



Ред База Данных

Версия 2.1

Примечания к выпуску

Оглавление

Оглавление.....	2
1. Контроль доступа к системным сервисам (Service access).....	3
2. Доступ к операциям управления метаданными (DDL access).....	4
3. Доступ к операциям манипулирования данными (DML access).....	5
4. Предопределенные роли (Default Roles).....	6
5. Кумулятивное (комбинированное) действие ролей (Combine Roles).....	7
6. Многофакторная аутентификация (Authentication).....	9
7. Политики безопасности (Login policies).....	11
8. Обезличивание памяти (Memory Elimination).....	14
9. Аудит.....	15
10. Адаптер для подключения бинарного файла аудита (LogAnalyzer).....	21
11. Контроль целостности метаданных.....	23
12. Контроль целостности файлов сервера.....	24
13. Функциональные доработки.....	25
Приложение А - Изменения в ODS.....	33

1. Контроль доступа к системным сервисам (Service access)

СУБД «Ред База Данных» 2.1 поддерживает распределение прав на вызов системных сервисов.

Список административных функций, доступ к которым разграничивается:

- резервное копирование (Backup Database, GBAK);
- восстановление базы из резервной копии (Restore Database, GBAK);
- получение списка пользователей (Display User, GSEC);
- добавление пользователя (Add User, GSEC);
- удаление пользователя (Delete User, GSEC);
- редактирование пользователя (Modify User, GSEC);
- получение свойств БД (Database Properties, GFIX);
- анализ и восстановление поврежденной БД (Repair Database, GFIX);
- получение статистики БД (Database Stats, GSTAT);
- получение лога сервера (Get Log File, GSTAT).

Право на запуск сервисов является глобальным для сервера, поэтому хранится в базе данных безопасности security2.fdb. Назначить право на запуск сервиса можно только пользователям и глобальным ролям. Назначить право на запуск сервиса может только администратор сервера СУБД «Ред База Данных» - пользователь SYSDBA и пользователь, которому назначена глобальная роль безопасности администратора безопасности — SECADMIN.

Для назначения права на запуск сервиса используется следующий оператор SQL:

```
GRANT EXECUTE ON SERVICE <SERVICE_NAME> TO USER | ROLE
```

Этот запрос выполняется при прямом подключении к базе данных безопасности security2.fdb (на пользовательской БД запрос приведет к выводу сообщения об ошибке назначения прав на сервис)

Параметр <SERVICE_NAME> может иметь следующие значения:

- BACKUP
- RESTORE
- GFIX
- GSTAT

Операции с сервисом GSEC может производить любой пользователь, но его права при этом ограничены возможностью изменения своего пароля. Полные права на сервис GSEC имеет только SYSDBA и пользователи с глобальной ролью SECADMIN.

2. Доступ к операциям управления метаданными (DDL access)

Суть доработки заключается в расширении функционала операторов GRANT и REVOKE для объектов следующих типов: PROCEDURE, FUNCTION, ROLE, TABLE, VIEW, EXCEPTION, GENERATOR, DOMAIN, SHADOW, POLICY.

Таким образом, права на операцию CREATE определяются конструкциями следующего вида:

```
GRANT CREATE OBJECT TO USER|ROLE [WITH GRANT OPTION];
REVOKE CREATE OBJECT FROM USER|ROLE;
```

Права на операции ALTER и DROP определяются конструкциями следующего вида:

```
GRANT ALTER|DROP [ANY] OBJECT TO USER|ROLE [WITH GRANT OPTION];
REVOKE ALTER|DROP [ANY] OBJECT FROM USER|ROLE;
```

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

Опция WITH GRANT OPTION предоставляет пользователю возможность передачи назначаемого ему права.

В таблице 1 представлены языковые конструкции для выполнения DDL операций над объектами СУБД.

Таблица 1. – Назначение прав на DDL операции

Операции	Объект	Назначение прав
CREATE, ALTER, DROP	TABLE	GRANT CREATE ALTER DROP [ANY] TABLE TO USER ROLE [WITH GRANT OPTION]
CREATE, ALTER, DROP	TRIGGER	Права определяются исходя из прав на таблицу/представление
CREATE, ALTER, DROP	PROCEDURE	GRANT CREATE ALTER DROP [ANY] PROCEDURE TO USER ROLE [WITH GRANT OPTION]
CREATE, ALTER, DROP	VIEW	GRANT CREATE ALTER DROP [ANY] VIEW TO USER ROLE [WITH GRANT OPTION]
CREATE, ALTER, DROP	DOMAIN	GRANT CREATE ALTER DROP [ANY] DOMAIN TO USER ROLE [WITH GRANT OPTION]
CREATE, ALTER, DROP	ROLE	GRANT CREATE ALTER DROP [ANY] ROLE TO USER ROLE [WITH GRANT OPTION]
CREATE, ALTER, DROP	GENERATOR	GRANT CREATE ALTER DROP [ANY] GENERATOR TO USER ROLE [WITH GRANT OPTION]
CREATE, ALTER, DROP	SEQUENCE	GRANT CREATE ALTER DROP [ANY] SEQUENCE TO USER ROLE [WITH GRANT OPTION]
CREATE, ALTER, DROP	EXCEPTION	GRANT CREATE DROP [ANY] EXCEPTION TO USER ROLE [WITH GRANT OPTION]
CREATE, DROP	SHADOW	GRANT CREATE DROP [ANY] SHADOW TO USER ROLE [WITH GRANT OPTION]
DECLARE, ALTER, DROP	FUNCTION	GRANT CREATE ALTER DROP [ANY] FUNCTION TO USER ROLE [WITH GRANT OPTION]
CREATE, ALTER, DROP	INDEX	Права определяются исходя из прав на таблицу
CREATE, DROP	POLICY	GRANT SECADMIN TO USER ROLE ¹

¹ DDL-операции с политиками может осуществлять только пользователь SYSDBA и пользователь с ролью глобального администратора безопасности — SECADMIN (подробнее о глобальных ролях см. гл. 4 «Предопределенные роли»).

3. Доступ к операциям манипулирования данными (DML access)

Суть доработки заключается в обеспечении полного разграничения доступа на операции DML: чтение данных из полей таблиц, установки и чтения значений генераторов и выполнения процедур.

Назначение прав на доступ к DML-операциям выполняется с помощью оператора GRANT, для этого расширяется уже существующий функционал GRANT и соответствующий ему REVOKE (для отбирания прав).

В таблице 2 приведены языковые конструкции для выполнения DML-операций над данными.

Таблица 2. - Назначение прав на DML операции.

