр не задан, лог-файл создаётся в той же папке, где находиться БД и имеет имя вида <имя_базы.fbtrace_text> или <имя_базы.fbtrace_bin>. Возможно использование регулярных выражений в этом параметре. Например, с их помощью можно разбить путь к БД на группы и обращаться к этим группам конструкциями вида \1, \2 и т.д. (см. примеры ниже). При явном указании имени файла, все события от всех логируемых БД будут сохранятся в данном файле. В этом параметре разделитель каталогов Windows - символ обратной косой черты \ - должен дублироваться. Применяется для текстового и бинарного формата лог-файла. Параметр игнорируется при запуске трассировки через rdbtracemgr.

max_log_size = <число>
: Задает максимальный размер log-файлов в мегабайтах. Если значение параметра равно 0, то размер файла журнала не ограничен, ротация логов не используется. Значение по умолчанию для текстового формата лог-файла – 50. Значение по умолчанию для бинарного формата – 500.

Значение по умолчанию для агрегатного аудита - 2048. Не применяется при записи в системный лог. Параметр игнорируется при запуске трассировки через `rdbtracemgr`.

`rotate_log = true/false`
Определяет, использовать ли систему ротации логов. Если значение равно `true`, то при достижении логом размера `max_log_size` будет выполняться его ротация. Если значение равно `false`, то при достижении логом размера `max_log_size` запись в лог будет остановлена, сессия трассировки будет отключена. Значение по умолчанию для текстового формата лог-файла – `true`. Значение по умолчанию для бинарного формата - `false`. Применяется для текстового и бинарного формата лог-файла. Параметр игнорируется при запуске трассировки через `rdbtracemgr`.

`archive_directory = <строка>`
Путь к директории, в которую будут записываться логи после ротации. Значение по умолчанию – пусто, то есть архив будет создан в той же директории, где лог. Если операция записи в журнал завершилась с ошибкой исчерпания места, то будет произведена принудительная ротация логов. Применяется для текстового и бинарного формата лог-файла. Параметр игнорируется при запуске трассировки через `rdbtracemgr`.

`time_format = <h/n/u/m/s>`
Единица измерения, в которой будет указано время выполнения запросов, процедур, транзакций и т.д. По умолчанию используются миллисекунды.

- h - человекочитаемый формат;
- n - наносекунды;
- u - микросекунды;
- m - миллисекунды (значение по умолчанию);
- s - секунды.

Применяется для текстового формата лог-файла и для записи в системный лог.

`cancel_on_error = true/false`
Если включен, отменяет текущую записываемую операцию, если при записи в лог-файл произошла ошибка. По умолчанию выключено (значение `false`).

`print_hostname = true/false`
Включает/отключает печать имени хоста. По умолчанию значение `false`.

`log_services = true/false`
Включает или отключает аудит событий присоединения/отсоединения и старта сервиса. Агрегатный аудит не логирует события сервисов. По умолчанию выключено.

`log_service_query = true/false`
Определяет, записывать ли события запросов к сервису. По умолчанию выключено. Применяется для текстового формата лог-файла и для записи в системный лог.

`log_message = true/false`
Определяет, записывать ли сообщения, добавленные с помощью функции `RDB$TRACE_MSG`. По умолчанию выключено. Применяется для текстового формата лог-файла и для записи в системный лог.

`connection_id = <число>`
Задаёт номер (идентификатор) подключения на сервере, которое будет отслеживаться. По умолчанию равно 0, т.е. отслеживаются все подключения. Не применяется для агрегатного формата.

`include_gds_codes = <GDS-коды>`
Это список GDS-кодов ошибок или предупреждений. Если список пустой, то в конечный лог будут включены все ошибки. Иначе в лог будут записываться только ошибки из этого списка. Не применяется для агрегатного формата.

`exclude_gds_codes = <GDS-коды>`
Это список GDS-кодов ошибок или предупреждений. Если список пустой, то в конечный лог будут

включены все ошибки. Иначе в лог будут записываться ошибки, не входящие в этот список. Не применяется для агрегатного формата.

`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. Не применяется для агрегатного формата.

`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. Не применяется для агрегатного формата.

`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. Не применяется для агрегатного формата.

`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. Не применяется для агрегатного формата.

Параметры текстового аудита

`include_filter = <регулярное выражение>`
: Этот параметр задаёт регулярное выражение в синтаксисе SQL (`SIMILAR TO`), которому должен удовлетворять текст SQL-запроса. Если текст запроса не удовлетворяет заданному здесь шаблону, этот запрос не записывается в лог. Значение по умолчанию – пусто, то есть в конечный лог будут включены все запросы. Применяется только для текстового формата лог-файла.

`exclude_filter = <регулярное выражение>`
: Задаёт регулярное выражение в синтаксисе SQL (`SIMILAR TO`), которому не должен удовлетворять текст SQL-запроса. Аналогично `include_filter`. Значение по умолчанию – пусто. Применяется только для текстового формата лог-файла.

`log_connections = true/false`
: Определяет, записывать ли события присоединения/отсоединения к БД в лог-файл. Применяется только для текстового формата лог-файла.

`log_transactions = true/false`
: Определяет, записывать ли события начала и завершения транзакций в лог-файл[4]. Применяется только для текстового формата лог-файла.

`log_statement_prepare = true/false`
: Определяет, записывать ли события подготовки запросов к БД в лог-файл. Применяется только для текстового формата лог-файла.

`log_statement_free = true/false`
: Определяет, записывать ли события освобождения запросов к БД в лог-файл. Применяется только для текстового формата лог-файла.

`log_statement_start = true/false`
: Определяет, записывать ли события начала выполнения запросов к БД в лог-файл. Применяется только для текстового формата лог-файла.

`log_statement_finish = true/false`

[4] При подтверждении транзакции записывается операция `commit`, при откате - `rollback`

Определяет, записывать ли события окончания выполнения запросов к БД в лог-файл. Применяется только для текстового формата лог-файла.

`log_procedure_compile = true/false`
Определяет, записывать ли события компиляции хранимых процедур в лог-файл. Применяется только для текстового формата лог-файла. По умолчанию значение `false`.

`log_procedure_start = true/false`
Определяет, записывать ли события начала выполнения хранимых процедур. Применяется только для текстового формата лог-файла.

`log_procedure_finish = true/false`
Определяет, записывать ли события завершения выполнения хранимых процедур. Применяется только для текстового формата лог-файла.

`log_function_compile = true/false`
Определяет, записывать ли события компиляции хранимых функций в лог-файл. Применяется только для текстового формата лог-файла. По умолчанию значение `false`.

`log_function_start = true/false`
Определяет, записывать ли события начала выполнения хранимых функций. Применяется только для текстового формата лог-файла.

`log_function_finish = true/false`
Определяет, записывать ли события завершения выполнения хранимых функций. Применяется только для текстового формата лог-файла.

`log_trigger_compile = true/false`
Определяет, записывать ли события компиляции триггеров в лог-файл. Применяется только для текстового формата лог-файла. По умолчанию значение `false`.

`log_trigger_start = true/false`
Определяет, записывать ли события начала выполнения триггеров. Применяется только для текстового формата лог-файла.

`log_trigger_finish = true/false`
Определяет, записывать ли события завершения выполнения триггеров. Применяется только для текстового формата лог-файла.

`log_context = true/false`
Определяет, записывать ли события изменений значений контекстных переменных в лог-файл. Применяется только для текстового формата лог-файла.

`log_errors = true/false`
Включает/отключает запись об ошибках. Применяется только для текстового формата лог-файла.

`log_warnings = true/false`
Включает/отключает запись о предупреждениях. Применяется только для текстового формата лог-файла.

`log_initfini = true/false`
Определяет, записывать ли события начала/окончания ведения аудита БД в лог-файл. Применяется только для текстового формата лог-файла.

`log_sweep = true/false`
Определяет, записывать ли события процесса сборки мусора в лог-файл. Применяется только для текстового формата лог-файла.

`log_blr_requests = true/false`
Определяет, записывать ли события прямого выполнения откомпилированных запросов во внутреннем представлении сервера - BLR. Применяется только для текстового формата лог-файла.

`log_dyn_requests = true/false`
: Определяет, записывать ли события прямого выполнения откомпилированных запросов на изменение метаданных (DDL) во внутреннем представлении сервера - DYN. Применяется только для текстового формата лог-файла.

`log_privilege_changes = true/false`
: Включает/отключает запись событий, связанных с изменением правил разграничения доступа. Применяется только для текстового формата лог-файла.

`log_changes_only = true/false`
: Включает/отключает запись только тех событий, которые изменяли данные в базе. Применяется только для текстового формата лог-файла.

`time_threshold = <число>`
: Минимальное время выполнения запросов, процедур, транзакций и т.д. События, время выполнения которых меньше указанного, не будут регистрироваться в журнале. Значение по умолчанию – 100 мс. Применяется только для текстового формата лог-файла.

`max_sql_length = <число>`
: Максимальная длина одной записи SQL-запроса в лог-файле, в байтах. Значение по умолчанию – 0 (неограниченно), максимальное значение – 64K. Если длина запроса больше указанного здесь значения, запрос будет обрезан. Применяется только для текстового формата лог-файла[5].

`max_blr_length = <число>`
: Максимальная длина BLR-запроса, сохраняемого в лог, в байтах. Значение по умолчанию – 500 байт. Максимальное значение – 64K. Если длина запроса больше указанного здесь значения, запрос будет обрезан. Применяется только для текстового формата лог-файла.

`max_dyn_length = <число>`
: Максимальная длина DYN-запроса, сохраняемого в лог, в байтах. Значение по умолчанию – 500 байт. Максимальное значение – 64K. Если длина запроса больше указанного здесь значения, запрос будет обрезан. Применяется только для текстового формата лог-файла.

`max_arg_length = <число>`
: Максимальная длина одного параметра запроса / процедуры в лог-файле. Значение по умолчанию – 0 (неограниченно). Максимальное значение – 64K. Если длина параметра больше указанного здесь значения, параметр будет обрезан. Применяется только для текстового формата лог-файла.

`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.

`print_plan = true/false`
: Включает/отключает печать планов запросов. По умолчанию значение `false`. Применяется только для текстового формата лог-файла.

`explain_plan = true/false`
: Включает/отключает печать расширенных планов запросов. По умолчанию значение `false`. Применяется только для текстового формата лог-файла.

`print_perf = true/false`
: Включает/отключает печать статистики выполнения запросов. По умолчанию значение `false`. Применяется только для текстового формата лог-файла.

`print_blr = true/false`
: Если параметр установлен в `true`, то содержимое BLR-запросов будет преобразовываться в тек-

[5] 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.

стовое 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.

`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. По умолчанию значение `false`. Применяется только для текстового формата лог-файла.

`log_security_incidents = true/false`
Включает/отключает запись событий, связанных с нарушением безопасности сервера (инциденты безопасности). Если этот параметр включен, все такие события будут регистрироваться независимо от того, включена ли регистрация событий данного типа в других настройках. По умолчанию отключено. Применяется только для текстового формата лог-файла.

`print_security_level = true/false`
Включает/отключает печать уровня важности события. Уровни важности:

- `EMERG` - аварийный;
- `FATAL` - фатальный;
- `CRITICAL` - критический;
- `HIGH` - высокий;
- `MEDIUM` - средний;
- `LOW` - низкий;
- `DEBUG` - отладочный.

По умолчанию отключено. Применяется только для текстового формата лог-файла.

`print_security_type = true/false`
Включает/отключает печать типа события. Типы событий:

- `AUTHENTICATION` - аутентификация;
- `IDENTIFICATION` - идентификация;
- `USER MANAGEMENT` - управление пользователями и ролями;
- `ACCESS MANAGEMENT` - управление доступом;
- `START/STOP COMPONENT` - запуск/остановка сервера;
- `RESTORE` - восстановление из резервной копии;
- `ACCESS TO PROTECTED INFO` - изменение правил разграничения доступа;
- `VALIDATION OF THE SOFTWARE COMPONENTS` - проверка целостности объектов контроля;
- `OTHER` - другое.

По умолчанию отключено. Применяется только для текстового формата лог-файла.

Параметры агрегатного аудита

`reset_counters = true/false`
Опция `reset_counters` определяет, сбрасывать ли значения счётчиков для события. При `reset_counters = true` значения `perf`, `count`, `succeed` и `failed` у события будут обнуляться при вызове агрегатного аудита через сервисы с любыми опциями, кроме `-atrc_clear`. По умолчанию выключено.

`max_sql_length = <число>`
Определяет максимальную допустимую длину SQL-запроса. По умолчанию значение 0, то есть длина не ограничена.

`max_plan_length = <число>`
: Определяет максимальную допустимую длину плана SQL-запроса. По умолчанию значение 0, то есть длина не ограничена.

Примеры настройки

При помощи регулярных выражений можно задавать конкретные БД для логирования. Примеры конфигурационных файлов аудита:

Пример 1:

```
#Лог ведется для баз с именами test.fdb, azk2.fdb, rules.fdb
database = %[\\/](test|azk2|rules).fdb
{
    enabled = true
    # Логи сохраняются в файлы  test.log, azk2.log, rules.log соответственно
    log_filename = \1.log
}
```

Пример 2:

```
#Для всех БД на диске C с расширением fdb - формат лога текстовый
database = C:%.fdb
{
    enabled = true
    format = 0
}

#Для всех БД с расширением fdb на диске D - формат лога бинарный
database = D:%.fdb
{
    enabled = true
    format = 1
}
```

Пример 3:

В регулярных выражениях можно использовать группировку - ().

```
#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.

9.8.3 Текстовый файл аудита

Журнал аудита содержит записи в хронологическом порядке по времени завершения события. Далее будут описаны все варианты записей в зависимости от типа события.

Результат `UNAUTHORIZED` записывается, если возникла ошибка, связанная с недостаточными правами доступа или неверными данными при аутентификации. В случае возникновения других ошибок результатом будет `FAILED`.

Начало/окончание ведения аудита

Включить ведение записей об этом событии позволяет параметр `log_initfini`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>)
    SESSION_<ID сессии> <имя сессии>
    <путь к базе данных>
```

Где `<тип события>:= {TRACE_INIT | TRACE_FINI}`.

Присоединение/отсоединение от БД

Включить ведение записей об этом событии позволяет параметр `log_connections`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) <событие>, <важность
события>, <тип события>
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>

<тип события>:= {CREATE_DATABASE| ATTACH_DATABASE| DROP_DATABASE| DETACH_DATABASE}

<сведения о соединении> ::=
    <путь к БД> (ATT_<ID соединения>, <имя пользователя>:<роль>, <кодировка>,
    <протокол соединения>:<IP адрес или имя компьютера>:<MAC-адрес>)
```

В случае неуспешной или несанкционированной попытки выполнения присоединения или отсоединения от БД в типе события фиксируется результат `FAILED` или `UNAUTHORIZED`.

Начало/завершение транзакции

Включить ведение записей об этом событии позволяет параметр `log_transactions`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) <событие>
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
        (TRA_<ID транзакции>,<уровень изоляции>|<режим разр. бл-к>|<режим доступа>)
    [<глобальные счетчики>]
    [<табличные счетчики>]

<тип события> := {START_TRANSACTION | COMMIT_RETAINING | COMMIT_TRANSACTION |
                  ROLLBACK_RETAINING | ROLLBACK_TRANSACTION}
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
<сведения о соединении> ::=
    <путь к БД> (ATT_<ID соединения>, <имя пользователя>:<роль>, <кодировка>,
    <протокол соединения>:<IP адрес или имя компьютера>:<MAC-адрес>)

<уровень изоляции> ::= {CONSISTENCY | CONCURRENCY | {READ_COMMITTED|REC_VERSION} |
                        {READ_COMMITTED|NO_REC_VERSION} |
                        {READ_COMMITTED|READ_CONSISTENCY}}

<режим разрешения блокировок> ::= {WAIT [N] | NOWAIT};

<режим доступа к данным> := {READ_ONLY | READ_WRITE};

<глобальные счетчики> ::= m1 ms, m2 read(s), m3 write(s), m4 fetch(es), m5 mark(s)

<табл. счетчики> ::=  Table Natural Index Update Insert Delete Backout Purge Expunge
                      ************************************************************
```

В случае неуспешной или несанкционированной попытки выполнения старти или завершения транзакции в типе события фиксируется результат `FAILED` или `UNAUTHORIZED`.

Записи содержат табличные и глобальные счетчики, только если в настройках включен параметр `print_perf`.

Таблица 9.11 — Табличные счетчики

`Table`	Имя таблицы
`Natural`	Количество записей, считанных последовательно (без индекса)
`Index`	Количество записей, считанных по индексу
`Update`	Количество обновленных записей
`Insert`	Количество вставленных записей
`Delete`	Количество удаленных записей
`Backout`	Количество записей, для которых были восстановлены предыдущие версии из-за отката транзакции или точки сохранения
`Purge`	Количество записей, для которых были удалены устаревшие версии, не нужные ни одной активной транзакции
`Expunge`	Количество вычищенных средствами сборки мусора записей (`expunged records`). Это происходит, когда запись, которая не видна какой-либо активной транзакции, удаляется и удаляющая транзакция подтверждается (`commit`). Удалённая запись и все ранее подтверждённые версии этой записи удаляются, чтобы занятое ими дисковое пространство можно было повторно использовать. Если запись всё ещё видна для более старых snapshot транзакций, конечно, удаление может произойти только после завершения этих транзакций
`Lock`	Количество записей прочитанных с использованием предложения `WITH LOCK`
`Wait`	Количество попыток обновления/модификации/блокировки записей принадлежащих нескольким активным транзакциям. Транзакция находится в режиме `WAIT`

(разрыв таблицы)

(разрыв таблицы)

Conflict	Количество неудачных попыток обновления/модификации/блокировки записей принадлежащих нескольким активным транзакциям. В таких ситуациях сообщается о конфликте обновления (UPDATE CONFLICT)
BVersion	Количество прочитанных версий при поиске видимых версий записей
Fragment	Количество прочитанных фрагментов записей
Refetch	Количество повторно прочитанных записей

Таблица 9.12 — Глобальные счетчики

ms	время выполнения
read(s)	количество страниц, считанных с диска
write(s)	количество страниц, записанных на диск
fetch(es)	количество страниц, считанных из страничного кэша
mark(s)	количество страниц, изменённых в страничном кэше

Подготовка запросов к БД

Включить ведение записей об этом событии позволяет параметр log_statement_prepare. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) PREPARE_STATEMENT
<важность события>, <тип события>
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
        (TRA_<ID транзакции>,<уровень изоляции>|<режим разр. бл-к>|<режим доступа>)
Statement <идентификатор запроса>:
-------------------------------------------------------------------------------
<содержимое запроса>
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
[<план запроса>]
<время выполнения> ms

<сведения о соединении> ::=
    <путь к БД> (ATT_<ID соединения>, <имя пользователя>:<роль>, <кодировка>,
                 <протокол соединения>:<IP адрес или имя компьютера>:<MAC-адрес>)

<уровень изоляции> ::= {CONSISTENCY | CONCURRENCY | {READ_COMMITTED|REC_VERSION} |
                        {READ_COMMITTED|NO_REC_VERSION}}

<режим разрешения блокировок> ::= {WAIT [N] | NOWAIT};

<режим доступа к данным> := {READ_ONLY | READ_WRITE};
```

В случае неуспешной или несанкционированной попытки подготовки запроса в типе события фиксируется результат FAILED или UNAUTHORIZED.

План выполнения запроса выводится, если включен параметр print_plan.

Освобождение запросов к БД

Включить ведение записей об этом событии позволяет параметр `log_statement_free`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) {FREE_STATEMENT |
CLOSE_CURSOR} <важность события>, <тип события>
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
Statement <идентификатор запроса>:
-------------------------------------------------------------------------------
<содержимое запроса>
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
<план запроса>
```

План выполнения запроса выводится, если включен параметр `print_plan`.

Начало/окончание выполнения запросов к БД

Включить ведение записей об этом событии позволяют параметры `log_statement_start` и `log_statement_finish`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) <событие>, <важность
события>, <тип события>
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
        (TRA_<ID транзакции>,<уровень изоляции>|<режим разр. бл-к>|<режим доступа>)
Statement <идентификатор запроса>:
-------------------------------------------------------------------------------
<содержимое запроса>
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
[<план запроса>]
<параметры выполнения запроса>
[<количество выбранных записей> records fetched]
    [sorting memory usage: total: <суммарный размер кэша>,
     cached: <размер RAM кэша>, on disk: <размер дискового кэша>]
    [<глобальные счетчики>]
    [<табличные счетчики>]

<тип события>:= {EXECUTE_STATEMENT_START | EXECUTE_STATEMENT_FINISH}
```

В случае неуспешной или несанкционированной попытки выполнения запроса в типе события фиксируется результат `FAILED` или `UNAUTHORIZED`.

В типе события `{EXECUTE_STATEMENT_FINISH}`, при использовании сортировки, будет добавлена запись `sorting memory usage` со счетчиками выделенного кэша. В противном случае, если сортировка не использовалась, к `records fetched` будет добавлено `without sorting`.

План выполнения запроса выводится, если включен параметр `print_plan`.

Записи содержат табличные и глобальные счетчики (см. таблицу 9.11 и таблицу 9.12), только если в настройках включен параметр `print_perf`.

Изменение значений контекстных переменных

Включить ведение записей об этом событии позволяет параметр `log_context`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) SET_CONTEXT
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
        (TRA_<ID транзакции>,<уровень изоляции>|<режим разр. бл-к>|<режим доступа>)
[<пространство_имен>] <имя_переменной> =  <значение_переменной>
```

Изменение правил разграничения доступа

Включить ведение записей об этом событии позволяет параметр `log_privilege_changes`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) PRIVILEGES_CHANGE
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
        (TRA_<ID транзакции>,<уровень изоляции>|<режим разр. бл-к>|<режим доступа>)
    Executed by <executor> as <grantor>,operation:{ADD|DELETE} PRIVILEGE <прив-ия>
    <объект> for <имя пользователя>
    Attachment: <ID_соединения>, Transaction: <ID_транзакции>

<привилегия> ::= {ALL | INSERT | UPDATE | DELETE | SELECT | EXECUTE | REFERENCE |
CREATE | ALTER | ALTER ANY | DROP | DROP ANY | ROLE | ENCRYPTION KEY}
```

Начало/завершение выполнения хранимых процедур (функций)

Включить ведение записей об этом событии позволяют параметры `log_procedure_start` и `log_procedure_finish` (`log_function_start`, `log_function_finish`). Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) <событие>
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
     (TRA_<ID транзакции>,<уровень изоляции>|<режим разр. бл-к>|<режим доступа>)
Procedure (Function) <имя процедуры (функции)>:
    <входные параметры хранимой процедуры (функции)>
    [returns: <выходное значение функции>]
<количество выбранных записей> records fetched
    [sorting memory usage: total: <суммарный размер кэша>,
     cached: <размер RAM кэша>, on disk: <размер дискового кэша>]
    [<глобальные счетчики>]
    [<табличные счетчики>]

<тип события>:= {EXECUTE_PROCEDURE_START | EXECUTE_FUNCTION_START |
                 EXECUTE_PROCEDURE_FINISH | EXECUTE_FUNCTION_FINISH}
```

В случае неуспешной или несанкционированной попытки выполнения хранимой процедуры или функции в типе события фиксируется результат `FAILED` или `UNAUTHORIZED`.

В типе события {EXECUTE_PROCEDURE_FINISH}/{EXECUTE_FUNCTION_FINISH}, при использовании сортировки, будет добавлена запись sorting memory usage с счетчиками выделенного кэша. В противном случае, если сортировка не использовалась, к records fetched будет добавлено without sorting.

Записи содержат табличные и глобальные счетчики (см. таблицу 9.11 и таблицу 9.12), только если в настройках включен параметр print_perf.

Начало/завершение выполнения триггеров

Включить ведение записей об этом событии позволяют параметры log_trigger_start и log_trigger_finish. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>)
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
        (TRA_<ID транзакции>,<уровень изоляции>|<режим разр. бл-к>|<режим доступа>)
    <имя_триггера> [FOR <имя_таблицы (представления)>] ({ON <событие БД>} |
    {BEFORE | AFTER}  <событие таблицы (представления) или DDL-событие>)
    [<глобальные счетчики>]
    [<табличные счетчики>]

<тип события>:= {EXECUTE_TRIGGER_START | EXECUTE_TRIGGER_FINISH}
```

В случае неуспешной или несанкционированной попытки выполнения триггера в типе события фиксируется результат FAILED или UNAUTHORIZED.

Записи содержат табличные и глобальные счетчики (см. таблицу 9.11 и таблицу 9.12), только если в настройках включен параметр print_perf.

Компиляция BLR-запросов перед исполнением

Включить ведение записей об этом событии позволяет параметр log_blr_requests. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) COMPILE_BLR
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
        (TRA_<ID транзакции>,<уровень изоляции>|<режим разр. бл-к>|<режим доступа>)
Statement <идентификатор запроса>:
-------------------------------------------------------------------------------
<содержимое запроса>
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
<время выполнения> ms
```

В случае неуспешной или несанкционированной попытки компиляции BLR-запроса в типе события фиксируется результат FAILED или UNAUTHORIZED.

Выполнение BLR-запросов

Включить ведение записей об этом событии позволяет параметр log_blr_requests. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) EXECUTE_BLR
    <сведения о соединении>
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
    <клиентский процесс>:<ID клиентского процесса>
        (TRA_<ID транзакции>,<уровень изоляции>|<режим разр. бл-к>|<режим доступа>)
Statement <идентификатор запроса>:
-------------------------------------------------------------------------------
<содержимое запроса>
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    [<глобальные счетчики>]
    [<табличные счетчики>]
```

В случае неуспешной или несанкционированной попытки выполнения BLR-запроса в типе события фиксируется результат `FAILED` или `UNAUTHORIZED`.

Записи содержат табличные и глобальные счетчики (см. таблицу 9.11 и таблицу 9.12), только если в настройках включен параметр `print_perf`.

Выполнение DYN-запросов

Включить ведение записей об этом событии позволяет параметр `log_dyn_requests`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) EXECUTE_DYN
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
        (TRA_<ID транзакции>,<уровень изоляции>|<режим разр. бл-к>|<режим доступа>)
-------------------------------------------------------------------------------
<содержимое запроса>
<время выполнения> ms
```

В случае неуспешной или несанкционированной попытки выполнения DYN-запроса в типе события фиксируется результат `FAILED` или `UNAUTHORIZED`.

Присоединение/отсоединение к сервисам

Включить ведение записей об этом событии позволяет параметр `log_services`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>)
{ATTACH_SERVICE|DETACH_SERVICE} <важность события>, <тип события>
    service_mgr, ( Service  <ID сервиса> , <имя пользователя>,
                   <протокол соединения>:<IP адрес или имя компьютера>:<MAC-адрес>,
                   <клиентский процесс>:<ID клиентского процесса> )
```

В случае неуспешной или несанкционированной попытки присоединения (отсоединения) к сервису в типе события фиксируется результат `FAILED` или `UNAUTHORIZED`.

Старт сервиса

Включить ведение записей об этом событии позволяет параметр `log_services`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) START_SERVICE
    service_mgr, ( Service  <ID сервиса> , <имя пользователя>,
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
                    <протокол соединения>:<IP адрес или имя компьютера>:<MAC-адрес>,
                    <клиентский процесс>:<ID клиентского процесса>)
    <тип запроса к сервису>
    <опции, переданные сервис-менеджеру от клиента при запуске>
```

В случае неуспешной или несанкционированной попытки старта сервиса в типе события фиксируется результат `FAILED` или `UNAUTHORIZED`.

Запросы к сервису

Включить ведение записей об этом событии позволяют параметры `log_services` и `log_service_query`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) QUERY_SERVICE
    service_mgr, ( Service  <ID сервиса> , <имя пользователя>,
                <протокол соединения>:<IP адрес или имя компьютера>:<MAC-адрес>,
                <клиентский процесс>:<ID клиентского процесса>)
    <тип запроса к сервису>
    [Send portion of the query: <данные, переданные сервис-менеджеру> ]
    [Receive portion of the query: <данные, полученные сервис-менеджером>]
```

В случае неуспешной или несанкционированной попытки выполнения запроса к сервису в типе события фиксируется результат `FAILED` или `UNAUTHORIZED`.

Событие с ошибкой или предупреждением

Включить ведение записей об ошибках (предупреждениях) позволяет параметр `log_errors` (`log_warnings`). Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) {ERROR AT|WARNING
AT} <...>
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
<ошибки>
```

Чистка базы данных

Включить ведение записей о чистке базы данных позволяет параметр `log_sweep`. Структура записи в журнале аудита будет выглядеть так:

```
<дата события>T<время> (<ID процесса>:<ID потока>#<номер записи>) <тип события>
    <сведения о соединении>
    <клиентский процесс>:<ID клиентского процесса>
Transaction counters:
    Oldest interesting     <OIT>
    Oldest active     <OAT>
    Oldest snapshot  <OST>
    Next transaction  <Next>
[<глобальные счетчики>]
[<табличные счетчики>]
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
<тип события> ::= {SWEEP_START | SWEEP_FINISH | SWEEP_FAILED | SWEEP_PROGRESS}
```

Записи содержат табличные и глобальные счетчики (см. таблицу 9.11 и таблицу 9.12), только если в настройках включен параметр `print_perf`.

9.8.4 Адаптер для подключения бинарного файла аудита

Это инструмент анализа журнала аудита в бинарном формате, который позволяет подключать двоичные журналы в виде внешних таблиц и использовать SQL-запросы для поиска и фильтрации данных в них. Удаление и редактирование записей запрещено. Используется понятие версии формата бинарного лог-файла. Она проверяется при инициализации аудита (событие начала ведения аудита), если лог-файл уже существует и имеет бинарный формат. Если версия формата лога, используемая в РЕД Базе Данных, отличается от версии формата существующего файла, то производится ротация (переименование существующего лога, создание рабочего лог-файла с таким же именем).

Подключение журнала производится с помощью SQL-запроса вида:

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

- `table_name` – имя таблицы, которая будет хранить данные аудита;
- `filespec` – имя лог-файла, который должен быть подключен;
- `ADAPTER` – ключевое слово, необходимое для подключения в качестве внешней таблицы файла с нестандартным форматом данных;
- `fbtrace` – название адаптера, предназначенного для обработки бинарного лога системы аудита;
- `col_defs` – описание полей таблицы аудита (см. таблицу 9.13).

При подключение бинарного лога необходимо в `firebird.conf` настроить параметр `ExternalFileAccess`, по умолчание он выключен и подключение внешних файлов невозможно.

