АО «Концерн ГРАНИТ»

УТВЕРЖДАЮ

Генеральный директор

АО «Концерн ГРАНИТ»

С.С. Гостев

« 18 » июня 2024 г.

СУБД «Квант-Гибрид 1.5»

Руководство системного администратора

ВЕМР.00207-01 32 01

Листов 286

2024

АННОТАЦИЯ

Настоящий документ содержит сведения о назначении, функциях и структуре, режимах работы и ключевой настройке параметров для безопасной работы объектно-реляционной системы управления базами данных «Квант-Гибрид 1.5» (далее – СУБД «Квант-Гибрид 1.5», СУБД, QHB), включая ролевую модель, а также резервирование и восстановление данных.

СОДЕРЖАНИЕ

1 ОБЩИЕ СВЕДЕНИЯ

1.1 Назначение

СУБД «Квант-Гибрид 1.5» представляет собой цифровую платформу, применимую для широкого круга разработчиков государственных информационных систем, приложений, а также для корпоративных отделов по цифровой трансформации предприятий крупного и среднего бизнеса.

1.2 Функции

СУБД «Квант-Гибрид 1.5» обеспечивает выполнение следующих функций:

1) Организация необходимой структуры базы данных (Инициализация экземпляра базы данных и запуск сервера баз данных).
2) Создание и удаление экземпляра базы данных.
3) Создание и удаление новой учетной записи.
4) Запись данных в базу, обеспечение записи данных, вводимых пользователем в базу данных через интерактивный терминал psql.
5) Управление хранением данных.
6) Чтение данных (выполнение запросов пользователя на получение интересующих данных).
7) Редактирование существующих записей.
8) Очистка устаревших копий записей.
9) Удаление записей.
10) Реализация поддержки языка описания данных и языка запросов.
11) Очистка базы данных и генерация внутренней статистики.
12) Обеспечение совместного использования объектов базы данных.
13) Обеспечение возможности прямой загрузки данных.
14) Организация сбора, агрегации и пересылки метрик в различные системы.
15) Контроль доступа к объектам БД:
    a. Ролевой метод управления доступом.

b. Дискреционный метод управления доступом.

16) Идентификация и аутентификация.

17) Поддержка целостности и доступности данных:

a. Резервное копирование и восстановление кластеров баз данных.

b. Управление обновлениями.

c. Предоставление возможности использовать асинхронный пул соединений.

d. Обеспечение целостности данных.

e. Подсчет, проверка контрольных сумм хранимых данных СУБД.

f. Криптографическая защита данных.

18) Регистрация событий.

19) Очистка памяти.

1.3 Условия выполнения

СУБД «Квант-Гибрид 1.5» функционирует на ПЭВМ с характеристиками, не ниже следующих:

- процессор архитектур: x86-64 с тактовой частотой 1,8 ГГц;
- оперативная память: не менее 4 Гбайт оперативной памяти;
- жёсткий диск: не менее 400 МБ (не учитывая размер базы данных);
- устройство чтения компакт-дисков;
- звуковая карта;
- сетевая плата Fast Ethernet 100 Мбит/с.

СУБД «Квант-Гибрид 1.5» функционирует в среде ОС на базе Linux:

- операционная система Альт 8 СП (обозначение программного изделия — ЛКНВ.11100-01) (№ сертификата 3866);
- операционная система специального назначения «Astra Linux Special Edition» (в исполнении РУСБ.10015-01) x86_64, очередное обновление 1.7, (№ сертификата 2557);
- операционная система «РЕД ОС» (№ сертификата 4060).

1.4 Режимы функционирования

Функционирование СУБД «Квант-Гибрид 1.5» предусматривает следующие режимы работы:

- Нормальный режим функционирования компонента («Главный сервер»), включающий в себя:
  - обеспечение полной готовности СУБД «Квант-Гибрид 1.5»;
  - обеспечение круглосуточного (24*7) функционирования СУБД «Квант-Гибрид 1.5»;
  - передача данных репликации на резервные сервера;
  - обеспечение исправной работы оборудования, составляющего комплекс технических средств (включая каналы связи).
- Аварийный режим функционирования. СУБД «Квант-Гибрид 1.5» поддерживает аварийный режим функционирования – режим аварийного отказа. В этом случае резервные средства контроля и управления обеспечивают работу резервного экземпляра СУБД «Квант-Гибрид 1.5».
- Технологический режим функционирования. СУБД «Квант-Гибрид 1.5» поддерживает технологический режим функционирования, обеспечивающий конфигурирование, перенастройку или техническое обслуживание СУБД «Квант-Гибрид 1.5» и её баз данных с возможным частичным снижением быстродействия и блокировкой отдельных функциональных возможностей.

2 ПРИЕМКА И ПРОВЕРКА СУБД «КВАНТ-ГИБРИД 1.5»

При приемке СУБД «Квант-Гибрид 1.5» необходимо проверить контрольные суммы поставляемых пакетов программой ФИКС (уровень 3) и сравнить полученные данные с контрольным суммами в документе «Формуляр». Если контрольные суммы совпадают, то можно переходить к этапу проверки.

Для проверки СУБД «Квант-Гибрид 1.5» необходимо предварительно ее установить и выполнить запуск СУБД. Установка СУБД «Квант-Гибрид 1.5» производится в соответствии с документом «Руководство по установке и настройке».

СУБД «Квант-Гибрид 1.5» считается работоспособной, если при установке Системы не было сообщений об ошибках и контрольные суммы соответствуют контрольным суммам из документа «Формуляр. Приложения».

3 УСТАНОВКА КОМПЛЕКСА ПРОГРАММ ИЗ ИСХОДНЫХ МОДУЛЕЙ

Для установки комплекса программ в среде ОС на базе Linux Альт 8 СП \Astra Linux Special Edition 1.7 (в исполнении РУСБ.10015-01)\ РЕД ОС необходимо:

1. Установить* одну из поддерживаемых СУБД операционных систем:

   ALT Linux 8 СП: установка в минимальной конфигурации,

   Astra Linux SE: уровень защищенности Воронеж или Смоленск,

   Ред ОС: при выборе программного обеспечения установить сервер минимальный.

   * Указаны минимальные требования ОС, которые обеспечивают требуемый уровень безопасности работы с СУБД. Допустимо усиливать безопасность ОС или устанавливать дополнительные пакеты из поставок данных ОС.

2. Перейти в пользователя root, введя команду:

   для ALT Linux и РЕД ОС: `su -`

   для Astra Linux SE: `sudo su`

3. Скопировать пакеты qhb версии 1.5 для выбранной в первом пункте операционной системы.
4. Произвести обновление пакетов ОС до последних доступных версий.
5. Только для ALT Linux: установить **glibc**.

Ниже описана установка glibc для Альт 8 СП с установочного диска № 2.

– Необходимо убедиться, что вы подключили/вставили в сервер диск ALT 8 SP Addon x86_64

Проверить доступность образа:

```
[root@host-21 ~]# apt-cdrom ident
Используется точка монтирования носителя /media/ALTLinux/
Монтируется носитель
Распознаётся.. [29ec5f86561efc9286495a9613f9b96b-2]
Сохранённая метка: 'ALT 8 SP Addon x86_64 build 2021-12-21'
```

– Добавить CD-ROM в список источников пакетов:

```
[root@host-21 ~]# apt-cdrom add
```

```
Используется точка монтирования носителя /media/ALTLinux/
Размонтируется носитель
Вставьте, пожалуйста, носитель в привод и нажмите <Enter>
Монтируется носитель
Распознаётся.. [29ec5f86561efc9286495a9613f9b96b-2]
Носитель сканируется в поисках индексных файлов..  Найдено 1
индексов пакетов и 0 индексов исходных пакетов.
Этот носитель назван:
 'ALT 8 SP Addon x86_64 build 2021-12-21'
Reading Indexes... Завершено
Writing new source list
Список исходных пакетов для этого носителя:
rpm cdrom:[ALT 8 SP Addon x86_64 build 2021-12-21]/ ALTLinux
addon
Повторите эту процедуру для оставшихся носителей из Вашего
набора.
```

– Выполнить **apt-get update**

– Установить glibc:

```
[root@host-21 apt]# apt-get install glibc
Чтение списков пакетов... Завершено
Построение дерева зависимостей... Завершено
Следующие НОВЫЕ пакеты будут установлены:
  glibc
0 будет обновлено, 1 новых установлено, 0 пакетов будет удалено
и 4 не будет обновлено.
Необходимо получить 0B/41,7kB архивов.
После распаковки потребуется дополнительно 0B дискового
пространства.
Получено: 1 cdrom://ALT 8 SP Addon x86_64 build 2021-12-21
ALTLinux/addon                  glibc                  6:2.27-
alt14:c9f2+285733.100.1.1@1637835385 [41,7kB]
Получено 41,7kB за 0s (0B/s).
Совершаем изменения...
```

```
Подготовка...
##################################################################
##################################### [100%]
Обновление / установка...
1:glibc-6:2.27-alt14
##################################################################
##################################### [100%]
Завершено.
```

– Выполнить команду для инициализации контроля целостности (предыдущий пункт вносит изменения в проверку систему на целостность, и ее нужно восстановить):

```
/usr/bin/integrity-applier
```

Дождаться завершения работы команды (система будет перезагружена несколько раз).