Операция	Объект	Назначение прав
SELECT, INSERT, UPDATE, DELETE	TABLE, VIEW	GRANT SELECT INSERT UPDATE DELETE ON [TABLE] {table} TO {user role} [WITH GRANT OPTION] REVOKE SELECT INSERT UPDATE DELETE ON [TABLE] {table} FROM {user role} REVOKE GRANT OPTION FOR SELECT INSERT UPDATE DELETE ON [TABLE] {table} FROM {user role}
SET GENERATOR, функция GEN_ID	GENERATOR	GRANT SELECT UPDATE ON GENERATOR {generator} TO {user role} [WITH GRANT OPTION] REVOKE SELECT UPDATE ON GENERATOR {generator} FROM {user role} REVOKE GRANT OPTION FOR SET GET ON GENERATOR {generator} FROM {user role}
SELECT, INSERT, UPDATE	TABLE (столбец, ...), VIEW (столбец, ...)	GRANT SELECT INSERT UPDATE {{ column [, ...] }} ON [TABLE] {table} TO {user role} [WITH GRANT OPTION] REVOKE SELECT INSERT UPDATE {{ column [, ...] }} ON [TABLE] {table} FROM {user role} REVOKE GRANT OPTION FOR SELECT INSERT UPDATE {{ column [, ...] }} ON [TABLE] {table} FROM {user role}
EXECUTE	PROCEDURE	GRANT EXECUTE ON PROCEDURE {procedure} TO {user role} [WITH GRANT OPTION] REVOKE EXECUTE ON PROCEDURE {procedure} FROM {user role} REVOKE GRANT OPTION FOR EXECUTE ON PROCEDURE {procedure} FROM {user role}

4. Предопределенные роли (Default Roles)

Добавляются глобальные (действующие для всех баз сервера) роли SYSADMIN, SECADMIN, PUBLIC. и определяются их права

Таблица 3 - Права доступа (по умолчанию) предопределенных ролей

Объект / Роль	SYSADMIN	SECADMIN	PUBLIC
Операции DML	+	+/- (Всегда может назначить необходимые привилегии)	Согласно назначенных прав на операции с данными
Операции DDL	+	Только операции с ролями	-
Операции GRANT/REVOKE	+	+	-
Резервное копирование/восстановление (GBAK)	+	-	-
Операции с пользователями (GSEC)	-	+	-
Операции GFIX и GSTAT	+	-	-

Принцип работы PUBLIC остается прежним, т.е. права данной роли будут иметь все пользователи. Эта роль назначается пользователю автоматически при создании и по умолчанию не имеет прав на работу с данными и метаданными в БД.

Согласно таблице прав доступа предопределенных ролей, SYSADMIN имеет полные права на базу данных, но не может производить операции с пользователями и назначать роли пользователям.

Роль SECADMIN имеет полные права на операции назначения/отбора (GRANT/REVOKE) прав и поэтому имеет больше полномочий, чем SYSADMIN.

Вводится поддержка глобальных ролей, т.е. ролей хранящихся в security2.fdb.

Примечание: Отличие глобальных ролей от локальных в том, что последние назначаются при прямом соединении к базе данных безопасности security2.fdb.

Если пользователю в security2.fdb назначена какая-либо роль, то он является ее обладателем во всех базах данных сервера. Назначать глобальные роли пользователям может только SYSDBA или глобальный SECADMIN (при прямом подключении к security2.fdb).

5. Кумулятивное (комбинированное) действие ролей (Combine Roles)

Вводится возможность добавления группы пользователей в другую группу. Для этого необходимо назначить роль на другую роль. Вводятся следующие расширения операторов GRANT и REVOKE:

```
GRANT ROLE1 TO ROLE2;  
REVOKE ROLE1 FROM ROLE2;
```

Циклические ссылки ролей друг на друга недопустимы.

Правила кумулятивного действия ролей:

- если пользователь не указывает роль при подключении к серверу, то он получает права всех ролей, которые ему назначены;
- если пользователь при подключении указал конкретную роль, то он получает только её привилегии;
- при подключении происходит проверка, что данная роль существует и назначена данному пользователю.

Появляется возможность выполнения процедуры с правами владельца, а не только с правами вызывающего ее пользователя. Таким образом, пользователю для выполнения процедуры нужно будет иметь привилегию только на выполнение процедуры, но не права на объекты, с которыми работает эта процедура (эти права должны быть у владельца процедуры).

То, в контексте безопасности какого пользователя будет выполняться процедура, определяется при ее создании/изменении - в объявлении процедуры добавлена опция AUTHID OWNER|CALLER. По умолчанию параметр равен OWNER, т.е. процедура выполняется с правами её владельца. Синтаксис объявления процедуры следующий:

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

Вводится запрет на совпадение имён пользователей и ролей. Исключить такие совпадения невозможно, так как роли хранятся непосредственно в БД, а пользователи – в базе данных безопасности, поэтому при переносе БД с одного сервера на другой возможны совпадения имён пользователей и ролей. При таком совпадении действует следующее правило: права всегда назначаются на роль в первую очередь, затем, если роль не найдена – на пользователя.

6. Многофакторная аутентификация (Authentication)²

Особенности многофакторной аутентификации:

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

Вторичные факторы могут быть предъявлены после первичной аутентификации по защищенному каналу, установленному в результате первичной аутентификации. Вторичные факторы могут быть любыми, например данные биометрии. Их передача шифруется ключами, выработанными в результате первичной аутентификации. Ведутся доработки для использования следующих вторичных факторов аутентификации: отпечатки пальцев, сетчатки глаза и т.д.

При первичной аутентификации сервер проверяет первичные факторы и в результате аутентификации сервер и клиент вырабатывают сеансовые ключи. Далее клиент имеет возможность предъявить вторичные факторы, зашифрованные сеансовыми ключами.

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

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

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

Таблица 4 - Параметры конфигурационного файла, имеющие отношение к механизму многофакторной аутентификации:

Параметр	Возможное значение	Комментарий
Authentication	native trusted mixed multifactor	native — режим совместимости с более ранними версиями trusted — разрешена только trusted аутентификация (аутентификация Windows) multifactor — только многофакторная аутентификация mixed — любой из трех предыдущих вариантов аутентификации
LegacyHash	0 1	Позволяет пользователям которые созданы как не мультифакторные соединятся с базой при включенном режиме Authentication=multifactor

² Для работы данной функции необходимо наличие криптопровайдера и криптоплагина (входит в поставку СУБД «Ред База Данных»). Криптопровайдер реализует функции шифрования и хеширования, а криптоплагин реализует унифицированный интерфейс между криптопровайдером и сервером СУБД «Ред База Данных» в качестве криптопровайдера в настоящее время используется CryptoPro. Также идут работы по использованию встроенных средств Windows CryptoAPI. Однако в текущей версии рекомендуется использовать CryptoPro.

Параметр	Возможное значение	Комментарий
CryptoPluginName	Значение по умолчанию - crypto_plugin	Имя библиотеки криптопровайдера (расположена в каталоге plugins сервера)
KeyRepository	<имя контейнера>	Имя контейнера с ключами для установки защищенного соединения
CertRepository	<имя сертификата>	Имя контейнера с сертификатом для проверки подлинности сервера

7. Политики безопасности (Login policies)

Политики учетных записей включают в себя:

- требования к сложности пароля при его задании;
- количество предыдущих паролей, которые не должен повторять вновь заданный;
- срок действия пароля;
- количество одновременно открытых сессий пользователя;
- продолжительность простоя пользователя до отключения.

Политика определяет реакцию системы на неудачные попытки входа в систему и дает возможность заблокировать пользователя временно или постоянно.

Это позволяет управлять сложностью пароля и общей защищенностью базы данных. Также политика не позволяет производить подбор пароля и блокирует пользователя, от имени которого может производиться атака сервера.

