50 1190 0101

Утвержден

РУСБ.10015-01-УД

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	

ОПЕРАЦИОННАЯ СИСТЕМА СПЕЦИАЛЬНОГО НАЗНАЧЕНИЯ «ASTRA LINUX SPECIAL EDITION»

Руководство по КСЗ. Часть 3. Защищенная СУБД

РУСБ.10015-01 97 01-3

Листов 93

2024

Литера $О_1$

АННОТАЦИЯ

Настоящий документ является третьей частью руководства по КСЗ операционной системы специального назначения «Astra Linux Special Edition» РУСБ.10015-01 (далее по тексту — ОС).

В документе приведено описание защищенной СУБД, реализованной на основе СУБД Tantor (в исполнении Basic) и доработанной в соответствии с требованием интеграции с ОС в части защиты информации, в том числе мандатного управления доступом.

Защищенная СУБД обеспечивает реализацию процессов управления файлами базы данных, подключения к базе данных от клиентских приложений и выполняет действия с базой данных от имени клиентов (реализована с использованием СУБД PostgreSQL).

Основные сведения по синтаксису языка запросов SQL, поддерживаемым типам данных, встроенным функциям, установке и настройке сервера СУБД приведены в официальной документации на СУБД Tantor, которая доступна на информационном ресурсе разработчика СУБД Tantor `tantorlabs.ru/docs`.

Документ предназначен для администраторов и разработчиков баз данных.

Описание тестов для СУБД приведено описание тестов КСЗ.

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

1.1. Назначение

Защищенная СУБД из состава ОС реализована на основе СУБД Tantor (в исполнении Basic, версия 15) и доработана в соответствии с требованием интеграции с ОС в части защиты информации, в том числе мандатного управления доступом к информации. СУБД содержит реализацию ДП-модели управления доступом и информационными потоками. Данная ДП-модель описывает все аспекты дискреционного, мандатного и ролевого управления доступом с учетом безопасности информационных потоков.

ВНИМАНИЕ! ДП-модель в СУБД работает только при настроенном в ОС механизме мандатного управления доступом.

Защищенная СУБД обеспечивает реализацию процессов управления файлами базы данных, подключения к базе данных от клиентских приложений, и выполняет действия с базой данных от имени клиентов (реализована с использованием СУБД PostgreSQL).

Для работы СУБД на диске выделяется область для хранения БД, называемая кластером БД. Кластер БД является набором БД, управляемых одним экземпляром сервера СУБД. Настройка работы отдельного экземпляра сервера СУБД также определяется в рамках кластера соответствующими конфигурационными файлами.

Корректная работа с СУБД предполагает использование механизма ЕПП.

1.2. Состав

СУБД состоит из следующих компонентов:

- `postgresql` — сервисная служба, реализующая непосредственно сервер БД;
- `libpq` — клиентская библиотека, предоставляющая доступ к серверу СУБД;
- набор серверных утилит для управления работой сервера и создания кластеров БД;
- набор клиентских утилит для создания и управления БД.

Для единообразного управления кластерами БД, функционирующими под управлением разных версий СУБД, в состав ОС входит набор инструментов администрирования `postgresql-common`, включающий в себя следующие пакеты:

- `postgresql` — метапакет, устанавливающий основную версию СУБД;
- `postgresql-x` — метапакет, устанавливающий конкретную версию СУБД (где `x` — версия СУБД);

- `postgresql-client` — метапакет, устанавливающий основную версию клиентских утилит СУБД;
- `postgresql-client-x` — метапакет, устанавливающий конкретную версию клиентских утилит СУБД (где `x` — версия СУБД);
- `postgresql-doc` — метапакет, устанавливающий основную версию документации СУБД;
- `postgresql-common` — инструменты управления кластерами БД.

ВНИМАНИЕ! Рекомендуется управлять СУБД с помощью инструментов управления кластерами БД (см. раздел 8), предоставляемыми ОС, а не утилитами СУБД.

ВНИМАНИЕ! В состав СУБД входит набор модулей расширения, добавляющих новые типы данных, библиотеки функций, инструменты администрирования и разработки. Модули, предназначенные для администрирования и разработки, включая средства отладки хранимых процедур, исследования страниц данных, изменения конфигурационных файлов и доступа к внешним данным, не должны быть установлены в действующих системах, или доступ к ним непривилигированным пользователям должен быть ограничен.

В состав СУБД входит графическая утилита `pgadmin4`, предназначенная для администрирования БД СУБД, включая мандатное управление доступом к объектам БД, и позволяет:

- просматривать иерархическую структуру БД;
- удаленно редактировать конфигурационные файлы СУБД и загружать их на сервер;
- управлять пользователями и группами СУБД;
- управлять дискреционным и мандатным доступом к объектам;
- выполнять SQL-запросы;
- создавать, изменять и удалять различные объекты БД;
- просматривать и редактировать данные таблиц.

Список дополнительно поставляемых модулей приведен в таблице 1:

Таблица 1 – Дополнительно поставляемые модули

Модуль	Описание
`adminpack`	Функции администрирования
`auth_delay`	Позволяет устанавливать паузы между аутентификациями пользователей
`auto_explain`	Функции для автоматической регистрации `EXPLAIN` запросов
`btree_gin`	Операторный класс GIN
`btree_gist`	Индексный операторный класс GiST
`chkpass`	Тип данных для хранения паролей в защищенном виде
`citext`	Строковый тип данных, нечувствительный к регистру

Продолжение таблицы 1

Модуль	Описание
cube	Тип многомерного куба
dblink	Фукнции для подключения к другим базам данных из текущей сессии
dict_int	Пример реализации расширяемого шаблона словаря для полнотекстового поиска
dict_xsyn	Пример реализации расширяемого шаблона словаря синонимов для полнотекстового поиска
earthdistance	Функции для расчета расстояния между двумя точками земной поверхности
file_fdw	Обертка внешних данных (FOREIGN DATA WRAPPER) для файлов
fuzzystrmatch	Функции лексического сравнения строк
hstore	Тип данных для хранения пар ключ-значение
intagg	Агрегирующие функции для целого (integer) и перечисляемого (enum) типов данных
intarray	Функции и операторы для работы с непустыми массивами целых чисел
isn	Тип данных для хранения значений различных международных стандартов
lo	Функции для управления большими объектами (LARGE OBJECT)
ltree	Тип данных, позволяющий хранить метки объектов в виде иерархической структуры данных наподобие дерева
orafce	Функции для миграции с СУБД Oracle на PostgreSQL
pageinspect	Функции для получения информации о содержании страниц в базах данных
passwordcheck	Позволяет проверять пользовательские пароли, устанавливаемые командами ALTER ROLE и CREATE ROLE на простоту
pg_buffercache	Функции для получения текущей информации о разделяемом кеше
pg_freespacemap	Функции для получения информации о свободном пространстве отношения
pg_prewarm	Функции для загрузки всего отношения в память
pgrowlocks	Функции для получения информации о блокировках отношений
pg_stat_statements	Функции для получения информации о статистике исполнения запросов сервера
pgstattuple	Функции для получения различной информации о статистике записей
pg_trgm	Функции для сравнения текста
pldebugger	Функции для отладки запросов, написанных на pl/psql
plproxy	Язык для удаленного вызова функций на серверах баз данных
postgres_fdw	Внешняя обертка данных FOREIGN DATA WRAPPER для СУБД
seg	Тип данных для представления интервалов точек
smlar	Функции для определения идентичности массивов

Окончание таблицы 1

Модуль	Описание
spi	Примеры функций SPI
sslinfo	Функции для получения информации о SSL сертификатах
tablefunc	Функции, возвращающие таблицы
tcn	Триггерные функции, высылающие асинхронные сообщения слушателям
test_decoding	Пример плагина логического декодирования
test_shm_mq	Пример использования динамически разделяемой памяти для совместного использования серверов
unaccent	Функции полнотекстового поиска, удаляющие все диакритические знаки
uuid-ossp	Функции для генерации уникальных идентификаторов (UUID)
xml2	Функции для работы с XPath и XSLT

Примечания:

1. Расширение smlar содержится в пакете postgresql-x-smlar (где x — версия СУБД).
2. Расширение orafce содержится в пакете postgresql-x-orafce (где x — версия СУБД).
3. Расширение pldebugger содержится в пакете postgresql-x-pldebugger (где x — версия СУБД).
4. Расширение plproxy содержится в пакете postgresql-x-plproxy (где x — версия СУБД).
5. Остальные расширения содержатся в основном пакете.

2. НАСТРОЙКА СЕРВЕРА СУБД

Настройка сервера СУБД осуществляется заданием значений параметров в конфигурационном файле СУБД /etc/postgreslq/<версия>/<имя_кластера>/postgresql.conf. Также в СУБД используются дополнительные конфигурационные файлы /etc/postgresql/<номер_версии>/<имя_кластера>/pg_hba.conf и /etc/postgresql/<номер_версии>/<имя_кластера>/pg_ident.conf, которые контролируют аутентификацию клиента. По умолчанию эти файлы располагаются в каталоге данных кластера БД или в соответствующем кластеру конфигурационном каталоге, например /etc/postgresql/<версия_СУБД>/main. Расположение указанных файлов задается параметрами, которые приведены в таблице 2.

Таблица 2

Параметр	Описание
data_directory	Определяет каталог для хранения данных
config_file	Определяет основной конфигурационный файл сервера (postgresql.conf). Значение этого параметра может быть задано только в командной строке postgres
hba_file	Определяет конфигурационный файл для аутентификации по узлам (pg_hba.conf)
ident_file	Определяет конфигурационный файл для аутентификации по методу ident (pg_ident.conf)
external_pid_file	Определяет имя дополнительного файла с идентификатором процесса, который сервер создает для использования программами администрирования сервера

При использовании относительного пути для задания значений этих параметров путь будет отсчитываться от каталога, в котором запущен postgres.

Примечание. Некоторые способы использования СУБД (например, организация сервера для 1С) могут требовать дополнительной настройки сервера СУБД и ОС.

Примечание. Особенности настройки аутентификации для работы в текущей ОС приведены в разделе 3.

Настройка работы сервера СУБД в условиях мандатного управления доступом выполняется в соответствии с 5.1.

За установку соединений отвечают конфигурационные параметры, приведенные в таблице 3.

Таблица 3

Параметр	Описание
listen_addresses	Определяет TCP/IP-адреса, по которым сервер должен ожидать соединения от клиентских приложений. Значение формируется в виде перечня разделенных запятой имен узлов и/или IP-адресов. Специальный знак «*» соответствует всем доступным IP-адресам. Если список пуст, сервер не слушает ни один IP-интерфейс. В этом случае установка соединения с сервером возможна только с использованием доменных сокетов UNIX. По умолчанию значением параметра является localhost, которое позволяет создавать только локальные loopback-соединения. **ВНИМАНИЕ!** В настоящее время отсутствует поддержка протокола IPv6. Задаваемые в этом параметре адреса должны соответствовать протоколу IPv4
port	Определяет TCP-порт, на котором сервер должен ожидать соединения от клиентских приложений (по умолчанию — 5432). Следует отметить, что для всех IP-адресов, указанных в listen_addresses (string), используется один и тот же порт
max_connections	Определяет максимальное число одновременных соединений с сервером СУБД
superuser_reserved_connections	Определяет число соединений, зарезервированных для подключения суперпользователей СУБД
unix_socket_group	Определяет группу, владеющую доменным сокетом UNIX. Владельцем сокета всегда является пользователь, запускающий сервер. Используется в комбинации с параметром unix_socket_permissions, как дополнительный механизм контроля доступа для соединений в домене UNIX
unix_socket_permissions	Определяет права доступа для доменного сокета UNIX. Для доменного сокета UNIX имеет значение только разрешение на запись, и следовательно нет необходимости устанавливать или удалять разрешение на чтение или выполнение. Данный механизм контроля доступа не зависит от механизма аутентификации клиента

3. ИДЕНТИФИКАЦИЯ И АУТЕНТИФИКАЦИЯ ПОЛЬЗОВАТЕЛЕЙ В СУБД

Идентификация и аутентификация пользователей в СУБД осуществляется с учетом требований ГОСТ Р 5 8833-2020.

В соответствии с ролевой моделью (см. раздел 6) заведение учетных записей администраторов СУБД и администраторов БД (администраторов информационной системы) осуществляется администратором СУБД, а учетных записей пользователей БД — администратором БД. Первичный пароль для администраторов СУБД и администраторов БД устанавливается администратором СУБД, для пользователей БД — администратором БД. После первичной аутентификации имеется возможность смены пароля, заданного при заведении учетной записи.

Аутентификация в СУБД осуществляется путем ввода пароля. Защита пароля при его вводе обеспечивается за счет отображения вводимых символов условными знаками.

При вводе неправильного значения идентификатора учетной записи или пароля в доступе будет отказано. При исчерпании установленного максимального количества неуспешных попыток аутентификации учетная запись будет заблокирована. Разблокировать учетную запись пользователя БД может администратор БД, разблокировать учетную запись администратора СУБД или администратора БД может администратор СУБД. Также учетная запись может быть разблокирована автоматически по истечении заданного временного интервала.

Хранение аутентификационной информации в защищенном формате обеспечивается использованием библиотеки `libgost`. Пароли пользователей СУБД хранятся в каталоге `pg_authid`.

Настройка требований к сложности пароля, а также количество неуспешных попыток аутентификации осуществляется с помощью модулей PAM (см. документ РУСБ.10015-01 97 01-1). Описание настройки блокировки после установленного количества неуспешных попыток аутентификации приведено в 3.3.4.

3.1. Настройки аутентификации

СУБД предлагает несколько различных методов аутентификации клиента. Метод, используемый для аутентификации конкретного клиентского соединения, может быть выбран на основе адреса узла сети клиента, БД и пользователя.

Несмотря на то, что имена пользователей СУБД логически отделены от имен пользователей ОС, в которой запущен сервер, в соответствии с требованиями по защите информации от НСД требуется сопоставление пользователей СУБД пользователям ОС. Таким образом, при настройке аутентификации в СУБД следует использовать только методы аутентификации, в

которых осуществляется подобное сопоставление. Для других пользователей осуществляется доступ только к незащищенной информации.

Для соединения с сервером СУБД через клиентское приложение (например, `pgadmin-4`) необходимо ввести учетные данные (имя и пароль) пользователя СУБД, а также указать настройки сетевого соединения (IP-адрес или доменное имя сервера СУБД, порт подключения и т. п.). После установки соединения с сервером СУБД имя пользователя определяет права на объекты БД.

ВНИМАНИЕ! Для обеспечения корректной работы пользователя с сетевыми службами должны быть явно заданы диапазоны его уровней и категорий конфиденциальности с помощью соответствующих утилит, даже если ему не доступны уровни и категории конфиденциальности выше `0` (см. документ РУСБ.10015-01 97 01-1).

Для корректного выполнения авторизации необходимо пользователю, от имени которого работает СУБД (по умолчанию `postgres`), предоставить права на чтение информации из БД пользователей и сведений о метках безопасности и привилегиях:

```
usermod -a -G shadow postgres
setfacl -d -m u:postgres:r /etc/parsec/macdb
setfacl -R -m u:postgres:r /etc/parsec/macdb
setfacl -m u:postgres:rx /etc/parsec/macdb
setfacl -d -m u:postgres:r /etc/parsec/capdb
setfacl -R -m u:postgres:r /etc/parsec/capdb
setfacl -m u:postgres:rx /etc/parsec/capdb
```

3.2. Использование PAM аутентификации

Для настройки PAM в СУБД необходимо создать локальные учетные записи пользователей и назначить им минимальные и максимальные уровни конфиденциальности и категории конфиденциальности. В конфигурационном файле контроля доступа сервера СУБД `/etc/postgreslq/<версия>/main/pg_hba.conf` для параметра `host` указать требуемый метод аутентификации.

Пример

```
host all all 0.0.0.0/0 pam
```

3.3. Использование сквозной аутентификации в ЕПП

3.3.1. Общие условия

Для обеспечения сквозной аутентификации пользователей ЕПП (см. документ РУСБ.10015-01 95 01-1) в СУБД необходимо в качестве метода аутентификации указать `gss` и провести соответствующую настройку сервера и клиента СУБД.

Для работы СУБД с FreeIPA необходимо выполнение следующих условий:

1) наличие в системах, на которых функционируют сервер и клиенты СУБД, установленного пакета клиентской части FreeIPA `freeipa-client`;

2) разрешение имен должно быть настроено таким образом, чтобы имя системы разрешалось, в первую очередь, как полное имя (например, `postgres.example.ru`);

3) клиентская часть FreeIPA должна быть настроена на используемый FreeIPA домен.

3.3.2. Настройка серверной части FreeIPA

Для обеспечения совместной работы сервера СУБД с FreeIPA необходимо, чтобы сервер СУБД функционировал как служба Kerberos. Выполнение данного условия требует наличия в БД Kerberos принципала для сервера СУБД, имя которого задается в формате:

```
postgres/hostname@realm
```

где `hostname` — полное доменное имя системы, на которой функционирует сервер СУБД;
`realm` — имя домена FreeIPA.

Для обеспечения совместной работы сервера СУБД с FreeIPA выполнить следующие действия:

1) создать в БД FreeIPA с помощью утилиты администрирования FreeIPA принципала, соответствующего устанавливаемому серверу СУБД. Принципал создается с автоматически сгенерированным случайным ключом;

Пример

```
ipa service-add postgres/postgres.example.ru
```

2) создать файл ключа Kerberos для сервера СУБД с помощью утилиты администрирования FreeIPA `ipa service-add`.

Пример

Создание файла ключа Kerberos на контроллере домена:

```
ipa-getkeytab -s domain.example.ru -k /etc/apache2/keytab
-p HTTP/apache2.example.ru
```

Полученный файл должен быть доступен серверу СУБД по пути, указанному в конфигурационном параметре krb_server_keyfile конфигурационного файла /etc/postgreslq/<версия>/main/postgresql.conf (для приведенного примера путь /etc/apache2/keytab). Пользователю, от имени которого работает сервер СУБД (по умолчанию postgres), должны быть предоставлены права на чтение данного файла;

3) назначить владельцем файла krb5.keytab пользователя postgres, выполнив команду:

```
chown postgres /etc/postgresql/<версия>/main/krb5.keytab
```

4) задать в конфигурационном файле сервера СУБД /etc/postgreslq/<версия>/main/postgresql.conf значение для параметра krb_server_keyfile:

```
krb_server_keyfile = '/etc/postgresql/<версия>/main/krb5.keytab'
```

5) указать для внешних соединений в конфигурационном файле контроля доступа сервера СУБД /etc/postgreslq/<версия>/main/pg_hba.conf метод аутентификации gss.

Пример

```
host all all 192.168.32.0/24 gss
```

6) для включения аутентификации необходимо в конфигурационный файл /etc/sssd/sssd.conf в секцию [ifp] добавить следующую строку:

```
[ifp]
allowed_uids = postgres
```

Если строка уже существует и значения для параметра allowed_uids уже заданы, то добавить postgres через запятую:

```
allowed_uids = www-data, postgres
```

7) перезапустить службу postgresql и службу sssd:

```
sudo systemctl restart postgresql sssd
```

3.3.3. Клиент

Общие условия, при которых обеспечивается совместное функционирование клиентов СУБД в FreeIPA, см. 3.3.1. Кроме того, сервер СУБД должен быть также настроен в соответствии с 3.3.2. Для настройки клиента СУБД необходимо:

1) создать в БД FreeIPA учетную запись пользователя, зарегистрированного в СУБД (например, `pgusername`);

2) задать в качестве значения параметра соединения `krbsrvname` имя службы, используемое при создании принципала сервера СУБД (см. 3.3.2).