– Переименовать файл записи аудита /var/log/audit/audit.log:

```
mv /var/log/audit/audit.log /var/log/audit/audit_old.log
```

– Запустите службу аудита:

```
service auditd start
```

6. Только для РЕД ОС: установить из официального репозитория **llvm-libs-12.0.1**.
7. Только для Astra Linux: установить из официального репозитория пакеты **libllvm11**, **libz3-4**.
8. Установить пакеты qhb версии 1.5 для соответствующей операционной системы.

Блок `<full/path/to/>` в примерах команд необходимо скорректировать на путь до указанных файлов.

Альт 8 СП сервер:

```
apt-get install <full/path/to/>qhb-core-1.5.2-1.x86_64.rpm
<full/path/to/>qhb-contrib-1.5.2-1.x86_64.rpm
```

Astra Linux Special Edition:

```
apt-get install <full/path/to/>qhb-core_1.5.2_amd64.deb
<full/path/to/>qhb-contrib_1.5.2_amd64.deb
```

РЕД ОС:

```
yum install <full/path/to/>qhb-core-1.5.2-1.el7.x86_64.rpm
<full/path/to/>qhb-contrib-1.5.2-1.el7.x86_64.rpm
```

В ходе процесса установки скомпилированы все программные компоненты программы и установлены необходимые для их работы конфигурационные файлы.

9. Установить пакеты дополнительных модулей СУБД, которые планируется использовать, для соответствующей операционной системы, необходимо указывать полный путь до пакета (вместо блока `<full/path/to/>` в примерах команд).

Альт 8 СП сервер:

```
apt-get install <full/path/to/>qhb-license-bin-1.5.2-
2.x86_64.rpm
apt-get install <full/path/to/>qhb-serial-1.5.2-2.x86_64.rpm
apt-get install <full/path/to/>metricsd-1.5.2-2.x86_64.rpm
apt-get install <full/path/to/>qbackup-1.5.2-2.x86_64.rpm
apt-get install <full/path/to/>qcp-1.5.2-2.x86_64.rpm
apt-get install <full/path/to/>qdl-1.5.2-2.x86_64.rpm
apt-get install <full/path/to/>qdlm-1.5.2-2.x86_64.rpm
```

Astra Linux Special Edition:

```
apt-get install <full/path/to/>qhb-license-bin_1.5.2-
1_amd64.deb
apt-get install <full/path/to/>qhb-serial_1.5.2-1_amd64.deb
apt-get install <full/path/to/>metricsd_1.5.2-1_amd64.deb
apt-get install <full/path/to/>qbackup_1.5.2-2_amd64.deb
apt-get install <full/path/to/>qcp_1.5.2-1_amd64.deb
apt-get install <full/path/to/>qdl_1.5.2-1_amd64.deb
apt-get install <full/path/to/>qdlm_1.5.2-1_amd64.deb
```

РЕД ОС:

```
yum install <full/path/to/>qhb-license-bin-1.5.2-
1.el7.x86_64.rpm
yum install <full/path/to/>qhb-serial-1.5.2-1.el7.x86_64.rpm
yum install <full/path/to/>metricsd-1.5.2-1.el7.x86_64.rpm
yum install <full/path/to/>qbackup-1.5.2-2.el7.x86_64.rpm
yum install <full/path/to/>qcp-1.5.2-1.el7.x86_64.rpm
```

```
yum install <full/path/to/>qdl-1.5.2-1.el7.x86_64.rpm
yum install <full/path/to/>qdlm-1.5.2-1.el7.x86_64.rpm
```

4 РАБОТА С СУБД

В данном разделе кратко описана инструкция администратора по работе с СУБД.

4.1 Начало работы

После приемки СУБД, ее необходимо установить (в соответствии с разделом «Установка комплекса программ из исходных модулей») и настроить (в соответствии с разделом «Настройка комплекса программ»).

Кроме указанной в разделе настройки утилиты initdb можно использовать утилиту qhb_bootstrap. Пример использования:

/usr/local/qhb/bin/qhb_bootstrap -D /opt/qhb/data -U qh

После настройки необходимо сконфигурировать установленный экземпляр СУБД в соответствии в соответствии с разделами «Конфигурация сервера: настройка параметров», «Роли в базе данных», «Резервное копирование и восстановление».

Необходимо разделять пользовательские учетные записи СУБД и учетные записи операционной системы, на которой СУБД установлена. Администраторы СУБД и пользователи СУБД должны работать в операционной системе под учетными записями с ограниченными правами (пользователи могут не иметь учетных записей в операционной системе: рекомендуется подключаться к серверу с СУБД удаленно), для обеспечения невозможности влияния на загрузку СУБД и работу функций безопасности СУБД, в том числе оказания влияния на данные журналов регистрации событий. Администратор СУБД и пользователь операционной системы с правами администратора для безопасной работы СУБД должны быть разными лицами.

4.2 Подключение к СУБД

Для подключения к СУБД необходимо:

1) Авторизоваться на устройстве, с которого будет производиться подключение, используя учетную запись пользователя ОС (роль пользователя, роль администратора). Например, на сервер, на котором расположена СУБД.

2) Выполнить проверку наличия на сервере запущенной СУБД. Для этого необходимо в строке терминала ввести команду: systemctl status qhb.
3) Выполнить подключение к нужной БД, используя учетную запись пользователя СУБД. Для этого необходимо в командной строке терминала ввести: psql [параметр...] [имя_бд [имя_пользователя]].

   Более подробно данный пункт представлен в разделе «Подключение к СУБД» настоящего документа.
4) После этого, вы будете переведены в оболочку psql для взаимодействия с БД. Будут доступны операции с СУБД в зависимости от роли и полномочий, учетной записи, под которой был выполнен вход.

   Для дополнительной информации по данному блоку обращайтесь
   - к разделу «Подключение к СУБД» документа ВЕМР.00207-01 33 01 «Руководство пользователя»;
   - к технической документации по СУБД «Квант-Гибрид» 1.5 на ресурсе разработчика.

4.3 Создание и удаление экземпляра базы данных

Чтобы создать новую базу данных, необходимо использовать команде createdb (сервер должен быть установлен и запущен, осуществлено подключение к нему). Пример команды для создания базы данных с наименованием «mydb»:

```
createdb mydb -h localhost
```

Если в ответ на такой запуск нет ошибки, то этот шаг был успешным. Если вы видите сообщение, похожее на:

```
createdb: command not found
```

тогда QHB не был установлен должным образом. Либо он вообще не был установлен, либо путь поиска вашей оболочки не был настроен на его включение. Попробуйте вместо этого вызвать команду с абсолютным путем:

```
/usr/local/qhb/bin/createdb mydb
```

Другой ответ может быть таким:

```
createdb: could not connect to database qhb: FATAL:  role "имя роли" does not exist
```

где упоминается ваше имя пользователя. Это произойдет, если администратор не создал для вас учетную запись пользователя СУБД. (Учетные записи пользователей СУБД отличаются от учетных записей пользователей операционной системы). Если Вы являетесь администратором, обратитесь к разделу настоящего документа, описывающего работу с ролями. Вам нужно стать пользователем операционной системы, под которой была установлена СУБД (обычно ***qhb***), чтобы создать первую учетную запись пользователя. Возможно также, что вам было присвоено имя пользователя QHB, отличающееся от имени пользователя вашей операционной системы; в этом случае вам нужно использовать ключ **-U** или установить окружение **PGUSER**, чтобы указать свое имя пользователя в QHB.

Если у вас есть учетная запись пользователя, но у нее нет прав, необходимых для создания базы данных, вы увидите следующее:

```
createdb: database creation failed: ERROR:  permission denied
to create database
```

Не каждый пользователь имеет право создавать новые базы данных. Если вы установили СУБД самостоятельно, вы должны войти в систему под учетной записью пользователя, с которой вы запустили сервер.

Имена баз данных должны иметь буквенный первый символ и иметь длину не более 63 байтов.

Если необходимо удалить базу данных, то можно это сделать, используя команду dropdb. Пример:

```
dropdb <имя базы данных>
```

Для этой команды имя базы данных по умолчанию не совпадает с именем учетной записи пользователя. Вы всегда должны указывать его.

Это действие физически удаляет все файлы, связанные с базой данных, и не может быть отменено.

4.4 Очистка устаревших копий записей

Для очистки устаревших копий записей из базы данных, администратор СУБД должен выполнить следующие действия:

1) Войти в операционную систему (ОС), используя учетные данные администратора.
2) Запустить интерактивный терминал psql, используя команду psql с необходимыми параметрами, включая параметр -h для удаленного подключения и параметры аутентификации, такие как имя пользователя (-U) и пароль (-W).
3) Выбрать целевую базу данных с помощью команды "\c имя_базы_данных", чтобы переключиться на нужную базу данных для выполнения операций с данными.
4) Определить критерии устаревания копий записей. Критерии могут зависеть от политики хранения данных, законодательных требований или специфических потребностей вашей организации. Обычно устаревшие копии определяются по дате создания, дате последнего изменения или сроку хранения.
5) Выполнить SQL-запросы для удаления устаревших копий записей из соответствующих таблиц базы данных. Запросы должны быть сформулированы с учетом правил и ограничений, заданных администратором СУБД. Пример запроса на удаление устаревших копий записей:

```
DELETE FROM имя_таблицы WHERE условие_устаревания;
```

6) Перед удалением устаревших копий записей, рекомендуется создать резервную копию базы данных для обеспечения безопасности данных и возможности восстановления в случае ошибки или непредвиденных последствий.
7) Провести регулярную очистку устаревших копий записей согласно политике хранения данных, чтобы обеспечить оптимальное использование ресурсов базы данных и снизить риск потери актуальных данных.
8) Рассмотреть возможность автоматизации процесса очистки устаревших копий записей, используя планировщик задач операционной системы или встроенные функции СУБД для создания и выполнения периодических задач.

