СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

ЛИНТЕР®

ЛИНТЕР БАСТИОН

ЛИНТЕР СТАНДАРТ

# Администрирование комплекса средств защиты данных

НАУЧНО-ПРОИЗВОДСТВЕННОЕ ПРЕДПРИЯТИЕ

РЕЛЭКС

## Товарные знаки

РЕЛЭКС™, ЛИНТЕР® являются товарными знаками, принадлежащими АО НПП «Реляционные экспертные системы» (далее по тексту – компания РЕЛЭКС). Прочие названия и обозначения продуктов в документе являются товарными знаками их производителей, продавцов или разработчиков.

## Интеллектуальная собственность

Правообладателем продуктов ЛИНТЕР® является компания РЕЛЭКС (1990-2026). Все права защищены.

Данный документ является результатом интеллектуальной деятельности, права на который принадлежат компании РЕЛЭКС.

Все материалы данного документа, а также его части/разделы могут свободно размещаться на любых сетевых ресурсах при условии указания на них источника документа и активных ссылок на сайты компании РЕЛЭКС: relex.ru и linter.ru.

При использовании любого материала из данного документа несетевым/печатным изданием обязательно указание в этом издании источника материала и ссылок на сайты компании РЕЛЭКС: relex.ru и linter.ru.

Цитирование информации из данного документа в средствах массовой информации допускается при обязательном упоминании первоисточника информации и компании РЕЛЭКС.

Любое использование в коммерческих целях информации из данного документа, включая (но не ограничиваясь этим) воспроизведение, передачу, преобразование, сохранение в системе поиска информации, перевод на другой (в том числе компьютерный) язык в какой-либо форме, какими-либо средствами, электронными, механическими, магнитными, оптическими, химическими, ручными или иными, запрещено без предварительного письменного разрешения компании РЕЛЭКС.

## О документе

Материал, содержащийся в данном документе, прошел доскональную проверку, но компания РЕЛЭКС не гарантирует, что документ не содержит ошибок и пропусков, поэтому оставляет за собой право в любое время вносить в документ исправления и изменения, пересматривать и обновлять содержащуюся в нем информацию.

## Контактные данные

394006, Россия, г. Воронеж, ул. Бахметьева, 2Б.

Тел./факс: (473) 2-711-711, 2-778-333.

e-mail: info@linter.ru.

## Техническая поддержка

С целью повышения качества программного продукта ЛИНТЕР и предоставляемых услуг в компании РЕЛЭКС действует автоматизированная система учёта и обработки пользовательских рекламаций. Обо всех обнаруженных недостатках и ошибках в программном продукте и/или документации на него просим сообщать нам в раздел Поддержка на сайте ЛИНТЕР.

# Содержание

**Предисловие** ..... 3
Назначение документа ..... 3
Для кого предназначен документ ..... 3
Дополнительные документы ..... 3
**Архитектура комплекса средств защиты данных** ..... 4
Основные принципы ..... 4
Открытость ..... 4
Идентификация и аутентификация ..... 5
Контроль доступа в процессе обработки запроса ..... 6
Прямая передача прав ..... 6
Косвенная передача прав ..... 7
Регистрация действий ..... 7
Метка доступа ..... 8
Метка объекта ..... 8
Метка субъекта ..... 9
Схема хранения данных ..... 9
Категория доступа субъекта контроля ..... 9
Роли ..... 11
Управление привилегиями доступа ..... 11
Группы пользователей ..... 12
Создание группы ..... 12
Изменение имени группы ..... 13
Назначение группы пользователю ..... 13
Предоставление доверия группе ..... 13
Отмена доверия группе ..... 13
Уровни доступа ..... 14
Управление уровнем доступа пользователей ..... 16
Управление уровнем доступа к таблице ..... 17
Управление уровнем доступа к столбцу таблицы ..... 17
Представление групп и уровней в SQL-выражениях ..... 17
Получение информации о метках доступа ..... 18
SELECT-выражения ..... 18
UPDATE, INSERT и MERGE операторы ..... 19
Управление мандатной защитой на уровне сессии ..... 20
Маркировка документов ..... 22
Контроль доступа к БД с сетевых станций ..... 23
Создание/удаление сетевой станции ..... 25
Изменение параметров сетевой станции ..... 26
Управление доступом к станции ..... 26
Защита ввода-вывода на внешний носитель ..... 27
Контроль целостности средств защиты данных ..... 29
Подсчет контрольной суммы ..... 31
Диспетчер доступа ..... 31
Создание БД ..... 32
Системная БД ..... 32
Структура системных таблиц ..... 33
Рабочая БД ..... 33
Инициализация комплекса средств защиты данных ..... 34
**Мониторинг комплекса средств защиты данных** ..... 35
Управление мониторингом ..... 38
Настройка протоколирования ..... 39
Протоколирование пользовательского сообщения ..... 45
Начать комментирование протоколируемых событий ..... 45

Отменить комментирование протоколируемых событий ..... 45
Протоколируемые события ..... 45
Системные события ..... 45
События, связанные с БД ..... 46
События, связанные с подсистемой доступа ..... 47
События, связанные с таблицами ..... 47
События, связанные с пользователями ..... 48
**Механизм надежного восстановления** ..... 51
**Преобразование данных** ..... 52
**Очистка оперативной и внешней памяти** ..... 53
**Изоляция модулей** ..... 55
**SQL-запросы для работы с системой защиты данных** ..... 56
Именование объектов БД ..... 56
Предоставление привилегий ..... 56
Отмена привилегий ..... 57
Создание/удаление пользователя ..... 57
Изменение пароля пользователя ..... 58
Создание/удаление роли ..... 59
Назначение/отмена назначения роли ..... 59
**Приложение. Синтаксис команд для работы с комплексом средств защиты данных** ..... 60
**Указатель команд SQL-запросов** ..... 65

# Предисловие

## Назначение документа

Документ содержит описание процедур администрирования комплекса средств защиты информации (КСЗ) от несанкционированного доступа (НСД), описание контролируемых функций и рекомендации по созданию систем защиты:

- описание контролируемых функций;
- руководство по генерации КСЗ;
- описание старта системы и процедур проверки правильности старта;
- описание процедур работы со средствами регистрации;
- руководство по средствам надежного восстановления;
- руководство по работе со средствами контроля модификации;
- руководство по работе со средствами контроля дистрибуции.

Документ предназначен для СУБД ЛИНТЕР БАСТИОН 6.0 сборка 20.8, далее по тексту СУБД ЛИНТЕР.

## Для кого предназначен документ

Документ предназначен для администратора безопасности системы, а также для администратора защиты.

## Дополнительные документы

- Модель защиты данных
- Справочник по SQL
- Системные таблицы и представления
- Создание и конфигурирование базы данных

# Архитектура комплекса средств защиты данных

Архитектура комплекса средств защиты данных (КСЗ) включает в себя как информационные компоненты (описания объектов и субъектов в БД), так и программные компоненты (программный КСЗ ядра СУБД ЛИНТЕР).

## Основные принципы

КСЗ СУБД ЛИНТЕР строится на 9 основных принципах:

1) открытость;
2) идентификация и аутентификация – определяет начало сеанса работы субъекта;
3) права на доступ проверяются на всех этапах:
   - на этапе трансляции запроса;
   - на этапе обработки запроса;
   - при обращении к объекту.
4) прямая передача прав – прерогатива владельца объекта;
5) косвенная передача прав – совместные действия владельца объекта и создателя роли;
6) для протоколирования всех действий с участием КСЗ необходимо его активировать и настроить требуемый перечень протоколируемых событий;
7) все объекты контроля снабжаются метками доступа, включающими уровни конфиденциальности и принадлежность к группе субъектов; субъекты разбиты на непересекающиеся группы, каждая из которых изолирована по доступу от всех прочих групп;
8) метки объектов, как и метки субъектов, могут изменяться (их можно переводить из группы в группу, менять уровень доступа и пр.);
9) изменение меток доступа субъектов – прерогатива субъекта (администратора безопасности).

**Примечание**
Протоколирование событий и работа с метками доступа поддерживаются только в СУБД ЛИНТЕР БАСТИОН.

Ниже эти принципы разбираются более подробно.

## Открытость

СУБД ЛИНТЕР – открытая система. По технологии построения открытых систем все ее алгоритмы открыты для всех пользователей и работают для всех одинаково.

Алгоритм интерфейса нижнего уровня устроен таким образом, что установленная связь субъекта и СУБД ЛИНТЕР идентифицируется еще и идентификатором прикладной задачи (каждая операционная система присваивает задаче уникальный идентификатор) (см. пункт Идентификация и аутентификация), что исключает возможность перехвата

ответов от ядра СУБД. Это означает, что ответ, посланный задаче на ее запрос, придет именно к этой задаче и ни к какой другой.

При этом прикладные пользовательские программы отделены от СУБД. Общение между ядром СУБД и приложением, ее использующим, проходит только через интерфейс нижнего уровня. Тексты/алгоритмы интерфейса нижнего уровня открыты и не содержат элементов КСЗ.

Все утилиты СУБД ЛИНТЕР также написаны с помощью интерфейса нижнего уровня (то есть штатных средств) и не используют никаких скрытых особенностей СУБД.

То же самое можно сказать и о прочих программных интерфейсах, присутствующих в СУБД (например, встроенный SQL). В их основе также лежит интерфейс нижнего уровня СУБД ЛИНТЕР.

Это налагает на СУБД дополнительные требования, особенно в части КСЗ.

# Идентификация и аутентификация

Информация о каждом пользователе, имеющем право подключаться к СУБД, находится в соответствующей системной БД.

Каждый пользователь (субъект) имеет в системной БД уникальный идентификатор – идентификатор субъекта.

СУБД связывает субъекта с его идентификатором, исходя из имени и пароля, полученных в результате идентификации и аутентификации.

После этого на всем протяжении работы (до отсоединения пользователя от СУБД) ядру СУБД уже не требуется знание имени и пароля субъекта. Для алгоритмов доступа достаточно только идентификатора субъекта.

При хранении информации о правах субъекта (см. раздел Схема хранения данных) СУБД использует идентификатор субъекта, а также идентификаторы объектов, к которым субъект имеет доступ.

Для каждого установленного канала связи с приложением СУБД хранит у себя в памяти следующую информацию:

- идентификатор субъекта;
- идентификатор приложения (процесса и даже нити);
- адрес клиентской станции и тип сетевого протокола (при сетевой работе).

Таким образом, сеанс работы субъекта начинается с идентификации и аутентификации субъекта, при этом СУБД связывает последнего с его идентификатором. Данная связь прекращается только после отсоединения субъекта от СУБД.

Кроме того, СУБД особенно важно, чтобы ответы на все запросы приложения были получены только этим приложением и никаким другим. Это реализуется с помощью идентификатора приложения, который присваивается ему операционной системой в момент запуска. При установлении связи приложение, используя интерфейс нижнего уровня, сообщает ядру СУБД свой идентификатор (а при сетевой работе и адрес станции), так что, посылая приложению ответы, СУБД владеет полной информацией о том, кому можно пересылать полный адрес нужного приложения.

В случае, когда исчезает хотя бы один из компонентов связи (удаляется субъект, падает приложение, отключается клиентская станция), ядро СУБД, следуя установленным тайм-аутам, разрывает образованную связь.

При попытке пользователя открыть канал связи, СУБД проверяет **имя** пользователя и его **пароль**.

При этом возможны следующие ситуации:

- имя и пароль пользователя введены правильно – пользователь допускается к БД;
- имя пользователя введено неверно – выдается сообщение о том, что имя пользователя введено неверно (`Неизвестное имя пользователя`). Пользователь к БД не допускается;
- пароль пользователя введен неверно – выдается сообщение о том, что пароль пользователя введен неверно (`Неверный пароль пользователя`). Пользователь к БД не допускается.

# Контроль доступа в процессе обработки запроса

Обработка SQL-запроса состоит из нескольких этапов:

1) трансляция запроса и привязка его элементов к физическим объектам БД;
2) выполнение запроса;
3) передача результата выполнения запроса пользователю.

Проверка прав доступа выполняется на каждом этапе.

На первом этапе проверяется, в основном, логическая защита на самом высоком (общем) уровне.

Второй этап, в свою очередь, также состоит из нескольких этапов. При этом объекты, участвующие в запросе, последовательно вовлекаются в процесс обработки.

В СУБД ЛИНТЕР **права доступа проверяются только в момент привлечения очередного объекта к процессу обработки запроса**. Таким образом, СУБД может потратить довольно продолжительное время на обработку запроса, пока не определит, что она невозможна.

С другой стороны, если в процессе обработки запроса кто-то изменяет права субъекта, то данные изменения могут сказаться уже на обработке текущего запроса. По крайней мере, при таком подходе возрастает вероятность того, что СУБД примет во внимание изменение прав доступа.

Важность того, что проверка доступа перенесена в этап обработки, обусловлена освобождением первых двух этапов от дополнительных проверок. Это позволяет прикладным программам свободно использовать уже оттранслированные и привязанные запросы.

# Прямая передача прав

В СУБД ЛИНТЕР в качестве одного из основополагающих принят принцип прямой передачи прав.

Для того чтобы исключить возможность неконтролируемого распространения прав, принят следующий принцип: **прямая передача прав на объект разрешена только для владельца данного объекта**.

Субъект, получивший какие-либо права на чужой объект, не может осуществить прямую передачу прав.

# Косвенная передача прав

Косвенная передача/отмена прав на объект возможна только через аппарат ролей.

В случае косвенной работы роль должна быть наполнена соответствующими возможностями, предоставленными напрямую владельцами объектов.

Создатель роли может назначать ее другим пользователям (в том числе и себе). Владельцы объектов могут после этого продолжать назначать права на свои объекты роли или отнимать их у нее.

# Регистрация действий

Регистрация действий выполняется подсистемой протоколирования СУБД.

Запуск подсистемы протоколирования выполняется SQL-запросом:

```
AUDIT START;
```

Для выполнения запроса необходимы права `DBA`.

В результате в нулевой записи таблицы аудита устанавливается флаг активности подсистемы протоколирования.

Останов подсистемы протоколирования выполняется SQL-запросом:

```
AUDIT STOP;
```

Для выполнения запроса необходимы права `DBA`. В результате в нулевую запись таблицы аудита заносится информация о прекращении протоколирования.

При возобновлении работы подсистемы протоколирования продолжится протоколирование всех событий, установленных до остановки протоколирования.

В режиме протоколирования событий в таблице аудита могут быть зарегистрированы действия пользователей БД, связанные с:

- идентификацией и аутентификацией;
- запросами на доступ к объектам защиты БД;
- созданием/уничтожением объектов защиты БД;
- действиями по изменению ПРД.

Таблица регистрации событий аудита создается в процессе инициализации подсистемы расширенных функций КСЗ НСД СУБД ЛИНТЕР (см. пункт Инициализация комплекса средств защиты данных). Более подробно о таблице регистрации изложено в разделе Мониторинг комплекса средств защиты данных.

# Метка доступа

Всем объектам контроля присваивается **метка доступа**.

Метки доступа используются СУБД ЛИНТЕР для проверки доступности информации, а также для возможности организации принудительного управления доступом. Механизм меток доступа – главное звено обеспечения мандатного принципа контроля доступа.

Метка доступа содержит информацию об уровне конфиденциальности (для объектов, то есть данных) – **метку объекта**, и уровне доступа (для пользователей, субъектов) – **метку субъекта**.

Кроме иерархии уровней в метке содержится также информация о **группе**. Речь идет о группах пользователей, которые должны быть изолированы друг от друга (в смысле информации).

Соответственно группам пользователей объекты данных тоже разбиваются на группы.

Таким образом, метки предоставляют как иерархию доступа (уровни), так и горизонтальное деление (объектов, субъектов).

Метками доступа снабжается информация всех уровней – от таблицы, до столбца и записи и даже значения полей записи. То есть в СУБД ЛИНТЕР есть возможность снабжать все объекты доступа специальными метками (об их содержании упоминается ниже).

Все субъекты доступа также снабжаются метками. При проверке доступа субъекта к конкретному объекту СУБД осуществляет дополнительную проверку, не выполняя недоступные действия.

**Примечание**
СУБД ЛИНТЕР хранит информацию о файлах своей БД. В активном состоянии СУБД использует файлы БД монопольно и блокирует доступ к ним других программных средств. Ограничение доступа к файлам БД при неактивном состоянии СУБД должно выполняться средствами ОС.