Таблица 9.13 — Поля создаваемой таблицы аудита по типам событий

Общие поля таблицы аудита		
Имя поля	**Тип**	**Комментарий**
EVENT_TIME	TIMESTAMP	Время регистрации события
EVENT_PROCESS_ID	INTEGER	Идентификатор процесса, вызвавшего событие
EVENT_OBJECT_ID	VARCHAR(16)	Идентификатор объекта, вызвавшего событие (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
EVENT_NUMBER	INTEGER	Номер события, записанного процессом сервера
SECURITY_LEVEL	VARCHAR(16)	Уровень важности события
SECURITY_TYPE	VARCHAR(50)	Тип события

Начало ведения аудита		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	TRACE INIT

Окончание ведения аудита		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	TRACE FINI

Присоединение к базе данных		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	CREATE DATABASE или ATTACH DATABASE
EVENT_USER	BLOB TEXT	Пользователь, от имени которого выполнено присоединение к БД
EVENT_PROTOCOL	BLOB TEXT	Протокол, по которому произошло подключение к БД
EVENT_HOSTNAME	BLOB TEXT	Имя хоста, с которого произошло подключение к БД
EVENT_HW_ADDRESS	BLOB TEXT	Физический адрес клиента (MAC-адрес адаптера)
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором произошло подключение к БД
EVENT_RESULT	VARCHAR(14)	Результат подключения (успешно, неуспешно, несанкционированно)

Отсоединение от базы данных		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	DROP DATABASE или DETACH DATABASE
EVENT_USER	BLOB TEXT	Пользователь, от имени которого выполнено отключение от БД
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором произошло отключение от БД

Начало транзакции		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	START TRANSACTION
EVENT_USER	BLOB TEXT	Пользователь, от имени которого стартовала транзакция
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором стартовала транзакция
TRANS_ID	BIGINT	Идентификатор транзакции
TRANS_OPT	BLOB TEXT	Параметры транзакции
EVENT_RESULT	VARCHAR(14)	Результат старта транзакции (успешно, неуспешно, несанкционированно)

Завершение транзакции		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	COMMIT RETAINING, COMMIT TRANSACTION, ROLLBACK RETAINING или ROLLBACK TRANSACTION
EVENT_USER	BLOB TEXT	Пользователь, от имени которого завершилась транзакция
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором завершилась транзакция
TRANS_ID	BIGINT	Идентификатор транзакции
PERF_INFO	BLOB TEXT	Статистика производительности запроса
PERF_TIME	BIGINT	Время выполнения запроса
EVENT_RESULT	VARCHAR(14)	Результат завершения транзакции (успешно, неуспешно, несанкционированно)

Изменение значения контекстной переменной		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	SET CONTEXT
EVENT_USER	BLOB TEXT	Пользователь, от имени которого произвели изменения
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором выполнили изменения контекстной переменной
TRANS_ID	BIGINT	Идентификатор транзакции, в которой произвели изменения
VAR_NS	BLOB TEXT	Пространство имен, для которого устанавливается контекстная переменная
VAR_NAME	BLOB TEXT	Имя контекстной переменной
VAR_VALUE	BLOB TEXT	Значение контекстной переменной

Начало выполнения хранимой процедуры		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	START FUNCTION или START PROCEDURE
EVENT_USER	BLOB TEXT	Пользователь, от имени которого выполняется процедура
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором выполняется хранимая процедура
TRANS_ID	BIGINT	Идентификатор транзакции
PROC_NAME	BLOB TEXT	Имя хранимой процедуры или функции
PROC_PARAMS	BLOB TEXT	Параметры хранимой процедуры или функции
EVENT_RESULT	VARCHAR(14)	Результат начала выполнения процедуры (успешно, неуспешно, несанкционированно)

Завершение выполнения хранимой процедур		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	FINISH FUNCTION или FINISH PROCEDURE
EVENT_USER	BLOB TEXT	Пользователь, от имени которого завершилось выполнение процедуры
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором завершилось выполнение процедуры
TRANS_ID	BIGINT	Идентификатор транзакции
PROC_NAME	BLOB TEXT	Имя хранимой процедуры или функции
PROC_PARAMS	BLOB TEXT	Параметры хранимой процедуры или функции
FUNC_RESULT	BLOB TEXT	Возвращаемый результат хранимой функции
PERF_INFO	BLOB TEXT	Статистика производительности хранимой процедуры или функции
PERF_TIME	BIGINT	Время выполнения хранимой процедуры или функции
ROW_FETCHED	BIGINT	Число выбранных записей
TOTAL_SORTING_MEMORY_USAGE	BIGINT	Суммарный размер кэша (в байтах), выделенного в процессе сортировки
SORTING_CHACHE_USAGE	BIGINT	Размер RAM кэша (в байтах), выделенного в процессе сортировки
SORTING_FILE_CHACHE_USAGE	BIGINT	Размер временных файлов (в байтах), созданных в процессе сортировки
EVENT_RESULT	VARCHAR(14)	Результат завершения выполнения процедуры (успешно, неуспешно, несанкционированно)

Подготовка запроса		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	PREPARE STATEMENT
EVENT_USER	BLOB TEXT	Пользователь, от имени которого шла подготовка запроса
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором шла подготовка запроса
TRANS_ID	BIGINT	Идентификатор транзакции
STMT_ID	BIGINT	Идентификатор запроса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
STMT_SQL	BLOB TEXT	Содержимое SQL-запроса
PERF_TIME	BIGINT	Время подготовки запроса
STMT_ACCESS_PATH	BLOB TEXT	План запроса
EVENT_RESULT	VARCHAR(14)	Результат подготовки запроса (успешно, неуспешно, несанкционированно)

Начало выполнения запроса		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	START EXECUTE STATEMENT
EVENT_USER	BLOB TEXT	Пользователь, от имени которого выполняется запрос
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором выполняется запрос
TRANS_ID	BIGINT	Идентификатор транзакции
STMT_ID	BIGINT	Идентификатор запроса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
STMT_PARAMS	BLOB TEXT	Параметры выполнения запроса
EVENT_RESULT	VARCHAR(14)	Результат начала выполнения запроса (успешно, неуспешно, несанкционированно)

Окончание выполнения запроса		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	FINISH EXECUTE STATEMENT
EVENT_USER	BLOB TEXT	Пользователь, от имени которого выполняется запрос
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором выполняется запрос
TRANS_ID	BIGINT	Идентификатор транзакции
STMT_ID	BIGINT	Идентификатор запроса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
STMT_PARAMS	BLOB TEXT	Параметры выполнения запроса
PERF_INFO	BLOB TEXT	Статистика производительности запроса
PERF_TIME	BIGINT	Время выполнения запроса
ROW_FETCHED	BIGINT	Число выбранных записей
TOTAL_SORTING_MEMORY_USAGE	BIGINT	Суммарный размер кэша (в байтах), выделенного в процессе сортировки
SORTING_CHACHE_USAGE	BIGINT	Размер RAM кэша (в байтах), выделенного в процессе сортировки
SORTING_FILE_CHACHE_USAGE	BIGINT	Размер временных файлов (в байтах), созданных в процессе сортировки
EVENT_RESULT	VARCHAR(14)	Результат завершения выполнения запроса (успешно, неуспешно, несанкционированно)

Освобождение запроса		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	FREE STATEMENT или CLOSE CURSOR
EVENT_USER	BLOB TEXT	Пользователь, от имени которого происходит освобождение запроса

(разрыв таблицы)

(разрыв таблицы)

Освобождение запроса		
Имя поля	**Тип**	**Комментарий**
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором освобождается запрос
STMT_ID	BIGINT	Идентификатор запроса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)

Компилирование BLR-запроса		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	COMPILE BLR
EVENT_USER	BLOB TEXT	Пользователь, от имени которого компилируется запрос
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором компилируется запрос
TRANS_ID	BIGINT	Идентификатор транзакции
STMT_ID	BIGINT	Идентификатор BLR-запроса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
BLR_DATA	BLOB BLR	Содержимое BLR-запроса
PERF_TIME	BIGINT	Время компилирования BLR-запроса
EVENT_RESULT	VARCHAR(14)	Результат компилирования BLR-запроса (успешно, неуспешно, несанкционированно)

Выполнение BLR-запроса		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	EXECUTE BLR
EVENT_USER	BLOB TEXT	Пользователь, от имени которого выполняется BLR-запрос
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором выполняется BLR-запрос
TRANS_ID	BIGINT	Идентификатор транзакции
STMT_ID	BIGINT	Идентификатор BLR-запроса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
PERF_INFO	BLOB TEXT	Статистика производительности BLR-запроса
PERF_TIME	BIGINT	Время выполнения BLR-запроса
EVENT_RESULT	VARCHAR(14)	Результат выполнения BLR-запроса (успешно, неуспешно, несанкционированно)

Выполнение DYN-запроса		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	EXECUTE DYN
EVENT_USER	BLOB TEXT	Пользователь, от имени которого выполняется DYN-запрос
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором выполняется DYN-запрос
TRANS_ID	BIGINT	Идентификатор транзакции
DYN_DATA	BLOB DYN	Содержимое DYN-запроса
PERF_TIME	BIGINT	Время выполнения DYN-запроса
EVENT_RESULT	VARCHAR(14)	Результат выполнения DYN-запроса (успешно, неуспешно, несанкционированно)

Присоединение к сервису		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	ATTACH SERVICE
EVENT_USER	BLOB TEXT	Пользователь, от имени которого выполняется присоединение к сервису
SVC_ID	VARCHAR(16)	Идентификатор сервиса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
EVENT_PROTOCOL	BLOB TEXT	Протокол, по которому произошло подключение к сервису
EVENT_HOSTNAME	BLOB TEXT	Имя хоста, с которого произошло подключение к сервису
EVENT_HW_ADDRESS	BLOB TEXT	Физический адрес клиента (MAC-адрес адаптера)
EVENT_DATABASE	BLOB TEXT	База данных, для которой присоединились к сервису
EVENT_RESULT	VARCHAR(14)	Результат присоединения к сервису (успешно, неуспешно, несанкционированно)

Старт сервиса		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	START SERVICE
SVC_ID	VARCHAR(16)	Идентификатор сервиса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
SVC_NAME	BLOB TEXT	Имя вызываемого сервиса
SVC_SWITCHES	BLOB TEXT	Параметры запуска сервиса
EVENT_RESULT	VARCHAR(14)	Результат старта сервиса (успешно, неуспешно, несанкционированно)

Запрос к сервису		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	QUERY SERVICE
SVC_ID	VARCHAR(16)	Идентификатор сервиса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
SVC_QUERY	BLOB TEXT	Тип запроса к сервису
EVENT_RESULT	VARCHAR(14)	Результат запроса к сервису (успешно, неуспешно, несанкционированно)

Отсоединение от сервиса		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	DETACH SERVICE
SVC_ID	VARCHAR(16)	Идентификатор сервиса (размер идентификатора заранее неизвестен, зависит от размера указателей в системе)
EVENT_RESULT	VARCHAR(14)	Результат отсоединения от сервиса (успешно, неуспешно, несанкционированно)

Проверка предъявленного фактора аутентификации		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	VERIFY AUTH FACTOR
EVENT_USER	BLOB TEXT	Пользователь, проходящий аутентификацию
AUTH_FACTOR_TYPE	BLOB TEXT	Тип фактора, предъявленного при аутентификации
EVENT_RESULT	VARCHAR(14)	Результат проверки предъявленных факторов аутентификации (успешно, неуспешно, несанкционированно)

Начало выполнения триггера		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	START TRIGGER
EVENT_USER	BLOB TEXT	Пользователь, от имени которого выполняется триггер
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором выполняется триггер
TRANS_ID	BIGINT	Идентификатор транзакции
TRIG_NAME	BLOB TEXT	Название триггера
TRIG_ACTION	BLOB TEXT	Событие, на которое срабатывает триггер
EVENT_RESULT	VARCHAR(14)	Результат начала выполнения триггера (успешно, неуспешно, несанкционированно)

Завершение выполнения триггера		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	FINISH TRIGGER
EVENT_USER	BLOB TEXT	Пользователь, от имени которого выполняется триггер
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором выполняется триггер
TRANS_ID	BIGINT	Идентификатор транзакции
TRIG_NAME	BLOB TEXT	Название триггера
TRIG_ACTION	BLOB TEXT	Событие, на которое срабатывает триггер
PERF_INFO	BLOB TEXT	Статистика производительности триггера
PERF_TIME	BIGINT	Время выполнения триггера
EVENT_RESULT	VARCHAR(14)	Результат завершения выполнения триггера (успешно, неуспешно, несанкционированно)

Изменение правила разграничения доступа		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	ADD PRIVILEGE или DELETE PRIVILEGE
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором происходит изменение правил разграничения доступа
TRANS_ID	BIGINT	Идентификатор транзакции
EVENT_USER	BLOB TEXT	Пользователь, от имени которого происходит изменение правил разграничения доступа
PRIVILEGE_GRANTOR	BLOB TEXT	Пользователь, выдавший привилегию
PRIVILEGE_OBJECT	BLOB TEXT	Объект, на который выдали привилегию
PRIVILEGE_GRANTEE	BLOB TEXT	Кому выдана привилегия
PRIVILEGE	BLOB TEXT	Выданная привилегия
PRIVILEGE_OPTION	INTEGER	Наличие GRANT OPTION при выдаче привилегии
EVENT_RESULT	VARCHAR(14)	Результат изменения правил разграничения доступа (успешно, неуспешно, несанкционированно)

Ошибка		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	WARNING или ERROR
PROC_NAME	BLOB TEXT	Название функции, в которой произошла ошибка
ERROR_MESSAGE	BLOB TEXT	Сообщение об ошибке

Сборка мусора		
Имя поля	**Тип**	**Комментарий**
EVENT_TYPE	CHAR(24)	SWEEP

(разрыв таблицы)

(разрыв таблицы)

Сборка мусора		
Имя поля	**Тип**	**Комментарий**
EVENT_ATT_ID	BIGINT	Идентификатор соединения, в котором произошло отключение от БД
SWEEP_OIT	BIGINT	Oldest Interesting Transaction
SWEEP_OAT	BIGINT	Oldest Active Transaction
SWEEP_OST	BIGINT	Oldest Snapshot Transaction
SWEEP_NEXT	BIGINT	Next Transaction
PERF_TIME	BIGINT	Время выполнения сборки мусора
PERF_INFO	BLOB TEXT	Статистика производительности запроса
EVENT_RESULT	VARCHAR(14)	Состояние сборки мусора (SWEEP_START | SWEEP_FINISH | SWEEP_FAILED | SWEEP_PROGRESS | UNKNOWN)

Значения некоторых полей могут быть пустыми в зависимости от типа события.

Так как в случае подключения лог-файла структура таблицы заранее известна, то при указании ключевого слова ADAPTER определение ее полей не обязательно. Если пользователь указывает тип адаптера без перечисления полей таблицы, создается таблица соответствующего адаптера со всеми возможными полями. Если же в запросе одновременно задается и тип адаптера, и структура таблицы, перечисленные поля должны быть подмножеством полей таблицы адаптера. Названия полей и их типы также жестко определяются типом адаптера. Порядок объявления полей не учитывается.

Если одно из полей задано неверно (ему не соответствует ни одно поле в таблице адаптера), в файл firebird.log записывается соответствующая ошибка и выполнение запроса прерывается.

Если структура таблицы указана верно, то не перечисленные в ней поля таблицы адаптера игнорируются.

При открытии бинарного лог-файла определяется версия его формата. Если она отличается от номера версии, которая является рабочей для данной версии РЕД Базы Данных, в файл firebird.log записывается соответствующая ошибка

Если производится попытка использования адаптера в БД, ODS которой не поддерживает этого, в файл firebird.log записывается соответствующая ошибка.

Пример подключения журнала с набором конкретных полей:

```
CREATE TABLE log EXTERNAL FILE '/home/tester/employee.fdb.fbtrace_bin'
ADAPTER 'fbtrace' (
        EVENT_TYPE CHAR(20),
        EVENT_DATABASE BLOB SUB_TYPE 1,
        EVENT_USER BLOB SUB_TYPE 1,
        EVENT_RESULT VARCHAR(14)
);
```

9.8.5 Утилита трассировки в интерактивном режиме (rdbtracemgr)

Утилита `rdbtracemgr` позволяет выполнять трассировку в интерактивном режиме. Утилита включает следующий функционал:

- Запуск пользовательской трассировки и отображение событий сервера (`START`)
- Завершение сеанса трассировки (`STOP`)
- Приостановление и возобновление сеанса трассировки (`SUSPEND` и `RESUME`)
- Вывод списка всех существующих сеансов трассировки (`LIST`)

Синтаксис вызова утилиты:

```
rdbtracemgr -SE <сервис> -U <имя пользователя> {-P <пароль> | -FE <путь к файлу>} [
<параметры подключения>] <действие>

<действие>::=
  <запуск сессии трассировки>
| <завершение сессии трассировки>
| <приостановление сессии трассировки>
| <возобновление сессии трассировки>
| -L[IST]

<запуск сессии трассировки>::= -STA[RT] -C[ONFIG] <конфигурационный файл> [-N <имя
сессии>] [-F[ORMAT] <число>] [-O[UTPUT] <путь к файлу> [-REP[LACE]]]

<завершение сессии трассировки>::= -STO[P] -I[D] <ID сессии>

<приостановление сессии трассировки>::= -SU[SPEND] -I[D] <ID сессии>

<возобновление сессии трассировки>::= -R[ESUME] -I[D] <ID сессии>
```

Таблица 9.40 — Параметры подключения rdbtracemgr

Параметр	Описание
`-SE[RVICE] <сервис>`	Имя сервиса. Обязательный параметр.
`-U[SER] <имя пользователя>`	Имя пользователя. Обязательный параметр.
`-P[ASSWORD] <пароль>`	Пароль.
`-FE[TCH] <путь к файлу>`	Считать пароль из файла. Параметр принимает строку без кавычек и апострофов, содержащую абсолютный или относительный путь к файлу с паролем. Файл должен быть доступен текущему пользователю операционной системы для чтения.
`-T[RUSTED]`	Использовать доверенную аутентификацию.

Таблица 9.41 — Опции rdbtracemgr

Опция	Описание
-C[ONFIG] <конфигурационный файл>	Путь к конфигурационному файлу аудита. Обязательный параметр. Параметры format, log_filename, max_log_size, rotate_log и archive_directory игнорируются.
-I[D] <ID сессии>	ID сессии для завершения, приостановки или возобновления конкретной сессии. Используется совместно с опцией STOP, RESUME или SUSPEND.
-STA[RT]	Запуск сессии трассировки.
-F[ORMAT] <число>	Формат лог-файла: 0 - текстовый формат, 1 - бинарный формат, -1 - legacy формат вывода. Если указано значение 1, то необходимо использовать -O[UTPUT]. По умолчанию значени -1.
-O[UTPUT] <путь к файлу>	Перенаправить вывод в файл.
-REP[LACE]	Перезаписать файл вывода, если он уже существует. Используется совместно с опцией -OUTPUT.
-STO[P]	Завершение сессии трассировки. Используется совместно с -ID.
-SU[SPEND]	Приостановить сессию трассировки. Используется совместно с -ID.
-R[ESUME]	Возобновляет сессию трассировки. Используется совместно с -ID.
-N[AME] <имя сессии>	Имя сессии для отображения в списке сессий.
-L[IST]	Вывести список существующих сессий трассировки.
-Z	Вывести версию сервера.

Пример запуска аудита:

```
rdbtracemgr -SE remote_host:service_mgr -USER SYSDBA -PASS masterkey -START -NAME my_
trace -CONFIG my_cfg.txt
```

9.9 Агрегатный аудит

Агрегатный аудит собирает метрики, которые агрегируются по событиям и транзакциям. Метриками являются следующие значения запросов: `read`, `write`, `fetch`, `mark` и время выполнения. Аудит хранит значения метрик во время работы сервера. При выключении/перезагрузке сервера сохранённая статистика будет очищена.

9.9.1 Настройка агрегатного аудита

1. Для настройки агрегатного аудита нужно создать конфигурационный файл, например, `aggtrace.conf`, и указать в нём следующие параметры:

```
database
{
  format = 3
  reset_counters = false
  max_log_size = 2048
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
}

database [= /путь/к/бд]
{
  enabled = true
  format = 3
  max_sql_length = 0
  max_plan_length = 0
}
```

Файл конфигурации состоит из двух секций. В глобальной секции (`database`) указываются настройки для плагина `aggtrace`. Указанные параметры распространяющиеся на все базы данных. В ней должны быть указаны параметры `log_max_size` и `reset_counters`.

В секции `database [= /путь/к/бд]` указываются параметры, распространяющиеся на заданную базу данных.

Опция `enabled = true` включает ведение аудита.

Опция `format = aggtrace` указывает, что нужно использовать агрегатный аудит.

Опция `reset_counters` определяет, сбрасывать ли значения счётчиков для события. При `reset_counters = true` значения `perf`, `count`, `succeed` и `failed` у события будут обнуляться при вызове агрегатного аудита через сервисы с любыми опциями, кроме `-atrc_clear`. Параметр должен быть определён в глобальной секции.

Опция `max_sql_length` определяет максимальную допустимую длину SQL-запроса, которая может храниться. Запросы, длина которых больше, будут обрезаны до указанного значения. Значение 0 указывает, что длина не ограничена.

Опция `max_plan_length` определяет максимальную допустимую длину плана SQL-запроса, которая может храниться. Планы, длина которых больше, будут обрезаны до указанного значения. Значение 0 указывает, что длина не ограничена.

Опция `max_log_size` задает максимальный размер log-файлов в мегабайтах. Допускается значение от 5 до 4096. Если значение параметра равно 0 (по умолчанию), то размер файла журнала не ограничен. Параметр должен быть определён в глобальной секции.

Настройки агрегатного аудита перечитываются для каждого соединения.

1. В `firebird.conf` в параметре `AuditTraceConfigFiles` укажите конфигурационный файл агрегатного аудита:

```
AuditTraceConfigFiles = aggtrace.conf;
```

1. В `firebird.conf` для параметра `TracePlugin` укажите значение `aggtrace`.

```
TracePlugin = aggtrace
```

Можно использовать несколько плагинов. Для этого нужно указать их через запятую.

9.9.2 Запуск агрегатного аудита

Запросить значения, собранные агрегатным аудитом, можно следующей командой:

```
./rdbsvcmgr -service_mgr -user <имя пользователя> -password <пароль> -action_aggtrace
[опции]
```

Набор всех возможных опций представлен ниже.

Таблица 9.42 — Опции агрегатного трейса

Опция	Описание
`-atrc_statement`	Запросить метрики событий. Если указана какая-либо `"get"` опция (`atrc_get_new`, `atrc_get_old`, `atrc_get_all`, `atrc_with_query`, `atrc_get` или `atrc_get_updated`), то `atrc_statement` будет применена автоматически.
`-atrc_get`	Запросить метрики новых и обновлённых запросов.
`-atrc_get_updated`	Запросить метрики только обновлённых запросов т.е тех, у которых изменились значения метрик.
`-atrc_get_new`	Запросить метрики только новых запросов.
`-atrc_get_old`	Запросить метрики только старых запросов, то есть тех, которые уже запрашивались ранее, но значения метрик не изменились.
`-atrc_get_all`	Запросить метрики по всем статусам.
`-atrc_with_query`	Добавлять текст запроса в вывод.
`-atrc_with_plan`	Добавлять план запроса в вывод.
`-atrc_text {<хэш запроса>|<хэш плана>}`	Вывести запрос или план запроса по указанному хэшу.
`-atrc_transaction`	Запросить метрики событий транзакций.
`-atrc_get_version`	Вывести версию плагина агрегатного аудита.
`-atrc_clear`	Очистить сохранённую статистику.

9.9.3 Вывод собранных метрик

Схема вывода результата работы `aggtrace.schema.json` расположена в каталоге `doc` установки сервера.

Метрики событий

Аудит агрегирует значения событий по следующим параметрам:

- dbname - полный путь к базе данных;
- event - событие;
- query - текст запроса.

Формат вывода метрик событий:

```
[
  {
    "event": "<событие>",
    "hash": "<хэш>",
    "status": "<статус>",
    "perf": [
      [
        <среднее количество страниц, считанных из страничного кэша (fetch)>,
        <минимальное количество страниц, считанных из страничного кэша (fetch)>,
        <максимальное количество страниц, считанных из страничного кэша (fetch)>,
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
          <общее количество страниц, считанных из страничного кэша (fetch)>
        ],
        [
          <среднее количество прочитанных (read) страниц базы данных>,
          <минимальное количество прочитанных (read) страниц базы данных>,
          <максимальное количество прочитанных (read) страниц базы данных>,
          <общее количество прочитанных (read) страниц базы данных>
        ],
        [
          <среднее количество страниц, изменённых в страничном кэше (mark)>,
          <минимальное количество страниц, изменённых в страничном кэше (mark)>,
          <максимальное количество страниц, изменённых в страничном кэше (mark)>,
          <общее количество страниц, изменённых в страничном кэше (mark)>
        ],
        [
          <среднее количество страниц, записанных на диск (write)>,
          <минимальное количество страниц, записанных на диск (write)>,
          <максимальное количество страниц, записанных на диск (write)>,
          <общее количество страниц, записанных на диск (write)>
        ],
        [
         <средний объём памяти, выделенный под сортировки>,
         <минимальный объём памяти, выделенный под сортировки>,
         <максимальный объём памяти, выделенный под сортировки>,
         <общий объём памяти, выделенный под сортировки>
        ],
        [
         <средний объём памяти, выделенный под сортировки на диске>,
         <минимальный объём памяти, выделенный под сортировки на диске>,
         <максимальный объём памяти, выделенный под сортировки на диске>,
         <общий объём памяти, выделенный под сортировки на диске>
        ],
        [
         <средний объём памяти, кэшированной в процессе сортировки>,
         <минимальный объём памяти, кэшированной в процессе сортировки>,
         <максимальный объём памяти, кэшированной в процессе сортировки>,
         <общий объём памяти, кэшированной в процессе сортировки>
        ],
        [
          <среднее время выполнения (ns)>,
          <минимальное время выполнения (ns)>,
          <максимальное время выполнения (ns)>
          <общее время выполнения (ns)>,
        ]
      ],
      "count": <общее количество событий>,
      "succeed": <количество успешно выполненных событий>,
      "failed": <количество событий, завершённых с ошибкой>,
      "dbname": "<полный путь к базе данных>",
      "query_hash": "<хэш запроса>",
      "query": "<текст запроса>",
      "plan_hash": "<хэш плана запроса>",
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
    "plan": "<план запроса>"
  },
  ...
]
```

Описание значений в выводе:

- `event` – Название события. Отслеживаются следующие события:
  - Завершение выполнения хранимой процедуры (`FINISH PROCEDURE`);
  - Завершение выполнения триггера (`FINISH TRIGGER`);
  - Подготовка запроса (`PREPARE STATEMENT`);
  - Начало выполнения запроса (`START STATEMENT`);
  - Завершение выполнения запроса (`FINISH EXECUTE STATEMENT`);
  - Освобождение запроса или закрытие курсора (`FREE STATEMENT`);
  - Завершение выполнения хранимой функции (`FINISH FUNCTION`).
- `hash` – Хэш агрегированного значения;
- `status` – Статус запроса:
  - `new` – Запрос, метрики которого ранее не запрашивались из агрегатного трейса;
  - `old` – Старый запрос, у которого сохранённые значения мметрик не изменились;
  - `updated` – Обновлённый запрос, у которого обновились сохранённые значения метрик.
- `perf` – Собранные значения метрик;
- `count` – Количество выполнений запроса;
- `succeed` – Количество успешных выполнений;
- `failed` – Количество выполнений, завершённых с ошибкой;
- `dbname` – Полный путь к базе данных или алиас;
- `query_hash` - Хэш запроса;
- `query` – Текст запроса;
- `plan_hash` - Хэш плана запроса;
- `plan` – План запроса.

Метрики транзакций

Формат вывода метрик транзакций:

```
[
  {
    "dbname": "<база данных>",
    "hash": "<хэш>",
    "<тип транзакции>":{
      "perf": [
        [
          <среднее количество страниц, считанных из страничного кэша (fetch)>,
          <минимальное количество страниц, считанных из страничного кэша (fetch)>,
          <максимальное количество страниц, считанных из страничного кэша (fetch)>,
          <общее количество страниц, считанных из страничного кэша (fetch)>
        ],
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
      [
        <среднее количество прочитанных (read) страниц базы данных>,
        <минимальное количество прочитанных (read) страниц базы данных>,
        <максимальное количество прочитанных (read) страниц базы данных>,
        <общее количество прочитанных (read) страниц базы данных>
      ],
      [
        <среднее количество страниц, изменённых в страничном кэше (mark)>,
        <минимальное количество страниц, изменённых в страничном кэше (mark)>,
        <максимальное количество страниц, изменённых в страничном кэше (mark)>,
        <общее количество страниц, изменённых в страничном кэше (mark)>
      ],
      [
        <среднее количество страниц, записанных на диск (write)>,
        <минимальное количество страниц, записанных на диск (write)>,
        <максимальное количество страниц, записанных на диск (write)>,
        <общее количество страниц, записанных на диск (write)>
      ],
      [
        <средний объём памяти, выделенный под сортировки>,
        <минимальный объём памяти, выделенный под сортировки>,
        <максимальный объём памяти, выделенный под сортировки>,
        <общий объём памяти, выделенный под сортировки>
      ],
      [
        <средний объём памяти, выделенный под сортировки на диске>,
        <минимальный объём памяти, выделенный под сортировки на диске>,
        <максимальный объём памяти, выделенный под сортировки на диске>,
        <общий объём памяти, выделенный под сортировки на диске>
      ],
      [
        <средний объём памяти, кэшированной в процессе сортировки>,
        <минимальный объём памяти, кэшированной в процессе сортировки>,
        <максимальный объём памяти, кэшированной в процессе сортировки>,
        <общий объём памяти, кэшированной в процессе сортировки>
      ],
      [
        <среднее время выполнения (ns)>,
        <минимальное время выполнения (ns)>,
        <максимальное время выполнения (ns)>
        <общее время выполнения (ns)>,
      ]
    ],
    "count": <общее количество событий>,
    "succeed": <количество успешно выполненных событий>,
    "failed": <количество событий, завершённых с ошибкой>
  },
  "ROLLBACK_TRANSACTION":{
    "perf": [...],
    "count": <общее количество событий>
  },
  "ROLLBACK_RETAINING:{
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
      "perf": [...],
      "count": <общее количество событий>
    },
    "COMMIT_TRANSACTION:{
      "perf": [...],
      "count": <общее количество событий>
    },
    "COMMIT_RETAINING:{
      "perf": [...],
      "count": <общее количество событий>
    },
    "FAILED ROLLBACK_TRANSACTION:{
      "perf": [...],
      "count": <общее количество событий>
    },
    "FAILED ROLLBACK_RETAINING:{
      "perf": [...],
      "count": <общее количество событий>
    },
    "FAILED COMMIT_TRANSACTION:{
      "perf": [...],
      "count": <общее количество событий>
    },
    "FAILED COMMIT_RETAINING:{
      "perf": [...],
      "count": <общее количество событий>
    }
  }
]
```

Описание значений в выводе:

- `dbname` – Путь к базе данных или алиас;
- `hash` – Хэш агрегированного значения;
- `тип транзакции` - Выполненная транзакция;
- `perf` – Собранные значения метрик для транзакции;
- `count` – Количество запусков транзакции;
- `succeed` – Количество успешных выполнений;
- `failed` – Количество выполнений, завершённых с ошибкой;

Вывод версии

Формат вывода версии:

```
{
  "version": "<версия агрегатного аудита>"
}
```

9.10 Очистка освобождаемых ресурсов

Важная особенность СУБД РЕД База Данных — версионная архитектура. Это означает, что при изменении записи создается новая версия записи. Предыдущая версия остается существовать. Каждая транзакция, стартовавшая на сервере, имеет свой номер. Запись также имеет номер создавшей ее транзакции. Таким образом, каждая транзакция может видеть свою версию записи и именно к ней иметь интерес.