3.3.4. Блокировка после установленного количества неуспешных попыток аутентификации

Для настройки блокировки после установленного количества неуспешных попыток аутентификации необходимо выполнить следующие действия:

1) выполнить вход в ОС администратором СУБД `pg_astra_cl_admin`:

```
sudo login pg_astra_cl_admin
```

2) в каталоге `/etc/pam.d/` создать конфигурационный файл:

```
sudo touch /etc/pam.d/<имя_файла>
```

3) добавить в созданный файл `/etc/pam.d/<имя_файла>` следующие строки:

```
auth optional pam_echo.so "<имя_файла> pam config"
auth required pam_faillock.so preauth
conf=/etc/postgresql/<версия>/<имя_кластера>/faillock.conf
auth sufficient pam_unix.so
auth [default=die] pam_faillock.so authfail
conf=/etc/postgresql/<версия>/<имя_кластера>/faillock.conf
auth required pam_deny.so
auth required pam_permit.so
account required pam_faillock.so
conf=/etc/postgresql/<версия>/<имя_кластера>/faillock.conf
account required pam_unix.so
```

4) в каталоге `/etc/postgresql/<версия>/<имя_кластера>/` создать конфигурационный файл `faillock.conf`:

```
sudo touch /etc/postgresql/<версия>/<имя_кластера>/faillock.conf
```

5) в файл `/etc/postgresql/<версия>/<имя_кластера>/faillock.conf` добавить следующие строки:

```
dir = /etc/postgresql/<версия>/<имя_кластера>/faillock
silent
deny = 6
unlock_time = 0
even_deny_root
root_unlock_time = 0
```

6) создать каталог /etc/postgresql/<версия>/<имя_кластера>/faillock и назначить ему соответствующие права:

```
sudo mkdir /etc/postgresql/<версия>/<имя_кластера>/faillock
sudo chown postgres:postgres
/etc/postgresql/<версия>/<имя_кластера>/faillock
```

7) в файле /etc/postgresql/<версия>/<имя_кластера>/pg_hba.conf выполнить настройку прохождения обязательной аутентификации с использованием PAM и с указанием на конфигурационный файл /etc/pam.d/<имя_файла> (см. пункт перечисления 2)):

```
host all all 127.0.0.1/32 pam pamservice= <имя_файла>
```

8) перезагрузить кластер:

```
sudo pg_ctlcluster 15 <имя_кластера> reload
```

4. ДИСКРЕЦИОННОЕ УПРАВЛЕНИЕ ДОСТУПОМ В СУБД

СУБД является объектно-реляционной. Сущности (данные) хранятся в отношениях (таблицах), состоящих из строк и столбцов. При этом единицей хранения и доступа к данным является строка, состоящая из полей, идентифицируемых именами столбцов. Кроме таблиц также существуют другие объекты БД (виды, процедуры и т. п.), которые предоставляют доступ к данным, хранящимся в таблицах.

С каждым типом объектов БД ассоциируется определенный набор типов доступа (возможных операций). Каждому объекту явно задается список разрешенных типов доступа для каждого из поименованных субъектов БД (пользователей, групп или ролей), т. е. объектам БД назначается ACL. И в дальнейшем при разборе запроса к БД осуществляется проверка возможности предоставления субъекту запрашиваемого типа доступа к объекту.

В СУБД объектами дискреционного управления доступом могут быть столбцы таблицы, поскольку они однозначно идентифицируются по составному имени таблицы и столбца, т.к. имя столбца внутри таблицы является уникальным.

В то же время отдельная строка таблицы не является однозначно идентифицируемым объектом, поскольку каждая строка таблицы идентифицируется только набором содержимого своих полей, в связи с этим в общем случае дискреционные и любые другие правила разграничения доступа к ней применены быть не могут. Для идентификации строк таблицы разработчику потребуется выбрать ту или иную процедуру, например создание первичного ключа или создание физического уникального идентификатора строки в БД.

Дополнительно для ограничения набора данных, выдаваемых пользователю, можно применять входящую в СУБД систему фильтрации строк (POLICY) под названием ROW LEVEL SECURITY — фильтровать строки, выдаваемые из таблицы указанному пользователю (пользователям) на основании вычисления заданного логического выражения.

4.1. Порядок применения правил дискреционного управления доступом

В рамках дискреционных ПРД определены следующие операции над таблицами и хранящимися в них данными:

- `SELECT` — чтение данных из таблицы;
- `INSERT` — вставка новых данных в таблицу;
- `DELETE` — удаление некоторых/всех данных в таблице;
- `UPDATE` — изменение данных в таблице;
- `REFERENCES` — использование данных таблицы для внешних ключей;
- `TRIGGER` — создание и назначение для таблицы триггеров;
- `TRUNCATE` — очистка таблицы (удаление всех данных).

Для более гибкой работы с данными в СУБД введены следующие объекты, к каждому из которых также существует набор операций:

1) вид — способ организации предварительно подготовленных запросов. Набор операций совпадает с набором операций для таблиц, за исключением создания триггеров и внешних ключей:
- SELECT — чтение данных из вида;
- INSERT — вставка новых данных в вид;
- DELETE — удаление некоторых/всех данных в виде;
- UPDATE — изменение данных в виде;

2) последовательность — способ получения уникальных значений (счетчик). Определены следующие операции:
- SELECT — чтение значения счетчика;
- UPDATE — установка значения счетчика;
- USAGE — выполнение функций манипулирования счетчиком;

3) БД — способ организации области данных, содержащих все остальные объекты СУБД. Определены следующие операции:
- CREATE — создание БД;
- CONNECT — установка соединения с БД;
- TEMPORARY/TEMP — создание временных таблиц в БД;

4) функция — программный код манипулирования данными на сервере. Определена операция EXECUTE — выполнение функции;

5) язык — язык написания функций на сервере. Определена операция USAGE — использование языка для написания функций;

6) схема — способ организации объектов в пределах отдельной БД. Определены следующие операции:
- CREATE — создание объектов в указанной схеме;
- USAGE — использование объектов указанной схемы;

7) табличное пространство — способ организации БД в ФС ОС. Определена операция CREATE — создание объектов в указанном табличном пространстве.

8) бинарный объект — способ хранения больших двоичных объектов (файлов, документов, фотографий и т.п.) в БД. Определены следующие операции:
- SELECT — чтение бинарного объекта;
- UPDATE — изменение бинарного объекта;

9) в БД могут присутствовать дополнительные объекты, для использования которых определена операция USAGE.

Для контроля выполнения всех перечисленных операций дискреционных ПРД используются соответствующие права доступа. Полномочия на предоставление прав доступа к сущностям доступны только администратору БД и не могут быть предоставлены другим пользовате-

лям, но при соответствующих настройках сервера могут быть предоставлены и владельцу сущности.

Кроме рассмотренных (делегируемых) прав доступа, существует ряд прав, которые всегда принадлежат владельцам сущностей и администраторам СУБД. Эти права не могут быть делегированы или отменены средствами СУБД. К таким правам относятся: удаление и модификация сущности и назначение пользователям делегируемых прав доступа к сущностям.

Сразу же после создания сущности только ее владелец и администраторы СУБД могут ее использовать. Для того чтобы с этой сущностью могли работать другие пользователи, владелец сущности или администратор СУБД должен явно предоставить им соответствующие дискреционные права доступа.

Модификация метаданных осуществляется каждый раз при изменении структуры БД, что включает в себя создание, модификацию и удаление объектов БД.

Разграничение доступа к перечисленным операциям на уровне СУБД также реализуется применением дискреционных ПРД. Для этого используется право владения объектом, право на создание объектов. Право владения объектом предоставляет владельцу объекта возможность модифицировать и удалять объект. В общем случае, владельцем является создатель объекта или администратор БД. Право на создание (`CREATE`) применяется к объектам БД, являющимся контейнерами для других объектов, а именно: непосредственно сама БД, схема, табличное пространство.

При выполнении любого запроса пользователя (субъекта БД) к защищаемому ресурсу (объекту БД) выполняется дискреционное управление доступом на основе установленных пользователю прав. Для каждой выполняемой операции производится проверка наличия права у пользователя на выполнение данной конкретной операции.

Дискреционные ПРД применяются после разбора запроса пользователя и построения плана его выполнения.

Дискреционные ПРД к столбцам объекта применяются только при отсутствии явного разрешения на доступ к самой таблице. Таким образом, права доступа к объекту являются доминирующими. При этом, в случае отсутствия явно заданных прав на объект нельзя сказать определенно о предоставлении доступа до тех пор, пока не будут проверены права на столбцы объекта.

В СУБД параметр конфигурации `ac_enable_trusted_owner` позволяет администратору запретить владельцам объектов передавать права на доступ к ним другим пользователям СУБД. В случае установки значения этой переменной конфигурации в `FALSE` распределение прав доступа к объектам БД разрешено только администраторам СУБД.

Параметр конфигурации ac_allow_grant_options позволяет администратору запретить передачу уже имеющихся прав доступа на объект другим ролям. Если ac_allow_grant_options установлен в FALSE, то запрещается использовать команду GRANT с привилегией WITH GRANT OPTION. Если у роли есть привилегия GRANT OPTIONS и ac_allow_grant_options = false, то передача прав доступа другим ролям также запрещается. Изъятие (REVOKE) привилегии GRANT OPTIONS разрешается всегда.

Параметр конфигурации ac_allow_admin_options позволяет администратору запретить передачу прав членства роли другим ролям. Если ac_allow_admin_options установлен в FALSE, то запрещается использовать GRANT с привилегией WITH ADMIN OPTION. Если у роли есть привилегия ADMIN OPTIONS и ac_allow_admin_options = false, то передача прав членства другим ролям также запрещается. Изъятие (REVOKE) привилегии ADMIN OPTIONS разрешается всегда.

Параметр конфигурации ac_enable_truncate позволяет администратору запретить владельцам объектов и любым пользователям, обладающим соответствующим правом TRUNCATE, выполнять удаление всех записей из таблиц. В случае установки значения этой переменной конфигурации в FALSE выполнение команды TRUNCATE запрещено всем пользователям.

4.2. Средства управления дискреционными ПРД к объектам БД СУБД

Для управления дискреционными ПРД к объектам БД СУБД используется графическая утилита pgadmin4.

Для делегирования дискреционных прав доступа к объектам используется команда SQL GRANT, а для отмены — команда REVOKE. Например, если в системе существует пользователь ivanov, то ему может быть предоставлено право на изменение данных в таблице Счета с помощью следующей команды:

```
GRANT UPDATE ON "Счета" TO ivanov
```

Для предоставления прав доступа к объекту сразу всем пользователям системы существует специальное «имя пользователя» PUBLIC, а для предоставления всех прав — специальное «право» ALL. Например, чтобы дать всем пользователям полный доступ к таблице Счета, следует использовать следующую команду:

```
GRANT ALL ON "Счета" TO PUBLIC
```

При необходимости право доступа может быть предоставлено пользователю (но не группе) с возможностью делегирования данного права другим ролям. Для этого используется ключевая фраза WITH GRANT OPTION:

```
GRANT UPDATE ON "Счета" TO ivanov WITH GRANT OPTION
```

Владелец объекта может отменить собственные делегируемые права, например, переведя объект в режим «только для чтения» для себя, так же как и для всех остальных пользователей.

5. МАНДАТНОЕ УПРАВЛЕНИЕ ДОСТУПОМ В СУБД

В основе механизма мандатного управления доступом в СУБД лежит управление доступом к защищаемым ресурсам БД на основе иерархических и неиерархических меток доступа. Это позволяет реализовать многоуровневую защиту с обеспечением разграничения доступа пользователей к защищаемым ресурсам БД и управление потоками информации. В качестве иерархических и неиерархических меток доступа при использовании СУБД в ОС используются метки безопасности ОС.

ВНИМАНИЕ! Мандатное управление доступом в СУБД работает только при настроенном в ОС мандатном управлении доступом.

СУБД не имеет собственного механизма назначения, хранения и модификации меток пользователей и использует для этого механизмы ОС.

Для хранения метки объекта БД введено служебное поле `maclabel`. Объекту БД при его создании задается метка безопасности, равная текущей метке безопасности создавшего его пользователя. В дальнейшем по этой метке производится разграничение доступа к созданному объекту.

Согласно ДП-модели в части реализации мандатного управления доступом дополнительно к классификационной метке вводится понятие сущностей-контейнеров (сущностей, которые могут содержать другие сущности). Для задания способа доступа к сущностям внутри контейнеров используется мандатный признак CCR (Container Clearance Required). В случае когда он установлен, доступ к контейнеру и его содержимому определяется его классификационной меткой, в противном случае доступ к содержимому разрешен без учета уровня конфиденциальности контейнера.

В качестве главного контейнера выбрано табличное пространство `pg_global`, которое создается одно на кластер базы данных. Таким образом, кластер является совокупностью ролей, баз данных и табличных пространств.

ВНИМАНИЕ! ДП-модель накладывает ограничение на классификационную метку сущности: метка сущности не может превышать метку контейнера, в котором она содержится. Таким образом, для назначения классификационных меток данным сначала должны быть последовательно заданы максимальные классификационные метки соответствующих контейнеров: кластера, базы данных, табличного пространства, схемы и таблицы.

В реляционной модели в качестве структуры, обладающей меткой безопасности, выбран кортеж, поскольку именно на этом уровне детализации осуществляются операции чтения/записи информации в СУБД. При этом местом хранения метки безопасности выбран только сам кортеж — только так метка безопасности будет неразрывно связана с данными, содержащимися в кортеже. Кроме этого, метка безопасности также может быть определена для объектов БД, к которым применимы виды доступа на чтение/запись данных, а имен-

но: таблицы и виды. В этом случае метки безопасности объектов располагаются в записи системной таблицы, непосредственно описывающей защищаемый объект.

В качестве множества сущностей (сущностей-объектов и сущностей-контейнеров) с заданной иерархической структурой рассматриваются носящие подобный характер объекты реляционных баз данных, применяемые в СУБД. При этом, поскольку записи базы данных содержат в своем составе уровень конфиденциальности, они рассматриваются в модели в качестве объектов, а содержащие их таблицы, соответственно, — в качестве контейнеров.

Системный каталог (метаданные) рассматривается как самостоятельная БД, реализованная с помощью средств СУБД. При этом, все операции с этой БД осуществляются либо с помощью специальных конструкций языка запросов SQL, либо привилегированным пользователем в специальном режиме. Таким образом, мандатное управление доступом применяется ко всем объектам БД. Метки безопасности системных объектов располагаются в записях таблиц системного каталога, непосредственно описывающих защищаемый объект.

5.1. Порядок применения мандатных правил управления доступом

Мандатное управление доступом может быть определено только для видов доступа на чтение и на запись информации, поэтому в БД все множество операций с данными в защищаемых объектах приводится к этим типам доступа следующим образом:

- `INSERT` — доступ на запись;
- `UPDATE`, `DELETE` — последовательное выполнение доступа на чтение и запись информации;
- `SELECT` — доступ на чтение.

При обращении пользователя к БД определяются его допустимый диапазон меток безопасности и набор дополнительных мандатных атрибутов. Если пользователю не присвоена метка безопасности, то он получает по умолчанию нулевую метку безопасности, соответствующую минимальному уровню доступа. Максимальная метка безопасности определяется по заданной при регистрации пользователя в ОС. Поскольку сервер БД также может иметь метку безопасности, то если метка безопасности пользователя превышает метку безопасности сервера, то пользователю будут разрешены только операции чтения. Текущая метка безопасности пользователя определяется по установленному соединению и может быть установлена только в пределах назначенного ему диапазона мандатных атрибутов при наличии соответствующей привилегии.

Применение мандатных ПРД осуществляется на уровне доступа к объектам БД и на уровне доступа непосредственно к данным (на уровне записей).

Проверка мандатных прав доступа к объектам осуществляется одновременно с проверкой дискреционных прав доступа к ним после разбора и построения плана запроса и непосред-

ственно перед его выполнением, когда определены все необходимые для проверки данные и проверяемые объекты. Таким образом, доступ предоставляется только при одновременном разрешении мандатными и дискреционными ПРД.

Проверка мандатных прав доступа к записям таблиц осуществляется в процессе выполнения запроса при последовательном или индексном сканировании данных.

Все записи, помещаемые в таблицы, для которых установлена защита на уровне записей, наследуют текущую метку безопасности пользователя. Обновляемые записи сохраняют свою метку безопасности при изменении. Доступ к существующим записям и возможность их обновления и удаления определяются установленными мандатными правилами.

Для администратора БД предусмотрены системные привилегии игнорирования мандатного управления доступом, только таким образом можно производить регламентные работы с БД (например, восстановление резервной копии), т. к. это требует установки меток данных, сохраненных ранее.

Для настройки работы сервера в условиях дискреционного и мандатного управления доступом существует ряд конфигурационных параметров, указываемых в конфигурационном файле `postgresql.conf` конкретного кластера данных (таблица 4).

Таблица 4

Параметр	Описание
`ac_ignore_server_maclabel`	Определяет, будет ли сервер СУБД дополнительно использовать свою метку безопасности (метку безопасности процесса, например назначаемую параметром `PDPLabel`, см. РУСБ.10015-01 97 01-1) при определении прав пользователя на добавление, удаление и изменение данных или нет. Если этот параметр установлен в `FALSE`, то метка безопасности сервера используется для блокировки добавления в БД информации с меткой безопасности, превышающей метку безопасности сервера. Если этот параметр установлен в `TRUE`, то метка безопасности сервера не учитывается
`ac_ignore_socket_maclabel`	Определяет, будет ли сервер СУБД использовать метку безопасности входящего соединения. Если этот параметр установлен в `FALSE`, то метка безопасности входящего соединения будет учитываться при определении максимальной доступной метки безопасности сессии, и после подключения будет доступна только информация с меткой безопасности, не превышающей метку безопасности входящего соединения. При установке этого параметра в `TRUE` метки безопасности сеанса будут определяться максимальной меткой безопасности пользователя, полученной из ОС

Продолжение таблицы 4

Параметр	Описание
ac_ignore_maclabel	Обеспечивает возможность работы с базой при отключенном мандатном управлении доступом. Если данный параметр установлен в значение TRUE, то мандатные атрибуты пользователя не запрашиваются, метка безопасности сессии устанавливается нулевой вне зависимости от значения ac_ignore_socket_maclabel. Метка безопасности сервера принимается нулевой вне зависимости от значения параметра ac_ignore_server_maclabel
ac_enable_trusted_owner	Определяет, могут ли владельцы объектов назначать права на доступ к ним другим пользователям. Если этот параметр установлен в значение FALSE, то право назначать права на доступ к любым объектам БД имеют только суперпользователи. Это предотвращает неконтролируемое распространение прав на доступ к информации. Если этот параметр установлен в TRUE, то, кроме суперпользователей, каждый владелец объекта может назначать права на доступ пользователей к «своему» объекту
ac_enable_grant_options	Определяет, могут ли роли передавать права на доступ с параметром WITH GRANT OPTIONS другим ролям. Если ac_allow_grant_options установлен в FALSE, то запрещается использовать команду GRANT с привилегией WITH GRANT OPTION. Если у роли есть привилегия GRANT OPTIONS и ac_allow_grant_options = false, то передача прав доступа другим ролям также запрещается. Изъятие (REVOKE) привилегии GRANT OPTIONS разрешается всегда
ac_enable_admin_options	Определяет, могут ли роли передавать право членства роли другим ролям. Если ac_allow_admin_options установлен в FALSE, то запрещается использовать GRANT с привилегией WITH ADMIN OPTION. Если у роли есть привилегия ADMIN OPTIONS и ac_allow_admin_options = false, то передача прав членства другим ролям также запрещается. Изъятие (REVOKE) привилегии ADMIN OPTION разрешается всегда
ac_enable_truncate	Блокирует (FALSE) или разблокирует (TRUE) возможность выполнения команды TRUNCATE
ac_enable_sequence_ccr	При установке этого параметра в TRUE разрешается использование признака CCR для последовательностей аналогично таблицам и видам, в противном случае признак CCR для последовательностей считается всегда установленным
ac_enable_dblink_mac	Если параметр конфигурации установлен в TRUE, разрешается использование dblink и внешних таблиц FOREIGN TABLE в условиях мандатного управления доступом
ac_auto_adjust_macs	Если параметр конфигурации установлен в TRUE, разрешается автоматическая установка меток контейнеров. Применяется при восстановлении резервных копий, созданных в предыдущих версиях СУБД

Окончание таблицы 4

Параметр	Описание
ac_enable_copy_to_file	Блокирует (FALSE) или разблокирует (TRUE) возможность выполнения команды COPY с выводом результатов в файл, доступный серверу СУБД
ac_caps_ttl	Время жизни в секундах информации о привилегиях пользователя подсистемы безопасности PARSEC (определяет время жизни кэшированной информации о PARSEC-привилегиях пользователя; уменьшение значения приводит к увеличению числа обращений сервера СУБД к подсистеме безопасности PARSEC и как следствие — к снижению производительности сервера СУБД)
ac_debug_print	Если установлен в TRUE, добавляет в журнал сервера отладочную информацию о работе механизмов защиты

Для более гибкой настройки сервера СУБД расширен синтаксис конфигурационного файла pg_hba.conf параметром ignore_socket_maclabel. Данный параметр может быть указан после метода аутентификации.