4.5 Очистка данных в СУБД

Инструкция по работе с данной функцией приведена в разделе «Очистка данных в СУБД» документа ВЕМР.00207-01 33 01 «Руководство пользователя».

4.6 Обеспечение поддержки целостности данных

Для обеспечения поддержки целостности данных в СУБД необходимо выполнить действия, описанные ниже.

Порядок действий пользователя с ролью администратора:

- Выполнить вход в операционную систему.
- Установить соединение с базой данных, для этого необходимо ввести команду plsql с необходимыми параметрами, включая параметр -h для удаленного подключения.
- Определить список пользователей, имеющих доступ к СУБД.
- Создать ограничения целостности данных (например, первичные ключи, внешние ключи, проверки) для таблиц БД.
- Настроить триггеры и процедуры для контроля изменений данных и поддержания целостности.
- Реализовать резервное копирование и восстановление данных для поддержания целостности данных в случае сбоя или потери информации.

5 НАСТРОЙКА КОМПЛЕКСА ПРОГРАММ

5.1 Инициализация экземпляра базы данных

Для инициализации кластера (экземпляра) БД, необходимо выполнить следующие действия:

1) Создать каталог для размещения базы данных командой:

```
mkdir <путь/имя каталога/кластера>
```

2) Разрешить доступ к нему пользователю qhb командой:

```
chown qhb <путь/имя каталога/кластера>
```

3) Инициализировать кластер (экземпляр) базы данных при помощи утилиты initdb:

```
su - qhb -c "/usr/local/qhb/bin/initdb -D <путь/имя каталога/кластера> -U qhb"
```

5.2 Запуск сервера и проверка статуса

1) Выполнить запуск сервера БД при помощи команды:

```
su - qhb -c "/usr/local/qhb/bin/qhb_ctl -D <путь/имя каталога/кластера> -l <путь/имя каталога/кластера>/logfile start"
```

2) Выполнить проверку статуса сервера БД при помощи команды:

```
su - qhb -c "/usr/local/qhb/bin/qhb_ctl -D <путь/имя каталога/кластера> -l <путь/имя каталога/кластера>/logfile status"
```

6 КОНФИГУРАЦИЯ СЕРВЕРА: НАСТРОЙКА ПАРАМЕТРОВ

6.1 Имена и значения параметров

Имена всех параметров не чувствительны к регистру. Каждый параметр принимает значение одного из пяти типов: логический, строка, целое число, число с плавающей запятой или перечисление. От типа зависит синтаксис, применяемый для установки этого параметра:

- **Логический**. Значения могут вводиться как ***on***, ***off***, ***true***, ***false***, ***yes***, ***no***, ***1***, ***0*** (все без учета регистра) или любым однозначным префиксом одного из них.
- **Строка**. Обычно значение заключается в апострофы, при этом внутренние апострофы значения дублируются. Однако если значение является простым числом или идентификатором, апострофы обычно можно опустить (значения, совпадающие с каким-либо ключевым словом SQL, в некоторых контекстах все же требуют заключения в апострофы).
- **Число (целое или с плавающей запятой)**. Числовые параметры можно указывать в обычных форматах, принятых для целого числа и числа с плавающей запятой; если параметр имеет целочисленный тип, дробные значения округляются до ближайшего целого числа. Кроме того, целочисленные параметры дополнительно принимают шестнадцатеричные (с префиксом **0x**) и восьмеричные (с префиксом **0**) значения, но эти форматы не могут быть дробными. Разделители разрядов использовать нельзя. Кавычки требуются только для шестнадцатеричного значения.
- **Число с единицей измерения**. Некоторые числовые параметры имеют неявную единицу измерения, поскольку описывают количество памяти или времени. Этой единицей измерения могут быть байты, килобайты, блоки (обычно восемь килобайтов), миллисекунды, секунды или минуты. При указании только числового значения для одного из таких параметров будет использована единица измерения, установленная для него по

умолчанию, которую можно узнать из **pg_settings.unit**. Для удобства параметры можно задать, явно указав единицу измерения, например, ***'120 ms'*** для значения времени, которая будет преобразована в фактическую единицу измерения параметра. Обратите внимание, что для применения этой функции значение следует записывать в виде строки (с апострофами). Название единицы измерения чувствительно к регистру, а между ним и числовым значением можно ставить пробел.

  - Допустимые единицы памяти: B (байты), kB (килобайты), MB (мегабайты), GB (гигабайты) и TB (терабайты). Множитель для единиц памяти равен 1024, а не 1000.
  - Допустимые единицы времени: us (микросекунды), ms (миллисекунды), s (секунды), min (минуты), h (часы) и d (дни).

  Если с единицей измерения задается дробное значение, оно будет округлено до кратного следующей меньшей единицы, если таковая имеется. Например, 30.1 GB будут преобразованы в 30822 MB, а не в 32319628902 B. Если параметр имеет целочисленный тип, после любого преобразования единицы измерения значение окончательно округляется до целого.

- **Перечисление**. Параметры перечисляемого типа задаются так же, как и строковые параметры, но имеют ограниченный набор значений. Допустимые значения для такого параметра можно найти в **pg_settings.enumvals**. Значения перечислимых параметров не чувствительны к регистру.

6.2 Взаимодействие с параметрами через файл конфигурации

Самый основной способ установить эти параметры — отредактировать файл **qhb.conf**, который обычно хранится в каталоге данных. Копия по умолчанию устанавливается при инициализации каталога кластера базы данных. Пример того, как может выглядеть этот файл:

```
# Это комментарий
```

```
log_connections = yes
log_destination = 'syslog'
search_path = '"$user", public'
shared_buffers = 128MB
```

В каждой строке указывается один параметр. Знак равенства между именем и значением необязателен. Пробелы не играют роли (за исключением значения параметра, заключенного в апострофы), а пустые строки игнорируются. Знаки решетки (#) обозначают оставшуюся часть строки как комментарий. Значения параметров, которые не являются простыми идентификаторами или числами, должны быть заключены в апострофы. Чтобы вставить апостроф в само значение параметра, удвойте его (желательно) или добавьте перед ним обратный слэш. Если файл содержит несколько определений одного и того же параметра, то все они, кроме последнего, игнорируются.

Параметры, установленные таким образом, предоставляют значения по умолчанию для данного кластера. Эти значения будут настройками, видимыми в активных сеансах, если не будут переопределены. В следующих разделах описываются способы, которыми администратор или пользователь могут переопределить эти значения по умолчанию.

Файл конфигурации перечитывается всякий раз, когда основной процесс сервера получает сигнал SIGHUP; этот сигнал легче всего отправить, запустив qhb_ctl reload в командной строке или вызвав функцию SQL ***pg_reload_conf()***. Основной процесс сервера передает этот сигнал всем запущенным в данный момент процессам сервера, так что существующие сеансы тоже принимают новые значения (это произойдет после того, как они завершат любую выполняемую в данный момент клиентскую команду). Кроме того, этот сигнал можно отправить напрямую одному из серверных процессов. Некоторые параметры можно установить только при запуске сервера; любые изменения их значений в файле конфигурации будут игнорироваться до перезапуска сервера. При обработке SIGHUP также игнорируются (но регистрируются) неверные значения параметров в файле конфигурации.

В дополнение к **qhb.conf** каталог данных СУБД «Квант-Гибрид 1.5» содержит файл **qhb.auto.conf**, который имеет тот же формат, что и **qhb.conf**, но предназначен

для редактирования автоматически, а не вручную. Этот файл содержит параметры, настраиваемые командой ALTER SYSTEM. Он считывается одновременно с **qhb.conf**, и заданные в нем параметры действуют аналогичным образом. Параметры в **qhb.auto.conf** переопределяют параметры в ***qhb.conf***.

Внешние средства также могут изменять **qhb.auto.conf**. Не рекомендуется делать это во время работы сервера, поскольку параллельное выполнение команды ALTER SYSTEM может перезаписать такие изменения. Такие программы могут просто добавлять новые настройки параметров в конец файла или удалять дубликаты определений и\или комментарии (как делает ALTER SYSTEM).

Системное представление pg_file_settings может быть полезно для предварительной проверки изменений в файлах конфигурации или для диагностики проблем, если сигнал SIGHUP не дает желаемых результатов.

6.3 Взаимодействие с параметрами через SQL

СУБД «Квант-Гибрид 1.5» предоставляет три команды SQL для установки параметров конфигурации по умолчанию. Уже упомянутая команда ALTER SYSTEM предоставляет средства изменения глобальных значений по умолчанию, доступные для SQL; она функционально равнозначна редактированию **qhb.conf**. Кроме того, есть еще две команды, которые позволяют устанавливать значения по умолчанию на уровне базы данных или роли:

- Команда ALTER DATABASE позволяет переопределять глобальные параметры на уровне базы данных.
- Команда ALTER ROLE позволяет пользователю переопределять как глобальные параметры, так и параметры на уровне базы данных.

