



**Ред База Данных**

**Версия 2.1**

**Примечания к выпуску**

© Корпорация Ред Софт 2008

Данный документ содержит описание новых возможностей СУБД «Ред База Данных» 2.1. Документ рассчитан на пользователей, знакомых с принципами организации баз данных СУБД «Ред База Данных» и языком SQL. Более подробную информацию об использовании тех или иных функций можно получить из документов «Руководство по быстрому старту», «Руководство администратора», «Руководство по полнотекстовому поиску», «Руководство по внешним хранимым процедурам»

[www.red-soft.biz](http://www.red-soft.biz)

## Содержание

1 Доступ к операциям управления метаданными.....	4
2 Доступ к операциям манипулирования данными.....	5
3 Предопределенные роли.....	6
4 Контроль доступа к административным функциям.....	7
5 Кумулятивное (комбинированное) действие ролей.....	8
6 Выполнение процедур.....	9
7 Многофакторная аутентификация.....	10
8 Политики безопасности.....	12
9 Обезличивание памяти.....	14
10 Аудит.....	15
11 Адаптер для подключения бинарного файла аудита .....	17
12 Контроль целостности.....	19
12.1 Контроль целостности метаданных.....	19
12.2 Контроль целостности файлов сервера.....	19
13 Фильтрация полей и записей.....	21
14 Функциональные доработки.....	22
14.1 Архитектура суперклассик.....	22
14.2 Внешние процедуры и функции .....	22
14.3 Полнотекстовый поиск.....	22
14.4 Отладчик хранимых процедур и функций.....	22
Приложение А - Изменения в ODS.....	24

## 1 Доступ к операциям управления метаданными

Расширен функционал операторов GRANT и REVOKE для объектов следующих типов: PROCEDURE, FUNCTION, ROLE, TABLE, VIEW, EXCEPTION, GENERATOR, DOMAIN, SHADOW, POLICY.

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;
```

Если указано ключевое слово ANY, то пользователь может удалять и модифицировать любой объект указанного типа. В противном случае только те объекты, владельцем которых он является. Если при назначении прав использовалось ключевое слово ANY, то его необходимо указывать и в операторе REVOKE 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 <sup>1</sup>

<sup>1</sup> 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»).

## 2 Доступ к операциям манипулирования данными

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 надо 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}

### 3 Предопределенные роли

Добавлены глобальные (действующие для всех баз сервера) роли SYSADMIN, SECADMIN, PUBLIC:

**Таблица 3 - Права доступа (по умолчанию) предопределенных ролей**

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

Права роли PUBLIC будут иметь все пользователи. Эта роль назначается пользователю автоматически при его 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.

Кроме этих предопределенных ролей, пользователь SYSDBA или администратор безопасности (пользователь с ролью SECADMIN) могут создать дополнительные глобальные роли. Если пользователю назначена какая-либо глобальная роль, то он является ее обладателем во всех базах данных сервера. Основное назначение глобальных ролей — выполнять операции с сервисами. При этом назначение прав глобальным на DDL или DML операции необходимо проводить в каждой базе данных.

**Примечание: Глобальные роли могут быть назначены пользователю только при прямом соединении к базе данных безопасности security2.fdb. Такое подключение могут выполнить только SYSDBA или пользователь с ролью SECADMIN**

## 4 Контроль доступа к административным функциям

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

- резервное копирование/восстановление (backup/restore, GBAK);
- получение свойств БД, анализ и восстановление БД (GFIX);
- получение статистики БД (Database Stats, GSTAT).

Право на запуск сервисов является глобальным для сервера, поэтому хранится в базе данных безопасности security2.fdb. Назначить эти права сервиса можно только на пользователя или глобальную роль. 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 этого 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.

**Примечание: право на выполнение сервиса означает, что пользователь автоматически получает права на все DML и DDL операции, необходимые при работе с данным сервисом во всех базах данных сервера. Пользователь может обратиться к сервису и не имея прав на его запуск, однако он не сможет выполнить требуемых операций, так как не имеет на них прав.**

## 5 Кумулятивное (комбинированное) действие ролей

Добавлена возможность присвоения одной роли другой (добавления группы пользователей в другую группу). Для этого используется следующий синтаксис 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 назначенных ему 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.

Имена пользователей и ролей не должны совпадать. Полностью 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 Выполнение процедур

Добавлена 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
```

**Примечание: при переносе баз данных с Firebird на «Ред База Данных» 2.1 необходимо учитывать, что в Firebird процедура по умолчанию выполняется с правами вызывающего ее пользователя**

## 7 Многофакторная аутентификация<sup>2</sup>

**Идентификация** – предъявление пользователем имени (логина) для входа в систему.

**Аутентификация** – процедура подтверждения пользователем того, что он тот, чье имя он предъявил в ходе идентификации.

**Фактор аутентификации** — данные, предъявленные пользователем, для проверки одного из условий необходимых для прохождения аутентификации. Факторы могут быть первичными и вторичными. Первичные факторы — пароль, контекст безопасности ОС и сертификат пользователя. 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.