Параметр ignore_socket_maclabel=1 позволяет пользователям подключаться к базам данных с игнорированием метки безопасности сокета соединения. Если этот параметр не указывается в конфигурационном файле pg_hba.conf или установлен в 0, то игнорирование метки безопасности сокета для этого подключения не происходит.

В ОС каждый пользователь может иметь множество меток безопасности, которое задается минимальной и максимальной метками безопасности диапазона. Чтобы поддержать эту модель в СУБД каждой сессии пользователя назначаются три метки безопасности: максимальная, минимальная и текущая. Их начальная инициализация осуществляется по следующему алгоритму:

1) после прохождения пользователем стандартной процедуры аутентификации сервер считывает из ОС значения максимальной и минимальной меток безопасности пользователя и принимает их как максимальную и минимальную метки безопасности сессии. При этом, если запись о метках безопасности для пользователя не найдена, то максимальная и минимальная метки безопасности принимаются равными нулю. Следовательно, пользователи, зарегистрированные только в сервере СУБД и не имеющие учетной записи в ОС сервера, всегда имеют минимальный уровень доступа к информации;

2) если параметр конфигурации ac_ignore_socket_maclabel установлен в FALSE, считывается метка безопасности входящего соединения, и, если она попадает в диапазон меток безопасности, считанных из ОС, то максимальная метка безопасности сессии устанавливается равной метке безопасности входящего соединения;

3) если параметр конфигурации ac_ignore_server_maclabel установлен в FALSE, то считывается метка безопасности серверного процесса и, если она несовместима с максимальной меткой безопасности сессии, то процесс аутентификации прерывается;

4) текущей меткой безопасности сессии становится максимальная метка безопасности сформированного таким образом диапазона.

Если на любом из этих этапов возникает ситуация с несовместимостью меток безопасности или выходом за пределы диапазона, то процесс аутентификации клиента прерывается и доступ к БД блокируется.

Если пользователь имеет параметр ac_capable_setmac, то он может изменять свою текущую метку безопасности в диапазоне от минимальной до максимальной.

СУБД предоставляет пользователям возможность создавать функции (и, следовательно, триггеры), указывая при этом, будут ли они выполняться с уровнем доступа пользователя, прямо или косвенно вызвавшего функцию (SECURITY INVOKER), или с уровнем доступа пользователя, создавшего эту функцию (SECURITY DEFINER). При этом в понятие «уровня доступа» входят как дискреционный уровень доступа, так и мандатный, который в данном случае определяется текущими мандатными атрибутами пользователя СУБД, вызвавшего или создавшего функцию, соответственно. При этом метки безопасности текущей сессии пользователя, вызвавшего функцию, не изменяются.

Следует учитывать, что:

1) при определении функции как SECURITY DEFINER она будет всегда вызываться с переустановкой мандатных атрибутов на атрибуты создавшего ее пользователя;

2) при определении функции как SECURITY INVOKER она всегда будет выполняться без изменения текущего значения мандатных атрибутов;

3) при вызове функции в качестве триггера выполняются следующие правила в дополнение к указанным:

а) перед вызовом в качестве триггера встроенной в СУБД функции к текущим мандатным атрибутам всегда добавляются флаги ac_capable_ignmaclvl и ac_capable_ignmaccat, чтобы обеспечить полноценную проверку ссылочной целостности БД;

б) перед вызовом в качестве триггера не встроенной функции в качестве текущих мандатных атрибутов всегда устанавливаются мандатные атрибуты пользователя, запустившего данную сессию (соединение) (т. е. пользователя с именем SESSION_USER). Это необходимо, чтобы предотвратить получение функцией-триггером пользователя с низким уровнем доступа высоких привилегий в случае каскадного вызова триггеров.

После возврата управления из функции значения текущих мандатных атрибутов всегда восстанавливаются в исходные (до вызова функции) значения.

Функции, написанные на языках низкого уровня, после их подключения имеют полный доступ ко всем внутренним структурам сервера СУБД и могут произвольно их модифицировать. Кроме этого, поскольку они выполняются в рамках процесса сервера, они имеют соответствующие права доступа к объектам ОС в среде функционирования сервера. Именно поэтому права пользователя `postgres`, под которым запускается сервер, необходимо свести к необходимому минимуму, минуя какой-либо контроль с его стороны (включая текущие мандатные атрибуты).

Поскольку в процессе работы с данными в СУБД возможно изменение организации их хранения путем изменения схемы объектов БД (метаданных), к подобным операциям также применяются ПРД.

Модификация метаданных осуществляется каждый раз при изменении структуры БД, что включает в себя создание, модификацию и удаление объектов БД.

Так как некоторые действия над объектами БД могут влиять на хранящиеся в них данные (как правило, модификация или удаление объекта или его части), при использовании мандатного управления доступом к данным объекта необходимо разграничивать и доступ к изменению метаданных в части, относящейся к этому объекту.

Все действия с объектами БД, аналогично операциям с данными, должны быть приведены к видам доступа на чтение и на запись информации для возможности применения к ним мандатных ПРД. Все множество операций с метаданными может быть приведено к двум типам доступа следующим образом:

- `CREATE`, `ADD` — доступ на запись;
- `ALTER`, `DROP` — последовательное выполнение доступа на чтение и запись информации;
- использование или обращение к объекту в других SQL-командах — доступ на чтение.

Проверка мандатных прав доступа к метаданным осуществляется одновременно с проверкой дискреционных прав доступа к ним после разбора и построения плана запроса непосредственно перед его выполнением, когда определены все необходимые для проверки данные и проверяемые объекты. Таким образом, доступ предоставляется только при одновременном разрешении дискреционными и мандатными ПРД.

Некоторые операции над объектами, такие как `DROP` всего объекта или его столбца и `TRUNCATE`, влекут за собой удаление данных. В случае защиты метками безопасности записей объекта существуют ограничения на выполнение этих операций.

Операции удаления невозможны при наличии разных меток безопасности на записях, т. к. операция применяется ко множеству строк. Это связано с тем, что операция удаления интерпретируется как последовательное предоставление доступа на чтение и на запись, что возможно только при равенстве меток безопасности субъекта и объекта. В случае, когда строки имеют разные метки безопасности, данное условие выполниться не может.

Операция удаления доступна только для администратора и пользователей, обладающих привилегиями игнорирования мандатного управления доступом.

5.2. Средства управления мандатными ПРД к объектам БД

Для управления мандатными ПРД к объектам БД СУБД используется графическая утилита `pgadmin4`.

Объекту БД при его создании задается метка безопасности, равная текущей метке безопасности создавшего его пользователя, мандатный признак CCR при этом выставляется в значение `ON`.

Если пользователь имеет параметр `ac_capable_chmac`, то он может менять метку безопасности принадлежащих ему объектов в пределах своего диапазона меток безопасности с помощью следующей команды:

```
MAC LABEL ON <тип_объекта_БД> <имя_объекта_БД> IS <новая_метка_безопасности>;
```

В качестве типа объекта указывается тип объекта БД, например `DATABASE`, `TABLE`, `FUNCTION`.

Аналогичным образом для контейнеров может быть изменен мандатный признак CCR:

```
MAC CCR ON <тип_объекта_БД> <имя_объекта_БД> IS { ON | OFF };
```

Дополнительно введен новый тип объекта — кластер `CLUSTER`. Для установки метки безопасности на кластер используется следующий запрос:

```
MAC LABEL ON CLUSTER IS <новая_метка_безопасности>;
```

что является синонимом к команде

```
MAC LABEL ON TABLESPACE pg_global IS <новая_метка_безопасности>;
```

Для изменения мандатного признака CCR кластера используется следующая команда:

```
MAC CCR ON CLUSTER IS { ON | OFF };
```

что является синонимом к команде

```
MAC CCR ON TABLESPACE pg_global IS { ON | OFF };
```

Значения меток безопасности объектов содержатся в полях `maclabel` таблиц системного каталога, откуда могут быть выбраны соответствующим запросом.

Значения мандатного признака CCR контейнеров содержатся в следующих полях таблиц системного каталога:

- для баз данных — `datmacccr` системной таблицы `pg_database`;
- для схем — `nspmacccr` системной таблицы `pg_namespace`;
- для табличных пространств - `spcmacccr` системной таблицы `pg_tablespace`;
- для отношений — `relmacccr` системной таблицы `pg_class`.

Для просмотра мандатного признака CCR кластера может быть использована следующая команда:

```
SELECT cluster_macccr;
```

По умолчанию записи создаваемых таблиц не защищены метками безопасности. Для того чтобы создать таблицы с записями, защищенными метками безопасности, следует использовать следующий вариант команды `CREATE TABLE`:

```
CREATE TABLE <имя_таблицы> (...--<список_столбцов>) WITH (MACS = true, ... );
```

При этом все вставляемые записи по умолчанию наследуют текущие метки безопасности создавших их пользователей. Пользователи, имеющие установленный мандатный атрибут `ac_capable_chmac`, могут явно задать значение метки безопасности вставляемой записи. Задаваемая метка безопасности должна быть в пределах диапазона меток безопасности пользователя, либо пользователь должен иметь атрибуты игнорирования мандатного управления `ac_capable_ignmaclvl` и `ac_capable_ignmaccat` с помощью варианта команды `INSERT`:

```
INSERT INTO <имя_отношения> (maclabel,...<список_столбцов>)
VALUES (<значение_метки_безопасности>,...<значения_столбцов>)
```

Для защиты записей уже созданных таблиц без меток безопасности следует использовать следующий вариант команды `ALTER TABLE`:

```
ALTER TABLE <имя_таблицы> SET WITH MACS;
```

После исполнения этой команды все записи таблицы автоматически получат текущую метку безопасности таблицы.

Для того чтобы убрать защиту записей метками безопасности, следует использовать следующий вариант команды ALTER TABLE:

```
ALTER TABLE <имя_таблицы> SET WITHOUT MACS;
```

Для изменения меток безопасности существующих записей пользователи с атрибутом ac_capable_chmac могут использовать команду CHMAC:

```
CHMAC <имя_отношения> SET maclabel=<новая_метка_безопасности> WHERE ...
```

Совокупная метка безопасности записей таблицы (максимальная по уровню конфиденциальности и наиболее полная по категориям конфиденциальности) может быть получена с помощью агрегирующей функции supmaclabel:

```
SELECT supmaclabel (maclabel) FROM <имя_отношения>;
```

Просмотреть значения меток безопасности доступных записей можно с помощью команды SELECT:

```
SELECT maclabel FROM <имя_отношения>
```

В командах INSERT и CHMAC значения меток безопасности не обязательно должны быть заданы в явном виде. Для задания метки безопасности допускается использование любого скалярного выражения, возвращающего результат, приводимый к типу метки безопасности.

Для того чтобы сохранить записи вместе с их метками безопасности в архиве и в дальнейшем загрузить их обратно, предусмотрен специальный флаг MACS команды COPY. Вывести доступные пользователю данные вместе с метками безопасности может любой пользователь, загрузить же обратно — только пользователь с установленным мандатным атрибутом ac_capabel_chmac. При этом метки безопасности загружаемых записей должны находиться в пределах диапазона меток безопасности пользователя, либо пользователь должен иметь атрибуты игнорирования мандатного управления ac_capable_ignmaclvl и ac_capable_ignmaccat. Например, выгрузка и обратная загрузка данных из/в таблицы test может выглядеть так:

```
COPY <имя_отношения> TO stdout WITH MACS
COPY <имя_отношения> FROM stdin WITH MACS
```

Использовать команду COPY без указания меток может любой пользователь. Загруженные таким образом данные будут иметь метки безопасности, равные текущей метке безопасности сессии пользователя.

Параметр конфигурации сервера ac_enable_copy_to_file разрешает выполнять команду COPY с выводом результатов в файл, доступный серверу СУБД. Для этого он должен быть установлен в TRUE.

5.3. Целостность мандатных атрибутов кластера баз данных

ДП-модель накладывает ограничение на классификационную метку объекта: классификационная метка объекта не может превышать классификационную метку контейнера, в котором он содержится (см. раздел 5).

Для вывода информации о соблюдении ДП-модели между контейнерами и находящимися в них объектами реализована SQL-функция check_mac_integrity, которая выводит информацию в следующем виде:

- objid — идентификатор объекта;
- classid — идентификатор класса объекта;
- cobjid — идентификатор контейнера, содержащего объект;
- cclassid — идентификатор класса контейнера, содержащего объект;
- status — результат проверки. Может принимать следующие значения: OK (модель соблюдается для объекта и контейнера) и FAIL (модель не соблюдается для объекта и контейнера).

Примечание. Информация о соблюдении модели выводится только для отношений (таблиц, представлений, последовательностей), схем, баз данных и табличных пространств.

Для исправления некорректно установленной метки безопасности отношения, например, при восстановлении резервной копии кластера ранних версий, используется SQL функция fix_mac_integrity. Данная функция может быть исполнена только пользователем с правами администратора, а также при установленном параметре ac_auto_adjust_macs = true в конфигурационном файле postgresql.conf.

5.4. Ссылочная целостность мандатных атрибутов

Средства обеспечения целостности в реляционных БД представляют собой механизмы автоматической поддержки системы правил, определяющих допустимость и корректность обрабатываемых данных, и могут быть разбиты на несколько видов:

1) ограничение целостности полей данных — ограничения, накладываемые на используемые в отношении домены (задание разрешенных диапазонов значений,

запрет наличия неопределенных значений NULL, задание значений по умолчанию) или ограничения непосредственно таблицы (уникальность каждой записи, ограничение уникальности по выбранным столбцам, вычисляемые значения столбцов и условия допустимых сочетаний значений в столбцах);

2) декларативная ссылочная целостность — описание зависимостей между разными отношениями в БД (наличия в одном отношении вторичного ключа, ссылающегося на первичный ключ в другом). Подобные зависимости могут быть выявлены при анализе предметной области между ее сущностями или при проектировании БД в процессе нормализации (в этом случае одна сущность предметной области может состоять из набора отношений). Ссылочная целостность реализуется путем описания ограничений на значения вторичных ключей в одном отношении и правил их обработки в случае изменения первичных ключей в другом;

3) динамическая ссылочная целостность — триггеры, назначаемые для выполнения при реализации заданного вида доступа к конкретной таблице. Триггер представляет собой исполняемый код, который может динамически проверить заданные условия корректности выполняемой операции, и при необходимости внести изменения в другие таблицы.

В приведенном определении отсутствуют мандатные атрибуты, так как в общем случае между классификационными метками записей разных таблиц может не быть зависимости. С другой стороны, существует частный случай ссылочной целостности, образованный между таблицами, являющимися частями одной сущности предметной области, что может возникнуть в процессе нормализации.

Для обеспечения целостности мандатных атрибутов список событий для ограничений целостности наряду с существующими событиями `ON SELECT`, `ON INSERT`, `ON UPDATE`, `ON DELETE` расширен событием изменения мандатных атрибутов `ON CHMAC`. Таким образом, возможно создание ограничение ссылочной целостности следующим образом:

```
ALTER TABLE <имя_таблицы1> ADD CONSTRAINT <имя_ограничения>
FOREIGN KEY (<список_полей1>) REFERENCES <имя_таблицы2> (<список_полей2>)
ON CHMAC {NO ACTION | RESTRICT | CASCADE }
```

Механизм действия такого ограничения целостности аналогичен механизму действия подобного ограничения для события `ON UPDATE` для данных: при изменении мандатных атрибутов записи в одной таблице можно указать каскадное изменение мандатных атрибутов связанных записей второй таблицы, либо запрет на возможность такого изменения.

Аналогично существует возможность создания триггеров для указанного события изменения мандатных атрибутов (`CHMAC`), например:

```
CREATE TRIGGER <имя_триггера>
BEFORE CHMAC ON <имя_таблицы>
```

```
FOR EACH ROW
EXECURE PROCEDURE <имя_процедуры>()
```

Таким же образом могут быть заданы и правила для видов:

```
CREATE RULE <имя_правила>
AS ON CHMAC TO <имя_вида>
DO ALSO ...
CREATE RULE <имя_правила>
AS ON CHMAC TO <имя_вида>
DO INSTEAD ...
```

В этом случае при изменении мандатных атрибутов записи будут вызываться соответствующие функции и правила, что дает возможность запрограммировать произвольную логику обеспечения целостности мандатных атрибутов внутри БД.

5.5. Особенности создания правил, системы фильтрации и триггеров

В СУБД правила (RULE), системы фильтрации (POLICY) и триггеры (TRIGGER) наследуют метку таблицы, для которой они созданы. Эти объекты могут быть созданы пользователем-владельцем таблицы (метка которого будет совпадать с меткой таблицы), либо пользователями с привилегиями игнорирования мандатного доступа. В противном случае генерируется ошибка доступа.

Если триггер использует триггерную функцию, метка которой отлична от {0,0}, то при исполнении триггера для таблицы со сброшенным мандатным признаком CCR возможны нарушения в работе триггеров. Это обусловлено тем, что запись триггерной функции может быть недоступна для пользователя в связи с мандатными ПРД. В таких случаях будет выведено сообщение:

```
cache lookup failed: внутренняя ошибка или отсутствуют необходимые мандатные
атрибуты
```

Во избежание этого настоятельно рекомендуется устанавливать мандатную метку триггерной функции в значение {0,0}.