Значения, установленные с помощью команд ALTER DATABASE и ALTER ROLE, применяются только при запуске нового сеанса базы данных. Они переопределяют значения, полученные из файлов конфигурации или командной строки сервера, и назначаются значениями по умолчанию до конца сеанса. Обратите внимание, что некоторые параметры нельзя изменить после запуска сервера, и

поэтому их невозможно установить с помощью этих или перечисленных ниже команд.

После подключения клиента к базе данных он может воспользоваться двумя дополнительными командами SQL (и равнозначными им функциями), предоставляемыми СУБД «Квант-Гибрид 1.5» для взаимодействия с локальными параметрами конфигурации сеанса:

– Команда SHOW позволяет проверить текущее значение всех параметров. Соответствующая функция SQL — ***current_setting(setting_name text)***.
– Команда SET позволяет изменять текущее значение тех параметров, которые могут быть установлены локально для сеанса; это не влияет на другие сеансы. Соответствующая функция SQL — ***set_config(setting_name, new_value, is_local)***.

Кроме того, для просмотра и изменения локальных значений сеанса можно воспользоваться системным представлением pg_settings:

– Запрос этого представления аналогичен использованию команды SHOW ALL, но выдает более подробную информацию. Этот подход также более гибкий, так как позволяет задавать условия фильтрации или объединять результат с другими отношениями.
– Использование UPDATE в этом представлении, в частности, обновление столбца ***setting***, эквивалентно выполнению команд SET. Например, команде

```
SET configuration_parameter TO DEFAULT;
```

равнозначен запрос:

```
UPDATE pg_settings SET setting = reset_val WHERE name =
'configuration_parameter';
```

6.4 Взаимодействие с параметрами через командную оболочку

Помимо установки глобальных значений по умолчанию или добавления переопределений на уровне базы данных или роли, параметры СУБД «Квант-Гибрид 1.5» можно задавать с помощью средств оболочки. Через оболочку значения параметров принимают и сервер, и клиентская библиотека libpq.

Во время запуска сервера значения параметров можно передать команде qhb посредством параметра командной строки -c. Например:

```
qhb -c log_connections=yes -c log_destination='syslog'
```

Параметры, предоставленные таким образом, переопределяют те, что были заданы в **qhb.conf** или командой ALTER SYSTEM, поэтому их нельзя изменить глобально без перезапуска сервера.

При запуске клиентского сеанса через libpq настройку параметров можно провести с помощью переменной среды ***PGOPTIONS***. Значения, установленные таким образом, назначаются значениями по умолчанию до конца сеанса, но не влияют на другие сеансы. Формат ***PGOPTIONS*** похож на тот, что используется при запуске команды qhb; в частности, следует указывать флаг -c. Например,

```
env PGOPTIONS="-c geqo=off -c statement_timeout=5min" psql
```

Другие клиенты и библиотеки могут предоставлять свои собственные механизмы (через оболочку или иным образом), позволяющие пользователю изменять параметры сеанса без непосредственного выполнения команд SQL.

6.5 Управление содержимым файла конфигурации

СУБД «Квант-Гибрид 1.5» предоставляет несколько функций для разбиения сложных файлов **qhb.conf** на вложенные файлы. Эти функции особенно полезны при управлении несколькими серверами со связанными, но не идентичными конфигурациями.

Помимо значений отдельных параметров файл **qhb.conf** может содержать ***директивы включения***, благодаря которым другой файл будет прочитан и обработан, как если бы он был вставлен в данное место файла конфигурации. Эта особенность позволяет разделить файл конфигурации на физически отдельные части. Директивы включения записываются в виде:

```
include 'имя_файла'
```

Если значение параметра *'имя_файла'* не является абсолютным путем, оно рассматривается относительно каталога, содержащего включающий файл конфигурации. Включения файлов могут быть вложенными.

Существует также директива ***include_if_exists***, которая действует так же, как ***include***, за исключением случаев, когда указанный файл не существует или не может быть прочитан. Обычная директива ***include*** будет считать это условием возникновения ошибки, но ***include_if_exists*** просто регистрирует сообщение и продолжает обрабатывать включающий файл конфигурации.

Файл **qhb.conf** также может содержать директивы ***include_dir***, которые указывают весь каталог файлов конфигурации, который нужно включить. Они записываются так:

```
include_dir 'каталог'
```

Неабсолютные имена каталогов берутся относительно каталога, содержащего включающий файл конфигурации. В указанном каталоге будут включены только файлы, не являющиеся каталогами, с именами, заканчивающимися суффиксом ***.conf***. Имена файлов, начинающиеся с символа «.» также игнорируется во избежание ошибок, поскольку такие файлы являются скрытыми на некоторых платформах. Множество файлов во включаемом каталоге обрабатывается в порядке их имен (в соответствии с правилами языка С, т. е. цифры идут перед буквами, а заглавные буквы — перед строчными).

Включение файлов или каталогов можно использовать для логического разделения конфигурации базы данных на части, вместо того чтобы вести один большой файл **qhb.conf**.

Например, рассмотрим компанию с двумя серверами баз данных, каждый с разным объемом памяти. Скорее всего, у их конфигураций есть общие элементы, которые будут использоваться, например, для ведения журнала. Но параметры, связанные с памятью, у них будут различаться. Также у каждого сервера могут быть и специфические параметры. Один из способов справиться с этой ситуацией — разбить изменения стандартной конфигурации вашей сети на три файла. Чтобы включить их, можно добавить в конец файла **qhb.conf** следующую запись:

```
include 'shared.conf'
include 'memory.conf'
include 'server.conf'
```

Все системы будут иметь одинаковый **shared.conf**. Все серверы с определенным объемом памяти могут использовать общий файл **memory.conf**; один файл для всех серверов с 8 ГБ ОЗУ, другой для серверов с 16 ГБ. И, наконец, **server.conf** может содержать действительно специфические для сервера параметры конфигурации.

Другая возможность — создать каталог с файлами конфигурации и поместить эту информацию в файлы. Например, можно указать каталог **conf.d** в конце **qhb.conf**:

```
include_dir 'conf.d'
```

Затем можно назвать файлы в каталоге **conf.d** следующим образом:

```
00shared.conf
01memory.conf
02server.conf
```

Это соглашение об именах устанавливает четкий порядок загрузки этих файлов. При этом будет использоваться только последнее значение параметра, найденное сервером при чтении файлов конфигурации. В данном примере значение, установленное в **conf.d/02server.conf**, переопределит значение, заданное в **conf.d/01memory.conf**.

Вместо этого можно использовать этот подход для описательного именования файлов:

```
00shared.conf
01memory-8GB.conf
02server-foo.conf
```

Такая расстановка дает уникальное имя для каждого варианта файла конфигурации. Это может помочь устранить неоднозначность, когда конфигурации нескольких серверов хранятся в одном месте, например в репозитории системы управления версиями (хранение файлов конфигурации базы данных в системе управления версиями — это полезная практика).

6.6 Протоколирование и регистрация событий

В СУБД реализовано два варианта протоколирования и регистрации событий: они будут рассмотрены в подразделах ниже.

6.6.1 QHB_audit

QHB_audit представляет собой расширение для осуществления аудита действий.

Цикл работы расширения:

- Процесс сервера выполняет операцию и генерирует событие.
- Расширение помещает событие в очередь в общей памяти.
- Фоновый процесс извлекает событие из очереди и либо записывает в специальный файл - лог аудита, либо передаёт внешнему HTTP агенту.

Подключение расширения осуществляется с помощью задания параметров в qhb.conf. Для автоматической загрузки расширения при старте экземпляра, нужно в qhb.conf добавить следующую строку:

```
shared_preload_libraries = 'libqhb_audit'
```

Если параметр shared_preload_libraries уже задан, то в него нужно добавить значение libqhb_audit через запятую, например:

```
shared_preload_libraries =
'pg_store_plans,pg_stat_statements,libqhb_audit'
```

Общие настройки, считываются только при инициализации расширения (при старте СУБД):

qhb_audit.queue_memory: объём разделяемой памяти, выделяемой для внутренней очереди сообщений. По умолчанию — 10 МБ.

qhb_audit.queue_capacity: максимальный размер внутренней очереди сообщений. По умолчанию — 1000 событий.

Если перезагрузка сервера невозможна, допускается обновление настроек с помощью команды qhb_ctl reload.

Настройки для записи событий в файл:

- **qhb_audit.csv_file**: путь к файлу, куда будут записываться события. Данную настройку можно менять "на лету". Если параметр не задан, запись событий в файл не осуществляется.

  Если заданное имя не существует в файловой системе, то будет создан новый файл.

Если заданное имя указывает на файл, запрещённый для записи, или на файл, который невозможно создать, в лог сервера будет выдано сообщение об ошибке, а запись событий в файл не будет осуществляться.

Если указан относительный путь к файлу, предполагается, что файл находится в каталоге PGDATA. Промежуточные каталоги, если указаны, не создаются автоматически; при их отсутствии в файловой системе такой путь будет ошибочным.