## Метка объекта

Метка объекта обеспечивает физическую защиту данных.

Метка объекта включает:

- группу субъекта, который создал данный объект;
- уровень доступа на чтение: RAL-уровень (`Read Access Level`);
- уровень доступа на запись: WAL-уровень (`Write Access Level`).

Появление в БД нового объекта контроля сопровождается присвоением ему соответствующей метки объекта, которая будет связана с ним до момента его уничтожения.

Даже перемещение (копирование) объектов (например, из одной таблицы в другую) не изменит их защитных атрибутов (по крайней мере, не снизит). Так, изменить метку записи таблицы можно только при замене всей информации этой записи, что, собственно, равносильно ее удалению и добавлению вновь.

## Метка субъекта

Метка субъекта включает:

- группу, к которой принадлежит субъект;
- RAL-уровень субъекта – максимальный RAL-уровень доступной субъекту информации;
- WAL-уровень субъекта, то есть минимальный RAL-уровень объекта, который может быть создан данным субъектом.

В отличие от меток объектов, метки субъектов (пользователей) могут быть изменены. Изменение меток доступа субъектов – прерогатива субъекта (администратора безопасности).

Он может перевести субъекта из группы в группу, изменить уровни доступа и доверия. Естественно, что изменение возможностей субъекта не должно остаться незамеченным – все подобные действия **протоколируются** в таблице аудита.

# Схема хранения данных

Информация обо всех субъектах, ролях и их возможностях по отношению к объектам БД хранится в словаре БД.

Для хранения данной информации предназначена системная таблица `$$$USR` (см. документ «Системные таблицы и представления», раздел «$$$USR»).

Поддерживаемые категории прав: `DBA`, `RESOURCE`, `CONNECT` и `BACKUP`.

Предоставляемые привилегии на манипулирование данными таблицы: `SELECT`, `INSERT`, `DELETE`, `UPDATE`, `ALTER`, `INDEX`, `REFERENCES` и `ALL` (все перечисленные ранее).

Категории прав имеют только пользователи. Привилегии – пользователи и роли.

# Категория доступа субъекта контроля

Пользователю назначается одна из 3-х категорий доступа к объектам БД:

1) CONNECT-категория. Предоставляет пользователю наименьшие права: доступ к БД с возможностью подавать SQL-запросы на манипулирование данными;
2) RESOURCE-категория. Предоставляет пользователю все права Connect-категории, а также право изменять схему БД (создавать/удалять/изменять структуру объектов БД) и передавать другим пользователям права на свои объекты;
3) DBA-категория. Предоставляет пользователю уровень администратора БД с максимальными правами, включающими права Resource-категории и право создавать новых пользователей БД.

Владельцем табличного объекта БД (таблицы/представления) считается пользователь, создавший этот объект (для его создания он должен иметь, как минимум, RESOURCE-категорию). Владелец табличного объекта получает на этот объект все возможные привилегии, вплоть до привилегии передачи (отмены) некоторых из них другим пользователям БД.

Владелец может передать на свой табличный объект следующие привилегии (или их совокупность):

- SELECT – на чтение данных;
- INSERT – на добавление данных;
- UPDATE – на модификацию данных;
- DELETE – на удаление данных;
- ALTER – на изменение параметров таблицы;
- INDEX – на построение/удаление индексов таблицы;
- REFERENCES – на создание внешних ключей, ссылающихся на таблицу;
- ALL – полный набор привилегий: SELECT + INSERT + UPDATE + DELETE + ALTER + INDEX + REFERENCES.

Удалить объект БД может только его владелец. Администратор БД может удалить объект каскадно, вместе со всей содержащей его схемой.

Для вызова хранимой процедуры необходимо иметь привилегию EXECUTE или EXECUTE AS OWNER.

Для оперативного архивирования БД целиком или её отдельных объектов средствами СУБД необходима привилегия BACKUP (по умолчанию эту привилегию имеет только создатель БД).

Назначение категории доступа субъекту контроля КСЗ НСД СУБД ЛИНТЕР может быть осуществлено следующими способами:

1) создание субъекта контроля с неявным заданием CONNECT-категории доступа:

   CREATE USER <имя пользователя> IDENTIFIED BY <пароль>;

2) создание субъекта контроля с явным заданием категории доступа:

   GRANT <категория> TO <имя пользователя> [IDENTIFIED BY <пароль>];

3) назначение (изменение) категории доступа существующему субъекту контроля:

   GRANT <категория> TO <имя пользователя>;

Если база данных работает в режиме CERTIFIED (см. документ «Создание и конфигурирование базы данных», пункт «Создание БД»), то любому пользователю, имеющему уровень прав DBA, дополнительно разрешаются следующие действия (DCL и DDL):

- назначение и отмена любых привилегий на любые таблицы;
- назначение и отмена любых привилегий на запуск любых процедур;
- создание и удаление таблиц, изменение их структуры в схемах любых пользователей (например, построение и удаление индексов);
- создание и удаление хранимых процедур, изменение и удаление их исходного текста, перетрансляция процедур в схемах любых пользователей;
- создание, удаление, разрешение и запрет триггеров на таблицы в схемах любых пользователей;
- любые операции с последовательностями в схемах любых пользователей.

**Примечание**
Поддерживается со сборки 6.0.20.4.

# Роли

Аппарат ролей используется для разграничения доступа субъектов контроля СУБД ЛИНТЕР к объектам:

1) создание роли:

   CREATE ROLE <имя роли>;

2) удаление роли:

   DROP ROLE <имя роли>;

3) присвоение роли пользователю:

   GRANT ROLE <имя роли> TO {<имя пользователя> [, …] | PUBLIC};

4) отмена присвоения роли:

   REVOKE ROLE <имя роли> FROM {<имя пользователя> [, …] | PUBLIC};

Если пользователь создает роль, то по умолчанию она ему не назначена (то есть по умолчанию у него нет привилегий этой роли). Но он может как назначить ее себе (как и всем другим) и получить ее привилегии, так и отнять ее у себя (как и у всех других) и опять не иметь ее привилегий.

# Управление привилегиями доступа

Привилегии доступа могут быть назначены как пользователям, так и ролям:

1) назначение привилегии доступа пользователю:

   GRANT <привилегия> ON <имя объекта> TO <имя пользователя>;

2) отмена привилегии доступа пользователю:

   REVOKE <привилегия> ON <имя объекта> FROM <имя пользователя>;

3) назначение привилегии доступа роли:

   GRANT <привилегия> ON <имя объекта> TO <имя роли>;

4) отмена привилегии доступа роли:

   REVOKE <привилегия> ON <имя объекта> FROM <имя роли>;

**Примечание**
Для доступа к объекту БД с требуемой привилегией данная привилегия должна быть назначена пользователю либо с помощью механизма назначения привилегий пользователям, либо с помощью механизма назначения привилегий роли, при назначении этой роли пользователю, либо при назначении привилегии доступа к объекту `PUBLIC` (под `PUBLIC` имеется в виду команда вида "grant <привилегия> on <объект> to PUBLIC").

То есть, для пользователя действует совокупность (объединение) всех привилегий дискреционного доступа: полученных напрямую, через роли, через PUBLIC и привилегии, которые он имеет, как владелец объектов.

# Группы пользователей

Все пользователи БД разделяются на непересекающиеся группы.

Группа описывает область доступных членам группы данных. Для каждой группы существует администратор группы (DBA-группы), созданный создателем БД (группа 0). Принадлежность группе задается командой ALTER.

Пользователи одной группы не видят данных пользователей другой группы.

**Примечание**
Группы не считаются заранее данными, каждую из них перед использованием необходимо создать.

Для управления группами создателем БД используются запросы:

- создание группы;
- изменение имени группы;
- назначение группы пользователю;
- предоставление доверия группе;
- отмена доверия группе.

## Создание группы

[1] <создание группы>::=
CREATE [IF NOT EXISTS] GROUP <спецификация группы>
[2] <спецификация группы>::=
<имя группы> [=<числовой идентификатор группы>]
[3] <имя группы>::= <символьный литерал>
[4] <числовой идентификатор группы>::= <целочисленный литерал>

<числовой идентификатор группы> представляет собой числовое значение в диапазоне [1-250]. Значения в диапазоне [251-255] зарезервированы.

Группа 0 – группа создателя БД. При создании пользователя ему автоматически присваивается группа создателя.

Только создатель БД может создать пользователя в группе, отличной от своей.

Примеры.

```
create group "1-й отдел";
create group age50;
create group "Военпреды"=8;
create group "1";
```

При создании группы в таблицу $$$GROUP заносится запись с RowId, равным числовому идентификатору группы. Если он не указан, то присваивается первый свободный.

# Изменение имени группы

[1] <изменение имени группы>::=
ALTER GROUP <спецификация группы> SET <спецификация группы>

Примеры.

```
alter group "первый отдел" set "группа маркетинга";
alter group age50 set age45;
```

# Назначение группы пользователю

Назначение группы пользователю возможно через команду определения пользователя (1-й способ) или через команду модификации определения пользователя с помощью дополнительной конструкции `GROUP (<спецификация группы>)` (2-й способ).

1-й способ:

[1] <назначение группы пользователю>::=
CREATE USER <имя пользователя> IDENTIFIED BY '<пароль>'
GROUP <спецификация группы>

2-й способ:

[1] <назначение группы пользователю>::=
ALTER USER <имя пользователя> GROUP <спецификация группы>

Примеры.

```
create user a identified by '12345678' group "1-й отдел";
alter User b Group Admin;
```

# Предоставление доверия группе

Пользователи группы `A` могут видеть данные пользователей группы `Б` в случае, если `DBA` группы `Б` установил флаг доверия группе `A`:

[1] <предоставление доверия группе>::=
GRANT ACCESS ON <группа-доверитель> TO {<группа-приемник> | ALL}

[2] <группа-приемник>::=
спецификация группы, которой «оказывают доверие».

[3] <группа-доверитель>::=
спецификация группы, к данным которой могут обращаться пользователи <группы-приемника>

Примеры.

```
grant access on "1-й отдел" to "Военпреды";
grant access on age45 to all;
```

# Отмена доверия группе

[1] <отмена доверия группе>::=
REVOKE ACCESS ON <группа-доверитель> FROM {<группа-приемник> | ALL}

[2] <группа-приемник>::=
спецификация группы, которой «отказывают в доверии».

[3] <группа-доверитель>::=
спецификация группы, которая отказывает <группе-приемнику> в доступе к своим данным.

Примеры.

```
revoke access on "Военпреды" from "группа маркетинга";
revoke access on age45 from all;
```

Уровни доверия не могут быть вложенными.

Все данные, созданные от имени пользователя, помечаются его группой.

Описания групп хранятся в системной таблице `$$$GROUP` (см. документ «Системные таблицы и представления», раздел «$$$GROUP»).

Описание уровней доверия зарезервировано для дальнейшего использования.

Имя группы, как и описание, может изменить либо пользователь с привилегией `DBA`, либо создатель БД при помощи SQL-предложения `ALTER GROUP`.

Номер группы входит в метку всех объектов БД.

# Уровни доступа

Уровни доступа вводятся для проверки на уровне ядра СУБД ЛИНТЕР прав на осуществление чтения/записи информации.

Вводятся следующие уровни доступа:

1) для пользователя (субъекта):
   - RAL-уровень доступа. Пользователь может получать (читать) информацию, RAL-уровень которой **не выше** его собственного уровня доступа;
   - WAL-уровень доверия на понижение уровня конфиденциальности. Пользователь не может вносить информацию с уровнем доступа (RAL-уровнем) более низким, чем данный WAL-уровень пользователя. То есть пользователь не может сделать доступную ему информацию менее конфиденциальной, чем указано в данном параметре.
2) для информации:
   - RAL-уровень чтения. Пользователь может получать (читать) информацию, RAL-уровень которой не выше его собственного RAL-уровня (может читать менее конфиденциальные данные);
   - WAL-уровень ценности или уровень доступа на запись (модификацию, удаление). Пользователь может модифицировать (удалять) информацию, WAL-уровень которой не выше его RAL-уровня.

Возможные соотношения уровней представлены в таблице 1.

Таблица 1. Соотношения уровней при выполнении DML-операций

| | **RALп** | **WALп** |
|---|---|---|
| RALи | (чтение) RALи ≤ RALп | (запись) RALи ≥ WALп |
| WALи | (модификация, удаление) WALи ≤ RALп | - |

RALп – RAL-уровень доступа пользователя. WALп – WAL-уровень доверия пользователя на понижение уровня конфиденциальности. RALи – RAL-уровень чтения информации. WALи – WAL-уровень ценности или уровень доступа на запись (модификацию, удаление) информации.

Контролируются 10 уровней доступа (номера 1-10). Уровни 11-15 зарезервированы. Всем создаваемым пользователям БД по умолчанию устанавливается уровень доступа 0 (признак отсутствия контроля по мандатному доступу, то есть пользователи имеют возможность читать и модифицировать все доступные им по мандатному принципу контроля данные).

Создать пользователя с произвольными RAL/WAL-уровнями может только создатель БД. Остальные администраторы БД (пользователи категории `DBA`) имеют право создавать пользователей БД (или изменять им уровень доступа) только в пределах отведенных данному администратору RAL/WAL-уровней (на чтение – не выше, на запись – не ниже).

Пользователь может явно задавать уровень доступа вводимых данных, указывая в списке атрибутов уровни доступа для соответствующих записей и полей (при операции `INSERT` или `UPDATE`).

RAL-уровень и WAL-уровень вносимых данных берутся как максимум установленных RAL-уровня и WAL-уровня для канала/пользователя, вносящего/изменяющего данные.

Защищаемые объекты: пользователи, таблицы, столбцы, записи (вносятся при `INSERT`), поля записей (изменяются при `UPDATE`).

### Пример

```
create if not exists level "NS"  = 1;
create if not exists level "DSP" = 2;
create if not exists level "S" = 3;
create if not exists level "C" = 4;
create if not exists level "CC" = 5;

drop user U1 cascade;
create user U1 identified by '12345678';
grant dba to U1;
alter user U1 level("S", "C");

! <Метка доступа> пользователя ##3#4.
username U1/12345678
! <Метка доступа> таблицы ##3#4
! (наследуется от <метки доступа> создателя).
create or replace table TAB1 (id int, name char(10));
! <Метка доступа> добавляемой записи ##4#4.
! Пользователь может внести строку ##4#4,
! так как RAL-уровень информации совпадает с WAL-уровнем
! пользователя. Но у пользователя нет доступа на запись
! в таблицу, так как WAL-уровень таблицы выше RAL-уровня
! пользователя.
! ошибка 1070
insert into TAB1 values (101,'new1');
! <Метка доступа> добавляемой записи ##3#4.
! Пользователь не может внести строку ##3#4,
! так как RAL-уровень информации ниже WAL-уровня пользователя.
```

```
! ошибка 1070
insert into TAB1##3#4 values (102,'new2');
! ошибка 1070
insert into TAB1##4#4 values (103,'new3');
username SYSTEM/MANAGER8

drop user U2 cascade;
create user U2 identified by '12345678';
grant dba to U2;
alter user U2 level("S", "DSP");

! <Метка доступа> пользователя ##3#2.
username U2/12345678

! <Метка доступа> таблицы ##3#2.
create or replace table TAB2 (id int, name char(10));
insert into TAB2##4#4 values (102,'new2');
```

Информация об уровнях доступа хранится в системной таблице `$$$LEVEL` (см. документ «Системные таблицы и представления», раздел «$$$LEVEL»).

# Управление уровнем доступа пользователей

Для управления уровнями доступа пользователей создателем БД используются следующие запросы:

### Создание уровня

[1] <создание уровня>::=
CREATE [IF NOT EXISTS] LEVEL <имя уровня> = <номер уровня>
[2] <имя уровня>::= идентификатор
[3] <номер уровня>::= целочисленный литерал в диапазоне [1-10]

Пример.

```
create level L10 = 10;
```

**Примечание**
Номер уровня можно использовать в качестве идентификатора при определении доступа, например, `AA#10#2#3`.

### Переименование уровня

[1] <переименование уровня>::=
ALTER LEVEL <имя уровня> SET <новое имя уровня>

Выполнение данного запроса доступно только создателю БД.