Системы фильтрации (POLICY), такие как ROW LEVEL SECURITY, могут дополнять встроенные механизмы разграничения доступа путем добавления логики, реализующей правила выдачи строк пользователю. Фильтрация может основываться как на данных строки, так и на внешних факторах (например, текущем времени).

5.6. Особенности использования представлений и материализованных представлений

Представления (VIEW) и материализованные представления (MATERIALIZED VIEW) не могут иметь метки безопасности на строках, поскольку выполняют агрегацию данных из других источников данных (таблиц и представлений).

Примечание. Представления создаются с установленным флагом CCR, поэтому при попытке доступа к нему от имени пользователя, метка безопасности которого не превосходит метки безопасности представления, это представление не будет «видно» пользователю в силу мандатных ПРД. Для того, чтобы пользователь имел доступ к такому представлению, необходимо сбросить мандатный признак CCR представления.

5.7. Функции сравнения для типа maclabel

В СУБД изменено поведение следующих функций для типа maclabel: eq, ne, lt, le, gt, ge. В указанных функциях вместо индексного сравнения используется сравнение меток безопасности соответствующими операторами =, <>, <, <=, >, >=.

Для проверки на несравнимость меток безопасности используется функция nc.

Операторы индексного сравнения переименованы в maclabel_idx_eq, maclabel_idx_ne, maclabel_idx_lt, maclabel_idx_le, maclabel_idx_gt, maclabel_idx_ge.

5.8. Система привилегий СУБД

Система привилегий СУБД предназначена для передачи отдельным пользователям прав выполнения определенных административных действий. Обычный пользователь системы не имеет дополнительных привилегий.

Привилегии являются подклассом атрибутов пользователя СУБД.

Привилегии ОС, используемые в СУБД, приведены в таблице 5. Данные привилегии, кроме ac_session_maclabel, не могут быть изменены с помощью средств СУБД ни пользователями, ни администраторами.

Таблица 5

Привилегия	Описание
ac_session_maclabel	Текущая метка безопасности сессии пользователя СУБД. Эта метка безопасности определяет доступные пользователю объекты БД и является меткой безопасности по умолчанию для создаваемых пользователем объектов. При соединении пользователя с СУБД значение этого атрибута устанавливается равным метке безопасности соединения или ac_user_max_maclabel

Окончание таблицы 5

Привилегия	Описание
ac_user_max_maclabel	Максимально возможное значение для ac_session_maclabel
ac_user_min_maclabel	Минимально возможное значение для ac_session_maclabel
ac_capable_ignmaclvl	Позволяет пользователю игнорировать мандатное управление по уровням
ac_capable_ignmaccat	Позволяет пользователю игнорировать мандатное управление по категориям
ac_capable_mac_readsearch	Позволяет пользователю игнорировать мандатное управление по уровням и категориям при чтении данных
ac_capable_setmac	Позволяет пользователю изменять текущую метку безопасности своей сессии в пределах, заданных ее минимальным и максимальным значением
ac_capable_chmac	Позволяет пользователю изменять метки объектов БД

В случае если пользователь СУБД не зарегистрирован в ОС на стороне сервера СУБД, все его мандатные атрибуты имеют нулевое значение. Администраторам СУБД дополнительно к их атрибутам из ОС всегда добавляются атрибуты ac_capable_ignmaclvl, ac_capable_ignmaccat и ac_capable_chmac.

Для управления привилегиями СУБД может быть использована графическая утилита pgadmin4.

Просмотреть текущие значения привилегий (атрибутов пользователя) можно с помощью команды:

```
SHOW attr_name
```

Установить новое значение атрибута ac_session_maclabel можно с помощью команд:

```
SET ac_session_maclabel=<новое_значение_метки_безопасности>
SELECT set_config('ac_session_maclabel',<новая_метка_безопасности>,false);
```

В первой форме в качестве нового значения метки безопасности можно использовать только явно заданные значения метки, во второй — значение метки безопасности может быть любым выражением, возвращающим скалярное значение, приводимое к типу метки безопасности.

6. РОЛЕВОЕ УПРАВЛЕНИЕ ДОСТУПОМ В СУБД

СУБД была доработана для обеспечения соответствия требованиям по защите информации от несанкционированного доступа. При реализации названных требований была установлена необходимость обеспечения соответствия пользователей СУБД учетным записям в ОС.

Реализация ролевого метода управления доступом подразумевает назначение ролям индивидуальных пользователей СУБД одну из ролей:

- администратора СУБД;
- администратора БД;
- пользователя БД.

Роль администратора СУБД позволяет создавать и подключать кластеры баз данных, создавать и управлять учетными записями пользователей СУБД, управлять конфигурацией СУБД, управлять параметрами подключения пользователей СУБД к базам данных. Пользователю СУБД с данной ролью необходимо назначить высокий уровень целостности и включить его в группы `astra-admin` и `astra-audit`.

Пример

```
sudo usermod -a -G astra-admin <имя_администратора_СУБД>
sudo pdpl-user -i 63 <имя_администратора_СУБД>
```

Роль администратора БД позволяет создавать и управлять учетными записями пользователей кластеров баз данных, управлять конфигурацией БД, назначать права доступа пользователям кластера (пользователей информационной системы) к объектам доступа БД (базам данных, схемам, таблицам, записям, столбцам, представлениям и иным объектам доступа), создавать резервные копии БД и восстанавливать БД из резервной копии, создавать, модифицировать и удалять процедуры (программный код), хранимые в БД. Пользователю СУБД с данной ролью необходимо назначить групповую роль `pg_database_admin` и включить его в группу `astra-audit`.

В СУБД по умолчанию для роли администратора БД используется учетная запись `pg_astra_db_admin`.

Роль пользователя БД (пользователя информационной системы) позволяет создавать и манипулировать объектами доступа БД (таблица, запись или столбец, поле, представление и иные объекты доступа) и выполнять процедуры (программный код), хранимые в БД. Пользователю СУБД с данной ролью запрещено создание процедур (программного кода), хранимых в базах данных. Пользователю СУБД с данной ролью необходимо назначить групповую роль `pg_database_user`.

Назначение групповой роли для роли индивидуального пользователя СУБД или ее удаление выполняется с использованием команд GRANT и REVOKE соответственно:

```
GRANT <групповая_роль> TO <роль1>[, <роль2>] ;
REVOKE <групповая_роль> FROM <роль1>[, <роль2>] ;
```

Использование полномочий групповой роли ее членами возможно следующими способами:

1) временное — каждый член групповой роли может временно назначить себе полномочия данной роли с помощью команды:

```
SET ROLE <групповая_роль>;
```

Пример

```
SET ROLE pg_database_admin;
```

После выполнения команды для данного соединения с БД будут использоваться только полномочия групповой роли. Полномочия роли индивидуального пользователя СУБД, от имени которого было выполнено подключение к БД, использоваться не будут. Для всех создаваемых объектов БД владельцем будет назначена групповая роль, а не роль индивидуального пользователя СУБД;

2) постоянное — роли, которым назначен атрибут INHERIT, автоматически обладают полномочиями всех групповых ролей, членами которых они являются (в том числе и унаследованными этими групповыми ролями полномочиями).

7. КОНТРОЛЬ ЦЕЛОСТНОСТИ В СУБД

Для осуществления контроля целостности объектов внутри БД и СУБД используются следующие механизмы:

1) контроль целостности объектов (конфигураций и процедур) на основе вычисления их хеша (контрольных сумм) и его сравнения с эталонными значениями в соответствии с 7.1;

2) регламентный контроль целостности Another File Integrity Checker (AFICK) в соответствии с 7.2;

3) контроль целостности исполняемых файлов и расширений на основе внедрения цифровой подписи в соответствии с 11.3.

Контроль целостности осуществляется для следующих объектов:

- конфигурации СУБД;
- конфигурации БД (параметров GUC кластера);
- процедуры (программный код) СУБД;
- процедуры (программный код), хранимые в БД.

7.1. Контроль целостности объектов

Применение механизма контроля целостности объектов позволяет создавать списки эталонных значений контрольных сумм[1)] .

В СУБД реализован сценарий `pg_integrity`, который позволяет:

- включить или выключить контроль целостности, а также задать интервал периодического контроля целостности путем редактирования конфигурационного файла `/etc/postgreslq/<версия>/<имя_кластера>/postgresql.conf`;
- осуществить перерасчет контрольных сумм файлов и процедур, хранящихся в файле `/var/lib/postgresql/<версия>/<имя_кластера>/pg_integrity.conf`;
- перезагрузить кластер для применения настроек.

Контроль целостности в СУБД по умолчанию выключен. Для включения контроля целостности необходимо запустить сценарий `pg_integrity`:

```
sudo pg_integrity <версия> <имя_кластера> start
[--<уровень_реагирования>] [--delay=<интервал>]
```

[1)] С использованием алгоритма подсчета контрольных сумм по стандарту ГОСТ Р 34.11-2012) для объектов контроля и в дальнейшем при загрузке СУБД и в процессе ее эксплуатации обеспечивает вычисление контрольных сумм объектов и их сравнение с эталонными значениями.

где <уровень_реагирования> — действие при обнаружении нарушения контроля целостности (например, LOG (значение по умолчанию) или LIMIT_CONN). Соответствует значению параметра ac_integrity_check в конфигурационном файле /etc/postgreslq/<версия>/<имя_кластера>/postgresql.conf;

<интервал> — интервал периодического контроля целостности в секундах. Соответствует значению параметра ac_integrity_naptime в конфигурационном файле /etc/postgreslq/<версия>/<имя_кластера>/postgresql.conf.

Описание параметров ac_integrity_check и ac_integrity_naptime приведено в таблице 6.

Таблица 6

Привилегия	Описание
ac_integrity_check	Осуществляет контроль целостности объектов. По умолчанию выключен. Для запуска контроля целостности необходимо установить параметр в одно из следующих значений: LOG, LIMIT_CONN. Если параметр ac_integrity_check установлен в значение LOG, то при выявлении нарушений целостности объектов контроля будет зарегистрировано сообщение о нарушении в журнале безопасности /parsec/log/astra/events. Если параметр ac_integrity_check установлен в значение LIMIT_CONN, то при выявлении нарушений целостности объектов контроля будет зарегистрировано сообщение о нарушении в журнале безопасности /parsec/log/astra/events и доступ к СУБД и БД будет заблокирован для всех пользователей, кроме администратора СУБД
ac_integrity_naptime	Задает интервал периодического контроля целостности в секундах

При обнаружении и исправлении нарушений целостности объектов контроля требуется выполнить перезагрузку кластера для восстановления доступа пользователей БД к СУБД. Отключение контроля целостности выполняется следующей командой:

```
sudo pg_integrity <версия> <имя_кластера> stop
```

Устанавливает параметр ac_integrity_check в значение NONE.

Настройка контроля целостности осуществляется в файле /var/lib/postgresql/<версия>/<имя_кластера>/ pg_integrity.conf, для этого необходимо добавить в него контролируемые файлы и их контрольные суммы . Для конфигурации СУБД определена секция [FILE], и записи имеют вид:

```
pg_audit.conf : "22d6dac07b2a653e677dc8ca3b634b746dac1ee996b2df30c362e30fc461a
636203b1521d98775e4ca3348986d0fd438b45dbb0ce516a2070ad93f8731c61fb5"
```

Для параметров БД (параметров GUC) определена секция [PARAMETERS], и записи имеют вид:

```
ac_audit_mode : "internal"
ac_auto_adjust_macs : "off"
ac_capable_chmac : "да"
autovacuum : "on"
```

Для процедур БД определена секция [SQL], и записи имеют вид:

```
sql_name : "1f326a4cbfc35c9905aa7aa9c690786d5e3d1e3013ad0ce733cfda89502f99410
de32b82ecba27b3073199e7bc1079dfe17e63269134c82a6c0da8de2da351b1" :
"scheme_name" : "integer, integer" : "db_name"
```

В целях исключения потери производительности не рекомендуется использовать механизм контроля целостности для контроля свыше 100.000 объектов (файлов и процедур) в кластере СУБД.

Проверить работу контроля целостности, произвести расчет контрольных сумм для файлов и процедур, а также осуществить перерасчет контрольных сумм данных объектов при их изменении возможно путем применения соответствующих SQL-функций. Данные действия выполняются администратором СУБД. Описание SQL-функций приведено в таблице 7.

Таблица 7

Привилегия	Описание
SELECT get_integrity_state	Проверка контроля целостности
SELECT get_integrity_params	Проверка текущих значений параметров ac_integrity_check и ac_integrity_naptime
SELECT get_file_hash ('имя_файла')	Расчет контрольной суммы для файла (например, SELECT get_file_hash ('pg_audit.conf'))
SELECT get_procedure_hash ('имя_функции', 'имя_схемы_БД', 'аргумент_функции')	Расчет контрольной суммы для функции (например, SELECT get_procedure_hash ('test_func','test_db','integer, integer'))

Осуществить перерасчет контрольных сумм для файлов и процедур, хранящихся в файле /var/lib/postgresql/<версия>/<имя_кластера>/pg_integrity.conf, также возможно с помощью сценария pg_integrity. Для этого необходимо выполнить следующую команду:

```
sudo pg_integrity <версия> <имя_кластера> refill [--<объекты_перерасчета>]
```

где <объекты_перерасчета> — параметр, определяющий перерасчет секций процедур [SQL], файлов [FILE], параметров [PARAMETERS] в файле pg_integrity.conf.

Пример

```
sudo pg_integrity <версия> <имя_кластера> refill --sql --file
     --parameters
```

Для удаления контрольных сумм файлов и процедур, хранящихся в файле var/lib/postgresql/<версия>/<имя_кластера>/pg_integrity.conf, необходимо выполнить следующую команду:

```
sudo pg_integrity <версия> <имя_кластера> clear [--<объекты_перерасчета>]
```

Список доступных команд и описание параметров сценария pg_integrity также возможно вызвать с помощью команды:

```
sudo pg_integrity <версия> <имя_кластера>
```

Средствами СУБД также контролируется целостность поставленных на контроль объектов БД после их восстановления из резервной копии. Контроль целостности резервной копии при восстановлении из нее обеспечивается применением ЗПС — при нарушении целостности резервной копии ее восстановление будет заблокировано.

Регистрация событий безопасности, связанных с контролем целостности в СУБД, осуществляется подсистемой регистрации событий. Описание работы подсистемы регистрации событий приведено в документе РУСБ.10015-01 97 01-1. Информирование администратора СУБД о нарушении целостности объектов контроля, а также администраторов БД о нарушении целостности конфигураций БД и хранимых в БД процедур осуществляется подсистемой регистрации событий.

7.2. Регламентный контроль целостности AFICK

Организация регламентного контроля целостности объектов контроля обеспечивается программным средством AFICK путем вычисления контрольных сумм файлов и соответствующих им атрибутов подсистемы безопасности PARSEC (мандатных атрибутов и атрибутов расширенной подсистемы регистрации событий) с последующим сравнением вычисленных значений с эталонными. Описание AFICK приведено в документе РУСБ.10015-01 97 01-1.

Возможность проведения периодического контроля осуществляется с использованием системного планировщика заданий cron. Системный планировщик заданий cron также позволяет выполнять проверку целостности объектов контроля в процессе загрузки ОС.

Для контроля целостности необходимо в конфигурационном файле /etc/afick.conf задать список файлов и каталогов, которые будет контролировать AFICK. На контроль рекомендуется ставить файл /var/lib/postgresql/<версия>/<имя_кластера>/pg_integrity.conf для защиты базы эталонных контрольных сумм и файлы сценариев (например, /usr/bin/pg_integrity, /usr/bin/pg_ctlcluster и /usr/bin/dropcluster) для обнаружения нарушений работы СУБД и несанкционированных изменений.

8. УПРАВЛЕНИЕ КЛАСТЕРАМИ СУБД

Несмотря на возможность управления серверами СУБД с помощью штатных утилит PostgreSQL, настоятельно рекомендуется использовать для этого поставляемыми с ОС инструменты управления кластерами СУБД. Данные утилиты входят в состав пакета `postgresql-common` и представляют собой сценарии, осуществляющие вызов штатных утилит PostgreSQL, но с учетом специфики ОС и возможности одновременного управления кластерами разных версий СУБД.

Данные утилиты позволяют создать несколько экземпляров кластеров разных версий с собственными конфигурационными файлами. Правкой конфигурационных файлов кластера можно указать порт, IP-адреса, которые будет слушать кластер и т.д.

К описываемым утилитам относятся:

- `pg_createcluster` — утилита по созданию нового кластера (см. 8.1);
- `pg_ctlcluster` — утилита по управлению кластером (см. 8.2);
- `pg_dropcluster` — утилита по удалению кластера (см. 8.3);
- `pg_lsclusters` — утилита по просмотру состояний существующих кластеров (см. 8.4);
- `pg_upgradecluster` — утилита по обновлению кластера (см. 8.5).

Утилиты используют единый формат вызова следующего вида:

```
pg_<имя утилиты> [опции] версия-кластера имя-кластера <действие>
```

В зависимости от назначения утилиты некоторые элементы командной строки вызова могут отсутствовать.

Общий подход заключается в физическом разнесении файлов разных кластеров в пределах файловой системы. Как правило, используются следующее расположение:

- `/usr/lib/postgresql/<версия_СУБД>/` — загружаемые модули СУБД определенной версии;
- `/usr/share/postgresql/<версия_СУБД>/` — разделяемые файлы СУБД определенной версии;
- `/etc/postgresql/<версия_кластера>/<имя_кластера>` — конфигурационные файлы конкретного кластера;
- `/var/lib/postgresql/<версия_кластера>/<имя_кластера>` — непосредственно каталог данных конкретного кластера.

Далее приведено описание использования перечисленных утилит. Более подробная информация может быть найдена в руководстве man для каждой из утилит.

8.1. Создание кластера pg_createcluster

Для создания кластера СУБД применяется утилита pg_createcluster. Утилита принимает в качестве аргументов версию и имя требуемого кластера. При этом создается кластер с настройками по умолчанию. В конечном счете вызывается утилита СУБД PostgreSQL initdb, которой передаются принятые в ОС расположения файлов и текущие значения локали. Существует возможность изменения поведения по умолчанию с помощью опций.

Формат вызова:

```
pg_createcluster [опции] версия-кластера имя-кластера [--port PORT]
```

Например, для создания кластера main, работающего на порту 5433, необходимо выполнить:

```
pg_createcluster <номер_версии> main --port 5433
```

8.2. Управление кластером pg_ctlcluster

Для управления кластером СУБД применяется утилита pg_ctlcluster. Утилита принимает в качестве аргументов версию, имя требуемого кластера и одно из действий:

- start — запуск экземпляра СУБД для указанного кластера;
- stop — останов экземпляра СУБД для указанного кластера;
- restart — перезапуск экземпляра СУБД для указанного кластера;
- reload — повторное чтение конфигурационных файлов указанного кластера сервером СУБД;
- promote — специальная команда перевода резервного сервера из состояния восстановления в состояние приема запросов.

В конечном счете вызывается утилита СУБД pg_ctl, которой передаются принятые в ОС расположения файлов. Существует возможность изменения поведения по умолчанию с помощью опций.

Формат вызова:

```
pg_ctlcluster [опции] версия-кластера имя-кластера действие -- [опции pg_ctl]
```

Например, для перезапуска кластера main для СУБД необходимо выполнить:

```
pg_ctlcluster <номер_версии> main restart
```

8.3. Удаление кластера pg_dropcluster

Для удаления кластера СУБД применяется утилита `pg_dropcluster`. Утилита принимает в качестве аргументов версию и имя требуемого кластера. При этом удаляются все файлы кластера. Указание опции `--stop` приводит к остановке сервера СУБД перед удалением файлов кластера.