Специальная сервисная утилита `idlectrl` реализует следующие функции:

- проверяет пользователей, которые не обращались к БД дольше, чем определено политикой доступа;
- периодически проверяет соответствие числа сессий каждого пользователя допустимому числу, определяемому политикой;

Использование политик безопасности возможно при соблюдении следующих ограничений:

- Действие политик происходит только при использовании многофакторной аутентификации пользователя. При обычном подключении, политики не проверяются.
- Политики учетных записей создаются в базе `security2.fdb`. Каждому пользователю соответствует политика. Новым пользователям соответствует политика по умолчанию с именем `DEFAULT`.
- Проверка сложности пароля возможна только в режиме защищенной аутентификации, при которой вырабатываются сеансовые ключи передачи данных. При незащищенной аутентификации проверка сложности пароля не производится. При смене пароля он шифруется симметричным шифрованием, с помощью сеансовых ключей, выработанных после аутентификации пользователя. На приемной стороне пароль дешифруется и проверяется на соответствие политике учетных записей сервера. После этого вычисляются и сохраняются в базе пользователей хеш пароля и алгоритм, которым этот хеш получен.
- При подключении, если политикой доступа требуется пароль, проверяется срок действия пароля и если пароль правилен, но его срок действия истек, то система отвергает подключение, выдавая сообщение о необходимости смены пароля. При подключении проверяется существующее количество сессий у подключающегося пользователя в целом для сервера (не только для указанной базы). Если оно равно максимальному, то подключение отвергается, выдавая соответствующее сообщение. Неудачные попытки доступа увеличивают счетчик неудачных попыток доступа для каждого конкретного пользователя. Счетчик хранится вместе с информацией о пользователе в базе данных `security2.fdb`. При неудачной попытке доступа, в зависимости от текущего значения счетчика, пользователь блокируется на определенное время либо навсегда, если значение счетчика превысило определенный политикой предел. Время, после которого пользователь может под-

ключаться к БД прописывается вместе с информацией о пользователе и счетчиком неудачных попыток доступа.

- Управление политиками осуществляется с помощью новых операторов DDL (см. ниже)
- Изменение параметров политики подтверждается сразу и вступают в силу для обработки последующих подключений.

Для управления политиками и параметрами пользователей, добавлены следующие команды DDL

```
CREATE POLICY <policy_name> AS [param = value [, param = value]];
DROP POLICY <policy_name>;
ALTER POLICY <policy_name> AS [param = value [, param = value]];
GRANT POLICY <policy_name> TO <user_name>;
```

Оператор REVOKE POLICY – отсутствует, т.к. пользователю всегда должна быть назначена хотя бы одна политика. Вместо REVOKE POLICY необходимо делать назначение пользователю политики по умолчанию.

```
GRANT POLICY "DEFAULT" TO <user_name>;
```

Факторы аутентификации задаются условным выражением из символических обозначений. Выражение является объединением возможных наборов необходимых факторов аутентификации и позволяет задавать различные варианты требуемых факторов с помощью операций «И», «ИЛИ».

Таблица 5 — Символическое обозначение факторов аутентификации

Символическое обозначение	Расшифровка
W	WINDOWS_NTLM (результат аутентификации в Windows)
C	CERT_X509 (сертификат пользователя)
P	PASSWORD (пароль)
F	FINGERPRINT (отпечатки пальцев)
E	EYE (снимок сетчатки глаза)

Выражения требуемых факторов аутентификации задаются в виде: (A1A2... Ai) | (B1B2... Bm) | ... | (Z1Z2 ... Zn) — где Ax, Bx, Zx — символические обозначения факторов аутентификации.

Таблица 6 — Параметры политик безопасности

Название	Тип	Расшифровка
AUTH_FACTORS	auth_factors_expr_list	см. табл. 4
PSWD_NEED_CHAR	long_integer	Минимально допустимое число буквенных символов в пароле.
PSWD_NEED_DIGIT	long_integer	Минимально допустимое число цифровых символов в пароле
PSWD_NEED_DIFF_CASE	bool_const	Необходимость наличия в пароле буквенных символов в различных регистрах
PSWD_MIN_LEN	long_integer	Минимально допустимая длина пароля
PSWD_VALID_DAYS	long_integer	Срок действия пароля в днях
PSWD_UNIQUE_COUNT	long_integer	Количество не повторяющихся подряд паролей

Название	Тип	Расшифровка
MAX_FAILED_COUNT	long_integer	Максимальное число ошибок при прохождении аутентификации
MAX_SESSIONS	long_integer	Максимальное число одновременно запущенных сессий
MAX_IDLE_TIME	long_integer	Время бездействия пользователя до отключения сессии в секундах

8. Обезличивание памяти (Memory Elimination)

Оперативная память, используемая в работе сервера, при освобождении обезличивается. Удаляемые в процессе работы сервера записи и страницы данных, подлежат обезличиванию до того, как будут записаны на диск.

Данные базы данных потенциально могут храниться в 3-х местах:

- непосредственно записи таблиц;
- индексы, построенные по полям таблиц и соответственно содержащие ключевые значение из них;
- BLOB-поля, которые могут храниться либо на странице данных, если их размер достаточно мал, либо на отдельной странице, предназначенной для хранения BLOB.

Таким образом, для полного контроля обезличивания памяти необходимо контролировать:

- удаление любой записи данных;
- удаления записи индекса;
- удаление страницы данных;
- удаление страницы индекса;
- удаление страницы BLOB-данных.

Важная особенность СУБД «Ред База Данных» — версионная архитектура. Это означает, что при изменении записи, создается новая версия записи. Предыдущая версия остается существовать. Каждая транзакция, стартовавшая на сервере имеет свой номер. Запись также имеет номер создавшей ее транзакции. Таким образом, каждая транзакция может видеть свою версию записи и иметь к ней и именно к ней интерес.

Требование обезличивания памяти не распространяется на такие версии записей, которые интересны и используются какой-либо транзакцией. В том случае, если запись не интересна ни одной транзакции, т.е. любая из открытых транзакций не получит эту версию записи, то эта версия записи удаляется. В этот момент будет происходить обезличивание памяти, ранее занятой данной версией записи.

Для активации режима обезличивания необходимо в файле конфигурации выставить параметр `MemoryWipePasses = 1` Целочисленное значение, настраивающее необходимость и метод обезличивания освобождаемой памяти. Возможные значения:

- 0 — обезличивание не происходит;
- 1 — происходит обнуление памяти в один проход;
- n (n>1) — n проходов с записью в блок поочередно 0xFF и 0x00, последний проход при этом всегда заполняет блок нулями.

9. Аудит

FBTrace - это утилита для слежения за работой сервера «Ред База Данных». Она отслеживает события соединения и отсоединения от БД, операции DML и DDL, выполнение хранимых процедур и т.д., в зависимости от параметров, заданных в ее конфигурационном файле. Утилита стартует одновременно с сервером «Ред База Данных» и ведет лог-файлы для каждой из отслеживаемых баз данных. По умолчанию лог-файлы размещаются в том же каталоге что и отслеживаемая (логируемая) БД и имеют имя следующего вида: <имя_базы.fbtrace_text> или <имя_базы.fbtrace_bin> для текстового и бинарного формата лога соответственно. Запись в лог для каждой конкретной БД начинается с момента ее создания или присоединения к ней и до момента отсоединения или ее удаления. Регистрируются события, завершившиеся как удачно, так и неудачно (с ошибкой). Это относится к присоединению к БД, предъявлению фактора аутентификации, подготовке и выполнению запроса, выполнению хранимой процедуры, завершению транзакции.

Типы и параметры событий аудита

При ведении лога регистрируются следующие события:

- начало и окончание ведения аудита для БД;
- присоединение к БД и отсоединение от нее, присоединение к сервису и отсоединение от него, старт сервиса, предъявление фактора аутентификации;
- подготовка, выполнение и освобождение запроса к БД, запрос к сервису, компиляция BLR;
- выполнение в БД хранимой процедуры;
- установка значения контекстной переменной;
- начало и завершение транзакции.

Для каждого события, кроме предъявления фактора аутентификации, сохраняются:

- дата, время и тип события;
- идентификатор процесса, вызвавшего событие;
- результат (успешно, не успешно, не санкционировано);
- сведения о соединении:
 - путь к БД;
 - идентификатор соединения;
 - пользователь, от имени которого совершается действие;
 - протокол соединения;
 - имя компьютера, с которого соединение установлено.