### Назначение уровня доступа пользователю

Назначение уровня доступа пользователю возможно через команду определения пользователя либо команду модификации определения пользователя с помощью дополнительной конструкции `LEVEL` (<RAL>,<WAL>):

 © Администрирование комплекса средств защиты данных. АО НПП «РЕЛЭКС», 1990-2026

[1] <назначение уровня доступа пользователю>::=
CREATE USER <имя пользователя> IDENTIFIED BY '<пароль>'
LEVEL (<RAL-уровень>, <WAL-уровень>);
[2] <назначение уровня доступа пользователю>::=
ALTER USER <имя пользователя>
LEVEL (<RAL-уровень>, <WAL-уровень>);

Произвольные уровни доступа может назначать/изменять только создатель БД, прочие пользователи БД могут назначать/изменять уровни доступа в пределах назначенных им уровней.

Примеры.

```
create user a identified by '12345678'
  level("СОВ.СЕКРЕТНО","ДЛЯ СЛ.ПОЛЬЗОВАНИЯ");
  alter user b level("СОВ.СЕКРЕТНО","ДЛЯ СЛ.ПОЛЬЗОВАНИЯ");
```

## Управление уровнем доступа к таблице

В операторах ALTER TABLE и CREATE TABLE можно указать уровень доступа для всей таблицы через конструкцию LEVEL (<RAL>,<WAL>), например:

```
create table t (h INT, j char(10))
  level("СОВ.СЕКРЕТНО","ДЛЯ СЛ.ПОЛЬЗОВАНИЯ");
```

**Примечание**
Запрещено создание таблицы с высоким уровнем секретности на устройстве с низким уровнем секретности.

## Управление уровнем доступа к столбцу таблицы

В операторах ALTER TABLE и CREATE TABLE можно указать уровень доступа для столбца через конструкцию LEVEL (<RAL>,<WAL>), например:

```
create table t(h int
  level("СОВ.СЕКРЕТНО","ДЛЯ СЛ.ПОЛЬЗОВАНИЯ"), j char(10));
```

Если для таблицы заданы уровни мандатного доступа, а для каких-то ее столбцов не заданы, то уровни мандатного доступа таких столбцов берутся равными уровням доступа таблицы.

# Представление групп и уровней в SQL-выражениях

Выше было определено, что метка доступа (или метка пользователя) всегда состоит из трех составляющих: номера (или имени) группы и двух номеров (имен) уровней доступа (RAL, WAL):

[1] <метка доступа>::= #[[*]<группа>]#[[*]<RAL>]#[[*]<WAL>]

<метка доступа> может сопровождать как всю таблицу (запись таблицы), так и столбец таблицы (поле записи).

Конструкция вида ### присваивает полю метку доступа текущего пользователя, а конструкция вида #*#*#* оставляет метку доступа поля без изменения.

Примеры.

```
update TEST set i###=200 where i=3;
update TEST#*#*#* set i=200 where i=3;
update TEST#*#"ДСП"#* set i=200 where i=3;
```

# Получение информации о метках доступа

Для получения информации о метке доступа используется встроенная в SQL функция.

### Синтаксис

SECURITY({*|<имя столбца>},{'R' |'W' |'G'});

### Описание

1) `'R' | 'W' | 'G'` – тип части метки доступа или указание для выдачи значений `RAL`, `WAL` и `GROUP` соответственно;
2) функция всегда возвращает значение типа integer, являющееся значением указанной части метки. При этом если:
   - в качестве первого параметра употреблен символ звездочка (*), то результатом функции будет служить значение указанной (вторым параметром) части метки **текущей строки**;
   - в качестве первого параметра употреблено имя столбца, то результат функции – значение указанной части метки соответствующего **столбца** текущей строки.
3) функцию можно использовать в тех местах SQL-запроса, где по синтаксису допустимо использование скалярной функции. Это может быть получение справочной информации – на выходе SELECT-запроса или в поисковом условии (в WHERE-предложении).

# SELECT-выражения

Кроме вышеописанной функции `SECURITY` в SELECT-выражениях нет специальных возможностей для использования меток, конфиденциальности и пр. Все действия по проверке доступа производятся ядром СУБД ЛИНТЕР прозрачно для пользователя.

**Примечание**
Выполняя поисковые операции, ядро СУБД будет принимать во внимание **только доступные (видимые)** пользователю данные. Таким образом, у каждого пользователя будет свое представление о содержании данных в БД или ее таблицах.

Следовательно, субъекты, обличенные большим доверием, будут **видеть** больше информации там, где другие пользователи **найдут** лишь небольшую часть данных (доступных им).

Однако если при выполнении поисковой операции встретилась секретная запись, удовлетворяющая условиям поиска, но не соответствующая уровню доступа пользователя, такая запись не игнорируется, а выдается код завершения `1070 (Нарушение мандатного доступа)`.

Здесь исключены все возможные, даже косвенные, лазейки, так как правило видимости касается и таких конструкций, как агрегатные функции. Так, информация о минимальном значении данных какого-либо столбца будет выдана, исходя из

доступного пользователю множества значений этого столбца. То же касается и остальных агрегатных функций (MIN, MAX, AVG, COUNT и др.).

# UPDATE, INSERT и MERGE операторы

<Метка доступа> в этих операторах может относиться как ко всей таблице (строке), например,

```
insert into auto<метка доступа> values (...);
```

так и отдельным полям, например,

```
insert into auto (PersonID<метка доступа>) values (10000);
```

В первом случае считается, что метка доступа относится ко всей строке, во втором – только к указанному полю.

Если пропущена <метка доступа> строки, то по умолчанию группа берется равной группе пользователя, а RAL-уровень и WAL-уровень устанавливаются как максимум RAL-уровня и WAL-уровня пользователя. То есть для пользователя U1 с <меткой доступа> #1#5#1 вносимые им данные будут иметь <метку доступа> #1#5#5.

Если в конструкции UPDATE для строки или для поля <метка доступа> не указана, она остается той же, что была до корректировки строки (поля), при условии, что определяемый ею уровень доступа разрешен пользователю для этой операции. В противном случае (когда, например, секретный пользователь меняет не секретные данные без указания уровней) выдается код завершения 1070 (Нарушение мандатного доступа).

Если задана <метка доступа> поля, то результирующий уровень доступа для поля и для записи устанавливается как максимум уровней записи и поля.

Пример.

```
create if not exists level "NS" = 1;
create if not exists level "DSP" = 2;
create if not exists level "S"  = 3;
create if not exists level "C" = 4;
create if not exists level "CC" = 5;

drop user U1 cascade;
create or replace user U1 identified by '12345678';
grant dba to U1;
alter user U1 level("CC", "NS");

username U1/12345678
! <Метка доступа> таблицы ##5#1
create or replace table tab1 (id int LEVEL("CC","C"),
  name char(10) LEVEL("CC","C"));

! <Метка доступа> записи/полей ##4#4
insert into TAB1##"C"#"C"(ID##"C"#"C",NAME##"C"#"C") values
 (101,'qwerty');
```

```
! <Метка доступа> записи/полей ##4#4
insert into TAB1##"S"#"S"(ID##"C"#"C",NAME##"C"#"C") values
 (102,'abcd');
! <Метка доступа> записи/полей ##4#4
insert into TAB1##"C"#"DSP"(ID##"C"#"C",NAME##"C"#"C") values
 (103,'zzzzzzzz');
insert into
! <Метка доступа> записи/полей ##5#5
TAB1(ID##"C"#"C",NAME##"C"#"C") values (104,'new');
! <Метка доступа> записи/полей ##5#5
insert into TAB1 values (105,'new2');
```

### Примечания

1. Если для обновляемой таблицы задан псевдоним, то спецификация мандатной защиты должна стоять после этого псевдонима.
   update <имя таблицы>[AS <псевдоним>] [#[<группа>]#[<RAL>]#[<WAL>]]
2. При добавлении новой записи с BLOB-значениями вставка BLOB-значений считается отдельной операцией и в случае, если в режиме AUTOCOMMIT ошибка произойдёт при вставке BLOB-значений, то отката добавления записи не произойдёт.

# Управление мандатной защитой на уровне сессии

Для того чтобы не указывать группу, RAL и WAL в каждом INSERT- или UPDATE-запросе, выполняемом пользователем в текущей сессии, можно использовать команду:

SET SESSION DEFAULT SECURITY #[<группа>]#[<RAL>]#[<WAL>];

Команда распространяется на все открытые дочерние каналы (которые уже открыты ранее или будут открыты впоследствии).

Пример.

```
create level "NS" = 1;
create level "DSP" = 2;
create level "C" = 3;
create level "CC" = 4;

drop user "TEST" cascade;
create user "TEST" identified by '12345678';
grant DBA to "TEST";
username TEST/12345678
create or replace table a (i int, j int);
! TEST FOR 'INSERT'
insert into a values(1,1);
username SYSTEM/MANAGER8
alter user TEST level ("DSP","DSP");
username TEST/12345678
insert into a values(2,2);
```

```
!ok
set session default security ##C#C;
insert into a values(3,3);
!ok
set session default security ##CC#CC;
insert into a values(4,4);
!2 rows
select security(*,'R'),security(*,'W') from a;
username SYSTEM/MANAGER8
alter user TEST level ("CC","CC");
username TEST/12345678
!4 rows
select security(*,'R'),security(*,'W') from a;
! TEST FOR 'UPDATE'
update a set i=10,j=10 where i=1;
!4 rows
select security(*,'R'),security(*,'W') from a;
!error: MandatoryViolation
set session default security ##NS#NS;
```

SET SESSION SECURITY #[<группа>]#[<RAL>]#[<WAL>];

В отличие от команды SET SESSION DEFAULT SECURITY, данная команда влияет не на метки доступа в INSERT- или UPDATE-запросах, а устанавливает в канале метку доступа для работы с объектами БД – таблицами, представлениями и т. д. (по умолчанию эта метка берется равной метке уровня пользователя). То есть с меткой доступа, установленной командой set session security, будут выполняться любые запросы по этому каналу. Команда также распространяется на все дочерние каналы (которые уже открыты ранее или будут открыты впоследствии).

Устанавливаемый командой уровень чтения должен быть не больше установленного в канале уровня чтения (по умолчанию он берется равным RAL-уровню доступа пользователя).

Устанавливаемый командой уровень записи должен быть не меньше установленного в канале уровня записи (по умолчанию он берется равным WAL-уровню доступа пользователя).

Для команд SET SESSION DEFAULT SECURITY и SET SESSION SECURITY можно пропускать задание группы и уровней.

При попытке смены группы в команде SET SESSION SECURITY будет выдан код завершения 1070 (Нарушение мандатного доступа) (так как в другой группе мы работать не можем), хотя для команды SET SESSION DEFAULT SECURITY такой код завершения выдаваться не будет (так как мы можем внести БД данные для другой группы).

Пример.

```
create if not exists level "NS"   = 1;
create if not exists level "DSP"  = 2;
```

```
create if not exists level "S"  = 3;
create if not exists level "C" = 4;
create if not exists level "CC" = 5;

drop user U1 cascade;
create user U1 identified by '12345678';
grant dba to U1;
alter user U1 level("CC", "NS");

username U1/12345678
drop table TAB1 cascade;
set session security ##4#4;
! ошибка 1022
create table tab1 (id int, name char(10)) LEVEL("NS","NS" );
! ошибка 1070
set session security ##4#3;
! ошибка 1070
set session security ##5#4;
! ошибки нет
set session security ##3#4;
create or replace table tab1 (id int, name char(10));

username U1/12345678
<Метка доступа> записи/полей ##3#4
insert into TAB1##3#4 values (104,'new4');
<Метка доступа> записи/полей ##4#4
insert into TAB1##4#4 values (105,'new5');
set session security ##5#1;
<Метка доступа> записи/полей ##5#5
insert into TAB1 values (106,'new6');
```

# Маркировка документов

Согласно архитектуре открытых систем, СУБД ЛИНТЕР не имеет средств вывода информации на документы, однако ЛИНТЕР предоставляет разработчику приложения все возможности для реализации маркировки документов.

Функция `SECURITY`, описанная выше, обеспечивает не только получение справочной информации для администратора средств защиты, но и является главным инструментом, позволяющим приложению маркировать документы (выводимые данные).

На самом деле, главную роль при маркировке играет уровень конфиденциальности (чтения) – `RAL`-уровень получаемых данных. При получении разноуровневых (по `RAL`-уровню) данных маркировка документов должна основываться на уровне самых конфиденциальных из них, то есть на максимальном уровне полученной информации.

Следовательно, при маркировке документов основным аппаратом программирования приложения станет функция `SECURITY(…,'R')`.

# Контроль доступа к БД с сетевых станций

СУБД ЛИНТЕР представляет собой многопользовательскую систему. Пользователи могут получать удаленный доступ к БД с различных сетевых терминальных станций (удаленных пользовательских компьютеров). СУБД ЛИНТЕР отличает различные станции сети по их **протоколу обмена** данными и **уникальному сетевому адресу** в пределах одного протокола.

СУБД ЛИНТЕР позволяет администратору системы безопасности регулировать доступ пользователей к БД по следующим критериям (в порядке очередности контроля доступа):

1) по времени работы пользователя;
2) по количеству одновременно активных логических соединений к БД;
3) по списку разрешенных для доступа станций;
4) по уровням доступа;
5) по списку разрешенных для доступа групп.

Основополагающим понятием в процессе сопоставления пользователя с устройством является понятие *сетевой станции*. Сетевой станцией СУБД ЛИНТЕР считает любой компьютер, имеющий уникальный идентификатор – адрес в сети.

СУБД ЛИНТЕР поддерживает несколько типов сетей, что требует различного подхода к интерпретации сетевых адресов.

Рассмотрим структуру сетевого адреса. В общем случае сетевой адрес состоит из *адреса подсети* (уникального в пределах всей сети) и *адреса станции* (уникального в пределах подсети). СУБД ЛИНТЕР позволяет управлять доступом, как на уровне конечной станции, так и на уровне подсети. В последнем случае ограничения, наложенные на всю подсеть, влияют и на все станции, расположенные в данной подсети.

Каждый сетевой адрес в СУБД ЛИНТЕР характеризуется следующими параметрами:

- тип сети (определяет внутреннюю структуру адреса);
- тип адреса (указывает на подсеть или конкретный узел);
- адрес в сети (собственно сетевой адрес, в зависимости от типа сети, может включать адрес подсети, а может не включать его);
- маска разрешенных групп (стандартная битовая маска, описывающая, разрешен ли доступ данной группы к станции);
- уровень мандатного доступа (проверяется возможность пользователя выполнять соответствующие операции по отношению к данной станции);
- маска разрешенного времени доступа (с точностью до получаса описывается время, разрешенное для доступа со станции).

Создать новую сетевую станцию (изменить характеристики существующей) может только администратор безопасности БД.

Право на доступ с конкретной сетевой станции может быть предоставлено администратором безопасности.

Для локальной сети, в которой все существующие сетевые станции (и добавляемые в будущем) находятся в доверенном окружении в рамках ОС и сети, доступ может быть

предоставлен ко всем сетевым станциям либо сразу всем текущим и потенциальным пользователям СУБД, либо определенной группе текущих пользователей.

При предоставлении прав могут быть установлены временные ограничения для доступа данного пользователя с конкретной сетевой станции.

При попытке установления соединения подсистема защиты СУБД выполняет следующие действия:

1) проводит стандартные проверки идентификации и аутентификации пользователя;
2) проверяет наличие у пользователя категории `Connect`;
3) получает у операционной системы тип сети и адрес сети – источника запроса на установку соединения;
4) проверяет, ограничен ли доступ для данного пользователя (по временным показателям);
5) проверяет, существует ли для данного пользователя список разрешенных или запрещенных сетевых станций;
6) если такой список существует, подвергается проверке совпадение меток доступа для пользователя и разрешенной станции (группа пользователя должна содержаться в маске групп пользователей у станции; `RAL`-уровень пользователя **не должен быть выше** `RAL`-уровня станции, `WAL`-уровень пользователя **не должен быть ниже** `WAL`-уровня станции);
7) при успешной проверке рассматривается возможность данного пользователя работать с этой станции в текущий момент времени;
8) если запрещенных комбинаций не обнаружено, доступ разрешается.

Информация о сетевых станциях хранится в системной таблице `$$$STATION` (см. документ «Системные таблицы и представления», раздел «$$$STATION»).

Имя станции представляет собой идентификатор, который может использоваться администратором безопасности для разрешения (запрещения) доступа пользователей с данного устройства.