Формат вызова:

```
pg_dropcluster [--stop] версия-кластера имя-кластера
```

Например, для удаления кластера `main` необходимо выполнить:

```
pg_dropcluster <номер_версии> main
```

8.4. Просмотр состояние кластеров pg_lsclusters

Для просмотра состояния существующих кластеров СУБД применяется утилита `pg_lsclusters`.

Формат вызова:

```
pg_lsclusters
```

Утилита выводит список существующих кластеров в следующем виде:

```
# pg_lsclusters
Ver Cluster Port Status Owner    Data directory              Log file
15  main    5432 online postgres /var/lib/postgresql/15/main /var/log/...
```

8.5. Обновление кластера pg_upgradecluster

`pg_upgradecluster` обновляет существующий кластер СУБД на новую версию кластера, указанную с помощью параметра `newversion` (по умолчанию: последняя доступная версия). Конфигурационные файлы копируются со старого кластера на новый.

Формат вызова:

```
/usr/bin/pg_upgradecluster [опции] старая-версия имя-кластера
```

```
[новый-каталог-данных]
```

ВНИМАНИЕ! После завершения процесса обновления необходимо установить параметр `ac_auto_adjust_macs` в значение `false` в конфигурационном файле `postgresql.conf` «нового» кластера.

8.6. Особенности обновления кластера при использовании pg_upgrade

Для обновления кластеров СУБД используется штатная утилита `pg_upgrade`, входящая в состав пакета `postgresql-common`.

Из-за различий моделей мандатного управления доступом при выполнении обновления с помощью `pg_upgrade` на новую версию СУБД необходимо включить флаг автоматического подъема метки безопасности контейнерных объектов баз данных (параметр `ac_alow_auto_adjust_mac = on`). При штатной эксплуатации СУБД данный параметр должен быть отключен.

Для обновления кластера со следующими обозначениями:

- `NEW_VERSION` — «новая» версия кластера;
- `NEW_CLUSTER_NAME` — имя «нового» кластера;
- `OLD_BIN` — путь к «старому» каталогу с исполняемыми файлами СУБД;
- `NEW_BIN` — путь к «новому» каталогу с исполняемыми файлами СУБД;
- `OLD_CONF` — путь к каталогу конфигурационных файлов «старого» кластера;
- `NEW_CONF` — путь к каталогу конфигурационных файлов «нового» кластера;
- `OLD_PORT` — порт «старого» кластера СУБД;
- `NEW_PORT` — порт «нового» кластера СУБД

предусмотрен следующий типичный сценарий:

1) создать новый кластер:

```
sudo pg_createcluster PG_NEW_VERSION PG_NEW_CLUSTER_NAME
```

2) в конфигурационном файле `postgresql.conf` нового кластера указать

```
ac_allow_auto_adjust_mac = on
```

3) убедиться, что «старый» и «новый» кластера выключены. Если это не так, выключить их с помощью утилиты `pg_ctlcluster`;

4) используя утилиту `pg_upgrade`, выполнить процесс переноса данных со «старого» на «новый» кластер (команду необходимо выполнить от имени пользователя `postgres`):

```
/usr/lib/postgresql/15/bin/pg_upgrade -b PG_OLD_BIN -B
```

```
PG_NEW_BIN -d PG_OLD_CONF -D PG_NEW_CONF -p PG_OLD_PORT -P
PG_NEW_PORT}
```

ВНИМАНИЕ! После завершения процесса обновления необходимо установить параметр `ac_auto_adjust_macs` в значение `false` в конфигурационном файле `postgresql.conf` «нового» кластера.

9. УПРАВЛЕНИЕ БАЗАМИ ДАННЫХ

Под управлением БД подразумевается непосредственно создание и удаление БД, управление пользователями и процедурными языками.

Создание кластера БД состоит из создания каталогов для хранения данных БД, создания разделяемых таблиц системного каталога (таблиц, относящихся ко всему кластеру БД, а не к конкретной БД), и создания БД `template1` и `postgres`. При создании в дальнейшем новых БД в них копируется содержимое БД `template1`. Таким образом, все, что установлено в БД `template1`, автоматически будет скопировано в каждую создаваемую в дальнейшем БД. БД `postgres` является БД по умолчанию для использования пользователями, утилитами и сторонними приложениями.

Создание кластера выполняется администратором на сервере с помощью утилиты `initdb`.

9.1. Соединение с сервером БД

Работа с СУБД требует установки соединения с сервером БД. Задание свойств соединения осуществляется с помощью параметров, приведенных в таблице 8, которые указываются при использовании инструментов командной строки при обращении к БД.

Таблица 8

Параметр	Описание
`-d <имя_БД>`, `--dbname=<имя_БД>`	Указывает имя БД для подключения. Равнозначно указанию имени БД в команде первом аргументе, не являющемся параметром. Вместо имени может задаваться строка подключения. В этом случае параметры в строке подключения переопределяют одноименные параметры, заданные в команде. Пример `psql "service=myservice sslmode= require"`
`-h <имя_сервера>`, `--host=<имя_сервера>`	Указывает имя сервера БД или каталог сокетов UNIX, если начинается с символа «/»
`-l <база_данных>` `--database=<база_данных>`	Указать альтернативную БД — шаблон
`-p <порт>`, `--port=<порт>`	Указывает порт сервера БД или расширение имени сокета UNIX, по которым сервер принимает соединения
`-U <имя_пользователя>`, `--username=<имя_пользователя>`	Указывает имя пользователя для установки соединения

Окончание таблицы 8

Параметр	Описание
`-w`, `--no-password`	Подавление запроса пароля пользователя. В случае, когда установка соединения с сервером требует ввода пароля, а пароль недоступен, например из файла `.pgpass`, попытка установки соединения завершается ошибкой. Опция полезна при выполнении пакетов заданий или сценариев, в процессе обработки которых отсутствует пользователь, который может вводить пароль
`-W`, `--password`	Принудительный запрос пароля при установке соединения. Опция не является существенной, т. к. утилита по умолчанию всегда запрашивает пароль в случае, когда сервер требует ввода пароля при установке соединения. В тоже время для определения необходимости ввода пароля утилита делает дополнительный запрос к серверу, которого можно избежать, указав этот параметр

Если значение не соответствует ни одному из параметров, оно воспринимается как имя БД (или имя пользователя, если имя БД уже было получено).

При отсутствии перечисленных значений (или при их неверном указании) используются переменные окружения (`PGDATABASE`, `PGHOST`, `PGPORT`, `PGUSER`), определяющие параметры соединения по умолчанию. Также возможно использовать файл `~/.pgpass` для исключения необходимости регулярного ввода пароля.

Информацию о версии и способе вызова утилит и допустимых аргументов можно получить с помощью параметров `--help` (показать справку по вызову команды) и `--version` (показать версию).

9.2. Создание и удаление баз данных

Для создания новой БД используется инструмент `createdb`, а для удаления — инструмент `dropdb`.

По умолчанию владельцем новой БД становится пользователь, выполняющий команду. В то же время в качестве владельца новой БД может быть указан другой пользователь с помощью параметра `-O`, если выполняющий команду пользователь обладает соответствующими привилегиями. При этом удаление может выполнить только администратор или владелец БД.

Обобщенный способ вызова заключается в передаче параметров и имени БД. При этом используются правила установки соединения, приведенные в таблице 8.

Синтаксис команд:

```
createdb [параметр]... [база_данных] [описание]
dropdb [параметр]... [база_данных]
```

9.3. Управление пользователями

В СУБД для управления правами на доступ к БД используется концепция ролей. Под ролью, в зависимости от параметров, понимается пользователь или группа пользователей БД. Роли могут являться владельцами объектов БД (например, таблиц) и могут назначать привилегии на управление объектами для других ролей, имеющих доступ к данным объектам. Кроме того, существует возможность предоставления членства в роли для другой роли, что позволяет членам роли использовать привилегии, назначенные роли, членами которой они являются.

Корректная работа с СУБД предполагает использование механизма ЕПП, что подразумевает использование в качестве пользователей СУБД пользователей домена ЕПП.

Для создания нового пользователя или роли используется инструмент `createuser`, для удаления — инструмент `dropuser`.

Только администраторы и пользователи с привилегией `CREATEROLE` могут создавать и удалять пользователей и роли. Удалять администраторов может только администратор.

Синтаксис:

```
createuser [параметр]... [роль]
dropuser [параметр]... [роль]
```

При вызове используются правила установки соединения в соответствии с таблицей 8.

9.4. Использование процедурных языков

СУБД предоставляет пользователям возможность создавать процедуры (функции) и триггеры для обработки данных, хранящихся в БД. Для этого могут использоваться следующие процедурные языки: PL/Perl, PL/pgSQL, PL/Python и PL/Tcl.

Для возможности использования конкретного процедурного языка его необходимо установить в конкретную БД.

Для установки поддержки процедурного языка в БД используется утилита `createlang`, для удаления поддержки языка из БД — `droplang`.

Синтаксис:

```
createlang [параметр]... <язык> [база_данных]
droplang [параметр]... <язык> [база_данных]
```

Несмотря на то, что поддержка процедурного языка может быть выполнена непосредственно некоторыми SQL-командами (например, `DROP LANGUAGE`), рекомендуется использовать утилиты `createlang` и `droplang`, т. к. они осуществляют необходимые проверки.

При вызове используются правила установки соединения в соответствии с таблицей 8.

9.5. Оптимизация баз данных

Кроме работы с БД существует необходимость осуществлять ряд системных операций как для оптимизации работы СУБД, так и в качестве регламентных работ по обеспечению отказоустойчивости и возможности восстановления после сбоев.

С целью оптимизации работы СУБД для увеличения производительности используются как архитектурные способы при разработке конкретной схемы БД, так и применение различных способов индексирования информации. При этом может возникать необходимость перестройки индексов в процессе изменения большого количества данных.

Для пересоздания индексов в БД используется утилита `reindexdb`, а для кластеризации (оптимизации индексов) ранее кластеризованных таблиц в БД используется утилита `clusterdb`. Утилита `clusterdb` находит таблицы, которые были ранее кластеризованы, и кластеризует их заново по тем же индексам, которые были указаны до этого. Таблицы, которые до этого не были кластеризованы, не затрагиваются.

Также возможна очистка таблиц от ранее удаленных записей. Для очистки таблиц и сбора статистики, необходимой для работы оптимизатора запросов, в БД используется утилита `vacuumdb`.

Синтаксис утилит:

```
reindexdb [<параметр>]... [<база_данных>]
clusterdb [<параметр>]... [<база_данных>]
vacuumdb [<параметр>]... [<база_данных>]
```

При вызове используются правила установки соединения в соответствии с таблицей 8.

10. СРЕДСТВА ОБЕСПЕЧЕНИЯ НАДЕЖНОСТИ

10.1. Настройка репликации

Репликация — механизм синхронизации содержимого нескольких копий баз данных.

СУБД предоставляет несколько способов репликации БД:

- пофайловая — основана на покомандном копировании WAL-файлов (10.1.1);
- потоковая — основана на транслировании потока WAL-файлов ведущего сервера на ведомый (10.1.2);
- слоты репликации — модификация потоковой репликации, позволяющая реплицировать содержимое одного ведущего сервера на несколько ведомых. В этом режиме ведущий сервер не удаляет журналы транзакций до тех пор, пока ведомый (ведомые) сервер (серверы) не отчитается (отчитаются) об их применении (10.1.3);
- логическая — основана на пересылке изменений состояния данных ведущего сервера и применении их на ведомом;

При описании различных способов настройки репликации используются следующие обозначения:

- `VERSION` — версия СУБД;
- `MASTER` — имя кластера, являющегося ведущим сервером;
- `MASTER_IP` — IP-адрес ведущего сервера;
- `MASTER_PORT` — порт работы ведущего сервера;
- `SLAVE` — имя кластера, являющегося ведомым сервером;
- `SLAVE_IP` — IP-адрес ведомого сервера;
- `SLAVE_PORT` — порт работы ведомого сервера;
- `REPL` — имя пользователя, обладающего правами репликации (`REPLICATION`), от имени которого реализуется репликация;
- `WAL_DIR` — каталог для хранения WAL-файлов ведущего сервера.
- `REPL_PASSWORD` — пароль пользователя, от имени которого реализуется репликация;
- `MASTER_DB` — база данных, из которой реплицируются данные;
- `SLAVE_DB` — база данных, в которую реплицируются данные.

10.1.1. Настройка пофайловой репликации

Для настройки пофайловой (Log Shipping) репликации необходимо выполнить следующие действия:

1) остановить ведомый кластер баз данных:

```
sudo pg_ctlcluster $VERSION $SLAVE stop
```

2) создать каталог для хранения WAL файлов:

```
sudo mkdir $WAL_DIR
```

3) назначить владельцем этого каталога пользователя `postgres`:

```
sudo postgres.postgres $WAL_DIR
```

4) установить следующие параметры в конфигурационном файле `/etc/postgresql/<версия>/<имя_кластера>/postgresql.conf` ведущего сервера:

```
wal_level = replica
archive_mod = on
archive_command = 'cp %p $WAL_DIR/%f < /dev/null'
archive_timeout = 1
```

5) удалить все файлы в каталоге ведомого сервера за исключением файла `pg_audit.conf`:

```
sudo cd /var/lib/postgresql/$VERSION/$SLAVE/ &&
    sudo ls grep -v pg_audit.conf | xargs rm -rf
```

6) выполнить копию ведущего сервера (выполняется на ведомом сервере от имени пользователя `postgres`):

```
pg_basebackup -h $MASTER_IP -p $MASTER_PORT -U $REPL -D
    /var/lib/postgresql/$VERSION/$SLAVE/
```

7) установить следующие параметры в конфигурационном файле `/etc/postgresql/<версия>/<имя_кластера>/postgresql.conf` ведомого сервера:

```
hot_standby = on
restore_command = 'cp
    /usr/lib/postgresql/$VERSION/$MASTER/WAL_DIR/%f %p'
recovery_end_command = 'rm -f /tmp/trigger.$VERSION.$SLAVE_PORT'
promote_trigger_file = '/tmp/trigger_$SLAVE.$SLAVE_PORT'
```

8) создать пустой файл `standby.signal` на ведомом сервере (выполняется от имени пользователя `postgres`):

```
touch /var/lib/postgresql/$VERSION/$SLAVE/standby.signal
```

9) запустить ведомый сервер:

```
sudo pg_ctlcluster $VERSION $SLAVE start
```

10.1.2. Настройка потоковой репликации

Для настройки потоковой (Streaming) репликации необходимо выполнить следующие действия:

1) остановить ведомый кластер баз данных:

```
sudo pg_ctlcluster $VERSION $SLAVE stop
```

2) установить следующие параметры в конфигурационном файле /etc/postgresql/<версия>/<имя_кластера>/postgresql.conf ведущего сервера:

```
listen_addresses = '*'
wal_level = replica
max_wal_senders = 2
wal_keep_segments = 1024
```

3) установить переменную окружения $REPL для аутентификации пользователя и переменную окружения $SLAVE_IP для разрешения подключения к базе данных replication с определенного IP-адреса, добавив следующую строку в конфигурационном файле /etc/postgresql/<версия>/<имя_кластера>/pg_hba.conf ведущего сервера:

```
host replication $REPL $SLAVE_IP trust
```

Примечание. При использовании метода аутентификации trust сервер должен работать в защищенной сети;

4) удалить все файлы в каталоге ведомого сервера за исключением файла pg_audit.conf:

```
sudo -s
ls /var/lib/postgresql/$VERSION/$SLAVE/ | grep -v pg_audit.conf |
     xargs rm -rf
```

5) создать копию ведущего сервера (выполняется от имени пользователя postgres):

```
pg_basebackup -h $MASTER_IP -p $MASTER_PORT -U $REPL -D
    /var/lib/postgresql/$VERSION/$SLAVE/
```

6) установить следующие параметры в конфигурационном файле /etc/postgresql/<версия>/<имя_кластера>/postgresql.conf ведомого сервера:

```
hot_standby = on
primary_conninfo = 'host=$MASTER_IP port=$MASTER_PORT user=$REPL'
promote_trigger_file = '/tmp/trigger_$SLAVE.$SLAVE_PORT'
```

7) создать пустой файл standby.signal на ведомом сервере (выполняется от имени пользователя postgres):

```
touch /var/lib/postgresql/$VERSION/$SLAVE/standby.signal
```

8) запустить ведомый сервер:

```
sudo pg_ctlcluster $VERSION $SLAVE start
```

10.1.3. Настройка репликации с помощью слотов репликации

Для настройки репликации с помощью слотов репликации необходимо выполнить следующие действия:

1) остановить ведомый кластер баз данных:

```
sudo pg_ctlcluster $VERSION $SLAVE stop
```

2) установить следующие параметры в конфигурационном файле `/etc/postgresql/<версия>/<имя_кластера>/postgresql.conf` ведущего сервера:

```
listen_addresses = '*'
wal_level = replica
max_wal_senders = 1
max_replication_slots = 1
```

3) установить переменную окружения `$REPL` для аутентификации пользователя и переменную окружения `$SLAVE_IP` для разрешения подключения к базе данных `replication` с определенного IP-адреса, добавив следующую строку в конфигурационный файл `/etc/postgresql/<версия>/<имя_кластера>/pg_hba.conf` ведущего сервера:

```
host replication $REPL $SLAVE_IP trust
```

Примечание. При использовании метода аутентификации `trust` сервер должен работать в защищенной сети;

4) удалить все файлы в каталоге ведомого сервера за исключением файла `pg_audit.conf`:

```
sudo cd /var/lib/postgresql/$VERSION/$SLAVE/ &&
    sudo ls grep -v pg_audit.conf | xargs rm -rf
```

5) создать слот репликации на ведущем сервере:

```
psql -h $MASTER_IP -p $MASTER_PORT -U $REPL -d template1 -c "SELECT
    pg_create_physical_replication_slot('slave');"
```

6) создать копию ведущего сервера (выполняется на ведомом сервере от имени пользователя `postgres`):

```
pg_basebackup -h $MASTER_IP -p $MASTER_PORT -U $REPL -D
    /var/lib/postgresql/$VERSION/$SLAVE/ --slot=$SLOT_NAME
```

7) установить следующие параметры в конфигурационном файле /etc/postgresql/<версия>/<имя_кластера>/ postgresql.conf ведомого сервера:

```
hot_standby = on
primary_conninfo = 'host=$MASTER_IP port=$MASTER_PORT user=$REPL'
promote_trigger_file = '/tmp/trigger_$SLAVE.$SLAVE_PORT'
primary_slot_name = 'slave'
```

8) создать пустой файл standby.signal на ведомом сервере (выполняется от имени пользователя postgres):

```
touch /var/lib/postgresql/$VERSION/$SLAVE/standby.signal
```

Несмотря на то, что файл пустой, само наличие этого файла означает, что данный сервер будет использоваться для репликации;

9) перезапустить ведомый сервер:

```
sudo pg_ctlcluster $VERSION $SLAVE start
```

10.2. Pgpool-II

Pgpool-II представляет собой промежуточное программное обеспечение, которое функционирует между клиентом и сервером БД СУБД. Pgpool-II реализует следующие возможности:

- создание высокопроизводительной сетевой структуры между узлами, кластерами и пользователями;
- синхронную репликацию данных на множество серверов без остановки;
- балансировку нагрузки между узлами кластера;
- обнаружение отказа и переключение нагрузки (Failover и Switchover).