При включённой ротации лога значение параметра **csv_file** имеет смысл "базового" имени: новые файлы для ротации генерируются на его основе.

– **qhb_audit.csv_file_rotation**: режим ротации файла событий. Данный параметр можно менять «на лету».

Ротация предотвращает неконтролируемое разрастание файла событий путём разбиения его на множество файлов с перезаписью со временем "по кругу". Имена файлов ротации генерируются на основе значения параметра qhb_audit.csv_file. Файлы ротации создаются в том же каталоге, на который указывает csv_file. Этот каталог не создаётся автоматически, а должен существовать на момент запуска. Если указан относительный путь, файлы ротации (как и одиночный файл без ротации) создаются по указанному пути относительно каталога PGDATA.

Если этот параметр не задан или имеет пустое значение, ротация не осуществляется.

Если задано некорректное значение, работа всего расширения блокируется с выдачей сообщения

```
incorrect value of parameter csv_file_rotation
```

Разрешённые значения параметра:

– off - ротация отключена, всё пишется в единственный файл;
– day - дневная ротация в 24 файла;
– week - недельная ротация в 7 файлов;
– month - месячная ротация в 28-31 файл;

– year - годичная ротация в 12 файлов.

6.6.1.1 Предварительные шаги для настройки доступа к данным логов аудита через SQL-запросы

Для доступа к данным аудита через SQL-запросы используется расширение file_fdw. Для этого устанавливается расширение file_fdw, создается основная таблица qhb_audit_log, выбирается схема ротации логов. В зависимости от выбранной схемы берутся за основу приведенные варианты скриптов, где уточняется путь к каталогу данных PGDATA и каталогу файлов логов аудита.

В приведенных ниже скриптах подразумевается, что файлы аудита будут расположены в каталоге audit каталога данных PGDATA и начало имени файла будет также audit. Для этого в настройке qhb_audit.csv_file можно прописать либо абсолютный путь /audit/audit.csv (здесь путь нужно заменить на реальный путь к каталогу данных), либо задать путь относительно текущего каталога данных, например audit/audit.csv. При необходимости, отредактируйте приведенные ниже скрипты в соответствии с реальным расположением файлов.

6.6.1.2 Задание переменной PGDATA и создание каталога audit внутри PGDATA

Здесь и во всех скриптах далее необходимо заменить путь на реальный путь к каталогу PGDATA.

Создаем каталог для файлов логов аудита.

```
export PGDATA=<PGDATA>
sudo -u qhb mkdir $PGDATA/audit
```

Делаем данные аудита доступными на уровне системы только пользователю qhb:

```
sudo -u qhb chmod 700 $PGDATA/audit
```

6.6.1.3 Создание расширения file_fdw и регистрация стороннего сервера

Создайте расширение **file_fdw** и сторонний сервер (как вариант, можно сделать это в базе данных **qhb**).

```
CREATE EXTENSION file_fdw;
CREATE SERVER qhb_audit_log FOREIGN DATA WRAPPER file_fdw;
```

6.6.1.4 Создание основной таблицы

Данная таблица описывает все поля логов и от нее будут наследоваться при их создании все внешние таблицы логов, соответствующие выбранной схеме ротации.

```
create table qhb_audit_log
(
  event_id  bigint,
  event_ts      timestamp,
  event_name    text,
  event_type   text,
  user_name     text,
  db_name       text,
  table_name    text,
  query_text   text,
  event_level  text
);
```

6.6.1.5 Если ротация не нужна

Если необходимости в ротации нет, следует ограничиться созданием одной внешней таблицы для файла, указанного в параметре **qhb_audit.csv_file**. В следующей команде нужно заменить путь на реальный путь до каталога данных:

```
export PGDATA=<PGDATA>
```

В следующей команде нужно заменить **<file.csv>** на реальное имя файла из параметра **qhb_audit.csv_file**:

```
sudo -u qhb touch $PGDATA/audit/<file.csv>
```

В команде создания внешней таблицы при указании **filename** нужно заменить путь на реальный путь до каталога данных, а **<file.csv>** на реальное имя файла из параметра **qhb_audit.csv_file**:

```
CREATE FOREIGN TABLE qhb_audit_log_csv() INHERITS (qhb_audit_log) SERVER qhb_audit_log OPTIONS (filename '<PGDATA>/audit/<file.csv>', FORMAT 'csv', HEADER 'true', DELIMITER ',' );
```

6.6.1.6 Дневная ротация в 24 файла (1 час - 1 файл)

Команды ОС для создания файлов часовых логов

Перед выполнением скриптов не забудьте заменить путь на реальный путь к каталогу данных.

```
export PGDATA=<PGDATA>
sudo -u qhb touch $PGDATA/audit/audit_00h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_01h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_02h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_03h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_04h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_05h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_06h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_07h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_08h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_09h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_10h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_11h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_12h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_13h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_14h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_15h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_16h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_17h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_18h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_19h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_20h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_21h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_22h_of_day.csv
sudo -u qhb touch $PGDATA/audit/audit_23h_of_day.csv
```

Полный перечень примеров и настроек можно посмотреть в технической документации по СУБД «Квант-Гибрид» 1.5 на ресурсе разработчика.

6.6.1.7 Разделение доступности данных аудита разных баз данных

Пользователь с ролью информационной безопасности (ИБ) **qhb_dbms_admin** должен видеть все данные таблицы **qhb_audit_log**. Пользователь с ролью **qhb_db_admin** должен видеть данные аудита по тем базам данных, администратором

которых он является. Подробную информацию о ролях администраторов ИБ смотрите в подразделе про описание Ролей ИБ. Разделение прав реализуется с помощью механизма RLS (Row Level Security). Для этого необходимо задать правила видимости данных, опираясь на данные по ролям и пользователям, а также активизировать механизм RLS для таблицы **qhb_audit_log**. Полная видимость данных для пользователя с ролью **qhb_dbms_admin** реализуется либо через установку для роли атрибута **bypassrls** (этот вариант используется, если таких пользователей несколько) либо через установку владельцем таблицы соответствующего пользователя (удобно в случае, когда такой пользователь всего один). Для владельца таблицы правила видимости по умолчанию не работают и ему доступны все данные таблицы.

6.6.1.7.1 Создание тестовых пользователей и выдача им соответствующих ролей

Для создания пользователей (здесь в качестве примера - db_admin и dbms_admin) и выдачи им соответствующих ролей нужно выполнить следующие команды:

```
\c qhb qhb
CREATE USER db_admin;
SELECT oid as user_oid FROM pg_authid WHERE rolname = 'dbms_admin';
SELECT oid as database_oid FROM pg_database WHERE datname = '<database_name>';
SELECT qhb_make_db_admin(<user_oid>, <database_oid>) FROM pg_database d WHERE d.datname = '<db_name>';

CREATE USER dbms_admin WITH bypassrls; -- пользователь с правом игнорирования правил RLS
SELECT user_oid FROM pg_authid WHERE rolname = 'dbms_admin';
SELECT qhb_make_dbms_admin(<user_oid>);
```

Использовать oid, соответствующий имени роли, можно и таким образом:

```
SELECT qhb_make_dbms_admin('dbms_admin'::regrole::oid);
```

Альтернативой установки атрибута bypassrls может стать задание владения таблицей qhb_audit_log ролью dbms_admin, т.к. политики доступа к данным по умолчанию не срабатывают по отношению к владельцу таблицы:

```
\c qhb qhb
alter table qhb_audit_log owner to dbms_admin;
```

6.6.1.7.2 Создание политики доступа к строкам и активация RLS для таблицы аудита

Для доступности данных только по конкретным базам создадим политику ограничения видимости строк, которая опирается на данные словаря. Нужно активировать механизм RLS для таблицы qhb_audit_log и выдать права на чтение таблицы qhb_audit_log пользователям db_admin и dbms_admin:

```
create policy policy_audit_log on qhb_audit_log using (
  db_name in (select db.datname
              from pg_database db
             , qhb_db_admin a
             where db.oid = a.db_id
           and a.user_id = user::regrole::oid)
);

-- Включаем RLS для таблицы аудита
alter table qhb_audit_log enable row level security;

-- Права на чтение на основную таблицу (дочерние внешние таблицы
не затрагиваем)
grant select on qhb_audit_log to db_admin;
grant select on qhb_audit_log to dbms_admin;
```

Для проверки видимости данных можно поочередно подключиться и выполнить запрос. Для пользователя db_admin видны только записи из соответствующего списка баз:

```
\c qhb db_admin
select * from qhb_audit_log;
```

Для пользователя **dbms_admin** будут доступны все данные:

```
\c qhb dbms_admin
```

```
select * from qhb_audit_log;
```

Если нет строк, соответствующих правам пользователя, или не установлен атрибут **bypassrls**, выдается сообщение:

```
ERROR:  permission denied for table qhb_db_admin
```

6.6.1.7.3 Отключение RLS для таблицы, удаление политики RLS, отзыв прав на таблицу

```
\c qhb qhb
alter table qhb_audit_log disable row level security;
drop policy policy_audit_log on qhb_audit_log;
revoke select on qhb_audit_log from db_admin;
revoke select on qhb_audit_log from dbms_admin;
```

6.6.1.8 Параметры для отправки событий по REST API

Настройки для отправки событий по REST API:

- qhb_audit.rest_url: URL веб-сервера, принимающий сообщения через REST API. Данную настройку можно менять "на лету". Если не задана, отправка событий не осуществляется. Если заданы qhb_audit.rest_url и qhb_audit.csv_file, то осуществляется ТОЛЬКО запись событий в CSV файл.
- qhb_audit.rest_max_attempts: максимальное количество попыток отправить событие на сервер при его недоступности.
- qhb_audit.rest_attempts_interval_ms: интервал в миллисекундах между попытками отправить сообщение на сервер при его недоступности.