Требование обезличивания памяти не распространяется на такие версии записей, которые интересны и используются какой-либо транзакцией. В том случае, если запись не интересна ни одной транзакции, т.е. любая из открытых транзакций не получит эту версию записи, то эта версия записи удаляется. В этот момент будет происходить обезличивание памяти, ранее занятой данной версией записи.

При удалении файлов они должны быть перезаписаны последовательностью нулей, единиц (0xFF), случайных значений столько раз, сколько указано в параметре конфигурации. После этого файл переименовывается некоторое количество раз, чтобы в журнале файловой системы не осталось имени исходного файла.

Функция очистки (обезличивания) освобождаемых ресурсов памяти встроена в РЕД Базу 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.

Глава 10

Встроенный сервер

10.1 Общие сведения

В СУБД РЕД База Данных существует специальный режим работы, называемый «встроенным» (`embedded`). Этот режим предназначен для прямого локального доступа приложения клиента к файлу базы данных минуя клиент-серверное сетевое подключение. При этом отдельная настройка сервера не требуется.

10.2 Установка embedded сервера

10.2.1 В ОС Windows

Встроенный режим не требует процедуры установки СУБД как таковой. Загрузите с официального сайта РЕД Базы Данных архив вида: `bin/windows/x86_64/RedDatabase-OE-5.0.X.X-windows-x86_64.zip` для 64-х разрядной версии. Обратите внимание, что битность дистрибутива должна совпадать с битностью вашей ОС.

Распакуйте соответствующий дистрибутив в рабочий каталог вашей программы. Для работы встроенного режима достаточно скопировать файлы из корневого каталога дистрибутива СУБД РЕД База Данных, а также все файлы из папки `/intl` и `/plugins` (достаточно файла `engine12.dll`).

10.2.2 В ОС Linux

Данная установка производится аналогично установке под Windows. В данном режиме в качестве провайдера используется библиотека `/plugins.libengine12.so`. Для работы встроенного сервера достаточно скопировать файлы из корневого каталога дистрибутива СУБД РЕД База Данных, и папок `/bin`, `/intl` и `/plugins` (достаточно файла `libengine12.so`).

В системе должна быть установлена библиотека `libicu`.

В случае возникновения ошибки подключения: `"error while loading shared libraries: libtinfo.so.5"` потребуется установить пакет `ncurses-compat-libs`.

10.3 Подключение

Для подключения в режиме встроенного сервера через API РЕД Базы Данных клиенту следует указать в строке подключения *локальный путь* (без хоста) к файлу базы данных (или его алиас, прописанный в файле `databases.conf`).

Встроенный сервер не требует аутентификации. Тем не менее, имя пользователя и, если необходимо, роль могут быть указаны в параметрах подключения, поскольку они используются для контроля доступа к объектам базы данных. По умолчанию встроенный сервер будет использовать имя текущего пользователя компьютера.

Можно легко переключаться с `embedded` на полноценный клиент-серверный режим без изменения строк кода приложения, только поменяв строку подключения к базе данных.

10.4 Несколько одновременных подключений

По умолчанию встроенный сервер запускается на всех доступных платформах (Windows, Linux) в режиме `SuperServer`.

Настройка архитектуры находится в конфигурационном файле `firebird.conf`:

```
ServerMode = Super
```

В этом режиме встроенный сервер создает эксклюзивную блокировку для файла базы данных на подключение и, в то время как он подключен, предотвращает подключения от других экземпляров. При такой конфигурации невозможно, например, иметь клиент-серверные подключения одновременно с клиентами браузера, подключенными к одной и той же базе данных через приложение интрасети, использующее встроенный механизм.

Решение состоит в том, чтобы запустить встроенный сервер как процесс `[Super]Classic` совместно с вашим сетевым сервером работающим также в режимах `Superclassic` или `Classic`. Раскомментируйте параметр в конфигурационном файле `firebirds.conf` вашего встроенного сервера и установите для него значение `Classic` или `SuperClassic`:

```
ServerMode = Classic
```

Обратите внимание, что для встроенного сервера режимы `Classic` и `SuperClassic` эквивалентны.

10.5 Запуск инструментов администрирования

Нет необходимости создавать отдельную файловую среду, если вы планируете использовать встроенное соединение для запуска инструментов администрирования, таких как `gbak` или `gfix`, или запуска привилегированного DDL в режимах `Classic или Superclassic`. Просто сохраните исход- ную конфигурацию провайдеров и войдите в систему, используя в строке подключения *локальный путь* к файлу базы данных и любое имя пользователя, необходимое для выполнения задач.

10.6 Примеры подключения

Для консольного подключения к базе данных в режиме встроенного сервера можно исполь- зовать утилиту `isql` из состава дистрибутива.

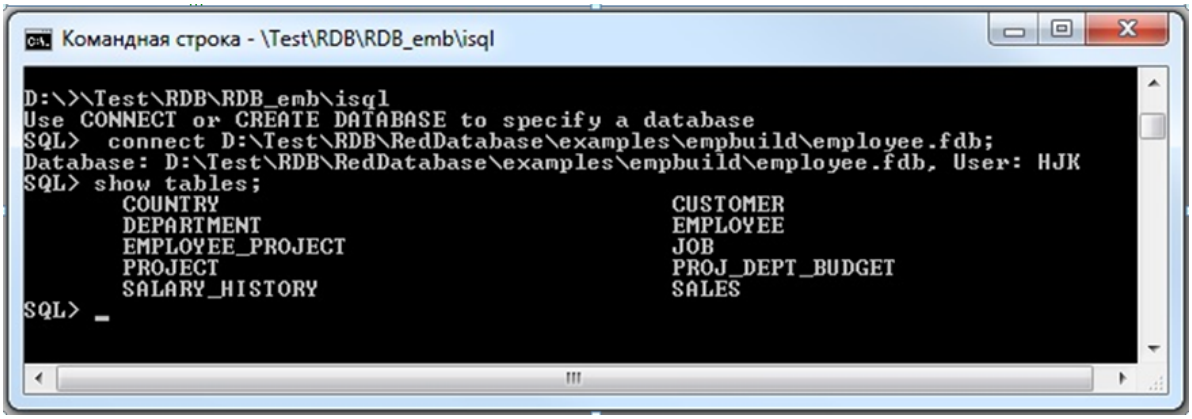


Рисунок 10.1 — Консольное подключение в режиме встроенного сервера через утилиту isql для Windows

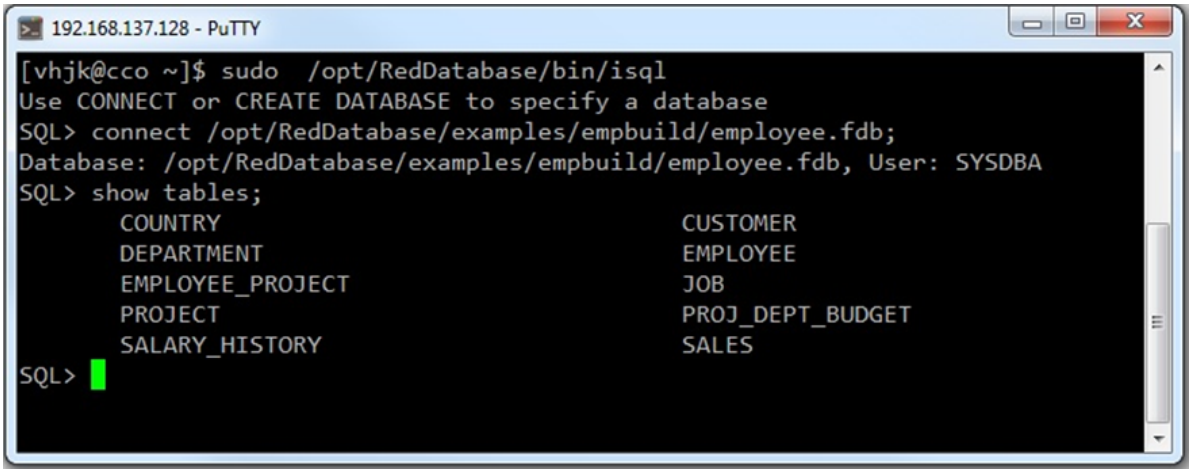


Рисунок 10.2 — Консольное подключение в режиме встроенного сервера через утилиту isql для Linux

Рисунок 10.3 — Пример кода подключения к базе данных на языке Python

Глава 11

Онлайн валидация базы данных

Проверка базы данных позволяет выполнять низкоуровневые проверки согласованности данных на диске и даже исправлять некоторые незначительные повреждения. Это рекомендуемая процедура для любой ценной базы данных, и администратор базы данных должен время от времени проверять базу, чтобы убедиться в ее работоспособности. Но процесс проверки требует монопольного доступа к базе данных, т.е. он запрещает любой параллельный доступ к базе данных во время выполнения проверки. В некоторых случаях, когда база данных велика, это может стать проблемой, т.к. проверка занимает заметное количество времени.

Проверка в режиме онлайн — это новая функция, которая позволяет выполнять некоторые проверки согласованности без монопольного доступа к базе данных.

11.1 Возможности проверки в режиме онлайн

Онлайн-проверка может делать следующее:

- проверять некоторые (или все) пользовательские таблицы в базе данных; системные таблицы не проверяются;
- проверять некоторые (или все) индексы;
- другие проверки ODS, такие как Страницы заголовка (Header), PIP, TIP, Страницы генераторов (Generators pages) не выполняются.

11.2 Защита во время онлайн проверки

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

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

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

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

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

Обе блокировки используют определенный пользователем тайм-аут блокировки. Если какой-либо запрос на блокировку завершается неудачно, сообщается об ошибке и таблица пропускается.

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

11.3 Команда для запуска онлайн проверки

Онлайн-валидация реализована в виде сервиса Firebird и доступна через Services API командой `isc_action_svc_validate`. Поэтому утилита `gfix` не может запустить онлайн-проверку. Утилита `rdbsvcmgr` имеет полную поддержку нового сервиса. Её синтаксис:

```
rdbsvcmgr [host:]service_mgr [user <...>] [password <...>]
action_validate dbname <файл БД>
[val_tab_incl <шаблон>]
[val_tab_excl <шаблон>]
[val_idx_incl <шаблон>]
[val_idx_excl <шаблон>]
[val_lock_timeout <число>]
```

где:

- `val_tab_incl` – шаблон для таблиц, включенных в проверку. По умолчанию все пользовательские таблицы включены, системные таблицы не проверяются;
- `val_tab_excl` – шаблон для таблиц, исключенных из проверки;
- `val_idx_incl` – шаблон для индексов, включенных в проверку. По умолчанию `%` – все индексы включены;
- `val_idx_excl` – шаблон для индексов, исключенных из проверки;
- `val_lock_timeout` – тайм-аут ожидания блокировки на таблицу (в секундах). По умолчанию 10 секунд, -1 - бесконечное ожидание.

- Шаблоны - это регулярные выражения, обрабатываемые по тем же правилами, что и выражения типа `SIMILAR TO`;
- Независимо от диалекта базы данных все шаблоны чувствительны к регистру;
- Если пропущен шаблон для таблиц, то будут проверены все пользовательские таблицы;
- Если пропущен шаблон для индексов, то будут проверены все индексы назначенных таблиц;
- Системные таблицы не проверяются;
- Для указания списка таблиц или индексов:
  1. необходимо разделять имена символом `"|"`;
  2. не добавляйте пробелы, т.е. шаблон `"TAB1 | TAB2"` является неверным;
  3. весь список должен быть заключен в двойные кавычки, чтобы интерпретатор команд его правильно обработал.

11.4 Примеры

1. Этой командой включается онлайн-проверка в базе данных `/home/test/db.fdb` таблиц, названия которых начинается с буквы `"A"`; индексы не проверяются; ожидание блокировки не выполняется:

   ```
   rdbsvcmgr service_mgr user SYSDBA password masterkey
   action_validate dbname /home/test/db.fdb
   val_tab_incl A%
   val_idx_excl %
   val_lock_timeout 0
   ```

2. Эта команда проверит таблицы `TAB1` и `TAB2` и все их индексы. Тайм-аут ожидания блокировки составляет 10 секунд.

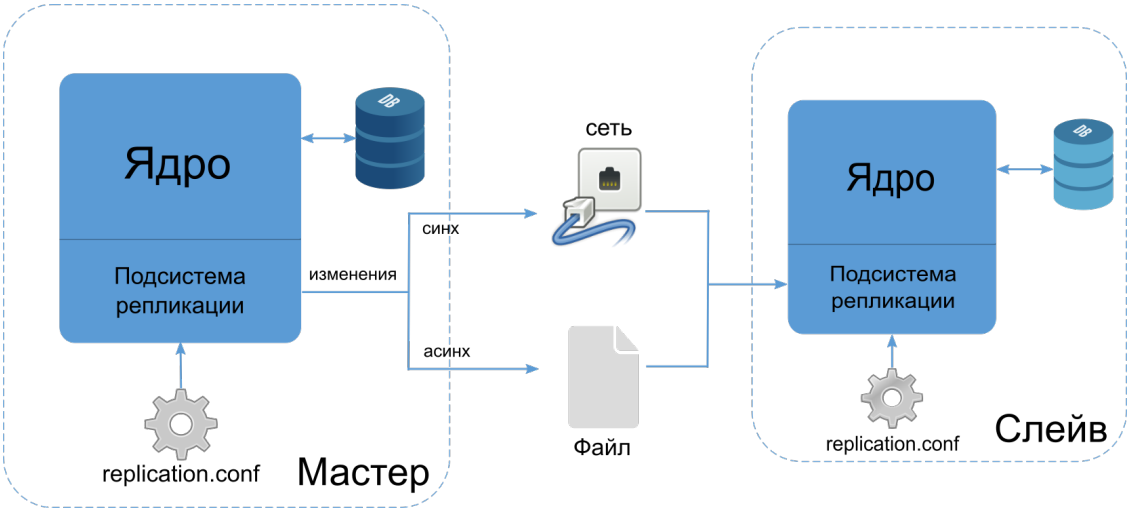
```
rdbsvcmgr service_mgr user SYSDBA password masterkey
action_validate dbname /home/test/db.fdb
val_tab_incl "TAB1|TAB2"
```

Глава 12

Репликация

12.1 Особенности репликации

Большая часть репликаторов, написанных для Firebird, представляет собой внешние по отношению к серверу приложения. Они хорошо подходят для миграции данных между различными базами данных или даже между различными СУБД, но плохо подходят для поддержания копии БД в рамках решения задачи создания отказоустойчивого кластера. Основной проблемой при этом является производительность, т.к. сторонние средства создают дополнительную нагрузку при синхронизации данных, причём она обычно избыточна и существенно замедляет работу СУБД. Кроме того, не все средства репликации могут гарантировать идентичность данных в основной базе и в копии при возникновении сбоя.



Встроенная репликация предназначена для обеспечения повышенной отказоустойчивости в случае повреждения физической структуры файла базы данных, вызванной техническим сбоем оборудования, программным сбоем операционной системы или самой СУБД. Она подразумевает перенос любых изменений данных с основного рабочего сервера на один или несколько резервных серверов, гарантируя, таким образом, их идентичность с точки зрения хранящихся на них данных. При этом главный сервер принято называть мастером, а резервные – слейвами.

Элемент репликации представляет собой запись в таблице, любые изменения которой передаются на резервные сервера. Также передаются и контекстные изменения, а именно соединения и транзакции, в которых происходит изменение записи. Таким образом, основными событиями репликации являются:

- создание и завершение соединений;
- начало, конец и откат каждой транзакции;
- начало, конец и откат точки сохранения;
- вставка, обновление и удаление записей;
- изменение генераторов.

Однозначное соответствие строк на основном и резервном серверах определяется уникальным индексом. Механизм репликации не содержит каких-либо средств, гарантирующих логическую непротиворечивость данных между основной базой и её резервными копиями. Полноценно реплицируются только таблицы, содержащие уникальный индекс, например, первичный ключ. В случае отсутствия

или неактивности такого индекса возможна репликация только операций вставки записи.

В процессе работы репликации возможны конфликты. Конфликтом считается обнаруженное несоответствие между данными на основном и резервном серверах. Примерами конфликтов являются:

- наличие строки в реплике при попытке ее вставки;
- отсутствие строки в реплике при ее изменении или удалении;
- наличие транзакции при попытке её запуска;
- отсутствие транзакции при её завершении.

Если произошел конфликт другого вида, это расценивается как ошибка репликации, которая приведёт либо к отключению репликации и продолжению работы мастера в штатном режиме, либо к отмене операции, во время которой произошла ошибка и сообщению о ней пользователю.

12.2 Режимы репликации

Существует два основных режима работы репликации: синхронный и асинхронный. В первом случае при наступлении события репликации в мастер-базе, оно должно быть также выполнено на всех слейв-базах для того, чтобы событие считалось завершенным. Логически это означает, что существует лишь одна версия данных. Основной недостаток синхронной репликации – она создаёт дополнительную задержку в работе мастера, т.к. СУБД должна дождаться ответа от реплики, чтобы считать операцию завершенной.

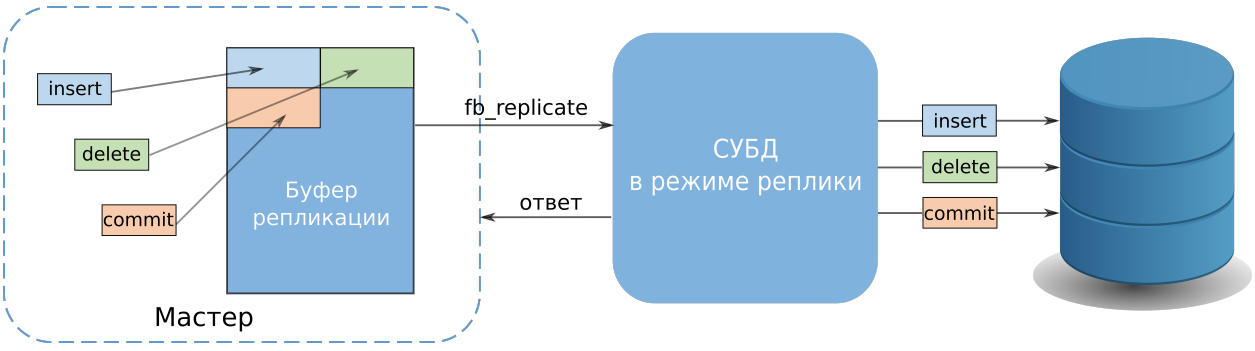
В случае асинхронной репликации обновления мастер-базы пишутся в журналы, которые распространяются на реплику спустя некоторое время, а не в той же транзакции. Таким образом, при асинхронной репликации уменьшаются простои мастера, но при этом вводится задержка, в течение которой реплики могут быть фактически неидентичными. Кроме того, больше вероятность конфликтов из-за того, что процесс копирования журналов репликации выполняется сторонними средствами.

Рассмотрим подробнее принципы работы и особенности разных режимов репликации.

12.2.1 Синхронная репликация

При синхронной репликации в процессе подключения к основному серверу проверяется наличие и доступность резервных серверов, после чего устанавливается с ними постоянное соединение. При ошибке подключения к слейву подключение к мастеру также завершается с ошибкой. Подключение к слейву может выполняться от имени текущего пользователя, либо от имени пользователя, заданного в конфигурации.

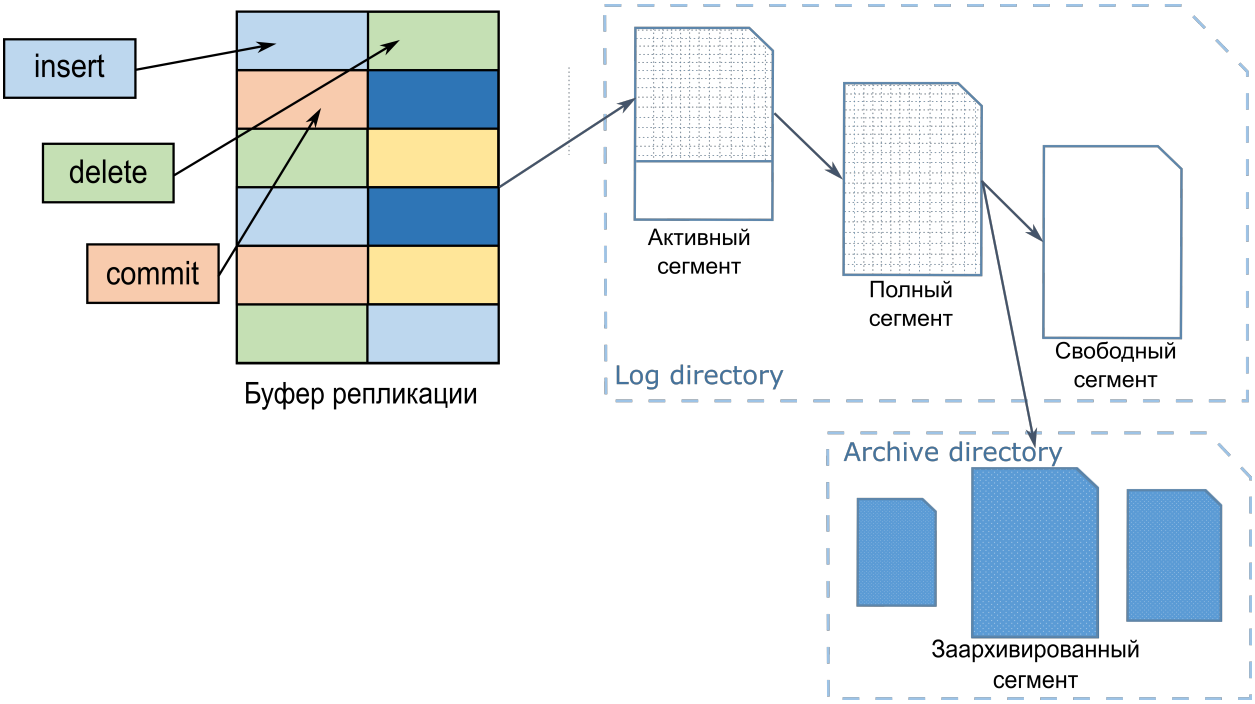
При наступлении событий в рамках транзакции, они записываются в буфер репликации, который передаётся на слейв либо при его заполнении, либо при наступлении определённых событий, например, коммита транзакции или завершения соединения. Слейв, получив репликационный пакет, должен выполнить все указанные в нём действия. При ошибке в любом из них применение пакета прекращается, а сообщение об ошибке репликации отправляется на мастер.



При синхронной репликации особую проблему составляет коммит транзакции. При наличии сложной отложенной работы время коммита может существенно увеличиться, если он будет последовательно выполняться на мастере и на слейве. Поэтому при репликации коммита на слейв посылается специальное событие подготовки коммита, и отложенная работа на мастере и слейве выполняется параллельно.

12.2.2 Асинхронная репликация

Асинхронная репликация также использует буферизацию событий репликации, но не посылает содержимое буфера на слейв, а пишет его в журнал репликации. Он представляет собой набор бинарных файлов (сегментов журнала) фиксированного размера, хранящихся в директории, заданной в конфигурации репликации. СУБД сбрасывает буферы с событиями репликации в текущий (активный) сегмент до тех пор, пока он не будет заполнен. После этого отдельный поток выполняет архивацию сегмента с помощью внешней команды. По окончании архивации сегмент считается свободным и может снова использоваться для записи событий репликации.



В данной модели существует четыре возможных состояния сегментов журнала:

- `USED` – сегмент в данный момент используется. В конкретный момент времени он единственный. Когда сегмент достигает заданного размера, он считается заполненным и запись в него прекращается. То же самое происходит при простое сегмента в течении минуты.
- `FULL` – сегмент заполнен и ждет архивации. Ей занимается отдельный поток, который оповещается с помощью семафора. Также возможна ручная архивация с помощью утилиты `rdblogmgr`.
- `ARCH` – сегмент в процессе архивации.
- `FREE` – помечается после архивации, как возможный к использованию. Как только сегмент `USED` переходит в состояние `FULL`, ищутся сегменты в состоянии `FREE`, чтобы использовать их снова. Если такие не найдены, создается новый.

Сегменты журнала могут быть применены вручную утилитой `rdblogmgr`.

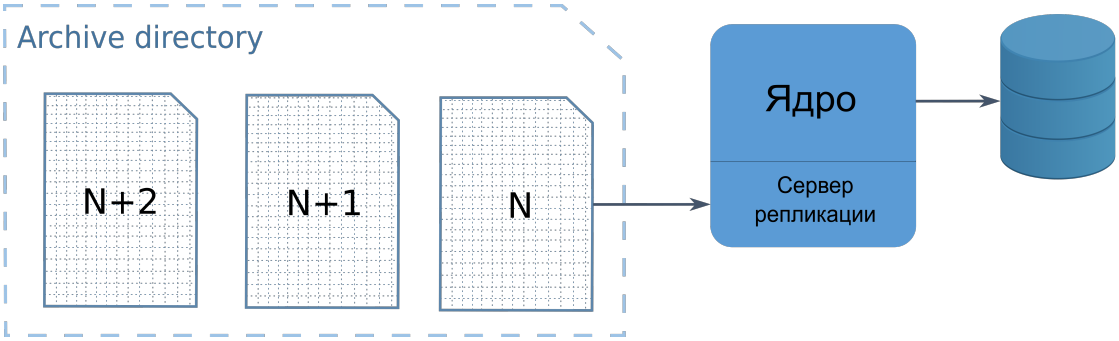
Каждый сегмент имеет уникальный номер, который генерируется последовательно. Этот номер

в сочетании с UUID базы данных обеспечивает глобальную уникальную идентификацию сегментов журнала. Глобальный счетчик последовательности хранится в реплицированной базе данных и никогда не сбрасывается (пока база данных не будет восстановлена из резервной копии).

Сегменты регулярно ротируются. Этот процесс контролируется либо максимальным размером сегмента, либо таймаутом, оба порога настраиваются. Как только активный сегмент достигает порогового значения, он помечается как «полный», и запись переключается на следующий доступный сегмент. Полные сегменты архивируются и затем повторно используются для последующих записей. Архивирование в основном означает копирование сегмента с целью передачи его на хост реплики и применения там. Копирование может быть выполнено самим сервером РЕД Базы Данных или, альтернативно, указанной пользователем командой.

Со стороны реплики сегменты журнала применяются в порядке последовательности репликации. Сервер РЕД Базы Данных периодически сканирует новые сегменты, появляющиеся в настроенном каталоге. Как только следующий сегмент найден, он реплицируется. Состояние репликации хранится в локальном файле с именем `{UUID}` (для каждого источника репликации) и содержит следующие маркеры: последняя последовательность сегментов (LSS), самая старая последовательность сегментов (OSS) и список активных транзакций, запущенных между OSS и LSS. LSS означает последний реплицированный сегмент. OSS означает сегмент, который начал самую раннюю транзакцию, которая не была завершена во время обработки LSS. Эти маркеры управляют двумя вещами: какой сегмент должен быть реплицирован следующим и когда файлы сегмента можно безопасно удалить. Сегменты с номерами между OSS и LSS сохраняются для воспроизведения журнала после отключения репликатора от базы данных реплики (например, из-за ошибки репликации или времени простоя). Если нет ожидающих активных транзакций и LSS был обработан без ошибок, все сегменты включая LSS удаляются. В случае любой критической ошибки репликация временно приостанавливается и повторяется после истечения времени ожидания.

Применение журнала к реплике выполняется сервером репликации. В случае Суперсервера или Суперклассика, это отдельный поток в запущенной СУБД. Он читает архивные сегменты из указанного каталога и применяет их к БД. Если в системе установлен и работает классик, то сервер репликации запускается отдельным процессом суперклассика с ключом `-R`.



12.2.3 Адаптивная репликация

> Адаптивная репликация доступна только в промышленной редакции. Подробнее различия функционала редакций описаны в разделе *Редакции СУБД РЕД База Данных 5.0*.

Адаптивная репликация объединяет в себе два описанных выше режима. Основным преимуществом этого режима является то, что мастер-база остаётся доступной на протяжении всего процесса репликации.

При старте репликации начинается ведение журнала репликации, таким образом выполняется асинхронная репликация. Если со слейв-базой возможно установить соединение и данные на ней актуальны, то включается ещё и синхронная репликация. Если соединение со слейв-базой потеряно, то

данные перестают реплицироваться синхронно. Есть возможность возобновить синхронную репликацию, применив накопившиеся асинхронные журналы к слейв-базе, когда она станет доступна.

Если используется архитектура `Classic`, то для восстановления синхронной репликации необходимо снять всю нагрузку на базу данных. Возобновить нагрузку можно после синхронизации узлов.

Для возобновления синхронной репликации необходимо, чтобы данные на слейве были актуальны. Проверка актуальности данных производится при попытке отправить синхронный репликационный пакет. Если разрыв между мастером и слейвом достигает количества журналов, которое меньше или равно `backoff_lag`, то плагин репликации запустит механизм восстановления синхронизации. Плагин репликации будет замедлять процесс коммита в мастере на `initial_backoff_delay` миллисекунд. Если журналы применились, но данные на слейве всё ещё не актуальны, то задержка будет увеличена в `backoff_factor` раз. Так с каждой попыткой синхронизации время задержки будет расти в `backoff_factor` раз до достижения `max_backoff_delay`. Задержка будет выполняться, пока данные на слейве не станут актуальными. или пока разница в журналах между мастером и репликой не превысит `backoff_lag`.

12.3 Отчет об ошибках

Все ошибки и предупреждения репликации (например, обнаруженные конфликты) записываются в файл `replication.log`, хранящийся в корневом каталоге установки РЕД Базы Данных. Этот файл также может содержать подробное описание операций, выполняемых репликатором.

12.4 Настройка системы репликации

12.4.1 Файл конфигурации

Для настройки системы репликации используется файл `replication.conf`, находящийся в корневом каталоге СУБД. Файл настраивается как на стороне мастер сервера, так и на стороне слейва. Все параметры настройки содержатся в блоке `database{ ... }`, и по умолчанию закомментированы. Настройки можно проводить глобально сразу для всех баз данных и отдельно для каждой базы данных.