Для настройки Pgpool-II необходимо отредактировать параметры в конфигурационном файле /etc/pgpool2/pgpool.conf. Настройка параметров в данном файле осуществляется аналогично настройке конфигурационного файла /etc/postgresql/<версия>/<имя_кластера>/postgresql.conf СУБД.

Описание основных параметров Pgpool-II приведено в таблице 9.

Таблица 9

Параметр	Описание
listen_addresses	Определяет TCP/IP-адреса, по которым сервер должен ожидать соединения от клиентских приложений. Значение формируется в виде перечня разделенных запятой имен узлов и/или IP-адресов. Специальный знак «*» соответствует всем доступным IP-адресам. Если список пуст, сервер не слушает ни один IP-интерфейс. В этом случае установка соединения с сервером возможна только с использованием доменных сокетов UNIX. Значение по умолчанию — localhost
port	Определяет TCP-порт, на котором сервер должен ожидать соединения от клиентских приложений. Для всех IP-адресов, указанных в listen_addresses (string), используется один и тот же порт
backend_hostnameN	Задает TCP/IP-адрес сервера N СУБД, его DNS-имя или путь к директории с сокетом UNIX для локального подключения
backend_portN	Определяет TCP-порт сервера N СУБД
backend_weightN	Определяет долю нагрузки на сервер N СУБД
backend_data_directoryN	Определяет путь к каталогу PGDATA сервера N СУБД
backend_flagN	Флаги поведения сервера N СУБД (например, может ли сервер быть автоматически исключен в случае сбоя)
backend_application_nameN	Определяет уникальное имя кластера СУБД, которое должно соответствовать значению параметра cluster_name в конфигурационном файле /etc/postgresql/<версия>/<имя_кластера>/postgresql.conf
enable_pool_hba	Определяет будет ли Pgpool-II использовать конфигурационный файл pool_hba.conf для аутентификации клиентов. По умолчанию аутентификация выключена
log_destination	Определяет журнал для вывода сообщений логирования. По умолчанию stderr, также поддерживается режим вывода в syslog
load_balance_mode	Включает или выключает режим балансировки нагрузки серверов. По умолчанию выключен
replicate_select	Возможность отправления прокси-сервером Pgpool-II команд по выбору данных (SELECT) на все узлы. По умолчанию выключена
sr_check_user	Определяет имя пользователя, проверяющего статус репликации. По умолчанию — nobody
sr_check_password	Определяет пароль пользователя, проверяющего статус репликации. По умолчанию используется пустое значение
health_check_user	Определяет имя пользователя, проверяющего статус узла
health_check_period	Определяет периодичность проверки состояния узла в секундах

Окончание таблицы 9

Параметр	Описание
health_check_max_retries	Определяет количество попыток проверки состояния узла, после которых он будет определен как недоступный
health_check_retry_delay	Определяет задержку для повторной проверки состояния узла в секундах
backend_clustering_mode	Определяет тип репликации (например, streaming_replication или logical_replication)
use_watchdog	Включает или отключает функцию мониторинга состояния узлов Pgpool-II
arping_path	Путь к исполняемому файлу arping для управления виртуальным IP-адресом Pgpool-II
arping_cmd	Команда для отправки ARP-запросов для анонсирования виртуального IP-адреса Pgpool-II на сетевом интерфейсе
auto_failback	Автоматическое включение узлов СУБД под управлением Pgpool-II в кластер после их восстановления (синхронизации с ведущим узлом СУБД)
recovery_1st_stage_command	Сценарий синхронизации ведомого узла с ведущим. Описание и шаблон сценария представлен в каталоге /etc/pgpool2/sample_scripts/ в файле с расширением *.sample
wd_remove_shutdown_nodes	Удаляет узел Pgpool-II из кластера при недоступности узла (если ранее отключенный узел будет запущен снова, он автоматически будет добавлен обратно в кластер Pgpool-II)
wd_lost_node_removal_timeout	Время ожидания в секундах, по истечении которого узел Pgpool-II будет удален из кластера
heartbeat_hostnameN	Задает TCP/IP-адрес узла N Pgpool-II, его DNS-имя или путь к директории с сокетом UNIX для локального подключения
heartbeat_portN	Порт проверки доступности узла N Pgpool-II
heartbeat_deviceN	Имя сетевого устройства для проверки доступности узла N Pgpool-II
hostnameN	Задает TCP/IP-адрес узла N Pgpool-II, его DNS-имя или путь к директории с сокетом UNIX для проверки состояния узла
pgpool_portN	Порт узла N Pgpool-II, на котором работает балансировщик нагрузки
wd_portN	Порт узла N Pgpool-II для мониторинга его состояния

Подробное описание настройки кластерной службы СУБД и балансировки нагрузки под управлением Pgpool-II приведено в 16.2.

10.2.1. Настройка аутентификации

Pgpool-II поддерживает следующие методы аутентификации: trust, reject, md5 и pam.

По умолчанию аутентификация в Pgpool-II выключена. Для ее включения необходимо в конфигурационном файле `/etc/pgpool2/pgpool.conf` для параметра `enable_pool_hba` установить значение `on`. Аутентификация в Pgpool-II может использоваться в совокупности с применением аутентификации на узлах СУБД для повышения информационной безопасности.

ВНИМАНИЕ! Методы аутентификации на узлах Pgpool-II и узлах СУБД должны быть идентичными.

Для настройки аутентификации используется конфигурационный файл `/etc/pgpool2/pool_hba.conf`, который имеет следующий синтаксис:

```
local DATABASE USER METHOD [OPTION]
host DATABASE USER CIDR-ADDRESS METHOD [OPTION]
```

Для настройки аутентификации по методу `md5` требуется выполнить следующие действия:

1) создать пароль для пользователя:

```
sudo pg_md5 --md5auth <пароль> --username <имя_пользователя>
```

2) установить метод аутентификации `md5` в конфигурационном файле `/etc/pgpool2/pool_hba.conf`:

```
host all all <сеть_подключения> md5
```

Пример

```
host all all 192.168.1.0/24 md5
```

3) перезапустить Pgpool-II:

```
sudo systemctl restart pgpool2
```

ВНИМАНИЕ! Если в процессе работы службы Pgpool-II был сменен пароль, то необходимо выполнить перезапуск службы.

10.2.2. Настройка регистрации событий

Для того чтобы настроить регистрацию событий в файл, необходимо выполнить следующие действия:

1) в конфигурационном файле `/etc/pgpool2/pgpool.conf` задать:

```
log_destination = 'syslog'
```

2) в конфигурационный файл `/etc/syslog-ng/syslog-ng.conf` добавить следующие строки:

```
destination pgpool2 { file("/var/log/pgpool.log"); };
filter f_pgpool2 { program("pgpool"); };
log { source(s_src); filter(f_pgpool2); destination(pgpool2);
    flags(final);};
```

3) перезапустить службу протоколирования `syslog-ng`:

```
sudo systemctl restart syslog-ng
```

4) перезапустить Pgpool-II:

```
sudo systemctl restart pgpool2
```

После перезапуска службы Pgpool-II регистрация событий будет выполняться в журнал `/var/log/pgpool.log`.

11. ОГРАНИЧЕНИЕ ПРОГРАММНОЙ СРЕДЫ В СУБД

11.1. Блокировка загрузки в адресное пространство СУБД неразрешенного программного обеспечения

Блокировка загрузки в адресное пространство СУБД программного обеспечения, не включенного в список разрешенного для выполнения, реализуется применением ЗПС (см. документ РУСБ.10015-01 97 01-1). При включенном режиме ЗПС:

- разрешена загрузка исполняемых файлов ПО и разделяемых библиотек, подписанных цифровой подписью;
- запрещена загрузка исполняемых файлов ПО и разделяемых библиотек без цифровой подписи, а также при нарушении их целостности (с неверной цифровой подписью).

В СУБД из состава ОС реализована возможность ограничивать местоположение для разделяемых библиотек, подключаемых к СУБД. Для запрета подключения разделяемых библиотек из недоверенных источников необходимо в конфигурационном файле `/etc/postgresql/<версия>/<имя_кластера_СУБД>/postgresql.conf` для параметра `ac_restrict_sh_libs_location` задать значение `true`:

```
ac_restrict_sh_libs_location = true
```

Если для параметра установлено значение `true`, то будет разрешено подключение разделяемых библиотек только из каталога `/usr/lib/postgresql/<версия>/lib` (является каталогом по умолчанию) и каталогов, указанных в параметре `dynamic_library_path`. Если для параметра установлено значение `false`, то подключение разделяемых библиотек будет возможно из любых каталогов, в том числе из недоверенных источников.

При включении параметра `ac_restrict_sh_libs_location` нельзя будет создавать расширения с помощью `CREATE EXTENSION`, если файл разделяемой библиотеки у этого расширения находится в каталоге отличном от `/usr/lib/postgresql/<версия>/lib` или каталогов, указанных в параметре `dynamic_library_path`. При установке данного запрета для исключения возможности несанкционированной загрузки библиотек в каталог по умолчанию или каталоги, указанные в параметре `dynamic_library_path`, необходимо дополнительно ограничить право пользователей СУБД добавлять в данные каталоги иные разделяемые библиотеки, которые потенциально могут быть вредоносными.

11.2. Запрет создания и модификации исполняемого кода при эксплуатации СУБД

Для запрета создания и модификации исполняемого кода при эксплуатации СУБД из состава ОС реализован режим киоска. При включении данного режима пользователям БД (в том числе суперпользователям СУБД) запрещается создание, изменение

и удаление процедур (PROCEDURE), функций (FUNCTION) и агрегатных функций (AGGREGATE). Кроме того, блокируется возможность прямого изменения записей системной таблицы pg_proc, которая хранит информацию о всех функциях и процедурах в БД. Для включения режима киоска необходимо в конфигурационном файле /etc/postgresql/<номер_версии>/<имя_кластера_СУБД>/postgresql.conf для параметра ac_kiosk_mode задать значение true:

```
ac_kiosk_mode = true
```

Если для параметра установлено значение true, то будет запрещено создание, изменение и удаление процедур (программного кода) для всех пользователей БД (в том числе для суперпользователей СУБД). Если для параметра установлено значение false, то создание процедур (программного кода), хранимых в базах данных, будут иметь только суперпользователи СУБД. В рамках ролевого управления доступом к категории суперпользователей СУБД относятся администраторы БД.

В СУБД из состава ОС запрет на создание процедур (программного кода), хранимых в базах данных, пользователям БД (пользователям информационной системы) реализуется при использовании ролевого управления доступом (см. раздел 6). Пользователи БД, которым назначена роль pg_database_user, не могут создавать процедуры.

11.3. Блокировка загрузки в адресное пространство СУБД программного обеспечения, целостность которого нарушена

Блокировка загрузки в адресное пространство СУБД программного обеспечения, целостность которого нарушена, реализуется применением ЗПС (см. документ РУСБ.10015-01 97 01-1), а также с помощью параметра, позволяющего запретить загрузку в адресное пространство СУБД внешнего ПО. При включенном режиме ЗПС:

- разрешена загрузка исполняемых файлов ПО и разделяемых библиотек, подписанных цифровой подписью;
- запрещена загрузка исполняемых файлов ПО и разделяемых библиотек без цифровой подписи, а также при нарушении их целостности (с неверной цифровой подписью).

Дополнительно в СУБД из состава ОС реализован параметр, который позволяет полностью запретить загрузку в адресное пространство СУБД внешнего ПО. Выполнение загрузки внешнего ПО доступно суперпользователю СУБД с помощью соответствующей команды:

```
COPY <имя_таблицы> FROM PROGRAM <команда_запуска_программы>
```

Реализация запрета загрузки в адресное пространство СУБД внешнего ПО осуществляется с помощью параметра ac_forbid_copy_from_program — необходимо в конфигурацион-

ном файле `postgresql.conf` для параметра `ac_forbid_copy_from_program` задать значение `true`:

```
ac_forbid_copy_from_program = true
```

Если для параметра установлено значение `true`, то будет установлен запрет на загрузку и выполнение ПО для всех пользователей системы (в том числе для суперпользователей СУБД). Если для параметра установлено значение `false`, то возможность загрузки и выполнения ПО в адресном пространстве СУБД будут иметь только суперпользователи СУБД.

11.4. Защита ядра и процессов пользователей при эксплуатации СУБД

Средства ограничения прав доступа к страницам памяти реализованы на основе стандартных возможностей ядра, а также набора изменений и особых параметров ядра ОС. Включенные параметры ядра ОС предоставляют защиту задачам ядра и процессам пользователей СУБД при доступе к страницам оперативной памяти: запрет записи в область памяти, помеченную как исполняемая. Гарантия того, что адреса с произвольным доступом не будут одновременно доступны на запись и выполнение, реализуется использованием возможностей системных вызовов `mmap()` и `mprotect()`. Подробное описание работы механизмов защиты памяти приведено в документе РУСБ.10015-01 97 01-1.

12. ОЧИСТКА ПАМЯТИ В СУБД

Процедура очистки памяти выполняется с использованием механизма очистки освобождаемой внешней памяти из состава ОС, который выполняет очистку неиспользуемых блоков ФС непосредственно при их освобождении. Подробное описание управления механизмом очистки памяти приведено в документе РУСБ.10015-01 97 01-1. Для гарантированного удаления БД и журналов, а также очистки оперативной памяти применяются средства очистки памяти ОС, которые обеспечивают удаление данных путем:

- многократной перезаписи уничтожаемых (стираемых) объектов файловой системы специальными битовыми последовательностями;
- многократной перезаписи уничтожаемых (стираемых) объектов файловой системы случайной битовой псевдопоследовательностью.

В СУБД дополнительно реализована возможность удаления объектов доступа БД путем перезаписи удаленных или модифицированных участков файловой системы. Для управления механизмами удаления используются параметры конфигурационного файла `postgresql.conf`, описание которых приведено в таблице 10.

Таблица 10

Параметр	Описание
`ac_wipe_files`	Включает (`true`) или выключает (`false`) механизм очистки файлов, позволяющий удалять объекты доступа БД с предварительной перезаписью участков объектов файловой системы
`ac_wipe_pages`	Включает (`true`) или выключает (`false`) механизм очистки страниц памяти файловой системы, позволяющий удалять объекты доступа БД путем перезаписи модифицированных участков объектов файловой системы при выполнении операции обновления и/или удаления

Для гарантированного удаления хранящихся в оперативной памяти данных, которые могут содержать чувствительную информацию, используется механизм очистки и перезаписи высвобождаемой памяти. При включении данного механизма оставшиеся в оперативной памяти данные затираются случайной битовой последовательностью. Управление механизмом осуществляется с помощью параметра `ac_rand_free_memory`, для его включения необходимо в конфигурационном файле `postgresql.conf` для параметра задать значение `true`:

```
ac_rand_free_memory = true
```

Перезапись уничтожаемых объектов случайной битовой последовательностью позволяет гарантировать, что перед возвратом используемой СУБД памяти обратно в ОС освобожденная память не будет содержать читаемых остаточных данных.

13. РЕГИСТРАЦИЯ СОБЫТИЙ В СУБД

13.1. Регистрация событий средствами ОС

В СУБД регистрация событий безопасности выполняется с учетом требований ГОСТ Р 59548-2022. Регистрация событий безопасности, настройка реагирования системы на события и информирование администратора осуществляется подсистемой регистрации событий из состава ОС. Описание подсистемы регистрации событий и журнала событий приведено в РУСБ.10015-01 97 01-1.

Подсистема регистрации событий собирает информацию о событиях безопасности, связанных с функционированием СУБД и действиями пользователей СУБД, и обеспечивает их хранение в журнале событий безопасности `/parsec/log/astra/events`, а также предоставляет инструменты для просмотра собранных данных и реагирования на события.

Для настройки событий безопасности СУБД используются графическая утилита `fly-admin-events` («Настройка регистрации системных событий») и инструмент командной строки `astra-admin-events`.

Оповещение администратора СУБД, администратора БД (администратора информационной системы) о событиях безопасности реализуется службой `astra-event-watcher`. Настройка уведомлений выполняется с использованием графической утилиты `fly-admin-event`.

13.2. Встроенные средства регистрации событий в СУБД

13.2.1. Режимы регистрации событий

В СУБД для настройки режима регистрации событий используется конфигурационный параметр `ac_audit_mode` файла `postgresql.conf`. Этот параметр может быть изменен только перезапуском сервера. Параметр может принимать следующие значения:

- `internal` — для настройки регистрации событий используются соответствующие команды SQL, а настройки хранятся в таблице `pg_db_role_settings`;
- `external` — для настройки регистрации событий используется внешний файл `pg_audit.conf`;
- `external,internal` — смешанный режим. Настройки регистрации событий берутся сначала из внешнего файла `pg_audit.conf`, после чего дополняются настройками из таблицы `pg_db_role_settings`;
- `internal,external` — смешанный режим. Настройки регистрации событий берутся сначала из таблицы `pg_db_role_settings`, после чего дополняются настройками из внешнего файла `pg_audit.conf`;
- `none` — регистрация событий отключена.

13.2.2. Настройка маски регистрации событий

В СУБД маска регистрации событий устанавливается в процессе авторизации пользователя согласно выбранному режиму регистрации событий и находится в атрибуте сессии `ac_session_audit`.

При этом реализован следующий порядок применения настроек регистрации событий:

1) настройки для конкретной роли и конкретной базы данных;
2) настройки для конкретной роли;
3) настройки для конкретной базы данных;
4) для всех остальных.

Маска регистрации событий имеет вид `{УСПЕХ:ОТКАЗ}`, где `УСПЕХ` — список успешных событий, `ОТКАЗ` — список неуспешных событий. Она может быть задана с помощью буквенных кодов или с помощью шестнадцатеричного числа. Вывод маски производится в текстовом виде.

В таблице 11 приведено соответствие между событиями, буквенным и шестнадцатеричным значением маски регистрации событий.

Таблица 11

Событие	Символ	Шестнадцатеричное значение	Описание
SUBJECT	S	1	Добавление/изменение/удаление пользователей и групп
CONFIGURATION	s	2	Изменение конфигурации, влияющей на доступ к данным (запрос на изменение значения переменной `ac_session_maclabel`)
RIGHTS	R	4	Запрос на модификацию прав доступа к объектам БД
CHECK_RIGHTS	V	8	Проверка прав доступа
SELECT	r	10	Выборка информации из БД
INSERT	a	20	Добавление информации в БД
UPDATE	w	40	Изменение информации в БД
DELETE	d	80	Удаление информации из БД
TRUNCATE	D	100	Очистка данных
REFERENCES	x	400	Задание столбца таблицы в качестве внешнего ключа
TRIGGER	t	800	Добавление триггера к таблице
EXECUTE	X	1000	Запуск хранимой процедуры или триггера
USAGE	U	2000	Использование объекта БД

Окончание таблицы 11

Событие	Символ	Шестнадцатеричное значение	Описание
CREATE	C	10000	Создание объектов в БД
CREATE TEMP	T	20000	Создание временного объекта
DROP	E	40000	Удаление объектов БД
ALTER	M	80000	Изменение объекта БД
CONNECT	c	40000000	Запрос на начало сессии
DISCONNECT	e	80000000	Запрос на окончание сессии
Зарезервированный символ	*	C00F3DFF	Полная маска регистрации событий
Зарезервированный символ	O	0	Регистрация событий отключена

Атрибут сессии ac_session_audit может быть изменен только администратором с помощью команды SET:

```
SET ac_session_audit TO 'новое_значение';
```

и просмотрен с помощью команды:

```
SHOW ac_session_audit;
```

Для просмотра маски сессии используется следующая команда:

```
SELECT session_audit;
```

Для просмотра текущей маски используется следующая команда:

```
SELECT current_audit;
```

Для конвертации маски регистрируемых событий из текстового (буквенного) в шестнадцатеричное значение и из шестнадцатеричного в текстовое (буквенное) используются SQL-функции text_to_auditmask(TEXT) и auditmask_to_text(TEXT) соответственно.