Характеристики станции включают:

- общее число неуспешных попыток обращения со станции;
- текущее число неуспешных попыток обращения со станции;
- флаги доступа (доступ запрещен, требуется проверка группы);
- уровни доступа с устройства для мандатного доступа;
- маска разрешенных для доступа групп;
- маска временного доступа: битовое поле разрешенного времени работы со станции (с точностью до получаса на неделю);
- время последнего неудачного доступа;
- время последнего успешного доступа;
- дата и время, начиная с которого доступ с данной станции разрешается;
- дата и время, до которого доступ с данной станции разрешается;

- маска разрешенных для доступа дней недели.

Уровни мандатного доступа для сетевой станции представляют собой два числа (со значениями из отрезка [1-10]).

Маска разрешенных для доступа групп представляет собой битовую маску из 256 бит, первые 250 бит которой кодируют разрешение соответствующей группы на доступ к станции.

Дата и время, начиная с которого доступ с данной станции разрешается, представляют собой одностороннее ограничение на дату начала разрешения доступа с данной станции.

Дата и время, начиная с которого доступ с данной станции запрещается, представляют собой одностороннее ограничение на дату прекращения доступа с данной станции.

Маска разрешенных для доступа дней недели представляет собой битовую маску разрешенных для доступа дней недели.

# Создание/удаление сетевой станции

Перед использованием станции ее необходимо сделать доступной для СУБД ЛИНТЕР.

Это можно сделать двумя способами:

1) создать станцию (SQL-синтаксис приведен ниже), то есть включить ее описание в список станций (в системную таблицу `$$$STATION`);
2) перевести СУБД ЛИНТЕР в режим беспрепятственной работы с неизвестными станциями (как уже установленными в сети, так и добавляемыми в будущем) (`UNLISTED STATION`). Предполагается, что все такие станции входят (и будут входить) в доверенное окружение в рамках ОС и сети.

Для управления рабочими станциями создателем БД используются следующие команды:

1) создание станции:

```
[1] <создание станции>::=
    CREATE [IF NOT EXISTS] STATION <имя станции>
    PROTOCOL <сетевой протокол>
    ADRESS <адрес станции>
    [LEVEL (<RAL>,<WAL>)]
    [<рабочее время>];
[2] <имя станции>::= <идентификатор>
[3] <сетевой протокол>::= <символьный литерал>
[4] <адрес станции>::= <символьный литерал>
[5] <RAL>::= <идентификатор>
[6] <WAL>::= <идентификатор>
[7] <рабочее время>::=
    {ENABLE | DISABLE} LOGIN
    {ALWAYS | <график по времени>|<график по дням>}
[8] <график по времени>::=
    FROM <время начала работы>
    TO <время окончания работы>[FOR <дни работы>]
[9] <график по дням>::=
    {SINCE <дата начала>}|{UNTIL <дата окончания>}
```

[10] <время начала работы> : : = 'HH:MM'
[11] <время окончания работы> : : = 'HH:MM'
[12] <дни работы> : : =
<день недели> {[,<день недели>] ...}
[13] <день недели> : : =
'MON'|'TUE'|'WED'|'THU'|'FRI'|'SAT'|'SUN'
[14] <дата начала> : : = 'DD.MM.YYYY'
[15] <дата окончания> : : = 'DD.MM.YYYY'

Опция `IF NOT EXISTS` отменяет выполнение оператора, если указанная станция уже зарегистрирована в БД.

2) удаление станции:

DROP STATION <имя станции>;

# Изменение параметров сетевой станции

При необходимости параметры станции можно изменить с помощью SQL-предложения `ALTER STATION`, которое имеет следующий синтаксис:

ALTER STATION <имя станции> [SET <новое имя станции>]

[LEVEL(<RAL>,<WAL>)] [<рабочее время>];

# Управление доступом к станции

Запрос:

GRANT ACCESS ON UNLISTED STATION TO {<имя группы> | ALL};

разрешает работу со всех текущих и вновь добавляемых станций, которые находятся в доверенном окружении в рамках ОС и сети либо всем пользователя СУБД ЛИНТЕР, либо только пользователям указанной группы.

Запрос:

REVOKE ACCESS ON UNLISTED STATION FROM {<имя группы> | ALL};

запрещает работу с текущих и вновь добавляемых станций, которые находятся в доверенном окружении в рамках ОС и сети либо всем пользователя СУБД ЛИНТЕР, либо только пользователям указанной группы.

Следующие два запроса позволяют администратору безопасности управлять доступом с сетевых станций для идентифицированных станций.

GRANT ACCESS ON STATION <имя станции> TO {<имя группы> | ALL};

REVOKE ACCESS ON STATION <имя станции> FROM {<имя группы> | ALL};

SQL-предложение `GRANT ACCESS` разрешает доступ для группы с именем <имя группы> или для всех групп со станции <имя станции>.

SQL-предложение `REVOKE ACCESS` запрещает доступ для группы с именем <имя группы> или для всех групп со станции <имя станции>.

# Защита ввода-вывода на внешний носитель

СУБД ЛИНТЕР использует внешние устройства постоянного хранения информации для размещения таблиц данных и временных рабочих файлов.

Расположение конкретного объекта БД (файла таблицы, временного рабочего файла) внутри БД однозначно идентифицируется *четырехсимвольным идентификатором* – ЛИНТЕР-именем устройства. ЛИНТЕР-имя используется при создании новых таблиц или изменении расположения файлов старых.

Информация о соответствии ЛИНТЕР-имени устройства и соответствующего ему реального физического устройства хранится в системной таблице `$$$DEVICE` (см. документ «Системные таблицы и представления», раздел «$$$DEVICE»). Данная таблица создается администратором безопасности БД и доступна только ему.

Описание доступа к устройству (столбец `$$$DESCR`) включает следующие составляющие:

1) *метка доступа* – набор из двух значений: `RAL` и `WAL`. Служит для принудительного контроля над созданием файлов БД (файлов таблиц и временных файлов) на описываемом устройстве:
   - RAL-уровень устройства представляет собой максимальный RAL-уровень объектов (таблиц), которые могут быть размещены на данном устройстве;
   - RAL-уровень устройства представляет собой минимальный уровень доступа пользователя (RAL-пользователя), необходимый для создания этим пользователем объектов (таблиц и временных файлов) на данном устройстве;
   - WAL-уровень устройства представляет собой максимальный WAL-уровень пользователя, необходимый для удаления таблицы;
2) *маска признаков доступа* – проводится или не проводится проверка доступа к соответствующему устройству, запрещен или нет доступ к нему целиком для системы. В последнем случае все запросы на создание новых (получение информации из существующих) объектов будут отвергаться;
3) *маска разрешения доступа* для групп пользователей – описываются все 250 групп.

Существует дополнительный параметр, позволяющий управлять доступом для устройств, **отсутствующих в таблице устройств**. Он указывает, будет ли запрашиваться информация о неописанных ЛИНТЕР-устройствах у операционной системы, или запросы на работу с данными устройствами будут отвергаться.

Для управления контролем устройств создателем БД используются следующие команды:

1) создание устройства:

   CREATE [IF NOT EXISTS] DEVICE `<имя устройства>`

   DIRECTORY `<каталог>`

   [COMMENT `<комментарий>`]

   [LEVEL (`<RAL-устройства>`,`<WAL-устройства>`)];

<имя устройства>

ЛИНТЕР-имя, используемое в качестве имени устройства при указании расположения файлов создаваемых таблиц.

<каталог>

Символьная строка (длиной до 256 символов), определяющая местоположение (путь) (в терминах соответствующей ОС) файлов таблиц при их создании (в команде CREATE TABLE) или модификации местоположения этих файлов (в команде ALTER TABLE).

Оператор CREATE DEVICE создает новое ЛИНТЕР-имя, которое впоследствии может использоваться в запросах на создание (модификацию) таблиц. При создании устройства может быть указан комментарий к нему (пояснение о назначении устройства) и уровни доступа к устройству.

Опция IF NOT EXISTS отменяет выполнение оператора, если указанное ЛИНТЕР-имя устройства уже существует в БД.

2) удаление устройства:

   DROP DEVICE <имя устройства>;

   Команда удаляет ЛИНТЕР-имя устройства из списка разрешенных имен.

3) модификация параметров устройства:

   ALTER DEVICE <имя устройства> [DIRECTORY <каталог>]

   [LEVEL (<RAL-устройства>,<WAL-устройства>)];

   Команда позволяет изменить значение ЛИНТЕР-имени (путь к физическому устройству) и уровни защиты устройства.

При использовании операторов ALTER DEVICE и DROP DEVICE необходимо учитывать, что существующие файлы, расположенные на новых устройствах, могут стать недоступными для некоторых пользователей.

Значение (содержание) устройства SY00 определяется в момент старта системы **и не может быть переопределено** в произвольный каталог, но все ограничения, наложенные на него, будут выполняться обычным образом, за исключением отношений, необходимых для функционирования СУБД ЛИНТЕР и соответствующих рабочих файлов.

4) разрешить группе/всем пользователям доступ к устройству:

   GRANT ACCESS ON DEVICE <имя устройства> TO {<имя группы> | ALL};

5) запретить группе/всем пользователям доступ к устройству:

   REVOKE ACCESS ON DEVICE <имя устройства>

   FROM {<имя группы> | ALL};

   Специальными командами администратор безопасности может разрешить или запретить доступ к устройствам, которые находятся (или могут находиться в

будущем) в доверенном окружении в рамках ОС и сети как для всех текущих и будущих пользователей СУБД, так и для определенной группы текущих пользователей.

6) разрешить группе/всем пользователям доступ к незарегистрированным устройствам:

   GRANT ACCESS ON UNLISTED DEVICE TO {<имя группы> | ALL};

   Команда разрешает работу с ЛИНТЕР-именами устройств из доверенного окружения средствами операционной системы (СУБД будет запрашивать информацию о физическом расположении устройства у операционной системы).

7) запретить группе/всем пользователям доступ к незарегистрированным устройствам:

   REVOKE ACCESS ON UNLISTED DEVICE FROM {<имя группы> | ALL};

   Команда запрещает работу с ЛИНТЕР-именами устройств из доверенного окружения средствами операционной системы.

   При использовании команды `REVOKE ACCESS` необходимо учитывать, что существующие объекты, расположенные на новом месте, могут стать недоступными для соответствующих групп.

   Информация о возможности работы с ЛИНТЕР-именами произвольно используемых устройств (то есть с устройств из доверенного окружения) располагается в описании БД.

# Контроль целостности средств защиты данных

В состав комплекса средств защиты данных СУБД ЛИНТЕР входят следующие компоненты:

- подсистема дискреционного доступа;
- подсистема мандатного доступа;
- диспетчер доступа;
- подсистема очистки памяти;
- подсистема контроля внешними физическими устройствами хранения информации;
- подсистема контроля доступа с внешних устройств;
- подсистема идентификации и аутентификации;
- подсистема регистрации событий.

КСЗ СУБД ЛИНТЕР реализован в виде части исполняемого кода ядра СУБД. Главный компонент КСЗ – диспетчер доступа, который получает управление на ранних этапах инициализации СУБД до запуска процедур обслуживания пользовательских запросов.

Дальнейшая работа СУБД происходит под контролем КСЗ.

Целостность КСЗ в СУБД ЛИНТЕР эквивалентна целостности программного кода ядра СУБД ЛИНТЕР.

Для проверки целостности СУБД применяется утилита подсчета контрольных сумм, которая создает на выходе файл, содержащий контрольные суммы проверяемых

файлов. Целостность СУБД считается подтвержденной в случае, если результирующие контрольные суммы совпали с контрольными суммами, приведенными в документации.

Проверка выполняется перед каждым запуском ядра СУБД.

Подсчет контрольных сумм ведется только для указанных в таблице 2 файлов с помощью утилиты count. Запуск данной утилиты (поставляемой в составе дистрибутива СУБД ЛИНТЕР) производится с помощью командной строки:

```
count <имя файла>
```

Значение контрольной суммы выдается на терминал, и его нужно только сравнить с приведенными в таблице 2 значениями.

Таблица 2. Проверяемые на целостность исполняемые модули СУБД ЛИНТЕР

| Имя файла | Тип ОС | Размер (в байтах) | Функциональное назначение | Контрольная сумма |
|---|---|---|---|---|
| gendb | | | Создание и конфигурирование БД | |
| linter | | | Ядро СУБД ЛИНТЕР | |
| sql[1)] | | | SQL-транслятор | |
| intsrt[1)] | | | Процессор сортировки | |
| tsp[1)] | | | Транслятор процедурного языка | |
| loltp | | | Управление распределенными транзакциями | |
| linapid | | | JDBC-сервер | |
| count | | | Утилита подсчета контрольной суммы файла | |
| testdb | | | Утилита тестирования БД | |
| dbs_tcp | | | Сетевой драйвер сервера | |
| dbc_tcp | | | Сетевой драйвер клиента | |

[1)] При наличии указанного файла в установочном каталоге СУБД.

При запуске ядра СУБД проверяется целостность файлов конфигурации базы данных `1.11`, `2.11`, `3.11`. В случае нарушения целостности файлов конфигурации базы данных на консоль и в файл протокола `linter.out` будут выданы сообщения о неуспешной проверке контрольной суммы, ядро запущено не будет.

При восстановлении базы данных из архива `lhb` проверяется целостность блоков архива. В случае нарушения целостности восстановление будет завершено ошибкой, на экран будет выведена диагностическая информация о месте нарушения целостности.

При запуске хранимых процедур выполняется проверка целостности кода процедуры. В случае нарушения целостности кода процедуры в выполнении процедуры будет отказано, функция вызова процедуры вернет соответствующий диагностический код, в файл протокола `linter.out` будет выдано сообщение о неуспешной проверке контрольной суммы. Процедура проверки целостности кода процедур выполняется: при вызове хранимой процедуры, при запуске ядра СУБД, а также не реже одного раза в сутки с протоколированием запуска процесса проверки в файл протокола `linter.out`.

# Подсчет контрольной суммы

## Назначение

Для контроля целостности КСЗ СУБД ЛИНТЕР используется утилита count, которая подсчитывает 32-битную контрольную сумму файла. Ее использование также целесообразно для контроля неизменности всех исполняемых модулей, контрольные суммы для которых прилагаются.

## Описание

Производится расчет 16-байтной последовательности символов, однозначно идентифицирующих заданный файл. Расчет осуществляется с использованием распространенного алгоритма вычисления аутентифицирующих кодов Message Digest в режиме сцепления по промежуточному результату вычислений.

Затем результат суммируется со сдвигом для получения результирующей 32-битной контрольной суммы.

## Использование

Командная строка:

```
count <имя файла>
```

Значение контрольной суммы выдается на терминал, и его нужно только сравнить с приведенными в таблице 2 значениями.

# Диспетчер доступа

Диспетчер доступа СУБД ЛИНТЕР реализован в виде нескольких взаимосвязанных программных фрагментов, каждый из которых контролирует свою область действий СУБД.

Первая часть обслуживает подсистему **дискреционного доступа**. Она получает управление после разбора запроса. В этот момент полностью определены все объекты доступа и запрашиваемые по отношению к ним действия. Диспетчер выбирает все правила, касающиеся запрашивающего пользователя, и, в соответствии с запросом, проверяет, доступны ли требуемые объекты в данном запросе. Также проверяется возможность работы с внешними устройствами. Данный фрагмент может обращаться к подсистеме идентификации за подтверждением (запретом) действий, связанных с категориями пользователей (`Connect`, `Resource`, `DBA`).

Вторая часть обслуживает подсистему идентификации и аутентификации. Она получает управление в момент открытия логической связи с СУБД при изменении прав пользователей и таблицы сетевых станций. Данный фрагмент обеспечивает идентификацию, аутентификацию, проверку категорий доступа, проверку сопоставления пользователей с устройством.

Третий фрагмент получает управление при каждой операции манипуляции с конкретными данными – при получении их из БД, проведении операций над ними, записи в таблицы БД, выдаче данных пользователю. В этом случае проверяются **мандатные правила доступа**.

Четвертый фрагмент диспетчера доступа СУБД ЛИНТЕР обеспечивает изоляцию параллельно выполняющихся запросов. Он также осуществляет планирование

обработки запросов, контролирует их квантование и отслеживает все возможные нарушения изоляции процессов, выполняющихся по разным логическим связям.

Все фрагменты диспетчера доступа в процессе работы обращаются к подсистеме регистрации событий СУБД ЛИНТЕР, которая может принять решение согласно своим параметрам о регистрации соответствующих событий в таблице событий.

