Руководство администратора Firebird 3.0

### Денис Симонов

### Дмитрий Еманов

### Дмитрий Кузьменко

### Алексей Ковязин

### Kamala Vadlamani

Содержание

* [Введение](#intro)
* [I. Установка и настройка](#install-and-settings)
  + [1. Установка, обновление и запуск сервера](#install)
    - [Установка на ОС Linux](#install-linux)
      * [Установка из репозитория](#install-linux-repo)
      * [Ручная установка из tar архива](#install-linux-tar)
      * [Ручная установка клиента](#install-linux-client)
    - [Установка на ОС Windows](#install-windows)
      * [Установка из исполняемого пакета](#install-windows-exe)
      * [Ручная установка из ZIP архива](#install-windows-zip)
        + [Инициализация SYSDBA](#install-windows-init-sysdba)

[Инициализация SYSDBA с помощью ISQL](#install-windows-isql)

[Инициализация SYSDBA с помощью GSEC](#install-windows-gsec)

* + - * + [Конфигурация](#install-windows-config)

[Режим сервера](#install-windows-config-mode)

[Авторизация с предыдущих версий клиента Firebird](#install-windows-config-auth)

[Одновременный запуск нескольких Firebird](#install-windows-config-with_other)

* + - * + [Запись в реестре](#install-windows-instreg)
        + [Установка и запуск Firebird как службы](#install-windows-instsvc)
        + [Использование install\_service.bat и uninstall\_service.bat](#install-windows-bat)
        + [Запуск Firebird как приложения](#install-windows-app)
        + [Установка клиента](#install-windows-client)

[Утилита instclient](#install-windows-client-instclient)

* + - * + [Обновление](#install-windows-update)
        + [Встроенный сервер](#install-windows-embedded)

[Конфигурация](#install-windows-embedded-config)

* + [2. Состав файлов сервера](#files)
    - [Файлы конфигурации](#d0e969)
    - [Инструменты администрирования и сервисы](#d0e1011)
    - [Динамические библиотеки](#d0e1120)
    - [Плагины](#d0e1214)
    - [Включаемые файлы](#d0e1318)
    - [Примеры](#d0e1326)
    - [Документации](#d0e1334)
    - [Другие файлы](#d0e1342)
  + [3. Настройка операционной системы для Firebird](#d0e1360)
    - [Настройка Linux](#d0e1364)
      * [Лимиты Linux](#d0e1368)
      * [Необходимые права для Firebird на Linux](#d0e1372)
      * [Рекомендуемые параметры сети](#d0e1376)
      * [Рекомендуемые параметры памяти](#d0e1380)
      * [Рекомендуемые параметры диска](#d0e1384)
    - [Настройка Windows](#d0e1388)
      * [Необходимые права для Firebird на Linux](#d0e1392)
      * [Рекомендуемые параметры](#d0e1396)
* [II. Утилиты командной строки](#command-line-tools)
  + [4. ISQL. Интерактивные запросы](#isql)
    - [Запуск isql](#isql-invoking)
    - [Переключатели командной строки](#isql-switches)
      * [Примеры](#isql-switches-examples)
    - [Начало сессии Isql](#isql-start_session)
    - [Завершение сессии Isql](#isql-end_session)
    - [Получение справки](#isql-help)
    - [Соединение с базой данных](#isql-connect)
      * [Использование псевдонимов БД](#isql-connect-aliasedb)
      * [Соединение с БД на удалённом сервере](#isql-connect-remotely)
      * [Имя пользователя, пароль и роль](#isql-connect-user)
      * [Набор символов](#isql-connect-charset)
      * [Примеры](#isql-connect-examples)
    - [Создание базы данных](#isql-create_db)
    - [Установка переменных окружения ISC\_USER и ISC\_PASSWORD](#isql-env)
    - [SQL диалект](#isql-sql_dialect)
    - [Символ терминатора](#isql-terminator)
    - [Подсказки ISQL](#isql-prompts)
    - [Обработка ошибок и поддержка исключений](#isql-errors)
    - [Обработка транзакций](#isql-transactions)
    - [Обработка скрипта](#isql-script_handling)
    - [ISQL команды](#isql-commands)
      * [BLOBDUMP](#isql-commands-blobdump)
      * [BLOBVIEW](#isql-commands-blobview)
      * [EDIT](#isql-commands-edit)
      * [HELP](#isql-commands-help)
      * [ADD](#isql-commands-add)
      * [COPY](#isql-commands-copy)
      * [INPUT](#isql-commands-input)
      * [OUTPUT](#isql-commands-output)
      * [SET](#isql-commands-set)
      * [SHELL](#isql-commands-shell)
      * [EXIT](#isql-commands-exit)
      * [QUIT](#isql-commands-quit)
    - [ISQL SET команды](#isql-commands_set)
      * [SET](#isql-commands_set-set)
      * [SET AUTODDL](#isql-commands_set-set_autoddl)
      * [SET BAIL](#isql-commands_set-set_bail)
      * [SET BLOB](#isql-commands_set-set_blob)
      * [SET BULK\_INSERT](#isql-commands_set-set_bulk_insert)
      * [SET COUNT](#isql-commands_set-set_count)
      * [SET ECHO](#isql-commands_set-set_echo)
      * [SET EXPLAIN](#isql-commands_set-set_explain)
      * [SET HEADING](#isql-commands_set-set_heading)
      * [SET LIST](#isql-commands_set-set_list)
      * [SET NAMES](#isql-commands_set-set_names)
      * [SET PLAN](#isql-commands_set-set_plan)
      * [SET PLANONLY](#isql-commands_set-set_planonly)
      * [SET SQLDA\_DISPLAY](#isql-commands_set-set_sqlda)
      * [SET SQL DIALECT](#isql-commands_set-set_sqldialect)
      * [SET STATs](#isql-commands_set-set_stats)
      * [SET TIME](#isql-commands_set-set_time)
      * [SET TERM](#isql-commands_set-set_term)
      * [SET WARNINGS](#isql-commands_set-set_warnings)
      * [SET WIDTH](#isql-commands_set-set_width)
    - [ISQL SHOW команды](#isql-commands_show)
      * [SHOW CHECK](#isql-commands_show-show_check)
      * [SHOW COLLATION](#isql-commands_show-show_collations)
      * [SHOW COMMENTS](#isql-commands_show-show_comments)
      * [SHOW DATABASE](#isql-commands_show-show_db)
      * [SHOW DEPENDENCIES](#isql-commands_show-show_dependencies)
      * [SHOW DOMAIN](#isql-commands_show-show_domain)
      * [SHOW EXCEPTION](#isql-commands_show-show_exeption)
      * [SHOW FILTER](#isql-commands_show-show_filter)
      * [SHOW FUNCTION](#isql-commands_show-show_function)
      * [SHOW GENERATOR](#isql-commands_show-show_generator)
      * [SHOW GRANTS](#isql-commands_show-show_grants)
      * [SHOW INDEXES](#isql-commands_show-show_indexes)
      * [SHOW MAPPING](#isql-commands_show-show_mapping)
      * [SHOW PACKAGE](#isql-commands_show-show_package)
      * [SHOW PROCEDURE](#isql-commands_show-show_procedure)
      * [SHOW ROLE](#isql-commands_show-show_role)
      * [SHOW SECCLASSES](#isql-commands_show-show_secclasses)
      * [SHOW SEQUENCE](#isql-commands_show-show_sequence)
      * [SHOW SQL DIALECT](#isql-commands_show-show_sql_dialect)
      * [SHOW SYSTEM](#isql-commands_show-show_system)
      * [SHOW TABLE](#isql-commands_show-show_table)
      * [SHOW TRIGGER](#isql-commands_show-show_trigger)
      * [SHOW USERS](#isql-commands_show-show_users)
      * [SHOW VERSION](#isql-commands_show-show_version)
      * [SHOW VIEW](#isql-commands_show-show_view)
  + [5. GFIX](#gfix)
  + [6. NBACKUP](#nbackup)
  + [7. FB\_LOCK\_PRINT](#d0e6389)
  + [8. FBTRACEMGR](#d0e6393)
  + [9. GSEC](#d0e6397)
  + [10. GSPLIT](#d0e6401)
* [III.](#tasks)
  + [11. Типичные проблемы с установкой](#d0e6407)
    - [На Windows](#d0e6411)
    - [На Linux](#d0e6415)
  + [12. Разрешение проблем с сетевыми настройками между клиентом и сервером Firebird](#d0e6419)
  + [13. Настройка сервера и клиента](#config)
    - [Типы параметров](#config-param-types)
    - [Область действия](#config-scope)
    - [Макроподстановки](#config-macro)
    - [Включение других файлов](#config-includes)
    - [Параметры уровня базы данных](#config-perdb)
      * [Формат конфигурационных записей](#config-perdb-format)
      * [Доступные параметры](#config-perdb-params)
    - [Общие настройки](#config-common)
      * [DatabaseAccess](#config-param-DatabaseAccess)
      * [RemoteAccess](#config-param-RemoteAccess)
      * [ExternalFileAccess](#config-param-ExternalFileAccess)
      * [UdfAccess](#config-param-UdfAccess)
      * [TempDirectories](#config-param-TempDirectories)
      * [AuditTraceConfigFile](#config-param-AuditTraceConfigFile)
      * [MaxUserTraceLogSize](#config-param-MaxUserTraceLogSize)
      * [DefaultDbCachePages](#config-param-DefaultDbCachePages)
      * [DatabaseGrowthIncrement](#config-param-DatabaseGrowthIncrement)
      * [FileSystemCacheThreshold](#config-param-FileSystemCacheThreshold)
      * [FileSystemCacheSize](#config-param-FileSystemCacheSize)
      * [RemoteFileOpenAbility](#config-param-RemoteFileOpenAbility)
      * [TempBlockSize](#config-param-TempBlockSize)
      * [TempCacheLimit](#config-param-TempCacheLimit)
      * [AuthServer и AuthClient](#config-param-AuthServer)
      * [UserManager](#config-param-UserManager)
      * [TracePlugin](#config-param-TracePlugin)
      * [WireCryptPlugin](#config-param-WireCryptPlugin)
      * [KeyHolderPlugin](#config-param-KeyHolderPlugin)
      * [Providers](#config-param-Providers)
      * [DeadlockTimeout](#config-param-DeadlockTimeout)
      * [MaxUnflushedWrites](#config-param-MaxUnflushedWrites)
      * [MaxUnflushedWriteTime](#config-param-MaxUnflushedWriteTime)
      * [BugcheckAbort](#config-param-BugcheckAbort)
      * [RelaxedAliasChecking](#config-param-RelaxedAliasChecking)
      * [ConnectionTimeout](#config-param-ConnectionTimeout)
      * [WireCrypt](#config-param-WireCrypt)
      * [WireCompression](#config-param-WireCompression)
      * [DummyPacketInterval](#config-param-DummyPacketInterval)
      * [RemoteServicePort или RemoteServiceName](#config-param-RemoteServicePort)
      * [RemoteAuxPort](#config-param-RemoteAuxPort)
      * [TcpRemoteBufferSize](#config-param-TcpRemoteBufferSize)
      * [TcpNoNagle](#config-param-TcpNoNagle)
      * [IPv6V6Only](#config-param-IPv6V6Only)
      * [RemoteBindAddress](#config-param-RemoteBindAddress)
      * [LockMemSize](#config-param-LockMemSize)
      * [LockAcquireSpins](#config-param-LockAcquireSpins)
      * [LockHashSlots](#config-param-LockHashSlots)
      * [EventMemSize](#config-param-EventMemSize)
    - [Настройки ядра](#config-engine)
      * [CpuAffinityMask](#config-param-CpuAffinityMask)
      * [GCPolicy](#config-param-GCPolicy)
      * [SecurityDatabase](#config-param-SecurityDatabase)
    - [Настройки для Windows систем](#config-windows)
      * [GuardianOption](#config-param-GuardianOption)
      * [ProcessPriorityLevel](#config-param-ProcessPriorityLevel)
      * [IpcName](#config-param-IpcName)
      * [RemotePipeName](#config-param-RemotePipeName)
    - [Настройки для Unix/Linux систем](#config-unix)
      * [Redirection](#config-param-Redirection)
    - [Настройки архитектуры](#config-servermode)
      * [ServerMode](#config-param-ServerMode)
  + [14. Конфигурирование и подбор аппаратного обеспечения](#hardware)
    - [Немного теории](#hardware-theory)
      * [Функциональные модули сервера](#hardware-theory-modules)
      * [Базовые операции с hardware](#hardware-theory-hardware_operations)
      * [Параллельность выполнения операций](#d0e9458)
      * [Потоки данных](#d0e9474)
      * [Резервное копирование](#d0e9491)
    - [Выбор подходящего железа](#d0e9504)
      * [CPU](#d0e9514)
        + [Какие запросы преобладают в приложении?](#d0e9529)
        + [Количество активных соединений с БД в среднем и в моменты пиковой нагрузки](#d0e9538)
        + [Версия и архитектура Firebird](#d0e9559)
      * [RAM](#d0e9566)
        + [ECC RAM](#d0e9578)
        + [Расчет необходимого объема RAM](#d0e9583)

[Формулы расчета необходимого объема RAM](#d0e9653)

[Излишняя память](#d0e9737)

* + - * [Дисковая подсистема](#d0e9796)
        + [Отдельные диски для всего](#d0e9801)
        + [SSD для базы данных](#d0e9810)
        + [Диск для временных файлов](#d0e9824)
        + [HDD for backups](#d0e9859)
        + [HDD для базы данных](#d0e9909)
        + [Надежность и RAID](#d0e9918)
        + [Настройка RAID](#d0e9936)
        + [СХД](#d0e9966)
    - [Краткие выводы и рекомендации](#d0e10005)
  + [15. Использование инструментов резервного копирования](#d0e10058)
    - [gbak](#d0e10062)
    - [nbackup](#d0e10066)
  + [16. Использование инструмента валидации и починки БД gfix](#d0e10070)
  + [17. Безопасность](#d0e10074)
    - [Плагины аутентификации и их использование](#d0e10078)
    - [Плагины шифрования и их использование](#d0e10082)
  + [18. Репликация (Firebird 4.0)](#d0e10086)
  + [19. Использование инструментов мониторинга Firebird](#d0e10090)
    - [Использование fb\_lock\_print](#d0e10094)
    - [Использование fbtracemgr для Trace API](#d0e10098)
    - [Использование таблиц мониторинга MON$](#d0e10102)
* [Алфавитный указатель](#d0e10106)

## Глава 1. Установка, обновление и запуск сервера

Скачать дистрибутивы Firebird можно на официальном сайте <https://firebirdsql.org> в разделе Downloads.

## Установка на ОС Linux

### Установка из репозитория

### Ручная установка из tar архива

### Ручная установка клиента

## Установка на ОС Windows

Firebird под Windows распространяется в двух вариантах: в виде выполняемого файла установщика с расширением exe и в виде zip архивов. Для 32-разрядного варианта в имени файла содержится суффикс Win32, для 64-разрядного — x64. Отладочный вариант имеет дополнительный суффикс pdb.

### Установка из исполняемого пакета

Скачайте и запустите инсталяционный пакет разрядности, соответствующей разрядности операционной системы (Win32 может работать на 64-разярдной Windows, но это не имеет смысла). Инсталляция СУБД Firebird 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.

### Примечание

Для установки Firebird необходимы права администратора.

Рекомендуется перед установкой этого пакета ДЕИНСТАЛЛИРОВАТЬ все предыдущие версии Firebird или InterBase. Особенно важно убедиться, что файлы fbclient.dll и gds32.dll удалены из каталога system32.

Если вам необходимо установить Firebird 3.0 совместно с другими версиями, то это необходимо сделать из zip-архива.

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

Рисунок 1.1. Выбор языка установки

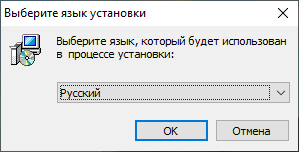
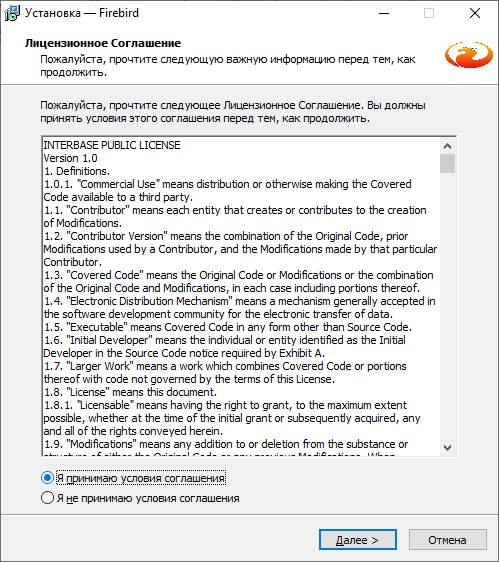
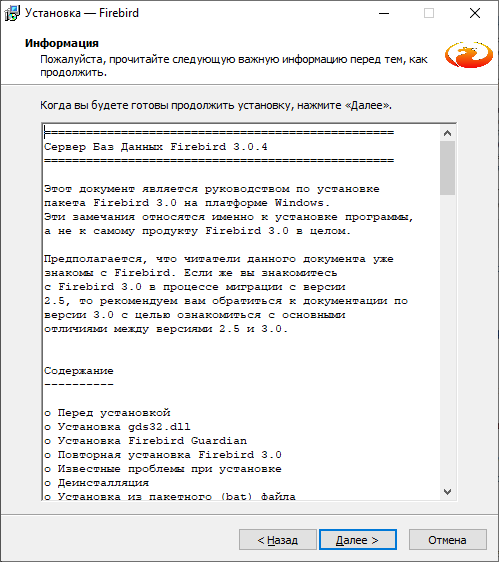


Рисунок 1.2. Лицензионное соглашение и информация об установке

В процессе инсталляции вам будет предложено выбрать папку для установки, а также выбрать устанавливаемые компоненты.

Рисунок 1.3. Выбор компонентов для установки

|  |
| --- |
| Выбор компонентов для установки |

#### Выбор архитектуры сервера

Далее будет предложено выбрать архитектуру сервера и некоторые дополнительные опции. Доступны следующие архитектуры:

* **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, страничный кэш 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 (общий странчный кеш и память 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 метаданных;
  + поддержка мультипроцессорности: потоки ОС легко распараллеливаются;
  + 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 страничным кэшем и кэшем метаданных;
  + поддержка мультипроцессорности: потоки ОС легко распараллеливаются;
  + возможный сбой в одном процессе разорвет все подключения.

Рисунок 1.4. Выбор архитектуры

|  |
| --- |
| Выбор архитектуры |

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.

#### Установка Guardian

Мы собираемся отказаться от использования Firebird Guardian в будущих версиях. Он не работает в случае архитектуры Classic Server и программа установки не предлагает его установку, если выбрана данная архитектура. Для архитектур SuperServer и SuperClassic установка Firebird Guardian возможна, но по умолчанию не выбрана.

#### Способ запуска сервера Firebird

Сервер может быть запущен в качестве приложения (менее предпочтительный вариант) или в качестве службы. Для запуска в качестве приложения используется следующая команда:

firebird -a

#### Установка gds32.dll

По умолчанию данная библиотека не копируется в системные папки Windows в процессе установки. Мы не можем гарантировать, что необходимые библиотеки MS VC runtime будут установлены и доступны в системе. Тем не менее, данная возможность доступна опционально в процессе установки, так же как и копирование fbclient.dll в системные папки Windows. Если вы используете данную возможность, то вы должны убедиться, что библиотеки MS VC runtime версии 10.0 установлены в системе.

#### Авторизация с предыдущих версий клиента Firebird

В Firebird 3.0 по умолчанию используется безопасная парольная аутентификация (SRP). Клиенты Firebird 2.5 и более ранние версии использовали традиционную аутентификацию (Legacy\_Auth), которая отключена в Firebird 3.0 по-умолчанию, поскольку не является безопасной.

Вы можете разрешить авторизацию предыдущими версиями клиентов Firebird в процессе инсталяции. Это можно сделать и после установки изменив параметры [AuthServer](#config-param-AuthServer), [UserManager](#config-param-UserManager) и [WireCrypt](#config-param-WireCrypt).

#### Пароль SYSDBA

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

Рисунок 1.5. Задание пароля SYSDBA

|  |
| --- |
| Задание пароля SYSDBA |

После этого шага будет отображено окно, в котором отображаются опции выбранные пользователем, если вас всё устраивает можно начать автоматическую установку сервера.

### Ручная установка из ZIP архива

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

#### Инициализация SYSDBA

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

### Примечание

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

##### Инициализация SYSDBA с помощью ISQL

Запустите инструмент для выполнения интерактивных запросов isql.exe. Соединитесь с базой данных безопасности в режиме встроенного сервера, указав при этом пользователя SYSDBA без пароля.

### Примечание

Пользователя SYSDBA ещё не существует в базе данных безопасности, но в embedded режиме пользователь и его пароль не проверяется, и Firebird доверяет любому указанному имени пользователя.

Выполите SQL запрос для создания пользователя SYSDBA:

CREATE USER SYSDBA PASSWORD '<password>';

Пользователь SYSDBA инициализирован, можно выходить из интерактивного режима.

Пример 1.1. Инициализация SYSDBA через isql

c:\Firebird\3.0>isql  
Use CONNECT or CREATE DATABASE to specify a database  
SQL> connect security.db user SYSDBA;  
Database: security.db, User: SYSDBA  
SQL> CREATE USER SYSDBA PASSWORD 'm8ku234pp';  
SQL> exit;  
  
c:\Firebird\3.0>

##### Инициализация SYSDBA с помощью GSEC

Запустите gsec.exe указав пользователя SYSDBA и базу данных security.db.

Выполите команду для добавления пользователя SYSDBA:

add SYSDBA -pw <password>

Пример 1.2. Инициализация SYSDBA через GSEC

c:\Firebird\3.0>gsec -user SYSDBA -database security.db  
\*\*\* gsec is deprecated, will be removed soon \*\*\*  
  
GSEC> add SYSDBA -pw m8ku234pp  
GSEC> quit  
  
c:\Firebird\3.0>

### Примечание

Инструмент gsec.exe является устаревшим, многие возможности доступные через SQL не доступны в нём.

#### Конфигурация

##### Режим сервера

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

ServerMode = Classic

Подробнее о режимах севера и параметре *ServerMode* читайте в [Выбор архитектуры сервера](#install-windows-exe-servermode) и [ServerMode](#config-param-ServerMode).

##### Авторизация с предыдущих версий клиента Firebird

В Firebird 3.0 по умолчанию используется безопасная парольная аутентификация (SRP). Клиенты Firebird 2.5 и более ранние версии использовали традиционную аутентификацию (Legacy\_Auth), которая отключена в Firebird 3.0 по-умолчанию, поскольку не является безопасной.

Для поддержки традиционной аутентификации необходимо изменить следующие параметры [AuthServer](#config-param-AuthServer), [UserManager](#config-param-UserManager) и [WireCrypt](#config-param-WireCrypt).

Пример 1.3. Включение авторизации с предыдущих версий клиента Firebird

AuthServer = Srp256, Srp, Legacy\_Auth  
UserManager = Srp, Legacy\_UserManager  
WireCrypt = Enabled

### Примечание

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

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

### Важно

Одноименные пользователи в разных менеджерах пользователей — это разные пользователи.

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

c:\Firebird\3.0>isql  
Use CONNECT or CREATE DATABASE to specify a database  
SQL> connect security.db user sysdba;  
Database: security.db, User: SYSDBA  
SQL> ALTER USER SYSDBA SET PASSWORD 'er34gfde' USING PLUGIN Legacy\_UserManager;  
SQL> exit;  
  
c:\Firebird\3.0>

##### Одновременный запуск нескольких Firebird

Для одновременного запуск нескольких Firebird, необходимо развести их по разным портам tcp (если конечно слушатель запущен в режиме прослушивания TCP/IP). Для этого необходимо изменить в firebird.conf параметр *RemoteServicePort*. Например, если у вас уже есть один сервер который слушает порт 3050, то необходимо установить любой другой свободный порт, например 3051. В этом случае в строке подключения необходимо будет указывать новый порт.

RemoteServicePort = 3051

Если необходимо обеспечить также одновременную работоспособность по локальному протоколу XNET и через именованные каналы WNET, то необходимо так же изменить параметры *IpcName* и *RemotePipeName*. Однако стоит учесть, что эти параметры придётся изменять и на стороне клиента через DPB.

### Важно

Если вы хотите запускать несколько экземпляров Firebird, то не используйте утилиту instreg.exe, которая записывает путь к серверу в реестр Windows.

#### Запись в реестре

Утилита утилита instreg.exe позволяет прописывать в реестр Windows путь к корневому каталогу сервера. Для этого необходимо запустить команду

instreg install -z

Для удаления записи из реестра необходимо запустит следующую команду

instreg remove -z

### Важно

Если вы хотите запускать несколько экземпляров Firebird, то не используйте утилиту instreg.exe.

### Примечание

Переключатель -z не является обязательным, он позволяет вывести установленную или удаленную версию Firebird.

#### Установка и запуск Firebird как службы

Утилита instsvc.exe записывает, удаляет или меняет информацию о запуске сервера в базе сервисов операционной системы. Кроме того, она позволяет управлять запуском и остановкой сервиса. Если запустить её без параметров, то будет выведена справка по командам и параметрам.

instsvc

Usage:  
 instsvc i[nstall]  
 [ -a[uto]\* | -d[emand] ]  
 [ -g[uardian] ]  
 [ -l[ogin] username [password] ]  
 [ -n[ame] instance ]  
 [ -i[nteractive] ]  
  
 sta[rt] [ -b[oostpriority] ]  
 [ -n[ame] instance ]  
 sto[p] [ -n[ame] instance ]  
 q[uery]  
 r[emove] [ -n[ame] instance ]  
  
  
 '\*' denotes the default values  
 '-z' can be used with any other option, prints version  
 'username' refers by default to a local account on this machine.  
 Use the format 'domain\username' or 'server\username' if appropriate.

Для установки сервиса необходимо ввести команду

instsvc install

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

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

instsvc install -n fb30

Для запуска службы воспользуйтесь командой

instsvc start

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

instsvc start -n fb30

Для остановки службы воспользуйтесь командой

instsvc stop

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

instsvc stop -n fb30

Для удаления сервиса необходимо ввести команду

instsvc remove

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

instsvc remove -n fb30

Для просмотра всех служб Firebird установленных в системе воспользуйтесь командой

instsvc query

Firebird Server - fb30 IS installed.  
 Status : running  
 Path : C:\Firebird\3.0\firebird.exe -s fb30  
 Startup : automatic  
 Run as : LocalSystem  
  
Firebird Server - fb40 IS installed.  
 Status : running  
 Path : C:\Firebird\4.0\firebird.exe -s fb40  
 Startup : automatic  
 Run as : LocalSystem

#### Использование install\_service.bat и uninstall\_service.bat

Для упрощения процедуру установки и удаления служб в ZIP архиве в комплекте с Firebird поставляются два BAT файла: install\_service.bat и uninstall\_service.bat.

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

install\_service.bat

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

install\_service.bat fb30

В этом случае процедура удаления службы Firebird выглядит следующим образом

uninstall\_service.bat

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

uninstall\_service.bat fb30

#### Запуск Firebird как приложения

Для запуска Firebird в качестве приложения достаточно выполнить команду

firebird -a

### Подсказка

Запуск Firebird как приложения удобен на компьютере разработчика. Это особенно полезно во время отладки UDF, UDR и различных плагинов Firebird.

#### Установка клиента

Если речь идёт об установке только о клиентской части, то обязательно требуется файл fbclient.dll. Клиент Firebird 3.0 обязательно требует наличие установленного Microsoft Runtime C++ 2010 соответствующей разрядности. Если данная библиотека не установлено, то можно скопировать дополнительные библиотеки, которые поставляются в ZIP архиве под Windows msvcp100.dll и msvcr100.dll.

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

Если требуется сжатие трафика при работе по TCP/IP, то потребуется так же библиотека zlib1.dll.

Для того чтобы клиентское приложение могло загрузить библиотеку fbclient.dll она должна располагаться либо рядом с приложением, либо в одной из директорий в которой производится поиск, например добавленной в PATH или системной директории для размещения общедоступных библиотек (system32 или SysWOW64).

### Важно

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

Если у вас ещё нет (и не планируется) установок сервера Firebird или его клиента, то вы можете зарегистрировать путь к корню Firebird в реестре.

instreg install

##### Утилита instclient

Утилита instclient.exe позволяет:

* установить клиентскую часть одной командой;
* установить клиентскую часть как fbclient.dll и/или gds32.dll;
* проверить наличие установленной библиотеки fbclient.dll и/или gds32.dll;
* удалить ранее установленные fbclient.dll и/или gds32.dll.

### Примечание

Утилита instclient.exe должна быть запущена с правами администратора.

Если запустить instclient.exe без параметров, то будет выведена краткая справка:

instclient

Usage:  
 instclient i[nstall] [ -f[orce] ] library  
 q[uery] library  
 r[emove] library  
  
 where library is: f[bclient] | g[ds32]

Для развёртывания клиентской библиотеки Firebird в системном каталоге Windows воспользуйтесь командой

instclient install fbclient

FBCLIENT.DLL has been installed to the System directory.

При развёртывании библиотеки проверяется версия и инкрементируется счётчик общих библиотек. Для отмены проверки версии вы можете воспользоваться переключателем -force.

Некоторые старые приложения требуют для работы клиентскую библиотеку gds32.dll (наследие Interbase). Кроме того, многие приложения требуют для работы не просто gds32.dll, но и проверяют версию этой библиотеки. Именно поэтому простое переименовывание fbclient.dll в gds32.dll для таких программ работать не будет — версия библиотеки окажется ниже 6.0 (т.к. соответствует версии Firebird, у которого нумерация версий идет с 1.0). Утилита instclient.exe позволяет не просто установить gds32.dll, но и изменить её версию на корректную.

instclient install gds32

GDS32.DLL has been installed to the System directory.

Утилита instclient.exe позволяет также проверить наличие установленной клиентской библиотеки с помощью команды

instclient q fbclient

Installed FBCLIENT.DLL version : 3.0.4.33054 (shared DLL count 1)

instclient q gds32

Installed GDS32.DLL version : 6.3.4.33054 (shared DLL count 2)

Для удаления воспользуйтесь командой

instclient remove fbclient

The FBCLIENT.DLL has been removed from the System directory.

instclient remove gds32

The GDS32.DLL has been removed from the System directory.

#### Обновление

В данном случае речь об обновлении в рамках пойнт-релизов (третья цифра в версии сервера) или так называемых сервис-паков. Обновление мажорной или минорной версии обычно требует гораздо больше действий, описанных в Release Note, и называется «Миграцией».

### Примечание

Обычно обновление в рамках пойнт-релизов не требует процедуры резервного копирования и восстановления базы данных средством gbak, но бывают исключения. Такие исключения обязательно описаны в Release Note, поэтому обязательно ознакомьтесь с ним перед обновлением.

Примерно те же действия требуются при обновлении на текущий снапшот. Снапшоты — это ежедневные сборки сервера Firebird, которые включают самые последние изменения текущей версии сервера. Это позволяет иметь самую свежую версию сервера с исправленными ошибками и улучшениями. Однако учтите, что снапшоты не проходят полный цикл тестирования и могут содержать новые ошибки или регрессии. Поэтому не обновляйтесь до снапшота без явной необходимости (исправленные в нём ошибки касаются вас непосредственно).