Для каждого из аудируемых событий записывается следующая информация:

1. Для присоединения к БД указывается, происходит ли подключение к существующей БД, либо создается новая база. При отключении от базы определяется произошло ли просто отключение, либо база была удалена.
2. Для фактора аутентификации указывается, дата, время и тип события; идентификатор процесса, вызвавшего событие; имя пользователя; тип фактора; сам предъявленный фактор (содержимое зависит от типа); результат предъявления (успешно, не успешно, не санкционировано).

3. Для присоединения к сервису, отсоединения от него, старта сервиса указываются сведения, общие для всех событий, кроме пути к БД и идентификатора соединения. Вместо них указывается имя сервиса.
4. Для события старта сервиса также указываются параметры старта (опции командной строки, с которыми он был запущен).
5. Для запроса к сервису также указывается имя сервиса, кроме того, сохраняется содержимое запроса, который был передан сервису.
6. Для установки значения контекстной переменной – сведения о транзакции, пространство имен (namespace), для которого она устанавливается, имя переменной, а также ее значение.
7. Для хранимой процедуры – сведения о транзакции, имя процедуры, входные параметры процедуры, время выполнения, статистика производительности.
8. Для транзакции – идентификатор транзакции и ее параметры (уровень изоляции, режим блокировки). При событии завершения транзакции также сохраняется результат выполнения – commit для подтвержденных транзакций и rollback если был произведен откат.
9. Для подготовки SQL-запроса – сведения о транзакции, идентификатор запроса, содержимое запроса, время подготовки запроса (в мс), план выполнения.
10. Для выполнения SQL или BLR запроса – сведения о транзакции, идентификатор запроса, время выполнения запроса (в мс), статистика производительности, параметры запроса, содержимое запроса (в случае текстового формата лога).
11. Для выполнения DYN-запроса – сведения о транзакции, время выполнения запроса (в мс), содержимое запроса (в текстовом представлении для текстового лога, в бинарном – для бинарного).
12. Для компиляции BLR-запроса - сведения о транзакции, идентификатор запроса, содержимое запроса (в текстовом представлении для текстового лога, в бинарном – для бинарного), время подготовки запроса (в мс).

Примечание: Несанкционированной попыткой выполнения действия считается такая, при которой не была пройдена аутентификация либо не оказалось прав на выполнение действия. Не успешной – любая другая неудачная попытка (закончившаяся ошибкой).

Примечание: По умолчанию система аудита выключена.

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

Примечание: Используется система ротации логов, которая активизируется по достижении файлом журнала аудита заданного пользователем максимального размера. При этом рабочий лог-файл переименовывается в файл с именем <log_filename>.<текущая дата и время>.<log_ext>, где дата и время записываются в виде <YYYY-MM-DDThh-mm-ss>, <log_ext> – расширение лог-файла. После переименования рабочего лога, создается новый файл с именем переименованного. Этот новый файл используется в дальнейшем в качестве рабочего лога. Удаление или архивирование старых лог-файлов не предусмотрено и может осуществляться средствами ОС и планировщиками задач.

Используется понятие версии формата бинарного лог-файла. Она проверяется при инициализации аудита (событие начала ведения аудита), если лог-файл уже существует и имеет бинарный формат. Если версия формата лога, используемая в «Ред Базе Данных», отличается от версии формата существующего файла, то производится ротация (переименование существующего лога, создание рабочего лог-файла с таким же именем).

Настройка аудита. Параметры конфигурационного файла

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

Строка, следующая за символом # считается комментарием. В параметрах, значения которых допускают использование регулярных выражений, используется синтаксис регулярных выражений POSIX 1003.2

В конфигурационном файле настраиваются следующие параметры:

enable = 0/1 – вести лог-файл или нет. 1 – лог-файл ведется, 0 – не ведется (по умолчанию).

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

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

max_sql_length = <число> - максимальная длина одной записи SQL-запроса в лог-файле, в байтах. Значение по умолчанию – 300 байт, максимальное значение – 64К. Если длина запроса больше указанного здесь значения, запрос будет обрезан. Применяется только для текстового формата лог-файла³.

max_blr_length = <число> максимальная длина BLR-запроса, сохраняемого в лог, в байтах. Значение по умолчанию – 500 байт. Максимальное значение – 64К. Если длина запроса больше указанного здесь значения, запрос будет обрезан. Применяется только для текстового формата лог-файла.

max_dyn_length = <число> максимальная длина DYN-запроса, сохраняемого в лог, в байтах. Значение по умолчанию – 500 байт. Максимальное значение – 64К. Если длина запроса больше указанного здесь значения, запрос будет обрезан. Применяется только для текстового формата лог-файла.

max_arg_length = <число> – максимальная длина одного аргумента в лог-файле. Значение по умолчанию – 80 в байтах. Максимальное значение – 64К. Если длина аргумента больше указанного здесь значения, аргумент будет обрезан. Применяется только для текстового формата лог-файла.

max_arg_count = <число> - максимальное количество аргументов в запросе, которые вносятся в лог-файл. Значение по умолчанию – 30. Аргументы, номера которых больше указанного здесь значения, отображаться не будут. Применяется только для текстового формата лог-файла.

include_filter = <регулярное выражение> – этот параметр определяет те виды SQL запросов, которые будут включаться в лог-файл. По умолчанию - не чувствителен к регистру. Под фильтр подпадают только те значения, которые удовлетворяют синтаксису регулярных выражений. Значение по умолчанию — пусто, то есть в конечный лог будут включены все запросы.

exclude_filter = <регулярное выражение> – определяет виды SQL запросов, которые не включаются в лог-файл. Аналогично **include_filter**. Значение по умолчанию — пусто.

log_connections = 0/1 – определяет, записывать ли события присоединения/отсоединения к БД в лог-файл. 1 – события присоединения/отсоединения записываются, 0 – не записываются. Значение по умолчанию — 0 (обрабатывается только при использовании текстового формата лога).

log_transactions = 0/1 – определяет, записывать ли события начала и конца транзакций в лог-файл. 1 – события начала и конца транзакций записываются, 0 – не записываются. Значение по умолчанию — 0 (обрабатывается только при использовании текстового формата лога)⁴

3 Для бинарного формата лог-файла запрос будет записан в лог целиком, однако при подключении такого лога к базе данных максимальный размер запросов не может превышать размер страницы БД. В противном случае запрос будет обрезан.