# Создание БД

Создание БД ЛИНТЕР выполняется с помощью утилиты `gendb` (см. документ «Создание и конфигурирование базы данных», пункт «Создание БД»). В процессе генерации создаются файлы системной и рабочей БД.

# Системная БД

Системная БД представляет собой набор из трех таблиц, содержащих метаинформацию, то есть данные о данных:

1) `$$$SYSRL` (файлы с именами `1.*`) – таблица таблиц. Содержит информацию о таблицах: имя, идентификатор владельца, число столбцов, размеры файлов и пр. Это каталог таблиц БД, выполненный также в виде таблицы;
2) `$$$ATTRI` (файлы с именами `2.*`) – таблица столбцов. Содержит информацию о столбцах таблиц: имя, принадлежность к таблице, тип данных, информация об индексах, ограничения целостности и пр.;
3) `$$$USR` (файлы с именами `3.*`) – таблица полномочий. Содержит информацию о пользователях и их полномочиях при работе с таблицами, ролях и пр.

Кроме описаний пользовательских таблиц, системные таблицы содержат и свои собственные описания.

После генерации системной БД с помощью утилиты `gendb` она будет содержать следующие таблицы:

- `$$$SYSRL` – описания трех системных таблиц: `$$$SYSRL`, `$$$ATTRI`, `$$$USR`;
- `$$$ATTRI` – описания 13 столбцов системных таблиц;
- `$$$USR` – описание единственного пользователя БД, имеющего полномочия на чтение при работе со всеми системными таблицами (администратора БД).

При создании новых таблиц и пользовательских представлений или при изменении полномочий состояние системной БД меняется.

Создание пользовательской таблицы/представления влечет за собой добавление ее описания в `$$$SYSRL` и описаний ее столбцов в `$$$ATTRI`. При удалении таблицы СУБД выполняет обратные действия.

Доступ к любой таблице (пользовательской/системной) БД определяется по информации из системных таблиц.

Системные таблицы содержат жизненно важную для работы БД информацию.

Изменение данной информации может привести к **катастрофическим последствиям**. Во избежание изменения пользователями системных таблиц доступ к ним **запрещен**

**для всех пользователей** (даже администратор БД имеет лишь привилегию `SELECT`). Все необходимые изменения в системных таблицах в процессе работы производит только ядро СУБД ЛИНТЕР.

# Структура системных таблиц

Системные таблицы, так же, как и пользовательские, состоят из столбцов, по значениям которых возможен поиск.

Однако среди столбцов системных таблиц есть столбцы, содержащие в байтовом виде интегрированную информацию о включенном в эту таблицу объекте БД. Приложению, не являющемуся системным, эта информация чаще всего не требуется. Клиентские приложения работают с СУБД на уровне SQL-запросов к пользовательским таблицам. Всю прочую работу выполняет СУБД.

Только системным приложениям может потребоваться информация о расположении файлов БД, о том, как отличить базовую таблицу от представления, о типе, длине столбца и пр.

Состав обязательных системных таблиц:

- `$$$SYSRL` – данные о всех таблицах/представлениях БД;
- `$$$ATTRI` – данные о всех столбцах и их атрибутах по всем таблицам БД;
- `$$$USR` – данные о всех пользователях БД, ролях, назначении ролей пользователям БД и их привилегиях.

Структура системных таблиц описана в документе «Системные таблицы и представления».

# Рабочая БД

В рабочей БД содержатся минимум четыре файла (при наличии нескольких файлов системного журнала их может быть гораздо больше):

1) `1.31` – рабочий файл бит-векторов. Предназначен для свопинга бит-векторов ответов одно и многопеременных запросов;
2) `1.41` – рабочий файл хранимых процедур и быстрой загрузки;
3) `1.51` – рабочий файл сортировки. Предназначен для выполнения сортировки ответов и предложений типа `group by` в SQL-операторах;
4) `000001.61`, `000002.61`, … – файлы системного журнала. Предназначены для ведения протокола обо всех изменениях, произведенных СУБД в системной и пользовательской БД (сеансы чтения в журнале не регистрируются).

Файл бит-векторов используется ядром СУБД для хранения информации о том, какие записи вошли/не вошли в ответ (может быть промежуточным).

Программа `sort` последовательно обрабатывает запросы на сортировку, поэтому каждый раз в файле `1.51` записываются новые данные, которые сортируются до конца.

Размеры файлов должны быть определены до запуска СУБД. Однако они не остаются фиксированными на протяжении всего сеанса работы ядра СУБД, а расширяются по мере необходимости до установленных лимитов.

# Инициализация комплекса средств защиты данных

Без инициализации КСЗ СУБД ЛИНТЕР не будет поддерживать большинство из описанных выше функций (управление группами, уровни доступа, защиту устройств и пр.).

В процессе инициализации КСЗ создаются необходимые для её работы ресурсы:

1) для поддержки механизма меток должны быть созданы таблицы:
   - `$$$LEVEL`;
   - `$$$GROUP`.
2) для работы аппарата сопоставления пользователя с сетевым устройством необходимо создать таблицу `$$$STATION`;
3) для работы подсистемы защиты ввода-вывода на внешний носитель требуется создать таблицу `$$$DEVICE`;
4) для работы подсистемы регистрации событий нужно создать таблицу аудита;
5) для установления «отношений» между конкретными пользователями и конкретными объектами следует создать таблицу `$$$RELATION`.

Все вышеперечисленные таблицы создаются с помощью SQL-файлов `extsec.sql` и `security.sql`, которые должны быть выполнены (с помощью утилиты `inl`) администратором безопасности.

Для этого необходимо запустить ядро СУБД ЛИНТЕР, затем, пользуясь утилитой inl, запустить на выполнение SQL-файлы `security.sql` и `extsec.sql`:

```
inl -u SYSTEM/MANAGER8 -f security.sql
inl -u SYSTEM/MANAGER8 -f extsec.sql
```

После выполнения данного пакета запросов ядро СУБД ЛИНТЕР необходимо перезапустить. Только при следующем запуске ядра СУБД будут включены указанные механизмы КСЗ и протоколирования.

# Мониторинг комплекса средств защиты данных

СУБД ЛИНТЕР позволяет хранить одновременно информацию, принадлежащую различным пользователям, поддерживать сложную иерархическую структуру доступа к данным. Для контроля над доступом пользователей к данным и для ликвидации ошибок или противоречий в структуре защиты БД служит подсистема регистрации событий (аудит).

Данная подсистема является неотъемлемой частью КСЗ СУБД ЛИНТЕР.

СУБД ЛИНТЕР позволяет регистрировать следующие события:

- включение механизмов идентификации и аутентификации;
- запросы на доступ к ресурсам БД;
- создание, модификацию и уничтожение объектов БД;
- действия по изменению правил разграничения доступа (ПРД);
- все попытки доступа к БД;
- действия администратора БД.

Таблица регистрации событий `$$$AUDIT` имеет 8 столбцов, представленных в таблице 3. Более подробно их содержание рассматривается ниже.

**Примечание**
Средства протоколирования СУБД ЛИНТЕР предоставляют администратору всего лишь возможность сбора информации об интересующих его событиях; для ее обработки и анализа должны разрабатываться специальные приложения.

Таблица 3. Структура таблицы регистрации $$$AUDIT

| Имя столбца | Тип данных | Содержание |
|---|---|---|
| EVENTTYPE | smallint | Источник события (пользователь, ядро СУБД, КСЗ ядра СУБД) |
| EVENTID | smallint | Тип события (запрос на доступ, изменение ПРД) |
| USERNAME | char(66) | Имя пользователя, инициирующего событие |
| SOURCEADR | char(24) | Сетевой адрес источника события |
| OBJECTNAME | char(134) | Полное имя объекта, обращение к которому вызвало событие |
| OBJECTTYPE | smallint | Тип объекта, к которому относится событие |
| BODY | byte(58) | Дополнительная информация о событии |
| USERTEXT | char(240) | Пользовательское сообщение |

Таблица аудита имеет следующие особенности:

1) она не создается автоматически при создании БД. Для ее создания администратору БД необходимо выполнить файл `security.sql`;
2) запуск мониторинга КСЗ (аудита) осуществляется командой `AUDIT START`, при этом в нулевой записи системной таблицы `$$$SYSRL` устанавливается флаг активности подсистемы аудита;

3) останов аудита происходит по команде AUDIT STOP, при этом в нулевую запись системной таблицы $$$SYSRL заносится информация об остановке протоколирования. При возобновлении протоколирования все события, установленные до остановки протоколирования, будут активированы;
4) отметка в таблице регистрации может быть произведена пользователем с помощью команды AUDIT MESSAGE <текст сообщения>. Это бывает необходимо в случае, если пользователь хочет оставить администратору безопасности некое сообщение;
5) специальной характеристикой таблицы аудита является параметр RECORDS_LIMIT – число записей в данной таблице, при превышении которого будет создана новая таблица аудита;
6) для исключения потери накопленной информации должно быть разработано специальное приложение, контролирующее степень заполнения таблицы аудита и автоматически выгружающее данные из нее при угрозе переполнения;
7) изменять значение параметра RECORDS_LIMIT может только администратор БД;
8) для доступа к таблице аудита необходимо иметь соответствующие права доступа и категорию пользователя DBA;
9) СУБД ЛИНТЕР не позволяет архивировать таблицу аудита отдельно от всей БД (то есть ее можно сохранить только в архивном файле всей БД);
10) при работе с таблицей аудита СУБД ЛИНТЕР использует только один режим работы транзакций – AutoCommit.

Описание столбцов таблицы $$$AUDIT:

1) EVENTTYPE – источник события, подлежащего регистрации (таблица 6). Источником регистрируемых событий могут быть ядро СУБД ЛИНТЕР (например, при создании нового объекта) и КСЗ (например, при регистрации обращения к недоступному объекту);
2) EVENTID – идентификатор типа события (таблицы 8, 9, 10, 11 и 12);
3) OBJECTTYPE – идентификатор объекта БД (таблица 7, породившее протоколируемое событие);
4) SOURCEADR – сетевой адрес станции клиента. Это значение несет реальную информацию только при не нулевом значении типа источника (при сетевой работе);
5) USERNAME – имя схемы (пользователя). Это наименование схемы (пользователя), от имени которого инициировано событие (например, имя пользователя, который обратился к недоступному ему объекту БД);
6) OBJECTNAME – полное имя объекта БД. Это имя объекта, при обращении к которому возникло регистрируемое событие;
7) BODY – дополнительная информация о событии (таблица 4);
8) USERTEXT – пользовательское сообщение о протоколируемом событии.

Таблица 4. Структура поля BODY таблицы $$$AUDIT

| Поле структуры | Тип данных SQL | Комментарий |
|---|---|---|
| EventTime | date | Дата и время события |
| Reserved | byte[16] | Зарезервировано |
| SourceType | smallint | Тип события |
| SourcePid | int | Идентификатор процесса-сервера |
| SourceRPid | int | Идентификатор процесса-клиента |
| SourceSocket | int | Сетевой порт (сокет) процесса-клиента |
| EventStatus | int | Состояние выполнения СУБД ЛИНТЕР |

| Поле структуры | Тип данных SQL | Комментарий |
|---|---|---|
| SourceStatus | int | Зарезервировано |
| SourceSystemStatus | int | Состояние операционной системы |

Для удобства анализа информации о протоколируемых событиях при выполнении файла `security.sql` создается представление `AUDIT_EVENTS` (таблица 5).

Таблица 5. Структура представления AUDIT_EVENTS

| Столбец | Тип данных SQL | Комментарий |
|---|---|---|
| Event_number | bigint | Уникальный идентификатор события |
| Event_time | date | Дата и время события |
| Severity | char(10) | Признак серьезности события |
| Username | char(66) | Имя пользователя |
| Event_type | char(19) | Тип события |
| Eventid | char(21) | Имя события |
| Networkaddress | char(24) | Сетевой адрес клиента |
| Objectname | char(134) | Имя объекта БД |
| Sourcepid | int | Идентификатор процесса-сервера |
| Sourcerealpid | int | Идентификатор процесса-клиента |
| Socket | int | Сетевой порт (сокет) процесса-клиента |
| Status | int | Состояние выполнения СУБД ЛИНТЕР |
| Osstatus | int | Состояние выполнения ОС |
| Usertext | char(240) | Пользовательское сообщение |

Таблица 6. Идентификаторы источников событий

| Идентификатор источника | Источник события | Комментарий |
|---|---|---|
| 1 | SYSTEM EVENT | Системные события |
| 2 | RESOURCE EVENT | События, связанные с изменением структуры БД |
| 3 | AUTHORIZATION EVENT | События, связанные с подсистемой авторизации |
| 4 | TABLE EVENT | События, связанные с конкретными таблицами |
| 5 | CHANNEL EVENT | Канальные события |

Таблица 7. Коды объектов БД

| Код | Мнемоника | Комментарий |
|---|---|---|
| 0 | LOBJ_NONE | Неопределенный объект |
| 1 | LOBJ_USER | Пользователь или схема БД |
| 2 | LOBJ_ROLE | Роль |
| 3 | LOBJ_LEVEL | Уровень доступа |

| Код | Мнемоника | Комментарий |
|---|---|---|
| 4 | LOBJ_GROUP | Группа пользователей БД |
| 5 | LOBJ_STATION | Рабочая станция |
| 6 | LOBJ_NODE | Узел локальной сети |
| 7 | LOBJ_DEVICE | Устройство хранения файлов БД |
| 8 | LOBJ_TABLE | Таблица БД |
| 9 | LOBJ_VIEW | Представление БД |
| 10 | LOBJ_SYNONYM | Синоним объекта |
| 11 | LOBJ_EVENT | Событие БД |
| 12 | LOBJ_PROCEDURE | Хранимая процедура БД |
| 13 | LOBJ_TRIGER | Триггер БД |
| 14 | LOBJ_CURSOR | Курсор (выборка записей) |
| 15 | LOBJ_TABLE_ID | Идентификатор таблицы, к которой у пользователя нет доступа |
| 16 | LOBJ_SEQ | Последовательность |
| 17 | LOBJ_CSET | Кодовая страница |
| 18 | LOBJ_TRANSL | Правило трансляции кодовых страниц |
| 19 | LOBJ_CSAL | Алиас |
| 20 | LOBJ_PROC_ID | Идентификатор хранимой процедуры, при обработке которой возникла внутренняя ошибка ядра СУБД ЛИНТЕР |
| 21 | LOBJ_TRIG_STRUCT | Идентификатор сработавшего триггера (при любом окончании) |
| 22 | LOBJ_COLUMN | Столбец таблицы |
| 23 | LOBJ_PARAMETER | Параметр параметрического запроса |
| 24 | LOBJ_GLVAR | Глобальная переменная процедурного языка |

# Управление мониторингом

Для управления мониторингом предназначены следующие команды:

1) настройка протоколирования;
2) протоколирование пользовательского сообщения;
3) начать комментирование протоколируемых событий;
4) отменить комментирование протоколируемых событий.

В состав дистрибутива входит скрипт `audit_info.sh` для мониторинга и оповещения администратора о событиях безопасности. Данный скрипт запускается администратором и далее циклически с заданным интервалом на экран выдается сводная информация по журналам (параметры настраиваются в блоке переменных скрипта):

1) компактная группировка по всему журналу, какие события регистрировались в каком общем количестве;
2) то же с агрегацией событий за последний час;
3) последние 10 сообщений журнала;
4) последние 10 сообщений журнала высокой важности;

5) время последней записи в журнале;
6) сообщений высокой важности из `linter.out`, 10 последних.