Алгоритм обновления поверх существующего сервера Firebird следующий:

1. Убедится что все пользователи отключены от всех баз данных обслуживаемых обновляемым сервером Firebird. С помощью команды

* gfix -shut -force <n>
* можно закрыть все подключения и запретить последующие.

1. Остановить сервер
2. Сделать резервную копию корневого каталога сервера
3. Распаковать из ZIP архива с пойнт-релизом или снапшотом файлы в корневую директорию сервера
4. Заменить новые файлы конфигурации (firebird.conf, databases.conf и другие), а также файл security3.fdb на прежние
5. Запустить обновленный сервер
6. Перевести базы данных в онлайн режим

* gfix -online

#### Встроенный сервер

Начиная с Firebird 3.0 встроенная (embedded) версия не распространяется отдельно. Всё что необходимо для развертывания Firebird Embedded находится в том же самом ZIP архиве, что и обычная версия.

Для функционирования Firebird Embedded необходимы следующие файлы:

* клиентская библиотека (fbclient.dll или libfbclient.so);
* движок для работы с ODS 12.0 (engine12.dll или libEngine12.so) из папки plugins;
* библиотека интернационализации, которая находится в каталоге intl;
* файл сообщений firebird.msg;
* для изменения некоторых параметров могут потребоваться firebird.conf и/или databases.conf;
* в Windows обязательно необходимы файлы ICU библиотеки;
* если в Windows не установлен MS VC++ Runtime 2010, то необходимы также файлы (msvcp100.dll и msvcr100.dll);
* если вам необходимы консольные инструменты администрирования, то скопируйте также gbak, gfix, gstat, isql, nbackup.

##### Конфигурация

По умолчанию Firebird работает в режиме Super сервера в том числе и в embedded варианте. Это не очень удобно, поскольку Super открывает файл базы данных в монопольном режиме, что обозначает что только один процесс может работать с базой данных. Для изменения режима запуска необходимо отредактировать (возможно раскомментировать) значение параметра *ServerMode* на Classic или SuperClassic в файле firebird.conf.

Поскольку в embedded варианте сетевой доступ не используется вы можете отредактировать также провайдеры *Providers*, удалив от туда лишние провайдеры Remote и Loopback.

Providers = Engine12

## Глава 2. Состав файлов сервера

## Файлы конфигурации

### 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. Если под управлением сервера Firebird находится более 1 базы данных, то рекомендуем все настройки уровня базы данных делать в этом файле.

### 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.

### firebird.conf

Файл содержит параметры конфигурации 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.

### udr\_engine.conf

Файл используется для настройки плагина UDR Engine — движок для подключения внешних процедур, функций и триггеров. Файл располагается в директории plugins.

### fbintl.conf

Файл конфигурации для библиотеки интернационализации. Файл располагается в директории intl.

## Инструменты администрирования и сервисы

В Windows исполняемые файлы имеют расширение .exe, в других операционных системах могут использоваться другие расширения или вовсе не использоваться.

### fb\_lock\_print

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

### fbguard

Исполняемый файл приложения Guardian. Он контролирует процесс сервера. Если сервер упал по какой-либо причине, Guardian автоматически перезапускает его.

### Примечание

В современных версиях Windows не требуется. Дело в том, что у служб можно установить свойство автоматического перезапуска в случае падения. Однако данная утилита может быть полезна, если Firebird запускается в качестве приложения.

### fbsvcmgr

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

### fbtracemgr

Утилита для работы в интерактивном режиме с трассировкой.

### firebird

Исполняемый файл сервера (слушатель).

### gbak

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

### gfix

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

### gpre

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

### gsec

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

### gsplit

Это фильтр, который позволяет преодолевать ограничения на максимальный размер файла, существующие в некоторых операционных системах, при создании резервной копии очень большой базы данных. Эта утилита поставляется только для ОС Windows и, к сожалению, кажется она не работает. К счастью, gbak позволяет разбивать файлы резервной копии на несколько частей, так что gsplit не требуется. В системах ОС Unix существует подходящие утилиты из операционной системы, которые могут быть использованы вместо gsplit, если это необходимо.

### gstat

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

### instclient

Назначение утилиты instclient состоит в том, что она:

* позволяет установить клиентскую часть Firebird одной командой;
* позволяет установить клиентскую часть как fbclient.dll, либо как gds32.dll;
* позволяет проверить наличие установленной библиотеки fbclient или gds32;
* позволяет удалить уже установленный в системе fbclient или gds32.

### instreg

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

### instsvc

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

### isql

Интерактивный инструмент, который позволяет выполнять запросы к базе данных.

### nbackup

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

### qli

Это Query Language Interpreter (интерпретатор языка запросов), интерактивный клиентский инструмент запросов. Он может обрабатывать операторы DDL и DML из SQL и GDML. Хотя уже есть isql и другие инструменты графического интерфейса сторонних разработчиков, qli все еще имеет значение по причине его способности осуществлять некоторые операции, до сих пор не реализованные в SQL Firebird. В отличие от isql, qli может одновременно соединяться более чем с одной базой данных и может симулировать обращение к нескольким базам данных в одном запросе.

## Динамические библиотеки

В Windows динамические библиотеки имеют расширение .dll, в Linux — .so, в Mac OS — .dylib.

### fbclient

Клиентская библиотека. Предоставляет интерфейс прикладного программирования (API) с функциями для подключения к сервисам, работы с базами данных и их созданию. Библиотека также выполняет роль y-valve для подключения и сопряжения различных плагинов.

В Windows имеет имя fbclient.dll, в Unix — libfbclient.so.

### ib\_util

Библиотека утилиты памяти.

В Windows имеет имя ib\_util.dll, в Unix — libib\_util.so.

### ICU

ICU — International Components for Unicode — библиотеки поддержки юникода.

В Windows используется библиотека поставляемая в комплекте Firebird, в Unix — используется системная библиотека. В Windows в комплекте с Firebird поставляются следующие файлы: icudt52l.dat, icudt52.dll, icuin52.dll, icuuc52.dll.

### Microsoft VC++ Runtime

Microsoft VC++ Runtime 2010 требуется только в Windows. Поставляется в комплекте с Firebird на случай, если библиотека соответствующей версии не установлена в системе. В комплекте Firebird под Windows входят файлы: msvcp100.dll, msvcr100.dll.

### zlib1.dll

Библиотека zlib1.dll используется для сжатия сетевого трафика в Windows, в UNIX используется системная библиотека.

### fbintl

Библиотека для поддержки интернационализации, кодировок и порядка сортировок. Располагается в директории intl относительно корневой директории сервера.

В Windows имеет имя fbintl.dll, в Unix — fbintl.

## Плагины

Модули плагинов размещаются в динамических библиотеках. В Firebird для модулей плагинов выделена папка plugins, которая находится в корневой директории сервера.

### Engine12

Поставщик (ядро) для работы с ODS 12.

В Windows имеет имя engine12.dll, в Unix — libEngine12.so, в MacOS — libEngine12.dylib.

### fbtrace

Плагин трассировки.

В Windows имеет имя fbtrace.dll, в Unix — libfbtrace.so, в MacOS — libfbtrace.dylib.

### legacy\_auth

Плагин для поддержки традиционной аутентификации (использовалась в Firebird 2.5 и в более ранних версиях).

В Windows имеет имя legacy\_auth.dll, в Unix — libLegacy\_Auth.so, в MacOS — libLegacy\_Auth.dylib.

### legacy\_usermanager

Менеджер пользователей, который используется в традиционной аутентификации.

В Windows имеет имя legacy\_usermanager.dll, в Unix — libLegacy\_UserManager.so, в MacOS — libLegacy\_UserManager.dylib.

### srp

Плагин аутентификации методом SRP — Secure remote Password.

В Windows имеет имя srp.dll, в Unix — libSrp.so, в MacOS — libSrp.dylib.

### udr\_engine

Внешний движок для подключения UDR написанных на компилируемых языках C, C++, Pascal.

В Windows имеет имя udr\_engine.dll, в Unix — libudr\_engine.so, в MacOS — libudr\_engine.dylib.

## Включаемые файлы

В директории include расположены заголовочные C файлы с функциями, константами, структурами и интерфейсами API Firebird.

## Примеры

В директории examples расположены примеры работы с Firebird API, написания плагинов, UDF и UDR.

## Документации

В директории doc расположены файлы документации.

## Другие файлы

### security3.fdb

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

### firebird.log

Лог-файл сервера.

### firebird.msg

Файл с сообщениями сервера (в основном об ошибках)

## Глава 3. Настройка операционной системы для Firebird

## Настройка Linux

### Лимиты Linux

### Необходимые права для Firebird на Linux

### Рекомендуемые параметры сети

### Рекомендуемые параметры памяти

### Рекомендуемые параметры диска

## Настройка Windows

### Необходимые права для Firebird на Linux

### Рекомендуемые параметры

## Глава 4. ISQL. Интерактивные запросы

В этой главе содержится описание консольной утилиты Firebird Interactive SQL Utility (isql) и инструкции по его использованию для выполнения задач в базе данных.

Утилита ISQL расположена в корневом каталоге установки Firebird. Она предоставляет интерфейс командной строки для интерактивного доступа к базе данных Firebird. ISQL принимает операторы DSQL вместе с группой команд SET и SHOW для запросов и взаимодействия с базой данных Firebird. Некоторые команды SET могут быть включены в сценарии DDL для выполнения пакетных операций в isql. Она также обрабатывает команды DDL, DML и консоли.

Утилита isql может использоваться в трех режимах: в качестве интерактивного сеанса; прямо из командной строки; и как неинтерактивный сеанс, используя сценарий оболочки или пакетный файл. В каждом из режимов могут выполняться разные задачи, как показано ниже:

* 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 её работы пользователь получает приглашение ввести следующую команду до тех пор, пока не будет введена команда выхода из интерактивного режима QUIT или EXIT. Isql можно использовать в интерактивном режиме для:
  + Создания, редактирования, удаления или просмотра данных и метаданных;
  + Ввода файла сценария содержащего пакет SQL запросов;
  + Добавлять и модифицировать данные;
  + Предоставлять разрешение пользователю и отбирать их;
  + Управлять учётными записями;
  + Выполнять административные функции базы данных.
* Непосредственно из командной строки, с отдельными параметрами и без запуска интерактивного сеанса. Команды выполняются и по завершении возвращают управление автоматически операционной системе.
* В неинтерактивном сеансе пользователь использует сценарий оболочки или пакетный файл для выполнения функций базы данных.

### Примечание

Поскольку другие приложения в среде Linux, например mySQL, также используют isql в качестве имени утилиты, рекомендуется запускать утилиту Firebird из ее собственного каталога или указывать абсолютный путь к файлу, если у вас есть другая реляционная база данных, кроме Firebird, установленная на вашей машине.

### Примечание

Некоторые из затронутых дистрибутивов, такие как Mint Linux — на основе Ubuntu — переименовали Firebird в isql в isql-fb. Аналогичные изменения могут быть и в других дистрибутивах.

Кроме того, не все дистрибутивы устанавливают Firebird в одно и то же место. Например, OpenSuse устанавливает практически все в /opt/firebird/3.0, но Ubuntu и его производные устанавливают его в различные места, но утилиты находятся в /usr/bin.

Далее предполагается что утилита называется isql.

## Запуск isql

Перейдите в корневой каталог установки сервера Firebird и введите в командной строке isql (Windows) или ./isql (Linux). Если каталог с isql добавлен в PATH, то нет необходимости переходить в него.

Синтаксис запуска isql следующий:

isql [<options>] [<database>]   
   
<database> ::= [<server spec>] {<db\_filename> | <db\_alias>}   
   
<server spec> ::=   
 <host>[\{<port> | <service>}]:   
 | \\<host>[@{<port> | <service>}]\   
 | <protocol>://[<host>[:{<port> | <service>}]/]   
   
<protocol> ::= inet | inet4 | inet6 | wnet | xnet

Если параметры *<options>* не указаны, isql запускает интерактивный сеанс. Если база данных не указана, то необходимо подключиться к существующей базе данных или создать новую. Если база данных была указана, isql запускает интерактивный сеанс, подключаясь к указанной базе данных.

Если указаны параметры, isql запускается в интерактивном или не интерактивном режиме, в зависимости от параметров. Например, чтение входного файла и запись в выходной файл являются не интерактивными задачами, поэтому параметры -input или -output не запускают интерактивный сеанс. Дополнительные не интерактивные параметры включают -a, -database, -extract и -x, которые используются при извлечении операторов DDL.

Когда вы запускаете интерактивный сеанс isql, появляется следующее приглашение:

SQL>

Вы должны завершить каждую команду символом-терминатором. Терминатором по умолчанию является точка с запятой (;). Вы можете изменить терминатор на любой символ или группу символов с помощью команды SET TERMINATOR или с помощью параметра командной строки -terminator. Если вы опустите терминатор, появится приглашение для продолжения ввода (CON>).

Ниже приведен самый простой способ запуска isql. После перехода в интерактивный режим вы должны начать с создания новой базы данных или подключения к ней. Сообщение, выданное isql, является подсказкой относительно того, что вы должны делать дальше.

c:\Firebird\3.0>isql  
  
Use CONNECT or CREATE DATABASE to specify a database  
SQL> CONNECT "inet://localhost:3053/employee"  
CON> user "SYSDBA" password 'masterkey';  
Database: "inet://localhost:3053/employee", User: SYSDBA  
SQL>

В следующем примере показано, как передать имя базы данных и учетные данные пользователя в командной строке.

c:\Firebird\3.0>isql -user sysdba -password masterkey "inet://localhost:3053/employee"  
  
Database: inet://localhost:3053/employee, User: SYSDBA  
SQL>

В этом примере мы использовали псевдоним для базы данных employee. Этот пример предопределен в файле databases.conf, который обычно находится в каталоге, в котором установлен Firebird, но некоторые дистрибутивы Linux помещают его в /etc/firebird/n.n, где n.n - версия сервера базы данных Firebird. Ниже приведена дополнительная информация о подключении к базам данных с использованием полных путей или псевдонимов.

## Переключатели командной строки

Переключатели командной строки — это аргументы, которые начинаются с символа минус/дефис (-). Ниже приведен пример того, что происходит, когда вы пытаетесь запустить isql с недопустимым переключателем — он отображает список допустимых ключей с кратким описанием каждого из них.

c:\Firebird\3.0>isql -help  
Unknown switch: -help  
usage: isql [options] [<database>]  
 -a(ll) extract metadata incl. legacy non-SQL tables  
 -b(ail) bail on errors (set bail on)  
 -c(ache) <num> number of cache buffers  
 -ch(arset) <charset> connection charset (set names)  
 -d(atabase) <database> database name to put in script creation  
 -e(cho) echo commands (set echo on)  
 -ex(tract) extract metadata  
 -f(etch\_password) fetch password from file  
 -i(nput) <file> input file (set input)  
 -m(erge) merge standard error  
 -m2 merge diagnostic  
 -n(oautocommit) no autocommit DDL (set autoddl off)  
 -nod(btriggers) do not run database triggers  
 -now(arnings) do not show warnings  
 -o(utput) <file> output file (set output)  
 -pag(elength) <size> page length  
 -p(assword) <password> connection password  
 -q(uiet) do not show the message "Use CONNECT..."  
 -r(ole) <role> role name  
 -r2 <role> role (uses quoted identifier)  
 -s(qldialect) <dialect> SQL dialect (set sql dialect)  
 -t(erminator) <term> command terminator (set term)  
 -tr(usted) use trusted authentication  
 -u(ser) <user> user name  
 -x extract metadata  
 -z show program and server version

### Примечание

Требуются только начальные символы в опции. Вы можете ввести любую часть текста, заключенную в скобки, включая полное имя опции. Например, указание -n, -no или -noauto имеет тот же эффект.

Таблица 4.1. Переключатели командной строки

|  |  |
| --- | --- |
| Опции | Описание |
| -a(ll) | Извлечение всех метаданных, включая не-SQL объекты. Используется совместно с командой extract |
| -b(ail) | Этот переключатель указывает утилите передать ошибку в ОС, но только в пакетном режиме (SET BAIL ON). Он был добавлен, чтобы предотвратить выполнение сценариев isql после обнаружения ошибки. Дальнейшие операторы не будут выполняться, и isql вернет код ошибки в ОС.  Пользователям по-прежнему необходимо использовать переключатель — e(cho) для вывода команд в выходной файл, чтобы выделить точный оператор, вызвавший ошибку.  Когда сервер предоставляет информацию о строках и столбцах, пользователи могут найти точную строку в DML скрипте, вызвавшую проблему. Когда сервер указывает только ошибку, то пользователи могут увидеть только первую строку оператора, вызвавшего сбой по отношению ко сему сценарию.  Эта функция также поддерживается во вложенных скриптах. Например, сценарий A включает в себя сценарий B, а сценарий B вызывает ошибку, номер строки связан со сценарием B. Когда сценарий B полностью прочитан, isql продолжает считать строки, относящиеся к сценарию A, поскольку каждый файл получает отдельный счетчик строк. Сценарий A включает сценарий B, когда сценарий A использует команду INPUT для загрузки сценария B. |
| -c(ache) *<num>* | Задаёт количество страниц, которые будут кэшированы при соединении с БД. |
| -ch(arset) *<charset>* | Задаёт кодировку для текущего соединения. Это же можно сделать с использовании команды SET NAMES |
| -d(atabase) *<database>* | Используется совместно с переключателем -x или -ex. Изменяет оператор CREATE DATABASE, который извлекается в файл:   * Без переключателя -d оператор CREATE DATABASE извлекается в скрипт как комментарий в стиле C, и использует имя базы данных указанное в командной строке isql * С переключателем -d оператор CREATE DATABASE извлекается не закомментированным, и подставляет имя базы данных указанное в переключателе |
| -e(cho) | Включает дублирование команд на указанное устройство вывода (монитор, в файл, и т. д.) Это можно сделать и с использованием команды SET ECHO ON |
| -ex(tract) | Извлечение метаданных из базы данных. Его можно использовать вместе с ключом -o(utput) для извлечения информации в указанный выходной файл. |
| -f(etch\_password) | Извлечение пароля из файла |
| -i(nput) *<file>* | Задаёт файл с SQL-скриптом для выполнения. Файлы скриптов могут содержать команду SET INPUT для включения других файлов. |
| -m(erge\_stderr) | Перенаправление ошибок (stderr) на поток стандартного вывода. Ключ командной строки -m(erge) может использоваться для включения сообщений об ошибках в выходные файлы. |
| -m2 | Отправлять информацию о статистике и планах в выходной файл, который указан в переключателе -o(utput). Ключ командной строки -m2 гарантирует, что статистика и информация о плане, полученные из команд SET STATS, SET PLAN и SET PLANONLY, также отправляются в выходной файл, а не просто возвращаются в консоль. |
| -n(oautocommit) | Отключить автоматическое подтверждение DDL-операций. По умолчанию операторы DDL фиксируются автоматически в отдельной транзакции. Может быть изменено командой SET AUTODDL OFF |
| -nod(btriggers) | Не запускать триггеры базы данных. |
| -now(arnings) | Не показывать предупреждения. |
| -o(utput) *<file>* | Задаёт файл для вывода результата выполнения запросов. Без аргументов перенаправляет вывод на стандартное устройство вывода (монитор). Может быть изменён так же командой SET OUTPUT |
| -pag(elength) *<size>* | Размер страницы. Печатает заголовки столбцов каждые *<size>* строк вместо значений по умолчанию 20. |
| -p(assword) *<password>* | Пароль пользователя при подключении к базе данных. |
| -q(uiet) | Не показывать сообщение "Use CONNECT...". |
| -r(ole) *<role>* | Имя роли при подключении к базе данных. |
| -r2 *<role>* | Имя роли (с использованием квотированных идентификаторов). Этот переключатель можно использовать для указания имени роли с учетом регистра.  По умолчанию для передачи роли используется переключатель -r(ole). В этом случае роль, представленная в командной строке, пишется в верхнем регистре. Если используется переключатель -r2, то роль передаются в движок в точности так, как это указано в командной строке. |
| -s(qldialect) *<dialect>* | Устанавливает SQL диалект к котором интерпретируются последующие команды до конца сеанса или до тех пор, пока диалект не будет изменён с помощью инструкции SET SQL DIALECT. |
| -t(erminator) *<term>* | Изменяет символ конца оператора (терминатора) с точки с запятой «;» по умолчанию на *<term>*, где *<term>* — любой символ или последовательность символов. Тоже самое можно сделать использую команду SET TERM. |
| -tr(usted) | Использовать доверительную аутентификацию. |
| -u(ser) *<user>* | Имя пользователя для подключения к базе данных. |
| -x | Извлечение метаданных из базы данных. Его можно использовать вместе с ключом -o(utput) для извлечения информации в указанный выходной файл. |
| -z | Показать версии утилиты и сервера. |

### Примеры

Пример 4.1. Выполнение скрипта создания базы данных

isql -input createdb.sql

Пример 4.2. Запуск интерактивного сеанса с подключением к удалённой базе данных

В следующем примере начинается интерактивное соединение с удаленной базой данных. Удаленный сервер jupiter, принимает указанную комбинацию пользователя и пароля с привилегиями, назначенными роли STAFF:

isql -user sales -password mycode -role staff 'jupiter:/usr/customer.fdb'

Пример 4.3. Запуск интерактивного сеанса без подключения к базе данных

Следующий пример запускает интерактивный сеанс, но не подключается к базе данных. Все введенные команды isql выводятся на печать, а результаты запроса печатают заголовки столбцов каждые 30 строк:

isql -echo -page 30

## Начало сессии Isql

Чтобы начать сеанс isql, введите параметры командной строки и имя базы данных в оболочке Linux/Unix или командной консоли Windows. Например:

isql [<options>] [<database>]

### Примечание

При подключении к удаленной базе данных вам нужно будет включить соответствующие параметры -user и -password в свои параметры, если только пользователи не объявили ISC\_USER и ISC\_PASSWORD в качестве переменных операционной системы. Например:

isql -user SYSDBA -password masterkey inet4://localhost/mydatabase

При подключении к локальной базе данных в embedded варианте достаточно указать только имя пользователя (пароль будет проигнорирован):

isql -user SYSDBA employee.db

В Windows вы можете использовать доверительную аутентификацию, при условии что она настроена в firebird.conf или databases.conf:

isql -trusted inet4://localhost/mydatabase

Если параметры *<options>* не указаны, isql запускает интерактивный сеанс. Если база данных не указана, то необходимо подключиться к существующей базе данных или создать новую. Если база данных была указана, isql запускает интерактивный сеанс, подключаясь к указанной базе данных.

Чтение входного файла и запись в выходной файл не считаются интерактивными задачами, поэтому параметры командной строки -input или -output не инициируют интерактивный сеанс. Опции, используемые для извлечения операторов DDL, такие как -a и -x, также инициируют только неинтерактивный сеанс.

Isql может быть запущен с локального или удаленного клиента:

* При подключении с использованием локального клиента вы можете установить переменные среды ISC\_USER и ISC\_PASSWORD.
* При подключении с удаленного клиента вам потребуется указать действительные имя и пароль.

## Завершение сессии Isql

Существует два способа выхода из isql:

* Если вы хотите откатить всю неподтвержденную работу и выйти из isql, введите в командной строке следующую команду:
* SQL> QUIT;
* Если вы хотите зафиксировать всю свою работу перед выходом из isql, введите следующую команду:
* SQL> EXIT;

## Получение справки

Isql поставляется с командой HELP. Это дает краткую информацию о большинстве доступных команд. Команда HELP также позволяет перейти к более подробной информации. Чтобы активировать справочную систему, просто введите HELP в командной строке, как показано ниже:

SQL> HELP;

Frontend commands:  
BLOBDUMP <blobid> <file> -- dump BLOB to a file  
BLOBVIEW <blobid> -- view BLOB in text editor  
EDIT [<filename>] -- edit SQL script file and execute  
EDIT -- edit current command buffer and execute  
HELP -- display this menu  
INput <filename> -- take input from the named SQL file  
OUTput [<filename>] -- write output to named file  
OUTput -- return output to stdout  
SET <option> -- (Use HELP SET for complete list)  
SHELL <command> -- execute Operating System command in sub-shell  
SHOW <object> [<name>] -- display system information  
 <object> = CHECK, COLLATION, DATABASE, DOMAIN, EXCEPTION, FILTER, FUNCTION,  
 GENERATOR, GRANT, INDEX, PACKAGE, PROCEDURE, ROLE, SQL DIALECT,  
 SYSTEM, TABLE, TRIGGER, VERSION, USERS, VIEW  
EXIT -- exit and commit changes  
QUIT -- exit and roll back changes  
  
All commands may be abbreviated to letters in CAPitals

Большинство из этих команд не имеют дальнейших уровней детализации, в то время как команда SET имеет. Чтобы углубиться в эти дополнительные уровни, выполните следующие действия:

SQL> HELP SET;

Set commands:  
 SET -- display current SET options  
 SET AUTOddl -- toggle autocommit of DDL statements  
 SET BAIL -- toggle bailing out on errors in non-interactive mode  
 SET BLOB [ALL|<n>] -- display BLOBS of subtype <n> or ALL  
 SET BLOB -- turn off BLOB display  
 SET COUNT -- toggle count of selected rows on/off  
 SET MAXROWS [<n>] -- limit select stmt to <n> rows, zero is no limit  
 SET ECHO -- toggle command echo on/off  
 SET EXPLAIN -- toggle display of query access plan in the explained form  
 SET HEADING -- toggle display of query column titles  
 SET LIST -- toggle column or table display format  
 SET NAMES <csname> -- set name of runtime character set  
 SET PLAN -- toggle display of query access plan  
 SET PLANONLY -- toggle display of query plan without executing  
 SET SQL DIALECT <n> -- set sql dialect to <n>  
 SET STATs -- toggle display of performance statistics  
 SET TIME -- toggle display of timestamp with DATE values  
 SET TERM <string> -- change statement terminator string  
 SET WIDTH <col> [<n>] -- set/unset print width to <n> for column <col>  
  
All commands may be abbreviated to letters in CAPitals

Если вы попытаетесь углубиться в какую-либо другую команду, эффект будет точно таким же, как при выполнении только команды HELP.

### Примечание

Команда HELP SET даёт справку только по собственным командам isql. Для управляющих SQL операторов, таких как SET TRANSACTION или SET ROLE подсказка не будет выведена.

## Соединение с базой данных

Если при запуске isql вы не указали спецификацию для подключения к базе данных, то будет выдана подсказка о том что необходимо соединиться с базой данных с помощью оператора CONNECT или создать новую базу данных с помощью оператора CREATE DATABASE.

Оператор CONNECT не является частью DSQL и доступен только внутри ISQL, а потому не описан в «Руководство по языку SQL Firebird». Рассмотрим его синтаксис.

Синтаксис:

CONNECT <database>   
[USER <username> [PASSWORD <password>]]  
[CACHE <cache size>]   
[ROLE <rolename>]   
   
<database> ::= [<server spec>] {<db\_filename> | <db\_alias>}   
   
<server spec> ::=   
 <host>[\{<port> | <service>}]:   
 | \\<host>[@{<port> | <service>}]\   
 | <protocol>://[<host>[:{<port> | <service>}]/]   
   
<protocol> ::= inet | inet4 | inet6 | wnet | xnet

Таблица 4.2. Параметры оператора CONNECT

|  |  |
| --- | --- |
| Параметр | Описание |
| *database* | Спецификация базы данных. |
| *username* | Имя пользователя для соединения с базой данных. Может быть заключено в одинарные или двойные кавычки. Если имя пользователя заключено в двойные кавычки, то оно чувствительно к регистру. |
| *password* | Пароль пользователя для соединения с базой данных. Чувствительно к регистру. Должен быть заключён в одинарный кавычки. |
| *cache size* | Размер страничного кеша. |
| *rolename* | Имя роли для соединения с базой данных. Может быть заключено в одинарные или двойные кавычки. Если имя роли заключено в двойные кавычки, то оно чувствительно к регистру. |
| *server spec* | Спецификация удалённого сервера. Включает в себя имя сервера и протокол. |
| *db\_filename* | Полный путь и имя файла базы данных, включая расширение. Имя файла должно быть задано в соответствии со спецификой используемой платформы. |
| *db\_alias* | Псевдоним (alias) базы данных, присутствующий в файле databases.conf |
| *host* | Имя сервера или IP адрес, на котором расположена база данных. |
| *port* | Номер порта, который слушает удалённый сервер (параметр *RemoteServicePort* файла firebird.conf). |
| *service* | Имя сервиса. Должно совпадать со значением параметра *RemoteServiceName* файла firebird.conf |
| *protocol* | Наименование протокола. |

База данных может состоять из одного или нескольких файлов. Первый, основной, файл называется первичным, остальные файлы — вторичными. При подключении к базе данных необходимо указывать только путь до первичного файла базы данных.

Спецификация первичного файла — имя файла базы данных и его расширение с указанием к нему полного пути в соответствии с правилами используемой операционной системы.

### Использование псевдонимов БД

Вместо полного пути к первичному файлу базы можно использовать псевдонимы (aliases). Псевдонимы описываются в файле databases.conf в формате:

alias = filepath

### Примечание

Помимо указания псевдонимов для базы данных в этом файле можно задать параметры уровня базы данных для каждой описываемой базы данных. Эти параметры задаются в фигурных скобках сразу после объявления псевдонима. Подробнее читайте в [Параметры уровня базы данных](#config-perdb)

### Соединение с БД на удалённом сервере

При соединении с базой данных на удалённом сервере необходимо указать спецификацию удалённого сервера. Спецификация удалённого сервера зависит от используемого протокола.

Если вы при соединении с базой данных используете протокол TCP/IP, то спецификация первичного файла должна выглядеть следующим образом:

host[/port | service]:{filepath | db\_alias}

Если вы при соединении с базой данных используете протокол под названием именованные каналы (Name Pipes), то спецификация первичного файла должна выглядеть следующим образом.

\\host[@port | service]\{filepath | db\_alias}

Существует также унифицированный URL-подобный синтаксис спецификации удалённого сервера. В этом синтаксисе первым параметром указывается наименование протокола, далее указывается имя сервера или IP адрес, номер порта и путь к первичному файлу базы данных или псевдоним. В качестве протокола можно указать следующие значения:

* INET — TCP/IP (сначала пробует подключится по протоколу TCP/IP v6, если не получилось, то TCP/IP v4);
* INET4 — TCP/IP v4 (начиная с 3.0.1);
* INET6 — TCP/IP v6 (начиная с 3.0.1);
* WNET — NetBEUI или протокол именованных каналов;
* XNET — локальный протокол.

protocol://[host[:port | service]/]{filepath | db\_alias}

### Имя пользователя, пароль и роль

Необязательные предложения USER и PASSWORD задают, соответственно, имя и пароль пользователя присутствующего в базе данных безопасности (security3.fdb или той, что указана в параметре *SecurityDatabase*). Пользователя и пароль можно не указывать, если установлены переменные окружения ISC\_USER и ISC\_PASSWORD. Их также можно не указывать при использовании доверительной аутентификации.

Если подключение происходит к embedded версии сервера, то достаточно указать только имя пользователя (проверка пароля не производится).

Необязательное предложение ROLE задаёт имя роли, права которой будут учитываться при подключении базы данных. Роль должна быть назначена пользователю в соответствующей базе данных безопасности.

### Набор символов

Команда CONNECT не позволяет указать набор символов для подключения, однако вы можете это сделать с помощью команды SET NAMES, которую необходимо ввести до использования CONNECT.