6.6.1.9 Формат CSV файла

Вновь созданные CSV файл будет состоять из заголовка (первая строка) и непосредственно событий (остальные строки) со следующими полями:

- уникальный в пределах файла или связанного набора ротируемых файлов идентификатор события;
- время, когда событие произошло, в формате RFC3339, например 2018-02-14T00:28:07Z;
- имя события;

- имя типа события;
- имя пользователя, под управлением которого произошло событие;
- имя базы данных при создании соединения;
- имя таблицы (задается только при DDL-операциях на таблице);
- команда SQL, выполнение которой привело к событию;
- степень важности события: low, medium, high, critical, fatal.

6.6.1.10 Формат сообщений через REST API

Сообщения о событиях отправляются в формате JSON со следующей схемой:

Таблица 1. Схема формата JSON

Наименование	**Тип переменной**	**Описание**
tags	string[]	Сквозные идентификаторы и метки, позволяющие группировать события
Datetime	integer	Unix time. Время возникновения события в миллисекундах, прошедших с полуночи (00:00:00 UTC) 1 января 1970 года. (Fri Apr 03 2020 10:28:37 соответствует 1585909717000)
serviceName	string	Идентификатор Сервиса
serviceVersion	string	Версия Сервиса
name	string	Наименование события аудита
params	object[]	Список параметров события аудита в формате: Name (string):

Наименование	**Тип переменной**	**Описание**
		наименование параметра события аудита, Value (string):значение параметра события
sessionID	string	Идентификатор сессии (если есть)
userLogin	string	Логин пользователя
userName	string	Имя пользователя (необязательный параметр)
userNode	string	Узел (IP/FQDN), с которого пользователь выполняет действия (необязательный параметр)

6.6.1.11 Отслеживаемые события

- Попытки подключения клиентов.
- Запуск и останов кластера QHB.
- Восстановление данных после нештатного останова кластера QHB.
- Команды управления ролями и их привилегиями: CREATE/ALTER/DROP ROLE/USER/GROUP, CREATE/ALTER/DROP USER MAPPING, GRANT, REVOKE.
- Команды DDL: CREATE/ ALTER / DROP DATABASE, CREATE/ALTER/TRUNCATE/DROP TABLE.
- Команды изменения конфигурации SET, ALTER SYSTEM и ALTER DATABASE SET.
- Нарушения целостности бинарных файлов (core и contrib) и каталога pg_proc (процедур).

– Команды управления процедурами и представлениями: CREATE/ALTER/DROP PROCEDURE, CREATE/ALTER/DROP VIEW, CREATE/ALTER/DROP MATERIALIZED VIEW.

Детальное описание настроек, перечня событий аудита и их типов можно посмотреть в технической документации по СУБД «Квант-Гибрид» 1.5 на ресурсе разработчика.

6.6.2 Регистрация ошибок и протоколирование

СУБД поддерживает несколько методов протоколирования сообщений сервера, включая stderr, csvlog и syslog. Значение этого параметра указывается в виде списка желаемых расположений журнала, разделенных запятыми. По умолчанию сообщения сервера протоколируются только в stderr. Этот параметр можно задать только в файле qhb.conf или в командной строке сервера.

Если в log_destination включено значение csvlog, то записи журнала выводятся в формате CSV (значения, разделенные запятыми), что удобно для загрузки журналов в программы. Для генерации вывода журнала в формате CSV должен быть включен logging_collector.

Если включено указание stderr или csvlog, то создается файл current_logfiles для записи местоположения файла (или файлов) журнала, который в данный момент используется сборщиком сообщений для соответствующего назначения. Это обеспечивает удобный способ поиска журналов, используемых в данный момент экземпляром сервера. Вот пример содержимого такого файла:

```
stderr log/qhb.log
csvlog log/qhb.csv
```

current_logfiles воссоздается, когда при прокрутке создается новый файл журнала и когда повторно загружается log_destination. Он удаляется, когда в log_destination не задается ни stderr, ни csvlog и когда сборщик сообщений выключен.

Вероятно, потребуется изменить конфигурацию демона syslog, чтобы использовать вариант syslog для log_destination. Чтобы это работало, понадобится добавить в конфигурацию демона syslog что-то вроде этого: local0.* /var/log/qhb.

Полный перечень настроек можно посмотреть в технической документации по СУБД «Квант-Гибрид» 1.5 на ресурсе разработчика.

Отдельно опишем параметр log_transaction_sample_rate (real).

Этот параметр задает долю транзакций, команды из которых будут записываться в журнал помимо команд, записываемых по другим причинам. Этот параметр действует на все новые транзакции, независимо от продолжительности выполнения ее команд. Значение по умолчанию — 0, то есть команды из транзакций дополнительно не записываются. При значении 1 записываются все команды из всех транзакций. Параметр **log_transaction_sample_rate** полезен для отслеживания выборки транзакции. Только суперпользователи могут изменять этот параметр. Как и все параметры, управляющие протоколированием команд, этот параметр может значительно увеличить издержки.

В таблице ниже поясняются уровни важности сообщений, используемые в СУБД. Также здесь показано, как эти уровни переводятся в системные при отправке выходных данных журнала в syslog.

Таблица 2. Уровни важности сообщений

Важность	**Использование**	**syslog**
DEBUG1..DEBUG5	Предоставляет более подробную информацию для разработчиков. Чем больше номер, тем подробнее.	DEBUG
INFO	Предоставляет неявно запрошенную пользователем информацию, например, вывод команды VACUUM VERBOSE.	INFO

Важность	**Использование**	**syslog**
NOTICE	Предоставляет информацию, которая может быть полезна пользователям, например, уведомление об усечении длинных идентификаторов.	NOTICE
WARNING	Предоставляет предупреждения о возможных проблемах, например, COMMITвне блока транзакций.	NOTICE
ERROR	Сообщает об ошибке, из-за которой прервана текущая команда.	WARNING
LOG	Сообщает информацию, интересующую администраторов, например, выполнение контрольных точек.	INFO
FATAL	Сообщает об ошибке, из-за которой прерван текущий сеанс.	ERR
PANIC	Сообщает об ошибке, из-за которой прерваны все сеансы базы данных.	CRIT

6.7 Расположение файлов

В дополнение к уже упомянутому файлу qhb.conf СУБД «Квант-Гибрид 1.5» использует еще два редактируемых вручную файла конфигурации, которые управляют аутентификацией клиента. По умолчанию все три файла конфигурации хранятся в каталоге данных кластера баз данных. Параметры, описанные в этом разделе, позволяют размещать файлы конфигурации в другом месте.

6.7.1 data_directory (string)

Задает каталог, в котором будут храниться данные. Этот параметр можно установить только при запуске сервера.

6.7.2 config_file (string)

Задает основной файл конфигурации сервера (обычно он называется **qhb.conf**). Этот параметр можно установить только в командной строке СУБД «Квант-Гибрид 1.5».

6.7.3 hba_file (string)

Задает файл конфигурации для проверки подлинности по узлу (обычно он называется **qhb_hba.conf**). Этот параметр можно установить только при запуске сервера.

6.7.4 ident_file (string)

Задает файл конфигурации для сопоставлений имен пользователей (обычно он называется **qhb_ident.conf**). Этот параметр можно задать только при запуске сервера.

6.7.5 external_pid_file (string)

Задает имя дополнительного файла с идентификатором процесса (PID), который сервер должен создать для использования программами администрирования сервера. Этот параметр можно установить только при запуске сервера.

При установке по умолчанию ни один из вышеперечисленных параметров не устанавливается явно. Вместо этого каталог данных указывается параметром командной строки -D или переменной среды ***PGDATA***, и все файлы конфигурации находятся в каталоге данных.

Если требуется сохранить файлы конфигурации не в каталоге данных, параметр командной строки СУБД «Квант-Гибрид 1.5» -D или переменная среды ***PGDATA*** должны указывать на каталог, содержащий файлы конфигурации, а в **qhb.conf** (или в командной строке) должен быть задан параметр **data_directory**, чтобы показать, где на самом деле находится каталог данных. Обратите внимание, что **data_directory** переопределяет путь, заданный в -D и ***PGDATA*** как расположение каталога данных, но не расположение файлов конфигурации.

При желании можно задать имена файлов конфигурации и их пути по отдельности, используя параметры **config_file**, **hba_file** и/или **ident_file**. **config_file** можно указывать только в командной строке СУБД «Квант-Гибрид 1.5», но остальные параметры можно задать в основном файле конфигурации. Если все три параметра и **data_directory** установлены явно, указывать-D или ***PGDATA*** необязательно.

При установке любого из этих параметров относительный путь будет рассматриваться от каталога, в котором запущен СУБД «Квант-Гибрид 1.5».