# Настройка протоколирования

### Спецификация

[1] <настройка протоколирования>::=
AUDIT <команда протоколирования>
[<имя события> [<объект протоколирования>]]
[<субъект протоколирования>]
[<уровень протоколирования>]
[<условие протоколирования>]
[2] <состояние протоколирования>::=
{START | STOP | ENABLE | DISABLE | CLEAR} [SET RECORDS LIMIT
<целочисленный литерал>]
[3] <имя события>::= <идентификатор>
[4] <объект протоколирования>::= ON <идентификатор>
[5] <субъект протоколирования>::= FOR <имя пользователя>
[6] <уровень протоколирования>::=
BY {SESSION | STATEMENT | ACCESS}
[7] <условие протоколирования>::= WHEN [NOT] SUCCESS | ALWAYS

### Синтаксические правила

1) Допустимые <имена событий> приведены в таблицах 8, 9, 10, 11 и 12.
2) Допустимые типы <объектов протоколирования>:
   - `USER` <имя пользователя>;
   - `USER` <имя схемы>;
   - `ROLE` <имя роли>;
   - `LEVEL` {<имя уровня доступа> | <идентификатор уровня доступа>};
   - `GROUP` <имя группы>;
   - `STATION` <идентификатор станции>;
   - `NODE` <имя узла локальной сети>;
   - `DEVICE` <имя устройства>;
   - `TABLE` [<имя схемы>].<имя таблицы>;
   - `VIEW` [<имя схемы>].<имя представления>;
   - `SYNONYM` <имя синонима>;
   - `EVENT` <имя события>;
   - `PROCEDURE` <имя хранимой процедуры>;
   - `TRIGGER` <имя триггера>;
   - `CURSOR` <имя курсора>;
   - `LOCK` <системный идентификатор блокируемой таблицы>;
   - `SEQUENCE` <имя пользователя> <имя последовательности>;
   - `CHARACTER SET` <имя кодировки>;

- `TRANSLATION` <имя правила трансляции>;
- `ALIAS` <имя алиаса>.

Если конкретный `<объект протоколирования>` не указан, будет протоколироваться доступ ко всем объектам БД.

`<Объект протоколирования>` должен существовать на момент создания события протоколирования.

## Общие правила

1) В подсистеме аудита СУБД ЛИНТЕР предусмотрено четыре типа установок:
   - *глобальные* – установки, действующие для всех пользователей и объектов БД;
   - *объектные* – установки для конкретных объектов БД;
   - *персональные* – установки для конкретных пользователей БД;
   - *локальные* – установки между конкретными пользователями и конкретными объектами БД.

   Сначала проверяются глобальные установки, затем установки для конкретных объектов БД, затем установки для конкретных пользователей. После проверяются локальные установки.

   Если задана глобальная установка, то проверяется наличие явных запретов протоколирования для конкретного объекта БД, для конкретного пользователя, для комбинации объекта и пользователя. В противном случае (если глобальная установка не задана) проверяется наличие установок для конкретных объектов БД.

   Если установка для конкретного объекта БД, то проверяется наличие явных запретов протоколирования для конкретного объекта БД, для конкретного пользователя, для комбинации объекта и пользователя. В противном случае (если установка для конкретного объекта БД не задана) проверяется наличие установок для конкретных пользователей.

   Локальные установки проверяются в любом случае (независимо от того заданы ли глобальные установки или установки для конкретных объектов/пользователей БД).

   Примеры:
   - AUDIT ENABLE SELECT; (глобальная установка);
   - AUDIT ENABLE SELECT ON AUTO; (объектная установка);
   - AUDIT ENABLE SELECT FOR SYS; (персональная установка);
   - AUDIT ENABLE SELECT ON AUTO FOR SYS; (локальная установка).
2) Опция `<состояние протоколирования>`

   `START`

   Команда активизирует подсистему протоколирования.

   `STOP`

   Команда отключает подсистему протоколирования.

ENABLE

Команда разрешает протоколирование указанного события и должна выполняться при активной подсистеме протоколирования (после выполнения команды AUDIT START).

Результаты выполнения команды:

а) если <имя события> не задано, протоколируются все события;

б) если <имя события>, <имя пользователя> и <объект протоколирования> не заданы, то изменяются глобальные установки;

в) если задано <имя пользователя> или <объект протоколирования>, то изменяются персональные установки;

г) если <имя события> задано без указания <условия протоколирования>, то устанавливается:

- заданное <состояние протоколирования> (ENABLE);
- протоколирование только неудачных событий (when not success).

д) если <состояние протоколирования> не указывается, по умолчанию принимается BY STATEMENT;

е) если задано только <состояние протоколирования> ENABLE, включается протоколирование сразу всех неуспешных системных событий.

Пример.

```
audit enable linter error;
```

DISABLE

Команда запрещает протоколирование указанного события.

Если <имя события> не задано, запрещается протоколирование всех событий. Команда должна выполняться при активной подсистеме протоколирования (после выполнения команды AUDIT START). Если при этом не заданы <имя пользователя> и <объект протоколирования>, то изменяются глобальные установки.

Если задано <имя пользователя>, то изменяются персональные установки. Если задан <объект протоколирования>, изменяются установки для конкретного объекта. Если заданы <имя пользователя> и <объект протоколирования>, изменяются локальные установки.

В случае подачи команды audit disable <имя события> выполняются следующие действия:

- если ранее <состояние протоколирования> не было установлено, то запрос игнорируется;
- если <состояние протоколирования> ранее было установлено и в запросе не задано <условие протоколирования>, то отменяются ранее установленное <состояние протоколирования> и <условие протоколирования>;

- если <состояние протоколирования> ранее было установлено и в запросе задано <условие протоколирования>, то ранее установленное <состояние протоколирования> не изменяется, а отменяется только ранее установленное <условие протоколирования> (если было задано).

Если задано <состояние протоколирования> disable, отменяется протоколирование сразу всех системных событий.

Пример.

```
audit disable linter error;
```

**Примечание**
В текущей версии запрет протоколирования неудачно завершенных событий не поддерживается. Например, по команде audit disable create table when not success попытка создания таблицы протоколируется всегда (как при успешном завершении, так и при неуспешном).

CLEAR

Команда предназначена для отмены явного запрета протоколирования для конкретного пользователя, для конкретного объекта, для комбинации пользователя и объекта, выполненные командами AUDIT ENABLE и AUDIT DISABLE:

- AUDIT CLEAR ... WHEN SUCCESS – очищает только «успешные» события;
- AUDIT CLEAR ... WHEN NOT SUCCESS – очищает только «неуспешные» события;
- AUDIT CLEAR ... – очищает все события.

Команда AUDIT CLEAR для некоторого события и некоторого пользователя, объекта либо комбинации того и другого очищает все соответствующие установки протоколирования (и разрешение, и явный запрет). Для общей установки протоколирования некоторого события очищается только разрешение, так как явный запрет там не ставится.

Здесь вторая команда не просто отменяет действие первой, а еще и усиливает запрет. Если убрать конструкцию FOR пользователь из обеих команд, то вторая будет просто отменять действие первой:

```
AUDIT ENABLE событие FOR пользователь;
AUDIT DISABLE событие FOR пользователь;
```

Здесь вторая команда просто отменяет действие первой:

```
AUDIT ENABLE событие FOR пользователь;
AUDIT CLEAR событие FOR пользователь;
```

Здесь вторая команда тоже просто отменяет действие первой:

```
AUDIT DISABLE событие FOR пользователь;
AUDIT CLEAR событие FOR пользователь;
```

3) Опция <имя события>

Имя события, для которого устанавливается протоколирование. Список поддерживаемых событий приведен в таблицах 8, 9, 10, 11 и 12. Если имя события явно не задано, устанавливается протоколирование для всех возможных событий.

Примеры.

```
audit enable create table;
audit disable create view;
```

4) Опция <объект протоколирования>

Для событий группы "RESOURCE EVENTS" (кроме EXECUTE TRIGGER и EXECUTE PROCEDURE) в случае, если для них указан модификатор "ON <OBJECT>" выдаётся код завершения 2499 (Возможность еще не реализована).

Нереализованные команды установки аудита для событий: "CREATE DEVICE", "ALTER DEVICE", "DROP DEVICE", "CREATE STATION", "ALTER STATION", "DROP STATION", "TRUNCATE TABLE". При попытке выполнения команды AUDIT для вышеуказанных событий будет выдан код завершения 99 (Операция не реализована).

5) Опция <имя пользователя>

Имя пользователя, чьи действия должны протоколироваться. Если указан конкретный пользователь, то событие возникает при условии, что текущий пользователь совпадает с <именем пользователя>. Если <имя пользователя> не задано, протоколироваться будут действия всех пользователей БД.

6) Опция <уровень протоколирования>

BY SESSION

Событие протоколируется один раз для одного подключения (CONNECTION). При этом уровень протоколирования BY SESSION срабатывает только для установки для конкретного пользователя.

Если уровень BY SESSION задан для глобальной установки, установки для конкретного объекта БД или локальной установки, он игнорируется.

Примеры.

```
AUDIT ENABLE BY SESSION; (игнорируется)
AUDIT ENABLE FOR SYS BY SESSION; (срабатывает)
```

BY STATEMENT

Событие протоколируется один раз для каждой SQL-операции (режим по умолчанию).

BY ACCESS

Событие протоколируется для каждого обращения к данным.

7) Опция <условие протоколирования>.

WHEN NOT SUCCESS

Событие протоколируется только при неудачном завершении (режим по умолчанию).

`WHEN SUCCESS`

Событие протоколируется только в случае успешного завершения.

`ALWAYS`

Событие протоколируется в случае успешного и неуспешного завершения.

8) Опция `<параметры протоколирования>`

`SET`

Устанавливает указанный `<параметр протоколирования>`:

- `RECORDS` – максимальное количество записей в таблице аудита. При исчерпании свободных записей новые записи размещаются на месте самых старых записей;
- `DAYS` – указывает количество предшествующих текущей дате дней, в течение которых необходимо хранить записи протоколирования. Зарезервировано для дальнейшего использования.

`CANCEL`

Отменяет ранее установленные `<параметры протоколирования>`.

Конструкция `AUDIT CANCEL RECORDS LIMIT` запрещена.

9) События протоколируются в таблицы вида `$$$AUDIT_N`. где `N` - целое число. Синоним `$$$AUDIT` указывает на наполняемую в данный момент таблицу вида `$$$AUDIT_N`. Для каждой таблицы установлен лимит `1000000`. При превышении лимита, заданного командой `AUDIT SET RECORDS LIMIT <новое значение>;`, происходит создание новой таблицы, а "старая" таблицы считается архивной копией. Если `AUDIT SET RECORDS LIMIT` делает лимит записей таблицы аудита меньшим, чем уже есть записей в этой таблице, то таблица с новым лимитом должна создаться в следующем кванте работы.

10) Для очистки базы данных от накопленных архивных копий журнала предусмотрена процедура `clear_old_audit`, текст которой находится в файле `audit_full_info.sql` каталога системных словарей. Процедура вызывается с одним аргументом - числом последних таблиц аудита, которое будет оставлено в базе. При запуске процедуры без аргументов будет оставлено `10` таблиц. Если задан аргумент меньше `10`, то он заменяется на `10`. Процедура возвращает `TRUE`, если все запросы завершились без ошибок, иначе `FALSE`.

11) Процедура `SET_AUDIT_LAST` устанавливает в качестве источника данных для представления `AUDIT_EVENTS` текущую таблицу, привязанную к синониму `$$$AUDIT`. Вызывается без аргументов. Возвращает `TRUE` при успешном выполнении всех запросов, `FALSE` в случае ошибок.

12) Процедура `SET_AUDIT_ALL` устанавливает в качестве источника данных для представления `AUDIT_EVENTS` объединение по `UNION ALL` всех таблиц аудита, которые есть в базе (или нескольких последних, если задан аргумент - количество таблиц). Вызывается без аргументов или с одним числовым аргументом. Возвращает `TRUE` при успешном выполнении всех запросов, `FALSE` в случае ошибок. Если произошла замена таблицы аудита, то эту процедуру следует вызвать заново перед обращением к представлению `AUDIT_EVENTS`.

13) Для получения информации из журнала регистрации событий вместо представления `AUDIT_EVENTS` следует вызывать функцию `get_audit_events()`. Данная

функция является оберткой для представления AUDIT_EVENTS и автоматически выполняет его перестроение при ротации таблиц $$$AUDIT.

Пример.

```
select event_number, eventid, status from get_audit_events();
```

# Протоколирование пользовательского сообщения

### Спецификация

AUDIT MESSAGE <текст сообщения>;

Команда создает в таблице аудита новую запись с типом «пользовательское сообщение» и <текстом сообщения> в поле USERTEXT.

При вызове скрипта завершения работы stop_linter.sh в первом параметре можно задать сообщение, которое при включенной системе протоколирования добавит указанное сообщение в журнал:

```
sh stop_linter.sh "scheduled maintenance"
```

В журнал будет добавлена запись:

```
shutdown - scheduled maintenance - by SYSTEM
```

# Начать комментирование протоколируемых событий

### Спецификация

AUDIT SET MESSAGE <комментарий>;

Для всех событий, протоколируемых по соединению, в котором подана эта команда, в поле USERTEXT таблицы аудита начинает добавляться текст <комментария>.

Команда доступна всем пользователям БД.

### Пример

```
audit set message 'Приложение "Склад"';
```

# Отменить комментирование протоколируемых событий

### Спецификация

AUDIT CANCEL MESSAGE;

Для всех событий, протоколируемых по соединению, в котором подана эта команда, отменяется добавление в поле USERTEXT таблицы аудита ранее заданного <комментария>.

Команда доступна всем пользователям БД.

# Протоколируемые события

## Системные события

Системные события представлены в таблице 8.

Таблица 8. Системные события

| ID события | Имя события | Пояснение |
|---|---|---|
| 3 | STARTUP | Старт ядра СУБД |
| 4 | SHUTDOWN | Останов ядра СУБД |
| 5 | WARM RESTART | Теплый рестарт ядра СУБД |
| 6 | LINTER ERROR | Регистрация диагностики СУБД ЛИНТЕР |
| 7 | AUDIT START | Активация системы аудита |
| 8 | AUDIT STOP | Деактивация системы аудита |
| 78 | SET DATABASE NAMES | Установка кодировки БД для системных таблиц |
| 79 | SET RECORD SIZE LIMIT | Ограничение размера записи |
| 80 | SET NAMES | Установка кодировки соединения по умолчанию |
| 85 | SET DATABASE CHARSET | Установка кодировки БД для пользовательских таблиц по умолчанию |

# События, связанные с БД

События, связанные с БД, представлены в таблице 9.

Таблица 9. События, связанные с БД

| ID события | Имя события | Пояснение |
|---|---|---|
| 9 | CREATE TABLE | Создание таблицы |
| 10 | CREATE VIEW | Создание представления |
| 52 | CREATE SYNONYM | Создание синонима |
| 11 | CREATE PROCEDURE | Создание процедуры |
| 16 | DROP TABLE | Удаление таблицы |
| 17 | DROP VIEW | Удаление представления |
| 18 | DROP SYNONYM | Удаление синонима |
| 19 | DROP PROCEDURE | Удаление процедуры |
| 20 | DROP TRIGGER | Удаление триггера |
| 59 | CREATE STATION | Создание станции |
| 60 | ALTER STATION | Изменение станции |
| 61 | DROP STATION | Удаление станции |
| 62 | CREATE DEVICE | Создание устройства |
| 63 | ALTER DEVICE | Изменение устройства |
| 64 | DROP DEVICE | Удаление устройства |
| 65 | ALTER \| REBUILD PROCEDURE | Изменение процедуры |
| 66 | CREATE TRIGGER | Создание триггера |
| 68 | GRANT TABLE | Назначение прав доступа к таблице/ хранимой процедуре |
| 70 | EXECUTE PROCEDURE | Выполнение процедуры |

| ID события | Имя события | Пояснение |
|---|---|---|
| 71 | EXECUTE TRIGGER | Выполнение триггера |
| 72 | CREATE SEQUENCE | Создание последовательности |
| 73 | DROP SEQUENCE | Удаление последовательности |
| 74 | CREATE CHARACTER SET | Создание кодировки |
| 75 | DROP CHARACTER SET | Удаление кодировки |
| 76 | CREATE TRANSLATION | Создание правила трансляции |
| 77 | DROP TRANSLATION | Удаление правила трансляции |
| 81 | CREATE ALIAS | Создание алиаса |
| 82 | DROP ALIAS | Удаление алиаса |
| 86 | TRUNCATE TABLE | Усечение таблицы |
| 87 | ALTER SEQUENCE | Модификация последовательности |

# События, связанные с подсистемой доступа

События, связанные с подсистемой доступа, представлены в таблице 10.