### Примеры

Пример 4.4. Подключение к локальной (embedded) базе данных

В следующем примере происходит подключение к базе данных в embedded варианте с использованием имени пользователя из переменной среды ISC\_USER.

SQL> CONNECT "C:\DATABASES\FIREBIRD\MY\_EMPLOYEE.FDB";

В Linux это будет выглядеть следующим образом:

SQL> CONNECT "/databases/firebird/MY\_EMPLOYEE.FDB";

При использовании псевдонимов, настроенных в databases.conf:

SQL> CONNECT MY\_EMPLOYEE;

Если переменная окружения ISC\_USER не задана, или требуется указать другого пользователя, то необходимо использовать предложение USER:

SQL> CONNECT MY\_EMPLOYEE USER SYSDBA;

Пример 4.5. Подключение к локальной (не embedded) базе данных

На Windows можно использовать локальный протокол XNET

SQL> CONNECT xnet://MY\_EMPLOYEE USER SYSDBA PASSWORD 'masterkey';

Во всех операционных системах через TCP

SQL> CONNECT inet://localhost/MY\_EMPLOYEE USER SYSDBA PASSWORD 'masterkey';

Пример 4.6. Подключение к удалённой базе данных с ролью RDB$ADMIN

SQL> CONNECT inet://dbserver/MY\_EMPLOYEE USER BOB PASSWORD 'W45xs' ROLE RDB$ADMIN;

Пример 4.7. Подключение к удалённой базе данных с набором символов WIN1251

SQL> SET NAMES WIN1251;   
SQL> CONNECT inet://dbserver/MY\_EMPLOYEE USER BOB PASSWORD 'W45xs';

## Создание базы данных

Если при запуске isql вы не указали спецификацию для подключения к базе данных, то будет выдана подсказка о том что необходимо соединиться с базой данных с помощью оператора CONNECT или создать новую базу данных с помощью оператора CREATE DATABASE.

Оператор CREATE DATABASE является частью DSQL и подробно описан в «Руководство по языку SQL Firebird», поэтому здесь будет приведён только краткий синтаксис и примеры использования.

Синтаксис:

CREATE {DATABASE | SCHEMA} '<filespec>'  
 [USER username [PASSWORD 'password'] [ROLE rolename]]  
 [PAGE\_SIZE [=] size]  
 [LENGTH [=] num [PAGE[S]]  
 [SET NAMES 'charset']  
 [DEFAULT CHARACTER SET default\_charset  
 [COLLATION collation]]  
 [<sec\_file> [<sec\_file> ...]]  
 [DIFFERENCE FILE 'diff\_file'];  
  
<filespec> ::= [<server\_spec>]{filepath | db\_alias}  
  
<server\_spec> ::=   
 host[\port | service]:   
 | \\host[@port | service]\   
 | <protocol>://[host[:port | service]/]  
  
<protocol> = inet | inet4 | inet6 | wnet | xnet  
  
<sec\_file> ::= FILE 'filepath'  
 [LENGTH [=] num [PAGE[S]] [STARTING [AT [PAGE]] pagenum]

В следующем примере будет создана база данных monkey.fdb и сохранена в директории test на диске С.

SQL>CREATE DATABASE 'C:\test\monkey.fdb' page\_size 8192  
CON>user 'SYSDBA' password 'masterkey';

### Примечание

Путь к базе данных и пароль обязательно должны быть заключены в одинарные или двойные кавычки. Для имени пользователя, пароля и набора символов кавычки необязательны. При использовании двойных кавычек буду использованы правила 3-диалекта, т.е. имя пользователя, роли и набора символов будут чувствительны к регистру символов.

Поскольку база данных создаётся в режиме embedded сервера, то файл базы данных будет создан с именем входа в систему в качестве владельца. Это может вызвать проблемы с доступом к файлу базы данных для других пользователей, особенно в Linux, которые могут захотеть подключится позднее. Если база данных создаётся с использованием сетевого или локального (Windows) протокола, то файл базы данных будет создан от имени пользователя службы (в Linux это пользователь firebird).

В данном случае база данных будет создана с использованием embedded версии Firebird. Для использования сетевых протоколов вы можете использовать префиксы протоколов или другие правила аналогичные правила для оператора CONNECT.

Пример 4.8. Создание базы данных на удалённом сервере

Создание базы данных на удалённом сервере baseserver расположенном по пути, на который ссылается псевдоним test, описанный в файле databases.conf. Используется протокол TCP. Владельцем базы данных будет пользователь wizard.

CREATE DATABASE 'baseserver:test'  
USER wizard PASSWORD 'player' ROLE RDB$ADMIN  
DEFAULT CHARACTER SET UTF8;

То же самое с использованием унифицированного URL-подобного синтаксиса задания спецификации удалённого сервера.

CREATE DATABASE 'inet://baseserver:3050/test'  
USER wizard PASSWORD 'player' ROLE RDB$ADMIN  
DEFAULT CHARACTER SET UTF8;

или

CREATE DATABASE 'inet://baseserver:gds\_db/test'  
USER wizard PASSWORD 'player' ROLE RDB$ADMIN  
DEFAULT CHARACTER SET UTF8;

Начиная с Firebird 3.0.1 можно указать какой именно из протоколов IP v4 или IP v6 вы хотите использовать.

CREATE DATABASE 'inet4://baseserver/test'  
USER wizard PASSWORD 'player' ROLE RDB$ADMIN  
DEFAULT CHARACTER SET UTF8;

или

CREATE DATABASE 'inet6://baseserver/test'  
USER wizard PASSWORD 'player' ROLE RDB$ADMIN  
DEFAULT CHARACTER SET UTF8;

Создание базы данных с указанием IP адреса (IPv4) вместо указания имени сервера.

CREATE DATABASE '127:0:0:1:test'  
USER wizard PASSWORD 'player' ROLE RDB$ADMIN  
DEFAULT CHARACTER SET UTF8;

Создание базы данных с указанием IP адреса (IPv6) вместо указания имени сервера.

CREATE DATABASE '[::1]:test'  
USER wizard PASSWORD 'player' ROLE RDB$ADMIN  
DEFAULT CHARACTER SET UTF8;

### Примечание

В отличие от оператора CONNECT в операторе CREATE DATABASE вы сразу можете указать набор символов который будет использован в подключении после выполнения оператора CREATE DATABASE с помощью предложения SET NAMES.

После создания базы данных вы можете выполнять к ней запросы в интерактивном режиме.

SQL>SELECT RDB$RELATION\_ID FROM RDB$DATABASE;  
  
RDB$RELATION\_ID  
===============  
128  
  
SQL> commit;

Чтобы вернуться в командную строку введите команду exit или quit.

### Примечание

Вышеуказанная методика, как показано, работает, но в идеале базы данных и объекты метаданных должны создаваться и поддерживаться с использованием сценариев определения данных.

## Установка переменных окружения ISC\_USER и ISC\_PASSWORD

Переменная среды — это именованный объект, который содержит информацию, используемую одним или несколькими приложениями. Они глобальны для своих конкретных операционных систем. Сервер Firebird распознает и использует определенные переменные среды, настроенные в Windows, Linux и других системах Unix.

Переменные среды ISC\_USER и ISC\_PASSWORD в Firebird предназначены для предоставления SYSDBA доступа к базе данных из служебных программ командной строки и клиентских приложений всем, кто имеет доступ к хост-машине.

### Внимание

При запуске таких утилит командной строки, как isql, gbak, gstat и gfix, Firebird будет искать, установлены ли переменные среды ISC\_USER и ISC\_PASSWORD. Если вы не предоставите имя пользователя и пароль при локальном подключении к базе данных, Firebird позволит вам войти в систему, если найдет эти переменные.

Из соображений безопасности не рекомендуется указывать имя пользователя и пароль SYSDBA с использованием этих двух переменных среды, особенно на незащищенном компьютере.

Переменные среды ISC\_USER и ISC\_PASSWORD могут быть установлены для локального запуска isql. Чтобы установить переменные среды:

* В Windows это делается в Панели управления -> Система -> Дополнительно -> Переменные среды. Любые изменения, сделанные здесь, будут постоянными. Вы также можете определить эти переменные в командном окне до запуска любой из утилит Firebird, например isql. Например:
* C:\> set ISC\_USER=sysdba  
  C:\> set ISC\_PASSWORD=secret  
  C:\> isql my\_employee  
    
  SQL>
* На платформах Linux и Unix это зависит от типа используемой оболочки и от конфигурации настольного компьютера. Обратитесь к документации вашей операционной системы, чтобы установить переменные среды. Для оболочки bash этот процесс показан в следующем примере:
* cosmos> export ISC\_USER=sysdba  
  cosmos> export ISC\_PASSWORD=secret  
  cosmos> /opt/firebird/bin/isql my\_employee  
    
  SQL>

## SQL диалект

Firebird поддерживает три диалекта SQL на каждом клиенте и сервере базы данных. Эти диалекты SQL различаются в контексте формата даты и времени и точности числового типа данных. Диалекты служат для инструктирования сервера Firebird о том, как обрабатывать функции, реализованные в устаревших базах данных Borland Interbase, более ранних, чем версия 6.0. Диалекты устанавливаются во время выполнения и могут быть изменены для клиента во время подключения или с помощью команды SET SQL DIALECT.

### Примечание

Диалект 2 используется только при преобразовании базы данных диалекта 1 в базу данных диалекта 3.

Следующая таблица демонстрирует некоторые различия между диалектами.

Таблица 4.3. Различия между SQL диалектами

|  |  |  |  |
| --- | --- | --- | --- |
| SQL | Dialect 1 | Dialect 2 | Dialect 3 |
| Тип данных DATE | Дата + время (Timestamp) | Сообщение об ошибке | Только дата |
| Тип данных TIMSTAMP | Timestamp | Timestamp | Timestamp |
| Тип данных TIME | Сообщение об ошибке | Сообщение об ошибке | Только время |
| Идентификаторы с разделителями (квотированные идентификаторы) "*<identifier>*" | Строка | Сообщение об ошибки | Идентификатор |
| Точность арифметических операций. Например 1/3 | 0.3333333... (double precision) | 0 | 0 (по правилам целочисленного деления) |
| Представление NUMERIC и DECIMAL с точностью больше 10 | double precision | 64-битное целое | 64-битное целое (BIGINT) |

### Примечание

В настоящее время возможно создание баз данных только на диалекте 1 и 3, однако рекомендуется использовать исключительно диалект 3, поскольку в конечном итоге диалект 1 будет устаревшим. Диалект 2 нельзя использовать для создания базы данных, поскольку он служит только для преобразования Диалекта 1 в Диалект 3.

При подключении к базе данных с использованием isql утилита использует диалект базы данных, если не указано иное. Диалекты не могут быть установлены в качестве параметра оператора CREATE DATABASE. Таким образом, при создании базы данных с использованием isql база данных будет на диалекте, который является текущим в isql во время выполнения оператора CREATE DATABASE. Вы можете установить диалект с помощью утилиты isql двумя способами:

* При старте isql укажите диалект с использованием переключателя -sql\_dialect
* cosmos> isql -sql\_dialect 3
* Внутри sql скрипта или в isql сессии с помощью команды SET SQL DIALECT
* SQL> SET SQL DIALECT 3;

## Символ терминатора

Символом терминатором по умолчанию для базы данных Firebird является точка с запятой «;». Операторы будут только выполнены если они заканчиваются точкой с запятой. Однако вы можете использовать isql для изменения символа на любой печатный символ или несколько символов из первых 127 символов подмножества ASCII с помощью команды SET TERM.

### Примечание

Терминатор по умолчанию может быть изменен во всех случаях, за исключением Procedural SQL или PSQL. PSQL не принимает никаких терминаторов, кроме точки с запятой.

Для изменения символа терминатора на тильду «~» введите следующую команду:

SQL> SET TERM ~ ;

Для возврата терминатора, вы должны использовать текущий терминатор! Изменение терминатора полезно, если вы хотите ввести функцию PSQL, как показано в следующем примере. Поскольку PSQL будет принимать только точку с запятой в качестве терминатора, isql должен знать, какая точка с запятой используется для кода PSQL, а какая используется для завершения вводимых команд SQL.

SQL> set term ~ ;  
  
SQL> create procedure test\_proc (iInput integer = 666)  
CON> returns (oOutput integer)  
CON> as  
CON> begin  
CON> oOutput = iInput;  
CON> suspend;  
CON> end~  
  
SQL> set term ; ~  
  
SQL> commit;  
  
SQL> select \* from test\_proc;  
  
 OOUTPUT  
============  
 666

Вы можете видеть, что внутри кода самой процедуры терминатором является точка с запятой. Однако вне фактического кода процедуры терминатором является тильда (~). isql обрабатывает одну команду CREATE PROCEDURE, но внутри этого одного оператора SQL есть несколько встроенных операторов PSQL:

oOutput = iInput;  
suspend;

Они имеют точку с запятой, как того требует PSQL. Конец команды CREATE PROCEDURE обозначается использованием тильды в качестве терминатора:

end~

При желании вы можете просто изменить терминатор, потому что вы предпочитаете что-то другое, а не точку с запятой.

SQL> -- Change terminator from ; to +  
SQL> set term + ;  
  
SQL> select count(\*) from employee+  
  
 COUNT  
============  
 42  
  
SQL> -- Change terminator from + to 'fred'  
SQL> set term fred +  
  
SQL> select count(\*) from employee fred  
  
 COUNT  
============  
 42  
  
SQL> -- Change back from 'fred' to ;  
SQL> set term ; fred

Однако вы должны быть осторожны, чтобы не выбрать символ-терминатор, который приведет к сбою операторов SQL из-за того, что терминатор используется в какой-то момент в операторе SQL.

SQL> select 600+60+6 as The\_Beast from rdb$database;  
  
THE\_BEAST  
=====================  
666  
  
SQL> set term + ;  
SQL> select 600+60+6 as The\_Beast from rdb$database+  
  
Statement failed, SQLSTATE = 42000  
Dynamic SQL Error  
-SQL error code = -104  
-Unexpected end of command - line 1, column 8  
...  
  
SQL> set term ; +

Присутствие терминатора в выражении вызвало ошибку «Unexpected end of command». Анализатор SQL в ядре базы данных Firebird определил, что select 600 не является допустимым оператором. По этой причине лучше всегда выбирать символ или символы, которые не будут путать синтаксический анализатор.

SQL> set term !! ;  
  
SQL> select 600+60+6 as The\_Beast from rdb$database!!  
  
THE\_BEAST  
=====================  
666

## Подсказки ISQL

Подсказка SQL. Как показано выше, обычная подсказка isql для ввода — это подсказка SQL>. Это указывает на то, что предыдущая команда была выполнена, и теперь isql ожидает обработки новой команды.

Подсказка CON. Подсказка CON> отображается, если пользователь нажимает клавишу ENTER без завершения оператора SQL с помощью терминатора. Например:

SQL> HELP  
CON>

Всякий раз, когда вы видите приглашение CON>, вы можете либо продолжить ввод оставшейся части команды, либо ввести терминатор для завершения команды. В последнем случае команда будет выполнена после нажатия клавиши ENTER.

## Обработка ошибок и поддержка исключений

Обработка исключений — это программная конструкция, предназначенная для обработки ситуаций, которые нарушает нормальное выполнение программы. Это так называемые ошибки. Исключением являются именованное сообщение об ошибке, определяемые пользователем, написанные специально для базы данных и сохраненные в этой базе данных для использования в хранимых процедурах, функциях, пакетах, анонимных блоках и триггерах.

Например, если в триггере обнаруживается, что значение для таблицы неверно, то возникает исключение. Это приводит к откату всей транзакции, которую клиентское приложение пытается зафиксировать. Исключения могут совместно использоваться различными модулями приложения и даже различными приложениями, совместно использующими базу данных. Они предоставляют простой способ стандартизировать обработку заранее запрограммированных ошибок ввода.

Исключения составляют объекты базы данных, такие как таблицы, представления и домены, и являются частью метаданных базы данных. Их можно создавать, изменять и удалять, как и все другие объекты Firebird, используя isql.

В isql сообщения об ошибках содержат переменную SQLSTATE и статус вектор Firebird. Расшифровку кодов ошибок SQLSTATE вы можете найти в "Руководстве по языку SQL СУБД Firebird".

Пример 4.9. Сообщение об ошибке в isql

SQL> select 1/0 from rdb$database;  
  
 DIVIDE  
=====================  
Statement failed, SQLSTATE = 22012  
arithmetic exception, numeric overflow, or string truncation  
-Integer divide by zero. The code attempted to divide an integer value by an integer divisor of zero.

## Обработка транзакций

В isql транзакция начинается сразу после запуска утилиты. Транзакция начинается в изоляции SNAPSHOT с разрешением блокировки в режиме WAIT. Поскольку утилита Firebird isql принимает DDL, DML и другие команды, транзакции обрабатываются соответствующим образом следующими способами:

* Операторы DDL фиксируются автоматически в двух случаях:
  + Когда операторы COMMIT включен в скрипт.
  + Когда включена автоматическая фиксация транзакций для DDL операторов (режим по умолчанию). В этом случае выполнение каждого DDL оператора транзакция автоматически подтверждается после ввода каждого оператора DDL, и стартует новая транзакция. Чтобы отключить режим автоматической фиксации, выполните инструкцию SET AUTODDL OFF. Для возврата в режим автоматической фиксации для DDL операторов выполните инструкцию SET AUTODDL ON.
* Операторы DML для выборки или изменения данных стартуют новую транзакцию с уровнем изоляции SNAPSHOT, если она не была начата ранее. Такая транзакция будет активной до тех пор, пока не будет вручную подтверждена оператором COMMIT или отменена оператором ROLLBACK.
* Вы можете использовать различные команды SHOW в isql для запроса метаданных базы данных. При этом ISQL запускает транзакцию с уровнем изоляции READ COMMITED, что дает возможность видеть все изменения метаданных, подтвержденные другими пользователями.

Пользователи могут указать уровень изоляции и другие параметры для следующей транзакции с помощью оператора SET TRANSACTION. Он может быть выполнен только тогда, когда не активна другая транзакция. В противном случае будет задан вопрос: подтвердить или откатить активную транзакцию. Синтаксис оператора SET TRANSACTION подробно описан в «Руководство по языку SQL Firebird».

Пример 4.10. Старт транзакции с режимом изолированности READ COMMITED

SET TRANSACTION WAIT READ COMMITTED LOCK TIMEOUT 10;

### Примечание

Поскольку оператор SET TRANSACTION является оператором DSQL, а не собственной командой isql, то информация о нём отсутствует в справке выдаваемой оператором HELP SET.

## Обработка скрипта

Пакет операторов DDL и/или DML в текстовом файле называется сценарием (скриптом). Скрипты могут быть использованы для создания и изменения объектов базы данных (метаданных). Они называются сценариями языка определения данных (DDL). Скрипты, которые манипулируют данными путем выбора, вставки, обновления, удаления или выполнения преобразований данных, называются скриптами языка манипулирования данными (DML).

Одной из наиболее важных задач, выполняемых isql, является обработка скриптов. Он может обрабатывать как сценарии DDL, так и сценарии DML. Большие файлы сценариев можно разбить на несколько файлов скриптов поменьше с помощью инструкции INPUT.

Синтаксис:

INPUT <filespec>

Данный оператор позволяет включить один файл сценария внутрь другого. Операторы сценариев выполняются в том порядке, в котором они появляются в файле сценариев. Настройка по умолчанию в isql для AUTODDL установлена на ON. Вы можете использовать команду SET AUTODDL, чтобы контролировать, где или когда будут подтверждены операторы изменения метаданных.

### Примечание

Параметр AUTODDL влияет только на операторы DDL. Он не фиксирует транзакцию для операторов DML. Если вы смешиваете инструкции DDL и DML в одном и том же интерактивном сеансе, то фиксации транзакций для DDL с помощью AUTODDL не фиксируют ваши транзакции для операторов DML. Например:

SQL> set autoddl on;  
  
SQL> insert into test(a) values (666);  
SQL> commit;  
  
SQL> select \* from test;  
  
A  
============  
666  
  
SQL> insert into test(a) values (999);  
SQL> select \* from test;  
  
A  
============  
666  
999  
  
SQL> create table another\_test(b integer);  
SQL> rollback;  
  
SQL> select \* from test;  
  
A  
============  
666

Скрипты могут перенаправить свои выходные данные в файл журнала с помощью команды OUTPUT <file\_name>. Это можно ввести непосредственно в приглашении isql или как часть самого файла скрипта.

## ISQL команды

Команды Isql влияют на работу самого isql и никак не влияют на базу данных или данные. Эти команды используются для отображения справки, чтения, записи и запуска сценариев, а также выполнение инструкций командной строки. Вы можете легко увидеть список доступных команд, набрав команду HELP, которая выдаст следующий вывод:

SQL> HELP;  
Frontend commands:  
BLOBDUMP <blobid> <file> -- dump BLOB to a file  
BLOBVIEW <blobid> -- view BLOB in text editor  
EDIT [<filename>] -- edit SQL script file and execute  
EDIT -- edit current command buffer and execute  
HELP -- display this menu  
INput <filename> -- take input from the named SQL file  
OUTput [<filename>] -- write output to named file  
OUTput -- return output to stdout  
SET <option> -- (Use HELP SET for complete list)  
SHELL <command> -- execute Operating System command in sub-shell  
SHOW <object> [<name>] -- display system information  
 <object> = CHECK, COLLATION, DATABASE, DOMAIN, EXCEPTION, FILTER, FUNCTION,  
 GENERATOR, GRANT, INDEX, PACKAGE, PROCEDURE, ROLE, SQL DIALECT,  
 SYSTEM, TABLE, TRIGGER, VERSION, USERS, VIEW  
EXIT -- exit and commit changes  
QUIT -- exit and roll back changes  
  
All commands may be abbreviated to letters in CAPitals

Теперь обсудим каждую из команд отдельно.

### Примечание

Обратите внимание на последнюю строку вывода команды help. Это объясняет, что каждая из команд может быть сокращена до букв, отображаемых заглавными буквами. В следующем обсуждении дополнительные символы будут отображаться, как указано выше, строчными буквами. Например, команда ввода будет показана как INput, чтобы указать, что символы 'put' являются необязательными.

### BLOBDUMP

Назначение: Сохранение данных BLOB в указанном файле.

Синтаксис:

BLOBDUMP <blob\_id> <filename>

Таблица 4.4. Параметры оператора BLOBDUMP

|  |  |
| --- | --- |
| Параметр | Описание |
| *blob\_id* | BLOB идентификатор. |
| *filename* | Имя файла в который необходимо сохранить BLOB. |

Эта команда позволяет вам скопировать BLOB из базы данных во внешний файл. Оператор BLOBDUMP требует два параметра, идентификатор BLOB-объекта и имя файла. Вам необходимо передать идентификатор BLOB-объекта в виде пары шестнадцатеричных чисел, разделенных двоеточием. Первое число — это идентификатор таблицы, а второе — последовательный номер в базе данных. Для получения этого идентификатора выполните любой оператор SELECT, который выбирает столбец BLOB. Вывод покажет шестнадцатеричный идентификатор BLOB выше или на месте столбца BLOB в зависимости от того, установлен ли SET BLOBDISPLAY в ON или OFF.

Пример 4.11. Сохранение BLOB во внешний файл

SQL> set blobdisplay off;  
  
SQL> select proj\_id, proj\_desc  
CON> from project  
CON> where proj\_id = 'MKTPR';  
  
PROJ\_ID PROJ\_DESC  
======= =================  
MKTPR 85:10  
  
SQL> blobdump 85:10 c:\project.jpg;

Идентификатор BLOB-объекта, необходимый в приведенном выше примере, равен 85:10.

Смотри также: [SET BLOBDISPLAY](#isql-commands_set-set_blob), [BLOBVIEW](#isql-commands-blobview).

### BLOBVIEW

Назначение: Отображение данных BLOB в текстовом редакторе по умолчанию.

Синтаксис:

BLOBVIEW <blob\_id>

Таблица 4.5. Параметры оператора BLOBVIEW

|  |  |
| --- | --- |
| Параметр | Описание |
| *blob\_id* | BLOB идентификатор. |

Команда BLOBVIEW отображает данные BLOB в текстовом редакторе по умолчанию.

Пример 4.12. Открытие BLOB во внешнем текстовом редакторе

SQL> set blobdisplay off;  
  
SQL> select proj\_id, proj\_desc  
CON> from project  
CON> where proj\_id = 'MKTPR';  
  
PROJ\_ID PROJ\_DESC  
======= =================  
MKTPR 85:10  
  
SQL> blobview 85:10;

В примере выше содержимое выбранного BLOB будет отображено во внешнем редакторе. Когда редактор закрыт, управление возвращается в isql. Вы не можете использовать isql, пока редактор открыт.

### Примечание

BLOBVIEW может вернуть ошибку «Invalid transaction handle» после закрытия редактора. Для исправления ситуации запустите транзакцию вручную с помощью команды SET TRANSACTION.

Смотри также: [BLOBDUMP](#isql-commands-blobdump).

### EDIT

Назначение: Вызов текстового редактора по умолчанию для скрипта.

Синтаксис:

EDIT [<filename>]

Таблица 4.6. Параметры оператора EDIT

|  |  |
| --- | --- |
| Параметр | Описание |
| *filename* | Имя файла для редактирования. |

Команда EDIT позволяет редактировать существующий файл. Это может быть файл команд SQL, который будет использоваться командой ввода isql (см. ниже), или любой другой текстовый файл. Файл должен существовать.

Если имя файла не указано, история всех ваших предыдущих команд будет отображаться для редактирования.

### Примечание

Обратите внимание, что при выходе из редактора в этом случае команды, оставленные в буфере в конце редактирования, будут выполняться как файл сценария.

### HELP

Назначение: Отображение список команд isql с их описанием.

Синтаксис:

HELP [SET]

Команда HELP отображает список основных команд isql с их описанием.

SQL> HELP;  
Frontend commands:  
BLOBDUMP <blobid> <file> -- dump BLOB to a file  
BLOBVIEW <blobid> -- view BLOB in text editor  
EDIT [<filename>] -- edit SQL script file and execute  
EDIT -- edit current command buffer and execute  
HELP -- display this menu  
INput <filename> -- take input from the named SQL file  
OUTput [<filename>] -- write output to named file  
OUTput -- return output to stdout  
SET <option> -- (Use HELP SET for complete list)  
SHELL <command> -- execute Operating System command in sub-shell  
SHOW <object> [<name>] -- display system information  
 <object> = CHECK, COLLATION, DATABASE, DOMAIN, EXCEPTION, FILTER, FUNCTION,  
 GENERATOR, GRANT, INDEX, PACKAGE, PROCEDURE, ROLE, SQL DIALECT,  
 SYSTEM, TABLE, TRIGGER, VERSION, USERS, VIEW  
EXIT -- exit and commit changes  
QUIT -- exit and roll back changes  
  
All commands may be abbreviated to letters in CAPitals

Если после HELP присутствует необязательное ключевое слово SET, то будет отображена справка по набору команд начинающихся с SET.

SQL> help set;  
Set commands:  
 SET -- display current SET options  
 SET AUTOddl -- toggle autocommit of DDL statements  
 SET BAIL -- toggle bailing out on errors in non-interactive mode  
 SET BLOB [ALL|<n>] -- display BLOBS of subtype <n> or ALL  
 SET BLOB -- turn off BLOB display  
 SET COUNT -- toggle count of selected rows on/off  
 SET MAXROWS [<n>] -- limit select stmt to <n> rows, zero is no limit  
 SET ECHO -- toggle command echo on/off  
 SET EXPLAIN -- toggle display of query access plan in the explained form  
 SET HEADING -- toggle display of query column titles  
 SET LIST -- toggle column or table display format  
 SET NAMES <csname> -- set name of runtime character set  
 SET PLAN -- toggle display of query access plan  
 SET PLANONLY -- toggle display of query plan without executing  
 SET SQL DIALECT <n> -- set sql dialect to <n>  
 SET STATs -- toggle display of performance statistics  
 SET TIME -- toggle display of timestamp with DATE values  
 SET TERM <string> -- change statement terminator string  
 SET WIDTH <col> [<n>] -- set/unset print width to <n> for column <col>  
  
All commands may be abbreviated to letters in CAPitals

### Примечание

Обратите внимание на последнюю строку вывода команды help. Это объясняет, что каждая из команд может быть сокращена до букв, отображаемых заглавными буквами. В следующем обсуждении дополнительные символы будут отображаться, как указано выше, строчными буквами. Например, команда ввода будет показана как INput, чтобы указать, что символы 'put' являются необязательными.

### ADD

Назначение: Добавление записей в таблицу.

Синтаксис:

ADD <tablename>

Таблица 4.7. Параметры оператора ADD

|  |  |
| --- | --- |
| Параметр | Описание |
| *tablename* | Имя таблицы для добавления данных. |

Команда ADD запрашивает данные для каждого столбца, указанной таблицы, и добавляет запись в таблицу. Вы можете добавить столько записей, сколько пожелаете, так как команда продолжается до тех пор, пока не произойдет ошибка или не будет нажата клавиша ENTER без данных. Если вы хотите установить для столбца значение NULL, введите его в точности так, как показано.

Пример 4.13. Добавление данных в таблицу с помощью команды ADD

SQL> add country;  
  
Enter data or NULL for each column. RETURN to end.  
Enter COUNTRY>Scotland  
Enter CURRENCY>GBP  
Enter COUNTRY>  
  
SQL> commit;

### COPY

Назначение: Копирование структуры таблицы.

Синтаксис:

COPY <from\_tablename> <to\_tablename> [<other\_database>]

Таблица 4.8. Параметры оператора COPY

|  |  |
| --- | --- |
| Параметр | Описание |
| *from\_tablename* | Имя таблицы из которой делается копия. |
| *to\_tablenamee* | Имя таблицы в которую будет сделана копия. |
| *other\_database* | Спецификация для подключения к другой базе данных, в которую будет производится копирование. |

Команда COPY позволяет скопировать большую часть структуры таблицы в новую таблицу в текущей базе данных или другой. К сожалению, у неё есть проблемы:

* Она выполняет командную строку для выполнения работы и подключается к принимающей базе данных с помощью приложения с именем isql. Если ваша система переименовала isql в isql-fb, вы на самом деле в конечном итоге запустите неправильное приложение isql, и единственным результатом будет запутанная ошибка.
* Предполагается, что isql будет находится в $PATH или %PATH%.
* Вам необходимо определить ISC\_USER и ISC\_PASSWORD для дочернего процесса isql для входа в получающую базу данных для создания таблицы. Это очень небезопасно.
* Из-за необходимости в ISC\_USER и ISC\_PASSWORD принимающая база данных должна работать на том же сервере, что и исходная база данных.
* Данные таблицы не копируются в принимающую базу данных (или таблицу). Копируются только следующие части структур таблицы:
  + Домены, необходимые для воссоздания таблицы. Это применимо, только если копия находится в другой базе данных.
  + Сама таблица будет создана.
  + Ограничение первичного ключа, если оно есть.
* Копируется не вся структура таблицы. Отсутствуют:
  + Ограничения внешнего ключа.
  + Ограничения проверок (CHECK).
  + Индексы, за исключением индекса необходимого для поддержки первичного ключа.
  + Триггеры.
  + Все данные таблицы.