6.8 Контроль целостности

Механизм контроля целостности реализован с помощью фонового процесса integrity_checker.

Фоновый процесс осуществляет проверку исполняемых модулей и утилит на целостность бинарного файла. Также фоновый процесс осуществляет проверку целостности программного кода хранимых процедур самой СУБД.

Проверка состоит в удостоверении корректности контрольных сумм, подсчитанных при сборке установочного пакета СУБД и после авторизованных изменений.

При использовании СУБД с модулем безопасного хранения QSS необходимо проводить дополнительный контроль целостности при старте ЭВМ и периодически с помощью модуля доверенной загрузки (МДЗ), который имеет сертификат соответствия ФСБ России по классу не ниже 2Б, настроенному согласно документации МДЗ.

6.8.1 Объекты контроля целостности QHB

Объектами являются поставляемые бинарные утилиты и программный код хранимых процедур, как встроенных, так и добавленных пользователем в процессе работы СУБД.

Для целостности бинарных утилит проверка происходит в соответствии с файлами, расположенными в **/var/lib/qhb/data/integrity** и имеющими расширение. **sha256.**

6.8.2 Мероприятия по обнаружению нарушений целостности

В случае обнаружения несовпадения контрольных сумм бинарных утилит и их комплектации в поставке СУБД, происходит блокировка пользователей СУБД, имеющих возможность логина, за исключением супер-пользователей и Администраторов СУБД (см. подраздел про Роли ИБ). Блокировку обеспечивает фоновый процесс СУБД logon_jobs. Блокировка действительная при логине по методам **md5** и **password**.

В случае обнаружения несовпадения контрольных сумм блоков БД, хранящих программный код хранимых процедур, происходит аналогичная блокировка пользователей.

Также происходит блокировка возможности загружать сторонние библиотеки (включая расширения QHB) в адресное пространство запущенной СУБД.

Внимание: в случае обнаружения несоответствия контрольных сумм бинарных утилит и/или программного кода хранимых процедур на реплике в режиме асинхронной репликации, происходит аварийное завершение реплики.

Внимание: после восстановления целостности каталога следует перезапустить СУБД, чтобы разблокировать пользователей и адресное пространство для изменений.

6.8.3 Включение подсистемы контроля целостности

Для включения следует указать опции logon_jobs и integrity_checks в конфигурационном файле:

```
logon_jobs=on
integrity_checks=on
```

По умолчанию данные параметры отключены.

Также следует выбрать периодичность проверок в единицах временной размерности:

```
integrity_period=20s
```

6.9 Параметры модулей и расширений

6.9.1 Общие параметры

6.9.1.1 mask_pg_version (boolean)

Значение по умолчанию —***off*** (выключен). При значении ***on*** (включен) СУБД на запросы версии сервера SELECT version(); будет выдавать версию ядра вместо версии СУБД.

Рекомендовано использовать параметр в случаях, когда внешняя система или клиент требует поддержки специфичной версии СУБД.

Для вступления в силу требуется перезагрузка СУБД.

6.9.1.2 mask_version_template (string)

По умолчанию никак не влияет на значение имени и версии, то есть если значение закомментировано или убрано, то представление версии стандартное и зависит только от **mask_pg_version (boolean)**.

В случае задания администратором в конфигурационном файле специфического выражения, например:

mask_version_template = 'This is QHB {qv} mimicking PG {pv}!\n {lv}'

где:

{qv} - текущая версия QHB;

{pv} - версия текущего ядра PostgreSQL;

{lv} - информация о лицензии QHB.

подменяет стандартное представление версии и имени продукта в соответствии с указанным выражением.

mask_version_template имеет более высокий приоритет, чем **mask_pg_version**. То есть при наличии в конфигурационном файле значения у mask_version_template значение mask_pg_version не учитывается.

Рекомендовано использовать параметр в случаях, когда внешняя система или клиент требует поддержки специфичной версии СУБД.

Для вступления в силу требуется перезагрузка СУБД.

6.9.1.3 enable_backtrace (boolean)

Значение по умолчанию — *true* (включен). Предназначен для автоматического создания файлов трассировки при возникновении ошибок уровня *FATAL*.

Отключение параметра может производиться на отключенной базе данных, с помощью изменения значения в **qhb.conf** на ***false***.

Отключение на работающей базе выполняется командой ALTER SYSTEM SET enable_backtrace TO false;. В результате выполнения команды выдается предупреждение: «WARNING: disabling this will limit support or even make it impossible» (выключение этого ограничит поддержку или даже сделает ее невозможной). Для вступления изменений в силу требуется перезагрузить конфигурацию сервера, вызвав функцию SQL *pg_reload_conf()*, выполнив qhb_ctl reload или отправив сигнал SIGHUP главному серверному процессу.

Если параметр отключен любым способом, то при каждом перезапуске базы данных будет появляться предупреждение об ограниченной поддержке.

Включение параметра также можно осуществлять на отключенной (изменение значения на ***true***) или включенной базе (команда ALTER SYSTEM SET enable_backtrace TO true; и *pg_reload_conf()*).

Файлы трассировки не содержат какой-либо персональной или конфиденциальной информации. Они необходимы для передачи в техподдержку с целью определения части кода, которая привела к сбою.

6.9.2 Параметры менеджера кэша дисковых блоков TARQ

6.9.2.1 use_qhb_cache (boolean)

Логический параметр. При установке значения в ***true*** будет использоваться новая версия кэша.

6.9.2.2 qhb_cache_size (integer)

Общий размер буферного кэша (при включении **use_qhb_cache=true** значение **shared_buffers** не используется).

6.9.2.3 shared_buffers_partitions (integer)

Размер фрагментов кэша (партиций; не имеют отношения к партициям таблицы). Обращение к каждому фрагменту происходит независимо от остальных.

Слишком большой размер приведет к возрастанию конкуренции за блокировки, слишком маленький может привести к задержкам, если в партиции не окажется пригодных к вытеснению блоков (все грязные или занятые фоновым процессом).

Рекомендуемое значение: 128.

6.9.2.4 tarq_cache.touch_queue_ignore (integer)

Процент заполнения фрагмента кэша, при котором обращения начинают приводить к операциям балансировки. Может принимать значения от 1 до 100 (целые числа). Для незаполненного или заполненного частично фрагмента балансировок не требуется.

Рекомендуемое значение: 50.

6.9.2.5 Параметры ребалансировки

Ребалансировка — затратная операция, в нагруженной среде к некоторым буферам (содержимое системных таблиц, словари, последовательности) может быть много тысяч обращений в секунду, существенно не влияющих на общий баланс. Для сглаживания можно использовать следующие параметры:

6.9.2.5.1 tarq_cache.touch_window (integer)

Время в секундах. Все обращения в течение этого периода считаются одним обращением.

Рекомендуемое значение: 3.

6.9.2.5.2 tarq_cache.touch_threshold (integer)

Количество обращений к буферу, по достижении которого происходит операция ребалансировки.

Рекомендуемое значение: 5.

6.9.3 Включение и управление алгоритмом вытеснения параметра HOLDMEM

Для включения и управления алгоритмом вытеснения для таблиц, по возможности хранящихся в памяти, используются следующие параметры в файле конфигурации.

6.9.3.1 use_possible_buffer (boolean)

Логический параметр. При установке значения в ***true*** будет использоваться буферный кэш.

6.9.3.2 qhb_possible_buffers_size (integer)

Общий размер буферного кэша.

6.9.3.3 shared_buffers_partitions (integer)

Используется общий параметр с ***TARQ*** — размер фрагментов кэша (партиций; не имеют отношения к партициям таблицы). Обращение к каждому фрагменту происходит независимо от остальных. Слишком большой размер партиций приведет к возрастанию конкуренции за блокировки, слишком маленький может привести к задержкам, если в партиции не окажется пригодных к вытеснению блоков (все «грязные» или занятые фоновым процессом).

Рекомендуемое значение: 128.

6.9.3.4 use_qhb_onlymem_cache (boolean)

Логический параметр. При установке значения в ***true*** будет использоваться кэш памяти.

6.9.3.5 qhb_onlymem_cache_size (integer)

Общий размер кэша памяти.

6.9.4 Параметры конфигурации расширения Rbytea

6.9.4.1 rbytea.filesystem_storage_path (string)

Определение каталога (точки монтирования файловой системы/тома) для сохранения образов данных.

Двоичные данные будут сохраняться в данном каталоге сервера. Для каждой базы будет создаваться свой подкаталог (по OID базы), а внутри него, во множестве подкаталогов, — собственно файлы с данными.

Имена вложенных каталогов от каталога базы данных до файла будут составлять ***uuid*** типа ***rbytea***, а расширение файла — номер транзакции, в которой данные впервые появились в системе.

Каталог должен быть доступен на чтение и запись в него для пользователя, от имени которого запускается сервер баз данных.

По умолчанию, если параметр опущен или пуст, каталогом для сохранения двоичных данных назначается ***'<каталог_базы_данных>'***.

6.9.4.2 shared_preload_libraries (string)

Загрузка разделяемой библиотеки при старте QHB.

Данный параметр обеспечивает загрузку разделяемой библиотеки при старте QHB и инициализацию фонового процесса для очистки устаревших образов. В противном случае автоочистка устаревших образов производится не будет.

Необходимо добавить значение ***'librbytea'***.