Таблица 10. События, связанные с подсистемой доступа

| ID события | Имя события | Пояснение |
|---|---|---|
| 21 | CREATE USER | Создание пользователя |
| 22 | DROP USER | Удаление пользователя |
| 23 | ALTER USER | Изменение привилегий пользователя |
| 24 | ALTER PASSWORD | Изменение пароля пользователя |
| 25 | CREATE ROLE | Создание роли |
| 26 | DROP ROLE | Удаление роли |
| 31 | GRANT ROLE | Назначение роли |
| 32 | REVOKE ROLE | Отмена назначенной роли |
| 27 | CREATE GROUP | Создание группы |
| 28 | ALTER GROUP | Изменение группы |
| 29 | CREATE LEVEL | Создание уровня |
| 30 | ALTER LEVEL | Изменение уровня |
| 33 | GRANT ACCESS | Разрешение доступа к группе |
| 34 | REVOKE ACCESS | Отмена доступа к группе |

# События, связанные с таблицами

События, связанные с конкретными таблицами, представлены в таблице 11.

Таблица 11. События, связанные с таблицами

| ID события | Имя события | Пояснение |
|---|---|---|
| 13 | INSERT | Добавление строки в таблицу |
| 14 | UPDATE | Изменение строки таблицы |

| ID события | Имя события | Пояснение |
|---|---|---|
| 12 | SELECT | Выборка строк из таблицы |
| 15 | DELETE | Удаление строк таблицы |
| 35 | INSERT IN PROCEDURE | Добавление строки в таблицу хранимой процедурой |
| 37 | UPDATE IN PROCEDURE | Изменение строки таблицы хранимой процедурой |
| 38 | SELECT IN PROCEDURE | Выборка строк таблицы хранимой процедурой |
| 36 | DELETE IN PROCEDURE | Удаление строк таблицы хранимой процедурой |
| 40 | UPDATE IN REFERENCES | Изменение строки в таблице при каскадном изменении |
| 39 | DELETE IN REFERENCES | Удаление строк из таблицы при каскадном удалении |
| 42 | DROP INDEX | Удаление индекса таблицы |
| 41 | CREATE INDEX | Создание индекса таблицы |
| 44 | REBUILD TABLE | Восстановление таблицы |
| 43 | PRESS TABLE | Сжатие таблицы |
| 45 | ALTER FILE | Изменение файла таблицы |
| 48 | LOCK TABLE | Блокировка таблицы |
| 49 | UNLOCK TABLE | Снятие блокировки с таблицы |
| 46 | GRANT TABLE | Передача привилегий на таблицу |
| 47 | REVOKE TABLE | Отмена привилегий на таблицу |
| 56 | ADD COLUMN | Добавление столбца |
| 57 | ALTER TABLE | Изменение структуры таблицы |
| 88 | MERGE | Вставка/изменение данных таблицы |

# События, связанные с пользователями

События, связанные с пользователями, представлены в таблице 12.

Таблица 12. События, связанные с пользователями

| ID события | Имя события | Пояснение |
|---|---|---|
| 50 | COMMIT | Фиксация транзакции |
| 51 | ROLLBACK | Откат транзакции |
| 1 | CONNECT | Подключение пользователя к БД |
| 2 | DISCONNECT | Отсоединение пользователя от БД |
| 54 | OPEN CURSOR | Открытие подчиненного канала |
| 55 | CLOSE CURSOR | Закрытие подчиненного курсора |
| 53 | USER MESSAGE | Создание пользовательского сообщения |

### Примечание 1

Следующие события протоколируются и в случае успешного и в случае неуспешного выполнения (если разрешено протоколирование этих событий):

- `AUDIT START;`
- `AUDIT STOP;`
- `AUDIT MESSAGE <текст сообщения>;`
- `SET DATABASE NAMES <имя кодировки>;`
- `ALTER DATABASE SET RECORD SIZE LIMIT <длина>;`
- `SET DATABASE DEFAULT CHARACTER SET <имя кодировки>;`

Это означает, что если, например, выставлен режим:
`AUDIT ENABLE SET DATABASE NAMES WHEN SUCCESS;`
протоколироваться будут и успешные, и неуспешные события `SET DATABASE NAMES`.

### Примечание 2

Чтобы протоколировать вход пользователя в систему необходимо использовать одну из команд:

- `AUDIT ENABLE CONNECT [WHEN SUCCESS | WHEN NOT SUCCESS];`
- `AUDIT ENABLE ACCESS DENIED [WHEN SUCCESS | WHEN NOT SUCCESS];`

Данные команды не зависят друг от друга.
Пример.

```
AUDIT ENABLE CONNECT;
AUDIT DISABLE ACCESS DENIED;
```

В результате работы данной последовательности команд будет выставлен режим:
`AUDIT ENABLE CONNECT;`
Событие `ACCESS DENIED` означает неуспешное подключение пользователя к БД, то есть подключение с кодом ошибки.

### Примечание 3

1. `AUDIT ENABLE;`
   включает протоколирование ВСЕХ НЕУСПЕШНЫХ событий.
   `AUDIT DISABLE;`
   запрещает протоколирование ВСЕХ событий.
   Команды:
   `AUDIT ENABLE LINTER ERROR;`
   `AUDIT ENABLE <имя события> WHEN SUCCESS;`
   включают протоколирование и успешных, и неуспешных событий.
   Пример.

```
username SYSTEM/MANAGER8
AUDIT START;
AUDIT ENABLE LINTER ERROR;
AUDIT ENABLE CREATE TABLE WHEN SUCCESS;
! регистрируется
CREATE TABLE T1( I INT);
! регистрируется - ошибка 1503
```

```
CREATE TABLE T1( I INT);
AUDIT STOP;
```

2. Для включения протоколирования неуспешных событий необходимо выполнить команды:
   ```
   AUDIT ENABLE LINTER ERROR;
   AUDIT ENABLE <имя события>;
   ```
   или
   ```
   AUDIT ENABLE LINTER ERROR;
   AUDIT ENABLE <имя события> WHEN NOT SUCCESS;
   ```

### Примечание 4

Установки для конкретного объекта БД и локальные установки срабатывают ТОЛЬКО для событий типа TABLE EVENT, и событий EXECUTE TRIGGER и EXECUTE PROCEDURE. Для всех событий типа RESOURCE EVENT, если не оговорено отдельно, ключи ON <идентификатор> и FOR <имя пользователя> будут либо игнорироваться и включать протоколирование всех событий данного типа, либо обрабатываться следующим образом:

- команда
  AUDIT ENABLE <имя события> FOR <имя пользователя> [BY {SESSION | STATEMENT | ACCESS}] [WHEN NOT SUCCESS | WHEN SUCCESS];
  выдает код завершения 99;
- команда
  AUDIT ENABLE <имя события> ON <идентификатор> [FOR <имя пользователя>] [BY {SESSION | STATEMENT | ACCESS}] [WHEN NOT SUCCESS | WHEN SUCCESS];
  выдает код завершения 2050;
- для события TRUNCATE TABLE и на ON <идентификатор>, и на FOR <имя пользователя> выдает код завершения 99;
- для всех остальных событий – игнорируется ключ «ON <идентификатор>».

# Механизм надежного восстановления

Механизм надежного восстановления обеспечивается СУБД ЛИНТЕР. Его основой является ведение системного журнала, где отображаются изменения, которые производятся с БД всеми её пользователями.

Действия, связанные с изменениями в системе защиты, также отображаются в системном журнале (создание/удаление нового пользователя/группы и т.д.).

Если пользователь получил уведомление о том, что его изменения перенесены в БД, то сбой оборудования не может привести к нарушению системы защиты.

# Преобразование данных

Преобразованию в БД ЛИНТЕР подлежат данные любой таблицы, содержащиеся в:

- файлах данных;
- BLOB-файлах.

Целью преобразования является затруднение доступа к данным БД средствами, отличными от средств СУБД ЛИНТЕР, в момент, когда ядро СУБД не работает (не загружено).

При загруженном ядре СУБД файлы таблиц монопольно блокируются и доступны только ядру СУБД ЛИНТЕР.

Преобразование данных представляет собой операцию «исключающее ИЛИ» с константой для всех машинных слов, содержащихся в данных.

В файлах индексов преобразование не требуется, так как сжатие, которому подвергаются ключи таблицы, настолько серьезно, что может потребовать месяцы упорного труда для получения реальной информации.

Кроме того, страницы всех файлов таблицы снабжены проверочной информацией, не позволяющей незаметно изменить данные.

Пароли пользователей, прежде чем записываться в БД, подвергаются необратимому преобразованию. При этом используется алгоритм MAC (Message Authentification Code).

# Очистка оперативной и внешней памяти

Перераспределение внешней памяти возможно в двух случаях:

1) при удалении/усечении файлов таблицы и системных журналов;
2) при завершении работы СУБД (усекаются ее рабочие файлы).

В обоих случаях освободившееся дисковое пространство очищается с помощью трехкратной записи маскирующей информации.

Первый раз очистка выполняется путем записи только 0-х битов, затем чередованием 0-х и 1-х битов, в третий раз запись осуществляется только 1-ми битами. Так как файлы системного журнала могут использоваться повторно, то их очистка выполняется путем записи новой информации поверх ранее записанной. Информация в СУБД записывается сначала в системный журнал (для обеспечения надежности), а потом уже в файлы таблицы. Дублирование информации необходимо для обеспечения надежности функционирования СУБД. При удалении таблицы происходит перераспределение внешней памяти, занимаемой файлами таблицы, однако перераспределения внешней памяти, занимаемой системным журналом, при этом не происходит. Поэтому информация в системном журнале сохраняется и после удаления файлов таблицы. Ее очистка произойдет позднее, при перераспределении или переиспользовании памяти журнала. В последнем случае очистка осуществляется записью новой информации поверх старой.

Во время работы СУБД ЛИНТЕР отслеживает занятое ею место в оперативной памяти.

При завершении своей работы СУБД ЛИНТЕР обнуляет весь объем оперативной памяти, который она занимала.

В процессе обработки информация может проходить следующие пути:

1) от приложения через механизмы межпроцессного обмена ОС к СУБД;
2) от удаленного приложения по сети через ОС к СУБД;
3) от СУБД через ОС к внешней памяти.

В каждом случае ОС может использовать оперативную память для выполнения этой функциональности. СУБД не очищает оперативную память, использующуюся для этих целей – эта обязанность должна возлагаться на ОС.

СУБД может быть настроена на использование математического преобразования информации, хранимой на диске и передаваемой по сети. Это может быть использовано для затруднения доступа к остаточной информации ОЗУ, контролируемой ОС.

Очистка памяти (внешней и оперативной) выполняется, если для БД действует опция `CERTIFIED` (см. документ «Создание и конфигурирование базы данных», пункт «Создание БД»).

Очистка оперативной памяти может быть принудительно отключена командой `ALTER DATABASE SET MEMORY CLEAR OFF;` и включена обратно командой `ALTER DATABASE SET MEMORY CLEAR ON;` (см. документ «Справочник по SQL», пункт «Включение/отключение режима очистки оперативной памяти»).

В состав дистрибутива входит утилита `erasefile` для очистки указанного файла и скрипт `erasefile.sh` для выполнения очистки и удаления файлов из

заданного каталога. Утилита работает по алгоритму записи маскирующей информации, описанному выше.

# Изоляция модулей

При рассмотрении механизмов работы КСЗ СУБД ЛИНТЕР необходимо отдельно исследовать работу механизма параллельной обработки запросов различных пользователей. Контроль над изолированностью запросов отдельных пользователей производит **диспетчер доступа** СУБД ЛИНТЕР. Он получает управление:

- при установке логических связей (каналов) с СУБД различными пользователями и при их разрыве;
- при подаче запроса пользователем;
- при выдаче ответа пользователю;
- в процессе параллельной обработки запроса;
- при окончании кванта текущего запроса.

В момент установления логической связи с СУБД и при условии выполнения всех процедур проверки доступа диспетчер доступа выделяет под обработку запросов от данного пользователя управляющую структуру, располагающуюся в адресном пространстве СУБД ЛИНТЕР. В случае если с СУБД установлено несколько логических соединений, для каждого из них будет выделена своя собственная управляющая структура. Управляющая структура канала полностью описывает пользователя, установившего соединение, и содержит всю информацию о текущем состоянии выполнения запросов данного пользователя.

СУБД ЛИНТЕР в процессе обработки запросов оперирует не данными пользователя, а только управляющей информацией, характеризующей их. Эта информация представляет собой, в основном, массивы ссылок на файлы данных. Вся управляющая информация хранится в управляющей структуре соответствующего канала.

При параллельном исполнении запросов диспетчер доступа просто выбирает очередную структуру управления каналом в соответствии с приоритетами исполнения запросов. Дальнейшая работа СУБД ЛИНТЕР, вплоть до передачи управления диспетчеру доступа, ведется в контексте параметров и информации данной структуры.

Все модули СУБД ЛИНТЕР (кроме диспетчера доступа) получают на входе и выдают на выходе управляющую структуру, соответствующую каналу, для которого эти модули будут осуществлять свои функции. В процессе выполнения запроса диспетчер доступа контролирует невозможность использования или повреждения структур, относящихся к другим каналам. Любая подобная попытка считается ошибочной ситуацией, и попытка доступа к БД соответствующего пользователя будет завершена с ошибкой, указывающей на нарушение изоляции.

При корректном закрытии или аварийном разрыве логической связи диспетчер доступа производит процедуру освобождения канала. В этом случае уничтожается вся управляющая информация, касавшаяся данного канала, освобождаются все использованные каналом ресурсы СУБД, и уничтожается структура управления каналом.

# SQL-запросы для работы с системой защиты данных

## Именование объектов БД

Имя объекта БД (<имя>) в СУБД ЛИНТЕР содержит не более 66 символов.

Этими символами могут быть цифры, латинские буквы, знаки подчеркивания (_) и доллара ($). Имя должно начинаться с буквы.

Полное имя таблицы включает также и имя схемы таблицы: [<имя схемы>.]<имя таблицы>.

## Предоставление привилегий

Предоставление привилегии выполняется оператором GRANT, схема которого приведена на рисунке 1.

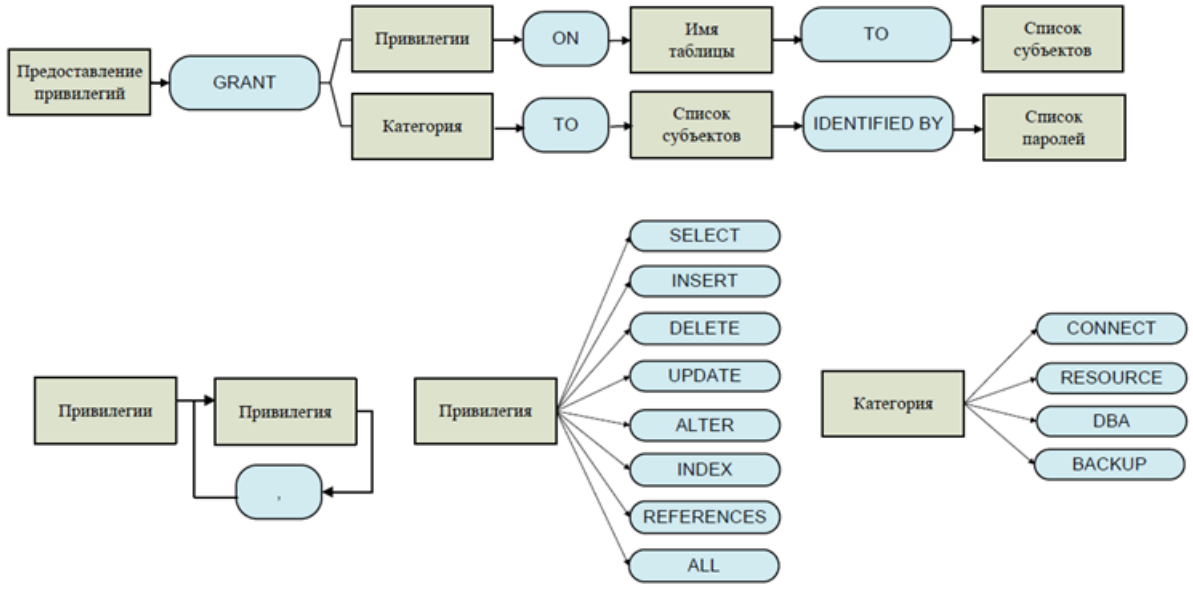



Рисунок 1. Схема оператора GRANT

Общие правила:

- ALL равнозначно списку, включающему все привилегии для таблицы (SELECT+INSERT+DELETE+UPDATE+ALTER+INDEX+REFERENCES);
- если в команде предоставления привилегий на указанную таблицу вместо списка пользователей указано PUBLIC, то всем пользователям (существующим и будущим) будут предоставлены указанные привилегии;
- вторая форма оператора GRANT предназначена как для создания нового пользователя с определением его категории, так и для изменений категории существующего пользователя;

- списки субъектов и их паролей могут различаться по числу элементов. При этом список субъектов должен быть не короче списка паролей. Соответствие «пользователь-пароль» производится слева направо. Пользователи, оставшиеся без соответствия, получат «пустой» пароль: `GRANT CONNECT TO U;` равнозначно `GRANT CONNECT TO U IDENTIFIED BY '';`
- строковый пароль может содержать до 18 символов;
- привилегии могут передаваться только владельцем указанной таблицы/представления;
- вводить нового пользователя (с определением его категории) может только администратор (категория `DBA`).