Если вы хотите скопировать в другую базу данных, то другая база данных должна быть на том же сервере, что и текущая. Например, вы не можете подключиться к базе данных на сервере с именем tux и скопировать таблицу в базу данных, работающую на сервере tuxrep. Команда COPY не может передать имя пользователя и/или пароль, установка ISC\_USER и ISC\_PASSWORD влияет сразу на все подключения, пользователь может отсутствовать на другом сервере или у него может быть другой пароль.

Пример 4.14. Копирование структуры таблицы командой COPY

tux>$ export ISC\_USER=SYSDBA  
tux>$ export ISC\_PASSWORD=secret  
tux>$ isql employee  
  
Database: employee, User: sysdba  
  
SQL> -- MAke a copy of the employee table into this database.  
SQL> copy employee employee\_2;  
  
SQL> -- Compare table structures...  
SQL> show table employee;  
  
EMP\_NO (EMPNO) SMALLINT Not Null  
FIRST\_NAME (FIRSTNAME) VARCHAR(15) Not Null  
LAST\_NAME (LASTNAME) VARCHAR(20) Not Null  
PHONE\_EXT VARCHAR(4) Nullable  
HIRE\_DATE TIMESTAMP Not Null DEFAULT 'NOW'  
DEPT\_NO (DEPTNO) CHAR(3) Not Null  
 CHECK (VALUE = '000' OR  
 (VALUE > '0' AND VALUE <= '999') OR VALUE IS NULL)  
JOB\_CODE (JOBCODE) VARCHAR(5) Not Null  
 CHECK (VALUE > '99999')  
JOB\_GRADE (JOBGRADE) SMALLINT Not Null  
CHECK (VALUE BETWEEN 0 AND 6)  
JOB\_COUNTRY (COUNTRYNAME) VARCHAR(15) Not Null  
SALARY (SALARY) NUMERIC(10, 2) Not Null DEFAULT 0  
 CHECK (VALUE > 0)  
FULL\_NAME Computed by: (last\_name || ', ' || first\_name)  
CONSTRAINT INTEG\_28:  
 Foreign key (DEPT\_NO) References DEPARTMENT (DEPT\_NO)  
CONSTRAINT INTEG\_29:  
 Foreign key (JOB\_CODE, JOB\_GRADE, JOB\_COUNTRY)  
 References JOB (JOB\_CODE, JOB\_GRADE, JOB\_COUNTRY)  
CONSTRAINT INTEG\_27:  
 Primary key (EMP\_NO)  
CONSTRAINT INTEG\_30:  
 CHECK ( salary >= (SELECT min\_salary FROM job WHERE  
 job.job\_code = employee.job\_code AND  
 job.job\_grade = employee.job\_grade AND  
 job.job\_country = employee.job\_country) AND  
 salary <= (SELECT max\_salary FROM job WHERE  
 job.job\_code = employee.job\_code AND  
 job.job\_grade = employee.job\_grade AND  
 job.job\_country = employee.job\_country))  
Triggers on Table EMPLOYEE:  
SET\_EMP\_NO, Sequence: 0, Type: BEFORE INSERT, Active  
SAVE\_SALARY\_CHANGE, Sequence: 0, Type: AFTER UPDATE, Active  
  
SQL> show table employee\_2;  
  
EMP\_NO (EMPNO) SMALLINT Not Null  
FIRST\_NAME (FIRSTNAME) VARCHAR(15) Not Null  
LAST\_NAME (LASTNAME) VARCHAR(20) Not Null  
PHONE\_EXT VARCHAR(4) Nullable  
HIRE\_DATE TIMESTAMP Not Null DEFAULT 'NOW'  
DEPT\_NO (DEPTNO) CHAR(3) Not Null  
 CHECK (VALUE = '000' OR  
 (VALUE > '0' AND VALUE <= '999') OR VALUE IS NULL)  
JOB\_CODE (JOBCODE) VARCHAR(5) Not Null  
 CHECK (VALUE > '99999')  
JOB\_GRADE (JOBGRADE) SMALLINT Not Null  
 CHECK (VALUE BETWEEN 0 AND 6)  
JOB\_COUNTRY (COUNTRYNAME) VARCHAR(15) Not Null  
SALARY (SALARY) NUMERIC(10, 2) Not Null DEFAULT 0  
CHECK (VALUE > 0)  
FULL\_NAME Computed by: (last\_name || ', ' || first\_name)  
CONSTRAINT INTEG\_93:  
 Primary key (EMP\_NO)  
   
SQL> -- Check indices on both tables...  
SQL> show indices employee;  
  
NAMEX INDEX ON EMPLOYEE(LAST\_NAME, FIRST\_NAME)  
RDB$FOREIGN8 INDEX ON EMPLOYEE(DEPT\_NO)  
RDB$FOREIGN9 INDEX ON EMPLOYEE(JOB\_CODE, JOB\_GRADE, JOB\_COUNTRY)  
RDB$PRIMARY7 UNIQUE INDEX ON EMPLOYEE(EMP\_NO)  
  
SQL> show indices employee\_2;  
RDB$PRIMARY27 UNIQUE INDEX ON EMPLOYEE\_2(EMP\_NO)  
  
SQL> -- Check data counts on both tables...  
SQL> select count(\*) from employee;  
  
 COUNT  
============  
 42  
  
SQL> select count(\*) from employee\_2;  
  
 COUNT  
============  
 0

Команда COPY работает только при условии, что ваше приложение isql действительно называется isql. Кроме того, если в таблице много данных, вам все равно придется копировать их вручную, так как команда копирования будет копировать только структуру таблицы. Помните, что в новой таблице не будет ни триггеров, ни внешних ключей, ни индексов, кроме первичного ключа, ни данных.

### Примечание

Возможно команда COPY будет удалена из ISQL в следующих версиях.

### INPUT

Назначение: Подключение файла скрипта.

Синтаксис:

INPUT <filename>

Таблица 4.9. Параметры оператора INPUT

|  |  |
| --- | --- |
| Параметр | Описание |
| *filename* | Имя файла с подключаемым скриптом. |

Команда INPUT 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.

Пример 4.15. Использование команды INPUT для включения скрипта

SQL> shell;  
  
$ cat test.sql  
drop table fred;  
commit;  
  
$ exit;  
  
SQL> show table fred;  
  
A INTEGER Nullable  
B INTEGER Not Null  
  
SQL> input test.sql;  
  
SQL> show table fred;  
There is no table FRED in this database

Смотри также: [SHELL](#isql-commands-shell).

### OUTPUT

Назначение: Перенаправление вывода.

Синтаксис:

OUTPUT [<filename>]

Таблица 4.10. Параметры оператора OUTPUT

|  |  |
| --- | --- |
| Параметр | Описание |
| *filename* | Имя файла для вывода. |

Команда OUTPUT перенаправляет весь вывод, который обычно отображается на экране, в определенный файл. Если указано имя файла, все последующие выходные данные отправляются в этот файл и не отображаются на экране. Если имя файла не указано, вывод снова перенаправляется на экран.

Пример 4.16. Использование команды OUTPUT

SQL> output test.log;  
  
SQL> show tables;  
  
SQL> output;  
  
SQL> shell;  
  
$ cat test.log  
  
COUNTRY CUSTOMER  
DEPARTMENT EMPLOYEE  
EMPLOYEE\_PROJECT FRED  
JOB PROJECT  
PROJ\_DEPT\_BUDGET SALARY\_HISTORY  
SALES

Смотри также: [SHELL](#isql-commands-shell).

### SET

Назначение: Отображение текущий установок.

Синтаксис:

SET

Команда SET выводит текущие опции установленные с помощью команд SET.

Пример 4.17. Использование команды SET

SQL> SET;  
  
Print statistics: OFF  
Echo commands: OFF  
List format: OFF  
Show Row Count: OFF  
Select maxrows limit: 0  
Autocommit DDL: ON  
Access Plan: OFF  
Access Plan only: OFF  
Explain Access Plan: OFF  
Display BLOB type: 1  
Column headings: ON  
Terminator: ;  
Time: OFF  
Warnings: ON  
Bail on error: OFF  
Local statement timeout: 0  
  
SQL>

### SHELL

Назначение: Предоставление доступа к командной строке.

Синтаксис:

SHELL [<command>]

Таблица 4.11. Параметры оператора SHELL

|  |  |
| --- | --- |
| Параметр | Описание |
| *command* | Команда системной консоли. |

Команда SHELL позволяет вам временно выйти из isql и использовать сеанс системной оболочки для дальнейшей обработки. При этом не происходит подтверждение или откат любой транзакции.

Аргумент *<command>* 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. При выходе из оболочки вы вернетесь в isql. Вы не можете использовать сеанс isql, который активировал оболочку, пока сеанс оболочки остается открытым.

Пример 4.18. Использование команды SHELL

SQL> shell;  
  
$ cat test.log  
  
 COUNTRY CUSTOMER  
 DEPARTMENT EMPLOYEE  
 EMPLOYEE\_PROJECT FRED  
 JOB PROJECT  
 PROJ\_DEPT\_BUDGET SALARY\_HISTORY  
 SALES  
$ exit  
SQL>

### EXIT

Назначение: Выход из isql.

Синтаксис:

EXIT

Команда EXIT подтверждает текущую транзакцию без подсказки, закрывает базу данных и завершает сессию isql.

### QUIT

Назначение: Выход из isql.

Синтаксис:

QUIT

Команда QUIT отменяет текущую транзакцию и закрывает базу данных и завершает сессию isql.

## ISQL SET команды

Как объясняется в команде HELP, вы можете ввести команду HELP SET, чтобы перейти к различным опциям доступных для SET команды. Команда HELP SET выводит следующую справку.

SQL> help set;  
  
Set commands:  
 SET -- display current SET options  
 SET AUTOddl -- toggle autocommit of DDL statements  
 SET BAIL -- toggle bailing out on errors in non-interactive mode  
 SET BLOB [ALL|<n>] -- display BLOBS of subtype <n> or ALL  
 SET BLOB -- turn off BLOB display  
 SET COUNT -- toggle count of selected rows on/off  
 SET MAXROWS [<n>] -- limit select stmt to <n> rows, zero is no limit  
 SET ECHO -- toggle command echo on/off  
 SET EXPLAIN -- toggle display of query access plan in the explained form  
 SET HEADING -- toggle display of query column titles  
 SET LIST -- toggle column or table display format  
 SET NAMES <csname> -- set name of runtime character set  
 SET PLAN -- toggle display of query access plan  
 SET PLANONLY -- toggle display of query plan without executing  
 SET SQL DIALECT <n> -- set sql dialect to <n>  
 SET STATs -- toggle display of performance statistics  
 SET TIME -- toggle display of timestamp with DATE values  
 SET TERM <string> -- change statement terminator string  
 SET WIDTH <col> [<n>] -- set/unset print width to <n> for column <col>  
  
All commands may be abbreviated to letters in CAPitals

### Примечание

В выше приведённом примере команды BLOB являются неполными. Они должны быть отображены как BLOBdisplay. Далее мы будем использовать полную версию команды.

Последняя строка вышеприведенного вывода указывает, что эти команды могут быть сокращены до заглавных букв. К сожалению, кроме команды SET AUTOddl, ни одна из других не имеет краткой формы.

### SET

Назначение: Отображение текущий установок.

Синтаксис:

SET

Команда SET выводит текущие опции установленные с помощью команд SET.

Пример 4.19. Использование команды SET

SQL> SET;  
  
Print statistics: OFF  
Echo commands: OFF  
List format: OFF  
Show Row Count: OFF  
Select maxrows limit: 0  
Autocommit DDL: ON  
Access Plan: OFF  
Access Plan only: OFF  
Explain Access Plan: OFF  
Display BLOB type: 1  
Column headings: ON  
Terminator: ;  
Time: OFF  
Warnings: ON  
Bail on error: OFF  
  
SQL>

### SET AUTODDL

Назначение: Установка или отключение автоматического подтверждения для DDL операторов.

Синтаксис:

SET AUTOddl [ON | OFF]

Команда SET AUTODDL устанавливает будет ли автоматически подтверждаться транзакция для DDL операторов. Команда без каких-либо параметров действует как переключатель, она отключает функцию AUTODLL, если она включена или наоборот. Вы можете указать конкретный параметр, чтобы прояснить свои намерения. Параметр должен быть одним из двух ON или OFF. Команда SET без параметров отобразит текущую настройку. По умолчанию функция AUTODDL включена.

### SET BAIL

Назначение: Передача обработки ошибок операционной системе.

Синтаксис:

SET BAIL [ON | OFF]

Команда SET BAIL устанавливает, будет ли isql останавливать выполнение скрипта в пакетном режиме при возникновении ошибки и передавать управление операционной системе, возвращая при этом код ошибки. В интерактивном режиме при возникновении ошибок isql не будет прекращать свою работу.

Выполнение этой команды без передачи параметра приводит к переключению текущего состояния. Если SET BAIL включен, то он отключится и наоборот.

### SET BLOB

Назначение: Включение или выключение отображения подтипа BLOB.

Синтаксис:

SET BLOB [<n> |ALL | OFF]   
   
SET BLOBDISPLAY [<n> |ALL | OFF]

Таблица 4.12. Параметры оператора SET BLOB

|  |  |
| --- | --- |
| Параметр | Описание |
| *n* | Номер подтипа BLOB. Значение по умолчанию n=1 (текст). Положительные числа определены в системе; отрицательные числа определяются пользователем. |

Команда SET BLOB определяет, должны ли данные столбца BLOB отображаться в выводе. Если параметры не переданы в команду SET BLOB или передан параметр OFF, то поля типа BLOB отображаться не будут, вместо них будут отображаться идентификаторы BLOB. Идентификатор BLOB представляет собой два шестнадцатеричных числа, разделенных двоеточием. Первое число — это идентификатор таблицы, а второе — последовательный номер в базе данных.

Если в команду SET BLOB передан параметр ALL, то будут отображаться BLOB поля всех подтипов.

Если передано число, представляющее подтип BLOB, то будут отображаться только BLOB с указанным подтипом. По умолчанию отображаются поля с подтипом 1 (текстовые данные).

Пример 4.20. Использование SET BLOB

SQL> -- Don't display any blob data.  
SQL> set blob off;  
  
SQL> select proj\_desc  
CON> from project  
CON> where proj\_id = 'HWRII';  
  
 PROJ\_DESC  
=================  
 85:1e4  
  
SQL> -- Display all blob data.  
SQL> set blobdisplay all;  
  
SQL> select proj\_desc  
CON> from project  
CON> where proj\_id = 'HWRII';  
  
 PROJ\_DESC  
=================  
 85:1e4  
==============================================================================  
PROJ\_DESC:  
Integrate the hand-writing recognition module into the  
universal language translator.  
  
==============================================================================  
  
SQL> -- Only display type 1 blob data = text.  
SQL> set blob 1;  
  
SQL> select proj\_desc  
CON> from project  
CON> where proj\_id = 'HWRII';  
  
 PROJ\_DESC  
=================  
 85:1e4  
==============================================================================  
PROJ\_DESC:  
Integrate the hand-writing recognition module into the  
universal language translator.  
  
==============================================================================  
  
SQL> -- Only display blob type 7 = not text!  
SQL> set blob 7;  
  
SQL> select proj\_desc  
CON> from project  
CON> where proj\_id = 'HWRII';  
  
 PROJ\_DESC  
=================  
 85:1e4  
==============================================================================  
PROJ\_DESC:  
BLOB display set to subtype 7. This BLOB: subtype = 1  
==============================================================================

В последнем примере вы заметите, что было отображено сообщение о том, что мы отображаем данные BLOB только для подтипа 7, а данные BLOB в этой таблице являются подтипом 1, поэтому данные не отображаются.

Смотри также: [BLOBDUMP](#isql-commands-blobdump), [BLOBVIEW](#isql-commands-blobview).

### SET BULK\_INSERT

Назначение: Пакетная вставка записей в таблицу.

Синтаксис:

SET BULK\_INSERT <prepared insert statement>

Таблица 4.13. Параметры оператора SET BULK\_INSERT

|  |  |
| --- | --- |
| Параметр | Описание |
| *prepared insert statement* | Подготовленный оператор INSERT. |

Команда SET BULK\_INSERT осуществляет примитивную обработку 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.

Пример 4.21. Использование SET BULK\_INSERT для пакетной вставки

SQL> create table t1(id int not null primary key, name varchar(50));  
  
SQL> set bulk\_insert insert into t1(id, name) values(?, ?);  
BULK> (1, 'a')  
BULK> (2, 'b')  
BULK> (3, 'c')  
BULK> commit;  
BULK>  
SQL> select \* from t1;  
  
 ID NAME  
============ ==================================================  
 1 a  
 2 b  
 3 c  
   
SQL> set bulk\_insert insert into t1(id, name) values(?, ?);  
BULK> -- insert 4 record  
BULK> (4, 'abcd')  
BULK> (5, +++  
CON> 'efg')  
BULK> stop  
  
SQL> commit;  
  
SQL> select \* from t1;  
  
 ID NAME  
============ ==================================================  
 1 a  
 2 b  
 3 c  
 4 abcd  
 5 efg

### SET COUNT

Назначение: Включение или выключение отображения количества строк затронутых запросом.

Синтаксис:

SET COUNT [ON|OFF]

Команда SET COUNT включает или выключает отображение количества строк затронутых DML запросом. Выполнение этой команды без передачи параметра приводит к переключению текущего состояния.

SQL> set count on;  
  
SQL> select count(\*) from employee;  
  
 COUNT  
=====================  
 42  
  
Records affected: 1

Количество затронутых записей отображается для всех DDL операторов, а не только для SELECT.

SQL> create table fred( a integer);  
SQL> commit;  
  
SQL> insert into fred values (666);  
Records affected: 1  
  
SQL> update fred set a = 123 where a = 666;  
Records affected: 1  
  
SQL> delete from fred;  
Records affected: 1  
  
SQL> commit;

Смотри также: SET ROWCOUNT.

### SET ECHO

Назначение: Включение или выключение отображения команд до их выполнения.

Синтаксис:

SET ECHO [ON|OFF]

Команда SET ECHO включает или выключает отображение команд до их выполнения. Вы может отключить эхо-вывод для части файла сценария, хотя по умолчанию эхо вывод выключен в ISQL. Выполнение этой команды без передачи параметра приводит к переключению текущего состояния.

Пример 4.22. Использование команды SET ECHO

SQL> set echo on;  
  
SQL> select count(\*) from rdb$database;  
select count(\*) from rdb$database;  
  
 COUNT  
=====================  
 1  
  
Records affected: 1  
SQL> set echo off;  
set echo off;  
  
SQL> select count(\*) from rdb$database;  
  
 COUNT  
=====================  
 1  
  
Records affected: 1

Эта команда может быть полезна в файле сценария. Если вы получаете сообщение об ошибке, иногда бывает трудно определить точный оператор SQL, вызвавший ее. Если вы установите ECHO ON в своем скрипте, вы по крайней мере сможете точно определить, какой оператор потерпел неудачу.

### SET EXPLAIN

Назначение: Включение или выключение отображения расширенного плана запроса.

Синтаксис:

SET EXPLAIN [ON|OFF]

Команда SET EXPLAIN включает или выключает отображение расширенного плана запроса. Расширенный план выводит более подробную информацию о методах доступа используемых оптимизатором, однако его нельзя включить в запрос. По умолчанию вывод расширенного плана запроса отключен. Выполнение этой команды без передачи параметра приводит к переключению текущего состояния.

Пример 4.23. Использование команды SET EXPLAIN

SQL> set explain on;  
SQL> set planonly on;  
SQL> select \* from employee cross join rdb$database;  
  
Select Expression  
 -> Nested Loop Join (inner)  
 -> Table "RDB$DATABASE" Full Scan  
 -> Table "EMPLOYEE" Full Scan  
SQL>

### Подсказка

Если вам необходимо вывести только план запроса без его выполнения, то вы можете выполнить команду SET PLANONLY ON.

Смотри также: [SET PLAN](#isql-commands_set-set_plan), [SET PLANONLY](#isql-commands_set-set_planonly).

### SET HEADING

Назначение: Включение или выключение отображения заголовков столбцов.

Синтаксис:

SET HEADING [ON|OFF]

Команда SET HEADING включает или выключает отображение заголовков столбцов. По умолчанию отображение заголовков столбцов включено. Выполнение этой команды без передачи параметра приводит к переключению текущего состояния.

Пример 4.24. Использование команды SET HEADING

SQL> set plan off;  
SQL> set heading off;  
SQL> select count(\*) from employee;  
  
 42  
  
SQL> set heading on;  
SQL> select count(\*) from employee;  
  
 COUNT  
=====================  
 42

### SET LIST

Назначение: Включение или выключение отображения вывода результатов SELECT запроса в виде списка.

Синтаксис:

SET LIST [ON|OFF]

Команда SET LIST управляет отображением данных, возвращаемых оператором SELECT. По умолчанию данные на экране выводятся в табличном варианте, где сверху названия столбцов, а под ними все строки, удовлетворяющие запросу SELECT. Включение режима списка приводит к другому формату, в котором заголовок каждого столбца отображается слева, а данные столбца — справа. Это повторяется для каждой строки, возвращаемой запросом.

Выполнение этой команды без передачи параметра приводит к переключению текущего состояния.

Пример 4.25. Использование команды SET LIST

SQL> set list off;  
SQL> select emp\_no, first\_name, last\_name, salary  
CON> from employee;  
  
 EMP\_NO FIRST\_NAME LAST\_NAME SALARY  
======= =============== ==================== =====================  
 2 Robert Nelson 105900.00  
 4 Bruce Young 97500.00  
 5 Kim Lambert 102750.00  
 8 Leslie Johnson 64635.00  
.............  
  
SQL> set list on;  
SQL> select emp\_no, first\_name, last\_name, salary  
CON> from employee;  
  
EMP\_NO 2  
FIRST\_NAME Robert  
LAST\_NAME Nelson  
SALARY 105900.00  
  
EMP\_NO 4  
FIRST\_NAME Bruce  
LAST\_NAME Young  
SALARY 97500.00  
  
..........

### SET NAMES

Назначение: Установка набора символов для последующий подключений.

Синтаксис:

SET NAMES [<character\_set>]

Таблица 4.14. Параметры оператора SET NAMES

|  |  |
| --- | --- |
| Параметр | Описание |
| *character\_set* | Набор символов для последующих подключений. |

Оператор SET NAMES устанавливает набор символов для последующих подключений к базе данных с помощью isql. Это позволяет вам переопределить набор символов по умолчанию для базы данных. Чтобы вернуться к использованию набора символов по умолчанию, используйте SET NAMES без аргументов.

Используйте команду SET NAMES перед подключением к базе данных для того, чтобы указать требуемый набор символов. Список поддерживаемых наборов символов можно посмотреть выполнив запрос:

SQL> select RDB$CHARACTER\_SET\_NAME  
CON> from RDB$CHARACTER\_SETS  
CON> order by 1;  
  
RDB$CHARACTER\_SET\_NAME  
==============================  
ASCII  
BIG\_5  
CP943C  
CYRL  
...  
DOS864  
DOS865  
DOS866  
DOS869  
...  
NONE  
OCTETS  
...  
UTF8  
...  
WIN1250  
WIN1251  
WIN1252  
...

Выбор набора символов ограничивает возможные порядки сопоставления (сортировки) подмножеством всех доступных порядков сопоставления для заданного набора символов.

Если набор символов по умолчанию для базы данных не равен NONE, то в ситуациях, когда клиент использует другой набор символов для базы данных, возможно пострадать от повреждения данных, поскольку некоторые наборы символов не могут преобразовать некоторые символы в подходящий символ в другом наборе символов.

Если вы не передадите набор символов, то по умолчанию будет использоваться набор символов NONE.

### SET PLAN

Назначение: Включение или отключение отображение плана запроса.

Синтаксис:

SET PLAN [ON | OFF]

Команда SET PLAN определяет, будет ли isql отображать план, который он использовал для доступа к данным для каждого выполненного оператора. По умолчанию isql никогда не отображает план. Как и во многих других командах, отсутствие параметра переключает текущее состояние.

Пример 4.26. Использование команды SET PLAN

SQL> set plan on;  
  
SQL> select emp\_no, first\_name, last\_name  
CON> from employee  
CON> where emp\_no = 107;  
  
PLAN (EMPLOYEE INDEX (RDB$PRIMARY7))  
  
 EMP\_NO FIRST\_NAME LAST\_NAME  
======= =============== ====================  
 107 Kevin Cook  
  
SQL> update employee  
CON> set first\_name = 'Norman'  
CON> where last\_name = 'Cook';  
  
PLAN (EMPLOYEE INDEX (NAMEX))  
  
SQL> select count(\*) from employee;  
  
PLAN (EMPLOYEE NATURAL)  
  
 COUNT  
=====================  
 42

План выполнения отображается перед выводом из оператора SELECT.

### Подсказка

Для вывода расширенного плана используйте команду SET EXPLAIN.

Смотри также: [SET EXPLAIN](#isql-commands_set-set_explain), [SET PLANONLY](#isql-commands_set-set_planonly).

### SET PLANONLY

Назначение: Включение или отключение режима отображение только плана запроса без его выполнения.

Синтаксис:

SET PLANONLY [ON | OFF]

Команда SET PLANONLY ON не позволяет Firebird фактически выполнить оператор SQL, а просто показывает план, который будет использоваться для доступа к данным. Эта команда опирается на команду SET PLAN. Если SET PLAN OFF был выполнен, то эта команда не имеет никакого эффекта, поэтому включение PLANONLY ON имеет дополнительный эффект неявного выполнения SET PLAN ON. Выполнение SET PLANONLY OFF не вызывает неявного выполнения SET PLAN OFF.

Пример 4.27. Использование команды SET PLANONLY

SQL> set planonly on;  
  
SQL> select count(\*) from employee;  
  
PLAN (EMPLOYEE NATURAL)

Как и во многих других командах, отсутствие параметра переключает текущее состояние.

Действие команды SET PLANONLY ON также распространяется и на вывод расширенного плана без фактического выполнения запроса.

Пример 4.28. Использование команды SET PLANONLY совместно с SET EXPLAIN

SQL> set explain on;  
SQL> set planonly on;  
  
SQL> select count(\*) from employee;  
  
Select Expression  
 -> Aggregate  
 -> Table "EMPLOYEE" Full Scan

Смотри также: [SET PLAN](#isql-commands_set-set_plan), [SET EXPLAIN](#isql-commands_set-set_explain).

### SET SQLDA\_DISPLAY

Назначение: Включение или отключение отображения информации о внутренних структурах для входных и выходных сообщений.

Синтаксис:

SET SQLDA\_DISPLAY [ON | OFF]

Это скрытая команда, которая не упоминается в выводе команды HELP SET. Она отображает внутренние подробности о структурах для входных и выходных сообщений операторов SQL, выполняемых isql. Раньше она была доступна только в специальной отладочной сборке, но начиная с версии 2.0 она доступна в isql.

Пример 4.29. Использование команды SET SQLDA\_DISPLAY

SQL> set sqlda\_display on;  
  
SQL> select count(\*) from employee;  
  
INPUT message field count: 0  
  
OUTPUT message field count: 1  
01: sqltype: 580 INT64 scale: 0 subtype: 0 len: 8  
 : name: COUNT alias: COUNT  
 : table: owner:  
  
 COUNT  
=====================  
 42  
   
SQL> select emp\_no as id, first\_name from employee fetch first row only;  
  
INPUT message field count: 0  
  
OUTPUT message field count: 2  
01: sqltype: 500 SHORT scale: 0 subtype: 0 len: 2  
 : name: EMP\_NO alias: ID  
 : table: EMPLOYEE owner: SYSDBA  
02: sqltype: 448 VARYING scale: 0 subtype: 0 len: 15 charset: 0 NONE  
 : name: FIRST\_NAME alias: FIRST\_NAME  
 : table: EMPLOYEE owner: SYSDBA  
  
 ID FIRST\_NAME  
======= ===============  
 2 Robert   
   
SQL> select emp\_no as id, first\_name from employee   
CON> where emp\_no = ? fetch first row only;  
  
INPUT message field count: 1  
01: sqltype: 500 SHORT scale: 0 subtype: 0 len: 2  
 : name: alias:  
 : table: owner:  
  
OUTPUT message field count: 2  
01: sqltype: 500 SHORT scale: 0 subtype: 0 len: 2  
 : name: EMP\_NO alias: ID  
 : table: EMPLOYEE owner: SYSDBA  
02: sqltype: 448 VARYING scale: 0 subtype: 0 len: 15 charset: 0 NONE  
 : name: FIRST\_NAME alias: FIRST\_NAME  
 : table: EMPLOYEE owner: SYSDBA  
Statement failed, SQLSTATE = 07002  
Dynamic SQL Error  
-SQLDA error  
-No SQLDA for input values provided

### Примечание

Обратите внимание, что при запуске команд HELP SET или SET информация об этой команде не отображается.

### Подсказка

Как видите последний оператор вызвал ошибку, поскольку isql не позволяет запрашивать значения для входных параметров. Вы можете подавить её отключив фактическое выполнение команды с помощью SET PLANONLY ON.

### SET SQL DIALECT

Назначение: Установка SQL диалекта.

Синтаксис:

SET SQL DIALECT {1 | 2 | 3}

Команда SET SQL DIALECT задает диалект SQL Firebird, на который должен быть изменен сеанс клиента. Если сеанс в настоящее время присоединен к базе данных, отличной от диалекта, указанного в команде, отображается предупреждение.

Посмотрите [Диалекты SQL](#isql-sql_dialect) для того, чтобы узнать о различиях между тремя диалектами.

Пример 4.30. Использование команды SET SQL DIALECT

SQL> set sql dialect 1;  
WARNING: Client SQL dialect has been set to 1 when   
connecting to Database SQL dialect 3 database.  
...  
SQL> set sql dialect 3;

### Примечание

Предупреждение в приведенном выше примере пришлось разделить на две строки, чтобы оно поместилось на странице. Обычно оно состоит из одной строки.

### SET STATs

Назначение: Включение или отключение отображения статистки.

Синтаксис:

SET STATs [ON | OFF]

Команда SET STATS определяет, должен ли isql отображать различную статистику о каждой выполненной команде SQL. Как и во многих других командах, отсутствие параметра переключает текущее состояние.

Пример 4.31. Использование команды SET STATS

SQL> set stat on;  
SQL> select count(\*) from employee;  
  
 COUNT  
=====================  
 42  
  
Current memory = 19458128  
Delta memory = 282544  
Max memory = 19531888  
Elapsed time= 0.103 sec  
Buffers = 2048  
Reads = 28  
Writes = 0  
Fetches = 360

### SET TIME

Назначение: Включение или отключение отображение времени для типа DATE в 1-диалекте.

Синтаксис:

SET TIME [ON | OFF]

Команда SET TIME применяется только к базам данных диалекта 1. Её выполнение приводит к тому, что часть времени отображается или нет, когда выбираются данные с типом данных DATE. Команда не имеет действия с базами данных в других диалектах.

### SET TERM

Назначение: Установка символа (символов) терминатора команды или оператора.

Синтаксис:

SET TERM <new\_terminator> <current\_terminator>

Таблица 4.15. Параметры оператора SET TERM

|  |  |
| --- | --- |
| Параметр | Описание |
| *new\_terminator* | Новый символ (символы) терминатор. |
| *current\_terminator* | Текущий символ (символы) терминатор. |

Команда SET TERM изменяет терминатор оператора по умолчанию с точки с запятой на что-то другое, определенное в переданной строке. Это полезно, когда вы собираетесь ввести строку SQL-операторов, составляющих, например, процедуру или триггер. Isql попытается выполнить каждый оператор, когда встречает завершающую точку с запятой, поэтому сначала нужно изменить терминатор, а затем ввести требуемый код. После завершения ввода, вы можете изменить его обратно. При этом необходимо помнить, что вы должны завершить команду SET TERM текущим символом (символами) терминатором.