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

log_statements = 0/1 - запись операторов подготовки и освобождения записей в лог-файл. 1 – операторы подготовки и освобождения записей заносятся в лог-файл, 0 – не заносятся. Значение по умолчанию — 0 (обрабатывается только при использовании текстового формата лога)

log_context = 0/1 – запись изменений значений контекстных переменных в лог-файл. 1 – изменения в контекстных переменных записи заносятся в лог-файл, 0 – не заносятся. Значение по умолчанию — 1 (обрабатывается только при использовании текстового формата лога)

log_execute = 0/1 – запись операторов выполнения/выборки записей. 1 – операторы выполнения/выборки записей заносятся в лог-файл, 0 – не заносятся. Значение по умолчанию — 1 (обрабатывается только при использовании текстового формата лога)

max_log_size = <число> задает максимальный размер log-файлов в мегабайтах. Если значение параметра не задано или равно 0, то размер файла журнала не ограничен, ротация логов не используется. Значение по умолчанию — 0.

print_blr = 0/1 если параметр установлен в 1, то содержимое BLR-запросов будет преобразовываться в текстовое представление. Если параметр установлен в 0, то BLR-запрос будет сохранен в двоичном виде (последовательность байт). Этот параметр будет работать только для текстового формата лога. В бинарном формате содержимое BLR всегда будет сохраняться в двоичном виде. Значение по умолчанию — 1.

print_dyn = 0/1 если параметр установлен в 1, то содержимое DYN-запросов будет преобразовываться в текстовое представление. Если параметр установлен в 0, то DYN-запрос будет сохранен в двоичном виде (последовательность байт). Этот параметр будет работать только для текстового формата лога. В бинарном формате содержимое DYN всегда будет сохраняться в двоичном виде. Значение по умолчанию — 1.

log_procedures = 0/1 параметр указывает на необходимость регистрации хранимых процедур. Значение по умолчанию — 0.

log_blr_requests = 0/1 параметр, отвечающий за регистрацию прямого выполнения BLR. Значение по умолчанию — 0.

log_dyn_requests = 0/1 параметр, отвечающий за регистрацию прямого выполнения DYN. Значение по умолчанию — 0.

log_services = 0/1 параметр указывает на необходимость регистрации вызовов сервисов. Значение по умолчанию — 0.

log_filename = <строка> - имя лог файла. Если этот параметр не задан, лог файл создаётся в той же папке, где находится БД и имеет имя вида <имя_базы.fbtrace_text> или <имя_базы.fbtrace_bin>. Возможно использование регулярных выражений в этом параметре. Например, `log_filename=$1.log`, означает использование файла лога с именем, совпадающим с полным именем аудируемой БД и с расширением log (данный пример будет корректно работать только в случае задания имени БД в виде регулярного выражения). При явном указании имени файла, все события от всех логируемых БД будут сохраняться в данном файле.

При помощи регулярных выражений можно задавать конкретные БД для логирования. Примеры конфигурационных файлов аудита:

Пример 1:

```
#Лог ведётся для баз с именами test.fdb, azk2.fdb,
rules.fdb,
[^\.*[\/](test|azk2|rules)\.fdb$]:
```

```
enabled = 1

#Логи сохраняются в файлы с именами test.log, azk2.log,
rules.log соответственно

log_filename = $1.log
```

Пример 2:

```
#Для всех БД на диске С с расширением fdb – формат лога
текстовый
[^C:.*\.fdb$]:
enabled = 1

#Формат лога – текстовый
format = 0

#

#Для всех БД с расширением fdb на диске D – формат лога
бинарный
[^D:.*\.fdb$]:
enabled = 1

format = 1
```

В регулярных выражениях можно использовать группировку - ()

Пример 3:

```
#Первая группа (.*[\/]) – любой путь к файлам БД
#Вторая группа (test|azk) – имя файла БД test или azk
[^(.*)[\/](test|azk)\.fdb$]:
enabled = 1

#$1 – Первая группа из регулярного выражения, путь.
#$2 – Вторая группа из регулярного выражения, имя файла
log_filename = $1../logs/$2.log
```

То есть будут создаваться логи с именами <имя_базы.log> в каталоге logs расположенном на одном уровне с каталогом содержащим базы.

Пример 4:

```
#В лог файл будут записываться все события для всех БД
enabled = 1

format = 0

time_threshold = 0

max_sql_length = 300

max_arg_length = 80

max_arg_count = 30

log_connections = 1
```

```
log_transactions = 1
log_statements = 1
log_context = 1
log_execute = 1
log_procedures = 1
log_auth_factors = 1
log_blr_requests = 1
log_dyn_requests = 1
log_services = 1
```

10. Адаптер для подключения бинарного файла аудита (LogAnalyzer)

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

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

Инструмент анализа разбирает бинарные лог-файлы системы аудита «Ред Базы Данных» 2.1 и предоставляет возможность просмотра и фильтрации записей о событиях. Удаление и редактирование записей не допускается.

Для работы с файлами аудита используется следующий механизм:

- подключение файла журнала к базе «Ред Базы данных» 2.1 в качестве внешней таблицы;
- просмотр, поиск и фильтрация записей с использованием встроенных средств СУБД (построение и выполнение запросов к подключенной таблице).

Это позволяет применять средство анализа файлов аудита независимо от используемой операционной системы, а также наличия графической оболочки в ОС. Подключение журнала производится с помощью SQL-запроса вида:

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

где:

- table_name – имя таблицы, которая будет хранить данные аудита;
- filespec – имя лог-файла, который должен быть открыт;
- ADAPTER – ключевое слово, необходимое для подключения в качестве внешней таблицы файла с нестандартным форматом данных;
- fbtrace – название адаптера, предназначенного для обработки бинарного лога системы аудита;
- col_defs – описание полей таблицы аудита.

Некоторые поля могут быть пустыми в зависимости от типа события.

Так как в случае подключения лог-файла структура таблицы заранее известна, то при указании ключевого слова ADAPTER определение ее полей не требуется.

- Если пользователь указывает тип адаптера без перечисления полей таблицы, создается таблица соответствующего адаптера. Если же в запросе одновременно задается и тип адаптера, и структура таблицы, перечисленные поля должны быть подмножеством полей таблицы адаптера. Названия полей и их типы также жестко определяются типом адаптера. Порядок объявления полей не учитывается.
- Если одно из полей задано неверно (ему не соответствует ни одно поле в таблице адаптера), выдается ошибка, выполнение запроса прерывается.
- Если структура таблицы указана верно, то не перечисленные в ней поля таблицы адаптера игнорируются.
- При открытии бинарного лог-файла определяется версия его формата. Если она отличается от номера версии, которая является рабочей для данного релиза, пользователю выдается соответствующая ошибка.

- Если производится попытка использования адаптера в БД, ODS которой не поддерживает этого, пользователю выдается соответствующая ошибка.

Таблица 7 - Поля создаваемой таблицы аудита:

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

11. Контроль целостности метаданных⁵

Контроль за целостностью метаданных в БД осуществляется следующим образом в состав сервера входит утилита `mint` (находиться в каталоге `bin/` установки сервера). Эта утилита предназначена для извлечения и хеширования метаданных из баз данных. Таким образом администратор может защитить структуру базы данных от изменений.

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

Утилита имеет следующие команды и опции:

```
-M <extract|check>
[<-m маска>]
-d <имя базы данных>
-u имя пользователя для присоединения к базе данных
-p <пароль пользователя для присоединения к базе данных>
-r <хранилище с ключами шифрования>
-R <алгоритм шифрования для хранилища ключей>
-i <файл для сохранения|проверки ЭЦП метаданных>
-I <название алгоритма цифровой подписи>
```

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

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

Например:

генерация подписи для всех системных объектов (начинающихся с префикса `RDB$`):

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

проверка подписи для этих объектов:

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

⁵ См. примечание 2 (Глава 6 «Многофакторная аутентификация»)

12. Контроль целостности файлов сервера⁶

Контроль за файлами сервера означает, что для всех критически важных файлов сервера (бинарные файлы, файлы конфигурации, база данных безопасности security2.fdb) может быть вычислен хеш. Файл с контрольными суммами (хеш-функциями) всех защищаемых файлов, а также файл конфигурации «Ред Базы Данных» должны быть защищены с помощью организационно-технических мер (например запись на носителе только для чтения и т.д.) Файл с контрольными суммами поставляется вместе с дистрибутивом СУБД «Ред База Данных»

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

При загрузке сервер загружает все строки из файла хешей и последовательно проверяет каждый. Файл хешей должен содержать строки вида:

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

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

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

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

Пример использования утилиты hashgen:

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

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

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

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

⁶ См. примечание 2 (Глава. 6 «Многофакторная аутентификация»)

13. Функциональные доработки

Архитектура суперклассик

СУБД «Ред База Данных» 2.1 поддерживает режим многопоточного классик сервера — суперклассик архитектура. В этом режиме создается один процесс сервера для всех соединений, при этом сервер может эффективно распараллеливать соединения между различными ядрами/процессорами.