Пример

```
SELECT text_to_auditmask('{ace:ce}');
text_to_auditmask
-------------------------
```

```
{0xC0000020:0xC0000000}
(1 строка)

SELECT auditmask_to_text('{0xC0000020:0xC0000000}');
auditmask_to_text
------------------
{ace:ce}
(1 строка)
```

13.2.3. Назначение списков регистрации событий в режиме `internal`

Для назначения маски событий в режиме `internal` используется команда `ALTER ROLE`:

```
ALTER ROLE { ALL | имя_роли } [ IN DATABASE имя_базы_данных ] SET
ac_session_audit TO новое_значение;
```

Для удаления списка регистрации событий используется следующая команда:

```
ALTER ROLE { ALL | имя_роли } [ IN DATABASE имя_базы_данных ] RESET
ac_session_audit;
```

При модификации маски происходит автоматическое обновление атрибута `ac_session_audit`.

Примечание. Для выполнения приведенных команд требуются права администратора.

Примечание. При инициализации кластера баз данных автоматически добавляются следующие правила:

```
ALTER ROLE postgres SET ac_session_audit TO '{SsRawdCTEMce:ce}';
ALTER ROLE ALL SET ac_session_audit TO '{SsRDxCTEMce:SsRVrawdDxtXUCTEMce}';
```

13.2.4. Назначение списков регистрации событий в режиме `external`

Для назначения маски событий в режиме `external` используется конфигурационный файл `pg_audit.conf` конкретного кластера данных, который имеет следующий формат:

```
success events mask = значение failure events mask = значение user =
имя_пользователя database = имя_базы_данных
success events mask = значение failure events mask = значение user =
имя_пользователя
```

```
success events mask = значение failure events mask = значение
```

Пример

Файл pg_audit.conf

- аудит действий администратора СУБД:

```
success events mask = F00E7 failure events mask = 0
     user = postgres
```

- для пользователя any выполнять регистрацию только неуспешных действий:

```
success events mask = 0 failure events mask = FFFFF user = any
```

- для всех остальных пользователей выполнять регистрацию всех неуспешных действий и всех успешных действий, кроме доступа к данным:

```
success events mask = F0707 failure events mask = FFFFF
```

В конфигурационном файле задаются списки успешных (success events mask) и неуспешных (failure events mask) типов запросов на доступ, которые будут регистрироваться в журнале СУБД и журнале аудита ОС для отдельных пользователей и по умолчанию. Списки типов запросов на доступ задаются в виде шестнадцатеричных чисел, в которых каждому типу запроса соответствует установленный (для регистрируемых запросов) или сброшенный (для не регистрируемых запросов) бит. Типы запросов и их описание приведены в таблице 12.

Таблица 12

Тип запроса	Описание	Бит	Шестнадцатеричное значение
SUBJECT	Добавление/изменение/удаление пользователей и групп	0	1
CONFIGURATION	Изменение конфигурации, влияющей на доступ к данным (запрос на изменение значения переменной ac_session_maclabel)	1	2
RIGHTS	Изменение прав доступа к объектам БД	2	4
CHECK_RIGHTS	Модификация прав доступа к объектам БД	3	8
SELECT	Выборка информации из БД	4	10
INSERT	Добавление информации в БД	5	20
UPDATE	Изменение информации в БД	6	40
DELETE	Удаление информации из БД	7	80
TRUNCATE	Очистка данных	8	100
REFERENCES	Задание столбца таблицы в качестве внешнего ключа	10	400

Окончание таблицы 12

Тип запроса	Описание	Бит	Шестнадцатеричное значение
TRIGGER	Добавление триггера к таблице	11	800
EXECUTE	Запуск хранимой процедуры или триггера	12	1000
USAGE	Использование объекта БД	13	2000
CREATE	Создание объектов в БД	16	10000
CREATE_TEMP	Создание временных объектов в БД	17	20000
DROP	Удаление объектов БД	18	40000
ALTER	Изменение объекта БД	19	80000
CONNECT	Соединение пользователя с БД	30	40000000
DISCONNECT	Разъединение пользователя с БД	31	80000000

Информация о соединении пользователей с БД (CONNECT) и разъединении с ней (DISCONNECT) регистрируется всегда, при условии, что список событий не установлен в 0.

Примечание. Любые изменения этого файла будут применены только при перезапуске сервера.

13.2.5. Назначение списков регистрации событий в режимах external, internal и internal,external

Загрузка маски регистрации событий в режиме external,internal двухэтапная: сначала выполняется загрузка маски регистрации событий из файла, после чего дополняется настройками из pg_db_role_settings, если в pg_db_role_settings есть более точные настройки. Для изменения маски регистрации событий сессии могут быть использованы команды из 13.2.3.

Аналогично и для режима internal,external.

13.2.6. Назначение списков регистрации событий в режиме none

В режиме none регистрация событий отключена, но администратор может изменять маску регистрации событий с помощью команд из 13.2.3.

14. ВОССТАНОВЛЕНИЕ СУБД ПОСЛЕ СБОЕВ И ОТКАЗОВ

Во избежание потерь данных должны регулярно создаваться резервные копии БД в СУБД. В случае возникновения ошибок в хранящихся данных, нарушения целостности или в случае программного и/или аппаратного сбоя сервера БД необходимо проведение процедуры восстановления БД. При этом, в зависимости от тяжести повреждений может осуществляться как сохранение существующего кластера БД с последующим его восстановлением, так и восстановление из резервных копий, созданных в процессе регулярного проведения регламентных работ.

Резервное копирование конфигурации СУБД выполняется встроенными средствами резервного копирования и восстановления из состава ОС (например, с использованием утилит архивирования и копирования: `tar`, `cpio`, `gzip`, `cp`, см. документ РУСБ.10015-01 95 01-1).

В СУБД проверка целостности резервной копии в процессе восстановления осуществляется путем контроля подписи в расширенных атрибутах при использовании ЗПС. Описание работы ЗПС приведено в документе РУСБ.10015-01 97 01-1. В дальнейшем контроль восстановленных данных БД (объектов, установленных на контроль) осуществляется применением контроля целостности СУБД.

В СУБД существуют три подхода к резервному копированию данных:

- SQL-дамп;
- резервное копирование на уровне ФС;
- непрерывное архивирование.

Более подробное описание этих методов и процедур копирования и восстановления приведено в документации на СУБД в пакете `postgresql-doc-x.x`.

СУБД содержит ряд стандартных средств резервного копирования и восстановления БД. К ним относятся инструменты командной строки `pg_dump` (см. 14.2), `pg_dumpall` (см. 14.3), `pg_restore` (см. 14.4) и интерактивный терминал `psql`, с помощью которого могут быть восстановлены резервные копии, сохраненные в виде сценария SQL.

14.1. Создание и восстановление резервных копий баз данных с мандатными атрибутами

Для создания и восстановления резервных копий баз данных с мандатными атрибутами необходимо, чтобы пользователь имел привилегии `PARSEC_CAP_SETMAC`, `PARSEC_CAP_CHMAC`.

В случае создания резервной копии необходимо назначить максимальную метку безопасности на каталог, в который будет выгружена база данных (для назначения метки безопасности на файл копии).

В случае восстановления, помимо указанных привилегий, требуются права администратора БД (для создания объектов и назначения мандатных атрибутов).

Для восстановления копии базы данных с мандатными атрибутами в другой базе данных необходимо:

1) назначить максимальную метку безопасности на каталог, в который будет проводиться выгрузка резервной копии;

2) создать резервную копию исходной базы данных от имени пользователя с привилегиями PARSEC_CAP_SETMAC и PARSEC_CAP_CHMAC;

3) на целевом кластере назначить мандатные атрибуты на кластер;

4) восстановить резервную копию базы данных с помощью psql (если резервная копия сделана в текстовом виде) или с помощью pg_restore (если резервная копия сделана в виде упакованного файла) от администратора БД с привилегиями PARSEC_CAP_SETMAC и PARSEC_CAP_CHMAC.

Примечания:

1. Инструмент командной строки pg_dump, поставляемый вместе с СУБД, выгружает команды по установке комментариев, меток безопасности и назначению мандатных атрибутов в следующем виде:

```
COMMENT ON DATABASE CURRENT_DATABASE IS 'комментарий';
SECURITY LABEL ON DATABASE CURRENT_DATABASE IS '...';
MAC LABEL ON DATABASE CURRENT_DATABASE IS '...';
MAC CCR ON DATABASE CURRENT_DATABASE IS '...';
```

Такая резервная копия может быть восстановлена с установкой комментариев, меток безопасности и мандатных атрибутов в желаемую базу данных.

2. Для восстановления резервных копий баз данных, сделанных в предыдущих версиях, в СУБД актуальной версии необходимо установить параметр ac_auto_adjust_macs = true;

3. Для переноса кластера с предыдущих версий СУБД на актуальную версию необходимо воспользоваться инструментами командной строки pg_upgradecluster или pg_upgrade (см. 8.5 и 8.6 соответственно).

14.2. pg_dump

Для создания резервной копии БД используется инструмент командной строки pg_dump. Инструмент позволяет создать резервную копию даже во время использования БД, при этом доступ к ней других пользователей (имеющих права доступа как на чтение, так и на запись) не блокируется.

Инструмент pg_dump создает копию только одной БД, при этом информация о ролях или табличных пространствах не сохраняется (эта информация относится ко всему кластеру, а не к отдельной БД). Для создания резервной копии всего содержимого кластера следует использовать инструмент pg_dumpall (см. 14.3).

Резервная копия может создаваться в виде сценария или в форматах упакованного файла. Сценарий резервной копии представляет собой текст, содержащий последовательность SQL-команд, необходимых для воссоздания БД до состояния, в котором она была сохранена.

Синтаксис команды:

```
pg_dump [<параметр>]... [<база_данных>]
```

Описание основных параметров pg_dump приведено в таблице 13.

Таблица 13

Параметр	Описание
-f, --file=FILENAME	Имя выходного файла. Если не используется, то SQL-сценарий будет направлен в стандартный вывод
-F, --format=c|t|p	Формат выходного файла (пользовательский, tar, текстовый)
-v, --verbose	Режим вывода всех сообщений
-Z, --compress=0-9	Уровень сжатия для форматов сжатия
--lock-wait-timeout=TIMEOUT	Завершение ошибкой после ожидания TIMEOUT для блокировки таблицы
--disable-macs	Исключить из выгрузки вывод команд MAC LABEL и MAC CCR
--help	Вывести справку и выйти

При использовании инструмента pg_dump установка соединения с БД осуществляется с помощью параметров, приведенных в таблице 8.

Резервная копия не содержит информации о статистике, которую использует оптимизатор запросов, поэтому после восстановления из резервной копии рекомендуется выполнять команду ANALYZE для достижения лучшей производительности. Резервная копия также не содержит команд вида:

```
ALTER DATABASE ... SET
```

Эти команды могут быть добавлены в резервную копию инструментом pg_dumpall вместе с информацией о пользователях и других глобальных параметров установки.

Примеры:

1. Создание резервной копии БД mydb в виде SQL-сценария:

```
pg_dump mydb > db.sql
```

2. Загрузка SQL-сценария в новую БД newdb:

```
psql -d newdb -f db.sql
```

3. Создание резервной копии всех схем, начинающихся с east или west и заканчивающихся на gsm, исключая все схемы, содержащие слово test:

```
pg_dump -n 'east*gsm' -n 'west*gsm' -N '*test*' mydb > db.sql
```

Для восстановления из формата упакованного файла используется инструмент pg_restore (см. 14.4). Инструмент позволяет просмотреть и/или выбрать, что именно восстанавливать, и изменить порядок элементов перед восстановлением.

Для восстановления из сценария резервной копии данный сценарий подается на вход инструмента командной строки psql:

```
psql <база_данных> <сценарий>
```

Сценарий может быть использован для воссоздания БД на другом сервере или архитектуре и с небольшими изменениями на других СУБД.

Информация о способах использования инструмента командной строки pg_dump также приведена в man pg_dump и man psql.

14.3. pg_dumpall

Инструмент командной строки pg_dumpall используется для создания резервной копии каждой БД кластера, а также сохранения глобальных объектов, единых для всех БД. Данные объекты включают в себя информацию о пользователях и группах, правах доступа, применяемых для всех БД в целом.

Резервная копия сохраняется в виде сценария, содержащего SQL-команды для дальнейшего восстановления.

В процессе создания резервной копии pg_dumpall сначала выполняет команды для воссоздания ролей, табличных пространств и пустых БД, затем осуществляет последовательный вызов pg_dump для каждой БД кластера. В данном случае каждая БД будет обладать

внутренней целостностью, но при этом «снимки» различных БД могут не быть полностью синхронизированы.

При создании резервной копии в качестве стартовой БД возможно указание любого имени, но при загрузке данных в пустой кластер, как правило, требуется указание `postgres`.

Синтаксис команды:

```
pg_dumpall [параметр]
```

Основные параметры инструмента `pg_dumpall` приведены в таблице 14.

Таблица 14

Параметр	Описание
`-f, --file=FILENAME`	Имя выходного файла
`--lock-wait-timeout=TIMEOUT`	Завершение ошибкой после ожидания `TIMEOUT` для блокировки таблицы
`--help`	Вывести справку и выйти

Информация о способах использования инструмента командной строки `pg_dumpall` также приведена в `man pg_dumpall` и `man psql`.

Параметры установки соединения приведены в таблице 8.

Для восстановления резервной копии из сценария, полученного с помощью `pg_dumpall`, необходимо использовать инструмент командной строки `psql`.

Восстановление из резервной копии выполняется с правами администратора БД, поскольку они требуются для восстановления информации о ролях и табличных пространствах. При использовании табличных пространств следует убедиться, что пути табличных пространств из резервной копии соответствуют новой конфигурации.

При восстановлении из резервной копии не имеет значения, с какой БД было осуществлено соединение, т. к. созданный с помощью `pg_dumpall` сценарий содержит соответствующие команды для соединения с сохраненными БД.

Примеры:

1. Создание резервной копии всех БД

```
$ pg_dumpall > db.out
```

2. Восстановление сохраненных БД

```
$ psql -f db.out postgres
```

Резервная копия не содержит информации о статистике, которую использует оптимизатор запросов, поэтому после восстановления из резервной копии рекомендуется выполнять команду ANALYZE для каждой БД для достижения лучшей производительности. Резервная копия также не содержит команд вида:

```
ALTER DATABASE ... SET
```

14.4. pg_restore

Для восстановления резервных копий БД, созданных в виде упакованных файлов с помощью инструмента командной строки pg_dump, используется инструмент командной строки pg_restore. Он выполняет команды, необходимые для воссоздания БД до состояния на момент времени создания резервной копии. Инструмент pg_restore позволяет выбрать, что именно восстанавливать из резервной копии, а также менять порядок восстанавливаемых элементов. Файлы резервных копий являются переносимыми между разными архитектурами.

Инструмент командной строки pg_restore может функционировать в двух режимах. При указании БД резервная копия восстанавливается непосредственно в нее. В другом случае, сценарий, содержащий необходимые для пересоздания БД SQL-команды, создается и выводится в файл или стандартный поток вывода. Результирующий сценарий эквивалентен формату текстового вывода инструмента командной строки pg_dump. Вследствие этого некоторые параметры, управляющие выводом, аналогичны параметрам pg_dump (см. 14.2).

Синтаксис команды:

```
pg_restore [параметр] [файл]
```

Основные параметры инструмента pg_restore приведены в таблице 15.

Таблица 15

Параметр	Описание
-d, --dbname=ИМЯ	Подсоединиться к указанной БД
-f, --file=FILENAME	Имя входного файла
-F, --format=c|t	Формат файла резервной копии (следует указывать, когда необходимо выбрать файл определенного формата; в остальных случаях рекомендуется не указывать, при этом формат будет выбран автоматически)
-l, --list	Напечатать итоговое оглавление архива

Окончание таблицы 15

Параметр	Описание
`-v, --verbose`	Режим вывода всех сообщений
`--help`	Вывести справку и выйти
`--version`	Вывести информацию о версии и выйти

Информация о способах использования инструмента командной строки `pg_restore` также приведена в `man pg_restore`.

Примеры:

1. Создание резервной копии БД `mydb` в пользовательском формате (`c`, custom):

```
pg_dump -Fc mydb > db.dump
```

2. Удаление БД и восстановление ее из резервной копии:

```
dropdb mydb
pg_restore -C -d postgres db.dump
```

Для параметра `-d` может быть указана любая БД кластера. Утилита `pg_restore` использует ее только для выполнения команды `CREATE DATABASE`. С параметром `-C` данные всегда восстанавливаются в БД, указанную в резервной копии.

3. Загрузка резервной копии в новую БД `newdb`:

```
createdb -T template0 newdb
pg_restore -d newdb db.dump
```

Параметр `-C` не был использован, вместо этого осуществлялось подключение непосредственно к восстанавливаемой БД. Новая БД была создана из шаблона `template0`, а не из `template1`, для обеспечения первоначальной чистоты базы.

При использовании инструмента `pg_restore` установка соединения с БД осуществляется с помощью параметров, приведенных в таблице 8.

15. ОБНОВЛЕНИЕ СУБД ПРИ СОХРАНЕНИИ ЕЕ ДОСТУПНОСТИ

В кластере СУБД реализована возможность поочередного обновления, связанного с устранением уязвимостей, узлов СУБД при сохранении их доступности. При неуспешном обновлении СУБД в кластере обеспечивается возможность возврата к ее предыдущему состоянию. Данное действие не приводит к прерыванию работы кластера СУБД.