При первом запуске isql использует точку с запятой в качестве терминатора по умолчанию.

Пример 4.32. Использование команды SET TERM

SQL> set term ~ ;  
  
SQL> create procedure test\_proc (iInput integer = 666)  
CON> returns (oOutput integer)  
CON> as  
CON> begin  
CON> oOutput = iInput;  
CON> suspend;  
CON> end~  
  
SQL> set term ; ~  
  
SQL> commit;  
  
SQL> select \* from test\_proc;  
  
 OOUTPUT  
============  
 666

Смотри также: [Символ терминатора](#isql-terminator).

### SET WARNINGS

Назначение: Включение или выключение вывода предупреждающих сообщений.

Синтаксис:

SET {WARNINGS | WNG} [ON | OFF]

Команда SET WARNINGS включает или выключает вывод предупреждающих сообщений.

### SET WIDTH

Назначение: Установка ширины вывода для символьных столбцов.

Синтаксис:

SET WIDTH <column\_name> [<width>]

Таблица 4.16. Параметры оператора SET WIDTH

|  |  |
| --- | --- |
| Параметр | Описание |
| *column\_name* | Наименование или псевдоним символьного столбца. |
| *width* | Ширина вывода в символах. |

Обычно ширина вывода символьных столбцов в таблице определяется по размеру выбираемых символьных полей. Использование команды SET WIDTH позволяет пользователю определять ширину выходного столбца в большую или меньшую сторону. Если параметр *width* не указан, то будет использована ширина столбца по умолчанию для этого столбца.

В следующем примере показано изменение ширины столбца last\_name. Первый SELECT отображается с настройками по умолчанию. В этом случае ширина столбца last\_name составляет 20 символов (считайте количество символов «=» в заголовке). Она задаётся определением столбца last\_name в таблице employee. Второй пример показывает таблицу с шириной, уменьшенной до 10 символов.

SQL> select emp\_no, last\_name  
CON> from employee  
CON> order by last\_name  
CON> fetch first 5 rows only;  
  
 EMP\_NO LAST\_NAME  
======= ====================  
 34 Baldwin  
 105 Bender  
 28 Bennet  
 83 Bishop  
 109 Brown  
  
SQL> set width last\_name 10;  
  
SQL> select emp\_no, last\_name  
CON> from employee  
CON> order by last\_name  
CON> fetch first 5 rows only;  
  
 EMP\_NO LAST\_NAME  
======= ==========  
 34 Baldwin  
 105 Bender  
 28 Bennet  
 83 Bishop  
 109 Brown

Столбец emp\_no имеет тип SMALLINT. ISQL не может изменить ширину столбцов, которые не являются символьными типами. В следующем примере команда set width emp\_no 10 не имеет никакого эффекта. В нём также показано отключение настройки ширины для столбца last\_name.

SQL> set width last\_name;  
  
SQL> set width emp\_no 10;  
  
SQL> select emp\_no, last\_name  
CON> from employee  
CON> order by last\_name  
CON> fetch first 5 rows only;  
  
 EMP\_NO LAST\_NAME  
======= ====================  
 34 Baldwin  
 105 Bender  
 28 Bennet  
 83 Bishop  
 109 Brown

## ISQL SHOW команды

Как объясняется в команде HELP, в isql есть несколько отдельных команд SHOW. Общий синтаксис следующий:

SHOW <object> [<name>]

Таблица 4.17. Параметры оператора SHOW

|  |  |
| --- | --- |
| Параметр | Описание |
| *object* | Тип отображаемого или отображаемых объектов метаданных. |
| *name* | Наименование объекта метаданных. |

Тип объекта метаданных требуется указывать всегда, а имя требуется для отображения сведений о конкретном объекте. Без имени команды обычно отображают все имена объекты запрошенного типа.

В отличие от команд SET, нет удобной детализации различных команд SHOW, c использованием команды HELP. Однако, если вы наберете SHOW без аргументов, то вам будет оказана небольшая помощь.

SQL> show;  
  
Valid options are:  
CHECKs COMMENTs COLLATEs  
COLLATIONs DOMAINs DB  
DATABASE DEPENdency DEPENdencies  
EXCEPtions FILTERs FUNCtions  
GENerators GRANTs INDexes  
INDICES MAPping PROCedures  
PACKages ROLEs SYStem  
SEQuences SECURITY CLAsses SECCLAsses  
SCHEmas TABLEs TRIGgers  
USERS VIEWs  
Command error: show

Каждый тип объекта метаданных может быть сокращен до букв, отображаемых заглавными буквами.

### Примечание

Подсказка выданная командой SHOW содержит несколько ошибок. Например следующие команды не работают:

* SHOW SCHEmas (очевидно зарезервирована для будущих версий где могут появится схемы);
* SHOW SECURITY CLAsses

Команды SHOW подробно детализированы и описаны ниже. Где это возможно, приведены примеры из базы данных employee.

### SHOW CHECK

Назначение: Отображение ограничений для указанной таблицы.

Синтаксис:

SHOW CHECKs <table\_name>

Таблица 4.18. Параметры оператора SHOW CHECK

|  |  |
| --- | --- |
| Параметр | Описание |
| *table\_name* | Наименование таблицы для которой требуется отобразить ограничения. |

Команда SHOW CHECKs отображает все проверочные ограничения, определенные пользователем для заданной таблицы. В отличие от других команд SHOW, здесь нет опции для отображения списка всех проверочных ограничений в базе данных. Вы всегда должны указывать имя таблицы как часть команды.

Пример 4.33. Использование команды SHOW CHECK

SQL> show check employee;  
  
CONSTRAINT INTEG\_30:  
 CHECK ( salary >= (SELECT min\_salary FROM job WHERE  
 job.job\_code = employee.job\_code AND  
 job.job\_grade = employee.job\_grade AND  
 job.job\_country = employee.job\_country) AND  
 salary <= (SELECT max\_salary FROM job WHERE  
 job.job\_code = employee.job\_code AND  
 job.job\_grade = employee.job\_grade AND  
 job.job\_country = employee.job\_country))

### SHOW COLLATION

Назначение: Отображение списка пользовательских параметров сортировок или параметров для указанной сортировке.

Синтаксис:

SHOW {COLLATIONs | COLLATION <collate\_name> }   
   
SHOW {COLLATEs | COLLATE <collate\_name> }

Таблица 4.19. Параметры оператора SHOW COLLATION

|  |  |
| --- | --- |
| Параметр | Описание |
| *collate\_name* | Имя определённой пользователем сортировки. |

Эти команды отображают список всех параметров для опреределённых пользователем сортировок в текущей базе данных. В первой форме отображается список всех параметров всех определенных пользователем сортировок. Если указано имя конкретной сортировки, то будут отображаться только её параметры.

Пример 4.34. Использование команды SHOW COLLATION

SQL> show collations;  
UNICODE\_ENUS\_CI, CHARACTER SET UTF8, FROM EXTERNAL ('UNICODE'), PAD SPACE,  
CASE INSENSITIVE, 'COLL-VERSION=58.0.6.48'  
UNICODE\_ENUS\_CS, CHARACTER SET UTF8, FROM EXTERNAL ('UNICODE'), PAD SPACE,  
'COLL-VERSION=58.0.6.48'  
  
SQL> show collation unicode\_enus\_ci;  
UNICODE\_ENUS\_CI, CHARACTER SET UTF8, FROM EXTERNAL ('UNICODE'), PAD SPACE,  
CASE INSENSITIVE, 'COLL-VERSION=58.0.6.48'

Выше приведённые результаты не являются частью базы данных employee. Некоторые строки в приведенном выше тексте пришлось разделить на две, чтобы они поместились на странице.

### SHOW COMMENTS

Назначение: Отображение всех комментариев, созданных для различных объектов базы данных.

Синтаксис:

SHOW COMMENTs

Команда SHOW COMMENTЫ отображает все комментарии, которые были созданы для различных объектов в текущей базе данных. Нет возможности отобразить конкретный комментарий. Каждый комментарий перечисляется вместе с типом объекта и именем, к которому он был применен.

Пример 4.35. Использование команды SHOW COLLATION

SQL> show comments;  
  
COMMENT ON DATABASE IS This is the demonstration EMPLOYEE database.;  
COMMENT ON TABLE EMPLOYEE IS The EMPLOYEE table has details of our employees.;

Фактический текст комментария отображается между ключевым словом IS и конечной точкой с запятой

### SHOW DATABASE

Назначение: Отображение информации о текущей базе данных.

Синтаксис:

SHOW {DATABASE | DB}

Команда SHOW DATABASE (или SHOW DB) отображает сведения о текущей базе данных (имя файла, владелец, размер и количество выделенных страниц, интервал очистки, счётчики транзакций, флаг Forced Writes, ODS, зашифрована ли база данных, набор символов по умолчанию).

Пример 4.36. Использование команды SHOW DATABASE

SQL> show database;  
  
Database: employee  
 Owner: SYSDBA  
PAGE\_SIZE 8192  
Number of DB pages allocated = 326  
Number of DB pages used = 297  
Number of DB pages free = 29  
Sweep interval = 20000  
Forced Writes are OFF  
Transaction - oldest = 181  
Transaction - oldest active = 214  
Transaction - oldest snapshot = 214  
Transaction - Next = 218  
ODS = 12.0  
Database not encrypted  
Default Character set: NONE

### SHOW DEPENDENCIES

Назначение: Отображение зависимости для указанного объекта базы данных.

Синтаксис:

SHOW DEPENdencies <object\_name>  
  
SHOW DEPENdency <object\_name>

Таблица 4.20. Параметры оператора SHOW DEPENDENCIES

|  |  |
| --- | --- |
| Параметр | Описание |
| *object\_name* | Наименование объекта метаданных для которого отображаются зависимости. |

Команда SHOW DEPENDENCIES отображают все зависимости для указанного имени объекта. Указанное имя объекта может быть таблицей, представлением, хранимой процедурой, функцией, пакетом и т.д.

Выводимый список представляет собой разделенный запятыми список других объектов в базе данных, от которых зависит указанный объект. Другими словами, процедура не сможет скомпилироваться, если, например, будет удалена какая-либо из перечисленных зависимостей.

SQL> show dependencies SET\_CUST\_NO;  
 [SET\_CUST\_NO:Trigger]  
CUSTOMER:Table<-CUST\_NO, CUST\_NO\_GEN:Generator  
+++

Вышеприведенный листинг показывает, что SET\_CUST\_NO является триггером и что он зависит от двух объектов, столбца CUST\_NO таблицы CUSTOMER и последовательности/генератора с именем CUST\_NO\_GEN. Если вы отобразите сам триггер, вы увидите оба упомянутых объекта:

SQL> show trigger set\_cust\_no;  
  
Triggers on Table CUSTOMER:  
SET\_CUST\_NO, Sequence: 0, Type: BEFORE INSERT, Active  
Trigger text:  
=============================================================================  
AS  
BEGIN  
 if (new.cust\_no is null) then  
 new.cust\_no = gen\_id(cust\_no\_gen, 1);  
END  
=============================================================================

Иногда вывод может показаться немного запутанным. Вы можете увидеть различные объекты в списке, которые не кажутся релевантными. Таблица RDB$DEPENDENCIES, также содержит сведения об объектах, от которых будет зависеть данный объект.

### SHOW DOMAIN

Назначение: Отображение списка имён доменов или определение указанного домена.

Синтаксис:

SHOW {DOMAINs | DOMAIN <domain\_name>}

Таблица 4.21. Параметры оператора SHOW DOMAIN

|  |  |
| --- | --- |
| Параметр | Описание |
| *domain\_name* | Наименование домена. |

Команда SHOW DOMAIN отображает информацию о домене. Домен — это определенный пользователем тип данных, глобальный для базы данных.

Команда SHOW DOMAIN имеют две формы. Первая отображает список всех известных доменов в базе данных, а вторая отображает сведения о конкретном домене.

Пример 4.37. Использование команды SHOW DOMAIN

SQL> show domains;  
  
 ADDRESSLINE BUDGET  
 COUNTRYNAME CUSTNO  
 DEPTNO EMPNO  
 FIRSTNAME JOBCODE  
 JOBGRADE LASTNAME  
 PHONENUMBER PONUMBER  
 PRODTYPE PROJNO  
 SALARY  
   
SQL> show domain addressline;  
  
ADDRESSLINE VARCHAR(30) Nullable

### SHOW EXCEPTION

Назначение: Отображение информации обо всех или указанном исключении.

Синтаксис:

SHOW {EXCEPTIONs | EXCEPTION <exception\_name>}

Таблица 4.22. Параметры оператора SHOW EXCEPTION

|  |  |
| --- | --- |
| Параметр | Описание |
| *exception\_name* | Наименование исключения. |

Команда SHOW EXCEPTION отображает все исключения, которые были определены в текущей базе данных. Отображаются также детальная информации об каждом исключениях: сообщение об ошибке, объекты, которые используют исключение. Вы можете отобразить детальную информацию об отдельном исключении, задав в качестве параметра имя исключения.

Пример 4.38. Использование команды SHOW EXCEPTION

SQL> show exceptions;  
  
Exception Name Used by, Type  
=============================== =============================================  
CUSTOMER\_CHECK SHIP\_ORDER, Stored procedure  
Msg: Overdue balance -- can not ship.  
  
CUSTOMER\_ON\_HOLD SHIP\_ORDER, Stored procedure  
Msg: This customer is on hold.  
  
ORDER\_ALREADY\_SHIPPED SHIP\_ORDER, Stored procedure  
Msg: Order status is "shipped."  
  
REASSIGN\_SALES DELETE\_EMPLOYEE, Stored procedure  
Msg: Reassign the sales records before deleting this employee.  
  
UNKNOWN\_EMP\_ID ADD\_EMP\_PROJ, Stored procedure  
Msg: Invalid employee number or project id.  
  
  
  
SQL> show exception customer\_on\_hold;  
  
Exception Name Used by, Type  
=============================== =============================================  
CUSTOMER\_ON\_HOLD SHIP\_ORDER, Stored procedure  
Msg: This customer is on hold.

### SHOW FILTER

Назначение: Отображение списка BLOB-фильтров или информации об указанном фильтре.

Синтаксис:

SHOW {FILTERs | FILTER <filter\_name>}

Таблица 4.23. Параметры оператора SHOW FILTER

|  |  |
| --- | --- |
| Параметр | Описание |
| *filter\_name* | Наименование BLOB-фильтра. |

Команда SHOW FILTER отображает список BLOB-фильтров определённых оператором DECLARE FILTER или информацию об указанном фильтре.

Пример 4.39. Использование команды SHOW FILTER

SQL> show filter;  
  
 FUNNEL  
...  
  
SQL> show filter funnel;   
  
BLOB Filter: FUNNEL  
Input subtype: 2 Output subtype: 1  
Filter library is myfilterlib  
Entry point is blr2asc

### SHOW FUNCTION

Назначение: Отображение информации обо всех определённых пользователем функциях или определение указанной функции.

Синтаксис:

SHOW {FUNCTIONs | FUNCTION <function\_name>}

Таблица 4.24. Параметры оператора SHOW FUNCTION

|  |  |
| --- | --- |
| Параметр | Описание |
| *function\_name* | Наименование функции. |

Команда SHOW FUNCTION отображает все определённые пользователем функции, а так же их зависимости, т.е. объекты которые используются функцией. Если указано имя функции в качестве параметра, то для будет отображена дополнительная информация, которая зависит от типа функции:

* для внешних Legacy функций будет отображена информация о входных параметре и возвращаемом результате, имя динамической библиотеки, имя точки входа;
* для PSQL функций будет отображена информация о входных параметре и возвращаемом результате и тексте функции;
* для внешних UDR функций будет отображена информация о входных параметре и возвращаемом результате, имени используемого плагина и точке входа.

Пример 4.40. Использование команды SHOW FUNCTION

SQL> show functions;  
Global functions  
  
Function Name Invalid Dependency, Type  
================================= ======= =====================================  
F T, Table  
GETEXACTTIMESTAMPUTC  
ISLEAPYEAR  
ISLEAPYEAR\_2  
SUM\_ARGS  
  
  
SQL> show function isleapyear;  
  
External function ISLEAPYEAR:  
Function library is fbudf  
Entry point is isLeapYear  
Returns BY VALUE INTEGER  
Argument 1: TIMESTAMP  
  
  
SQL> show function isleapyear\_2;  
Function text:  
=============================================================================  
begin  
 return mod(extract(year from adate), 4) = 0;  
end  
=============================================================================  
Parameters:  
 OUTPUT BOOLEAN  
ADATE INPUT TIMESTAMP  
  
  
SQL> show function f;  
Function text:  
=============================================================================  
begin  
 return (select count(\*) from t);  
end  
=============================================================================  
Parameters:  
 OUTPUT INTEGER  
   
  
SQL> show function sum\_args;  
External name: myudr!sum\_args  
Engine: UDR  
Parameters:  
 OUTPUT INTEGER  
N1 INPUT INTEGER  
N2 INPUT INTEGER  
N3 INPUT INTEGER

Смотри также: [SHOW PROCEDURE](#isql-commands_show-show_procedure), [SHOW TRIGGER](#isql-commands_show-show_trigger), [SHOW PACKAGE](#isql-commands_show-show_package).

### SHOW GENERATOR

Назначение: Отображение информации обо всех не системных генераторах или об одном указанном генераторе.

Синтаксис:

SHOW {GENERATORs | GENERATOR <generator\_name>}  
   
SHOW {SEQUENCEs | SEQUENCE <sequence\_name>}

Таблица 4.25. Параметры оператора SHOW GENERATOR

|  |  |
| --- | --- |
| Параметр | Описание |
| *generator\_name* | Наименование генератора (последовательности). |

Эти две команды идентичны. Генераторы — это старый термин Firebird для того, что обычно называют последовательностями в других базах данных, а также в SQL стандарте. Вам рекомендуется использовать последовательности, а не генераторы, но Firebird считает их одинаковыми.

Если имя генератора (последовательности) не указано, то будут выведены сведения обо всех генераторах (имя, текущее значение, начальное значение, значение шага приращения). Если указано имя генератора (последовательности), то будет выведены сведения о конкретном генераторе.

Пример 4.41. Использование команды SHOW GENERATOR

SQL> show generators;  
Generator CUST\_NO\_GEN, current value: 1015, initial value: 0, increment: 1  
Generator EMP\_NO\_GEN, current value: 145, initial value: 0, increment: 1  
  
SQL> show generator emp\_no\_gen;  
Generator EMP\_NO\_GEN, current value: 145, initial value: 0, increment: 1

Смотри также: [SHOW SEQUENCE](#isql-commands_show-show_sequence).

### SHOW GRANTS

Назначение: Отображение список всех разрешений в текущей базе данных или разрешений на указанный объект или разрешений для указанной роли.

Синтаксис:

SHOW {GRANTs | GRANT {<object\_name> | <role\_name>}}

Таблица 4.26. Параметры оператора SHOW GRANT

|  |  |
| --- | --- |
| Параметр | Описание |
| *object\_name* | Наименование объекта метаданных для которого отображаются разрешения. |
| *role\_name* | Наименование роли для которой отображается список пользователей которым эта роль назначена. |

Команда SHOW GRANT отображает список всех разрешений в текущей базе данных, если команда используется без параметров. Если в качестве аргумента команды указано имя объекта (таблица, процедура, функция, триггер, и т.д.), то будут отображены разрешения для указанного объекта. Если в качестве аргумента указано имя роли, то будет отображён список пользователей (вместе с оператором разрешения на роль) для которых выдана эта роль.

Пример 4.42. Использование команды SHOW GRANT

SQL> show grants;  
  
/\* Grant permissions for this database \*/  
GRANT DELETE, INSERT, SELECT, UPDATE, REFERENCES ON COUNTRY   
TO PUBLIC WITH GRANT OPTION  
GRANT DELETE, INSERT, SELECT, UPDATE, REFERENCES   
ON CUSTOMER TO PUBLIC WITH GRANT OPTION  
...  
GRANT EXECUTE ON PROCEDURE SHIP\_ORDER TO PUBLIC WITH GRANT OPTION  
GRANT EXECUTE ON PROCEDURE SHOW\_LANGS TO PUBLIC WITH GRANT OPTION  
GRANT EXECUTE ON PROCEDURE SUB\_TOT\_BUDGET TO PUBLIC WITH GRANT OPTION  
  
SQL> show grants employee;  
GRANT DELETE, INSERT, SELECT, UPDATE, REFERENCES   
ON EMPLOYEE TO PUBLIC WITH GRANT OPTION  
  
SQL> show grants ship\_order;  
GRANT EXECUTE ON PROCEDURE SHIP\_ORDER TO PUBLIC WITH GRANT OPTION  
  
SQL> show grants default\_user;  
GRANT DEFAULT\_USER TO SYSDBA

### Примечание

Обратите внимание, что некоторые строки в приведенном выше разделены, чтобы они могли поместиться на странице.

### SHOW INDEXES

Назначение: Отображение информации обо всех индексах или об указанном индексе.

Синтаксис:

SHOW {INDexes | INDICES}  
  
SHOW {INDICES | INDexes} <table\_name>  
   
SHOW INDex <index\_name>

Таблица 4.27. Параметры оператора SHOW INDICES

|  |  |
| --- | --- |
| Параметр | Описание |
| *table\_name* | Наименование таблицы для которой будет отображён список индексов. |
| *index\_name* | Наименование индекса по которому бкдет отображена информации. |

Первая форма этой команды перечислит все индексы в текущей базе данных. В этой форме вы можете использовать либо INDEXES, либо INDECES, они рассматриваются в Firebird как идентичные. Вторая форма команды будет отображать список индексов для указанной таблицы *table\_name*. Опять же, вы можете использовать INDEXES, либо INDECES. В последней форме команды отображаются сведения о указанном индексе *index\_name*, и в этой форме команды необходимо использовать ключевое слово INDEX.

Пример 4.43. Использование команды SHOW INDEX

SQL> show indices;  
RDB$PRIMARY1 UNIQUE INDEX ON COUNTRY(COUNTRY)  
CUSTNAMEX INDEX ON CUSTOMER(CUSTOMER)  
CUSTREGION INDEX ON CUSTOMER(COUNTRY, CITY)  
...  
RDB$PRIMARY24 UNIQUE INDEX ON SALES(PO\_NUMBER)  
SALESTATX INDEX ON SALES(ORDER\_STATUS, PAID)  
  
  
SQL> show indices employee;  
NAMEX INDEX ON EMPLOYEE(LAST\_NAME, FIRST\_NAME)  
RDB$FOREIGN8 INDEX ON EMPLOYEE(DEPT\_NO)  
RDB$FOREIGN9 INDEX ON EMPLOYEE(JOB\_CODE, JOB\_GRADE, JOB\_COUNTRY)  
RDB$PRIMARY7 UNIQUE INDEX ON EMPLOYEE(EMP\_NO)  
  
  
SQL> show index namex;  
NAMEX INDEX ON EMPLOYEE(LAST\_NAME, FIRST\_NAME)

### SHOW MAPPING

Назначение: Информация локальных и глобальных отображений для объектов безопасности.

Синтаксис:

SHOW {MAPping | MAPping <mapping\_name> }

Таблица 4.28. Параметры оператора SHOW MAPPING

|  |  |
| --- | --- |
| Параметр | Описание |
| *mapping\_name* | Имя отображения. |

Команда SHOW MAPPING выводит информацию обо всех отображениях (локальных и глобальных) для объектов безопасности. Если указано имя отображения, то будет выведена информация только он нём.

Пример 4.44. Использование команды SHOW MAPPING

SQL> show mapping;  
WIN\_ADMINS USING PLUGIN WIN\_SSPI FROM PREDEFINED\_GROUP DOMAIN\_ANY\_RID\_ADMINS TO ROLE RDB$ADMIN  
  
\*\*\* Global mapping \*\*\*  
TRUSTED\_AUTH USING PLUGIN WIN\_SSPI FROM ANY USER TO USER  
  
SQL> show mapping win\_admins;  
WIN\_ADMINS USING PLUGIN WIN\_SSPI FROM PREDEFINED\_GROUP DOMAIN\_ANY\_RID\_ADMINS TO ROLE RDB$ADMIN

### SHOW PACKAGE

Назначение: Отображение списка пакетов или информации об указанном пакете.

Синтаксис:

SHOW {PACKages | PACKage <package\_name> }

Таблица 4.29. Параметры оператора SHOW PACKAGE

|  |  |
| --- | --- |
| Параметр | Описание |
| *package\_name* | Имя пакета. |

Команда SHOW PACKAGE отображает список всех пакетов определенных в текущей базе данных. Если в качестве параметра указано имя пакета, то будет отображён исходный код заголовка и тела пакета.

Пример 4.45. Использование команды SHOW PACKAGE

SQL> show packages;  
  
 CHECK\_NAME\_UTILS DATE\_UTILS   
 REPL\_APPLY\_UTILS REPL\_LOG\_UTILS  
 REPL\_TRIGGERS\_UTILS REPL\_WRITE\_LOG  
   
SQL> show package date\_utils;  
  
DATE\_UTILS  
Header source:  
BEGIN  
 ...   
 -- Возвращает строковое представление даты в формате dd MMM yyyy  
 FUNCTION DateToChar\_Rus(ADate DATE) RETURNS VARCHAR(35);  
 -- Возвращает строковое представление даты в формате dd MMM yyyy  
 FUNCTION DateToChar\_En(ADate DATE) RETURNS VARCHAR(35);  
 ...  
END  
  
Body source:  
BEGIN  
 ...  
 /\*\*  
 \* Возвращает строковое представление даты в формате dd MMM yyyy  
 \*/  
 FUNCTION DateToChar\_Rus(ADate DATE) RETURNS VARCHAR(35)  
 AS  
 BEGIN  
 RETURN EXTRACT(DAY FROM ADate) || ' ' ||  
 TRIM (CASE EXTRACT(MONTH FROM ADate)  
 WHEN 1 THEN 'января'  
 WHEN 2 THEN 'февраля'  
 WHEN 3 THEN 'марта'  
 WHEN 4 THEN 'апреля'  
 WHEN 5 THEN 'мая'  
 WHEN 6 THEN 'июня'  
 WHEN 7 THEN 'июля'  
 WHEN 8 THEN 'августа'  
 WHEN 9 THEN 'сентября'  
 WHEN 10 THEN 'октября'  
 WHEN 11 THEN 'ноября'  
 WHEN 12 THEN 'декабря'  
 END) || ' ' ||  
 EXTRACT(YEAR FROM ADate);  
 END  
  
 /\*\*  
 \* Возвращает строковое представление даты в формате dd MMM yyyy  
 \*/  
 FUNCTION DateToChar\_En(ADate DATE) RETURNS VARCHAR(35)  
 AS  
 BEGIN  
 RETURN EXTRACT(DAY FROM ADate) || ' ' ||  
 TRIM (CASE EXTRACT(MONTH FROM ADate)  
 WHEN 1 THEN 'January'  
 WHEN 2 THEN 'February'  
 WHEN 3 THEN 'March'  
 WHEN 4 THEN 'April'  
 WHEN 5 THEN 'May'  
 WHEN 6 THEN 'June'  
 WHEN 7 THEN 'July'  
 WHEN 8 THEN 'August'  
 WHEN 9 THEN 'September'  
 WHEN 10 THEN 'October'  
 WHEN 11 THEN 'November'  
 WHEN 12 THEN 'December'  
 END) || ' ' ||  
 EXTRACT(YEAR FROM ADate);  
 END  
 ...  
END

Смотри также: [SHOW FUNCTION](#isql-commands_show-show_function), [SHOW TRIGGER](#isql-commands_show-show_trigger), [SHOW PROCEDURE](#isql-commands_show-show_procedure).

### SHOW PROCEDURE

Назначение: Отображение информации обо всех процедурах в текущей базе данных или об указанной процедуре.

Синтаксис:

SHOW {PROCedures | PROCedure <procedure\_name> }

Таблица 4.30. Параметры оператора SHOW PROCEDURE

|  |  |
| --- | --- |
| Параметр | Описание |
| *procedure\_name* | Имя хранимой процедуры. |

Команда SHOW PROCEDURE позволяет отобразить список всех хранимых процедур определённых в текущей базе данных и их зависимостей, т.е. объекты которые используются процедурой. Если указано имя хранимой процедуры, то будет отображён её исходный код, а также сведения о входных и выходных параметрах.

Пример 4.46. Использование команды SHOW PROCEDURE

SQL> show procedures;  
Global procedures  
  
Procedure Name Invalid Dependency, Type  
================================= ======= =====================================  
ADD\_EMP\_PROJ EMPLOYEE\_PROJECT, Table  
 UNKNOWN\_EMP\_ID, Exception  
ALL\_LANGS JOB, Table  
 SHOW\_LANGS, Procedure  
...  
  
  
SQL> show procedure all\_langs;  
Procedure text:  
=============================================================================  
BEGIN  
 FOR SELECT job\_code, job\_grade, job\_country FROM job  
 INTO :code, :grade, :country  
  
 DO  
 BEGIN  
 FOR SELECT languages FROM show\_langs  
 (:code, :grade, :country) INTO :lang DO  
 SUSPEND;  
 /\* Put nice separators between rows \*/  
 code = '=====';  
 grade = '=====';  
 country = '===============';  
 lang = '==============';  
 SUSPEND;  
 END  
 END  
=============================================================================  
Parameters:  
CODE OUTPUT VARCHAR(5)  
GRADE OUTPUT VARCHAR(5)  
COUNTRY OUTPUT VARCHAR(15)  
LANG OUTPUT VARCHAR(15)

Смотри также: [SHOW FUNCTION](#isql-commands_show-show_function), [SHOW TRIGGER](#isql-commands_show-show_trigger), [SHOW PACKAGE](#isql-commands_show-show_package).

### SHOW ROLE

Назначение: Отображение списка ролей в текущей базе данных, или списка пользователей назначенных указанной роли.

Синтаксис:

SHOW {ROLEs | ROLE <role\_name> }

Таблица 4.31. Параметры оператора SHOW ROLE

|  |  |
| --- | --- |
| Параметр | Описание |
| *role\_name* | Имя роли. |

Команда SHOW ROLE выводит список ролей в текущей базе данных. Если в качестве аргумента передано имя роли, то будет выведен список пользователей которым назначена указанная роль.

Пример 4.47. Использование команды SHOW ROLE

SQL> show roles;  
 R1 R2  
 R3  
   
SQL> show role r1;  
Role R1 is granted to:  
U1  
U2

### SHOW SECCLASSES

Назначение: Отображение информации о классах безопасности для заданного объекта.

Синтаксис:

SHOW SECCLAsses <object\_name>

Таблица 4.32. Параметры оператора SHOW SECCLASSES

|  |  |
| --- | --- |
| Параметр | Описание |
| *object\_name* | Имя объекта метаданных. |

Команда SHOW SECCLASSES отображает информацию о классах безопасности для заданного объекта. В качестве *object\_name* может быть передано не только имя таблицы, это может быть имя хранимой процедуры, представления и т.д.

Пример 4.48. Использование команды SHOW SECCLASSES

SQL> show secclasses employee;  
  
Table's main sec class SQL$440  
Table's default sec class SQL$DEFAULT54

### SHOW SEQUENCE

Назначение: Отображение информации обо всех не системных последовательностях или об одной указанной последовательности.