Для использования других функциональных возможностей, а именно:

- внешние процедуры на Java;
- отладчик хранимых процедур и триггеров;
- система полнотекстового поиска.

Необходимо наличие JDK 1.6 Кроме этого, в базу данных для которой предполагается использовать полнотекстовый поиск и/или отладчик хранимых процедур, необходимо добавить служебные таблицы и процедуры, которые содержатся в файле Init.sql. Этот файл расположен в корне каталога установки «Ред Базы данных» 2.1. Для того, чтобы автоматически пролить этот файл при создании БД необходимо указать этот файл в конфигурационном файле fitrebird.conf (параметр InitScript)

Процедуры и триггеры на Java

Объявление процедуры

```
CREATE PROCEDURE procedure_name
[(param <datatype> [, param <datatype> ...])]
[RETURNS (param <datatype> [, param <datatype> ...])]
LANGUAGE language
EXTERNAL NAME 'unique_procedure_identifier'
```

где

- procedure_name - уникальное допустимое имя процедуры
- param <datatype> - допустимые имена и типы входных и возвращаемых параметров.
- LANGUAGE - в нашем случае этот параметр должен иметь значение JAVA.
- unique_procedure_identifier - выражение, содержащее уникальное имя внешней процедуры, которое понимает внешний модуль ESP. В случае Java-процедуры оно содержит имя пакета, имя класса и имя метода.

Объявление функции:

```
DECLARE EXTERNAL FUNCTION <function name>
[<datatype> [, <datatype> ...]]
RETURNS <datatype>
LANGUAGE language
EXTERNAL NAME 'unique_function_identifier'
```

где

- function_name - уникальное допустимое имя UDF

- <datatype> - допустимые типы входных и возвращаемых параметров.
- LANGUAGE - в нашем случае этот параметр должен иметь значение JAVA.
- unique_function_identifier - выражение, содержащее уникальное имя UDF, которое понимает внешний модуль ESP.

Представленный вариант регистрации ESP и UDF отличается от стандарта SQLJ, но он более близок к текущему варианту описания SP в Firebird.

Таблица 8 - Соответствие поддерживаемых типов данных параметрах и возвращаемых значениях.

Тип SQL	Примитивный тип	Объектный тип
CHAR	-	String
VARCHAR	-	String
NUMERIC	-	java.math.BigDecimal
SMALLINT	short	Short
INTEGER	int	Integer
BIGINT	long	Long
FLOAT	float	Float
DOUBLE	double	Double
BLOB	-	org.firebirdsql.jdbc.FirebirdBlob
DATE	-	java.sql.Date
TIME	-	java.sql.Time
TIMESTAMP	-	java.sql.Timestamp

Любой public static method, принадлежащий к public классу (не внутреннему inner классу), с параметрами допустимого типа (BLOB, примитивные типы или обернутые примитивные типы) можно использовать как Java stored procedure.

Для executable procedure метод обязан возвращать void.

Любой public static method, принадлежащий к public классу и возвращающий объект типа java.sql.ResultSet, может быть использован как процедура, возвращающая данные. ResultSet может быть получен как результат внутреннего запроса, запроса к внешней базе данных или другим способом.

Получить доступ к контексту выполнения, откуда была вызвана Java ESP, можно следующим образом:

```
Connection conn =
    DriverManager.getConnection("jdbc:default:connection:");
```

Это позволит, например выполнить DDL, или другую операцию требующую автономности для выполнения.

Объявлять триггер, написанный на java нельзя. Но можно вызвать Java ESP из триггера, написанного на PSQL. При этом из процедуры можно получить доступ к контексту триггера:

- Получение имени таблицы, для которой вызван триггер
- Получение операции, вызвавшей триггер. Возможные значения:
 - ACTION_INSERT
 - ACTION_UPDATE
 - ACTION_DELETE
- Получение значения "new.<name>" в виде объекта

- Получение значения "old.<name>" в виде объекта
- Установка значения "new.<name>" в виде объекта

Пример использования внешних Java-процедур:

```
CREATE PROCEDURE FIELDCOUNT
RETURNS (FIELD_NAME VARCHAR(50), COUNTS INTEGER )
LANGUAGE JAVA
EXTERNAL NAME
'firebird.java.example.Examples.fieldNames';

CREATE PROCEDURE FIELDCOUNTEXTERN (
    dburl          VARCHAR(256) ,
    user_name      VARCHAR(256) ,
    password       VARCHAR(256)
)
RETURNS (FIELD_NAME VARCHAR(50), COUNTS INTEGER)
LANGUAGE JAVA
EXTERNAL NAME
'firebird.java.example.Examples.fieldNames';
```

Отладчик хранимых процедур и триггеров

Отладчик предназначен для пошагового выполнения хранимых процедур и триггеров. Функционально отладчик состоит из трех частей:

- сервер отладки;
- клиент сервера отладки в процессе сервера «Ред База Данных»;
- графический интерфейс клиента отладки.

Для регистрации клиента в сервере отладки из отлаживаемого приложения должна быть вызвана хранимая процедура `execute procedure dbg$update`;

При использовании архитектуры классик для запуска сервера отладки необходимо выполнить `debug_srv.cmd/debug_srv.sh`.

Для запуска отладчика используется команда `debug_gui.cmd/debug_gui.sh`

После этого запускается графический интерфейс отладчика, в котором имеется возможность подключиться к одной из баз данных, зарегистрированных в сервере отладки, просмотреть доступные процедуры и триггеры. Для отладки той или иной процедуры необходимо установить в ней в нужных местах точки останова и выполнить эту процедуру (триггер). Вызов процедуры непосредственно из отладчика невозможен.

Для возможности отладки процедуры должны быть перекомпилированы под текущим сервером. Процедуры скомпилированные под сервером Firebird будут выполняться, но отладка для них не будет возможна.

В отладчике используется формирование логов для Java, см.

<http://java.sun.com/javase/6/docs/technotes/guides/logging/overview.html>

Для связи между GUI клиентом и сервером отладки используется RMI

См <http://java.sun.com/javase/6/docs/platform/rmi/spec/rmiTOC.html> . По умолчанию порт используемый RMI 1099.

Полнотекстовый поиск позволяет:

- производить поиск по неточному условию;
- производить морфологический поиск;
- производить поиск по нескольким метаобъектам БД (поиск по нескольким столбцам и таблицам);

Поиск может осуществляться по текстовым полям, текстовым блокам и по блокам содержащим внутри себя документы следующих приложений:

- Acrobat (pdf)
- MS Word (doc)
- MS Excel (xls)
- Open Office Writer
- Html

Структура служебных таблиц.