# Отмена привилегий

Отмена привилегии выполняется оператором `REVOKE`, схема которого приведена на рисунке 2.

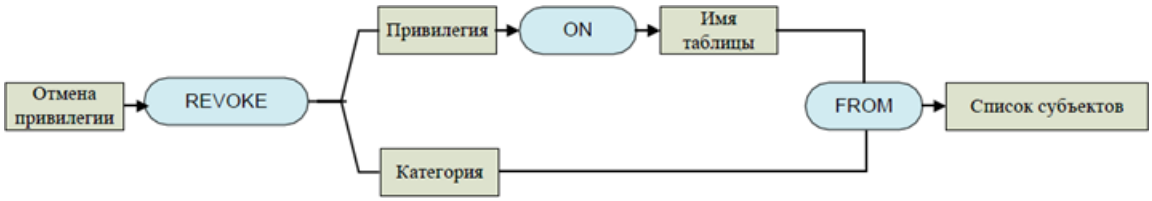



Рисунок 2. Схема оператора REVOKE

Общие правила:

- синтаксические правила те же, что и для оператора `GRANT`;
- привилегии может отменять только владелец таблицы;
- отнять (понизить) категорию пользователя может только администратор БД (категория `DBA`);
- при этом считается, что категория `DBA` включает в себя привилегии категории `RESOURCE` и категория `RESOURCE` включает в себя привилегии категории `CONNECT`, то есть категории упорядочены. Когда отнимается какая-либо категория, это означает снижение категории на один уровень.

| Исходная привилегия | Отменяемая привилегия | Результирующая привилегия |
|---|---|---|
| DBA | DBA | RESOURCE |
| DBA | RESOURCE | CONNECT |
| DBA | CONNECT | запрет подключения к БД |
| RESOURCE | RESOURCE | CONNECT |
| RESOURCE | CONNECT | запрет подключения к БД |
| CONNECT | CONNECT | запрет подключения к БД |

# Создание/удаление пользователя

Схемы создания/удаления пользователя представлены на рисунках 3, 4.

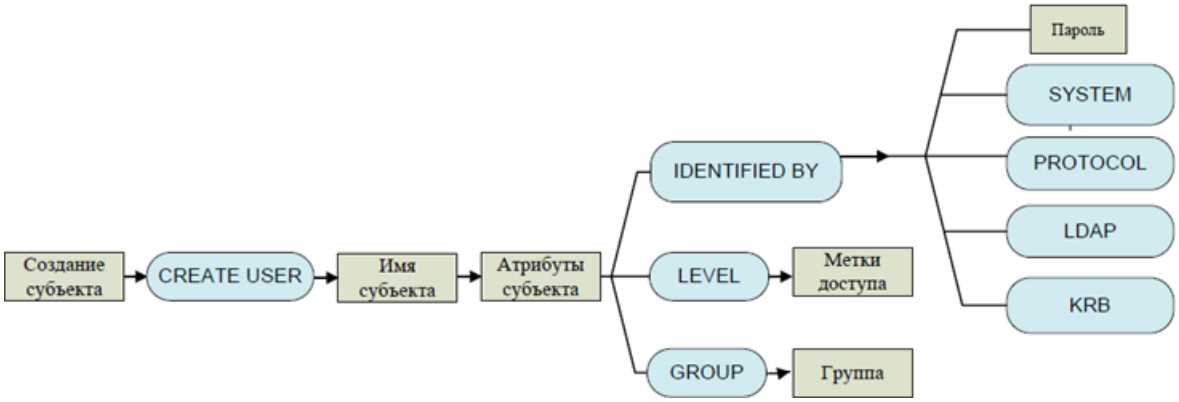



Рисунок 3. Схема создания пользователя




Рисунок 4. Схема удаления пользователя

Общие правила:

- для создания/удаления пользователя необходимо иметь уровень прав `DBA`;
- по умолчанию пользователь создается без пароля. Уровнем прав ему назначается `CONNECT`;
- режим удаления `CASCADE` используется для удаления пользователя, всех его объектов и всех имеющихся в БД ссылок на его объекты;
- запрещено не каскадное удаление пользователя, являющегося владельцем каких-либо объектов БД;
- вместе с пользователем удаляются все его привилегии и назначения ролей.

# Изменение пароля пользователя

Схема изменения пароля пользователя приведена на рисунке 5.

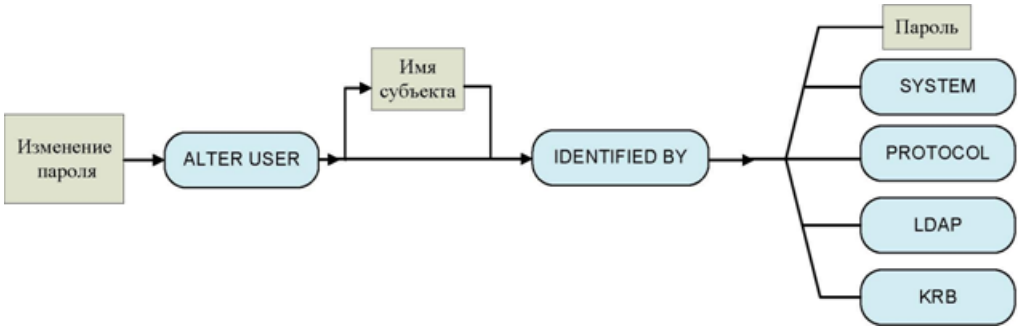



Рисунок 5. Схема изменения пароля пользователя

Общие правила:

- изменить пароль могут администратор (имеющий права `DBA`) и пользователь (сам себе);
- если имя пользователя не задано, то подразумевается пользователь, подавший эту команду.

**Примечание**
Администратор не может незаметно выполнить изменение пароля, так как не имеет возможности вернуть обратно старый пароль. Если он (администратор) изменит пароль пользователя и получит доступ к конфиденциальной информации с именем пользователя и его новым паролем, это не останется незамеченным. Пользователь не будет знать измененного пароля (старый восстановить администратор не сможет), не сможет получить доступ к БД и забьет тревогу.

# Создание/удаление роли

Схемы создания/удаления роли приведены на рисунке 6.




Рисунок 6. Схемы создания/удаления роли

Общие правила:

- для создания роли необходимо иметь уровень прав `RESOURCE`;
- удалить роль может только ее создатель;
- вместе с ролью удаляются записи обо всех назначенных ей привилегиях и обо всех ее назначениях пользователям и другим ролям.

# Назначение/отмена назначения роли

Схемы назначения/отмены назначения роли приведены на рисунке 7.

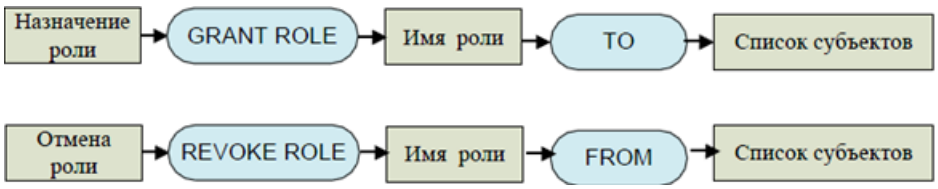



Рисунок 7. Схемы назначения/отмены назначения роли

Общие правила:

- назначить роль (отменить назначение роли) может только ее владелец.

# Приложение

## Синтаксис команд для работы с комплексом средств защиты данных

### Уровень доступа

Создание:

CREATE LEVEL <имя уровня> = <номер уровня>;

Изменение имени уровня:

ALTER LEVEL <имя уровня> SET <новое имя уровня>;

### Группы доступа

Создание:

CREATE GROUP <имя группы>[=<числовой идентификатор группы>];

Изменение имени группы:

ALTER GROUP <имя группы> SET <новое имя группы>;

Уровни доверия между группами:

GRANT ACCESS ON <группа-доверитель> TO {<группа-получатель> | ALL};

REVOKE ACCESS ON <группа-доверитель> FROM {<группа-получатель> | ALL};

### Модификация пользователя

Создание пользователя с категорией CONNECT по умолчанию:

CREATE USER <имя пользователя> IDENTIFIED BY <пароль>;

Назначение пользователю категории доступа:

GRANT <категория> TO <имя пользователя> [IDENTIFIED BY <пароль>];

Удаление пользователя:

DROP USER <имя пользователя> [CASCADE];

Изменение категории пользователя:

GRANT <категория> TO <имя пользователя> [IDENTIFIED BY <пароль>];

Изменение пароля пользователя:

ALTER USER <имя пользователя> [IDENTIFIED BY <новый пароль>];

Назначение группы доступа пользователю:

CREATE USER <имя пользователя> IDENTIFIED BY <пароль> GROUP <имя группы>;

ALTER USER <имя пользователя> GROUP <имя группы>;

Назначение уровня доступа пользователю:

CREATE USER <имя пользователя> IDENTIFIED BY <пароль>

LEVEL(<RAL>,<WAL>);

ALTER USER <имя пользователя> LEVEL(<RAL>,<WAL>);

Назначение привилегий пользователю:

GRANT <привилегия> ON <имя объекта> TO <имя пользователя>;

REVOKE <привилегия> ON <имя объекта> FROM <имя пользователя>;

### Роли

Создание:

CREATE ROLE <имя роли>;

Удаление:

DROP ROLE <имя роли>;

Назначение/отмена привилегии на объект:

GRANT <привилегия> ON <имя объекта> TO <имя роли>;

REVOKE <привилегия> ON <имя объекта> FROM <имя роли>;

Назначение/отмена роли пользователю:

GRANT ROLE <имя роли> TO <имя пользователя>;

REVOKE ROLE <имя роли> FROM <имя пользователя>;

### Уровни доступа

Определение уровня доступа для таблиц:

CREATE TABLE [<имя схемы>.]<имя таблицы>

(<имя столбца> <тип столбца>, […]) LEVEL(<RAL>,<WAL>);

ALTER TABLE <имя таблицы> SET LEVEL(<RAL>,<WAL>);

Определение уровня доступа для столбца:

CREATE TABLE [<имя схемы>.] <имя таблицы>

(<имя столбца> <тип столбца> LEVEL(<RAL>,<WAL>) [,…]);

ALTER TABLE [<имя схемы>.] <имя таблицы>

SET COLUMN <имя столбца> LEVEL(<RAL>,<WAL>);

Использование групп и уровней в SQL-выражении:

INSERT INTO [<имя схемы>.] <имя таблицы>[AS <псевдоним>] [#[<группа>]#[<RAL>]#[<WAL>]]

(<имя столбца>[#[<группа>]#[<RAL>]#[<WAL>]]) VALUES (<значение>);

UPDATE [<имя схемы>.] <имя таблицы>[AS <псевдоним>] [#[<группа>]#[<RAL>]#[<WAL>]]

SET <имя столбца>[#[<группа>]#[<RAL>]#[<WAL>]]=<значение> [,…];

Чтение меток доступа:

SELECT SECURITY({*|<имя столбца>},{'R' |'W' |'G'})

FROM <имя таблицы>;

### Сетевая станция

Создание:

CREATE STATION <имя станции>

PROTOCOL <сетевой протокол>

{ADDRESS <адрес станции>}

[LEVEL (<RAL>,<WAL>)]

[<рабочее время>];

<имя станции>::=<идентификатор>

<сетевой протокол>::=<символьный литерал>

<адрес станции>::=<символьный литерал>

<RAL>::=<идентификатор>

<WAL>::=<идентификатор>

<рабочее время> ::= {ENABLE |DISABLE} LOGIN

{ALWAYS |<график по времени>|<график по дням>}

<график по времени>::= FROM <время начала работы>

TO <время окончания работы> [FOR <дни работы>]

<график по дням>::= {SINCE <дата начала>}|{UNTIL <дата окончания>}

<время начала работы>::='HH:MM'

<время окончания работы>::='HH:MM'

<дни работы>::=<день недели> {[,<день недели>] …}

<день недели>::='MON'|'TUE'|'WED'|'THU'|'FRI'|'SAT'|'SUN'

<дата начала> ::='DD.MM.YYYY'

<дата окончания> ::='DD.MM.YYYY'

Изменение параметров:

ALTER STATION <имя станции>[SET <новое имя станции>]

[LEVEL(<RAL>,<WAL>)] [<рабочее время>];

Удаление:

DROP STATION <имя станции>;

Регулирование доступа:

GRANT ACCESS ON UNLISTED STATION TO {<имя группы> | ALL};

REVOKE ACCESS ON UNLISTED STATION FROM {<имя группы> | ALL};

GRANT ACCESS ON STATION <имя станции> TO {<имя группы>|ALL};

REVOKE ACCESS ON STATION <имя станции>

FROM {<имя группы>|ALL};

### Устройства

Создание:

CREATE DEVICE <имя устройства> DIRECTORY <каталог>

[COMMENT <комментарий>]

[LEVEL (<RAL-устройства>,<WAL-устройства>)];

Изменение параметров:

ALTER DEVICE <имя устройства> [DIRECTORY <каталог>]

[LEVEL (<RAL-устройства>,<WAL-устройства>)];

Удаление:

DROP DEVICE <имя устройства>;

Регулирование доступа:

GRANT ACCESS ON UNLISTED DEVICE TO {<имя группы> | ALL};

REVOKE ACCESS ON UNLISTED DEVICE FROM {<имя группы> | ALL};

GRANT ACCESS ON DEVICE <имя устройства> TO {<имя группы> | ALL};

REVOKE ACCESS ON DEVICE <имя устройства>

FROM {<имя группы> | ALL};

### Мониторинг КСЗ

Запуск протоколирования:

AUDIT START;

Останов протоколирования:

AUDIT STOP;

Управление протоколированием:

AUDIT {ENABLE|DISABLE|CLEAR} [<имя события> [ON <объект аудита>]]

[FOR <имя пользователя>] [BY {SESSION|STATEMENT|ACCESS}]

[WHEN [NOT] SUCCESS];

Параметры протоколирования:

AUDIT {SET |CANCEL} <параметры протоколирования>;

<параметры протоколирования>::= RECORDS LIMIT <количество записей> | DAYS LIMIT <срок хранения>;

Протоколирование пользовательского сообщения:

AUDIT MESSAGE <сообщение>;

Начать комментирование протоколируемых событий:

AUDIT SET MESSAGE <комментарий>;

Отменить комментирование протоколируемых событий:

AUDIT CANCEL MESSAGE;

# Указатель команд SQL-запросов


## A

ALTER DEVICE, 28
ALTER GROUP, 13
ALTER LEVEL, 16
ALTER STATION, 26
AUDIT CANCEL MESSAGE, 45
AUDIT CLEAR, 39
AUDIT DISABLE, 39
AUDIT ENABLE, 39
AUDIT MESSAGE, 45
AUDIT SET, 39
AUDIT SET MESSAGE, 45
AUDIT START, 39
AUDIT STOP, 7

## C

CREATE DEVICE, 27
CREATE GROUP, 12
CREATE LEVEL, 16
CREATE ROLE, 11
CREATE STATION, 25
CREATE USER, 10

## D

DROP DEVICE, 28
DROP ROLE, 11
DROP STATION, 26

## G

GRANT, 11
GRANT ACCESS ON, 13
GRANT ACCESS ON DEVICE, 28
GRANT ACCESS ON STATION, 26
GRANT ACCESS ON UNLISTED DEVICE, 29
GRANT ACCESS ON UNLISTED STATION, 26
GRANT ROLE, 11
GROUP, 13

## L

LEVEL, 16

## R

REVOKE, 11
REVOKE ACCESS ON, 13
REVOKE ACCESS ON DEVICE, 28
REVOKE ACCESS ON STATION, 26
REVOKE ACCESS ON UNLISTED DEVICE, 29
REVOKE ACCESS ON UNLISTED STATION, 26
REVOKE ROLE, 11