**Многофакторная аутентификация** – аутентификация с использованием нескольких факторов аутентификации (пароль, сертификат). Факторы, необходимые для прохождения аутентификации определяются политикой безопасности. Только в режиме многофакторной аутентификации производится проверка соответствия пароля требованиям политики безопасности.

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 поль-

---

<sup>2</sup> 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.

зователя. 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 сессионных 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 для настройки многофакторной аутентификации**

Параметр	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
CryptoPluginName	Значение по умолчанию - cryptopro_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

## 8 Политики безопасности

Политики учетных записей включают в себя:

- требования к сложности пароля;
- количество предыдущих паролей, которые не должен повторять вновь заданный;
- срок действия пароля;
- количество одновременных подключений (сессий) пользователя;
- продолжительность простоя пользователя до отключения.

Политики учетных записей создаются и хранятся в базе данных безопасности security2.fdb. Политика определяет реакцию системы на неудачные попытки входа в систему, 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;
- 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 определенной время либо навсегда, если значение счетчика превысило определенный политикой предел. Время, после которого пользователь может подключаться к БД прописывается вместе с информацией о пользователе и счетчиком неудачных попыток доступа;

- изменение параметров политики подтверждается сразу и вступают в силу для обработки последующих подключений.

Для управления политиками безопасности, используются 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. 5
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
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

## 9 Обезличивание памяти

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.

## 10 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.

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

Подробнее о настройке аудита см. гл. 4 «Руководства администратора»

## 11 Адаптер для подключения бинарного файла аудита

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

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

Инструмент анализа разбирает бинарные лог-файлы системы аудита «Ред 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)

## 12 Контроль целостности

### 12.1 Контроль целостности метаданных<sup>3</sup>

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
```

### 12.2 Контроль целостности файлов сервера<sup>4</sup>

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»

<sup>3</sup> См. примечание к главе 6 «Многофакторная аутентификация»

<sup>4</sup> См. примечание к главе 6 «Многофакторная аутентификация»

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.

## 13 Фильтрация полей и записей

В «Ред База Данных» 2.1 реализован механизм фильтрации полей и записей таблиц и предотвращения прямого изменения данных системного каталога. Режим фильтрации записей системного каталога активируется с помощью параметра

```
UseRecFilter = 0|1
```

из файла конфигурации «Ред База Данных».

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

Для задания условий фильтрации полей и записей используются следующие языковые конструкции:

```
CREATE TABLE table [EXTERNAL [FILE] "<filespec>"]  
(<col_def> [, <col_def> | <tconstraint> ...], [COLFILTER  
<col_name> (<condition>), ...]) [, RECFILTER  
(<condition>)];
```

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

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

Установка условия фильтрации записей <condition> для таблицы

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

Удаление условия фильтрации записей:

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

Установка условия фильтрации полей <condition> для таблицы:

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

Удаление условия фильтрации полей для <table>

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

Кроме того, при включении режима фильтрации записей активируются установки по умолчанию для системных таблиц. Подробнее о правах доступа к системным таблицам, видимости полей и записей в них описано в «Руководстве администратора» п. 4.8.2.

## 14 Функциональные доработки<sup>5</sup>

### 14.1 Архитектура суперклассик

СУБД «Ред База Данных» 2.1 поддерживается 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.

### 14.2 Внешние процедуры и функции

В «Ред База Данных» 2.1 есть возможность подключать внешние хранимые процедуры и функции на Java — External Stored Procedures, ESP. Эти процедуры и функции выполняются на стороне сервера и совмещают в себе достоинства обычных процедур PSQL и функций, определенными пользователем — UDF. Внешние хранимые процедуры позволяют:

- работать с внешними объектами;
- производить обработку данных из нескольких баз данных;
- права на выполнение ESP задаются точно так же как и для обычных процедур PSQL;
- ESP можно использовать в триггерах;

Кроме того в настоящее время ведутся доработки по использованию в ESP других языков программирования — например C++ и Java.

Подробнее об использовании ESP можно узнать из документа «Руководство по внешним процедурам» на этом диске.

### 14.3 Полнотекстовый поиск

В «Ред База Данных» 2.1 реализована система полнотекстового поиска на основе библиотеки Lucene. Система полнотекстового поиска позволяет:

- производить морфологический поиск
- производить поиск по неточному соответствию
- производить поиск по текстовым и бинарным полям, содержащим документы следующих типов:
  - документы Microsoft Office (doc, xls);
  - документы Open Office (odt);
  - документы в формате PDF;
  - html-файлы.

Подробнее об использовании системы полнотекстового поиска см. документ «Руководство по полнотекстовому поиску» на этом диске

### 14.4 Отладчик хранимых процедур и функций

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.

---

<sup>5</sup> Для использования полнотекстового поиска и/или внешних хранимых процедур необходимо установить JDK 1.6

Для регистрации клиента в сервере отладки из отлаживаемого приложения должна быть вызвана хранимая процедура `DBG$UPDATE`. Также эту процедуру необходимо вызвать после изменения метаданных в базе данных, для того, чтобы сделать возможным отладку процедур и триггеров с учетом внесенных изменений.

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 сервер 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.

## 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.

### Переопределенные роли

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 (
    PASSWORD, 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;
```