Для выполнения обновления кластера с сохранением доступности СУБД необходимо:

1) на каждом узле кластера СУБД настроить синхронную репликацию данных и балансировку нагрузки путем применения Pgpool-II (см. 16.2);

2) исключить из Pgpool-II один ведомый узел СУБД, для этого:

а) в конфигурационном файле `/etc/pgpool2/pgpool.conf` для прекращения использования узла установить `0` в качестве значения параметра `backend_weight<*>`:

```
backend_weight<*> = 0
```

б) перезагрузить конфигурацию узла в Pgpool-II:

```
/usr/sbin/pcp_reload_config -h <адрес_узла_pgpool>
-p <порт_узла_pgpool> -U <имя_администратора_БД> --scope=cluster
```

в) осуществить непосредственное исключение узла из Pgpool-II следующей командой:

```
/usr/sbin/pcp_detach_node -h <адрес_узла_pgpool>
-p <порт_узла_pgpool> -U <имя_администратора_БД> node_id=<*>
```

3) обновить СУБД отключенного узла путем установки пакета новой версии `postgresql-<номер_версии>`:

```
sudo apt install postgresql
```

В случае неуспешного обновления выполнить установку пакета предыдущей версии `postgresql-<номер_версии>`;

4) для возобновления использования узла необходимо в конфигурационном файле `/etc/pgpool2/pgpool.conf` вернуть значение параметра `backend_weight<*>` в исходное состояние:

```
backend_weight<*> = <исходное_значение>
```

Пример

```
backend_weight0 = 1
```

5) в Pgpool-II включить узел после выполненного обновления:

```
/usr/sbin/pcp_recovery_node -h <адрес_узла_pgpool>
```

```
-p <порт_узла_pgpool> -U <имя_администратора_БД> node_id=<*>
```

6) перезагрузить конфигурацию узла в Pgpool-II:

```
/usr/sbin/pcp_reload_config -h <адрес_узла_pgpool>
-p <порт_узла_pgpool> -U <имя_администратора_БД> --scope=cluster
```

7) выполнить действия пунктов перечисления 2) – 6) на всех ведомых узлах;

8) сделать ведущим узлом СУБД один из обновленных ведомых узлов:

```
/usr/sbin/pcp_promote_node -h <адрес_узла_pgpool>
-p <порт_узла_pgpool> -U <имя_администратора_БД> --switchover
```

9) обновить узел, бывший ведущим, путем установки пакета новой версии `postgresql-<номер_версии>` и включить его в качестве ведомого:

```
sudo apt install postgresql
/usr/sbin/pcp_recovery_node -h <адрес_узла_pgpool>
-p <порт_узла_pgpool> -U <имя_администратора_БД> node_id=<*>
```

16. СРЕДСТВА ОБЕСПЕЧЕНИЯ ОТКАЗОУСТОЙЧИВОСТИ И ВЫСОКОЙ ДОСТУПНОСТИ СУБД

Для функционирования СУБД в отказоустойчивом кластере используются встроенные в ОС средства кластеризации и высокой доступности, а также средство из состава СУБД — прокси-сервер Pgpool-II.

Кластер обеспечивает доступность СУБД за счет одновременного функционирования нескольких ее экземпляров на разных узлах. В этом случае один узел назначается ведущим, а другой (другие) — ведомым. При выходе из строя ведущего узла ведомый узел будет назначен ведущим автоматически.

Прокси-сервер Pgpool-II помимо обеспечения высокой доступности, позволяет осуществить балансировку нагрузки между узлами кластера, обеспечить синхронную репликацию данных на множество узлов и другие функции (см. 16.2)

16.1. Доступность и отказоустойчивость в кластере pacemaker

Настройка кластерной службы СУБД выполняется на развернутом кластере `pacemaker`. Настройка кластера `pacemaker` приведена в документе РУСБ.10015-01 95 01-1. Узлам кластера и самому кластеру необходимо назначить IP-адреса из одной подсети. В кластере `pacemaker` необходимо настроить отказоустойчивую кластерную службу СУБД с репликацией данных между узлами.

16.1.1. Установка СУБД

Установка СУБД выполняется на каждом узле кластера `pacemaker`. Установка осуществляется с помощью команды:

```
sudo apt install postgresql
```

При установке СУБД будет создан кластер СУБД (не является кластером `pacemaker`) с именем `main`.

Для работы с поддержкой мандатного управления доступом следует использовать агент (набор предопределенных параметров) `ocf:astra:pgsql`. Для использования агента необходимо установить пакет `astra-resource-agents`:

```
sudo apt install astra-resource-agents
```

16.1.2. Настройка СУБД

Настройка СУБД осуществляется путем корректировки конфигурационных файлов. Для этого необходимо на каждом узле:

1) в конфигурационном файле СУБД `/etc/postgresql/<номер_версии>/<имя_кластера_СУБД>/postgresql.conf` установить значения параметров:

```
listen_addresses = '*'
wal_level = replica
wal_keep_segments = 32
```

где `listen_addresses` — интерфейсы, на которые СУБД будет принимать подключения ('*' — все интерфейсы);
`wal_level` — информация, которая будет записываться в журнал предзаписи Write-Ahead Logging (для репликации необходимо установить значение `replica`);
`wal_keep_segments` — количество файлов сегментов журнала, сохраняемых в каталоге `pg_wal`, чтобы выбрать их на ведомом узле для репликации данных (1 файл сегмента имеет размер 16 МБ)

2) в конфигурационный файл `/etc/postgresql/<номер_версии>/<имя_кластера>/pg_hba.conf` добавить строки:

```
host replication postgres 192.168.23.101/32 trust
host replication postgres 192.168.23.102/32 trust
```

где 192.168.23.101/32 и 192.168.23.102/32 - IP-адреса узлов ранее настроенного кластера `pacemaker`

3) перезапустить службу `postgresql` для вступления в силу внесенных изменений:

```
sudo systemctl restart postgresql
```

Следует определить какой узел будет использоваться в качестве ведущего (`master`), а какой — в качестве ведомого (`slave`), после чего необходимо провести синхронизацию узлов кластерной службы СУБД и создать резервную копию ведущего узла (см. 16.1.3).

16.1.3. Синхронизация узлов кластерной службы СУБД

Синхронизация узлов кластерной службы СУБД предназначена для поддержания согласованности их работы. Для синхронизации данных между узлами кластерной службы СУБД необходимо на ведомом узле (на всех ведомых узлах):

1) от имени пользователя `postgres` удалить содержимое кластера СУБД:

```
sudo -u postgres sh -c "rm -rf /var/lib/postgresql/*/main/*"
```

2) от имени пользователя `postgres` создать резервную копию ведущего узла. Обращение к ведущему узлу будет выполняться по его адресу. Созданная резервная копия будет использована для создания локальной копии базы данных.
Для создания резервной копии выполнить команду:

```
sudo -u postgres pg_basebackup -h <ip-адрес_ведущего_узла_кластера_СУБД>
-D /var/lib/postgresql/<номер_версии>/<имя_кластера> -P
```

3) для исключения конфликтов между кластером СУБД и службой `systemd` необходимо отключить на всех узлах (включая ведущий) автоматический запуск службы `postgresql` при перезагрузке ОС и остановить службу `postgresql`:

```
sudo systemctl disable postgresql
sudo systemctl stop postgresql
```

16.1.4. Создание кластерного ресурса

Для создания кластерного ресурса необходимо добавить кластерную службу `postgresql` как ресурс с именем `pgsql`. Для управления ресурсом в кластерной среде используется агент, который позволяет обеспечить запуск, остановку, мониторинг и другие функции службы `postgresql` путем применения предопределенных параметров. При работе в СУБД с поддержкой мандатного управления доступом следует использовать агент (набор предопределенных параметров) `ocf:astra:pgsql`, в случае если поддержка мандатного управления доступом не требуется — использовать агент `ocf:heartbeat:pgsql`.

Для создания кластерного ресурса необходимо на ведущем узле:

1) определить будет ли использоваться поддержка мандатного управления доступом и в зависимости от этого выполнить один из следующих вариантов настройки:

а) для работы с поддержкой мандатного управления доступом выполнить команду создания ресурса и указать имя создаваемого ресурса `pgsql`, а также указать использование агента `ocf:astra:pgsql`:

```
sudo pcs resource create pgsql ocf:astra:pgsql \
pgversion="<номер_версии>" \
rep_mode="sync" \
master_ip="192.168.23.100" \
node_list="pcmk-1 pcmk-2"
```

Подробное описание параметров см. `man pcs` и `man ocf_astra_pgsql`;

б) для работы без поддержки мандатного управления доступом выполнить команду создания ресурса, указать имя создаваемого ресурса `pgsql` , а также указать использование агента `ocf:heartbeat:pgsql`. При этом дополнительно необходимо определить параметры `pgctl`, `pgdata`, `config`, `socketdir` и `op monitor interval` создаваемого ресурса:

```
sudo pcs resource create pgsql ocf:heartbeat:pgsql \
```

```
pgctl="/usr/lib/postgresql/<номер_версии>/bin/pg_ctl" \
pgdata="/var/lib/postgresql/<номер_версии>/main" \
config="/etc/postgresql/<номер_версии>/main/postgresql.conf" \
socketdir="/var/run/postgresql/" \
rep_mode="sync" \
master_ip="192.168.23.100" \
node_list="pcmk-1 pcmk-2" \
op monitor interval=1min
```

Подробное описание параметров см. `man pcs` и `man ocf_astra_pgsql`;

2) установить параметры службы `postgresql`:

- объявить ресурс `pgsql` продвигаемым (promotable) и создать вторичный ресурс, идентичный ресурсу `pgsql`:

```
sudo pcs resource promotable pgsql promoted-max=1
promoted-node-max=1 clone-max=2 clone-node-max=1 notify=true
```

Продвигаемый и вторичный ресурсы могут переключаться между двумя режимами работы - master и slave. В к оманде явно не указано имя вторичного ресурса, в этом случае будет автоматически использовано имя вида `<имя_ресурса>-clone` (например, `pgsql-clone`);

- установить связь между продвигаемым и вторичным ресурсами кластера и задать возможность размещения ресурсов на одном узле кластера `pacemaker`:

```
sudo pcs constraint colocation add ClusterIP with master
pgsql-clone INFINITY
```

где `constraint colocation` — задает возможность размещения на одном узле кластера `pacemaker`;
`add ClusterIP with master pgsql-clone` — устанавливает связь между продвигаемым и вторичным ресурсами;

3) запустить кластер `pacemaker`:

```
sudo pcs cluster start --all
```

4) проверить статус службы `pcs`:

```
sudo pcs status
```

Результат выполнения команды:

```
Cluster name: astracluster
Stack: corosync
Current DC: pcmk-1 (version 2.0.1-9e909a5bdd) - partition with quorum
Last updated: Sat Mar 11 18:40:21 2023
Last change: Sat Mar 11 18:39:28 2023 by root via crm_attribute
on pcmk-1
```

```
2 nodes configured
3 resources configured

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-1
Clone Set: pgsql-clone [pgsql] (promotable)
Masters: [ pcmk-1 ]
Slaves: [ pcmk-2 ]

Daemon Status:
corosync: active/disabled
pacemaker: active/disabled
pcsd: active/enabled
```

Если команда проверки статуса службы сообщит об ошибке:

```
exitreason='Replication(rep_mode=async or sync) requires
Master/Slave configuration.'
```

то для устранения ошибки необходимо остановить и запустить кластер:

```
sudo pcs cluster stop --all
sudo pcs cluster start --all
```

16.1.5. Восстановление неработоспособного узла

При нарушении процесса функционирования узла потребуется процедура его восстановления. Для восстановления неработоспособного узла необходимо выполнить на нем следующие действия:

1) выполнить синхронизацию данных с действующим узлом (см.16.1.3)
2) удалить файл `/var/lib/pgsql/tmp/PGSQL.lock`:

```
sudo rm /var/lib/pgsql/tmp/PGSQL.lock
```

3) запустить кластер pacemaker:

```
sudo pcs cluster start
```

После запуска кластера `pacemaker` неработоспособный узел будет восстановлен и получит статус ведомого.
Результат выполнения команды:

```
Cluster name: astracluster
Stack: corosync
```

```
Current DC: pcmk-2 (version 2.0.1-9e909a5bdd) - partition with quorum
Last updated: Tue Apr 11 14:13:34 2023
Last change: Tue Apr 11 14:05:24 2023 by root via crm_attribute on
pcmk-2

2 nodes configured
3 resources configured

Online: [ pcmk-1 pcmk-2 ]

Full list of resources:

ClusterIP (ocf::heartbeat:IPaddr2): Started pcmk-2
Clone Set: pgsql-clone [pgsql] (promotable)
Masters: [ pcmk-2 ]
Slaves: [ pcmk-1 ]

Daemon Status:
corosync: active/disabled
pacemaker: active/disabled
pcsd: active/enabled
```

16.2. Настройка кластерной службы СУБД и балансировка нагрузки под управлением Pgpool-II

Для обеспечения отказоустойчивости под управлением Pgpool-II в кластере должно одновременно функционировать несколько экземпляров СУБД (как минимум — два). С целью обеспечения балансировки нагрузки необходимо использовать Pgpool-II, количество узлов которого должно составлять минимум три (для функционирования кворума). Для настройки репликации данных и балансировки нагрузки под управлением Pgpool-II необходимо:

1) в конфигурационном файле `/etc/postgresql/<версия>/<имя_кластера>/postgresql.conf` на каждом узле СУБД задать следующие параметры:

```
max_wal_senders = 5
max_replication_slots = 2
max_slot_wal_keep_size = 10240
hot_standby = on
promote_trigger_file = 'failover'
wal_level = 'replica'
recovery_target_timeline = 'latest'
cluster_name = '<имя_кластера>'
```

где max_wal_senders — максимально допустимое число одновременных подключений ведомых узлов;

max_replication_slots — максимально допустимое число слотов репликации;

max_slot_wal_keep_size — максимальный размер файлов WAL в МБ для слотов репликации;

hot_standby — возможность подключения к узлу и выполнения запросов только на чтение, пока узел находится в режиме восстановления или в режиме ведомого узла;

promote_trigger_file — файл сценария в каталоге /etc/postgresql/<версия>/<имя_кластера>/, который останавливает режим ведомого узла и переводит его в режим ведущего;

wal_level — количество информации, записываемой в файлы WAL (для репликации — установить replica);

recovery_target_timeline — временной промежуток восстановления (например, latest или current);

cluster_name — уникальное имя кластера, которое идентифицирует данный кластер для различных целей;

2) в конфигурационном файле /etc/postgresql/<версия>/<имя_кластера>/pg_hba.conf на каждом узле СУБД настроить разрешение репликации:

```
host replication replication_user <IP-адрес_1_узла_СУБД>/32 trust
host replication replication_user <IP-адрес_2_узла_СУБД>/32 trust
```

Метод аутентификации trust используется если узлы СУБД работают в доверенной локальной сети.

3) в конфигурационном файле /etc/postgresql/<версия>/<имя_кластера>/pg_hba.conf на каждом узле СУБД настроить разрешение подключения для узлов Pgpool-II и разрешение подключения через виртуальный IP-адрес кластера Pgpool-II:

```
host all all <IP-адрес_1_узла_Pgpool-II>/32 trust
host all all <IP-адрес_2_узла_Pgpool-II>/32 trust
host all all <IP-адрес_3_узла_Pgpool-II>/32 trust
host all all <виртуальный_IP-адрес_кластера_Pgpool-II>/32 trust
```

4) на каждом узле СУБД в каталоге кластера необходимо создать и прописать сценарии recovery_1st_stage_command и pgpool_remote_start. Описание и шаблоны данных сценариев представлены в каталоге /etc/pgpool2/sample_scripts/ в файлах с расширением *.sample;

5) на каждом узле СУБД необходимо создать физический слот репликации для всех остальных узлов путем применения SQL-команды:

```
SELECT * FROM pg_create_physical_replication_slot('<адрес_узла_СУБД>')
```

6) в конфигурационном файле /etc/pgpool2/pgpool.conf на каждом узле Pgpool-II настроить параметры балансировки нагрузки:

```
load_balance_mode = on
backend_clustering_mode = streaming_replication
health_check_user = '<имя_администратора_БД>'
sr_check_user = '<имя_администратора_БД>'
health_check_max_retries = 3
health_check_retry_delay = 5
use_watchdog = on
arping_path = '/usr/bin'
arping_cmd = '/usr/bin/sudo /usr/bin/arping -U
<виртуальный_IP-адрес_кластера_Pgpool-II> -w 1 -I eth0'
auto_failback = on
delegate_IP = <виртуальный_IP-адрес_кластера_Pgpool-II>
recovery_user = '<имя_администратора_БД>'
recovery_1st_stage_command = 'recovery_1st_stage_command'
wd_remove_shutdown_nodes = on
wd_lost_node_removal_timeout = 120
```

Подробное описание параметров приведено в таблице 9.

7) в конфигурационном файле /etc/pgpool2/pgpool.conf на каждом узле Pgpool-II указать в параметрах уникальные идентификаторы узлов СУБД (например, backend_hostname0, backend_port0, backend_weight0 — настройки для первого узла СУБД и backend_hostname1, backend_port1, backend_weight1 — для второго соответственно), а также настроить параметры подключения (порт, путь к каталогу кластера и т.п.):

```
backend_hostname<*> = '<адрес_узла_СУБД>'
backend_port<*> = 5432
backend_weight<*> = 1
backend_data_directory<*> = '/var/lib/postgresql/<версия>/
<имя_кластера>'
backend_flag<*> = 'ALLOW_TO_FAILOVER'
backend_application_name<*> = '<имя_кластера_СУБД>'
health_check_period<*> = 2
#
heartbeat_hostname<*> = '<адрес_узла_pgpool>'
heartbeat_port<*> = 9694
heartbeat_device<*> = ''
#
hostname<*> = '<адрес_узла_pgpool>'
pgpool_port<*> = 5440
wd_port<*> = 9000
```

Подробное описание параметров приведено в таблице 9.

8) необходимо создать и прописать сценарий `failover.sh` на каждом узле Pgpool-II. Описание и шаблон сценария представлен в каталоге `/etc/pgpool2/sample_scripts/` в файле с расширением `*.sample`. Кроме этого, необходимо создать на каждом узле файл `/etc/pgpool2/pgpool_node_id`, в который нужно записать индекс текущего узла. Этот индекс должен совпадать с индексом параметра `hostname<*>` в файле `/etc/pgpool2/pgpool.conf`.

9) для того чтобы служба `pgpool2` не кешировала состояние узлов СУБД и не выводила неверную информацию при перезапуске, необходимо в файле `/usr/lib/systemd/system/pgpool2.service` службы `pgpool2` добавить в секцию `[Service]` параметр `CapabilitiesParsec`, а также изменить параметр запуска бинарного файла, добавив дополнительный параметр `--discard-status`:

```
[Service]
CapabilitiesParsec=PARSEC_CAP_PRIV_SOCK PARSEC_CAP_MAC_SOCK
#
ExecStart=/usr/sbin/pgpool -n --discard-status
```

10) перезапустить Pgpool-II для применения настроек:

```
sudo systemctl daemon-reload
sudo systemctl restart pgpool2
```

ПЕРЕЧЕНЬ ТЕРМИНОВ

Ключ — параметр в виде последовательности псевдослучайных чисел (не предназначен для защиты информации в контексте использования для целей, установленных в документации изделия; к ключам не предъявляются требования по источнику псевдослучайных чисел, криптографической стойкости, времени действия и т.п.).

Сертификат открытого ключа — артефакт, содержащий открытый ключ, информацию о владельце ключа и подтверждающий принадлежность открытого ключа владельцу, защищенный с применением закрытого ключа.

Хеш — строка бит, являющаяся выходным результатом функции хеширования.

Цифровая подпись — результат преобразования хеша для его защиты от несанкционированного доступа с использованием закрытого ключа (не предназначена для криптографической защиты информации).

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

БД — база данных
ЕПП — единое пространство пользователей
КСЗ — комплекс средств защиты
НСД — несанкционированный доступ
ОС — операционная система специального назначения «Astra Linux Special Edition»
СУБД — система управления базами данных

HBA — Host-based Authentication (аутентификация на основе адресов узлов сети)
IP — Internet Protocol (межсетевой протокол)
PAM — Pluggable Authentication Modules (подключаемые модули аутентификации)
SQL — Structured Query Language (язык структурированных запросов)
SSL — Secure Sockets Layer (протокол защищенных сокетов)
TCP — Transmission Control Protocol (протокол управления передачей данных)
UID — User Identifier (идентификатор пользователя)

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	Номер документа	Входящий номер сопроводительного документа и дата	Подпись	Дата
	измененных	замененных	новых	аннулированных					