Синтаксис:

SHOW {SEQUENCEs | SEQUENCE <sequence\_name>}   
  
SHOW {GENERATORs | GENERATOR <generator\_name>}

Таблица 4.33. Параметры оператора SHOW SEQUENCE

|  |  |
| --- | --- |
| Параметр | Описание |
| *sequence\_name* | Наименование последовательности (генератора). |

Эти две команды идентичны. Генераторы — это старый термин Firebird для того, что обычно называют последовательностями в других базах данных, а также в SQL стандарте. Вам рекомендуется использовать последовательности, а не генераторы, но Firebird считает их одинаковыми.

Если имя последовательности ( генератора) не указано, то будут выведены сведения обо всех последовательностях (имя, текущее значение, начальное значение, значение шага приращения). Если указано имя последовательности (генератора), то будет выведены сведения о конкретном последовательности.

Пример 4.49. Использование команды SHOW SEQUENCE

SQL> show sequences;  
Generator CUST\_NO\_GEN, current value: 1015, initial value: 0, increment: 1  
Generator EMP\_NO\_GEN, current value: 145, initial value: 0, increment: 1  
  
SQL> show sequence emp\_no\_gen;  
Generator EMP\_NO\_GEN, current value: 145, initial value: 0, increment: 1

Смотри также: [SHOW GENERATOR](#isql-commands_show-show_generator).

### SHOW SQL DIALECT

Назначение: Отображение SQL диалекта клиента и текущей базы данных.

Синтаксис:

SHOW SQL DIALECT

Команда SHOW SQL DIALECT отображает SQL диалект установленный в ISQL и диалект подключенной базы данных.

Пример 4.50. Использование команды SHOW SQL DIALECT

SQL> show sql dialect;  
 Client SQL dialect is set to: 3 and database SQL dialect is: 3

### SHOW SYSTEM

Назначение: Отображение списка системных объектов.

Синтаксис:

SHOW SYStem [{TABLEs | FUNCTIONs | COLLATIONs | COLLATEs}]

Команда SHOW SYSTEM выводит список внутренних, то есть системных объектов, созданных и используемых в текущей базе данных. По умолчанию отображаются системные таблицы, функции и сортировки. Необязательный параметр позволяет уточнить какой именно вид системных объектов необходимо отобразить.

Пример 4.51. Использование команды SHOW SYSTEM

SQL> show system;  
  
Tables:  
 MON$ATTACHMENTS MON$CALL\_STACK  
 MON$CONTEXT\_VARIABLES MON$DATABASE  
...  
 SEC$DB\_CREATORS SEC$GLOBAL\_AUTH\_MAPPING  
 SEC$USERS SEC$USER\_ATTRIBUTES  
  
Collations:  
 ASCII BIG\_5  
 BS\_BA CP943C  
 CP943C\_UNICODE CS\_CZ  
 CYRL DA\_DA  
....  
 WIN1257\_EE WIN1257\_LT  
 WIN1257\_LV WIN1258  
 WIN\_CZ WIN\_CZ\_CI\_AI  
 WIN\_PTBR  
   
SQL> show system tables;  
  
 MON$ATTACHMENTS MON$CALL\_STACK  
 MON$CONTEXT\_VARIABLES MON$DATABASE  
...  
 SEC$DB\_CREATORS SEC$GLOBAL\_AUTH\_MAPPING  
 SEC$USERS SEC$USER\_ATTRIBUTES

### Примечание

В Firebird 2.0-2.5 существовали системные функции RDB$GET\_CONTEXT и RDB$SET\_CONTEXT, которые позже стали встроенными. В Firebird 2.5 команда SHOW SYSTEM отобразит следующее:

SQL> show system;  
  
Tables:  
 MON$ATTACHMENTS MON$CALL\_STACK  
 MON$CONTEXT\_VARIABLES MON$DATABASE  
...  
  
Functions:  
 RDB$GET\_CONTEXT RDB$SET\_CONTEXT  
  
Collations:  
 ASCII BIG\_5  
 BS\_BA CP943C  
....  
 WIN\_CZ WIN\_CZ\_CI\_AI  
 WIN\_PTBR

Если вы хотите отобразить детали определенного объекта, просто используйте соответствующую команду show.

SQL> show table mon$io\_stats;  
  
MON$STAT\_ID (RDB$STAT\_ID) INTEGER Nullable  
MON$STAT\_GROUP (RDB$STAT\_GROUP) SMALLINT Nullable  
MON$PAGE\_READS (RDB$COUNTER) BIGINT Nullable  
MON$PAGE\_WRITES (RDB$COUNTER) BIGINT Nullable  
MON$PAGE\_FETCHES (RDB$COUNTER) BIGINT Nullable  
MON$PAGE\_MARKS (RDB$COUNTER) BIGINT Nullable  
  
SQL> show collation ascii;  
  
ASCII, CHARACTER SET ASCII, PAD SPACE, SYSTEM

### SHOW TABLE

Назначение: Отображение списка таблиц или информации о заданной таблице.

Синтаксис:

SHOW {TABLEs | TABLE <table\_name>}

Таблица 4.34. Параметры оператора SHOW TABLE

|  |  |
| --- | --- |
| Параметр | Описание |
| *table\_name* | Имя таблицы. |

В первом варианте команда SHOW TABLES отображает список таблиц, определённых в базе данных, в алфавитном порядке. Во втором варианте отображается список столбцов и их типы или домены для указанной в качестве параметра таблицы.

Пример 4.52. Использование команды SHOW TABLE

SQL> show tables;  
  
 COUNTRY CUSTOMER  
 DEPARTMENT EMPLOYEE  
 EMPLOYEE\_PROJECT JOB  
 PROJECT PROJ\_DEPT\_BUDGET  
 SALARY\_HISTORY SALES  
  
SQL> show table country;  
  
COUNTRY (COUNTRYNAME) VARCHAR(15) Not Null  
CURRENCY VARCHAR(10) Not Null  
CONSTRAINT INTEG\_2:  
 Primary key (COUNTRY)

Если для таблицы определены комментарии, эта команда не будет их отображать. Вы можете использовать команду SHOW COMMENTS, но помните, что тогда вам будут отображены все комментарии в базе данных. По-видимому, не существует метода извлечения комментариев для одного объекта, если только вы не запросите системные таблицы напрямую.

SQL> comment on table country is 'This table holds details about countries.';  
SQL> commit;  
  
SQL> show comments;  
...  
COMMENT ON TABLE COUNTRY IS This table holds details about countries.;  
...  
  
SQL> show table country;  
  
COUNTRY (COUNTRYNAME) VARCHAR(15) Not Null  
CURRENCY VARCHAR(10) Not Null  
CONSTRAINT INTEG\_2:  
 Primary key (COUNTRY)  
   
SQL> select rdb$description  
CON> from rdb$relations  
CON> where rdb$relation\_name = 'COUNTRY';  
  
 RDB$DESCRIPTION  
=================  
 6:1ec  
==============================================================================  
RDB$DESCRIPTION:  
This table holds details about countries.  
==============================================================================

Вывод последнего запроса выше не идеален, но, по крайней мере, он отображает гораздо меньше информации, когда в вашей базе данных много комментариев.

### SHOW TRIGGER

Назначение: Отображение списка триггеров или подробной информации об указанном триггере.

Синтаксис:

SHOW {TRIGgers | TRIGger <trig\_name>}

Таблица 4.35. Параметры оператора SHOW TRIGGER

|  |  |
| --- | --- |
| Параметр | Описание |
| *trig\_name* | Имя триггера. |

Команда SHOW TRIGGER отображает список всех триггеров определённых в текущей базе данных. Для табличных триггеров будет выведено имя таблицы для которого был создан триггер.

Пример 4.53. Использование команды SHOW TRIGGERS

SQL> show triggers;  
Trigger name Invalid  
================================ =======  
TR\_BEFORE\_DISCONNECT  
  
Table name Trigger name Invalid  
================================ ================================ =======  
AGE AGE\_BD  
AGE AGE\_BI  
AGE AGE\_BU  
AIMTYPE TR\_AIMTYPE\_BI  
ALGORITHM TR\_ALGORITHM\_BI  
......

Если указано имя триггера, то будет отображена подробная информация о триггере (имя триггера, порядок срабатывания, событие, активен ли триггер), а также его исходный код.

Пример 4.54. Использование команды SHOW TRIGGER

SQL> show trigger set\_cust\_no;  
  
Triggers on Table CUSTOMER:  
SET\_CUST\_NO, Sequence: 0, Type: BEFORE INSERT, Active  
Trigger text:  
=============================================================================  
AS  
BEGIN  
 if (new.cust\_no is null) then  
 new.cust\_no = gen\_id(cust\_no\_gen, 1);  
END  
=============================================================================

Смотри также: [SHOW FUNCTION](#isql-commands_show-show_function), [SHOW PROCEDURE](#isql-commands_show-show_procedure), [SHOW PACKAGE](#isql-commands_show-show_package).

### SHOW USERS

Назначение: Отображение списка пользователей в текущей БД безопасности.

Синтаксис:

SHOW USERs

Команда SHOW USERS отображает список пользователей определённых в текущей базе данных безопасности. Если пользователь не является администратором в базе данных безопасности, то будет отображены только сведения о нём самом. Перед каждым именем пользователя указано число, которое отображает количество соединений выполненное этим пользователем. Если перед именем пользователя напечатана решётка «#», то это текущий пользователь.

Пример 4.55. Использование команды SHOW USERS

SQL> show users;  
Users in the database  
 1 #SYSDBA 2 U1  
 0 U3

В данном случае есть одно соединение пользователем SYSDBA — это текущий пользователь, и два соединения пользователем U1.

### SHOW VERSION

Назначение: Отображение информации о версии ISQL, сервера и ODS базы данных.

Синтаксис:

SHOW VERsion

Команда SHOW VERSION отображает информацию о версии ISQL, версии сервера Firebird, сетевом протоколе, ODS текущей базы данных.

Пример 4.56. Использование команды SHOW VERSION

SQL> show version;  
  
ISQL Version: WI-V3.0.5.33168 Firebird 3.0  
Server version:  
Firebird/Windows/AMD/Intel/x64 (access method),   
version "WI-V3.0.5.33168 Firebird 3.0"  
Firebird/Windows/AMD/Intel/x64 (remote server),   
version "WI-V3.0.5.33168 Firebird 3.0/tcp (DESKTOP-E3INAFT)/P15:C"  
Firebird/Windows/AMD/Intel/x64 (remote interface),   
version "WI-V3.0.5.33168 Firebird 3.0/tcp (DESKTOP-E3INAFT)/P15:C"  
on disk structure version 12.0

### Примечание

Обратите внимание, что некоторые строки в приведенном выше разделены, чтобы они могли поместиться на странице.

### SHOW VIEW

Назначение: Отображение списка представлений или информации об указанном представлении.

Синтаксис:

SHOW {VIEWs | VIEW <view\_name>}

Таблица 4.36. Параметры оператора SHOW VIEW

|  |  |
| --- | --- |
| Параметр | Описание |
| *view\_name* | Имя представления. |

Команда SHOW VIEW отображает список представлений определённых в текущей базе данных. Если в качестве аргумента указано имя представления, то будет отображён список его столбцов с типами или доменами, и исходный код представления.

Пример 4.57. Использование команды SHOW VIEW

SQL> show views;  
  
 PHONE\_LIST  
   
SQL> show view phone\_list;  
  
EMP\_NO (EMPNO) SMALLINT Not Null  
FIRST\_NAME (FIRSTNAME) VARCHAR(15) Not Null  
LAST\_NAME (LASTNAME) VARCHAR(20) Not Null  
PHONE\_EXT VARCHAR(4) Nullable  
LOCATION VARCHAR(15) Nullable  
PHONE\_NO (PHONENUMBER) VARCHAR(20) Nullable  
View Source:  
==== ======  
SELECT  
 emp\_no, first\_name, last\_name, phone\_ext, location, phone\_no  
 FROM employee, department  
 WHERE employee.dept\_no = department.dept\_no

## Глава 5. GFIX

## Глава 6. NBACKUP

## Глава 7. FB\_LOCK\_PRINT

## Глава 8. FBTRACEMGR

## Глава 9. GSEC

## Глава 10. GSPLIT

## Глава 11. Типичные проблемы с установкой

## На Windows

## На Linux

## Глава 12. Разрешение проблем с сетевыми настройками между клиентом и сервером Firebird

## Глава 13. Настройка сервера и клиента

Для настройки сервера Firebird используется файл firebird.conf. Настройки считываются из файла один раз при старте сервера, если архитектура SuperSever или SuperClassic, и при каждом соединении с базой, если архитектура сервера Classic.

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

#комментарий  
DefaultDbCachePages = 2048 #комментарий

Максимальная длина строки в файле конфигурации сервера равна 80 символов.

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

## Типы параметров

В файле конфигурации присутствуют параметры трех типов:

* целочисленный (Integer);
* строковый (String);
* логический (Boolean).

### Целочисленные параметры

Целочисленные как ясно из названия могут содержать только целые числа. Они могут быть записаны со множителями k, m и g.

Пример 13.1. Целочисленные параметры

MaxUnflushedWrites = 100   
DefaultDbCachePages = 8K # 8 \* 1024 = 8192  
DatabaseGrowthIncrement = 128M # 128 \* 1024 \* 1024

### Примечание

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

### Логические параметры

Логическое значение выражается в виде целочисленных значений, где 0 (ноль) означает false, а ненулевое значение означает true. Для согласованности мы рекомендуем использовать только 0/1. Также строки 'y', 'yes' и 'true' означают true, а 'n', 'no' и 'false' — false.

Пример 13.2. Логические параметры

RemoteAccess = true # true  
RemoteFileOpenAbility = 0 # false

### Строковые параметры

Строковые как ясно из названия представляют собой строки.

Пример 13.3. Строковые параметры

RemoteServiceName = gds\_db  
RemotePipeName = pipe47

## Область действия

Параметры конфигурации могут быть применены к экземпляру сервера (Per-server), отдельной базе данных (Per-database) или отдельному соединению (Per-connection).

Параметры которые могут быть применены к отдельной базе данных можно указывать в файле databases.conf. В этом случае они перекроют значение одноимённого параметра указанного в firebird.conf.

Параметры, которые могут быть применены к отдельному соединению, предназначены для использования на стороне клиента. Они устанавливаются с помощью *isc\_dpb\_config* параметра в DPB (Database parameters buffer) (*isc\_spb\_config* для служб).

### Примечание

Параметры уровня базы данных могут быть установлены используя DPB при первом соединении с базой данных, если используется встроенный режим сервера.

Параметры, которые могут быть применены к отдельному соединению, могут быть прочитаны из файлов firebird.conf и databases.conf расположенных в том же каталоге что и клиентская библиотека fbclient. Эти параметры будут перекрыты установками через DPB.

## Макроподстановки

Существует ряд предопределенных макроподстановок, которые могут быть использованы в файлах конфигурации, где требуется имя каталога. Они используют следующий синтаксис:

$(<name>)

Полный список таких макроподстановок представлен ниже:

* root — корневой каталог экземпляра сервера;
* install — директория в которую установлен Firebird. Изначально $(root) и $(install) одинаковые. $(root) может быть переопредедна установкой или изменением переменной окружения FIREBIRD. В таком случае эта подстановка отлична от $(install);
* this — каталог, в котором находится текущий файл конфигурации;
* dir\_conf — каталог, в котором расположены файлы конфигурации firebird.conf и databases.conf;
* dir\_secDb — директория, в которой расположены база данных безопасности по умолчанию;
* dir\_plugins — директория, в которой расположены плагины;
* dir\_udf — директория, предназначенная для размещения UDF функций по-умолчанию;
* dir\_sample — каталог с примерами (examples);
* dir\_sampleDb — директория, где лежит пример базы данных (examples/empbuild);
* dir\_intl — директория, в которой расположены библиотеки интернационализации;
* dir\_msg — каталог, в котором расположен файл с сообщениями сервера firebird.msg. Обычно он совпадает с $(root), но может быть переопределён переменной окружения FIREBIRD\_MSG.

### Примечание

Внутренние макроподстановки не чувствительны к регистру. Большие буквы используются исключительно для повышения читабильности.

Пример 13.4. Пример использования макроподстановки в файле databases.conf

employee = $(dir\_sampleDb)/employee.fdb

## Включение других файлов

Один файл конфигурации 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 расположенный по пути /opt/config/some\_file.conf.

Директива include поддерживает групповые символы \* и ?. Все совпадающие с шаблоном файлы будут подключены, порядок включения не определён.

Пример 13.5. Использование групповых символов с директивой include

include $(dir\_plugins)/config/\*.conf

## Параметры уровня базы данных

Настройка конфигурации на уровне базы данных осуществляется с помощью формальных записей в файле databases.conf. Такие параметры помечены как Per-database.

### Формат конфигурационных записей

Если вы не добавляете каких либо специфичных для базы данных директив конфигурации для псевдонима, то формат будет такой же как он был в aliases.conf (Firebird 2.5 и ранее):

emp = c:\Program Files\examples\empbuild\employee.fdb  
# или  
emp = /opt/firebird/examples/empbuild/employee.fdb  
# или  
emp = $(dir\_sampleDb)/employee.fdb

Несколько более сложный формат используется для случаев, когда определенные неглобальные параметры должны быть нацелены на отдельные базы данных. Запись для базы данных определяется объявлением псевдонима, как и ранее. Директивы, относящиеся к базе данных, перечислены ниже в фигурных скобках.

# Directives for MYBIGDB  
MYBIGDB = opt/databases/mybigdb.fdb  
{  
 LockMemSize = 32M # We know that MYBIGDB needs a lot of locks  
 LockHashSlots = 19927 # and a hash table large enough for them  
}

### Доступные параметры

Следующие параметры можно скопировать/вставить в файл database.conf и использовать в качестве переопределений для конкретных баз данных.

Таблица 13.1. Параметры доступные в databases.conf

**Связанные с ядром сервера**

[DatabaseGrowthIncrement](#config-param-DatabaseGrowthIncrement)

[DeadlockTimeout](#config-param-DeadlockTimeout)

[DefaultDbCachePages](#config-param-DefaultDbCachePages)

[EventMemSize](#config-param-EventMemSize)

[FileSystemCacheThreshold](#config-param-FileSystemCacheThreshold)

[ExternalFileAccess](#config-param-ExternalFileAccess)

[GCPolicy](#config-param-GCPolicy)

[LockAcquireSpins](#config-param-LockAcquireSpins)

[LockHashSlots](#config-param-LockHashSlots)

[LockMemSize](#config-param-LockMemSize)

[MaxUnflushedWrites](#config-param-MaxUnflushedWrites)

[MaxUnflushedWriteTime](#config-param-MaxUnflushedWriteTime)

[SecurityDatabase](#config-param-SecurityDatabase)

[UserManager](#config-param-UserManager)

[WireCompression](#config-param-WireCompression)

[WireCrypt](#config-param-WireCrypt)

[WireCryptPlugin](#config-param-WireCryptPlugin)

**Связанные с клиентом**

Некоторые параметры могут быть настроены в клиентском соединении через DPB/SPB, в качестве альтернативы их настройке в database.conf. Подробнее смотри в [Область действия](#config-scope).

[AuthClient](#config-param-AuthServer)

[Providers](#config-param-Providers)

Следующие параметры могут быть настроены на стороне клиента только через DPB/SPB.

[ConnectionTimeout](#config-param-ConnectionTimeout)

[DummyPacketInterval](#config-param-DummyPacketInterval)

[IpcName](#config-param-IpcName)

[RemoteAuxPort](#config-param-RemoteAuxPort)

[RemotePipeName](#config-param-RemotePipeName)

[RemoteServiceName](#config-param-RemoteServicePort)

[RemoteServicePort](#config-param-RemoteServicePort)

[TCPNoNagle](#config-param-TcpNoNagle)

## Общие настройки

### DatabaseAccess

Область действия: Per-server.

Тип параметра: String (специальный формат).

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

Параметр *DatabaseAccess* имеет строковый тип; по умолчанию значение параметра равно Full — полный доступ. Для ограничения доступа используется параметр Restrict. В этом случае после слова Restrict указываются директории, в которых могут быть сохранены файлы базы данных.

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

Если параметр *DatabaseAccess* установлен в значение None, то позволено соединяться только с базами данных, перечисленными в файле databases.conf.

Пример 13.6. Значения параметра DatabaseAccess

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

### Предупреждение

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

### RemoteAccess

Область действия: Per-database.

Тип параметра: Boolean.

Параметр предоставляет или отменяет удалённый 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 безопасности. то рекомендуем отключить удалённый доступ к ним в файле databases.conf.

Для повышения безопасности можно отключить *RemoteAccess* глобально в файле firebird.conf и включить его для отдельных баз данных в databases.conf.

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

### ExternalFileAccess

Область действия: Per-database.

Тип параметра: Строка (специальный формат).

Параметр *ExternalFileAccess* позволяет управлять правами размещения файлов внешних таблиц. Разрешение на доступ к внешним файлам может быть полным (Full), ограниченным (Restrict) или запрещённым (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 Firebird. В качестве разделителя директорий используется символ «;».

Пример 13.7. Примеры задания значения параметра ExternalFileAccess

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

### Предупреждение

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

### UdfAccess

Область действия: Per-server.

Тип параметра: String (специальный формат).

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 запрещённым (None).

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

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

UdfAccess = Restrict UDF

### Предупреждение

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.

### Примечание

Этот параметр ни как не влияет на правила размещения нового типа внешних библиотек UDR.

### TempDirectories

Область действия: Per-server.

Тип параметра: String (специальный формат).

С помощью параметра *TempDirectories* можно задать временный каталог сервера Firebird. Временный каталог необходим для выгрузки данных во время сортировки (и в некоторых других случаях), когда исчерпывается выделенная оперативная память.

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

В качестве значения параметра может быть задан путь к одному или нескольким каталогам. В этом случае выгрузка временных данных при сортировке будет осуществляться в указанные каталоги. Для папок допускаются как абсолютные, так и относительные пути. Относительные пути берутся от корневого каталога инсталляции сервера Firebird. Если требуется определить несколько временных каталогов, то в качестве разделителя используется символ «;».

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

Пример 13.8. Примеры задания значения параметра TempDirectories

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

### AuditTraceConfigFile

Область действия: Per-server.

Тип параметра: String.

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

Указанный файл конфигурации должен иметь ту же структуру, что и fbtrace.conf.

Пример 13.9. Примеры задания значения параметра AuditTraceConfigFile

AuditTraceConfigFile = fbtrace.conf

### MaxUserTraceLogSize

Область действия: Per-server.

Тип параметра: Integer.

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.

Пример 13.10. Примеры задания значения параметра MaxUserTraceLogSize

MaxUserTraceLogSize = 10

### DefaultDbCachePages

Область действия: Per-database.

Тип параметра: Integer.

Параметр *DefaultDbCachePages* используется для настройки количества страниц одной базы данных, находящихся в кэш-памяти одновременно. Суперсервер использует единый кэш (2048 страниц по умолчанию) для всех подключений. Классик и суперклассик создает отдельный кэш (по умолчанию 256 страниц) для каждого соединения.

Параметр имеет целочисленный тип. Допустимо использовать множители (k, m, g). Максимальное значение 2147483647 страниц.

Этот параметр может быть установлен для каждой базы данных индивидуально в файле databases.conf. Это особенно удобно если на вашем сервере находятся несколько разных баз данных под управлением одного экземпляра Firebird.

Размер страничного кэша может быть установлен на странице заголовков файла базы данных с помощью утилиты gfix.

gfix -b[uffers] <page-count> <database>

В этом случае размер страничного кэша устанавливается для каждой базы одновременно. На наш взгляд предпочтительно делать это в файле databases.conf, где вы можете установить и другие параметры индивидуальные для вашей базы данных.

Обратите внимание, размер страничного кеша установленный на уровне заголовка базы данных перекрывает значение установленное в firebird.conf и databases.conf.

### Подсказка

Чтобы проверить установлен ли размер страничного кэша на уровне заголовка базы данных вы можете воспользоваться утилитой gstat

d:\fb\fb30>gstat -u sysdba -p masterkey -h testdb

Database "d:\fb\fb30\data\testdb.fdb"  
Gstat execution time Wed Jul 17 12:59:26 2019  
  
Database header page information:  
 Flags 0  
 Generation 2180  
 System Change Number 0  
 Page size 16384  
 ODS version 12.0  
 Oldest transaction 2037  
 Oldest active 2038  
 Oldest snapshot 2038  
 Next transaction 2038  
 Sequence number 0  
 Next attachment ID 391  
 Implementation HW=AMD/Intel/x64 little-endian OS=Windows CC=MSV  
C  
 Shadow count 0  
 Page buffers 0  
 Next header page 0  
 Database dialect 1  
 Creation date Jun 28, 2016 18:04:35  
 Attributes force write  
  
 Variable header data:  
 Sweep interval: 20000  
 \*END\*  
Gstat completion time Wed Jul 17 12:59:26 2019

Page buffers — размер кэша базы данных. Если это значение равно 0, то будет использоваться значние параметра *DefaultDbCachePages* в firebird.conf или databases.conf.

При изменении данного параметра стоит учитывать архитектуру сервера, максимальное количество соединений с базой данных (Classic, SuperClassic), количество доступной оперативной памяти, а так же некоторые другие настройки сервера.

Пример 13.11. Установка параметра DefaultDbCachePages в firebird.conf

DefaultDbCachePages = 2048

Пример 13.12. Установка параметра DefaultDbCachePages в databases.conf для базы данных testdb

testdb = d:\fb\fb30\data\testdb.fdb  
{  
 DefaultDbCachePages = 8K  
}

Смотри также: [FileSystemCacheThreshold](#config-param-FileSystemCacheThreshold)

### DatabaseGrowthIncrement

Область действия: Per-database.

Тип параметра: Integer.

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

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

Пример 13.13. Установка параметра DatabaseGrowthIncrement

DatabaseGrowthIncrement = 128M

### Примечание

Пространство под теневые копии баз данных не резервируется.

### FileSystemCacheThreshold

Область действия: Per-database.

Тип параметра: Integer.

Параметр *FileSystemCacheThreshold* устанавливает порог использования системного файлового кэша сервером Firebird. Системный файловый кэш используется, если размер страничного кэша (установленного явно в заголовке базы данных или через параметр конфигурации *DefaultDbCachePages*) меньше чем значение *FileSystemCacheThreshold*.

Параметр имеет целочисленный тип. Единица измерения – страница базы данных. 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 файловый кэш.

Пример 13.14. Установка параметра FileSystemCacheThreshold

FileSystemCacheThreshold = 128K

### Примечание

Системный файловый кэш имеет смысл отключать только при достаточно большом размере страничного кэша базы данных. Это позволяет избегать двойного кэширования и снизить потребление оперативной памяти. Однако стоит учитывать, что отключение системного файлового кэша может снизить производительность при работе с базой данных, особенно на непрогретом страничном кэше (собственный страничный кэш пока не умеет делать упреждающего чтения, поэтому наполняется существенно медленней файлового кэша системы).

Смотри также: [DefaultDbCachePages](#config-param-DefaultDbCachePages)

### FileSystemCacheSize

Область действия: Per-server.

Тип параметра: Integer, измеряется в процентах от объёма доступной оперативной памяти.

Параметр *FileSystemCacheSize* устанавливает максимальный размер оперативной памяти, используемый системным файловым кешем 64-битными Windows XP или Windows Server 2003 с Service Pack 1 или выше. Этот параметр не оказывает никакого эффекта в Unix-подобных операционных системах.

Параметр содержит целое число, представляющее собой количество (в процентах) оперативной памяти, которое может быть использовано под файловый кеш. Значение может быть от 10 до 95%.

Значение считывается при старте сервера и не может быть изменено. Требуется перезагрузка операционной системы для вступления изменений в силу.

Если задать значение 0, операционная система сама будет определять размер файлового кеша (обычно это 30%). Это и есть значение по умолчанию.

Пример 13.15. Установка параметра FileSystemCacheSize

FileSystemCacheSize = 25

### Примечание

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

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

### RemoteFileOpenAbility

Область действия: Per-server.

Тип параметра: Boolean.

Параметр *RemoteFileOpenAbility* отключает защиту от открытия баз данных на смонтированных томах NFS в Linux/Unix и SMB/CIFS в Windows.

### Предупреждение

Эта опция удаляет важную функцию безопасности Firebird и может привести к неисправимому повреждению базы данных. Не используйте эту опцию, если вы не понимаете риски и не готовы принять потерю содержимого вашей базы данных.

Параметр имеет логический тип. По умолчанию его значение равно 0. В этом случае Firebird может открыть базу данных, только если она хранится на диске, физически подключенном к локальному компьютеру, на котором установлена ​​эта копия Firebird. Запросы на подключение к базам данных, хранящимся на смонтированных дисках NFS, перенаправляются на сервер Firebird, работающий на компьютере, который «владеет» диском.

Пример 13.16. Установка параметра RemoteFileOpenAbility

RemoteFileOpenAbility = 0

Это ограничение не позволяет двум разным копиям Firebird открывать одну и ту же базу данных без координации их действий. Несогласованный доступ нескольких копий Firebird приведет к повреждению базы данных. В локальной системе блокировка файлов на уровне системы предотвращает несогласованный доступ к файлу базы данных.

NFS не обеспечивает надежный способ обнаружения доступа нескольких пользователей к файлу на смонтированном диске NFS. Если вторая копия Firebird подключится к базе данных на смонтированном диске NFS, она повредит базу данных. В некоторых случаях запуск сервера Firebird на компьютере, который владеет томами, смонтированными в NFS, неудобен или невозможен. Приложения, которые используют embedded вариант Firebird и никогда не разделяющие доступ к базе данных, могут использовать эту опцию, чтобы разрешить прямой доступ к базам данных на смонтированных томах NFS.

Ситуация для SMB / CIFS очень похожа на NFS, поскольку не все конфигурации предоставляют механизмы блокировки файлов, необходимые для безопасной работы. Использование механизма SuperServer с базой данных на файловом сервере NT может считаться относительно безопасным, поскольку блокировка файлов защищает базу данных от использования несколькими механизмами. Сетевой стек по-прежнему может изменять порядок записи, поэтому вы можете получить поврежденную базу данных в случае сетевых ошибок или отключения питания.

Относительно полезный и безопасный случай — это работа с общей базой данных, помеченной только для чтения.

### TempBlockSize

Область действия: Per-server.

Тип параметра: Integer.

Параметр *TempBlockSize* используется для управления временным пространством. Временное хранилище используются большими сортировками, или для промежуточного хранения набора данных. Параметр *TempBlockSize* определяет минимальный размер блока, выделяемого для временного хранилища. Это значение отражает гранулярность выделения пространства.

Параметр имеет целочисленный тип. Единица измерения — байты. По умолчанию параметр имеет значение 1 Мбайт. Максимально допустимое значение 2 Гбайт. Минимальное значение параметра — 0.

Пример 13.17. Установка параметра TempBlockSize

TempBlockSize = 2M

Смотри также: [TempCacheLimit](#config-param-TempCacheLimit)

### TempCacheLimit

Область действия: Per-server.

Тип параметра: Integer.

Параметр *TempCacheLimit* определяет максимальный объём оперативной памяти, который используется для кэширования временного пространства.