Таблица 9 — Служебные таблицы системы полнотекстового поиска

Таблица	Описание
FTS\$INDICES	Метаданные индекса
FTS\$INDEX_SEGMENTS.	Метаданные поля БД, входящего в индекс
FTS\$POOL	Содержит RDB\$DB_KEY для измененных, но не проиндексированных полей
FTS\$ LUCENE_FILE_SYSTEM	Эмуляция файловой системы для хранения данных индекса.
FTS\$ANLYZER_MAP	Служебная таблица для соотношения collations с анализаторами текста Lucene

Таблица 10 — Структура таблицы FTS\$INDICES

Имя поля	Тип	Описание
FTS\$INDEX_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя индекса
FTS\$STORE	BLOB SUB_TYPE 1 SEGMENT SIZE 80 CHARACTER SET UNICODE_FSS	Описание, где хранится индекс.
FTS\$DESCRIPTION	BLOB SUB_TYPE 1 SEGMENT SIZE 80 CHARACTER SET UNICODE_FSS	Комментарии к индексу
FTS\$INDEX_STATUS	CHAR(1)	Статус индекса, 1. 'I' inactive – индекс неактивный 2. 'N' new – индекс создан, требует полное переиндексирование 3. 'U' needs metadata update – требуется изменение метаданных, триггеров и т.д. 4. 'D' drop – индекс отмечен к удалению. 5. 'C' complete – для индекса сделаны все изменения в метаданных и он индексируется.

Таблица 11 - Структура таблицы FTS\$INDEX_SEGMENTS

Имя поля	Тип	Описание
FTS\$INDEX_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя индекса
FTS\$RELATION_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Индексируемая таблица
FTS\$FIELD_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Индексируемое поле

Имя поля	Тип	Описание
FTS\$TRIGGER_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя триггера, который будет заполнять таблицу FTS\$POOL, после изменения данных.
FTS\$ANALIZER	CHAR(255) CHARACTER SET UNICODE_FSS	Служебная информация, имя анализатора.
FTS\$MIME_TYPE	CHAR(127) CHARACTER SET UNICODE_FSS	Имя MIME формата документа, хранящегося в индексируемом поле, если в столбце хранятся документы одного типа.
FTS \$MIME_FIELD_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя столбца, в котором хранится имя файла для определения MIME формата документа хранящегося в индексируемом поле.

Таблица 12 — Структура таблицы FTS\$POOL

Имя поля	Тип	Описание
FTS\$DB_KEY	CHAR(8) CHARACTER SET OCTETS	RDB\$DB_KEY запись которая была изменена или добавлена (в данном случае оправданно использование RDB\$DB_KEY т.к., если индексирование будет сильно отставать от изменений данных, то повторное использование RDB\$DB_KEY не будет критично).

Таблица 13 — Структура таблицы FTS\$ LUCENE_FILE_SYSTEM

Имя поля	Тип	Описание
FTS\$INDEX_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя индекса
FTS\$FILE_NAME	VARCHAR(255) CHARACTER SET UNICODE_FSS	Имя файла
FTS\$CREATE_TIME	TIMESTAMP	Время создания файла
FTS \$LAST_MODIFY_TIME	TIMESTAMP	Время последнего изменения файла
FTS\$FILE_BODY	BLOB SUB_TYPE 0 SEGMENT SIZE 1024	Содержимое файла.

Таблица 14 - Структура таблицы FTS\$ANLYZER_MAP.

Имя поля	Тип	Описание
RDB \$CHARACTER_SET_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя таблицы символов
FTS\$ANALIZER	BLOB SUB_TYPE 1 SEGMENT SIZE 80 CHARACTER SET UNICODE_FSS	Служебная информация, имя анализатора.

Хранимые процедуры для управления индексами.

Процедура FTS\$CREATE_INDEX создает FTS индекс.

Входные параметры.

Имя поля	Тип	Описание
FTS\$INDEX_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя индекса
FTS\$STORE	BLOB SUB_TYPE 1 SEGMENT SIZE 80 CHARACTER SET UNICODE_FSS	Описание, где хранится индекс. Может быть NULL , тогда индекс будет, хранится в таблице FTS\$ LUCENE_FILE_SYSTEM. Значение по умолчанию NULL .
FTS\$DESCRIPTION	BLOB SUB_TYPE 1 SEGMENT SIZE 80 CHARACTER SET UNICODE_FSS	Комментарии к индексу. Значение по умолчанию NULL .

Процедура FTS\$DROP_INDEX удаляет FTS индекс.

Входные параметры.

Имя поля	Тип	Описание
FTS\$INDEX_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя индекса

Процедура FTS\$ADD_FILED_TO_INDEX добавляет индексируемое поле в индекс.

Входные параметры.

Имя поля	Тип	Описание
FTS\$INDEX_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя индекса
FTS\$RELATION_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Индексируемая таблица
FTS\$FIELD_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Индексируемое поле
FTS\$ANALYZER	CHAR(255) CHARACTER SET UNICODE_FSS	Служебная информация, имя анализатора. Может принимать значение NULL, в таком случае Analyzer будем выбираться исходя из collate индексируемого поля. Значение по умолчанию NULL .

Процедура FTS\$ADD_MIME_FLD2IDX добавляет индексируемое поле в индекс.

Входные параметры.

Имя поля	Тип	Описание
FTS\$INDEX_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя индекса
FTS\$RELATION_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Индексируемая таблица
FTS\$FIELD_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Индексируемое поле
FTS\$ANALYZER	CHAR(255) CHARACTER SET UNICODE_FSS	Служебная информация, имя анализатора. Может принимать значение NULL, в таком случае Analyzer будем выбираться исходя из collate индексируемого поля. Значение по умолчанию NULL .
FTS\$MIME_TYPE	CHAR(127) CHARACTER SET UNICODE_FSS	MIME тип индекса указывает, что индекс строится для документов

Процедура FTS\$ADD_MIME_FILE_FLD2IDX добавляет индексируемое поле для документа MIME.

Входные параметры.

Имя поля	Тип	Описание
FTS\$INDEX_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя индекса
FTS\$RELATION_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Индексируемая таблица
FTS\$FIELD_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Индексируемое поле

Имя поля	Тип	Описание
FTS\$ANALIZER	CHAR(255) CHARACTER SET UNICODE_FSS	Служебная информация, имя анализатора. Может принимать значение NULL, в таком случае Analyzer будем выбираться исходя из collate индексируемого поля. Значение по умолчанию NULL .
FTS \$MIME_FIELD_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя поля для определение MIME формата документов

Процедура для удаления индексируемого поля из индекса: FTS \$DROP_FILED_FROM_INDEX.

Входные параметры.

Имя поля	Тип	Описание
FTS\$INDEX_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя индекса
FTS\$RELATION_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Индексируемая таблица
FTS\$FIELD_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Индексируемое поле

Процедура для указания, что изменения метаданных закончились: APPLAY_META_CHANGES. Входные и выходные параметры отсутствуют. Данную процедуру необходимо вызывать после создания индекса и добавления индексируемого поля в индекс.

Обновления статистики для одного индекса - переиндексация, производится с помощью вызова процедуры FTS\$REINDEX

Входные параметры.

Имя поля	Тип	Описание
FTS\$INDEX_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя индекса

Процедура, запускающая демон для индексирования изменяемых данных - FTS\$STARTDAEMON.

Процедура, возвращающая набор данных извлеченных из индекса: FTS_SEARCH

Входные параметры.

Имя поля	Тип	Описание
FTS\$INDEX_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Имя индекса
FTS\$RELATION_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Индексируемая таблица. Может принимать значение NULL, тогда поиск будет идти по всем таблицам, входящим в индекс.
FTS\$FILTER	VARCHAR(4000) CHARACTER SET UNICODE_FSS	Фильтр, по которому будет осуществляться поиск.
FTS\$FILTER_TYPE	CHAR(31) CHARACTER SET UNICODE_FSS	[TBD]
FTS\$ HIGHLIGHT _TYPE	CHAR(31) CHARACTER SET UNICODE_FSS	[TBD]

Выходные параметры.