### Выполнение процедур

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 значение поля RDB\$POLICY\_NAME равным 'DEFAULT'.

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

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

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

Для хранения хешей старых паролей в security2.fdb добавилась таблица

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

Добавлены два новых представления. Для получения информации о пользователе используется представление POLICY\_USERS, которое объединяет пользователя и политику его доступа в единое отношение:

```
CREATE VIEW POLICY_USERS (  
    USER_NAME, UID, GID, PASSWORD, HASH_ALG, FAILED_COUNT,  
    MAX_FAILED_COUNT, MAX_SESSIONS, PASSWORD_TIME,  
    PSWD_VALID_DAYS, ACCESS_TIME, AUTH_FACTORS  
)  
AS SELECT  
    U.RDB$USER_NAME, U.RDB$UID, U.RDB$GID, U.RDB$PASSWORD,  
    U.RDB$HASH_ALG, U.RDB$FAILED_COUNT,  
    P.RDB$MAX_FAILED_COUNT, P.RDB$MAX_SESSIONS,  
    U.RDB$PASSWORD_TIME, P.RDB$PSWD_VALID_DAYS,  
    U.RDB$ACCESS_TIME, P.RDB$AUTH_FACTORS  
FROM  
    RDB$USERS U LEFT JOIN RDB$POLICIES P ON  
    U.RDB$POLICY_NAME = P.RDB$POLICY_NAME  
WHERE  
    ((U.RDB$ACCESS_TIME IS NULL)  
    OR (CURRENT_TIMESTAMP > U.RDB$ACCESS_TIME))  
    AND ((COALESCE(U.RDB$FAILED_COUNT, 0) <  
    P.RDB$MAX_FAILED_COUNT)  
    OR (COALESCE(P.RDB$MAX_FAILED_COUNT, 0) = 0));
```

Представление PASSWORD\_POLICY объединяет сведения о пользователе и требованиях к паролю назначенной ему политики безопасности:

```
CREATE VIEW PASSWORD_POLICY (  

```

```
USER_NAME, POLICY_NAME, PSWD_NEED_CHAR,  
PSWD_NEED_DIGIT, PSWD_NEED_DIFF_CASE, PSWD_MIN_LEN,  
PSWD_UNIQUE_COUNT  
)  
AS SELECT  
    U.RDB$USER_NAME, P.RDB$POLICY_NAME,  
    P.RDB$PSWD_NEED_CHAR, P.RDB$PSWD_NEED_DIGIT,  
    P.RDB$PSWD_NEED_DIFF_CASE, P.RDB$PSWD_MIN_LEN,  
    P.RDB$PSWD_UNIQUE_COUNT  
FROM  
    RDB$USERS U LEFT JOIN RDB$POLICIES P ON  
    U.RDB$POLICY_NAME = P.RDB$POLICY_NAME;
```

Также в таблицу MON\$ATTACHMENTS добавлено поле MON\$LAST\_ACTIVITY\_TIME типа DATETIME, которое содержит время последнего обращения пользователя к серверу.

### **Инструмент анализа журнала аудита**

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