Параметр имеет целочисленный тип. Единица измерения — байты. Минимальное значение равно 0, максимальное — (264 - 1) байт. Значение по умолчанию для классического сервера равно 8 Мбайт, для архитектур SuperClassic и SuperServer — 64 Мбайт.

### Важно

Для классического сервера это ограничение распространяется для каждого соединения с базой данных, для SuperServer и SuperClassic — для каждого запущенного экземпляра сервера Firebird.

Пример 13.18. Установка параметра TempCacheLimit

TempCacheLimit = 256M

Смотри также: [TempBlockSize](#config-param-TempBlockSize)

### AuthServer и AuthClient

Область действия: Per-database.

Тип параметра: String.

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.

Firebird поддерживает следующие методы аутентификации:

* Безопасная парольная аутентификация (Srp);
* Традиционная (Legacy\_Auth) аутентификация;
* Доверительная (Win\_Sspi) аутентификация для ОС Windows.

По умолчанию на стороне сервера используется метод Secure remote passwords (Srp256), представленный соответствующей ОС плагином (libSrp.so | Srp.dll | Srp.dylib).

AuthServer = Srp256

На клиентской стороне по умолчанию используется следующий список методов аутентификации:

AuthClient = Srp256, Srp, Win\_Sspi, Legacy\_Auth # для Windows клиентов   
AuthClient = Srp256, Srp, Legacy\_Auth # для не Windows клиентов

### Примечание

Первоначально Firebird 3.0 использовал по умолчанию плагин Srp, который использовал SHA-1 для генерации ключа шифрования, начиная с Firebird 3.0.4 введён усовершенствованный метод аутентификации Srp256, который использует SHA-256.

Если вы хотите использовать плагины аутентификации, которые не предоставляют ключа шифрования (Win\_Sspi, Legacy\_Auth), то следует отключить обязательное (Required) шифрование каналов передачи данных (параметр *WireCrypt*), кроме случаев, когда вы работаете с протоколом XNET, который никогда не использует шифрование.

### Примечание

Традиционная (Legacy\_Auth) не умеет работать с новым менеджером пользователей Srp. Поэтому если вы хотите соединяться с Firebird 3.0 клиентами fbclient версией 2.5 и ниже, то необходимо также включить в список плагинов управления пользователями (параметр UserManager) плагин Legacy\_UserManager.

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

### Подсказка

Поскольку плагины аутентификации пробуются в порядке их перечисления, то для наилучшей производительности переместите наиболее часто используемый метод аутентификации на первое место.

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

Смотри также: [UserManager](#config-param-UserManager), [WireCrypt](#config-param-WireCrypt).

### UserManager

Область действия: Per-database.

Тип параметра: String.

Устанавливает плагин, который будет работать в базе данных безопасности. Это может быть список с пробелами, запятыми или точками с запятой в качестве разделителей. По-умолчанию используется первый подключаемый плагин из списка. Firebird поддерживает следующие плагины управления пользователями:

* Srp;
* Legacy\_UserManager.

Для поддержки старой базы данных безопасности и управления пользователями в ней, следует установить значение параметра Legacy\_UserManager.

Пример 13.19. Установка параметра UserManager

UserManager = Srp

В SQL операторах управления пользователями можно явно указать какой плагин будет использоваться с помощью ключевого слова USING.

### Примечание

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.

### TracePlugin

Область действия: Per-server.

Тип параметра: String.

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

Пример 13.20. Установка параметра TracePlugin

TracePlugin = fbtrace

### WireCryptPlugin

Область действия: Per-connection.

Тип параметра: String.

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

По-умолчанию устанавливается значение параметра Arc4, что означает использование плагина потокового шифра Alleged RC4. Сконфигурированный плагин, который требует ключ, сгенерированный настроенным подключаемым модулем аутентификации, может быть переопределен в API для конкретного соединения через DPB или SPB.

Пример 13.21. Установка параметра WireCryptPlugin

WireCryptPlugin = Arc4

Смотри также: [WireCrypt](#config-param-WireCrypt).

### KeyHolderPlugin

Область действия: Per-server.

Тип параметра: String.

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

Реализованного плагина по-умолчанию нет, но пример для Linux под названием libCryptKeyHolder\_example.so можно найти в папке /plugins/.

### Providers

Область действия: Per-database и per-connection.

Тип параметра: String.

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

В firebird.conf доступны по-умолчанию следующие провайдеры:

Providers = Remote,Engine12,Loopback

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

Архитектура провайдеров (известная как Open Systems Relational Interface, OSRI) очень эффективна для поддержки сочетания старых и новых форматов базы данных (с разными ODS) на одном сервере, имеющих смешанные подключения к локальным и удаленным базам данных.

Провайдеры реализованные в Firebird, позволяют поддерживать все эти режимы (удаленные соединения, базы данных с разными ODS), а также цепочки провайдеров. Цепочка — это термин для ситуации, когда провайдер использует обратный вызов стандартного API для выполнения операции над базой данных.

Главным элементом архитектуры провайдеров является **y-valve**. На начальном этапе соединения с базой данных или её создания **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. Конфигурация по-умолчанию содержит три провайдера:

* **Remote** (используется для сетевого соединения);
* **Engine12** (ядро для работы с ODS 12);
* **Loopback** (принудительное соединение с локальным сервером, когда задано только имя базы данных без явного указания протокола).

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

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

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

* Если файл базы данных, к которому нужно подключиться, не существует, то это не интересно всем.
* 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.

### Примечание

* Это также может происходить, если выбран режим SuperServer и уже существует соединение с базой данных, поскольку SuperServer требует эксклюзивного доступа к файлу базы данных.
* После отказа провайдера Engine12 в получении доступа к базе данных, пытается подключиться провайдер Loopback. Он не очень отличается от Remote, за исключением того, что он пытается получить доступ к именованной базе данных *<dbname>* на сервере c сетевым интерфейсом «внутренней петли» (loopback) в сетевом протоколе TCP/IP.
* В Windows XNET пробуется первым, затем TCP/IP loopback (localhost:*<dbname>*), затем Named Pipes (NetBEUI) loopback. Сервер может быть запущен с отключенным XNET (или любым другим протоколом), поэтому перебираются все варианты. В POSIX поддерживается только TCP/IP протокол, остальные варианты не доступны.

### Примечание

Если вы хотите добиться поведения, аналогичного Firebird 2.5 и ниже, когда при наличии запущенного сервера и указании в строке соединения только алиаса базы данных или пути до неё, устанавливалось соединение по локальному проколу, то необходимо поменять провайдеры Engine12 и Loopback местами.

Для подключения по локальному протоколу рекомендуем использовать URI-подобную строку подключения с явным указанием префикса протокола xnet://.

Провайдеры не ограничены тремя вышеперечисленными. Версия 3.0 не поддерживает pre- ODS 12 провайдер. Тем не менее, архитектура провайдеров делает возможным доступ к старым базам данных при переходе на более высокую версию Firebird.

### Примечание

Firebird 4.0 для работы с ODS 13 реализован провайдер Engine13, тем не менее вы можете работать с базой данных ODS 12, через провайдер Engine12 из Firebird 3.0.

### DeadlockTimeout

Область действия: Per-database.

Тип параметра: Integer.

Параметр *DeadlockTimeout* определяет количество секунд, в течение которых диспетчер блокировок будет ожидать после возникновения конфликта, прежде чем очистить блокировки от мертвых процессов и выполнить дополнительный цикл сканирования взаимоблокировок. Firebird обнаруживает взаимные блокировки мгновенно во всех обычных случаях, поэтому это значение влияет на ситуации, когда что-то идет не так.

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

Пример 13.22. Установка параметра DeadlockTimeout

DeadlockTimeout = 10

### Предупреждение

Слишком низкое значение может снизить производительность системы.

### MaxUnflushedWrites

Область действия: Per-database.

Тип параметра: Integer.

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 накопленных страниц не сброшенных на диск, ожидающих сброса при подтверждения транзакции.

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

Пример 13.23. Установка параметра MaxUnflushedWrites

MaxUnflushedWrites = 100

### Предупреждение

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

Смотри также: [MaxUnflushedWriteTime](#config-param-MaxUnflushedWriteTime)

### MaxUnflushedWriteTime

Область действия: Per-database.

Тип параметра: Integer.

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 сброса на диск при подтверждении транзакции, будут выгружены на диск.

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

Пример 13.24. Установка параметра MaxUnflushedWriteTime

MaxUnflushedWriteTime = 5

### Предупреждение

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

Смотри также: [MaxUnflushedWrites](#config-param-MaxUnflushedWrites)

### BugcheckAbort

Область действия: Per-server.

Тип параметра: Boolean.

Опция *BugcheckAbort* определяет, прерывать ли работу сервера при возникновении внутренней ошибки или снимать дамп ядра для последующего анализа. Если опция отключена, то ядро пытается минимизировать ущерб и продолжить работу.

Параметр имеет логический тип. Возможные значения 0 и 1. Значение по умолчанию равно 0, в этом случае механизм снятия дампов отключен. Для отладочных сборок (DEV\_BUILD) значение по умолчанию равно 1.

Пример 13.25. Установка параметра BugcheckAbort

BugcheckAbort = 0

### Примечание

Обратите внимание, что установка этой опции в 1 заставляет движок производить трассируемые дампы, когда внутри UDF происходит что-то неприятное, например SIGSEGV. В Windows включение этой опции заставляет ядро вызывать средства отладки JIT при возникновении ошибок.

### RelaxedAliasChecking

Область действия: Per-server.

Тип параметра: Boolean.

Параметр *RelaxedAliasChecking* позволяет снять ограничение на обязательное использование псевдонимов имен таблиц в запросах. Если опция включена, то Firebird позволяет выполнять подобные запросы:

SELECT MY\_TABLE.X FROM MY\_TABLE A

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

Пример 13.26. Установка параметра RelaxedAliasChecking

RelaxedAliasChecking = 0

Не рекомендуется включать этот параметр. Его следует рассматривать как временный обходной путь для портирования неаккуратного устаревшего кода до тех пор, пока не удастся пересмотреть такой код.

### Внимание

Нет никаких гарантий что данная установка будет доступна в следующих версиях Firebird.

Пример 13.27. Установка параметра RelaxedAliasChecking

RelaxedAliasChecking = 0

### ConnectionTimeout

Область действия: Per-connection.

Тип параметра: Integer.

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.

Пример 13.28. Установка параметра ConnectionTimeout

ConnectionTimeout = 180

### WireCrypt

Область действия: Per-connection.

Тип параметра: String (предопределённые значения).

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 клиента.

Пример 13.29. Установка параметра WireCrypt

WireCrypt = Enabled # для клиента

WireCrypt = Required # для сервера

Чтобы получить доступ к серверу с использованием традиционной (Legacy\_Auth) аутентификации, 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 *WireCryptPlugin* 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 13.2. 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. |

Смотри также: [AuthServer и AuthClient](#config-param-AuthServer), [WireCryptPlugin](#config-param-WireCryptPlugin).

### WireCompression

Область действия: Per-connection.

Тип параметра: Boolean.

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

По-умолчанию сжатие отключено.

Пример 13.30. Установка параметра WireCompression

WireCompression = false

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

* Чтобы включить *WireCompression* на стороне сервера, поставьте параметр в значение true в файле firebird.conf или database.conf.
* Для того, чтобы активировать *WireCompression* на стороне клиента, передайте соответствующий тег в вызов DPB и SPB:
* isc\_dbp\_config <string-length> "WireCompression=true"  
  isc\_sbp\_config <string-length> "WireCompression=true"
* Клиент и сервер должны использовать протокол >= 13 (Firebird 3.0 и старше).

### DummyPacketInterval

Область действия: Per-connection.

Тип параметра: Integer.

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.

### Предупреждение

Эта опция может привести к зависанию или сбоям Windows NT4 или Windows 2000 pre SP3 на стороне клиента, как описано в [296265](https://support.microsoft.com/ru-ru/help/296265/windows-2000-non-paged-pool-is-exhausted-by-afd-sys), или не может предотвратить возможное отключение неактивного клиента для других ОС.

Обычно Firebird использует опцию сокета 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 Unix-подобных ОС необходимо изменить содержимое /proc/sys/net/ipv4/tcp\_keepalive\_\*;
* в Windows необходимо вносить изменения в ветку реестра HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\ ключи (KeepAliveTime, KeepAliveInterva, TcpMaxDataRetransmissions).

### RemoteServicePort или RemoteServiceName

Область действия: Per-connection.

Тип параметра: Integer или String.

Параметры *RemoteServiceName* и *RemoteServicePort* используются для установки номера порта или имени сервиса, которые будут использоваться для клиентских соединений с базами данных.

Параметр *RemoteServiceName* имеет строковый тип. Значение по умолчанию равно gds\_db.

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

Пример 13.31. Установка параметра RemoteServiceName

RemoteServiceName = gds\_db

Пример 13.32. Установка параметра RemoteServicePort

RemoteServicePort = 3050

### Важно

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*.

### Примечание

Обычно один из этих параметров (чаще всего *RemoteServicePort*) меняют только на стороне севера в firebird.conf, поскольку на клиентской стороне имя сервиса или номер порта можно указать в строке подключения.

### RemoteAuxPort

Область действия: Per-connection.

Тип параметра: Integer.

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

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

Пример 13.33. Установка параметра RemoteAuxPort

RemoteAuxPort = 0

### TcpRemoteBufferSize

Область действия: Per-connection.

Тип параметра: Integer.

Параметр *TcpRemoteBufferSize* определяет размер TCP/IP буфера для обмена сообщениями между сервером и клиентом. Firebird может делать упреждающее чтение клиентом и может отправлять несколько строк за один сетевой пакет. Чем больше размер пакета, тем больше данных будет передаваться за одну передачу.

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

Пример 13.34. Установка параметра TcpRemoteBufferSize

TcpRemoteBufferSize = 8192

### TcpNoNagle

Область действия: Per-connection.

Тип параметра: Boolean.

Параметр *TcpNoNagle* включает или отключает использование Nagle алгоритма (опция TCP\_NODELAY для сокета) в TCP/IP соединениях.

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

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

Пример 13.35. Установка параметра TcpNoNagle

TcpNoNagle = 1

### IPv6V6Only

Область действия: Per-server.

Тип параметра: Boolean.

Этот параметр можно устанавливать только в firebird.conf. Firebird поддерживает **IPv6** подключение на стороне сервера и клиента. Параметр может принимать значения true/false, 1/0 или Yes/No. Значение по умолчанию равно false.

### Примечание

В Windows эта опция поддерживается начиная с Windows Vista, в более ранних версиях значение параметра всегда равно true.

Пример 13.36. Установка параметра IPv6V6Only

IPv6V6Only = 0

#### Сервер

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

#### Клиент

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

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

### RemoteBindAddress

Область действия: Per-server.

Тип параметра: String.

Параметр *RemoteBindAddress* позволяет привязать входящие соединения к IP адресу определенной сетевой карты. При этом все входящие соединения через другие сетевые интерфейсы будут запрещены. По умолчанию подключения из любого доступного сетевого интерфейса разрешены. Если вы используете классический сервер, этот параметр предназначен только для Windows. В Linux, BSD или Mac OS X с сервером Classic используйте конфигурационный файл xinetd или launchd (параметр *bind*).

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

Пример 13.37. Установка параметра RemoteBindAddress

RemoteBindAddress =

### LockMemSize

Область действия: Per-database.

Тип параметра: Integer.

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

Параметр имеет целочисленный тип. Единица измерения — байты. Значение по умолчанию равно 1 Мбайт. Минимальное значение равно 0. максимальное — 2 Гбайта.

Пример 13.38. Установка параметра LockMemSize

LockMemSize = 1M

На размер таблицы блокировок влияют:

1. В архитектурах Classic и SuperClassic размер страничного кэша (Super сервер для блокировки страниц использует лёгковесные защёлки). Страница, помещенная в кэш, блокируется, как минимум, один раз, страницы, которые читаются несколькими клиентами, могут блокироваться несколько раз.
2. Число одновременных транзакций. Каждая транзакция имеет блокировку. Блокировка используется для синхронизации транзакций.
3. События. Механизм оповещения о событиях использует блокировки. Число событий и число клиентов, ожидающих эти события, влияют на размер таблицы блокировок.

### LockAcquireSpins

Область действия: Per-database.

Тип параметра: Integer.

В архитектуре сервера Classic только одно клиентское соединение может обратиться к таблице блокировки в одно и то же время. Доступ к таблице блокировки управляется с помощью mutex(а). 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.

### Примечание

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

Пример 13.39. Установка параметра LockMemSize

LockMemSize = 1M

### LockHashSlots

Область действия: Per-database.

Тип параметра: Integer.

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

Параметр имеет целочисленный тип. По умолчанию значение параметра равно 8191. В качестве значения рекомендуется указывать простое число, чтобы хэш-алгоритм производил хорошее распределение.

Пример 13.40. Установка параметра LockHashSlots

LockHashSlots = 30011

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

fb\_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 хэш цепочки). 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).

### Примечание

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

### EventMemSize

Область действия: Per-database.

Тип параметра: Integer.

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

Параметр *EventMemSize* имеет целочисленный тип. Единица измерения — байты. Значение по умолчанию равно 64 Кбайта. Минимально допустимое значение равно 0. Максимальное значение равно 2 Гбайта.

Пример 13.41. Установка параметра EventMemSize

EventMemSize = 64K

## Настройки ядра

### CpuAffinityMask

Область действия: Per-server.

Тип параметра: Integer.

Параметр *CpuAffinityMask* позволяет указать, какие процессоры (ядра) будут использоваться сервером (только для ОС Windows в архитектуре Super сервер).

### Примечание

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 (могут быть использованы все доступные процессоры).

Пример 13.42. Установка параметра CpuAffinityMask

CpuAffinityMask = 64K

### GCPolicy

Область действия: Per-database.

Тип параметра: String.

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 немедлен- но 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 Super сервера «сборщик мусора» работает в комбинированном режиме. В архитектурах Classic и SuperClassic этот параметр игнорируется, а «сборщик мусора» всегда работает в оперативном режиме.

Пример 13.43. Установка параметра GCPolicy

GCPolicy = combined

### SecurityDatabase

Область действия: Per-database.

Тип параметра: String (путь к базе данных или алиас).

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

По-умолчанию в firebird.conf:

SecurityDatabase = $(dir\_secDb)/security3.fdb

Параметр может быть переопределен для определенной базы данных в файле databases.conf. Любая база данных может быть базой данных безопасности, в том числе и для самой себя.

## Настройки для Windows систем

### GuardianOption

Область действия: Per-server.

Тип параметра: Boolean.

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

* 0 — сервер стартует единожды;
* 1 — сервер стартует каждый раз после некорректного завершения.

Параметр имеет логический тип. Значение по умолчанию равно 1 (истина).

Пример 13.44. Установка параметра GuardianOption

GuardianOption = 1

### Примечание

В современных версиях Windows службы могут перезапускаться при аварийном завершении, если указаны соответствующие настройки. Это делает сторож (Guardian) не нужным при установке Firebird в качестве службы. Однако он всё ещё может быть полезным, если Firebird стартует в режиме приложения.

### ProcessPriorityLevel

Область действия: Per-server.

Тип параметра: Integer.

Параметр определяет уровень приоритетов процессов сервера Firebird. Параметр имеет целочисленный тип и может принимать значения:

* 0 — нормальный приоритет (значение по умолчанию);
* положительное значение — повышенный приоритет (тоже самое что опция -B в командной строке);
* отрицательное значение — пониженный приоритет.

### Важно

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

Пример 13.45. Установка параметра ProcessPriorityLevel

ProcessPriorityLevel = 0

### IpcName

Область действия: Per-connection.

Тип параметра: String.

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

### Примечание

Локальный протокол не совместим с Firebird версия которого меньше 2.0.

Пример 13.46. Установка параметра IpcName

IpcName = FIREBIRD

### Примечание

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

### RemotePipeName

Область действия: Per-connection.

Тип параметра: String.

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

Параметр имеет строковый тип. Значение по умолчанию равно interbas и совместимо с InterBase/Firebird 1.0.

Пример 13.47. Установка параметра RemotePipeName

RemotePipeName = interbas

## Настройки для Unix/Linux систем

### Redirection

Область действия: Per-server.

Тип параметра: Boolean.

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.

### Предупреждение

На сегодняшний день использование этой возможности (прокси сервер) представляет угрозу безопасности. Например, вы используете защищенный сервер Firebird, 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 (false). В этом случае возможность перенаправления запросов отключена. Для включения этой опции следует значение параметра выставить равным 1 (true).

Пример 13.48. Установка параметра Redirection

Redirection = 0

### Предупреждение

Не включайте эту опцию если вы не уверены, что именно она делает.

## Настройки архитектуры

### ServerMode

Область действия: Per-server.

Тип параметра: String.

Параметр *ServerMode* определяет архитектуру сервера. Существует 3 варианта архитектуры:

* Super (или ThreadedDedicated) — база данных открывается эксклюзивно одним серверным процессом, им же обслуживаются все соединения с базой данных. Подключения обрабатываются потоками из общего пула; используется общий страничный кэш на каждую базу данных для всех соединений и общий кэш для временного пространства (сортировок).
* SuperClassic (или ThreadedShared) — базы данных открываются одним серверным процессом, но доступ не исключительный — embedded процессы могут открыть одновременно одну ту же базу. Подключения обрабатываются потоками из общего пула; используется собственный страничный кэш для каждого соединения и общий кэш для временного пространства (сортировок).
* Classic (или MultiProcess) — создаётся отдельный процесс на каждое соединение с базой данных. Каждая база данных может быть открыта несколькими процессами (включая локальные для embedded доступа); используется отдельный кеш страниц на каждое соединение и отдельный кэш для временного пространства (сортировок).

## Глава 14. Конфигурирование и подбор аппаратного обеспечения

## Немного теории

Чтобы понять, какое аппаратное обеспечение лучше всего подойдет для вашей БД Firebird, мы должны понять, как Firebird использует его компоненты: CPU, RAM, HDD/SSD, и как эти компоненты взаимодействуют с ОС (например, с файловым кэшем).

### Функциональные модули сервера

Прежде всего, мы рассмотрим функциональные компоненты Firebird, в чем нам поможет следующий рисунок:

Рисунок 14.1. Firebird modules

|  |
| --- |
| Firebird modules |

Firebird включает в себя следующие основные функциональные компоненты:

1. Объекты метаданных — это представления таблиц, индексов, триггеров, хранимых процедур и других объектов БД. Объекты метаданных располагаются в адресном пространстве процесса Firebird (это может быть fbserver, fb\_inet\_server или firebird.exe).
2. Кэш страничных буферов содержит страницы базы данных, прочитанные с диска, и располагается в адресном пространстве процесса сервера. Механизм кэширования страниц достаточно сложный, поэтому ограничимся утверждением, что Firebird кэширует наиболее часто используемые страницы базы данных.
3. При выполнении сортировок Firebird выполняет ее в памяти (в адресном пространстве процесса сервера), пока размер используемой памяти для всех выполняемых одновременно сортировок не достигнет предела, установленного параметром TempCacheLimit (firebird.conf). При превышении этого лимита создается временный файл (с соответствующим флагом операционной системы) в папке временных файлов, и в нем выполняется сортировка. В случае, если в системе есть свободная память (RAM), то файл сортировки будет кэширован на уровне ОС и сортировка будет производиться в памяти.
4. Глобальные временные таблицы (GTT) создаются как временные файлы ОС. При наличии свободной памяти у ОС операции с GTT выполняются в RAM.

### Базовые операции с hardware

Давайте рассмотрим взаимодействие функциональных модулей Firebird с компонентами hardware в контексте операций, которые выполняются при работе с базами данных.

При старте Firebird процесс сервера занимает в RAM минимальный объем (несколько мегабайт), и не производит никаких интенсивных операций с CPU или RAM.

При соединении с БД сервер начинает читать её метаданные и создавать соответствующие объекты в памяти, отчего размер процесса увеличивается пропорционально количеству используемых таблиц, индексов, триггеров и других метаданных. Использование памяти увеличивается, но CPU практически не задействован на этом этапе.

Когда клиент начинает выполнять SQL-запросы (включая хранимые процедуры), сервер выполняет соответствующие операции, обращающиеся к hardware. Среди этих операций мы можем выделить следующие базовые операции, взаимодействующие с компонентами hardware:

* чтение страниц базы данных с диска;
* запись страниц БД на диск;
* чтение страниц базы данных из кэша;
* запись страниц БД в кэш;
* чтение и запись страниц данных в глобальные временные таблицы (Global Temporary Tables);
* обработка SQL запроса (например, JOINs );
* сортировка записей в результирующем наборе данных (resultset).

Для выполнения каждой из этой операции требуется определенный набор системных ресурсов. В таблице ниже представлено потребление ресурсов, выраженное в условных единицах интенсивности (1 означает небольшую интенсивность, 10 – максимальную):

Таблица 14.1. Оценка затрат на различные операции

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Read page from disk | Write page to disk | Read page from page buffers cache | Write page to page buffers cache | Read from GTT | Write to GTT | Сортировка записей | Обработка SQL |
| CPU | 1 | 1 | 1 | 1 | 1 | 1 | 5 | 10 |
| RAM | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 2 |
| Disk IO | 10 | 10 | 1 | 1 | 1 | 1 | 1 | 1 |

Как видите, что наиболее тяжелыми операциями являются те, которые включают работу с диском, так как дисковая подсистема, несмотря на прогресс последних лет, связанный с SSD, все равно является наиболее медленным компонентом hardware.

### Примечание

Отсюда вытекает одно из направлений оптимизации производительности, в полной мере относящееся к hardware – максимально переносить все операции чтения-записи в RAM. Сразу отметим, что подход «давайте сделаем страничный кэш побольше» не работает. Мы подробно рассмотрим этот вопрос в разделе «RAM».

### Параллельность выполнения операций

Обычно требует подобрать hardware для сервера, который будет обслуживать большое количество клиентов, поэтому важно понимать, как реализована параллельность выполнения операций.

С точки зрения компонентов аппаратного обеспечения можно говорить о параллельности использования CPU, диска и RAM. Современные CPU имеют несколько ядер, которые могут параллельно выполнять наборы инструкций, поэтому сервер СУБД распределяет операции между ядрами, то есть, можно сделать простой вывод – чем больше ядер у CPU, тем больше клиентов сможет работать на этом сервере.

С точки зрения дисковой подсистемы все не так однозначно. Традиционные жесткие диски (HDD) при считывании информации физически перемещают головку по магнитным дорожкам с некоторой конечной скоростью. База данных может быть достаточно большой, например, размером в 3 терабайта, и если клиентские SQL запросы будут параллельно обращаться к данным, расположенным в разных областях файла, то головка диска будет метаться между разными областями диска, серьезно затормаживая процессы чтения-вывода. При этом значительно вырастет очередь диска, а остальные ресурсы (CPU, RAM) будут простаивать. Разумеется, кэш дисковой подсистемы (кэш диска или RAID-контроллера) в какой-то мере компенсирует такое замедление, но недостаточно.

SSD диски, в отличие от традиционных HDD, в значительно меньшей мере страдают от деградации производительности при параллельном доступе к данным. Особенно заметно преимущество SSD при параллельной записи данных – наши тесты показывали 7-кратное преимущество SSD над SATA (<http://ib-aid.com/en/articles/firebird-performance-degradation-tests-myths-and-truth/> ). Однако, у SSD есть ряд моментов, которые надо обязательно учитывать при эксплуатации (см. раздел «Выбор дисковой подсистемы»), чтобы избежать падения скорости работы, преждевременной поломки диска и потери данных.

Операции с RAM на современном компьютере выполняются очень быстро, практически ограничены только пропускной способностью шины данных, и поэтому не являются узким местом даже при множестве параллельных SQL-запросов.

### Потоки данных

При выполнении SQL-запросов Firebird читает и записывает большое количество данных, перемещает их между функциональными модулями и соответствующими компонентами hardware. Нам необходимо изучить пути, по которым происходит обмен данными, чтобы идентифицировать возможные узкие места, в этом нам поможет рисунок ниже:

Рисунок 14.2. Data flows between RAM and persistent storage

|  |
| --- |
| Data flows between RAM and persistent storage |

Очевидно, что основные затраты времени возникают при перемещении данных из постоянного хранилища в RAM и назад. При этом возникает два потока данных – чтение/запись страниц данных файла базы данных и чтение/запись файлов сортировок. Так как файлов сортировок может быть несколько и они могут быть достаточно большими, то они могут существенно нагружать дисковую подсистему, поэтому желательно развести эти потоки ввода-вывода по разным дискам.

### Резервное копирование

Firebird предлагает 2 способа резервного копирования – верифицированный backup с помощью утилиты gbak, и неверифицированный инкрементальный backup c помощью утилиты nbackup.

Бэкап с помощью gbak является верифицированным, т.к. при его создании происходит чтение каждой записи в каждой таблице БД, и таким образом проверяется целостность информации в БД, а nbackup при создании бэкапа переносит страницы данных целиком, не проверяет целостность данных, и соответственно, является неверифицированным.

### Подсказка

Мы рекомендуем использовать комбинацию средств резервного копирования – частые запуски nbackup (например, каждый час, день и неделю) и ежедневный ночной верифицируемый backup с помощью gbak.

При резервном копировании любым способом 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.

## Выбор подходящего железа

Теперь, когда мы представляем, как Firebird взаимодействует с аппаратным обеспечением, необходимо подробно рассмотреть факторы, влияющие на выбор конкретного компонента и его характеристики.

В ряде случаев на выбор компонентов hardware сильно влияют реальные статистические данные конкретной БД, поэтому мы будем использовать инструменты из HQbird (профессионального дистрибутива Firebird для предприятий от IBSurgeon) для получения этих характеристик. Вы можете скачать триальную версию HQbird на сайте <http://ib-aid.com/en/hqbird/>.

### CPU

Для выбора CPU необходимо учитывать 3 вещи:

1. Какие запросы преобладают в приложении
2. Количество активных соединений с БД в среднем и в моменты пиковой нагрузки
3. Версию и архитектуру Firebird

#### Какие запросы преобладают в приложении?

Firebird 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, и чем больше ядер, тем меньше шанс, что все процессорные мощности будут заняты, а пользователи увидят замедление работы приложения.

Если приложение в основном исполняет простые короткие SQL запросы, все запросы хорошо отлажены, и не используется генерация запросов на лету (ad hoc) (например, для отчетов), то CPU не будет являться узким местом производительности, и можно выбрать младшую модель с меньшим количеством ядер.

Если приложение содержит генератор отчетов или большое количество медленных запросов, возвращающих большое количество данных, то необходим процессор с большим количеством ядер.

#### Количество активных соединений с БД в среднем и в моменты пиковой нагрузки

Количество соединений (активных пользователей) также влияет на выбор CPU. К сожалению, часто даже разработчики приложений не представляют, сколько в точности соединений, запросов и транзакций выполняется в данный конкретный момент. Для уточнения этой информации мы рекомендуем воспользоваться инструментом MON$ Logger из HQbird, и снять несколько снэпшотов в моменты рабочей нагрузки, на которых будет четко видно, сколько в действительности установлено соединений.

Рисунок 14.3. MON$Logger: number of attachments

|  |
| --- |
| MON$Logger: number of attachments |

Например, в данном примере видно, что число соединений составляет 296. Очевидно, что использование 4-х ядерного процессора в данном случае будет слишком оптимистичным, а вот 24-ядерный вариант вполне подойдет. Также желательно подсчитать число одновременно активных запросов, так как соединения могут простаивать, не выполняя никаких SQL запросов.