При настройке параметров на уровне базы данных в разделе `database` должен быть указан полный путь к базе данных. Псевдонимы и подстановочные знаки не принимаются.

```
database = /your/db.fdb
{ ... }
```

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

Параметры настройки на мастере

plugin

Плагин, используемый для репликации. По умолчанию используется встроенная репликация.

```
plugin =
```

buffer_size
Указывает размер буфера операций на мастер-базе, при достижении которого он будет отправлен на слейв-базы или записан в журнал. Этот параметр не влияет на такие операции, как `commit` или `rollback`; при выполнении этих операций буфер отправляется немедленно и ожидается подтверждение от всех удачного исполнения.

```
buffer_size = 1048576 # 1MB
```

include_filter
Строковый параметр в формате регулярного выражения в синтаксисе SQL, который задает, какие таблицы базы данных необходимо включить в репликацию. По умолчанию реплицируются все таблицы. Любая таблица, которая должна быть реплицирована, должна иметь первичный ключ или, по крайней мере, уникальный ключ.

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

exclude_filter
Строковый параметр в формате регулярного выражения в синтаксисе SQL, который задает, какие таблицы базы данных необходимо исключить из репликации. По умолчанию реплицируются все таблицы. Любая таблица, которая должна быть реплицирована, должна иметь первичный ключ или, по крайней мере, уникальный ключ.

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

exclude_without_pk
Если параметр включен, то из репликации будут исключены таблицы без первичного ключа (или уникального индекса). Если параметр выключен (по умолчанию), то при репликации таблиц без первичного ключа выдается ошибка.

```
exclude_without_pk = true
```

log_errors
Логический параметр, определяющий как должны обрабатываться ошибки при репликации. Если значение параметра `true`, то все ошибки (и предупреждения) будут записываться в `replication.log`.

```
log_errors = true
```

report_errors
Логический параметр, определяющий как должны обрабатываться ошибки при репликации. Если значение параметра `true`, то ошибки будут возвращаться клиентскому приложению.

```
report_errors = false
```

disable_on_error
Логический параметр, определяющий как должны обрабатываться ошибки при репликации. Если значение параметра `true`, то после возникновения ошибки репликация будет прекращена.

```
disable_on_error = false
```

journal_directory ***(только для асинхронного режима)***
Задание этого параметра включает асинхронную репликацию и говорит серверу начать созда-

вать сегменты журнала. Данный параметр указывает на каталог для хранения файлов журнала репликации.

```
journal_directory = /your/db/chlog
```

journal_file_prefix ***(только для асинхронного режима)***
Префикс для имен файлов журнала репликации. Если параметр не указан, то в качестве префикса файлов журнала будет имя файла базы данных (без пути).

```
journal_file_prefix = warehouse_log
```

journal_segment_size ***(только для асинхронного режима)***
Максимально допустимый размер для одного сегмента репликации. Он должен быть, по крайней мере, в два раза больше размера, указанного в `buffer_size`.

```
journal_segment_size = 16777216 # 16MB
```

journal_segment_count ***(только для асинхронного режима)***
Максимально допустимое число полных сегментов репликации. После того, как предел достигнут, процесс репликации задерживается на `journal_archive_timeout` секунд (см.ниже), чтобы догнать процесс архивирования файлов сегментов. Если какой-либо из полных сегментов не архивируется и отмечен для повторного использования в течение тайм-аута, то репликация завершается с ошибкой. Ноль означает неограниченное количество сегментов, ожидающих архивирования.

```
journal_segment_count = 8
```

journal_group_flush_delay ***(только для асинхронного режима)***
Задержка (в миллисекундах) между коммитами для ожидания синхронной записи изменений в журнал. По умолчанию параметр принимает значение 0, то есть во время коммита происходит скидывание буфера в активный сегмент. Этот параметр задает задержку перед скидыванием буфера. Так, коммиты будут писаться в буфер до тех пор, пока он не заполнится или пока не завершится время задержки между коммитами.

```
journal_group_flush_delay = 30
```

journal_archive_directory ***(только для асинхронного режима)***
Данный параметр определяет директорию, куда будут архивироваться сегменты журнала. Для этого значения определена переменная `$(archivepathname)` (см. ниже).

```
journal_archive_directory = /temp/archive_logs
```

journal_archive_command ***(только для асинхронного режима)***
Программа, которая выполняется, когда некоторый сегмент репликации становится полным и нуждается в архивировании. Архивирование выполняется путем копирования сегментов из `journal_directory` в `journal_archive_directory`. Сервер РЕД Базы Данных сам копирует эти сегменты. Можно задать команду архивирования самостоятельно. Эта программа должна возвращать ноль, только если архивирование успешно выполнено. Например, должен вернуться не ноль, если целевой архив уже существует.

Доступны специальные предопределенные переменные:

- `$ (filename)` - имя файла (без пути) сегмента, который нужно заархивировать;
- `$ (pathname)` - полный путь к файлу сегмента, который нужно заархивировать.

Равен `journal_directory + $(filename)`;

- `$ (archivepathname)` - полный путь к файлу заархивированного сегмента.

  Равен `journal_archive_directory + $(filename)`.

```
journal_archive_command = "test ! -f $(archivepathname) && cp $(pathname)
$(archivepathname)"
```

```
journal_archive_command = "copy $(pathname) $(archivepathname)"
```

Пользовательское архивирование позволяет использовать любую shell команду (включая скрипты / пакетные файлы) для доставки сегментов на сторону реплики. Она может использовать сжатие, FTP или что-либо еще доступное на сервере. Реальная реализация доставки зависит от администратора баз данных, РЕД База Данных просто создает сегменты на стороне ведущей базы и ожидает, что они появятся на стороне реплики.

journal_archive_timeout ***(только для асинхронного режима)***

Время ожидания (в секундах) заполнения активного сегмента. Если в течении этого времени не было никаких изменений в базе данных, то текущий сегмент помечается как полный и отправляется на архивацию.

Это позволяет минимизировать разрыв репликации, если база данных редко изменяется. Значение 0 означает отсутствие промежуточного архивирования, т.е. сегменты архивируются только после достижения их максимального размера (определяется `journal_segment_size`).

Используется только для асинхронного режима. По умолчанию значение параметра равно 60.

```
journal_archive_timeout = 60
```

sync_replica ***(только для синхронного режима)***

Задание этого параметра включает синхронную репликацию. Он указывает на базу данных реплики.

Значение параметра можно указывать в двух форматах. В первом варианте значение указывается одной строкой и есть ограничение на использование символа `@` в имени пользователя и пароле:

```
sync_replica = [<пользователь>:<пароль>@]<адрес сервера>:<путь к БД>
```

Во втором варианте `sync_replica` определяется в виде подсекции:

```
sync_replica = <адрес сервера>:<путь к БД>
{
   username[_env | _file] = {<имя пользователя> | <переменная окружения> | <путь
к файлу>}
   password[_env | _file] = {<пароль> | <переменная окружения> | <путь к файлу>}
}
```

При использовании суффикса `_env` значение будет взято из заданной переменной окружения.

Если РЕД База Данных запущена как сервис, то передать переменные окружения нужно одним из следующих способов:

- Задать переменную окружения в файле сервиса:
  1. Укажите переменную окружения в секции `[Service]` файла `firebird.service` в формате: `Environment=<переменная окружения>=<значение>`.
  2. Примените изменения в настройках службы командой `systemctl daemon-reload`.

3. Перезапустите сервис командой `systemctl restart firebird`.

> После обновления СУБД настройки вернутся к значениям по умолчанию.

- Задать переменную окружения в отдельном файле:
  1. Создайте файл `/etc/systemd/system/firebird.service.d/override.conf`.
  2. Укажите переменную окружения в секции `[Service]` в формате: `Environment=<переменная окружения>=<значение>`.
  3. Примените изменения в настройках службы командой `systemctl daemon-reload`.
  4. Перезапустите сервис командой `systemctl restart firebird`.
- Задать путь к файлу с переменными окружения в файле сервиса:
  1. В секции `[Service]` файла `firebird.service` определите путь к файлу, из которого необходимо читать переменные окружения, указав `EnvironmentFile=<путь к файлу>`.
  2. В указанном файле задайте переменные окружения в формате `<переменная окружения>=<значение>`.
  3. Примените изменения в настройках службы командой `systemctl daemon-reload`.
  4. Перезапустите сервис командой `systemctl restart firebird`.

При использовании суффикса `_file` значение будет взято из первой строки определённого файла. Можно указывать как абсолютный, так и относительный путь (относительно корневого каталога инсталляции РЕД Базы Данных). Если такого файла не существует, либо файл пустой, то в лог будут записаны следующие ошибки:

```
"<replica> specifies missing file: <...>/replicaton_pass" -- файл не найден
"<replica> specifies empty file: <...>/replication_pass" -- файл пустой
```

В архитектурах SuperServer и SuperClassic подключение к базе данных реплики происходит при первом соединении пользователя с мастером. А отключение происходит, когда последний пользователь отсоединяется от базы данных мастера.

В архитектуре Classic каждый серверный процесс сохраняет свое собственное активное соединение с базой данных реплики.

sync_reconnect_timeout ***(только для синхронного режима)***

Параметр определяет таймаут (в секундах) для переподключения синхронной репликации в случае потери соединения. Если параметр не задан или указанный таймаут истёк, то в случае потери соединения либо возникнет ошибка, либо данный слейв будет исключён из репликации.

По умолчанию имеет значение 60.

```
sync_reconnect_timeout = 60
```

auto_resync ***(только для адаптивного режима)***

> Только для промышленной редакции.

Параметр `auto_resync` включает использование адаптивной репликации, позволяя переключаться из синхронного режима репликации в асинхронный после потери соединения. И восстановить синхронный режим репликации после восстановления соединения.

Если указано значение `true`, но при этом не настроена синхронная и асинхронная репликация, то

при инициализации репликации будет получено сообщение об ошибке. Также ошибка возникнет, если включить эту опцию в стандартной редакции, поскольку адаптивная репликация доступна только в промышленной версии.

По умолчанию выключено (значение `false`).

```
auto_resync = false
```

backoff_lag ***(только для адаптивного режима)***

> Только для промышленной редакции. Параметр работает, только если `auto_resync = true`.

Целочисленный параметр, определяющий при каком отставании слейва необходимо запускать механизм возобновления синхронной репликации. Если разница между слейв-базой и мастером меньше указанного количества журналов или равна ему, то на сервере будет задержка на время, указанное в параметре `initial_backoff_delay`. По умолчанию имеет значение 20.

```
backoff_lag = 20
```

initial_backoff_delay ***(только для адаптивного режима)***

> Только для промышленной редакции. Параметр работает, только если `auto_resync = true`.

Время в миллисекундах, на которое будет задерживаться сервер (троттлить) для синхронизации слейв-базы с мастером. По умолчанию имеет значение 1000.

```
initial_backoff_delay = 1000
```

backoff_factor ***(только для адаптивного режима)***

> Только для промышленной редакции. Параметр работает, только если `auto_resync = true`.

Параметр определяет во сколько раз необходимо увеличить задержку сервера, если с текущим значением `initial_backoff_delay` синхронизировать реплику с мастер-базой не удалось. По умолчанию имеет значение 1.2.

```
backoff_factor = 1.2
```

max_backoff_delay ***(только для адаптивного режима)***

> Только для промышленной редакции. Параметр работает, только если `auto_resync = true`.

Определяет максимальную задержку сервера в миллисекундах. По умолчанию имеет значение 30000.

```
max_backoff_delay = 30000
```

throttle_timeout ***(только для адаптивного режима)***

Только для промышленной редакции. Параметр работает, только если `auto_resync = true`.

Таймаут (в секундах), по истечении которого троттлинг сервера будет отключён.

Если `throttle_timeout = 0` (по умолчанию), троттлинг будет выполняться до синхронизации между слейв-базой и мастером.

```
throttle_timeout = 0
```

Параметры настройки на слейве

cascade_replication
Если параметр включен, то изменения, примененные к реплике, будут подлежать дальнейшей репликации (если она настроена).

```
cascade_replication = false
```

journal_source_directory ***(только для асинхронного режима)***
Указывает на директорию, где искать архивные журналы для применения к слейву.

```
journal_source_directory = /your/db/incominglog
```

journal_applied_directory
Данный параметр определяет директорию, куда будут записываться примененные журналы. Если каталог не указан, то примененные журналы не будут сохраняться.

journal_applied_count
Определяет количество примененных журналов, которое необходимо хранить. Если лимит достигнут, то старые журналы будут удаляться по мере применения новых. По умолчанию значение 0, то есть журналы не будут удаляться.

```
journal_applied_count = 0
```

source_guid
GUID мастер-базы в формате `{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}`, с которой идет репликация. Его можно узнать из вывода `GSTAT -h`.

```
source_guid = "{6F9619FF-8B86-D011-B42D-00CF4FC964FF}"
```

verbose_logging
Если параметр включен, в `replication.log` будет записан подробный вывод операций, выполняемых сервером репликации. В противном случае (по умолчанию) регистрируются только ошибки и предупреждения.

```
verbose_logging = false
```

apply_idle_timeout ***(только для асинхронного режима)***
Время ожидания (в секундах) перед сканированием новых сегментов репликации. Он используется для приостановки сервера репликации, когда все существующие сегменты уже применены к базе данных реплики, и в указанном каталоге нет новых сегментов.

```
apply_idle_timeout = 10
```

apply_error_timeout ***(только для асинхронного режима)***

Время ожидания (в секундах) перед повторной попыткой применения сегментов в очереди после ошибки. Он используется для приостановки сервера репликации после возникновения критической ошибки во время репликации. В этом случае сервер отключается от базы данных реплики, спит в течение указанного времени ожидания, затем снова подключается и пытается повторно применить последние сегменты с момента сбоя.

```
apply_error_timeout = 60
```

db_copy_command

Программа (или любая `shell` команда), которая выполняется, когда необходимо пересоздать базу данных реплики копированием мастер-базы. Такая программа должна возвращать ноль, только если копирование выполнено успешно.

Доступны следующие предопределённые переменные:

- `$(masterdb)` - строка соединения для подключения к мастер-базе;
- `$(masteruser)` - имя пользователя для подключения к мастер-базе;
- `$(masterpwd)` - пароль для подключения к мастер-базе;
- `$(replicadb)` - путь к базе данных реплики;
- `$(replicauser)` - имя пользователя для подключения к базе данных реплики;
- `$(replicapwd)` - пароль для подключения к базе данных реплики;
- `$(guid)` - `GUID` мастер-базы.

Пример команды, которая делает копию базы с помощью `nbackup`, устанавливает реплику и копирует базу по пути `replicadb`:

Пример для Windows:

```
db_copy_command = "del d:\db\repl.fdb && nbackup -u $(masteruser) -p
$(masterpwd) -b 0 $(masterdb) d:\db\repl.fdb && nbackup -f
d:\db\repl.fdb && gfix -user $(replicauser) -pas $(replicapwd)
-replica $(guid) d:\db\repl.fdb && move /Y d:\db\repl.fdb
$(replicadb)"
```

Пример для Linux:

```
db_copy_command = "export PATH=/opt/RedDatabase/bin:$PATH && rm -f
/db/repl.fdb && nbackup -u $(masteruser) -p $(masterpwd) -b 0
$(masterdb) /db/repl.fdb && nbackup -f /db/repl.fdb && gfix -user
$(replicauser) -pas $(replicapwd) -replica $(guid) /db/repl.fdb && mv
-f /db/repl.fdb $(replicadb)"
```

apply_tablespaces_ddl

Если параметр выключен, то связанные с табличными пространствами DDL операторы и правила внутри `CREATE TABLE`, `ALTER TABLE`, `CREATE INDEX` и `ALTER INDEX` не будут применены к базе данных реплики.

```
apply_tablespaces_ddl = true
```

12.4.2 Инициализация асихронной репликации

1. Остановить СУБД, убедиться, что не осталось процессов сервера в системе.
2. Асинхронная репликация требует настройки механизма журналирования. Основным параметром является `journal_directory`, который определяет местоположение журнала репликации. Как только это местоположение указано, асинхронная репликация включается, и сервер РЕД Базы Данных начинает создавать сегменты журнала.

   В `replication.conf` на мастере прописать блок `database = /путь/к/бд`. В нем задать обязательные параметры:

   - `journal_directory` — каталог с журналами репликации;
   - `journal_archive_directory` — каталог с архивными журналами.

   Учесть, что пользователь, от имени которого запускается СУБД, должен иметь права на запись в оба этих каталога.

   Минимальная настройка будет выглядеть так:

   ```
   database = /data/mydb.fdb
   {
      journal_directory = /dblogs/mydb/
      journal_archive_directory = /shiplogs/mydb/
   }
   ```

   Чтобы применить измененные настройки на мастере, все пользователи должны быть переподключены.
3. В самой базе данных репликация должна быть включена с помощью DDL команды:

   ```
   ALTER DATABASE ENABLE PUBLICATION;
   ```

   А также нужно задать таблицы, которые будут реплицироваться (набор репликации), с помощью DDL команд:

   ```
   /* включение всех таблиц в репликацию, в том числе
      и тех, которые будут созданы в процессе */
   ALTER DATABASE INCLUDE ALL TO PUBLICATION;

   /* включение конкретных таблиц в репликацию */
   ALTER DATABASE INCLUDE TABLE T1, T2, T3 TO PUBLICATION;
   ```

   Чтобы исключить таблицы из процесса репликации, выполните следующие команды:

   ```
   /* исключение всех таблиц из репликации, в том числе и новых */
   ALTER DATABASE EXCLUDE ALL FROM PUBLICATION;

   /* исключение указанных таблиц из репликации */
   ALTER DATABASE EXCLUDE TABLE T1, T2, T3 FROM PUBLICATION;
   ```

   > Таблицы, включенные для репликации, могут быть дополнительно отфильтрованы с помощью двух параметров в файле конфигурации: `include_filter` и `exclude_filter`. Это регулярные выражения, которые применяются к именам таблиц и определяют правила включения таблицы в набор репликации или исключения их из набора репликации.

4. Копировать базу данных на слейв:

- Заблокировать базу данных:

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

- Запустить СУБД на мастере.
- Скопировать заблокированную базу данных на слейв, например, командой:

```
scp -c arcfour
```

- По окончании копирования базу данных на мастере можно разблокировать:

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

5. На реплике установить СУБД РЕД База Данных.
6. На слейве в `replication.conf` прописать блок `database = /путь/к/бд` реплики. В нем задать обязательные параметры:
   - `journal_source_directory` — каталог, в котором СУБД будет искать журналы для вливания. СУБД должна иметь на него права на запись.

   Пример конфигурации выглядит так:

```
database = /data/mydb.fdb
{
   journal_source_directory = /incominglogs/
}
```

   Чтобы применить измененные настройки на стороне реплики, сервер Ред Базы Данных должен быть перезапущен.
7. Разблокировать базу данных на слейве (копия блокированной базы данных является так же блокированной). Но не с параметром `-UN(LOCK)`, а с параметром `-F(IXUP)`:

```
nbackup -F <replica_database>
```

8. Для базы-реплики активируется режим репликации с помощью опции `-replica` утилиты `GFIX`:

```
gfix -replica {read_only|read_write} <replica_database>
```

   Если реплика доступна только для чтения, то базу данных может изменять только процесс репликации. Это главным образом предназначено для обеспечения высокой доступности, поскольку база данных реплики гарантированно совпадает с мастером и может использоваться для быстрого восстановления. Обычные пользовательские соединения могут выполнять любые операции, разрешенные для транзакций только для чтения: выборку из таблиц, выполнять процедуры только для чтения, записывать в глобальные временные таблицы и т.д. Также допускается обслуживание базы данных, такое как сборка мусора, перевод базы в режим `shutdown`, мониторинг.

   Реплики для чтения и записи позволяют одновременно в процессе репликации подключаться обычным пользователям. В этом режиме нет гарантии, что база данных реплики будет синхронизирована с мастером. Поэтому использование реплики для чтения и записи для условий высокой доступности не рекомендуется, если только пользовательские подключения на стороне реплики не ограничены изменением только таблиц, которые исключены из репликации.
9. Обеспечить доставку архивных журналов с мастера на слейв (например, через сетевой каталог NFS).
10. Запустить СУБД на слейве.

    Перевести реплику в режим штатной работы можно командой:

```
gfix -replica none <replica_database>
```

Чтобы изменения настроек репликации, сделанные на мастере, вступили в силу, все пользователи должны быть повторно подключены. Чтобы изменения настроек репликации на слейве были применены, сервер нужно перезапустить.

Пример настройки асинхронной репликации

Для мастера:

```
database = d:\temp\repl\master.fdb
{
    journal_directory = d:\temp\repl\master_log
    journal_file_prefix = repl_test
    journal_segment_size = 1048576 # 1MB
    journal_segment_count = 4
    journal_archive_directory = d:\temp\repl \master_arch
    journal_archive_command = "copy $(pathname) $(archivepathname)"
    journal_archive_timeout = 0
}
```

Для слейва:

```
database = d:\temp\repl\slave.fdb
{
    journal_source_directory = d:\temp\repl\slave_log
    source_guid = "{6F9619FF-8B86-D011-B42D-00CF4FC964FF}"
}
```

12.4.3 Инициализация синхронной репликации

1. Остановить СУБД на мастере, убедиться, что не осталось процессов сервера в системе.
2. В `replication.conf` прописать блок `database = /путь/к/бд`. В нем задать обязательный параметр подключения к реплике в виде:

   ```
   sync_replica = [<login>:<password>@]<database connection string>
   ```

   Их может быть несколько.
3. В самой базе данных репликация должна быть включена с помощью DDL команды:

   ```
   ALTER DATABASE ENABLE PUBLICATION;
   ```

   А также нужно задать таблицы, которые будут реплицироваться (набор репликации), с помощью DDL команд:

   ```
   /* включение всех таблиц в репликацию, в том числе
      и тех, которые будут созданы в процессе */
   ALTER DATABASE INCLUDE ALL TO PUBLICATION;
   ```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
/* включение конкретных таблиц в репликацию */
ALTER DATABASE INCLUDE TABLE T1, T2, T3 TO PUBLICATION;
```

Чтобы исключить таблицы из процесса репликации, выполните следующие команды:

```
/* исключение всех таблиц из репликации, в том числе и новых */
ALTER DATABASE EXCLUDE ALL FROM PUBLICATION;

/* исключение указанных таблиц из репликации */
ALTER DATABASE EXCLUDE TABLE T1, T2, T3 FROM PUBLICATION;
```

Таблицы, включенные для репликации, могут быть дополнительно отфильтрованы с помощью двух параметров в файле конфигурации: `include_filter` и `exclude_filter`. Это регулярные выражения, которые применяются к именам таблиц и определяют правила включения таблицы в набор репликации или исключения их из набора репликации.

4. Существует два способа инициализации реплики: из физической копии и из логической копии.
   - Для инициализации реплики из физической копии необходимо скопировать базу данных на слейв, например, командой:

     ```
     scp -c arcfour
     ```

   - Для инициализации реплики из логической копии необходимо создать резервную копию базы данных:

     ```
     gbak -B <база_данных-источник> <файл резервной копии>
     ```

     При наличии триггеров, способных рассинхронизировать мастер и слейв (например, триггеры на отключение или подключение), нужно использовать опцию `-nod` при создании резервной копии.

     Далее нужно восстановить базу данных из резервной копии на стороне слейва:

     ```
     gbak -C <файл резервной копии> <база_данных>
     ```

5. На реплике установить СУБД РЕД База Данных.
6. На слейве в `replication.conf` прописать блок `database = /путь/к/бд реплики` . В нем задать рекомендуемый параметр `source_guid` — GUID мастер-базы.
7. Для базы-реплики активируется режим репликации с помощью опции `-replica` утилиты `GFIX`:

   ```
   gfix -replica {read_only|read_write} <replica_database>
   ```

   Если реплика доступна только для чтения, то базу данных может изменять только процесс репликации. Это главным образом предназначено для обеспечения высокой доступности, поскольку база данных реплики гарантированно совпадает с мастером и может использоваться для быстрого восстановления. Обычные пользовательские соединения могут выполнять любые операции, разрешенные для транзакций только для чтения: выборку из таблиц, выполнять процедуры только для чтения, записывать в глобальные временные таблицы и т.д. Также допускается обслуживание базы данных, такое как сборка мусора, перевод базы в режим `shutdown`, мониторинг.

   Реплики для чтения и записи позволяют одновременно в процессе репликации подключаться обычным пользователям. В этом режиме нет гарантии, что база данных реплики будет синхронизирована с мастером. Поэтому использование реплики для чтения и записи для условий высокой доступности не рекомендуется, если только пользовательские подключения на стороне реплики не

ограничены изменением только таблиц, которые исключены из репликации.

8. Запустить СУБД на мастере.

При необходимости можно настроить более одной синхронной репликации.

Перевести реплику в режим штатной работы можно командой:

```
gfix -replica none <replica_database>
```

Чтобы изменения настроек репликации, сделанные на мастере, вступили в силу, все пользователи должны быть повторно подключены. Чтобы изменения настроек репликации на слейве были применены, сервер нужно перезапустить.

Пример настройки синхронной репликации

Для мастера:

```
database = d:\temp\repl\master.fdb
{
   sync_replica = sysdba:masterkey@otherhost:d:\temp\repl\slave.fdb
}
```

12.4.4 Инициализация адаптивной репликации

1. Остановить СУБД на мастере, убедиться что не осталось процессов сервера в системе.
2. Определить местоположение журнала репликации.

   В `replication.conf` на мастере прописать блок `database = /путь/к/бд`. В нем задать обязательные параметры:

   - `journal_directory` — каталог с журналами репликации;
   - `journal_archive_directory` — каталог с архивными журналами;
   - `sync_replica` - слейв-база для синхронной репликации, их может быть несколько.

   Учесть, что пользователь, от имени которого запускается СУБД, должен иметь права на запись в каталоги `journal_archive_directory` и `journal_directory`.

   Минимальная настройка будет выглядеть так:

   ```
   database = /data/mydb.fdb
   {
      journal_directory = /dblogs/mydb/
      journal_archive_directory = /shiplogs/mydb/
      sync_replica = [<login>:<password>@]<database connection string>
   }
   ```

   Чтобы применить измененные настройки на мастере, все пользователи должны быть переподключены.
3. В самой базе данных репликация должна быть включена с помощью DDL команды:

   ```
   ALTER DATABASE ENABLE PUBLICATION;
   ```

   А также нужно задать таблицы, которые будут реплицироваться (набор репликации), с помощью DDL команд:

```
/* включение всех таблиц в репликацию, в том числе
   и тех, которые будут созданы в процессе */
ALTER DATABASE INCLUDE ALL TO PUBLICATION;

/* включение конкретных таблиц в репликацию */
ALTER DATABASE INCLUDE TABLE T1, T2, T3 TO PUBLICATION;
```

Чтобы исключить таблицы из процесса репликации, выполните следующие команды:

```
/* исключение всех таблиц из репликации, в том числе и новых */
ALTER DATABASE EXCLUDE ALL FROM PUBLICATION;

/* исключение указанных таблиц из репликации */
ALTER DATABASE EXCLUDE TABLE T1, T2, T3 FROM PUBLICATION;
```

Таблицы, включенные для репликации, могут быть дополнительно отфильтрованы с помощью двух параметров в файле конфигурации: `include_filter` и `exclude_filter`. Это регулярные выражения, которые применяются к именам таблиц и определяют правила включения таблицы в набор репликации или исключения их из набора репликации.

4. Существует два способа инициализации реплики: из физической копии и из логической копии.
   - Для инициализации реплики из физической копии необходимо скопировать базу данных на слейв, например, командой:

     ```
     scp -c arcfour
     ```

   - Для инициализации реплики из логической копии необходимо создать резервную копию базы данных:

     ```
     gbak -B <база_данных-источник> <файл резервной копии>
     ```

     При наличии триггеров, способных рассинхронизировать мастер и слейв (например, триггеры на отключение или подключение), нужно использовать опцию `-nod` при создании резервной копии.

     Далее нужно восстановить базу данных из резервной копии на стороне слейва:

     ```
     gbak -C <файл резервной копии> <база_данных>
     ```

5. На реплике установить СУБД РЕД База Данных.
6. На слейве в `replication.conf` прописать блок `database = /путь/к/бд реплики` . В нем задать обязательные параметры:
   - `source_guid` — GUID мастер-базы;
   - `journal_source_directory` — каталог, в котором СУБД будет искать журналы для вливания. СУБД должна иметь на него права на запись.

   Пример конфигурации:

```
database = /data/mydb.fdb
{
   journal_source_directory = /incominglogs/
   source_guid = "{6F9619FF-8B86-D011-B42D-00CF4FC964FF}"
}
```

7. Для базы-реплики активируется режим репликации с помощью опции `-replica` утилиты `GFIX`:

```
gfix -replica {read_only|read_write} <replica_database>
```

   Если реплика доступна только для чтения, то базу данных может изменять только процесс репликации. Это главным образом предназначено для обеспечения высокой доступности, поскольку база данных реплики гарантированно совпадает с мастером и может использоваться для быстрого восстановления. Обычные пользовательские соединения могут выполнять любые операции, разрешенные для транзакций только для чтения: выборку из таблиц, выполнять процедуры только для чтения, записывать в глобальные временные таблицы и т.д. Также допускается обслуживание базы данных, такое как сборка мусора, перевод базы в режим `shutdown`, мониторинг.

   Реплики для чтения и записи позволяют одновременно в процессе репликации подключаться обычным пользователям. В этом режиме нет гарантии, что база данных реплики будет синхронизирована с мастером. Поэтому использование реплики для чтения и записи для условий высокой доступности не рекомендуется, если только пользовательские подключения на стороне реплики не ограничены изменением только таблиц, которые исключены из репликации.

8. Обеспечить доставку архивных журналов с мастера на слейв (например, через сетевой каталог NFS).
9. Запустить СУБД на мастере.

   Перевести реплику в режим штатной работы можно командой:

```
gfix -replica none <replica_database>
```

Чтобы изменения настроек репликации, сделанные на мастере, вступили в силу, все пользователи должны быть повторно подключены. Чтобы изменения настроек репликации на слейве были применены, сервер нужно перезапустить.

12.5 Переключение на реплику

1. Дождитесь, когда все сегменты будут применены к реплике. На мастере они удалятся автоматически. Узнать расположение каталогов, в которых хранятся эти сегменты, можно в файле настроек репликации на мастере `/opt/RedDatabase/replication.conf`.

   С помощью утилиты `rdbreplmgr` принудительно примените сегменты, которые не влились в реплику:

```
/opt/RedDatabase/bin/rdbreplmgr -a /path/to/slave-db.fdb
```

2. Остановите СУБД на слейве:

```
systemctl stop firebird
```

   Убедитесь, что она остановлена:

```
systemctl status firebird
```

3. Убедитесь, что к базе данных нет подключений:

```
ps -ef|grep rdbserver
sudo lsof /path/to/db.fdb
```

   Если есть подключения, то необходимо дождаться их завершения или принудительно отключить их:

```
sudo /opt/RedDatabase/bin/isql -u <пользователь> -p <пароль>
/path/to/db.fdb
delete from mon$attachments where mon$attachment_id<>current_connection;
commit;
select count(*) from mon$attachments;
exit;
```

Выполните принудительное завершение всех процессов:

```
sudo killall rdbserver
```

4. На слейве закомментируйте настройки репликации в файле `replication.conf`.
5. В зависимости от того, как планируется использовать базу данных, выполните следующее:
    1. Если планируется копировать реплику на другой сервер, который будет использоваться в качестве мастера, то для предотвращения возможных подключений к базе во время копирования необходимо перевести ее в режим `shutdown`:

    ```
    sudo /opt/RedDatabase/bin/gfix -sh full -force 1 /path/to/db.fdb
    ```

    После копирования разблокируйте базу на обоих серверах:

    ```
    sudo /opt/RedDatabase/bin/gfix -online /path/to/db.fdb
    ```

    Снимите режим репликации с базы данных на мастере:

    ```
    sudo /opt/RedDatabase/bin/gfix -replica none -user <пользователь> -
    password <пароль> <путь до скопированной базы>
    ```

    Настройте репликацию на мастере (подробнее см. раздел *Пример настройки асинхронной репликации*).
    2. Если планируется использовать зеркало как мастер, то переведите слейв в состояние мастера:

    ```
    sudo /opt/RedDatabase/bin/gfix -replica none -user <пользователь> -
    password <пароль> <путь до слейва>
    ```

    Настройте параметры мастера в файле `replication.conf`. Измените значение параметра `ServerMode` в файле `firebird.conf`.

12.6 Утилиты

12.6.1 Утилита настройки журнала репликации (rdblogmgr)

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

Для запуска утилиты необходимо выполнить следующую команду:

```
rdblogmgr -D[atabase] <имя_базы_данных> -U[ser] <имя_пользователя> -P[assword]
<пароль>
```

Таблица 12.1 — Опции `rdblogmgr`

Опция	Описание
`-D[atabase] <имя_базы_данных>`	Имя базы данных
`-U[ser] <имя_пользователя>`	Имя пользователя
`-P[assword] <пароль>`	Пароль пользователя
`-G[uid] <guid>`	guid базы данных
`-C[onfig]`	Показывает настройки асинхронной репликации (блок `Log configuration`)
`-S[egments]`	Показывает список используемых сегментов из каталога `journal_directory``(блок ``Available log segments`). Показывает в каком состоянии находится тот или иной сегмент, его размер.
`-A[rchive] <номер_сегмента>`	Архивирует отдельный сегмент журнала
`-A[rchive] all`	Архивирует все сегменты журнала
`-F[orce]`	Принудительно помечает сегмент как полный, даже если он используется.
`-Z`	Показывает версии утилиты и сервера
`-?`	Вывод доступных опций

При каждом запуске утилиты выводится общее состояние журнала (блок `Log status`).

Ручное принудительное архивирование логов можно использовать, если не успевает фоновое архивирование (был сбой связи и сегменты долго не архивировались) или если база мастера вышла из строя и нужно обновить последний сегмент на слейве.

Если сегмент активный, то он будет заархивирован только при указании опции `-F [orce]`.

Приведем пример:

```
rdblogmgr -D /tmp/firebird/RDB-tpcc.fdb -U sysdba -P masterkey -C -S
------------------------------------------------------------------------
Log status:
Current sequence: 2
Last modified: 2015-08-06 12:37:07
Active segment: RDB-tpcc.fdb.log-001, size: 49051
Total log size: 3483922 bytes in 2 segments
Free segments: 0, full segments: 1, archived segments: 0

Log configuration:
Log directory = /tmp/firebird/RDB-tpcc-log/
Log file prefix = RDB-tpcc.fdb
Max segment count = 0
Max segment size = 16777216
Archive directory =
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
Archive command =
Archive timeout = 0

Available log segments:
File name: RDB-tpcc.fdb.log-000
Sequence: 1
State: full
Size in use: 3434871 bytes
File name: RDB-tpcc.fdb.log-001
Sequence: 2
State: used
Size in use: 49051 bytes
```

12.6.2 Утилита для применения файлов асихронной репликации к базе (rdbreplmgr)

Утилита `rdbreplmgr`, ориентированная на слейв, накатывает журналы на реплику в ручном режиме, выводит информацию о состоянии асинхронной репликации, а также создает копию мастер-базы, если определен параметр конфигурации `db_copy_command`.

Для запуска утилиты необходимо выполнить следующую команду:

```
rdbreplmgr <команды> [<опции>] <replica_name>
```

Таблица 12.2 — Команды и опции `rdbreplmgr`

Команды и опции	Описание
`-C[reate]`	Создает копию мастер-базы
`-A[pply]`	Подключается к реплике и применяет журналы из каталога, указанного в `replication.conf`
`-L[og]`	Выводит детальную информацию о журнале
`-S[tatus]`	Выводит информацию о состоянии репликации
`-U[ser] <username>`	Имя пользователя
`-P[assword] <password>`	Пароль пользователя
`-M[aster] <master>`	Путь к мастер-базе
`-G[uid]`	Ручное указание guid-мастера.
`-V[erbose]`	Подробный вывод о проверке
`-Z`	Показывает версии утилиты и сервера
`?`	Помощь

Команда `-Create` выполняет программу, определенную в параметре `db_copy_command`, для пересоздания слейва. Пример такой команды см. *выше*.

Команда `-Log <путь_до_журнала>` выводит информацию об операциях, которые хранятся внутри журнала репликации. Пример выполнения команды `-Log`:

```
Report for log file <путь_до_журнала>:
Version: 1
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
State: arch
GUID: 478F611E-D0CC-418F-5D90-F461CAFA657A
Sequence: 5
Protocol: 2
Length: 2494
Log file segments:
Segment started at 2021-12-21 15:30:28 (1063 bytes):
StartTransaction: att_id=4585, tra_id=32432
StartSavepoint: tra_id=32432
...
```

Для вывода информации о журналах `mc` с помощью `rdbreplmgr -L` необходимо в `/etc/mc/mc.ext` прописать следующее:

```
regex/.log-[0-9]3$
View=%viewascii /opt/RedDatabase/bin/rdbreplmgr -L %p
regex/.arch-[0-9]9$
View=%viewascii /opt/RedDatabase/bin/rdbreplmgr -L %p
```

Команда `-Status` выводит информацию о состоянии репликации. `GUID` определяется параметром `source_guid` в `replication.conf`. Если `source_guid` не указан, то значение будет взято из опции `-Guid` утилиты `rdbreplmgr`. Если опция `-Guid` на задана, то `GUID` запрашивается из мастера, заданного опцией `-Master` утилиты `rdbreplmgr`. Если опция `-Master` не используется, то значение берётся из имени контрольного файла, находящегося в каталоге, который определён параметром `journal_source_directory` в `replication.conf`. Если найдено несколько контрольных файлов, то вызов утилиты завершится ошибкой, сообщающей о необходимости о необходимости указать `GUID`. Пример выполнения команды `-Status`:

```
Status for replica d:\repl\slave_async.fdb:
Master GUID: {232822C2-9A04-498D-0C8B-971C3E0DF421}
Archive directory: d:\temp\repl\logs\
Control file: d:\temp\repl\arch\{232822C2-9A04-498D-0C8B-971C3E0DF421}
Current segment: 2
Oldest segment: absent
Active transaction: 0
Total segments in the queue: 1
```

12.6.3 Утилита проверки целостности данных (rdbrepldiff)

Для выполнения проверки целостности данных после репликации используется утилита `rdbrepldiff`. Суть работы данной утилиты — произвести считывание всех таблиц и всех данных в мастер-базе и слейв-базе и сверить их идентичность. Утилиту можно запускать периодически, когда сервера слабо нагружены или в моменты, когда есть предположения, что данные на мастере и на слейве, в связи с какой-либо ошибкой, отличаются друг от друга и необходимо применить какие-то меры для их синхронизации.

Для того, чтобы проверить целостность базы данных, необходимо выполнить команду:

```
rdbrepldiff [опции] -D[atabase] <master_name> -R[eplica] <replica_name>
```

Спецификация базы данных задается в формате `<адрес сервера>:<путь к БД>`.

Таблица 12.3 — Опции `rdbrepldiff`

Опции	Описание
`-U[ser] <username>`	Имя пользователя
`-P[assword] <password>`	Пароль пользователя
`-C[onfig] <config>`	Имя измененного файла конфигурации для возможности сравнивать не все сущности базы данных, а только необходимые. При этом процесс репликации не прерывается
`-M[etadata]`	Производит проверку только метаданных
`-Si[mple]`	Сравнение двух нереплицируемых баз данных
`-S[ynchronize]`	Ожидание синхронизации мастера и слейва в течение одной минуты или времени, указанного в параметре `-Timeout`
`-N[odbtriggersoff]`	Включает ипользование триггеров при подключении к мастеру
`-T[imeout]`	Тайм-аут ожидания синхронизации (в секундах)
`-V[erbose]`	Подробный вывод о проверке
`-Z`	Показать версии утилиты и сервера
`-F`	Показать расширенный вывод о проверке. Расширенный вывод отображает: • Записи, которые отсутсвуют на слейве; • Поля, которые отличаются у записей; • Разницу в количестве записей на мастере и на слейве.
`-Va[lues]`	Показывает значения, которые отличаются у записей. Можно использовать только совместно с опцией `-F`
`?`	Помощь

Для проверки должен быть указан адрес мастер-сервера и адрес слейв-сервера. После запуска утилита подключается к ним (одновременно) и выполняется чтение метаданных обеих баз. Выбирается первая таблица и идет чтение из таблицы обеих баз, сравниваются записи. Если в одной из таблиц оказалось больше данных или если данные не совпадают, то таблицы считаются не идентичными (выдается сообщение об ошибке). Этот процесс продолжается для всех остальных таблиц. Если все таблицы идентичны, то делается вывод, что базы идентичны.

При сравнении двух нереплицируемых баз данных с помощью `-Simple` спецификацию базы данных задавать не нужно. Если у сравниваемых баз отличается пользователь и/или пароль, то необходимо передать его в формате `[<логин>:<пароль>@]<путь к второй БД>`. Такой расширенный путь можно указывать ещё и для асинхронно реплицируемых баз данных.

Например:

```
server2:/my/replica/database.fdb
```

```
john:smith@server2:/my/replica/database.fdb
```

При сравнении простых баз данных сравниваются все системные таблицы, а при сравнении реплик пропускаются таблицы без уникального индекса.

Таблица 12.4 — Теги для запуска `rdbrepldiff` через сервис

Теги	Описание
`-action_svc_diff`	Запуск сравнения
`-dbname`	Первая база для сравнения (должна быть локальной)
`-dif_replica`	Вторая база (реплика)
`-dif_config`	Путь к файлу конфигурации. Если не задан, то ищется в папке с базой
`-dif_metaonly`	Производит проверку только метаданных
`-dif_simple`	Сравнение двух нереплицируемых баз данных
`-dif_sync`	Ожидание синхронизации мастера и слейва в течение одной минуты или времени, указанного в параметре `-dif_timeout`
`-dif_timeout`	Тайм-аут ожидания синхронизации (в секундах)
`-dif_full`	Показать расширенный вывод о проверке. Расширенный вывод отображает: • Записи, которые отсутсвуют на слейве; • Поля, которые отличаются у записей; • Разницу в количестве записей на мастере и на слейве.
`-dif_values`	Показывает значения, которые отличаются у записей. Можно использовать только совместно с опцией `dif_full`
`-verbose`	Подробный вывод о проверке

Пример запуска `rdbrepldiff` через сервис:

```
./rdbsvcmgr localhost:service_mgr -user SYSDBA -password masterkey -action_diff
-dbname localhost:/my/other/database.fdb -dif_simple -dif_replica john:smith@/my/
replica/database2.fdb
```

Глава 13

Настройка производительности РЕД Базы Данных

Производительность конкретной системы зависит от аппаратного и программного обеспечения. Причины плохой производительности могут быть с обеих сторон - слабая дисковая подсистема, мало памяти, плохое управление транзакциями в приложении, плохие запросы и т.д.

В первом разделе будет описано, как подобрать эффективное аппаратное обеспечение для РЕД Базы Данных. Далее даются рекомендации по диагностике производительности системы. Заключительная часть главы посвящена проблемам производительности и пути их устранения.

13.1 Выбор аппаратного обеспечения

Прежде чем подобрать эффективное аппаратное обеспечение для БД, следует понять, как РЕД База Данных использует его компоненты: `CPU`, `RAM`, `HDD/SSD`.

13.1.1 Основные операции взаимодействия БД с аппаратным обеспечением

При старте СУБД процесс сервера занимает в `RAM` минимальный объем (несколько мегабайт) и не производит никаких интенсивных операций с `CPU` или `RAM`.

При соединении с базой данных сервер запрашивает у ОС оперативную память в размере страничного кэша базы. Фактически эта память сразу не используется, но выделение её осуществляется сразу при первом подключении к базе. Потом сервер начинает читать метаданные базы. `CPU` практически не задействован на этом этапе.

Когда клиент начинает выполнять SQL-запросы (включая хранимые процедуры), сервер выполняет соответствующие операции, обращающиеся к аппаратной части. Среди этих операций можно выделить следующие базовые операции, потребляющие определенный набор системных ресурсов. Они представлены в таблице ниже с указанием интенсивности потребления (1 означает небольшую интенсивность, 10 – максимальную):

Таблица 13.1 — Потребление системных ресурсов

	Чтение страниц БД с диска	**Запись страниц БД на диск**	**Чтение страниц БД из кэша**	**Запись страниц БД в кэш**	**Чтение страниц данных из GTT**	**Запись страниц данных в GTT**	**Сортировка записей**	**Обработка SQL запроса**
CPU	1	1	1	1	1	1	5	10
RAM	5	5	5	5	5	5	5	2
Disc IO	10	10	1	1	1	1	1	1

Видно, что наиболее тяжелыми операциями являются те, которые включают работу с диском, так как дисковая подсистема является наиболее медленным компонентом аппаратной части.

13.1.2 Процессор

При выборе процессора следует принять во внимание следующие вещи:

Преобладание сложных запросов

СУБД всегда исполняет один запрос на одном ядре, поэтому сложные или плохо оптимизированные запросы могут занимать до 100% одного ядра, заставляя остальные запросы переместиться на менее загруженные ядра. Поэтому чем больше ядер, тем меньше шанс, что все процессорные мощности будут заняты.

Если преобладают простые SQL-запросы и все запросы хорошо отлажены, то CPU не будет являться узким местом производительности, и можно выбрать модель с меньшим количеством ядер.

Если преобладают медленные запросы, возвращающие большое количество данных, то необходим процессор с большим количеством ядер.

Число активных соединений

Для грубой оценки необходимого количества ядер в CPU можно пользоваться правилом: от 10 до 30 соединений на 1 ядро. 10 соединений/ядро – приложение с преобладанием сложных и медленных запросов, 30 соединений/ядро – приложение с преобладанием простых, хорошо отлаженных запросов.

13.1.3 Память

При выборе RAM следует уделить внимание двум моментам:

Модули памяти должны быть с коррекцией ошибок (ECC RAM)

ECC RAM значительно снижает количество ошибок при работе с памятью и настоятельно рекомендуется для использования в промышленных системах.

Правильно рассчитать объем RAM

РЕД База Данных 5.0 архитектуры Classic запускает отдельный процесс для обслуживания каждого соединения, SuperClassic запускает отдельный поток для каждого соединения, но практически с той же структурой потребления памяти - каждое соединение имеет свой независимый страничный кэш. SuperServer запускает один процесс с общим страничным кэшем для всех соединений с каждой базой данных. При подключении к нескольким базам для каждой из них будет выделен свой объем страничного кэша, заданный настройками в файлах конфигурации или в самой базе данных.

На потребление памяти влияют следующие параметры:

1. Количество соединений.
2. Размер объектов метаданных (пропорционален количеству таблиц, триггеров, хранимых процедур и др.).
3. Размер страничного кэша (определяется параметрами в заголовке БД или в firebird.conf или в свойствах конкретного соединения).
4. Размер кэша для сортировок (определяется параметром в firebird.conf).
5. Размер таблицы блокировок (важнее для Classic/SuperClassic).
6. Объекты СУБД в памяти (запросы, блобы, управляющие структуры и т.д.).

Ниже представлены формулы приблизительного расчета необходимого объема памяти для РЕД Базы Данных:

- для Classic:

```
Количество соединений * (Кол-во страниц в кэше * Размер страницы + Размер кэша
для сортировок)
```

- для SuperClassic:

```
Количество соединений * (Кол-во страниц в кэше * Размер страницы) + Размер кэша
для сортировок
```

- для SuperServer:

```
(Кол-во страниц в кэше * Размер страницы) + Размер кэша для сортировок
```

Реальное потребление памяти может отличаться, так как в этом расчете не учитывается объем памяти под метаданные, под битовые маски индексов, и т.д., что может увеличить расход памяти, но одновременно предполагается, что память под сортировки будет использована полностью во всех соединениях, чего обычно не происходит.

Количество RAM сверх рассчитанного минимального будет эффективно использоваться операционной системой для кэширования файла БД.

Когда база данных уже находится в эксплуатации, можно просто посмотреть средний размер памяти, используемый процессом СУБД.

13.1.4 Дисковая подсистема

Чтобы уменьшить конкуренцию за дисковый ввод-вывод между операциями с файлом БД, сортировками и резервными копиями, а также уменьшить шанс одновременной потери и базы данных и резервных копий, рекомендуется иметь 3 разных диска (или raid-массива):

Для базы данных

Для работы с базой данных лучше всего использовать SSD-диски, так как они обеспечивают высокую скорость произвольного доступа. Обязательно следует использовать диски промышленного класса с увеличенным числом циклов перезаписи, иначе велик риск потери данных из-за поломки SSD. SSD с наличием Enhanced Power Loss Data Protection существенно повысит производительность.

При этом рекомендуется оставлять до 30% свободного места на диске (из-за повышенного износа) и планировать замену диска примерно раз в 3 года.

Наилучшим выбором является использование SSD эксклюзивно для работы с базой данных, так как любые операции ввода-вывода сокращают срок службы дисков.

Если SSD диск оказывается слишком дорогим решением или размер БД слишком велик, то можно подобрать альтернативные варианты в порядке уменьшения приоритета:

1. HDD с интерфейсом SAS.
2. Диски SATA с интерфейсом nSAS.
3. Обычные диски SATA.

Для временных файлов

Так как временные файлы на диске возникают только при отсутствии достаточного количества RAM, то лучше всего вообще избегать их появления на диске.

Но всё-таки РЕД База Данных требует указания папки, где будут храниться временные файлы (параметры TempTableDirectory и TempDirectories). Обычно этот параметр оставляют со значением по умолчанию, т.е. используется стандартный временный каталог ОС. Чтобы предупредить исчерпание свободного места на системном диске, в firebird.conf указывают второй диск в качестве дополнительного резервного места.

Следует учитывать, что при создании индекса размер данных сортировки может быть очень большим и СУБД будет выгружать её данные на диск. Это может привести к созданию больших файлов (десятки гигабайт для больших таблиц) в каталогах, заданных в TempDirectories.

Аналогичная ситуация с большими временными файлами в десятки гигабайт может произойти при использовании очень больших временных таблиц (GTT) в каталоге TempTableDirectory.

Для резервных копий

При резервном копировании происходит чтение файла базы данных (всего или части), и запись резервной копии (полной или частичной). Операции записи при создании резервной копии идут последовательно, это означает, что обычные недорогие жесткие диски с интерфейсом SATA (HDD SATA) хорошо подойдут для хранения резервных копий, так как скорость последовательной записи у них довольно велика.

На диске для резервных копий всегда должно оставаться свободное место в размере последнего бэкапа + 10%. В этом случае есть возможность создать свежий бэкап, убедиться в корректном завершении процесса копирования и только потом удалить предыдущий бэкап.

Под «отдельными дисками» понимается, что физически потоки данных должны идти через разные каналы ввода-вывода. Если создать 3 логических диска на одном физическом диске, никакого улучшения производительности не произойдет. Однако, если 3 логических диска будут организованы на устройстве хранения данных (СХД), оснащенном многоканальными контроллерами, то производительность может возрасти, так как устройство может распределять потоки данных между контроллерами.

При наличии в контроллере дисковой подсистемы возможности кэширования записи (режим write back) желательно её включить, но только при условии гарантированного сохранения данных кэша в случае потери питания (использование BBU, кэш на базе flash и т.д.). Если сохранение данных кэша не гарантировано, его следует отключить (режим write through).

13.1.5 RAID

Для базы данных и бэкапов следует увеличивать надежность дисковой подсистемы путем объединения дисков в RAID.

Для SSD дисков следует обязательно использовать RAID1 - это 2 "зеркальных" диска, на которые одновременно пишутся изменения, что значительно уменьшает шанс полной потери данных.

Для HDD дисков, используемых для бэкапов, достаточно использовать RAID1, который обеспечит надежное хранение бэкапов и приемлемую скорость записи и чтения.

HDD диски, используемые для БД, необходимо объединять в RAID10 (минимум 4 диска), который обеспечит оптимальное сочетание стоимости, надежности и производительности.

13.2 Диагностика системы

Чтобы собрать информацию о производительности системы, следует выполнить следующие шаги:

- *Измерить параметр IOPS произвольной записи*, используя IOMeter на Windows и fio на Linux
- *Измерить параметр IOPS произвольного чтения.*
- *Измерить скорость последовательной записи*:
- *Дисковая подсистема* довольно часто становится узким местом в работе. Поэтому очень важно уметь диагностировать проблемы с дисками. Одним из основных инструментов для наблюдения за производительностью дисковой подсистемы являются счетчики производительности:
  - % Disk Write Time (процент загруженности диска операциями записи)
  - % Disk Read Time (процент загруженности диска операциями чтения)
  - Avg. Disk sec/Write (среднее время в секундах, требуемое для выполнения диском одной операции записи)
  - Avg. Disk sec/Read (среднее время в секундах, требуемое для выполнения диском одной операции чтения)
  - Disk Read Bytes/sec (средняя скорость чтения)
  - Disk Reads/sec (количество обработанных за секунду запросов на чтение)

  - `Disk Write Bytes/sec` (средняя скорость записи)
  - `Disk Writes/sec` (количество обработанных за секунду запросов на запись)
- *Включить аудит событий*, изменив параметры в конфигурационном файле `fbtrace.conf`:
  - `enabled = true`,
  - `time_threshold = 1000`,
  - `log_statement_finish = true`.

  и проверить полученные с помощью него логи.
- *Собирать данные менеджера блокировок* с регулярными интервалами:

```
rdb_lock_print -a -d </path/to/db> > db_lock_print.txt
```

- *Мониторить характеристики процессора*, используя `CPU-Z`. Диспетчеру задач Windows доверять не стоит.
- С помощью утилиты `gstat` *проверить нет ли старых активных транзакций*, которые препятствуют сборке мусора:

```
gstat -h </path/to/db>
```

- *Проверить таблицы мониторинга* или сохранить результаты этих таблиц во внешние файлы:

```
MON$DATABASE;MON$ATTACHMENTS;MON$TRANSACTIONS;MON$STATEMENTS; MON$IO_STATS; MON
$RECORD_STATS;
MON$MEMORY_USAGE;
```

13.3 Узкие места и их устранение

13.3.1 Узкое место записи данных на диск с БД

Признаки

- Высокий показатель `% Disk Write Time (>75%)`
- Среднее значение счетчика `Disk Writes/sec` близко или превышает параметр `IOPS` произвольной записи.
- Запросы `Insert/update/delete`, которые обрабатывают небольшое количество строк и обычно выполняются за время менее 100 мсек, выполняются за одну и более секунды и появляются в журнале аудита.

Рекомендации

- Убедиться, что количество кэшируемых страниц достаточно большое. Для архитектуры суперсервер значение параметра `DefaultDbCachePages` рекомендуется рассчитать по формуле:

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

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

  Для архитектуры классик значение параметра рекомендуется рассчитать по формуле:

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

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

Установить другое значение кэшируемых страниц можно с помощью утилиты gfix:

```
GFIX -buffers 400 <db_name>
```

- Улучшить показатели параметра IOPS произвольной записи можно применением следующих мер:
  1. Избегайте применять RAID5. Лучше использовать RAID1 или RAID10.
  2. Используйте SSD для хранения БД.
- Разобраться какие запросы выполняют столько IO-операций (с помощью таблиц мониторинга и аудита).
- Как временный, запасной вариант, можно установить режим асинхронной записи данных - в этом случае изменения и новые данные хранятся в памяти и периодически сбрасываются на диск подсистемой ввода/вывода операционной системы. Общепринятое название такой конфигурации - forced writes off. По умолчанию БД устанавливается с включенным режимом принудительной записи - forced writes - синхронная запись. В этом режиме изменения и новые данные сразу записываются на диск.

  Затем в конфигурационном файле firebird.conf отключить параметры MaxUnflushedWrites и MaxUnflushedWriteTime.

13.3.2 Узкое место чтения данных с диска БД

Признаки

- Высокий показатель % Disk Read Time (> 75%)
- Среднее значение счетчика Disk Reads/sec близко или превышает параметр IOPS произвольного чтения, измеряемый в течение нескольких минут.
- Запросы на чтение и обновление выполняются чрезвычайно долго и появляются в журнале аудита.

Рекомендации

- Убедитесь, что ОС можно использовать всю оперативную память для кэширования. Для этого можно проверить установленные границы размера файла системного кэша в результате использования утилиты SetSystemFileCacheSize. Она должна показывать:

```
Current cache working set size
min: none set
max: none set
```

  Если утилита показывает что-то иное, установите FileSystemCacheSize=0 в конфигурационном файле firebird.conf. При внесении изменений потребуется перезагрузка.
- Убедитесь, что достаточно оперативной памяти доступно для кэширования.
- Определите, какие запросы выполняют столько IO-операций (с помощью таблиц мониторинга и аудита).

13.3.3 Проблемы с троттлингом процессора (только для Windows)

Признаки

- Измерения с помощью CPU-Z показали, что тактовая частота процессора ниже заявленной.

Когда система перегружена, все остальные операции начинают занимать больше времени, и система все больше склоняется к образованию "узких мест" в других местах (подсистемах, уровнях) из-за взаимных блокировок.

Рекомендации

- В разделе Электропитание панели управления Windows выберите план «Высокая производительность». Далее «Настройка плана электропитания» → «Изменить дополнительные параметры питания», далее выберите пункт «Управление питанием процессора», где «Минимальное состояние процессора» должно быть 100%. С помощью CPU-Z убедитесь, что проблема решена.
- Если предыдущий шаг не помог, значит BIOS или прошивка снизили частоту процессора. Прочтите руководство Server's BIOS и выберите настройки, которые соответствуют параметру «Performance Optimized». Возможно потребуется отключить Intel SpeedStep в BIOS.
- Не исключено, что тактовая частота процессора снижена из-за проблем с охлаждением. Измерьте температуру процессора и чипсета (например, с помощью утилиты Open Hardware Monitor).

13.3.4 Проблемы с сортировкой

При выполнении сортировок РЕД База Данных выполняет ее в памяти (в адресном пространстве процесса сервера), пока размер используемой памяти для всех выполняемых одновременно сортировок не достигнет предела, установленного параметром TempCacheLimit в firebird.conf. При превышении этого лимита создается временный файл (с соответствующим флагом операционной системы) в папке временных файлов, и в нем выполняется сортировка. В случае, если в системе есть свободная память (RAM), то файл сортировки будет кэширован на уровне ОС и сортировка будет производиться в памяти. Измерения показывают, что запросы с большими сортировками выполняются в 2 раза быстрее в ОЗУ, а не на диске. Поэтому рекомендуется установить параметр TempCacheLimit = 2147483647 в firebird.conf для 64-битных систем и TempCacheLimit = 1000000000 для 32-битных систем.

Информацию о том, сколько памяти занимают временные файлы, можно узнать из таблицы мониторинга MON$STATEMENTS. Также, при включенном логировании операций типа finish, в трейсе отразится объём кэша, выделенного для сортировки.

Имя временного файла формируется следующим образом:

```
<тип временного объекта>_att<id подключения>_stmt<id запроса>_<случайная строка>
```

Например: /tmp/rdb_sort_att342316_stmt106_M2OSLu.

13.3.5 Проблемы со сборкой мусора

Признаки

- Большая разница между «Oldest transaction» и «Next transaction» при выводе статистики: gstat -h
- Большая разница между Oldest active (или Oldest Snapshot) и Next transaction.

- Запросы выполняются неожиданно долго.

Удержание `Oldest active` и `Oldest Snapshot` возможны по следующим причинам:

- некоторое приложение работает и держит открытой хотя бы одну транзакцию, например, пользователь не закрыл приложение и какая-либо его форма очень долго открыта;
- приложения работают длительное время, и «теряют» идентификаторы транзакций в коде;
- используемая библиотека компонент или драйвер работает с «транзакцией по умолчанию», которая может длиться очень долго.

Рекомендации

- Посмотреть таблицы мониторинга, чтобы узнать, какая транзакция активна больше всего времени, принять необходимые меры по ее закрытию и провести принудительную сборку мусора `gfix -sweep n <db_name>`.
- Сборку мусора (`sweep`) выполнять по крайней мере раз в неделю (`gfix -sweep n <db_name>`). Можно запланировать запуск команды с помощью планировщика Windows.
- Выполнять периодическое резервное копирование-восстановление БД по крайней мере раз в год или после крупного изменения в данных.

13.3.6 Длительное ожидание блокировок

Признаки

- Запросы выполняются неожиданно долго.
- В менеджере блокировок высокий процент (более `20%`) попыток (`Mutex wait`), которые были заблокированы, когда владелец старался обратиться к таблице блокировок.

Рекомендации

- Обычно причина высокого показателя `Mutex wait` заключается в недостаточно оптимальной структуре хэш-таблицы. Не рекомендуется, чтобы среднее значение `«Hash lengths»` было больше 3 или максимальное больше 10. Рекомендуемое число слотов (`LockHashSlots`): 65521. В качестве значения размера хэш-таблицы лучше указывать простое число.

13.3.7 Неэффективное использование таблиц мониторинга

Признаки

- Запросы выполняются неожиданно долго.
- Дамп лок-таблицы показывает цепочки ожидания для блокировки 19-го типа (`series 19`).

Сбор информации для таблиц мониторинга требует глобальной блокировки базы данных и является достаточно длительной операцией. Не рекомендуется обращаться к таблицам мониторинга на рабочем сервере чаще, чем раз в несколько минут.

Рекомендации

- Автоматические процессы, процедуры или триггеры не должны обращаться таблицам мониторинга так часто. Эта проблема должна быть перенаправлена на разработчиков приложений для исправления.

13.3.8 Проблемы с блокировками транзакций

Признаки

- DML-запросы выполняются неожиданно долго.
- Дамп лок-таблицы показывает цепочки ожидания для блокировки 4го типа.

Эта проблема вызвана одновременным обновлением одинаковых наборов записей в режиме WAIT разрешения блокировок или неправильным выбором параметров транзакций (например, использование параметра no_rec_version).

Рекомендации

- Этот вопрос должен быть переадресован разработчикам приложения на исправление.

13.3.9 Узкое место в подсистеме памяти

Признаки

- Запросы, обрабатывающие большие объемы данных, выполняются неожиданно долго.
- Большое потребление ресурсов центрального процессора, где сервер является основным их потребителем.
- Мониторинг процессора показывает, что процент времени нахождения в режиме ядра высок (20% и более).

Причина таких признаков заключается в том, что во время чтения страницы БД должны быть скопированы с кэша файловой системы на буферный кэш РЕД Базы Данных. Существуют ограничения в пропускной способности памяти, что особенно заметно в крупных многопроцессорных конфигурациях, тогда каналы обмена с памятью становятся узким местом.

Рекомендации

- Исключить проблемы уровня ОС.
- Уменьшить размер страниц (с 16K до 8K) и увеличить размер буферного кэша.
- Оптимизировать проблемные запросы (проверить по таблицам мониторинга и с помощью аудита).
- Убедиться в правильной настройке кэша сервера (возможно он слишком мал), проверить параметры конфигурационного файла:
  - TempBlockSize - можно увеличить до 2 или 3Мб, но не до 16Мб;
  - TempCacheLimit - сделать значение больше суммарного размера временных файлов в каталогах TempDirectories;
  - TempDirectories - каталоги для хранения данных сортировок указываются через точку с запятой, следуют в порядке очереди, пока не заполнится первая директория, вторая не будет использоваться. Указать первой директорией RAM диск, потом SSD или HDD;
  - TempTableDirectory задать каталог для хранения данных глобальных временных таблиц и временных блобов.
- Достигнуты границы масштабируемости ядра Windows. Переходите на ОС, которая позволяет полное профилирование системы (Linux, Unix).

13.3.10 Проблемы с блокировкой страниц

Признаки

- Запросы выполняются неожиданно долго.
- Дамп лок-таблицы показывает цепочки ожидания для блокировки 3-го типа (`series 3`).

Это проблема могла быть вызвана выше описанными ситуациями: узкое место записи данных на диск с БД, длительное ожидание блокировок, узкое место в подсистеме памяти.

Если проблема не в этом, то причины могут быть следующие:

- много одновременных соединений выполняют крупные изменения в разных записях одинаковых таблиц (может быть выведено из журнала аудита).
- чрезвычайно высокий рост числа транзакций, генерируемых приложениями (может быть выведено из быстрого роста счетчиков транзакций).

Если и эти причины исключены, то можно получить номера страниц, используя ключ `Key` - идентификатор заблокированно ресурса, и выяснить, какие именно страницы вызывают проблемы и какие таблицы, индексы или другие структуры являются узким местом.

Рекомендации

- Разрешить проблемы, описанные выше.
- Перевести задачу на разработчиков приложения для того, чтобы они могли исключить проблемы, возникающие при одновременных изменениях в таблицах или высоком росте числа транзакций.
- Можно обратиться в техническую поддержку РЕД Базы Данных, если все указанные причины исключены, а проблема осталась.

13.3.11 Долго выполняемые запросы

Признаки

- Пользователи отмечают плохую производительность, зависание клиента или зависание всей системы, если запрос выполняется сервером, когда удерживается некоторая блокировка.
- В случае зависания проблемный запрос можно увидеть в таблице `MON$STATEMENTS` со строкой `MON$STATE <> 0`.
- Проблемы с производительностью при определенных запросах.
- В таблице мониторинга `MON$RECORD_STATS` количество неиндексных чтений гораздо больше, чем индексных (`MON$RECORD_SEQ_READS > MON$RECORD_IDX_READS`).

Рекомендации

- Если определили проблемный запрос в `MON$STATEMENTS`, завершите его удалением соответствующей строки.
- Запустите запрос отдельно с включенным отображением плана и статистики:

```
set stats on;
set plan on;
<запрос>
.........................
Current memory = 2574604
Delta memory = -5660
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
Max memory = 2653012
Elapsed time= 0.03 sec
Cpu = 0.000 sec
Buffers = 512
Reads = 1
Writes 0
Fetches = 92
```

`Current memory`
: Объём памяти, используемой сервером (в байтах).

`Delta memory`
: Показывает, на сколько байт изменилось использование памяти (значение `Current memory`) за время выполнения запроса.

`Max memory`
: Максимальное использование памяти, которое было зафиксировано с момента запуска сервера.

`Elapsed time`
: Время выполнения запроса.

`Cpu`
: Время использования `CPU`.

`Buffers`
: Количество кэшированных страниц.

`Reads`
: Показатель дискового `I/O`, сколько страниц считано с диска в кэш сервера (тех, которые еще не находились в кэше). Первое выполнение запроса показывает точные данные. При повторном вызове, если `Reads = 0`, то все нужные страницы уже в кэше, а если значение больше 0, то в кэше не хватает места, чтобы поместить необходимые страницы, то есть на самом деле считано страниц `Buffers + Reads`.

`Writes`
: Страницы, записанные из кэша на диск (или в кэш операционной системы). `Insert`, `update`, `delete` - влияют на этот показатель, также влияет сборка мусора при `select`.

`Fetches`
: Индикатор операций с памятью (включая `CPU`). Сколько было обращений к странице в кэше, чтение записей, чтение ключей, чтение страниц указателей, и т.п.

- Опыт показывает, что наиболее значительные проблемы с производительностью вызваны отсутствием нужных индексов. Проверьте, существуют ли индексы, которые может использовать данный запрос. Если нет - постройте их.

  При наличие нескольких индексов оптимизатор выберет имеющие лучшую селективность и отбросит индексы, имеющие худшую, поэтому необходимо ее периодически пересчитывать (вручную или по расписанию).

13.3.12 Проблема с Антивирусом

Признаки

- Система медленно работает.

Антивирус перехватывает большинство системных вызовов, поэтому может вызвать замедление в совершенно неожиданных местах. Наблюдалось замедление ввода-вывода, замедление сети, замедления доступа к памяти.

Рекомендации

- Отключите или удалите антивирус. Если производительность улучшилась, попробуйте выборочно отключить антивирус для временной папки СУБД.

13.3.13 Производительность файловой системы

Признаки

- Показатель `IOPS` производительности системы хранения данных значительно медленнее, чем производительность самого устройства. Можно ожидать значения `IOPS` для каждого `SSD` по крайней мере 5000 (для пары, если они "зеркальные") и по крайней мере 130 `IOPS` для каждого жесткого диска (для пары, если они "зеркальные").

Рекомендации

- Рекомендуется иметь размер кластера файловой системы, совпадающий с размером страницы базы данных.
- Для многоуровневых конфигураций (например виртуализация) должны быть оценены накладные расходы на каждом уровне в плане задержек и пропускной способности операций ввода-вывода. Конфигурация должна быть настроена, чтобы обеспечить приемлемый уровень производительности.

Глава 14

PSQL-профайлер

14.1 Общие сведения

Профайлер позволяет пользователям измерять затраты на производительность кода SQL и PSQL. Это реализовано с помощью системного пакета, передающего данные в плагин профайлера. В этом тексте части сервера и плагина рассматриваются как единое целое, так же, как будет использоваться профайлер по умолчанию (Default_Profiler).

Пакет RDB$PROFILER позволяет профилировать выполнение кода PSQL, собирая статистику, о том сколько раз была выполнена каждая строка, а также ее минимальное, максимальное и общее время выполнения (с точностью до наносекунд). Также он дает доступ к статистике явных и неявных SQL-курсоров.

Чтобы собрать данные, необходимо запустить сеанс профайлера с помощью RDB$PROFILER.START_SESSION. Эта функция возвращает идентификатор сеанса профайлера, который позже сохраняется в таблицах снапшотов профайлера для запроса и анализа. Сеанс профайлера может быть локальным (то же подключение) или удаленным (другое подключение).

Удаленное профилирование пересылает команды удаленному подключению. Возможно одновременно профилировать несколько подключений. Также возможно, что локально или удаленно запущенный сеанс профайлера содержит команды, переданные другим подключением.

Удаленно переданные команды требуют, чтобы целевое подключение находилось в состоянии ожидания, то есть не выполняло другие запросы. Когда оно не простаивает, вызов команд блокируется в ожидании этого состояния.

Если удаленное подключение происходит от другого пользователя, то вызывающий пользователь должен иметь системные привилегии PROFILE_ANY_ATTACHMENT.

После запуска сеанса статистика PSQL и SQL операторов начинает собираться в памяти. Обратите внимание, что сеанс профайлера собирает данные только об операторах, выполняемых в подключении, связанном с сеансом.

Данные агрегируются и сохраняются для каждого запроса. При запросе таблиц снапшотов можно выполнять дополнительную агрегацию для каждого оператора или использовать вспомогательные представления, которые делают это автоматически.

Сеанс может быть приостановлен, чтобы временно отключить сбор статистики. Он может быть возобновлен позже, чтобы вернуть сбор статистики в том же сеансе.

Новый сеанс может быть запущен, когда сеанс уже активен. В этом случае он имеет ту же семантику завершения текущего сеанса с помощью RDB$PROFILER.FINISH_SESSION(FALSE), поэтому таблицы снапшотов не обновляются в то же время.

Чтобы проанализировать собранные данные, необходимо сбросить данные в таблицы снапшотов, что может быть сделано при завершении или приостановке сеанса (с параметром FLUSH, установленным в TRUE) или вызовом RDB$PROFILER.FLUSH. Данные удаляются с помощью автономной транзакции (транзакция, запущенная и завершенная для конкретной цели обновления данных профайлера).

14.1.1 Накладные расходы, возникающие во время работы профайлера

Во время работы профайлера возникают накладные расходы, которые замедляют работу сервера. Эти расходы зависят от операционной системы, версии ядра и модели процессора, их трудно предсказать. Но иногда они могут быть очень высокими, например, более 100%.

Если такие накладные расходы возникают в Linux, то с помощью следующей команды вы можете увидеть, какой источник сигнала тактовой частоты используется для синхронизации:

```
cat /sys/devices/system/clocksource/clocksource0/current_clocksource
```

Если результат отличается от `tsc`, это может быть причиной проблемы.

Еще одна возможная причина замедления в Linux - эта ошибка.

14.1.2 Пример сеанса профайлера

Ниже приведен пример сеанса профайлера и запросов для анализа данных:

```
/* Preparation - create table and routines that will be analyzed */

create table tab (
   id integer not null,
   val integer not null
);

set term !;

create or alter function mult(p1 integer, p2 integer) returns integer
as
begin
   return p1 * p2;
end!

create or alter procedure ins
as
   declare n integer = 1;
begin
   while (n <= 1000)
   do begin
      if (mod(n, 2) = 1) then
         insert into tab values (:n, mult(:n, 2))
      n = n + 1;
   end
end!

set term ;!

/* Start profiling */

select rdb$profiler.start_session(\rr{Profile Session 1}) from rdb$database;

set term !;

execute block
as
begin
   execute procedure ins;
   delete from tab;
end!
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
set term ;!

execute procedure rdb$profiler.finish_session(true);
execute procedure ins;

select rdb$profiler.start_session(\rr{Profile Session 2}) from rdb$database;

select mod(id, 5), sum(val)
from tab
where id <= 50
group by mod(id, 5)
order by sum(val);

execute procedure rdb$profiler.finish_session(true);

/* Data analysis */

set transaction read committed;

select * from plg$prof_sessions;

select * from plg$prof_psql_stats_view;

select * from plg$prof_record_source_stats_view;

select preq.*
from plg$prof_requests preq
join plg$prof_sessions pses
   on pses.profile_id = preq.profile_id and
      pses.description = 'Profile Session 1';

select pstat.*
from plg$prof_psql_stats pstat
join plg$prof_sessions pses
   on pses.profile_id = pstat.profile_id and
      pses.description = 'Profile Session 1'
order by pstat.profile_id,
         pstat.request_id,
         pstat.line_num,
         pstat.column_num;

select pstat.*
from plg$prof_record_source_stats pstat
join plg$prof_sessions pses
   on pses.profile_id = pstat.profile_id and
      pses.description = 'Profile Session 2'
order by pstat.profile_id,
         pstat.request_id,
         pstat.cursor_id,
         pstat.record_source_id;
```

14.2 Функция START_SESSION

RDB$PROFILER.START_SESSION запускает новый сеанс профайлера, превращает его в текущий сеанс (с заданным ATTACHMENT_ID) и возвращает его идентификатор.

Если значение FLUSH_INTERVAL не NULL, то автосброс настраивается таким же образом, как и при вызове RDB$PROFILER.SET_FLUSH_INTERVAL.

Если значение PLUGIN_NAME равно NULL (по умолчанию), то будет использован плагин профайлера конфигурации базы данных по умолчанию.

PLUGIN_OPTIONS - это параметры, настраиваемые для плагина, в данном случае для Default_Profiler должно быть значение NULL.

Входные параметры:

- DESCRIPTION - тип VARCHAR(255) CHARACTER SET UTF8 по умолчанию NULL;
- FLUSH_INTERVAL - тип INTEGER по умолчанию NULL;
- ATTACHMENT_ID - тип BIGINT NOT NULL по умолчанию CURRENT_CONNECTION;
- PLUGIN_NAME - тип VARCHAR(255) CHARACTER SET UTF8 по умолчанию NULL;
- PLUGIN_OPTIONS - тип VARCHAR(255) CHARACTER SET UTF8 по умолчанию NULL.

Возвращает тип BIGINT NOT NULL.

14.3 Процедура PAUSE_SESSION

RDB$PROFILER.PAUSE_SESSION приостанавливает текущий сеанс профайлера (с заданным ATTACHMENT_ID), поэтому статистика следующих выполненных инструкций не собирается.

Если значение FLUSH равно TRUE, таблицы снапшотов обновляются данными до текущего момента. В противном случае данные остаются только в памяти для последующего обновления.

Вызов RDB$PROFILER.PAUSE_SESSION(TRUE) имеет ту же семантику, что и вызов RDB$PROFILER.PAUSE_SESSION(FALSE), за которым следует RDB$PROFILER.FLUSH (используя тот же ATTACHMENT_ID).

Входные параметры:

- FLUSH тип BOOLEAN NOT NULL по умолчанию FALSE;
- ATTACHMENT_ID тип BIGINT NOT NULL по умолчанию CURRENT_CONNECTION.

14.4 Процедура RESUME_SESSION

RDB$PROFILER.RESUME_SESSION возобновляет текущий сеанс профайлера (с заданным ATTACHMENT_ID), если он был приостановлен, запуская снова сбор статистики.

Входные параметры:

- ATTACHMENT_ID тип BIGINT NOT NULL по умолчанию CURRENT_CONNECTION.

14.5 Процедура FINISH_SESSION

RDB$PROFILER.FINISH_SESSION завершает текущий сеанс профайлера (с заданным ATTACHMENT_ID).

Если значение FLUSH равно TRUE, таблицы снапшотов обновляются данными завершенного сеанса (и старых завершенных сеансов, которые еще не присутствуют в снапшоте). В противном случае данные остаются только в памяти для последующего обновления.

Вызов RDB$PROFILER.FINISH_SESSION(TRUE) имеет ту же семантику, что и вызов RDB$PROFILER.FINISH_SESSION(FALSE), за которым следует RDB$PROFILER.FLUSH (используя тот же ATTACHMENT_ID).

Входные параметры:

- FLUSH тип BOOLEAN NOT NULL по умолчанию TRUE;
- ATTACHMENT_ID тип BIGINT NOT NULL по умолчанию CURRENT_CONNECTION.

14.6 Процедура CANCEL_SESSION

RDB$PROFILER.CANCEL_SESSION отменяет текущий сеанс профайлера (с заданным ATTACHMENT_ID).

Все данные сеанса, присутствующие в плагине профайлера, удаляются и не будут сброшены.

Данные, которые уже были сброшены, не удаляются автоматически.

Входные параметры:

- ATTACHMENT_ID тип BIGINT NOT NULL по умолчанию CURRENT_CONNECTION.

14.7 Процедура DISCARD

RDB$PROFILER.DISCARD удаляет все сеансы (с заданным ATTACHMENT_ID) из памяти, не сбрасывая их данные.

Если есть активный сеанс, он завершается.

Входные параметры:

- ATTACHMENT_ID тип BIGINT NOT NULL по умолчанию CURRENT_CONNECTION.

14.8 Процедура FLUSH

RDB$PROFILER.FLUSH обновляет таблицы снапшотов данными из сеансов профайлера (с заданным ATTACHMENT_ID) в памяти.

После обновления данные сохраняются в таблицах PLG$PROF_SESSIONS, PLG$PROF_STATEMENTS, PLG$PROF_RECORD_SOURCES, PLG$PROF_REQUESTS, PLG$PROF_PSQL_STATS и PLG$PROF_RECORD_SOURCE_STATS и могут быть прочитаны и проанализированы.

Данные обновляются с помощью автономной транзакции, поэтому, еесли процедура вызывается в снапшот-транзакции, данные не будут считываться в той же транзакции.

Как только происходит обновление, завершенные сеансы удаляются из памяти.

Входные параметры:

- ATTACHMENT_ID тип BIGINT NOT NULL по умолчанию CURRENT_CONNECTION.

14.9 Процедура SET_FLUSH_INTERVAL

RDB$PROFILER.SET_FLUSH_INTERVAL включает (FLUSH_INTERVAL > 0) или выключает (FLUSH_INTERVAL = 0) периодический сброс данных. FLUSH_INTERVAL - количество секунд.

Входные параметры:

- FLUSH_INTERVAL тип INTEGER NOT NULL;
- ATTACHMENT_ID тип BIGINT NOT NULL по умолчанию CURRENT_CONNECTION.

14.10 Таблицы снапшотов

Таблицы снапшотов (а также представления и последовательность) автоматически создаются при первом использовании профайлера. Они принадлежат текущему пользователю с правами на чтение/запись для PUBLIC. Когда сеанс завершается, связанные данные в других таблицах снапшотов профайлера также автоматически удаляются с помощью внешних ключей с опцией DELETE CASCADE.

Ниже приведен список таблиц, в которых хранятся данные профайлера.

PLG$PROF_SESSIONS

Идентификатор столбца	Тип данных	Описание
PROFILE_ID	BIGINT	Идентификатор сеанса профайлера.
ATTACHMENT_ID	BIGINT	Идентификатор подключения.
USER_NAME	CHAR(63) CHARACTER SET UTF8	Имя пользователя.
DESCRIPTION	VARCHAR(255) CHARACTER SET UTF8	Описание, переданное в RDB$PROFILER.START_SESSION.
START_TIMESTAMP	TIMESTAMP WITH TIME ZONE	Момент запуска сессии профайлера.
FINISH_TIMESTAMP	TIMESTAMP WITH TIME ZONE	Момент, когда сессия профайлера была завершена (NULL, если не завершена).

PLG$PROF_STATEMENTS

Идентификатор столбца	Тип данных	Описание
PROFILE_ID	BIGINT	Идентификатор сеанса профайлера.
STATEMENT_ID	BIGINT	Идентификатор оператора.
PARENT_STATEMENT_ID	BIGINT	Идентификатор родительского оператора - связан с подпрограммами.
STATEMENT_TYPE	VARCHAR(20) CHARACTER SET UTF8	Блок, функция, процедура или триггер.
PACKAGE_NAME	CHAR(63) CHARACTER SET UTF8	Пакет функции или процедуры.
ROUTINE_NAME	CHAR(63) CHARACTER SET UTF8	Имя функции, процедуры или триггера.
SQL_TEXT	BLOB	SQL-текст для BLOCK.

PLG$PROF_CURSORS

Идентификатор столбца	Тип данных	Описание
PROFILE_ID	BIGINT	Идентификатор сеанса профайлера.
STATEMENT_ID	BIGINT	Идентификатор оператора.

(разрыв таблицы)

(разрыв таблицы)

Идентификатор столбца	Тип данных	Описание
CURSOR_ID	INTEGER	Идентификатор курсора.
NAME	CHAR(63) CHARACTER SET UTF8	Имя явного курсора.
LINE_NUM	INTEGER	Номер строки оператора.
COLUMN_NUM	INTEGER	Номер столбца оператора.

PLG$PROF_RECORD_SOURCES

Идентификатор столбца	Тип данных	Описание
PROFILE_ID	BIGINT	Идентификатор сеанса профайлера.
STATEMENT_ID	BIGINT	Идентификатор оператора.
CURSOR_ID	BIGINT	Идентификатор курсора.
RECORD_SOURCE_ID	BIGINT	Идентификатор источника записи.
PARENT_RECORD_SOURCE_ID	BIGINT	Идентификатор источника родительской записи.
ACCESS_PATH	VARCHAR(255) CHARACTER SET UTF8	План запроса.

PLG$PROF_PSQL_STATS

Идентификатор столбца	Тип данных	Описание
PROFILE_ID	BIGINT	Идентификатор сеанса профайлера.
REQUEST_ID	BIGINT	Идентификатор запроса.
LINE_NUM	INTEGER	Номер строки оператора.
COLUMN_NUM	INTEGER	Номер столбца оператора.
STATEMENT_ID	BIGINT	Идентификатор оператора.
COUNTER	BIGINT	Количество выполненных раз строки/столбца.
MIN_ELAPSED_TIME	BIGINT	Минимальное затраченное время (в наносекундах) выполнения строки/столбца.
MAX_ELAPSED_TIME	BIGINT	Максимальное затраченное время (в наносекундах) выполнения строки/столбца.
TOTAL_ELAPSED_TIME	BIGINT	Общее затраченное время (в наносекундах) выполнения строки/столбца.

PLG$PROF_RECORD_SOURCE_STATS

Идентификатор столбца	Тип данных	Описание
PROFILE_ID	BIGINT	Идентификатор сеанса профайлера.
REQUEST_ID	BIGINT	Идентификатор запроса.
CURSOR_ID	BIGINT	Идентификатор курсора.
RECORD_SOURCE_ID	BIGINT	Идентификатор источника записи.
STATEMENT_ID	BIGINT	Идентификатор оператора.
OPEN_COUNTER	BIGINT	Количество раз открытия источника записи.
OPEN_MIN_ELAPSED_TIME	BIGINT	Минимальное время открытия источника записи (в наносекундах).
OPEN_MAX_ELAPSED_TIME	BIGINT	Максимальное время открытия источника записи (в наносекундах).
OPEN_TOTAL_ELAPSED_TIME	BIGINT	Общее время открытия источника записи (в наносекундах).
FETCH_COUNTER	BIGINT	Количество выборок из источника записи.
FETCH_MIN_ELAPSED_TIME	BIGINT	Минимальное время выборки из источника записи (в наносекундах).
FETCH_MAX_ELAPSED_TIME	BIGINT	Максимальное время выборки из источника записи (в наносекундах).
FETCH_TOTAL_ELAPSED_TIME	BIGINT	Общее время выборки из источника записи (в наносекундах).

PLG$PROF_REQUESTS

Идентификатор столбца	Тип данных	Описание
PROFILE_ID	BIGINT	Идентификатор сеанса профайлера.
STATEMENT_ID	BIGINT	Идентификатор оператора.
REQUEST_ID	BIGINT	Идентификатор запроса.
CALLER_STATEMENT_ID	BIGINT	Идентификатор оператора (STATEMENT_ID), вызвавшего текущий запрос.
CALLER_REQUEST_ID	BIGINT	Идентификатор запроса (REQUEST_ID), вызвавшего текущий запрос.
START_TIMESTAMP	TIMESTAMP WITH TIME ZONE	Момент запуска сессии профайлера.
FINISH_TIMESTAMP	TIMESTAMP WITH TIME ZONE	Момент, когда сессия профайлера была завершена (NULL, если не завершена).
TOTAL_ELAPSED_TIME	BIGINT	Общее время выполнения запроса (в наносекундах).

14.11 Дополнительные представления

Эти представления помогают извлекать данные, агрегированные на уровне операторов. Они должны быть предпочтительным способом анализа собранных данных. Их также можно использовать вместе с таблицами для получения дополнительных данных, отсутствующих в представлениях.

PLG$PROF_STATEMENT_STATS_VIEW

```
select req.profile_id,
       req.statement_id,
       sta.statement_type,
       sta.package_name,
       sta.routine_name,
       sta.parent_statement_id,
       sta_parent.statement_type parent_statement_type,
       sta_parent.routine_name parent_routine_name,
      (
         select sql_text
         from plg$prof_statements
         where profile_id = req.profile_id and
               statement_id = coalesce(sta.parent_statement_id, req.statement_id)
      ) sql_text,
      count(*) counter,
      min(req.total_elapsed_time) min_elapsed_time,
      max(req.total_elapsed_time) max_elapsed_time,
      cast(sum(req.total_elapsed_time) as bigint) total_elapsed_time,
      cast(sum(req.total_elapsed_time) / count(*) as bigint) avg_elapsed_time

from plg$prof_requests req
join plg$prof_statements sta
      on sta.profile_id = req.profile_id and
         sta.statement_id = req.statement_id
left join plg$prof_statements sta_parent
      on sta_parent.profile_id = sta.profile_id and
         sta_parent.statement_id = sta.parent_statement_id
group by req.profile_id,
         req.statement_id,
         sta.statement_type,
         sta.package_name,
         sta.routine_name,
         sta.parent_statement_id,
         sta_parent.statement_type,
         sta_parent.routine_name
order by sum(req.total_elapsed_time) desc
```

PLG$PROF_PSQL_STATS_VIEW

```
select pstat.profile_id,
       pstat.statement_id,
       sta.statement_type,
       sta.package_name,
       sta.routine_name,
       sta.parent_statement_id,
       sta_parent.statement_type parent_statement_type,
       sta_parent.routine_name parent_routine_name,
       (select sql_text
        from plg$prof_statements
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
         where profile_id = pstat.profile_id and
               statement_id = coalesce(sta.parent_statement_id, pstat.statement_id)
       ) sql_text,
       pstat.line_num,
       pstat.column_num,
       cast(sum(pstat.counter) as bigint) counter,
       min(pstat.min_elapsed_time) min_elapsed_time,
       max(pstat.max_elapsed_time) max_elapsed_time,
       cast(sum(pstat.total_elapsed_time) as bigint) total_elapsed_time,
       cast(sum(pstat.total_elapsed_time) / nullif(sum(pstat.counter), 0) as bigint)
                                                                    avg_elapsed_time
from plg$prof_psql_stats pstat
join plg$prof_statements sta
      on sta.profile_id = pstat.profile_id and
         sta.statement_id = pstat.statement_id
left join plg$prof_statements sta_parent
      on sta_parent.profile_id = sta.profile_id and
         sta_parent.statement_id = sta.parent_statement_id
group by pstat.profile_id,
         pstat.statement_id,
         sta.statement_type,
         sta.package_name,
         sta.routine_name,
         sta.parent_statement_id,
         sta_parent.statement_type,
         sta_parent.routine_name,
         pstat.line_num,
         pstat.column_num
order by sum(pstat.total_elapsed_time) desc
```

PLG$PROF_RECORD_SOURCE_STATS_VIEW

```
select rstat.profile_id,
       rstat.statement_id,
       sta.statement_type,
       sta.package_name,
       sta.routine_name,
       sta.parent_statement_id,
       sta_parent.statement_type parent_statement_type,
       sta_parent.routine_name parent_routine_name,
       (select sql_text
        from plg$prof_statements
        where profile_id = rstat.profile_id and
              statement_id = coalesce(sta.parent_statement_id, rstat.statement_id)
       ) sql_text,
       rstat.cursor_id,
       rstat.record_source_id,
       recsrc.parent_record_source_id,
       recsrc.access_path,
       cast(sum(rstat.open_counter) as bigint) open_counter,
       min(rstat.open_min_elapsed_time) open_min_elapsed_time,
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
        max(rstat.open_max_elapsed_time) open_max_elapsed_time,
        cast(sum(rstat.open_total_elapsed_time) as bigint) open_total_elapsed_time,
        cast(sum(rstat.open_total_elapsed_time) / nullif(sum(rstat.open_counter), 0)
                                                  as bigint) open_avg_elapsed_time,
        cast(sum(rstat.fetch_counter) as bigint) fetch_counter,
        min(rstat.fetch_min_elapsed_time) fetch_min_elapsed_time,
        max(rstat.fetch_max_elapsed_time) fetch_max_elapsed_time,
        cast(sum(rstat.fetch_total_elapsed_time) as bigint) fetch_total_elapsed_time,
        cast(sum(rstat.fetch_total_elapsed_time) / nullif(sum(rstat.fetch_counter), 0)
                                                   as bigint) fetch_avg_elapsed_time,
        cast(coalesce(sum(rstat.open_total_elapsed_time), 0) +
             coalesce(sum(rstat.fetch_total_elapsed_time), 0) as bigint)
                                                          open_fetch_total_elapsed_time
from plg$prof_record_source_stats rstat
join plg$prof_record_sources recsrc
      on recsrc.profile_id = rstat.profile_id and
         recsrc.statement_id = rstat.statement_id and
         recsrc.cursor_id = rstat.cursor_id and
         recsrc.record_source_id = rstat.record_source_id
join plg$prof_statements sta
      on sta.profile_id = rstat.profile_id and
         sta.statement_id = rstat.statement_id
left join plg$prof_statements sta_parent
      on sta_parent.profile_id = sta.profile_id and
         sta_parent.statement_id = sta.parent_statement_id
group by rstat.profile_id,
         rstat.statement_id,
         sta.statement_type,
         sta.package_name,
         sta.routine_name,
         sta.parent_statement_id,
         sta_parent.statement_type,
         sta_parent.routine_name,
         rstat.cursor_id,
         rstat.record_source_id,
         recsrc.parent_record_source_id,
         recsrc.access_path
order by coalesce(sum(rstat.open_total_elapsed_time), 0) +
         coalesce(sum(rstat.fetch_total_elapsed_time), 0) desc
```

Глава 15

Коннектор CDC

15.1 Настройка коннектора CDC

Коннектор CDC позволяет отслеживать изменения на сервере РЕД Базы Данных в таблицах на уровне строк и генерировать соответствующие события вставки, обновления и удаления. События для каждой таблицы записываются в отдельную тему `Kafka`, где они могут быть использованы приложениями и сервисами. Коннектор должен быть включен локально на каждом сервере РЕД Базы Данных, где требуется отслеживание изменений в таблицах. После включения коннектор начинает ослеживать изменения на сервере и публиковать изменения в отдельных темах `Kafka`. Коннектор CDC для РЕД Базы Данных отличается от других коннекторов `Debezium` - он не реализуется поверх механизма `Kafka Connect`. Коннектор представляет собой встроенный в процесс сервера `JVM-процесс`, публикующий события в `Kafka` с помощью продюсера `Kafka`. Коннектор реализован на основе механизма репликации сервера РЕД Базы Данных, что позволяет отслеживать изменения без создания дополнительных нагрузок на сервер. Для работы коннектора необходимо настроить механизм репликации на сервере РЕД Базы Данных.

Коннектор реализуется в виде плагина репликации `Debezium_Connector`, использующем `JAVA` библиотеки платформы `Debezium` с открытым исходным кодом для сбора измененных данных. Плагин реализован в виде нативной `dll/so` библиотеки, используемой для инициализации плагина, и `java-библиотеки`, реализующей работу коннектора. Для включения коннектора плагин должен быть доступен в конфигурации плагинов `plugins.conf`, а также в конфигурации репликации `replication.conf` в секции `database` в настройке `plugin` должен быть указан `Debezium_Connector`. Параметры коннектора задаются в конфигурационном файле `reddatabase.connector.yaml`.

Для работы коннектора в конфигурационном файле плагинов `plugins.conf` необходимо добавить секции плагина коннектора:

```
Plugin = Debezium_Connector {
        Module = $(dir_plugins)/debezium_connector
        Config = Debezium_Connector_config
}

Config = Debezium_Connector_config {
        JarDirs = $(this)/debezium
}
```

- `Plugin` - Плагин, необходимый для работы коннектора (`Debezium_Connector`).
- `Module` - Определяет путь к нативной библиотеке коннектора. По умолчанию она находится в каталоге `plugins` сервера и имеет имя `libdebezium_connector.so/debezium_connector.ddl`. Если нативная библиотека имеет имя по умолчанию, то значением параметра можно оставить `$(dir_plugins)/debezium_connector`.
- `Config` - Определяет секцию параметров плагина в формате `<имя секции>_config`. Имя секции может быть любым. По умолчанию `Debezium_Connector_config`.
- `JarDirs` - Путь к каталогу с `Java-плагином` относительно каталога сервера. По умолчанию имеет значение `$(this)/debezium`.

Если сервер РЕД Базы Данных установлен инсталятором, то для включения коннектора достаточно раскомментировать секции `Plugin = Debezium_Connector` и `Config = Debezium_Connector_config`.

Настройка репликации осуществляется в файле `replication.conf`, находящегося в каталоге сер-

вера РЕД Базы Данных. Необходимые настройки репликации для базы или баз данных, в которых требуется отслеживать изменения:

```
database
{
        plugin = Debezium_Connector
}
```

В конфгирукции репликации по умолчанию включена секция `database`, но в ней не настроены никакие параметры репликации. Секция `database` применяется ко всем базам данных. Для включения коннеткора необходимо в секции `database` в параметре `plugin` указать `Debezium_Connector`.

Если требуется отслеживание определенной базы данных (или нескольких баз), то для неё должна быть указана секция `database` с указанием пути к базе:

```
database = /your/db.fdb
{
        plugin = Debezium_Connector
}

database = /var/db/rdb.fdb
{
        plugin = Debezium_Connector
}
```

При настройке параметров на уровне базы данных в разделе `database` должен быть указан полный путь к базе данных. Псевдонимы и подстановочные знаки не допускаются. После настройки сервера необходимо его перезапустить.

После настройки репликации в файле `replication.conf` необходимо в каждой базе данных, для которой требуется отслеживание изменений, включить репликацию и добавить таблицы для отслеживания в публикации. Включение репликации в базе данных с помощью `DDL` команды:

```
ALTER DATABASE ENABLE PUBLICATION;
```

Включение таблицы в список реплицируемых осуществляется следующей `DDL` командой:

```
ALTER DATABASE INCLUDE TABLE <имя таблицы> TO PUBLICATION
```

Можно добавить все доступные таблицы (в том числе и те, которые будут созданы в процессе) в список реплицируемых:

```
ALTER DATABASE INCLUDE ALL TO PUBLICATION
```

Подробнее о фильтрации таблиц в репликации см. раздел *Фильтрация таблиц и столбцов*.

Последним этапом настройки нужно указать конфигурацию коннектора РЕД Базы Данных в файле `jvm.args`, если не используется конфигурационный файл по умолчанию. Для этого в `jvm.args` необходимо добавить строку с парамтером `-Dfirebird.connector.config=`:

```
-Dfirebird.connector.config=/var/config/custom.rdb.connector.yaml
```

Конфигурация коннектора может располагаться в любом каталоге, к которому есть права доступа у сервера РЕД Базы Данных. Доступные параметры коннектора описаны в разделе ниже.

15.2 Параметры коннектора

В конфигурационном файле `reddatabase.connector.yaml` настраиваются следующие параметры коннектора и продюсера `Kafka`:

- `connector.name` - Имя коннектора. Обязательный параметр.

  `Тип: string`

  `Значение по умолчанию: ""`

- `connector.journal.directory` - Каталог для записи журналов репликации. Обязательный параметр.

  `Тип: string`

  `Значение по умолчанию: ""`

- `connector.journal.check.interval.ms` - Период просмотра наличия журналов транзакций в миллисекундах. Обязательный параметр.

  `Тип: long`

  `Значение по умолчанию: 0`

  `Допустимые значения: [0,...9223372036854775807]`

- `kafka.producer.bootstrap.servers` - Список пар хост/порт, используемых для установления начального соединения с кластером `Kafka`. Клиенты используют этот список для загрузки и обнаружения полного набора брокеров `Kafka`. Хотя порядок серверов в списке не имеет значения, рекомендуется указывать несколько серверов, чтобы обеспечить устойчивость в случае падения какого-либо сервера. Этот список не обязательно должен содержать весь набор брокеров, так как клиенты `Kafka` автоматически управляют и обновляют соединения с кластером. Список имет вид: `host1:port1,host2:port2,.....` Поскольку эти сервера используются только для начального подключения для обнаружения полного состава кластера, который может динамически меняться, список не обязательно должен содержать полный набор серверов.

  `Тип: list`

  `Значение по умолчанию: ""`

- `kafka.producer.retries` - При значении больше нуля клиент будет повторно отправлять любую запись, отправка которой не удалась из-за случайной ошибки. Обратите внимание, что такая повторная отправка ничем не отличается от того, если бы клиент повторно отправил запись после получения ошибки. Разрешение повторных попыток без установки `max.in.flight.requests.per.connection` равным 1 потенциально изменит порядок записей, поскольку если две партии отправляются в один раздел, и первая не удалась и была повторно отправлена, а вторая удалась, то записи из второй партии могут появиться первыми. Кроме того, запросы продюсера будут отклонены до того, как будет исчерпано количество повторных попыток, если таймаут, настроенный в параметре `delivery.timeout.ms`, истечет до успешного подтверждения. Обычно предпочитают вместо него использовать `delivery.timeout.ms` для управления поведением повторных попыток.

  `Тип: int`

  `Значение по умолчанию: 2147483647`

  `Допустимые значения: [0,...,2147483647]`

- `kafka.producer.retry.backoff.ms` - Время ожидания перед попыткой повторного запроса к данному тематическому разделу. Это позволяет избежать многократной отправки запросов в замкнутом цикле при некоторых сценариях отказа.

  `Тип: long`

  `Значение по умолчанию: 100`

  `Допустимые значения: [0,...,9223372036854775807]`

- kafka.producer.transactional.id - Используется для доставки транзакций. Обеспечивает семантику надежности, охватывающую несколько сессий Producer, поскольку позволяет клиенту гарантировать, что транзакции, использующие тот же transactional.id, были завершены до начала любых новых транзакций. Если transactional.id не указан, то продюсер ограничивается идемпотентной доставкой. Если transactional.id настроен, то подразумевается enable.idempotence. По умолчанию transactional.id не настроен, что означает, что транзакции не могут быть использованы.

  Тип: string

  Значение по умолчанию: null

- kafka.producer.max.block.ms - Время, в течение которого будут блокироваться методы send(), partitionsFor(), initTransactions(), sendOffsetsToTransaction(), commitTransaction() и abortTransaction() KafkaProducer. Для send() этот таймаут ограничивает общее время ожидания выборки метаданных и выделения буфера (блокировка в пользовательских сериализаторах или разделителе не учитывается в этом таймауте). Для partitionsFor() этот таймаут ограничивает время ожидания метаданных, если они недоступны. Методы, связанные с транзакциями, всегда блокируются, но могут прерваться, если координатор транзакций не может быть обнаружен или не отвечает в течение таймаута

  Тип: long

  Значение по умолчанию: 60000 (1 минута)

  Допустимые значения: [0,...,9223372036854775807]

- kafka.producer.delivery.timeout.ms - Верхняя граница времени сообщения об успехе или неудаче после возврата вызова send(). Это ограничивает общее время задержки записи перед отправкой, время ожидания подтверждения от брокера (если ожидается), а также время, допустимое для повторных попыток отправки. Продюсер может сообщить о неудачной отправке записи раньше указанного значения, если возникла неисправимая ошибка, исчерпаны все повторные попытки, или запись добавлена в пакет, у которого истекает срок доставки раньше. Значение этого параметра должно быть больше суммы request.timeout.ms и linger.ms или равно ей.

  Тип: int

  Значение по умолчанию: 120000 (2 минуты)

  Допустимые значения: [0,...,2147483647]

- kafka.producer.request.timeout.ms - Параметр контролирует максимальное время, в течение которого клиент будет ожидать ответа на запрос. Если ответ не получен до истечения таймаута, клиент повторно отправит запрос, если это необходимо, или отклонит запрос, если количество повторных попыток исчерпано. Это значение должно быть больше, чем replica.lag.time.max.ms (конфигурация брокера), чтобы уменьшить вероятность дублирования сообщений из-за ненужных повторных попыток продюсера.

  Тип: int

  Значение по умолчанию: 30000 (30 секунд)

  Допустимые значения: [0,...,2147483647]

- kafka.producer.close.timeout.ms - Время в миллисекундах, в течение которого продюсер завершает отправку запросов в темы kafka, по истечении указанного времени продюсер будет закрыт.

  Тип: Long

  Значение по умолчанию: 60000(60 секунд)

  Допустимые значения: [0,...,9223372036854775807]

- kafka.topic.prefix - Логическое имя, которое идентифицирует и обеспечивает пространство имен для конкретного сервера РЕД Базы Данных, на котором фиксируются изменения. Логическое имя должно быть уникальным для всех других коннекторов, поскольку оно ис-

пользуется в качестве префикса для всех имен тем Kafka, которые получают события, испускаемые этим коннектором. В префиксе сервера должны использоваться только буквы, цифры, дефисы, точки и символы подчеркивания.

Тип: string

Значение по умолчанию: ""

- key.converter - Конвертер ключей.

  Тип: class

  Значение по умолчанию: null

  Допустимые значения: подкласс org.apache.kafka.connect.storage.Converter, класс с публичным конструктором без аргумента

- value.converter - Конвертер значений.

  Тип: class

  Значение по умолчанию: null

  Допустимые значения: подкласс org.apache.kafka.connect.storage.Converter, класс с публичным конструктором без аргумента

Пример конфигурации:

```
connector.name: reddatabase
connector.journal.directory: /tmp/journal
connector.journal.check.interval.ms: 0

kafka.producer.bootstrap.servers: localhost:9092
kafka.producer.retries: 3
kafka.producer.retry.backoff.ms: 1000
kafka.producer.max.block.ms: 1000
kafka.producer.timeout.ms: 1000
kafka.producer.transactional.id: reddatabase-transaction-id
kafka.topic.prefix: reddatabase_topic

key.converter: org.apache.kafka.connect.json.JsonConverter
value.converter: org.apache.kafka.connect.json.JsonConverter
```

Логирование действий коннектора осуществляется с помощью библиотеки Logback. Пример конфигурации логирования в файл:

```
<configuration>
        <appender name="FILE"class="ch.qos.logback.core.FileAppender">
                <file>/tmp/debezium-connector-reddatabase.log</file>

                <encoder>
                        <pattern>%date %level [%thread] %logger50 [%file:%line] %msg%n
</pattern>
                </encoder>
        </appender>
        <root level="debug">
                <appender-ref ref="FILE"/>
        </root>
</configuration>
```

Для логирования работы коннектора в jvm.args необходимо добавить строку с паратером -Dlogback.configurationFile, в котором указывается путь к файлу с конфигурацией логирования:

```
-Dlogback.configurationFile=<путь>/logback.xml
```

15.3 Наименование тем

По умолчанию коннектор записывает события изменений для всех операций `INSERT`, `UPDATE` и `DELETE`, происходящих в таблице, в одну тему `Apache Kafka`, определённую для этой таблицы. Коннектор использует следующий формат для наименования тем событий изменений, который задаётся в файле `reddatabase.connector.yaml`:

```
<kafka.topic.prefix>.<путь к бд>.<имя таблицы>
```

Символы "\", "/" и ":", указанные в пути к базе данных, будут заменены на символ "_".

15.4 Фильтрация таблиц и столбцов

Коннектор отслеживает изменения для всех таблиц, для которых включена репликация. Список реплицируемых таблиц является вариантом реализации списка отслеживаемых таблиц (так называемый `whitelist/includelist`). Включение таблицы в список реплицируемых осуществляется командой `ALTER DATABASE INCLUDE TABLE <ТАБЛИЦА> TO PUBLICATION`:

```
ALTER DATABASE INCLUDE TABLE TEST_TABLE TO PUBLICATION
```

Возможно включение нескольких таблиц в список реплицируемых:

```
ALTER DATABASE INCLUDE TABLE T1, T2, T3 TO PUBLICATION
```

При желании можно добавить все доступные таблицы (в том числе и те, которые будут созданы в процессе) в список реплицируемых:

```
ALTER DATABASE INCLUDE ALL TO PUBLICATION
```

Исключить конкретные таблицы из списка реплицируемых:

```
ALTER DATABASE EXCLUDE TABLE T1, T2, T3 FROM PUBLICATION;
```

Исключить все таблицы из списка реплицируемых:

```
ALTER DATABASE EXCLUDE ALL FROM PUBLICATION;
```

Кроме инструментов РЕД Базы Данных коннектор имеет встроенные механизмы фильтрации таблиц и столбцов. Настройка фильтрации таблиц и столбцов осуществляется в файле `reddatabase.connector.yaml` следующими параметрами:

- `table.include.list` - Необязательный список регулярных выражений, разделенных запятыми, которые соответствуют идентификаторам таблиц, изменения которых необходимо отслеживать. Коннектор не фиксирует изменения в таблицах, невключенных в `table.include.list`. Каждый идентификатор имеет вид: `<Имя_БД>.<Имя_Таблицы>`. По умолчанию коннектор фиксирует изменения в каждой несистемной таблице, добавленной в список реплицируемых. Несовместим с параметром `table.exclude.list`;
- `table.exclude.list` - Необязательный список регулярных выражений, разделенных запятыми, которые соответствуют идентификаторам таблиц, изменения которых не отслеживаются. Коннектор фиксирует изменения в любой таблице, добавленной в список реплика-

ции, но не указанной в table.exclude.list. Каждый идентификатор имеет вид <Имя_БД>.<Имя_Таблицы>. Несовместим с параметром table.include.list;

- column.include.list - Необязательный список регулярных выражений, разделенных запятыми, которые соответствуют именам столбцов для включения в событие изменения. Полные имена столбцов имеют вид: <Имя_БД>.<Имя_Таблицы>.<Имя_Столбца>. Несовместим с параметром column.exclude.list;
- column.exclude.list - Необязательный список регулярных выражений, разделенных запятыми, которые соответствуют именам столбцов для исключения из события изменения. Полные имена столбцов имеют вид: <Имя_БД>.<Имя_Таблицы>.<Имя_Столбца>. Несовместим с параметром column.include.list.

Если в настройках указаны оба параметра table.include.list и table.exclude.list, то приоритет будет за table.exclude.list. Такое же поведение касается параметров column.include.list и column.exclude.list - приоритет у column.exclude.list. Включение таблиц в список реплицируемых (ALTER DATABASE INCLUDE TABLE <ТАБЛИЦА> TO PUBLICATION) является обязательным условием отслеживания изменений таблиц. Без включения репликации для таблиц использование парамтеров table.include.list, table.exclude.list, column.include.list, column.exclude.list не имеет смысла.

Примеры:

```
# Включение таблицы TABLE1 из БД /opt/databases/employee.fdb в список для
отслеживания:
table.include.list: /opt/databases/employee.fdb.table1

# Включение таблицы TABLE1 из ЛЮБОЙ БД в список для отслеживания:
table.include.list: .*table1

# Включение таблицы TABLE1 из ЛЮБОЙ БД и таблицы TEST_TABLE из любой БД с именем
data.fdb в список для отслеживания:
table.include.list: .*table1, .*data.fdb.test_table

# Включение таблицы TABLE1 из ЛЮБОЙ employee.fdb в список для отслеживания:
table.include.list: .*employee.fdb.table1

# Исключение таблицы TABLE1 базы данных /opt/databases/employee.fdb из списка
отслеживаемых:
table.include.list: /opt/databases/employee.fdb.table1

# Включение столбца ID таблицы TABLE1 из БД /opt/databases/employee.fdb в сообщение
изменения:
column.include.list: /opt/databases/employee.fdb.table1.id

# Включение столбца ID таблицы TABLE1 из ЛЮБОЙ БД в сообщение изменения:
column.include.list: .*table1.id

# Включение столбцов ID и NAME ЛЮБОЙ таблицы ЛЮБОЙ БД в сообщение изменения:
column.include.list: .*id, *.name

# Исключение столбца F_VCHAR ЛЮБОЙ таблицы ЛЮБОЙ БД из сообщения изменения:
column.exclude.list: .*f_vchar
```

15.5 События изменения данных

Каждое событие изменения данных, которое формирует коннектор РЕД Базы Данных, имеет ключ и значение. Структура ключа и значения зависит от таблицы, из которой исходят события изменения. Коннектор, как и `Kafka Connect`, разработан с учетом непрерывных потоков сообщений о событиях. Каждый ключ и значение сообщения состоит из двух частей: схемы и полезной нагрузки. Схема описывает структуру полезной нагрузки, а полезная нагрузка содержит фактические данные. Для каждой измененной таблицы ключ события изменения формируется из полей в первичном ключе (или уникальном ключевом ограничении) таблицы на момент создания события.

Как и ключ сообщения, значение сообщения о событии изменения имеет раздел схемы и раздел полезной нагрузки. Раздел полезной нагрузки каждого значения события изменения, создаваемого коннектором, имеет структуру конверта со следующими полями:

- `op` - обязательное поле, содержащее строковое значение, описывающее тип операции: `CREATE` - создание (или вставка), `UPDATE` - обновление, `DELETE` - удаление;
- `before` - необязательное поле, которое, если присутствует, содержит состояние строки до наступления события;
- `after` - необязательное поле, которое, если присутствует, содержит состояние строки после произошедшего события. Структура описывается той же схемой, которая использовалась ранее;
- `source` - обязательное поле, содержащее структуру, описывающую исходные метаданные для события, которая в случае РЕД Базы Данных содержит такие поля: версия коннектора, имя коннектора, имя сервера, номер транзакции, время (по системным часам в `JVM`, выполняющей задачу `Kafka Connect`), в которое коннектор обработал событие, имя базы данных, имя таблицы.

Пример формирования события изменения данных для операции `INSERT`:

```
INSERT INTO TABLE11 VALUES(42,'NEW VALUE');
```

Когда в кластер `Kafka` попадает сообщение об удалении, то соответствующий ключ удаляется из хранилища состояний, освобождая таким образом место.

Если используется `Jdbc Sink Connector`, то включение `tombstone records` необходимо для генерации `DELETE` запросов для результирующей БД.

Отключить генерацию `tombstone` можно настройкой `tombstones.on.delete`:

```
tombstones.on.delete: false
```

15.6 События изменения схем

Коннектор формирует события изменения схемы, которые формируются из выполняемых `DDL` операций на сервере. Коннектор отправляет сообщение каждый раз, когда происходит изменение структуры базы данных. Сообщения, которые коннектор отправляет в тему изменения схемы, содержат полезную нагрузку и также содержат схему о событии изменения. Полезная нагрузка сообщения о событии изменения схемы включает следующие элементы:

- `databaseName\schemaName` - Идентифицирует базу данных и схему, содержащую изменение.
- `ddl` - Это поле содержит `DDL`, который отвечает за изменение схемы.
- `tableChanges` - Массив из одного или нескольких элементов, которые содержат изменения схемы, созданные командой `DDL`.

- `type` - Описывает вид изменения. Значением может быть одно из следующих: `CREATE` - таблица создана; `ALTER` - таблица изменена; `DROP` - таблица удалена.
- `id` - Полный идентификатор таблицы, которая была создана, изменена или удалена.
- `table` - Представляет метаданные таблицы после примененного изменения.
- `primaryKeyColumnNames` - Список столбцов, составляющих первичный ключ таблицы.
- `columns` - Метаданные для каждого столбца в измененной таблице.

Пример формирования события изменения схемы при выполнение операции `CREATE TABLE`:

```
create table table24(f_id integer primary key, f_vchar varchar(100));
```

15.7 Журналы транзакций

Коннектор для каждой репликационной транзакции создает журнал транзакции. Журнал транзакции используется для сохранения данных транзакции (`DDL` и `DML` событий) и отправки в кластер `Kafka`. Также журналы используются для повторной отправки, если кластер был недоступен или возникли другие ошибки при первичной отправке сообщений транзакции. Для включения записи журналов транзакций необходимо указать путь к каталогу журналов в опции `connector.journal.directory`, например:

```
connector.journal.directory: /tmp/debezium-connector-reddatabase
```

В случае возникновени ошибок при отправке сообщений журналы транзакции повторно вычитываются, и при первой доступности кластера `Kafka` сообщения отправляются снова. При успешном отправлении всех сообщений репликационной транзакции журнал транзакции удаляется. Период просмотра наличия журналов транзакций и их повторной отправки по умолчанию равен 0. Период можно изменить через опцию `connector.journal.check.interval.ms`, которая принимает положительное число в миллисекундах. Например, задать время просмотра наличия журналов транзакций и их отправки равное 10 минутам:

```
connector.journal.check.interval.ms: 600000
```

Имена журналов транзакций имеют следующий формат: `<Случайный UUID>.log[.live]`. Журнал незафиксированной транзакции имеет расширение `.live`, журналы зафиксированных транзакций представлены с расширением `.log`. Коннектор отправляет сообщения только из журналов зафиксированных транзакций (с расширением `.log`). Если серверная транзакция была отменена (`rollback`), журнал репликационной транзакции удаляется в любом случае.

Приложение А Описание таблиц мониторинга

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 Базы Данных

Таблица	Описание
MON$ATTACHMENTS	Сведения о текущих соединениях с базой данных.
MON$CALL_STACK	Обращения к стеку активными запросами хранимых процедур и триггеров.
MON$CONTEXT_VARIABLES	Сведения о пользовательских контекстных переменных.
MON$DATABASE	Сведения о базе данных, с которой выполнено соединение.
MON$IO_STATS	Статистика по вводу-выводу.
MON$MEMORY_USAGE	Статистика использования памяти.
MON$RECORD_STATS	Статистика на уровне записей.
MON$STATEMENTS	Подготовленные к выполнению запросы.
MON$STATEMENT_PARAMETERS	Параметры выполняемых запросов, вложенных процедур и функций
MON$TABLE_STATS	Статистика на уровне таблиц.
MON$TRANSACTIONS	Запущенные на выполнение транзакции.
MON$TEMP_SPACES	Сведения о временных объектах
MON$TEMP_FILES	Сведения о временных файлах
MON$COMPILED_STATEMENTS	Хранит информацию о скомпилированных запросах.
MON$REPLICATION	Сведения о статусе репликации базы данных

A.1 MON$ATTACHMENTS

Таблица А.2 — Сведения о текущих соединениях с базой данных

Идентификатор столбца	Тип данных	Описание
MON$ATTACHMENT_ID	BIGINT	Идентификатор соединения.
MON$SERVER_PID	INTEGER	Идентификатор серверного процесса.
MON$STATE	SMALLINT	Состояние соединения: 0 — бездействующее, 1 — активное. Соединение считается активным, если в нем есть хотя бы одна транзакция с хотя бы одним открытым запросом.
MON$ATTACHMENT_NAME	VARCHAR(255)	Полный путь к файлу и имя первичного файла базы данных.
MON$USER	CHAR(63)	Имя пользователя, соединенного с базой данных.

(разрыв таблицы)

(разрыв таблицы)

Идентификатор столбца	Тип данных	Описание
MON$ROLE	CHAR(63)	Имя роли, указанное при соединении. Если роль во время соединения не была задана, поле содержит текст NONE.
MON$REMOTE_PROTOCOL	VARCHAR(10)	Имя удаленного протокола.
MON$REMOTE_ADDRESS	VARCHAR(255)	Удаленный адрес (адрес и имя сервера).
MON$REMOTE_PID	INTEGER	Идентификатор удаленного клиентского процесса.
MON$CHARACTER_SET_ID	SMALLINT	Идентификатор набора символов в соединении.
MON$TIMESTAMP	TIMESTAMP WITH TIME ZONE	Дата и время начала соединения.
MON$GARBAGE_COLLECTION	SMALLINT	Разрешена ли сборка мусора для этого соединения: 1 - разрешена, 0 - нет.
MON$REMOTE_PROCESS	VARCHAR(255)	Полное имя файла и путь к исполняемому файлу, который установил данное соединение.
MON$STAT_ID	INTEGER	Идентификатор статистики.
MON$CLIENT_VERSION	VARCHAR(255)	Версия клиентской библиотеки.
MON$REMOTE_VERSION	VARCHAR(255)	Версия удалённого протокола.
MON$REMOTE_HOST	VARCHAR(255)	Имя удалённого хоста.
MON$REMOTE_OS_USER	VARCHAR(255)	Имя пользователя в операционной системе.
MON$AUTH_METHOD	VARCHAR(255)	Метод аутентификации, используемый при подключении.
MON$SYSTEM_FLAG	SMALLINT	Флаг того, что подключение системное: 0 — пользовательское подключение; 1 — системное подключение.
MON$IDLE_TIMEOUT	INTEGER	Тайм-аут простоя соединения уровня соединения. Содержит значение тайм-аута простоя уровня соединения, в секундах. Если тайм-аут не установлен — 0.
MON$IDLE_TIMER	TIMESTAMP WITH TIME ZONE	Время истечения таймера ожидания. Содержит NULL, если тайм-аут простоя соединения не установлен, или если таймер не запущен.
MON$STATEMENT_TIMEOUT	INTEGER	Тайм-аут SQL запроса уровня соединения. Содержит значение тайм-аута, установленное на уровне соединения, в миллисекундах. Если тайм-аут не установлен — 0.
MON$WIRE_COMPRESSED	BOOLEAN	Используется ли сжатие сетевого трафика. Если используется сжатие сетевого трафика значение равно TRUE, если не используется — FALSE. Для встроенных соединений возвращает NULL.
MON$WIRE_ENCRYPTED	BOOLEAN	Используется ли шифрование сетевого трафика. Если используется шифрование сетевого трафика значение равно TRUE, если не используется — FALSE. Для встроенных соединений — возвращает NULL.
MON$LAST_ACTIVITY_TIME	TIMESTAMP WITH TIME ZONE	Время последнего обращения пользователя к серверу.
MON$SESSION_TIMEZONE	CHAR(63)	Часовой пояс сессии

A.2 MON$REPLICATION

Сведения собираются только для мастер базы.

Таблица A.3 — Сведения о статусе репликации базы данных

Идентификатор столбца	Тип данных	Описание
MON$TYPE	SMALLINT	Тип состояния базы данных: • 1 – мастер с синхронной репликацией; • 2 – мастер с асинхронной репликацией;
MON$CONNECTION_STRING	VARCHAR(255)	Строка подключения: • для мастера в синхронном режиме – путь к базе в формате [<логин>:<пароль>@]<путь к базе>; • для мастера в асинхронном режиме – путь к каталогу с журналами репликации;
MON$ACTIVE	BIGINT	Статус активности подключения: • -2 - реплика отключена после ошибки (для мастера с адаптивной репликацией); • -1 - данные на слейве не актуальны (для мастера с адаптивной репликацией); • 0 – отключена (для мастера с синхронной репликацией); • 1 – подключена (для мастера с синхронной репликацией); • <номер_файла> – номер текущего файла журнала (для мастера с асинхронной репликацией).
MON$LAST_MODIFIED	TIMESTAMP	Содержит время последней успешной отправки буфера на слейв.
MON$WAITFLUSH_COUNT	INTEGER	Количество отправленных пакетов (с задержкой мастера, при коммитах).
MON$WAITFLUSH_TIME	BIGINT	Количество времени мастера для отправки пакетов (в микросекундах).
MON$WAITFLUSH_TRANSFER	BIGINT	Размер отправленных данных слейву (в байтах).
MON$BACKGROUND_COUNT	INTEGER	Количество отправленных пакетов (в фоне, в рамках одной транзакции).
MON$BACKGROUND_TIME	BIGINT	Количество времени мастера для отправки пакетов (в фоне, в рамках одной транзакции).
MON$BACKGROUND_TRANSFER	BIGINT	Размер отправленных данных слейву (в фоне, в рамках одной транзакции).
MON$SERVER_PID	INTEGER	Идентификатор серверного процесса.

A.3 MON$CALL_STACK

Таблица A.4 — Обращения к стеку запросами хранимых процедур и триггеров

Идентификатор столбца	Тип данных	Описание
MON$CALL_ID	BIGINT	Идентификатор обращения.
MON$STATEMENT_ID	BIGINT	Идентификатор верхнего уровня SQL—запроса, инициировавшего цепочку обращений.
MON$CALLER_ID	BIGINT	Идентификатор обращающегося триггера, хранимой функции или хранимой процедуры.
MON$COMPILED_STATEMENT_ID	BIGINT	Идентификатор запроса.
MON$OBJECT_NAME	CHAR(63)	Имя объекта PSQL.
MON$OBJECT_TYPE	SMALLINT	Тип объекта PSQL: • 2 — триггер; • 5 — хранимая процедура; • 15 — хранимая функция.
MON$TIMESTAMP	TIMESTAMP WITH TIME ZONE	Дата и время старта обращения.
MON$SOURCE_LINE	INTEGER	Номер исходной строки запроса SQL, выполняющегося в настоящий момент.
MON$SOURCE_COLUMN	INTEGER	Номер исходного столбца запроса SQL, выполняющегося в настоящий момент.
MON$STAT_ID	INTEGER	Идентификатор статистики.
MON$PACKAGE_NAME	CHAR(63)	Имя пакета для упакованных процедур/функций.

A.4 MON$CONTEXT_VARIABLES

Таблица A.5 — Сведения о пользовательских контекстных переменных

Идентификатор столбца	Тип данных	Описание
MON$ATTACHMENT_ID	BIGINT	Идентификатор соединения. Содержит корректное значение только для контекстных переменных уровня соединения, для переменных уровня транзакции устанавливается в NULL.
MON$TRANSACTION_ID	BIGINT	Идентификатор транзакции. Содержит корректное значение только для контекстных переменных уровня транзакции, для переменных уровня соединения устанавливается в NULL.
MON$VARIABLE_NAME	VARCHAR(80)	Имя контекстной переменной.
MON$VARIABLE_VALUE	VARCHAR(32765)	Значение контекстной переменной.

A.5 MON$DATABASE

Таблица A.6 — Сведения о базе данных, с которой выполнено соединение

Идентификатор столбца	Тип данных	Описание
MON$DATABASE_NAME	VARCHAR(255)	Полный путь и имя первичного файла базы данных или псевдоним базы данных.
MON$PAGE_SIZE	SMALLINT	Размер страницы файлов базы данных в байтах.
MON$ODS_MAJOR	SMALLINT	Старшая версия ODS.
MON$ODS_MINOR	SMALLINT	Младшая версия ODS.
MON$OLDEST_TRANSACTION	BIGINT	Номер старейшей заинтересованной транзакции — OIT, Oldest Interesting Transaction.
MON$OLDEST_ACTIVE	BIGINT	Номер старейшей активной транзакции — OAT, Oldest Active Transaction.
MON$OLDEST_SNAPSHOT	BIGINT	Номер транзакции, которая была активной на момент старта транзакции OAT, — транзакция OST, Oldest Snapshot Transaction.
MON$NEXT_TRANSACTION	BIGINT	Номер следующей транзакции.
MON$PAGE_BUFFERS	INTEGER	Количество страниц, выделенных в оперативной памяти для кэша.
MON$SQL_DIALECT	SMALLINT	SQL диалект базы данных: 1 или 3.
MON$SHUTDOWN_MODE	SMALLINT	Текущее состояние останова (**shutdown**) базы данных: • 0 — база данных активна (**online**), • 1 — останов для нескольких пользователей (**multi-user shutdown**), • 2 — останов для одного пользователя (**single-user shutdown**), • 3 — полный останов (**full shutdown**).
MON$SWEEP_INTERVAL	INTEGER	Интервал чистки (**sweep interval**).
MON$READ_ONLY	SMALLINT	Признак, является ли база данных только для чтения, **read only** (значение 1) или для чтения и записи, **read-write** (значение 0).
MON$FORCED_WRITES	SMALLINT	Указывает, установлен ли для базы режим синхронного вывода (**forced writes**, значение 1) или режим асинхронного вывода (значение 0).
MON$RESERVE_SPACE	SMALLINT	Флаг, указывающий на полное заполнение страниц БД (**full**) или 80% заполнение по умолчанию (**reserve**). 100%-е заполнение имеет смысл для баз только для чтения.
MON$CREATION_DATE	TIMESTAMP WITH TIME ZONE	Дата и время создания базы данных.
MON$PAGES	BIGINT	Количество страниц, выделенных для базы данных на внешнем устройстве.

(разрыв таблицы)

(разрыв таблицы)

Идентификатор столбца	Тип данных	Описание
MON$STAT_ID	INTEGER	Идентификатор статистики.
MON$BACKUP_STATE	SMALLINT	Текущее физическое состояние backup: • 0 — база не затронута бэкапом, • 1 — база заблокирована для резервирования, • 2 — объединение временного файла дельты и основного файла базы данных.
MON$CRYPT_PAGE	BIGINT	Страница, которая сейчас находится на шифровании/-дешифровании. Ноль если этот процесс закончился или не начинался.
MON$OWNER	CHAR(63)	Владелец базы данных.
MON$SEC_DATABASE	CHAR(7)	Отображает, какой тип базы данных безопасности используется: • Default - база данных безопасности по умолчанию, т.е. security5.fdb; • Self — в качестве базы данных безопасности используется текущая база данных; • Other — в качестве базы данных безопасности используется другая база данных.
MON$CRYPT_STATE	SMALLINT	Состояние шифрование БД: • 0 - база данных не зашифрована, • 1 - зашифрована, • 2 - в процессе шифрования.

A.6 MON$IO_STATS

Таблица A.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 — запрос (statement), • 4 — вызов (call), • 5 — кэшированный компилированный запрос (compiled statement).
MON$PAGE_READS	BIGINT	Количество прочитанных (read) страниц базы данных.
MON$PAGE_WRITES	BIGINT	Количество записанных (write) страниц базы данных.

(разрыв таблицы)

(разрыв таблицы)

Идентификатор столбца	Тип данных	Описание
MON$PAGE_FETCHES	BIGINT	Количество страниц, считанных из страничного кэша (fetch).
MON$PAGE_MARKS	BIGINT	Количество отмеченных (mark) страниц базы данных. Это "грязные" страницы, т.е. изменённые в памяти (в кэше), но пока не записанные на диск.

A.7 MON$MEMORY_USAGE

Таблица A.8 — Статистика использования памяти

Идентификатор столбца	Тип данных	Описание
MON$STAT_ID	INTEGER	Идентификатор статистики.
MON$STAT_GROUP	SMALLINT	Группа статистики: • 0 — база данных (database), • 1 — соединение с базой данных (connection), • 2 — транзакция (transaction), • 3 — запрос (statement), • 4 — вызов (call), • 5 — кэшированный компилированный запрос (compiled statement).
MON$MEMORY_USED	BIGINT	Количество используемой памяти (в байтах). Информация о высокоуровневом распределении памяти, выполненной сервером из пулов. Может быть полезна для отслеживания утечек памяти и чрезмерного потребления памяти в соединениях, процедурах и т.д.
MON$MEMORY_ALLOCATED	BIGINT	Количество памяти, выделенной ОС (в байтах). Информация о низкоуровневом распределении памяти, выполненном менеджером памяти — объем памяти, выделенный операционной системой, что позволяет контролировать физическое потребление памяти. Обратите внимание, не все записи этого столбца имеют ненулевые значения. Малые выделения памяти здесь не фиксируются, а вместо этого добавляются к пулу памяти базы данных. Только MON$DATABASE (MON$STAT_GROUP = 0) и связанные с выделением памяти объекты имеют ненулевое значение.
MON$MAX_MEMORY_USED	BIGINT	Максимальное количество байт, используемое данным объектом.
MON$MAX_MEMORY_ALLOCATED	BIGINT	Максимальное количество байт, выделенное ОС данному объекту.

A.8 MON$RECORD_STATS

Таблица A.9 — Статистика на уровне записей

Идентификатор столбца	Тип данных	Описание
MON$STAT_ID	INTEGER	Идентификатор статистики.
MON$STAT_GROUP	SMALLINT	Группа статистики: • 0 — база данных (database), • 1 — соединение с базой данных (connection), • 2 — транзакция (transaction), • 3 — запрос (statement), • 4 — вызов (call), • 5 — кэшированный компилированный запрос (compiled statement).
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). Это происходит при откате транзакции. Если обнаружена версия записи, созданная в результате отката транзакции, она помечается для сборки мусора, а предыдущая подтвержденная версия переносится в основной слот на странице данных.
MON$RECORD_PURGES	BIGINT	Количество удаленных ненужных записей (purged records). Это происходит, когда сборщик мусора удаляет старые версии записи, которые больше не нужны какой-либо активной транзакции.
MON$RECORD_EXPUNGES	BIGINT	Количество вычищенных средствами сборки мусора записей (expunged records). Это происходит, когда запись, которая не видна какой-либо активной транзакции, удаляется и удаляющая транзакция подтверждается (commit). Удаленная запись и все ранее подтвержденные версии этой записи удаляются, чтобы занятое ими дисковое пространство можно было повторно использовать. Если запись все еще видна для более старых snapshot транзакций, конечно, удаление может произойти только после завершения этих транзакций.
MON$RECORD_LOCKS	BIGINT	Количество записей прочитанных с использованием предложения WITH LOCK.
MON$RECORD_WAITS	BIGINT	Количество попыток обновления/модификации/блокировки записей принадлежащих нескольким активным транзакциям. Транзакция находится в режиме WAIT.

(разрыв таблицы)

(разрыв таблицы)

Идентификатор столбца	Тип данных	Описание
MON$RECORD_CONFLICTS	BIGINT	Количество неудачных попыток обновления/модификации/блокировки записей принадлежащих нескольким активным транзакциям. В таких ситуациях сообщается о конфликте обновления (UPDATE CONFLICT).
MON$BACKVERSION_READS	BIGINT	Количество прочитанных версий при поиске видимых версий записей.
MON$FRAGMENT_READS	BIGINT	Количество прочитанных фрагментов записей.
MON$RECORD_RPT_READS	BIGINT	Количество повторно прочитанных записей.
MON$RECORD_IMGC	BIGINT	Количество записей, затронутых промежуточной сборкой мусора. Например, есть цепочка версий записи N1 - N2 - N3 - N4 - N5. Какая-то старая транзакция видит версию записи N1, все остальные - видят N5. Версии N2 - N3 - мусорные. До версии 5.0 сервер не мог их удалить, до тех пор, пока станет ненужной N1. Теперь промежуточная сборка позволяет удалять такие мусорные версии из цепочек.

A.9 MON$STATEMENTS

Таблица A.10 — Подготовленные к выполнению запросы

Идентификатор столбца	Тип данных	Описание
MON$STATEMENT_ID	BIGINT	Идентификатор запроса.
MON$ATTACHMENT_ID	BIGINT	Идентификатор соединения.
MON$COMPILED_STATEMENT_ID	BIGINT	Идентификатор запроса.
MON$TRANSACTION_ID	BIGINT	Идентификатор транзакции.
MON$STATE	SMALLINT	Состояние запроса: • 0 — бездействующий (idle), • 1 — активный (active), • 2 — приостановленный (stalled). То есть запрос или курсор "живой", но в данный момент не выполняется. Например, это состояние является перерывом между клиентскими фетчами, т.е. запрос создал курсор, он еще недофетчен и в данный момент фетч не делается (обрабатывает предыдущую порцию).
MON$TIMESTAMP	TIMESTAMP WITH TIME ZONE	Дата и время старта запроса.
MON$SQL_TEXT	BLOB TEXT	Текст запроса на языке SQL.
MON$STAT_ID	INTEGER	Идентификатор статистики.
MON$EXPLAINED_PLAN	BLOB TEXT	План запроса в расширенной форме.

(разрыв таблицы)

(разрыв таблицы)

Идентификатор столбца	Тип данных	Описание
MON$STATEMENT_TIMEOUT	INTEGER	Тайм-аут SQL запроса уровня. Содержит значение тайм-аута, установленное на уровне соединения/запроса, в миллисекундах. Если тайм-аут не установлен — 0.
MON$STATEMENT_TIMER	TIMESTAMP WITH TIME ZONE	Время истечения таймера SQL запроса. Содержит NULL, если тайм-аут SQL запроса не установлен, или если таймер не запущен.
MON$SORTING_FILES_SIZE	BIGINT	Размер временных файлов сортировок.

A.10 MON$STATEMENT_PARAMETERS

Таблица A.11 — Параметры выполняемых запросов, вложенных процедур и функций

Идентификатор столбца	Тип данных	Описание
MON$STATEMENT_ID	BIGINT	Идентификатор запроса.
MON$CALL_ID	BIGINT	Идентификатор обращения.
MON$PARAMETER_NUMBER	SMALLINT	Номер параметра.
MON$PARAMETER_TYPE	CHAR(30)	Тип параметра.
MON$PARAMETER_VALUE	CHAR(1024)	Значение параметра.

A.11 MON$TABLE_STATS

Таблица A.12 — Статистика на уровне таблицы

Идентификатор столбца	Тип данных	Описание
MON$STAT_ID	INTEGER	Идентификатор статистики.
MON$STAT_GROUP	SMALLINT	Группа статистики: • 0 — база данных (database), • 1 — соединение с базой данных (connection), • 2 — транзакция (transaction), • 3 — запрос (statement), • 4 — вызов (call), • 5 — кэшированный компилированный запрос (compiled statement).
MON$TABLE_NAME	CHAR(63)	Имя таблицы.
MON$RECORD_STAT_ID	INTEGER	Ссылка на MON$RECORD_STATS.

A.12 MON$TEMP_SPACES

Таблица A.13 — Сведения о временных объектах

Идентификатор столбца	Тип данных	Описание
MON$TEMP_SPACE_ID	BIGINT	Идентификатор временного объекта.
MON$TEMP_TYPE	SMALLINT	Тип временного объекта: • 0 - буфер записей(record buffer) • 1 - слияние данных (merge) • 2 - undo-лог (undo) • 3 - временный BLOB (blob) • 4 - сортировка (sort) • 5 - пакет (batch) • 6 - данные мониторинга (monitor) • 7 - курсор (cursor)
MON$TRANSACTION_ID	BIGINT	Идентификатор (номер) транзакции.
MON$CALL_ID	BIGINT	Идентификатор обращения.
MON$STATEMENT_ID	BIGINT	Идентификатор запроса.
MON$TOTAL_SIZE	BIGINT	Общий размер памяти, выделенный для временного объекта.
MON$CACHED_SIZE	BIGINT	Объём оперативной памяти, выделенный для временного объекта.
MON$USED_SIZE	BIGINT	Объём оперативной памяти, используемый временным объектом.

A.13 MON$TEMP_FILES

Таблица A.14 — Сведения о временных объектах

Идентификатор столбца	Тип данных	Описание
MON$TEMP_SPACE_ID	BIGINT	Идентификатор временного объекта.
MON$FILE_NAME	VARCHAR(255)	Имя временного файла в формате: <тип временного объекта>_att<id подключения>_stmt<id запроса>_<случайная строка> Например: /tmp/rdb_table_att342316_stmt106_M2OSLu
MON$FILE_SIZE	BIGINT	Размер временного файла.

A.14 MON$TRANSACTIONS

Таблица A.15 — Описывает запущенные на выполнение транзакции

Идентификатор столбца	Тип данных	Описание
MON$TRANSACTION_ID	BIGINT	Идентификатор (номер) транзакции.
MON$ATTACHMENT_ID	BIGINT	Идентификатор соединения.
MON$STATE	SMALLINT	Состояние транзакции: • 0 — бездействующая (нет открытых запросов), • 1 — активная (есть открытые запросы).
MON$TIMESTAMP	TIMESTAMP WITH TIME ZONE	Дата и время старта транзакции.
MON$TOP_TRANSACTION	INTEGER	Верхний предел используемый транзакцией чистильщика (sweeper) при продвижении глобального OIT. Все транзакции выше этого порога считаются активными. Обычно он эквивалентен MON$TRANSACTION_ID, но использование COMMIT RETAINING или ROLLBACK RETAINING приводит к тому, что MON$TOP_TRANSACTION останется неизменным ("зависшим") при увеличении идентификатора транзакции.
MON$OLDEST_TRANSACTION	INTEGER	Номер старейшей заинтересованной транзакции — OIT, Oldest Interesting Transaction.
MON$OLDEST_ACTIVE	INTEGER	Номер старейшей активной транзакции — OAT, Oldest Active Transaction.
MON$ISOLATION_MODE	SMALLINT	Режим (уровень) изоляции: • 0 — consistency (snapshot table stability), • 1 — concurrency (snapshot), • 2 — read committed record version, • 3 — read committed no record version, • 4 — read committed read consistency.
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 (значение 0).
MON$AUTO_COMMIT	SMALLINT	Признак, используется ли автоматическое подтверждение транзакции auto-commit (значение 1) или нет (значение 0).
MON$AUTO_UNDO	SMALLINT	Признак, используется ли автоматическая отмена транзакции auto-undo (значение 1) или нет (значение 0).
MON$STAT_ID	INTEGER	Идентификатор статистики.

A.15 MON$COMPILED_STATEMENTS

Таблица A.16 — Хранит информацию о скомпилированных запросах

Идентификатор столбца	Тип данных	Описание
MON$COMPILED_STATEMENT_ID	BIGINT	Идентификатор запроса.
MON$SQL_TEXT	BLOB	Текст запроса, если он доступен.
MON$EXPLAINED_PLAN	BLOB	Расширенный план запроса.
MON$OBJECT_NAME	CHAR(63)	Имя объекта PSQL.
MON$OBJECT_TYPE	SMALLINT	Тип объекта PSQL.
MON$PACKAGE_NAME	CHAR(63)	Имя пакета объекта PSQL.
MON$STAT_ID	INTEGER	Идентификатор статистики.

Приложение Б Псевдотаблицы безопасности

Псевдотаблицы безопасности имеют префикс имени SEC$. Они отображают данные из текущей базы данных безопасности. Эти таблицы являются виртуальными в том смысле, что до обращения к ним со стороны пользователя, никаких данных в них не записано. Они заполняются данными только в момент запроса пользователя. При этом описания этих таблиц в базе данных присутствуют постоянно. В этом смысле эти псевдотаблицы подобны таблицам семейства MON$, используемых для мониторинга сервера.

Полный доступ ко всей информации, предоставляемой таблицами безопасности, имеют SYSDBA и владелец базы данных. Обычные пользователи ограничены информацией о самих себе, другие пользователи невидимы для них.

> Эти функции во многом зависят от плагина управления пользователями. Имейте в виду, что некоторые опции игнорируются при использовании устаревшего плагина управления пользователями.

Список псевдотаблиц представлен в таблице Б.1.

Таблица Б.1 — Список псевдотаблиц РЕД Базы Данных

Таблица	Содержание
SEC$DB_CREATORS	Список пользователей, имеющих привилегию CREATE DATABASE.
SEC$GLOBAL_AUTH_MAPPING	Сведения о глобальных отображениях.
SEC$USERS	Список пользователей в текущей базе данных безопасности.
SEC$USER_ATTRIBUTES	Сведения о дополнительных атрибутах пользователей.
SEC$POLICIES	Сведения о созданных политиках безопасности.

Б.1 SEC$DB_CREATORS

Таблица хранит сведения о пользователях, имеющих привилегию CREATE DATABASE.

Идентификатор столбца	Тип данных	Описание
SEC$USER	CHAR(63)	Имя пользователя или роли, которому даны полномочия на создание базы данных.
SEC$USER_TYPE	SMALLINT	Тип пользователя: • 8 — пользователь; • 13 — роль.

Б.2 SEC$GLOBAL_AUTH_MAPPING

Таблица хранит сведения о глобальных отображениях.

Идентификатор столбца	Тип данных	Описание
SEC$MAP_NAME	CHAR(63)	Имя глобального отображения.

(разрыв таблицы)

(разрыв таблицы)

Идентификатор столбца	Тип данных	Описание
SEC$MAP_USING	CHAR(1)	Является ли аутентификация общесерверной (S) или обычной (P).
SEC$MAP_PLUGIN	CHAR(63)	Имя плагина аутентификации, из которого происходит отображение.
SEC$MAP_DB	CHAR(63)	Имя базы данных, в которой прошла аутентификация. Из неё происходит отображение.
SEC$MAP_FROM_TYPE	CHAR(63)	Тип объекта, который будет отображён.
SEC$MAP_FROM	CHAR(255)	Имя объекта, из которого будет произведено отображение.
SEC$MAP_TO_TYPE	SMALLINT	Тип объекта, в который будет произведено отображение: • 0 — USER; • 1 — ROLE.
SEC$MAP_TO	CHAR(63)	Наименование объекта, в который будет произведено отображение (имя пользователя или роли).
SEC$DESCRIPTION	BLOB TEXT	Комментарий для отображения.

Б.3 SEC$USERS

Таблица хранит список пользователей в текущей базе данных безопасности.

Идентификатор столбца	Тип данных	Описание
SEC$USER_NAME	CHAR(63)	Имя пользователя.
SEC$FIRST_NAME	VARCHAR(32)	Имя.
SEC$MIDDLE_NAME	VARCHAR(32)	Отчество.
SEC$LAST_NAME	VARCHAR(32)	Фамилия.
SEC$ACTIVE	BOOLEAN	Флаг активности пользователя.
SEC$ADMIN	BOOLEAN	Отражает, имеет ли пользователь права RDB$ADMIN в базе данных безопасности.
SEC$DESCRIPTION	BLOB TEXT	Комментарий к пользователю.
SEC$PLUGIN	CHAR(63)	Имя плагина управления пользователями, с помощью которого был создан данный пользователь.

Б.4 SEC$USER_ATTRIBUTES

Таблица хранит сведения о дополнительных атрибутах пользователей.

Идентификатор столбца	Тип данных	Описание
SEC$USER_NAME	CHAR(63)	Имя пользователя.
SEC$KEY	VARCHAR(63)	Имя атрибута.

(разрыв таблицы)

(разрыв таблицы)

Идентификатор столбца	Тип данных	Описание
SEC$VALUE	VARCHAR(255)	Значение атрибута.
SEC$PLUGIN	CHAR(63)	Имя плагина управления пользователями, с помощью которого был создан данный пользователь.

Б.5 SEC$POLICIES

Таблица хранит сведения о созданных политиках безопасности.

Идентификатор столбца	Тип данных	Описание
SEC$POLICY_NAME	VARCHAR(32)	Имя политики безопасности. Политика по умолчанию имеет значение поля равное DEFAULT.
SEC$PSWD_NEED_CHAR	INTEGER	Минимально допустимое число буквенных символов в пароле.
SEC$PSWD_NEED_DIGIT	INTEGER	Минимально допустимое число цифровых символов в пароле.
SEC$PSWD_NEED_DIFF_CASE	INTEGER	Необходимость наличия в пароле буквенных символов в различных регистрах.
SEC$PSWD_MIN_LEN	INTEGER	Минимально допустимая длина пароля.
SEC$PSWD_VALID_DAYS	INTEGER	Срок действия пароля в днях.
SEC$PSWD_UNIQUE_COUNT	INTEGER	Количество не повторяющихся подряд паролей.
SEC$PSWD_NEED_SPECIAL	INTEGER	Количество специальных символов, требуемых в пароле. Специальным считается любой печатный символ, за исключением букв, цифр и пробельного символа.
SEC$MAX_FAILED_COUNT	INTEGER	Максимальное число ошибок при прохождении аутентификации.
SEC$AUTH_FACTORS	VARCHAR(64)	Методы аутентификации хранятся в виде строки, состоящей из символов: • S: SRP • L: Legacy (традиционная) • W: WIN_SSPI (доверенная) • G: GSS • C: Certificate • P: GostPassword • V: VerifyServer
SEC$MAX_UNUSED_DAYS	INTEGER	Максимальное время неактивности учетных записей пользователя в днях.

Приложение B Схема LDAP

B.1 OpenLDAP

```
attributetype ( 1.2.643.2.63.1.1.1
        NAME 'rdbPassword'
        DESC 'Native RDB password'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.2
        NAME 'rdbSecurePassword'
        DESC 'Secure RDB password'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.3
        NAME 'rdbPasswordAlgorithm'
        DESC 'RDB password hashing algorithm'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.4
        NAME 'rdbPasswordHistory'
        DESC 'RDB password change history'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.5
        NAME 'rdbPolicy'
        DESC 'User policy in security database'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.6
        NAME 'rdbPasswordTime'
        DESC 'Date/time of last password change'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.7
        NAME 'rdbFailedCount'
        DESC 'Count of failed authentication events'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.8
        NAME 'rdbAccessTime'
        DESC 'Date/time when ends user ban'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
        SINGLE-VALUE )
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
attributetype ( 1.2.643.2.63.1.1.9
        NAME 'rdbMlsLevel'
        DESC 'Security level'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.10
        NAME 'rdbMlsCompartment'
        DESC 'User compartment'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.11
        NAME 'rdbSrpVerifier'
        DESC 'RDB password for SRP protocol'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.12
        NAME 'rdbSrpSalt'
        DESC 'Salt for SRP'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.13
        NAME 'rdbActive'
        DESC 'Blocking flag'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.14
        NAME 'rdbLegacyHistory'
        DESC 'Legacy password change history'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.15
        NAME 'rdbSrpHistory'
        DESC 'SRP password change history'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.16
        NAME 'rdbLegacyPasswordTime'
        DESC 'Date/time of last legacy password change'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
        SINGLE-VALUE )

attributetype ( 1.2.643.2.63.1.1.17
        NAME 'rdbSrpPasswordTime'
        DESC 'Date/time of last SRP password change'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
        SINGLE-VALUE )

objectclass ( 1.2.643.2.63.2.1
        NAME 'rdbAuth'
        SUP top
        AUXILIARY
        DESC 'RDB authentication data'
        MAY ( rdbPassword $ rdbSecurePassword $
              rdbPasswordAlgorithm $ rdbPasswordHistory $
              rdbPolicy $ rdbPasswordTime $ rdbFailedCount $ rdbAccessTime $
              rdbMlsLevel $ rdbMlsCompartment $
              rdbSrpVerifier $ rdbSrpSalt $ rdbActive $
              rdbLegacyHistory $ rdbSrpHistory $
              rdbLegacyPasswordTime $ rdbSrpPasswordTime
        )
)
```

Приложение Г База данных безопасности

Информация обо всех пользователях баз данных СУБД «РЕД База Данных» хранится в общей базе данных безопасности, которая расположена в корневой директории каталога установки сервера и называется `security5.fdb`. В этой базе хранятся параметры пользователей системы, созданных с помощью различных плагинов управления пользователями, политики безопасности, хэши старых паролей. Эти данные располагаются в системных таблицах, которые представлены в таблице Г.1.

Таблица Г.1 — Список системных таблиц `security5.fdb`

Таблица	Содержание
`PLG$USERS`	Хранит параметры пользователей системы, созданных с помощью плагина управления пользователями `Legacy_UserManager`
`PLG$SRP`	Хранит параметры пользователей системы, созданных с помощью плагина управления пользователями `Srp`
`PLG$MF`	Хранит параметры пользователей системы, созданных с помощью плагина управления пользователями `GostPassword_Manager`
`PLG$POLICIES`	Хранит политики учетных записей пользователей
`PLG$PASSWD_HISTORY`	Хранит хэши старых паролей
`PLG$USER_POLICY`	Хранит информацию о назначенных пользователям политиках

Г.1 PLG$USERS

Хранит параметры пользователей системы, созданных с помощью плагина управления пользователями `Legacy_UserManager`.

Идентификатор столбца	Тип данных	Описание
`PLG$USER_NAME`	`VARCHAR(63)`	Уникальное имя пользователя (логин).
`PLG$GROUP_NAME`	`VARCHAR(63)`	Название группы.
`PLG$UID`	`INTEGER`	ID пользователя.
`PLG$GID`	`INTEGER`	ID группы.
`PLG$PASSWD`	`VARBINARY(64)`	Пароль.
`PLG$COMMENT`	`BLOB SUB_TYPE TEXT SEGMENT 80`	Комментарий.
`PLG$FIRST_NAME`	`VARCHAR(32)`	Имя пользователя.
`PLG$MIDDLE_NAME`	`VARCHAR(32)`	Отчество пользователя.
`PLG$LAST_NAME`	`VARCHAR(32)`	Фамилия пользователя.
`PLG$PASSWD_TIME`	`TIMESTAMP`	Дата установки пароля

Г.2 PLG$SRP

Идентификатор столбца	Тип данных	Описание
PLG$USER_NAME	VARCHAR(63)	Уникальное имя пользователя (логин).
PLG$VERIFIER	VARCHAR(128)	Зашифрованный пароль.
PLG$SALT	VARCHAR(32)	Некоторая случайная последовательность для формирования VERIFIER
PLG$COMMENT	BLOB SUB_TYPE TEXT SEGMENT 80	Комментарий.
PLG$FIRST	VARCHAR(32)	Имя пользователя.
PLG$MIDDLE	VARCHAR(32)	Отчество пользователя.
PLG$LAST	VARCHAR(32)	Фамилия пользователя.
PLG$ATTRIBUTES	BLOB SUB_TYPE TEXT SEGMENT 80	Пользовательские атрибуты
PLG$ACTIVE	BOOLEAN	Состояние пользователя: активное или неактивное
PLG$PASSWD_TIME	TIMESTAMP	Дата установки пароля

Г.3 PLG$MF

Идентификатор столбца	Тип данных	Описание
PLG$USER_NAME	VARCHAR(63)	Уникальное имя пользователя (логин).
PLG$HASH_ALG	VARCHAR(32)	Название алгоритма хэширования. Этот алгоритм будет использован для проверки пароля. Если поле не указано, то используется старый алгоритм хэширования.
PLG$MF_PASSWD	VARCHAR(256)	хэш пароля многофакторного пользователя.
PLG$COMMENT	BLOB SUB_TYPE TEXT SEGMENT 80	Комментарий.
PLG$FIRST	VARCHAR(32)	Имя пользователя.
PLG$MIDDLE	VARCHAR(32)	Отчество пользователя.
PLG$LAST	VARCHAR(32)	Фамилия пользователя
PLG$PASSWD_TIME	TIMESTAMP	Дата установки пароля
PLG$ATTRIBUTES	BLOB SUB_TYPE TEXT SEGMENT 80	Пользовательские атрибуты.
PLG$ACTIVE	BOOLEAN	Состояние пользователя: активное или неактивное.

Г.4 PLG$POLICIES

Используется для хранения политик учетных записей пользователей.

Идентификатор столбца	Тип данных	Описание
PLG$POLICY_NAME	VARCHAR(32)	Имя политики безопасности. Политика по умолчанию имеет значение поля равное DEFAULT.

(разрыв таблицы)

(разрыв таблицы)

Идентификатор столбца	Тип данных	Описание
PLG$PSWD_NEED_CHAR	INTEGER	Минимально допустимое число буквенных символов в пароле.
PLG$PSWD_NEED_DIGIT	INTEGER	Минимально допустимое число цифровых символов в пароле
PLG$PSWD_NEED_DIFF_CASE	INTEGER	Необходимость наличия в пароле буквенных символов в различных регистрах
PLG$PSWD_MIN_LEN	INTEGER	Минимально допустимая длина пароля
PLG$PSWD_VALID_DAYS	INTEGER	Срок действия пароля в днях.
PLG$PSWD_UNIQUE_COUNT	INTEGER	Количество не повторяющихся подряд паролей
PLG$MAX_FAILED_COUNT	INTEGER	Максимальное число ошибок при прохождении аутентификации.
PLG$AUTH_FACTORS	VARCHAR(64)	Методы аутентификации хранятся в виде строки, состоящей из символов: • S: SRP • L: Legacy (традиционная) • W: WIN_SSPI (доверенная) • G: GSS • C: Certificate • P: GostPassword • V: VerifyServer
PLG$MAX_UNUSED_DAYS	INTEGER	Максимальное время неактивности учетных записей пользователя в днях.

Г.5 PLG$PASSWD_HISTORY

Хранит хэши старых паролей.

Идентификатор столбца	Тип данных	Описание
PLG$KEY_ID	INTEGER	Суррогатный первичный ключ
PLG$PLUGIN	VARCHAR(64)	Плагин управления пользователями
PLG$USER_NAME	VARCHAR(63)	Ссылается на логин пользователя, определенный в системной таблице RDB$USERS.
PLG$PASSWD	VARBINARY(64)	Пароль Legacy
PLG$MF_PASSWD	VARCHAR(256)	Хэш пароля многофакторного пользователя
PLG$HASH_ALG	VARCHAR(32)	Название алгоритма хэширования. Этот алгоритм будет использован для проверки пароля. Если поле не указано, то используется старый алгоритм хэширования.
PLG$VERIFIER	VARCHAR(128)	Зашифрованный пароль (верификатор) SRP
PLG$SALT	VARCHAR(32)	Соль SRP

Г.6 PLG$USER_POLICY

Информация о назначенных пользователям политиках.

Идентификатор столбца	Тип данных	Описание
PLG$USER_NAME	VARCHAR(63)	Уникальное имя пользователя (логин)
PLG$POLICY_NAME	VARCHAR(32)	Имя политики безопасности
PLG$FAILED_COUNT	INTEGER	Число ошибок при прохождении аутентификации
PLG$ACCESS_TIME	TIMESTAMP	Время после которого пользователю будет разрешен доступ
PLG$LAST_ONLINE	TIMESTAMP	Дата последней активности пользователя