Имя поля	Тип	Описание
FTS\$RELATION_NAME	CHAR(31) CHARACTER SET UNICODE_FSS	Таблица, в которой найдено поле, удовлетворяющее фильтру.

FTS\$DB_KEY	BIGINT	Значение RDB\$DB_KEY для таблицы содержащей запись.
FTS\$SOPE	DOUBLE PRECISION	Оценка соответствия возвращаемой записи условию поиска

Пример использования полнотекстового поиска

Создание индекса:

```
EXECUTE PROCEDURE FTS$CREATE_INDEX('SIMPLE_INDEX',  
'file', 'Simple index for FTS')
```

Добавление индексируемого поля DESCRIPTION таблицы REMARK в индекс SIMPLE_INDEX:

```
FTS$ADD_FIELD_TO_INDEX('SIMPLE_INDEX', 'REMARK',  
'DESCRIPTION', NULL)
```

Выполнение операций по добавлению и обновлению записей таблицы REMARK:

```
EXECUTE PROCEDURE FTS$APPLY_META_CHANGES;
```

Переиндексация:

```
EXECUTE PROCEDURE FTS$REINDEX('SIMPLE_INDEX')
```

Поиск записей, поле DESCRIPTION которых содержит текст "searching_test":

```
SELECT * from FTS$SEARCH('SIMPLE_INDEX', NULL,  
'searching_test', NULL, NULL)
```

Удаление индекса:

```
EXECUTE PROCEDURE FTS$DROP_INDEX('SIMPLE_INDEX')
```

Приложение А - Изменения в ODS

Доступ к административным функциям (сервисам)

Права на запуск сервисов хранятся в таблице RDB\$USER_PRIVILEGES БД security2.fdb.

Доступ к DDL операциям

Новые значения привилегий в таблице RDB\$USER_PRIVILEGES в поле RDB\$PRIVILEGE:

```
C - CREATE,  
L - ALTER,  
T - ALTER ANY,  
O - DROP,  
P - DROP ANY.
```

В RDB\$RELATION_NAME хранится имя типа объекта (RDB\$TABLE, RDB\$PROCEDURE и т.д.).

В RDB\$OBJECT_TYPE хранится тип объекта.

В системных таблицах RDB\$EXCEPTIONS, RDB\$GENERATORS, RDB\$FUNCTIONS, RDB\$FIELDS, RDB\$FILES добавилось поле RDB\$OWNER_NAME.

В RDB\$SECURITY_CLASSES хранятся глобальные типы объектов (OBJ\$TABLE, OBJ\$PROCEDURE и т.д.).

Доступ к DML операциям

Добавление в таблицу RDB\$USER_PRIVILEGES записи со значением S в поле RDB\$PLIVILEGE, дает пользователю RDB\$USER право на выборку поля RDB\$FIELD_NAME из отношения RDB\$RELATION_NAME.

Роли по умолчанию

Предопределенные права доступа для ролей SYSADMIN, SECADMIN, PUBLIC. Поддержка глобальных ролей

Для того, чтобы SECADMIN мог работать с пользователями, доработана структура представления USERS в security2.fdb:

```
CREATE VIEW USERS (  
    USER_NAME, SYS_USER_NAME, GROUP_NAME, UID, GID,  
    PASSWD, PRIVILEGE, COMMENT, FIRST_NAME, MIDDLE_NAME,  
    LAST_NAME, FULL_NAME, HASH_ALG, HASH_SIZE) AS  
SELECT  
    RDB$USER_NAME, RDB$SYS_USER_NAME, RDB$GROUP_NAME,  
    RDB$UID, RDB$GID, RDB$PASSWD, RDB$PRIVILEGE, RDB$  
    $COMMENT,  
    RDB$FIRST_NAME, RDB$MIDDLE_NAME, RDB$LAST_NAME,  
    COALESCE (RDB$first_name || _UNICODE_FSS ' ', '') ||
```

```
COALESCE (RDB$middle_name || _UNICODE_FSS ' ', '')
||
COALESCE (RDB$last_name, ''),
RDB$HASH_ALG,
RDB$HASH_SIZE
FROM RDB$USERS
WHERE
CURRENT_USER = 'SYSDBA' OR CURRENT_ROLE = 'SECADMIN'
OR CURRENT_USER = RDB$USERS.RDB$USER_NAME;
/* Access rights */
GRANT ALL ON RDB$USERS to VIEW USERS;
GRANT SELECT ON USERS to PUBLIC;
GRANT UPDATE (PASSWD, GROUP_NAME, UID, GID, FIRST_NAME,
MIDDLE_NAME, LAST_NAME)
ON USERS TO PUBLIC;
GRANT INSERT ON USERS to SECADMIN;
GRANT DELETE ON USERS to SECADMIN;
```

Кумулятивное действие ролей

Для хранения контекста процедуры в таблице RDB\$PROCEDURES добавлено новое поле RDB\$PROCEDURE_CONTEXT типа SMALLINT, которое принимает значение либо 0 (OWNER) либо 1 (CALLER).

Многофакторная аутентификация

В security2.fdb в таблицу RDB\$USERS добавилось поле RDB\$HASH_ALG с названием алгоритма хеширования. Этот алгоритм будет использован для проверки пароля. Если поле не указано, то используется старый алгоритм хеширования.

Политики доступа

Для хранения политик пользователя в security2.fdb создана таблица

```
CREATE TABLE RDB$POLICIES (
RDB$POLICY_NAME RDB$POLICY_NAME NOT NULL PRIMARY
KEY,
RDB$PSWD_NEED_CHAR RDB$COUNT,
RDB$PSWD_NEED_DIGIT RDB$COUNT,
RDB$PSWD_NEED_DIFF_CASE RDB$BOOL,
RDB$PSWD_MIN_LEN RDB$LENGTH,
RDB$PSWD_VALID_DAYS RDB$DAY_COUNT,
RDB$PSWD_UNIQUE_COUNT RDB$COUNT,
RDB$MAX_FAILED_COUNT RDB$COUNT,
```

```
RDB$MAX_SESSIONS RDB$COUNT,  
RDB$MAX_IDLE_TIME RDB$MAX_IDLE_TIME,  
RDB$AUTH_FACTORS RDB$AUTH_FACTORS  
);
```

Политика по умолчанию имеет RDB\$POLICY_NAME == 'DEFAULT'.

Поле RDB\$AUTH_FACTORS содержит строку с условным выражением из символических обозначений факторов аутентификации.

Для хранения даты установки пароля, последующей проверке его валидности, регистрации неудачных попыток авторизации, сопоставления политики, хранения названия алгоритма, с помощью которого получен хеш пароля, и времени, после которого пользователь может получить доступ к БД, в таблицу RDB\$USERS добавились поля

```
RDB$HASH_ALG RDB$HASH_ALG,  
RDB$POLICY_NAME RDB$POLICY_NAME REFERENCES RDB$POLICIES,  
RDB$PASSWD_TIME TIMESTAMP,  
RDB$FAILED_COUNT RDB$COUNT,  
RDB$ACCESS_TIME RDB$ACCESS_TIME
```

Для хранения хешей старых паролей в security2.fdb добавилась таблица

```
CREATE TABLE RDB$PASSWD_HISTORY (  
RDB$KEY_ID INTEGER PRIMARY KEY NOT NULL,  
RDB$USER_NAME RDB$USER_NAME REFERENCES RDB$USERS,  
RDB$PASSWD RDB$PASSWD,  
RDB$HASH_ALG RDB$HASH_ALG  
);
```

Инструмент анализа журнала аудита

В системной таблице RDB\$RELATIONS объявлено новое поле RDB\$ADAPTER типа char [31].