### Подсказка

Для грубой оценки необходимого количества ядер в CPU можно пользоваться правилом от 10 до 30 соединений на 1 ядро. 10 пользователей/ядро – приложение с преобладанием сложных и медленных запросов, 30 соединений/ядро – приложение с преобладанием простых, хорошо отлаженных запросов. Для более точной оценки можно посмотреть на закладке Statements число активных запросов (состояние Active) и всех остальных (IDLE И др.)

#### Версия и архитектура Firebird

Если Вы используете версию Firebird 2.5, то следует иметь в виду, что для распараллеливания обработки на несколько ядер следует использовать архитектуры Classic или SuperClassic. Архитектура SuperServer в версии 2.5 может использовать только одно ядро на одну БД, поэтому ее не следует применять в высоконагруженных системах.

В версии Firebird 3.0 и SuperServer, и Classic, и SuperClassic используют возможности многоядерных CPU. Наибольшую производительность показывает Firebird 3.0 SuperServer.

### RAM

При выборе RAM следует уделить внимание 2 моментам:

1. Модули памяти должны быть с коррекцией ошибок (ECC RAM)
2. Правильно рассчитать объем RAM

#### ECC RAM

ECC RAM значительно снижает количество ошибок при работе с памятью и настоятельно рекомендуется для использования в промышленных системах.

#### Расчет необходимого объема RAM

Для расчета памяти нам придется немного углубиться в особенности различных архитектур Firebird.

Firebird 2.5 архитектуры Classicи Firebird 3.0 Classic запускают отдельный процесс для обслуживания каждого соединение, SuperClassic запускает отдельный поток для каждого соединения, но практически с той же структурой потребления памяти – каждое соединение имеет свой независимый страничный кэш.

Firebird SuperServer запускает один процесс с единым страничным кэшем для всех соединений.

Таким образом, потребление памяти складывается из следующих основных параметров:

1. Количество соединений
2. Размер страницы базы данных
3. Размер объектов метаданных (пропорционален количеству таблиц, триггеров, хранимых процедур и др., не регулируется, определяется по фактическому использованию)
   * Для Classic и SuperClassic – на соединение
   * Для SuperServer – на экземпляр открытой базы данных
4. Размер страничного кэша (определяется параметрами в заголовке БД или в firebird.conf или в свойствах конкретного соединения)
   * Для Classic и SuperClassic – на соединение
   * Для SuperServer – на экземпляр открытой базы данных
5. Размер кэша для сортировок (определяется параметром в firebird.conf) Обратите внимание, что память для сортировок выделяется по мере необходимости, а не сразу же.
   * Для Classic – на соединение
   * Для Super Serverи SuperClassic – на процесс (т.е., кэш сортировок единый)
6. Для Classic/SuperClassic – размер таблицы блокировок (так как обычно он небольшой, то его выведем из расчетов).

Компанией IBSurgeon в ряде экспериментов получен ряд оптимальных значений количества страниц в страничном кэше Firebird:

* Classic/SuperClassic – от 256 до 2000 страниц
* SuperServer 2.5 – 10000 страниц
* SuperServer 3.0 – 100000 страниц

На основании этих экспериментов были созданы оптимизированные конфигурационные файлы Firebird для серверов с 4-6Гб памяти, вы можете скачать их отсюда: <http://ib-aid.com/ru/optimized-firebird-configuration/>

##### Формулы расчета необходимого объема RAM

Ниже представлены формулы приблизительного расчета необходимого объема памяти для Firebird. Реальное значение потребления памяти может отличаться, так как в этом расчете не учитывается объем памяти под метаданные, под битовые маски индексов, и т.д., что может увеличить расход памяти, но одновременно предполагается, что память под сортировки будет использована полностью во всех соединениях, чего обычно не происходит.

Когда база данных уже находится в эксплуатации, можно просто посмотреть средний размер памяти, используемый процессом Firebird (с помощью TaskManager или ProcessExplorer).

Расчет для Classic:

Кол-во соединений \* ((Кол-во страниц кэша \* Размер страницы) + Размер кэша для сортировок)

Пример для Classic: пусть мы ожидаем 100 активных пользователей, размер страницы БД мы установили в 8Кб, а количество страниц в страничном кэше установили в 256, размер кэша для сортировок увеличили с 8Мб (значение по умолчанию для Classic и SuperClassic) до 64Мб:

100\*((256\*8кб)+64 Мб) = 6600 Мб

Расчет для SuperClassic:

Кол-во соединений \* (Кол-во страниц кэша \* Размер страницы) + Размер кэша для сортировок

Пример для SuperClassic: 100 пользователей, размер страницы БД 8Кб, количество страниц в страничном кэше 256, размер кэша для сортировок 1024 Мб

100\*(256\*8Кб) + 1024 Мб = 2024 Мб

Расчет для SuperServer:

Кол-во страниц кэша \* Размер страницы + Размер кэша для сортировок

Пример для SuperServer (Firebird 2.5): 1БД, 100 пользователей, размер страницы БД 8кб, количество страниц в страничном кэше 10000, размер кэша для сортировок 1024 Мб:

(10000\*8Кб) + 1024 = 1102 Мб

Пример для SuperServer (Firebird 3.0): 1 БД, 100 пользователей, размер страницы БД 8кб, количество страниц в страничном кэше 100000, размер кэша для сортировок 1024 Мб:

(100000\*8Кб) + 1024 = 1805 Мб

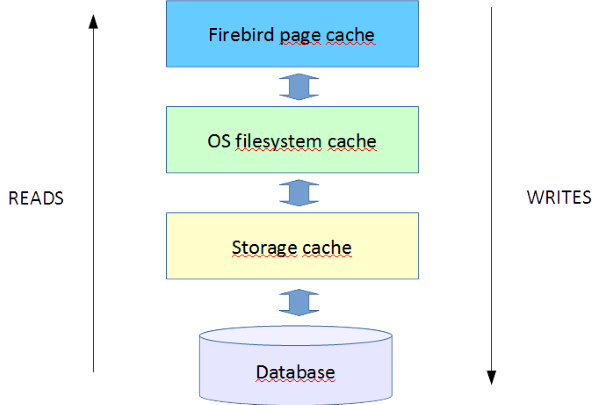
##### Излишняя память

Часто Firebird упрекают в неэффективном использовании памяти, когда работающий процесс сервера потребляет небольшое количество RAM, а остающаяся память якобы не используется.

На самом деле это неверное суждение, происходящее, в основном, из-за непонимания работы механизма кэширования Firebird и несовершенства инструментов мониторинга операционных систем.

Прежде всего, необходимо четко представлять, что Firebird активно использует файловый кэш операционной системы. Когда страница загружается в страничный кэш Firebird, она проходит через файловый кэш ОС. Когда Firebird выгружает страницу из своего страничного кэша, операционная система при наличии свободной памяти продолжает держать этот кусок базы данных в RAM.

Рисунок 14.4. Cache levels: Firebird, OS and storage



Однако, при поверхностном взгляде, операционная система не показывает занятую под файловую кэш память как используемую. Например, вот типичная ситуация распределения памяти при работающем сервер Firebird, как ее показывает TaskManager:

Рисунок 14.5. TaskManager does not show file cache usage

|  |
| --- |
| TaskManager does not show file cache usage |

Выглядит так, как будто используется только 6.3Гб из 16Гб.

Однако, если использовать инструмент RAMMap (из набора утилит SysInternals от Microsoft), то картина выглядит более логичной:

Рисунок 14.6. RAMMap shows details about memory usage: mapped files are cached databases

|  |
| --- |
| RAMMap shows details about memory usage: mapped files are cached databases |

Файлы баз данных (dbw350\_fb252x64.fdb и dbw250\_fb252x64.fdb) закэшированы ОС, и занимают всю память, декларированную TaskManager как свободную:

Рисунок 14.7. RAMMap: details about file cache usage

|  |
| --- |
| RAMMap: details about file cache usage |

Отсюда вывод — операционная система эффективно использует всю доступную память для кэширования базы данных, вплоть до полной загрузки БД в память.

### Дисковая подсистема

Правильная конфигурация дисковой подсистемы является очень важной частью выбора и конфигурирования нardware для Firebird, так как ошибки на этом этапе дорого обходятся и тяжело исправляются.

#### Отдельные диски для всего

Чтобы уменьшить конкуренцию за дисковый ввод-вывод между операциями с файлом БД, сортировками и резервными копиями, а также уменьшить шанс одновременной потери и БД и резервных копий, рекомендуется иметь 3 разных диска (или raid-массива): для БД, для временных файлов и для создания и хранения бэкапов.

Когда мы говорим «отдельные диски», это означает, 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, скорее всего, возрастет, так как устройство может распределять потоки данных между контроллерами.

Иногда сообщают о том, что выделение отдельного диска для хранения файлов ОС и swap-файла ОС приводит к увеличению производительности.

#### SSD для базы данных

Для работы с базой данных лучше всего использовать SSD-диски, так как они обеспечивают отличное масштабирование при параллельном вводе-выводе. Обязательно следует использовать диски промышленного класса, с увеличенным числом циклов перезаписи, иначе велик риск потери данных из-за поломки SSD.

Некоторое время назад SSD-диски страдали от повышенного износа в случае, если на диске оставалось небольшое количество (менее 30%) свободного места. Упрощенно излагая, каждое изменение на SSD диске пишется в новую свободную ячейку, поэтому недостаток свободного места приводил к повышенному износу ячеек, которые остались свободными, и сокращению срока жизни диска.

Производители современных контроллеров SSD декларируют, что эта проблема была решена с помощью превентивного перемещения статичных данных, и теперь износ ячеек идет более или менее равномерно. Однако, точные спецификации и алгоритмы работы SSD закрыты производителями, поэтому мы все равно рекомендуем оставлять до 30% места на SSD-дисках свободным, а ожидаемый срок жизни занижать и планировать замену дисков не реже чем 1 раз в 3 года.

Например, если размер Вашей БД в данный момент составляет 100Гб, рост БД идет по 1Гб в месяц, то нельзя приобретать SSD минимального размера (120Гб), а лучше выбрать следующий в линейке — 250Гб. В тоже время, покупка 512Gb SSD будет пустой тратой денег, т.к. через 3 года диск будет желательно заменить.

Наилучшим выбором является использование SSD эксклюзивно для работы с базой данных, так как любые операции ввода-вывода сокращают срок службы дисков.

#### Диск для временных файлов

Так как временные файлы на диске возникают только при отсутствии достаточного количества RAM, то конечно, лучше всего вообще избегать их появления на диске. Оценку количества и размера временных файлов в production системе можно провести лишь путем мониторинга папки с временными файлами. Такой мониторинг осуществляет FBDataGuard из дистрибутива HQbird. Узнав, сколько временных файлов сортировок и когда создается на диске, можно соответствующим образом нарастить RAM и изменить параметры firebird.conf.

В любом случае, Firebird требует указания папки, где будут хранится временные файлы. Обычно его оставляют по умолчанию, т.е. используется папка ОС для временных файлов. В случае, если свободной RAM достаточно, это достаточно хороший выбор.

Однако, есть еще один важный момент в размещении временных файлов на диске – это создание индексов при восстановлении верифицированного бэкапа (созданного утилитой gbak). При создании индекса создается временный файл, который содержит все ключи этого индекса. Если база данных достаточно велика, то размер индекса для какой-нибудь большой таблицы может быть также значительным. Например, в БД размером 1 Тб индекс для самой большой таблицы в 3.2 миллиардов записей в базе данных имеет размер 29 Гб, а при создании такого индекса потребовалось 180 гигабайт свободного места:

Рисунок 14.8. IBAnalyst: tables and indexes

|  |
| --- |
| IBAnalyst: tables and indexes |

Чтобы предупредить исчерпание свободного места на системном диске, в Firebird.conf указывают второй диск в качестве дополнительного резервного места:

TempDirectories = C:\temp; H:\Temp

Если на первом диске пространство кончится, Firebird продолжит использовать для временных файлов второй диск, и так далее.

#### HDD for backups

Для создания и хранения бэкапов подойдут обычные HDD диски с интерфейсом SATA или nSAS. Они обеспечивают быструю линейную запись и чтение файлов бэкапа, и достаточно дешевы, чтобы не экономить на их размере, и хранить несколько резервных копий.

Для создания и хранения бэкапов подойдут обычные HDD диски с интерфейсом SATA или nSAS. Они обеспечивают быструю линейную запись и чтение файлов бэкапа, и достаточно дешевы, чтобы не экономить на их размере, и хранить несколько резервных копий.

Если же удалять предыдущий бэкап до окончания процесса создания нового, возможна ситуация, что в результате сбоя копирования новый бэкап не будет создан, старый уже удален, а база данных также повреждена, например, в результате сбоя диска.

Если Вы используете рекомендованный выше способ организации резервного копирования в виде комбинации инкрементального бэкапа в 3 уровнях и верифицированного бэкапа один раз в сутки, хранением только 1 последней копии, то формула расчета минимального необходимого места для бэкапа следующая:

Размер\_БД \* 3 + 0.2 \* Размер\_БД

Пример расчета места, необходимого для бэкапа. Пусть у нас есть БД в 100Гб, для которой мы храним 3 уровня инкрементального бэкапа неделя-день-час (по 1 копии) и 1 копию ежедневного верифицировованного бэкапа. Тогда резервные копии будут занимать следующее место:

* Nbackup\_level\_0\_weekly - 100Гб
* Nbackup\_level\_1\_daily – 5Гб (приблизительная оценка)
* Nbackup\_level\_2\_hourly – 200 Мб (приблизительная оценка)
* Daily verified backup – 100Гб (приблизительная оценка)
* Плюс нужен резерв в 110Гб для создания следующей копии бэкапа.

Итого – 316Гб.

### Важно

Размер инкрементного файла уровня 1 и выше зависит от количества страниц, которые были изменены с момента предыдущего выполнения nbackup. Определить размер этих файлов можно только экспериментально, т.к. объем изменений в БД зависит от приложений.

размер инкрементного файла уровня 1 и выше зависит от количества страниц, которые были изменены с момента предыдущего выполнения nbackup. Определить размер этих файлов можно только экспериментально, т.к. объем изменений в БД зависит от приложений.

### Примечание

Разумеется, интеллектуальные средства резервного копирования (FBDataGuard из HQbird), отследят недостаток места для бэкапов, предотвратят старт неудачного бэкапа и вышлют соответствующее сообщение администратору.

#### HDD для базы данных

Может случиться так, что SSD окажется слишком дорогим решением, или размер БД слишком велик, и необходимо обойтись более дешевыми альтернативами.

В этом случае следует использовать HDD с интерфейсом SAS, при невозможности — диски SATA с интерфейсом nSAS, и совсем бюджетный вариант — обычные диски SATA.

Для ускорения работы жестких дисков (а также, надежности – см. ниже) следует объединять их в RAID10. RAID10 – это комбинация зеркалирования (RAID1) и stripe (RAID0). Хороший и правильно настроенный RAID контроллер с большим кэшем является неплохой альтернативой SSD.

#### Надежность и RAID

Разумеется, во всех упомянутых выше вариантах (за исключением диска, выделенного эксклюзивно для временных файлов) следует увеличивать надежность дисковой подсистемы путем объединения дисков в RAID.

* Для SSD дисков следует обязательно использовать RAID1 – т.е. 2 «зеркальных» диска, на которые одновременно пишутся изменения, что значительно уменьшит шанс полной потери данных. RAID 10 из SSD скорее всего будет избыточным, т.к. шина RAID будет ограничивать пропускную способность. Например, интерфейс 6 Гбит/с имеет пропускную способность в 600 мегабайт в секунду, а современные одиночные SSD уже достигли такой скорости. Таким образом, для RAID 10 мы получим тот же самый лимит в 600 мб/с. Разве что с PCI Express 3.0 можно организовывать RAID 10 из SSD, т.к. пропускная способность этой шины уже 16 гигабит в секунду и выше.
* 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. Некоторые пользователи используют также RAID5, жертвуя производительностью взамен увеличенного пространства.

#### Настройка RAID

Прежде всего, необходимо проверить наличие и заряд батареи резервного питания (Backup Battery Unit, BBU) в RAID. При отсутствии данной батареи большинство RAID переходят в режим безопасной записи (полностью отключено кэширование записи), который обеспечивает меньшую скорость IO, чем обычный SATA диск!

### Примечание

Именно с этим фактом связано большинство горестных писем в техподдержку от пользователей, которые приобрели дорогой сервер и обнаружили, что он работает медленнее, чем десктопный компьютер. К сожалению, часть вендоров по умолчанию не снабжает свои RAID батареями, поэтому это первое, что необходимо проверить и, при необходимости, исправить.

Затем необходимо провести настройку кэширования чтения и записи. Часто по умолчанию кэш полностью выключен, а если мы хотим добиться приличной скорости от RAID, то кэш необходимо включить.

Кроме включения кэша, необходимо проверить режим его работы, это может быть write through и write back. Быстрый способ работы с кэшем – write back, в этом случае изменения пишутся в кэш контроллера, и через некоторое время – непосредственно на диск.

Проверка наличия батареи, кэша и режима работы RAID может быть осуществлена в фирменных утилитах, которые производители поставляют вместе с RAID.

Современные RAID-контроллеры также имеют возможности тонкой настройки кэша — его можно ориентировать на чтение или на запись. Обычно по умолчанию стоит 50%/50% на чтение-запись.

Чтобы определить, в каком отношении стоит настраивать кэш, можно вновь воспользоваться инструментом MON$ Logger из продвинутого дистрибутива HQbird. Она показывает отношение числа операций чтения к операциям записи (в агрегированном виде с момента первого соединения к серверу):

Рисунок 14.9. HQbird MON$logger: reads/write ratio

|  |
| --- |
| HQbird MON$logger: reads/write ratio |

Как видите, в данном примере операций чтения гораздо больше, чем операций записи, поэтому имеет смысл настроить RAID контроллер на 80% чтений и 20% записи.

#### СХД

В последние годы получили распространение интегрированные системы хранения данных (СХД), которые включают в себя массив дисков с гибкими возможностями конфигурации (все типы RAID) и продвинутым кэшированием. Обычно СХД имеют несколько контроллеров ввода-вывода, что позволяет обслуживать несколько серверов одновременно, а также работать достаточно быстро.

Многие организации приобретают СХД и используют их для работы с базами данных Firebird. При условии правильной настройки от СХД можно добиться очень хорошей производительности. Основные моменты, которые нужно учитывать при использовании СХД:

1. Наличие нескольких высокопроизводительных контроллеров дисков, обеспечивающих многоканальный обмен данными
2. Наличие батарей резервного питания (BBU), если они предусмотрены конструкцией.
3. Диски для баз данных должны быть организованы в RAID10.
4. Кэш должен быть включен, режим записи установлен в write back.
5. Если к СХД подключены несколько компьютеров, то каждый из них обслуживается своим контроллером.
6. Установлены свежие драйвера для СХД. В нашей практике были случаи, когда более новые драйвера обеспечивали 30% прирост скорости.
7. Если на СХД созданы несколько логических дисков (для БД, бэкапов, ОС), то они разделены по каналам ввода вывода. Попытка использовать один канал для всех дисков сразу приведет к снижению производительности.
8. Аналогично, если СХД будут использовать сразу несколько серверов и БД, производительность может уменьшиться из-за превышения полосы пропускания (bandwidth) контроллеров ввода-вывода.
9. Часто используются комбинированные схемы, когда ОС и временные файлы находятся на локальных дисках, а база данных и бэкапы на СХД.

Часто СХД используются в схеме «2 сервера – 1 СХД», с целью построения отказоустойчивого кластера. Надо отметить, что такой кластер поможет победить только проблемы, связанные непосредственно с поломкой hardware одного из серверов, чтобы быстро переключиться на второй сервер. Если проблема возникнет с СХД или с самой базой данных, то такое решение будет бесполезно.

Для построения реального отказоустойчивого решения необходимо использовать решения с репликацией данных между экземплярами баз данных. Такие решения доступны для Firebird 2.5 и 3.0.Вы можете обратиться на support@ib-aid.com, чтобы узнать больше о решениях высокой доступности для Firebird.

## Краткие выводы и рекомендации

Давайте коротко суммируем выводы и рекомендации по hardware для Firebird.

1. Необходимо использовать многоядерные CPU для обслуживания большого количества пользователей
2. Оперативная память RAM – обязательное минимальное количество рассчитывается из количества пользователей и параметров БД, количество сверх того будет эффективно использоваться операционной системой для кэширования файла БД.
3. Использовать отдельные диски для БД, temp-файлов и бэкапов
4. Для размещения БД лучше использовать SSD диски.
   * Резервировать от 30% свободного места на SSD дисках.
   * Желательно эксклюзивно использовать диск под БД.
   * Использовать SSD промышленного класса (с большим количеством циклов перезаписи).
5. Обязательно использовать RAID.
   * Для SSD – RAID 1, для HDD – RAID10, для HDD под бэкапы – RAID1.
   * Обязательно проверять наличие и заряд батареи RAID
   * Обязательно проверять настройку write back/through (нужно ставить write back).
   * Ряд контроллеров имеет настройку размера кэша RAID, например, 75% для чтения, 25% для записи, или 50/50, и т.д. Необходимо ставить Mon$logger, ПО для управления параметрами RAID, смотреть соотношение чтения и записи, и менять настройки RAID.
   * В использовании СХД есть свои плюсы и минусы, чтобы получить эффект от ее использования, необходимо правильно настраивать СХД.
6. Для построения отказоустойчивого решения необходимо использовать решения с репликацией, работающие на нескольких серверах.

## Глава 15. Использование инструментов резервного копирования

## gbak

## nbackup

## Глава 16. Использование инструмента валидации и починки БД gfix

## Глава 17. Безопасность

## Плагины аутентификации и их использование

## Плагины шифрования и их использование

## Глава 18. Репликация (Firebird 4.0)

## Глава 19. Использование инструментов мониторинга Firebird

## Использование fb\_lock\_print

## Использование fbtracemgr для Trace API

## Использование таблиц мониторинга MON$

### Символы

Параметр конфигурации

AuditTraceConfigFile, [AuditTraceConfigFile](#config-param-AuditTraceConfigFile)

AuthClient, [AuthServer и AuthClient](#config-param-AuthServer)

AuthServer, [AuthServer и AuthClient](#config-param-AuthServer)

BugcheckAbort, [BugcheckAbort](#config-param-BugcheckAbort)

ConnectionTimeout, [ConnectionTimeout](#config-param-ConnectionTimeout)

CpuAffinityMask, [CpuAffinityMask](#config-param-CpuAffinityMask)

DatabaseAccess, [DatabaseAccess](#config-param-DatabaseAccess)

DatabaseGrowthIncrement, [DatabaseGrowthIncrement](#config-param-DatabaseGrowthIncrement)

DeadlockTimeout, [DeadlockTimeout](#config-param-DeadlockTimeout)

DefaultDbCachePages, [DefaultDbCachePages](#config-param-DefaultDbCachePages)

DummyPacketInterval, [DummyPacketInterval](#config-param-DummyPacketInterval)

EventMemSize, [EventMemSize](#config-param-EventMemSize)

ExternalFileAccess, [ExternalFileAccess](#config-param-ExternalFileAccess)

FileSystemCacheSize, [FileSystemCacheSize](#config-param-FileSystemCacheSize)

FileSystemCacheThreshold, [FileSystemCacheThreshold](#config-param-FileSystemCacheThreshold)

GCPolicy, [GCPolicy](#config-param-GCPolicy)

GuardianOption, [GuardianOption](#config-param-GuardianOption)

IpcName, [IpcName](#config-param-IpcName)

IPv6V6Only, [IPv6V6Only](#config-param-IPv6V6Only)

KeyHolderPlugin, [KeyHolderPlugin](#config-param-KeyHolderPlugin)

LockAcquireSpins, [LockAcquireSpins](#config-param-LockAcquireSpins)

LockHashSlots, [LockHashSlots](#config-param-LockHashSlots)

LockMemSize, [LockMemSize](#config-param-LockMemSize)

MaxUnflushedWrites, [MaxUnflushedWrites](#config-param-MaxUnflushedWrites)

MaxUnflushedWriteTime, [MaxUnflushedWriteTime](#config-param-MaxUnflushedWriteTime)

MaxUserTraceLogSize, [MaxUserTraceLogSize](#config-param-MaxUserTraceLogSize)

ProcessPriorityLevel, [ProcessPriorityLevel](#config-param-ProcessPriorityLevel)

Providers, [Providers](#config-param-Providers)

Redirection, [Redirection](#config-param-Redirection)

RelaxedAliasChecking, [RelaxedAliasChecking](#config-param-RelaxedAliasChecking)

RemoteAccess, [RemoteAccess](#config-param-RemoteAccess)

RemoteAuxPort, [RemoteAuxPort](#config-param-RemoteAuxPort)

RemoteBindAddress, [RemoteBindAddress](#config-param-RemoteBindAddress)

RemoteFileOpenAbility, [RemoteFileOpenAbility](#config-param-RemoteFileOpenAbility)

RemotePipeName, [RemotePipeName](#config-param-RemotePipeName)

RemoteServiceName, [RemoteServicePort или RemoteServiceName](#config-param-RemoteServicePort)

RemoteServicePort, [RemoteServicePort или RemoteServiceName](#config-param-RemoteServicePort)

SecurityDatabase, [SecurityDatabase](#config-param-SecurityDatabase)

ServerMode, [ServerMode](#config-param-ServerMode)

TcpNoNagle, [TcpNoNagle](#config-param-TcpNoNagle)

TcpRemoteBufferSize, [TcpRemoteBufferSize](#config-param-TcpRemoteBufferSize)

TempBlockSize, [TempBlockSize](#config-param-TempBlockSize)

TempCacheLimit, [TempCacheLimit](#config-param-TempCacheLimit)

TempDirectories, [TempDirectories](#config-param-TempDirectories)

TracePlugin, [TracePlugin](#config-param-TracePlugin)

UdfAccess, [UdfAccess](#config-param-UdfAccess)

UserManager, [UserManager](#config-param-UserManager)

WireCompression, [WireCompression](#config-param-WireCompression)

WireCrypt, [WireCrypt](#config-param-WireCrypt)

WireCryptPlugin, [WireCryptPlugin](#config-param-WireCryptPlugin)

### I

ISQL, [ISQL. Интерактивные запросы](#isql)

ADD, [ADD](#isql-commands-add)

BLOBDUMP, [BLOBDUMP](#isql-commands-blobdump)

BLOBVIEW, [BLOBVIEW](#isql-commands-blobview)

CONNECT, [Соединение с базой данных](#isql-connect)

COPY, [COPY](#isql-commands-copy)

CREATE DATABASE, [Создание базы данных](#isql-create_db)

EDIT, [EDIT](#isql-commands-edit)

EXIT, [Завершение сессии Isql](#isql-end_session), [EXIT](#isql-commands-exit)

HELP, [Получение справки](#isql-help), [HELP](#isql-commands-help)

HELP SET, [Получение справки](#isql-help)

INPUT, [INPUT](#isql-commands-input)

OUTPUT, [OUTPUT](#isql-commands-output)

QUIT, [Завершение сессии Isql](#isql-end_session), [QUIT](#isql-commands-quit)

SET, [SET](#isql-commands-set), [SET](#isql-commands_set-set)

SET AUTODDL, [SET AUTODDL](#isql-commands_set-set_autoddl)

SET BAIL, [SET BAIL](#isql-commands_set-set_bail)

SET BLOB, [SET BLOB](#isql-commands_set-set_blob)

SET BLOBdisplay, [SET BLOB](#isql-commands_set-set_blob)

SET BULK\_INSERT, [SET BULK\_INSERT](#isql-commands_set-set_bulk_insert)

SET COUNT, [SET COUNT](#isql-commands_set-set_count)

SET ECHO, [SET ECHO](#isql-commands_set-set_echo)

SET EXPLAIN, [SET EXPLAIN](#isql-commands_set-set_explain)

SET HEADING, [SET HEADING](#isql-commands_set-set_heading)

SET LIST, [SET LIST](#isql-commands_set-set_list)

SET NAMES, [SET NAMES](#isql-commands_set-set_names)

SET PLAN, [SET PLAN](#isql-commands_set-set_plan)

SET PLANONLY, [SET PLANONLY](#isql-commands_set-set_planonly)

SET SQL DIALECT, [SET SQL DIALECT](#isql-commands_set-set_sqldialect)

SET SQLDA\_DISPLAY, [SET SQLDA\_DISPLAY](#isql-commands_set-set_sqlda)

SET STATs, [SET STATs](#isql-commands_set-set_stats)

SET TERM, [SET TERM](#isql-commands_set-set_term)

SET TIME, [SET TIME](#isql-commands_set-set_time)

SET WARNINGS, [SET WARNINGS](#isql-commands_set-set_warnings)

SET WIDTH, [SET WIDTH](#isql-commands_set-set_width)

SET WNG, [SET WARNINGS](#isql-commands_set-set_warnings)

SHELL, [SHELL](#isql-commands-shell)

SHOW CHECK, [SHOW CHECK](#isql-commands_show-show_check)

SHOW COLLATE, [SHOW COLLATION](#isql-commands_show-show_collations)

SHOW COLLATION, [SHOW COLLATION](#isql-commands_show-show_collations)

SHOW COMMENTS, [SHOW COMMENTS](#isql-commands_show-show_comments)

SHOW DATABASE, [SHOW DATABASE](#isql-commands_show-show_db)

SHOW DB, [SHOW DATABASE](#isql-commands_show-show_db)

SHOW DEPENDENCIES, [SHOW DEPENDENCIES](#isql-commands_show-show_dependencies)

SHOW DEPENDENCY, [SHOW DEPENDENCIES](#isql-commands_show-show_dependencies)

SHOW DOMAIN, [SHOW DOMAIN](#isql-commands_show-show_domain)

SHOW EXCEPTION, [SHOW EXCEPTION](#isql-commands_show-show_exeption)

SHOW FILTER, [SHOW FILTER](#isql-commands_show-show_filter)

SHOW FUNCTION, [SHOW FUNCTION](#isql-commands_show-show_function)

SHOW GENERATOR, [SHOW GENERATOR](#isql-commands_show-show_generator)

SHOW GRANTS, [SHOW GRANTS](#isql-commands_show-show_grants)

SHOW INDEX, [SHOW INDEXES](#isql-commands_show-show_indexes)

SHOW MAPPING, [SHOW MAPPING](#isql-commands_show-show_mapping)

SHOW PACKAGE, [SHOW PACKAGE](#isql-commands_show-show_package)

SHOW PROCEDURE, [SHOW PROCEDURE](#isql-commands_show-show_procedure)

SHOW ROLE, [SHOW ROLE](#isql-commands_show-show_role)

SHOW SECCLASSES, [SHOW SECCLASSES](#isql-commands_show-show_secclasses)

SHOW SEQUENCE, [SHOW SEQUENCE](#isql-commands_show-show_sequence)

SHOW SQL DIALECT, [SHOW SQL DIALECT](#isql-commands_show-show_sql_dialect)

SHOW SYSTEM, [SHOW SYSTEM](#isql-commands_show-show_system)

SHOW TABLE, [SHOW TABLE](#isql-commands_show-show_table)

SHOW TRIGGER, [SHOW TRIGGER](#isql-commands_show-show_trigger)

SHOW USERS, [SHOW USERS](#isql-commands_show-show_users)

SHOW VERSION, [SHOW VERSION](#isql-commands_show-show_version)

SHOW VIEW, [SHOW VIEW](#isql-commands_show-show_view)