Примечание: если параметр shared_preload_libraries уже содержит указание на загрузку других библиотек, нужно не перезаписать его значение, а добавить через разделитель ***librbytea***.

6.9.4.3 rbytea.worker_restart_time (integer)

Интервал запуска фонового процесса очистки.

Фоновый процесс не работает постоянно. Поскольку данные достаточно статичны, не требуется запускать процесс очистки слишком часто. Задержка от окончания предыдущего запуска до следующего запуска задается в данном параметре. Значение указывается в секундах. Значение по умолчанию — 86400 секунд (сутки).

6.9.4.4 rbytea.databases_for_vacuuming (string)

Задание баз данных для фонового процесса очистки.

Параметр указывает, какие базы данных подлежат очистке. Указываются названия баз данных через запятую. Главный процесс базы данных будет запускать столько фоновых процессов очистки, сколько баз данных перечислено в данном параметре. Значение по умолчанию — ***qhb***.

6.9.4.5 rbytea.filesystem_qss_mode (integer)

Примечание: параметр зависит от наличия QSS и связанных его параметров, см. Параметры модуля безопасного хранения QSS.

Включение фонового шифрования.

Если в системе доступно фоновое шифрование при записи на диск, данный параметр позволяет зашифровывать также и двоичные данные ***rbytea***. Для шифрования в параметре необходимо установить значение 2. Значение по умолчанию — 0.

Вопросы смены ключа шифрования должны решаться администратором базы данных.

6.9.5 2В: Поддержка решений 1С

6.9.5.1 Конфигурационные параметры немедленного обновления статистики

online_analyze.enable (boolean)

Включает функцию немедленного анализа. Значение по умолчанию — ***false*** (выключен).

online_analyze.verbose (boolean)

Включает возможность протоколирования в журнал при сборе статистики (аналогично команде SQL ANALYZE VERBOSE). Значение по умолчанию — ***false*** (выключен).

online_analyze.table_type (enum)

Указывает тип таблиц, для которых следует применять немедленный анализ. Возможные варианты значений: ***all*** (все), ***persistent*** (постоянные), ***temporary*** (временные), ***none*** (никакие). Значение по умолчанию — ***temporary***.

online_analyze.exclude_tables (string)

Указывает список таблиц, исключаемых из немедленного анализа. Например, "**имя_таблицы_1**, **имя_таблицы_2** ...". Значение по умолчанию — пустая строка ('').

online_analyze.include_tables (string)

Задает список таблиц, подлежащих немедленному анализу (этот параметр переопределяет параметр **online_analyze.exclude_tables**). Значение по умолчанию — пустая строка ('').

Примечание: значения этих двух параметров записываются в виде списка имен таблиц через запятую.

online_analyze.local_tracking (boolean)

Включает хранение статистики временных таблиц в локальном кэше. Значение по умолчанию — ***false*** (выключен).

online_analyze.capacity_threshold (integer)

Задает максимальное число временных таблиц, сохраняемых в локальном кэше. Значение по умолчанию — 100000.

Примечание: по умолчанию используется системная статистика, но имеет смысл хранить ее в локальном кэше процесса, обслуживающего сессию.

online_analyze.scale_factor (floating point)

Задает процент измененных строк от всей таблицы, при котором начинается немедленный анализ. Значение по умолчанию — 0.1.

online_analyze.threshold (integer)

Задает минимальное число измененных строк, после которого может начаться немедленный анализ. Значение по умолчанию — 50.

online_analyze.lower_limit (integer)

Указывает минимальное число строк в таблице, при котором может начаться немедленный анализ. Значение по умолчанию — 0.

online_analyze.min_interval (integer)

Минимальный интервал времени между вызовами команды ANALYZE для конкретной таблицы (в миллисекундах). Значение по умолчанию — 10000.

6.9.5.2 Конфигурационные параметры для управления использованием индексов

plantuner.disable_index (string)

Задает список индексов, которые нельзя использовать.

plantuner.enable_index (string)

Задает список индексов, которые все-таки можно использовать (этот параметр переопределяет параметр *plantuner.disable_index).*

plantuner.only_index (string)

Задает точный список индексов, которые можно использовать (этот параметр переопределяет параметры *plantuner.disable_index* и *plantuner.enable_index*).

Примечание: значения этих трех параметров записываются в виде списка имен индексов через запятую.

Разрешение имени индекса происходит для конкретного пользователя, даже если параметр задан на уровне сервера. Если индекс не найден по имени, например, когда индекс вообще не существует, он игнорируется.

plantuner.fix_empty_table (boolean)

Этот параметр меняет оценку планировщиком совсем пустых таблиц (только что созданных или опустошенных при помощи команды TRUNCATE). Если значение равно ***true***, то СУБД планирует в предположении, что эти таблицы так и будут оставаться пустыми. При значении ***false*** СУБД по умолчанию будет считать, что таблицы содержат некоторое количество записей (обычно 20).

6.9.6 Очистка памяти с повышенной безопасностью

QHB предоставляет возможность перед удалением данных выполнить соответствующую очистку памяти, то есть выполнить перезапись этих данных нулевыми байтами. Управление данными возможностями осуществляется с помощью соответствующих конфигурационных параметров в файле конфигурации **qhb.conf**.

В стандартной версии данные параметры по умолчанию отключены (***false***) для исключения торможения. В сертифицированной версии эти параметры должны быть включены (***true***) в обязательном порядке.

wipe_file_on_delete (boolean)

Включает удаление файлов из внешней памяти. Для SQL-команд, выполняющих удаление файлов и возврат операционной системе соответствующего дискового пространства, включение данного параметра обеспечит заполнение файлов нулевыми байтами перед удалением. К данным командам относятся:

- команды по удалению объектов базы данных: DROP DATABASE, DROP INDEX, DROP MATERIALIZED VIEW, DROP SCHEMA, DROP TABLE, DROP TEMPORARY TABLE, TRUNCATE;
- команды по пересозданию объектов базы данных: ALTER TABLE ADD COLUMN, ALTER TABLE ALTER COLUMN TYPE, REINDEX, VACUUM FULL.

wipe_heaptuple_on_delete (boolean)

Включает очистку страниц. Включение данного параметра обеспечит заполнение нулевыми байтами всех мест на странице, которые помечаются как свободные при выполнении команды VACUUM.

wipe_memctx_on_free (boolean)

Включает очистку блоков памяти в ОЗУ. Включение данного параметра обеспечит заполнение нулевыми байтами освобождаемой области оперативной памяти (ОЗУ), соответствующей удаляемому контексту.

wipe_xlog_on_free (boolean)

Включает очистку файлов WAL. Включение данного параметра обеспечит заполнение нулевыми байтами сегментов журнала упреждающей записи (WAL) перед удалением или переиспользованием.

6.9.7 Запуск планировщика, брокера очереди сообщений при старте СУБД

with_qsched (string) —включает запуск планировщика задач при старте СУБД.

with_mq_broker (string) —включает запуск брокера очереди сообщений при старте СУБД.

Примечание: значения этих параметров записываются в виде списка пар владельцев баз данных и имен баз данных через запятую. Например, ***'db_owner1:dbname1, db_owner2:db_name2'*** или ***'db_owner3:dbname3'***, где ***db_owner$*** — имя владельца базы данных, а ***db_name$*** — имя базы данных.

6.9.8 Автоматическая очистка журнала выполненных задач планировщика

Включение очистки журнала выполненных задач требует указания параметров *qsched_log_rotation_max_lifetime* и/или *qsched_log_rotation_max_entries*, а также *qsched_log_rotation_period*.

qsched_log_rotation_max_lifetime (string) —максимальное время жизни записи в журнале, указывается в формате *'1d 1h 1m'*.

qsched_log_rotation_max_entries (string) —максимальное количество записей в журнале, указывается в формате *'1d 1h 1m'*.

qsched_log_rotation_period (string) —периодичность удаления записей в журнале, указывается в формате *'1d 1h 1m'*.

qsched_log_rotation_strategy (integer) —стратегия очистки журнала, ***0*** — DELETE, ***1*** — TRUNCATE, ***2*** — автоматически. Значение по умолчанию — ***0***.

Примечание: поскольку TRUNCATE системных таблиц требует разрешения на модификацию системных таблиц, для работ стратегий ***1*** и ***2*** требуется дополнительный параметр конфигурации *allow_system_table_mods = true*.

qsched_log_rotation_adjustment_period (integer)

Периодичность подстройки автоматической стратегии. Значение по умолчанию — 10.

6.9.9 Включение фоновых процессов целостности и безопасности

Включение фонового процесса контроля целостности требует указания параметра ***integrity_checks***.

Подсистема контроля целостности поддерживает конфигурацию периодичности проверок целостности на запущенной СУБД: используется параметр ***integrity_period***.

Включение фонового процесса контроля профилей безопасности требует указания параметра ***logon_jobs***.

integrity_checks (bool)

Включение фонового процесса контроля целостности, указывается в формате ***'on off'***. Значение по умолчанию — ***off*** (выключен).

integrity_period (string)

Период проведения проверок целостности на данной СУБД, указывается в формате временных единиц ***'1d 1h 1m'***. Значение по умолчанию — 12 часов (***12h***).

logon_jobs (bool)

Включение фонового процесса контроля профилей безопасности, указывается в формате ***'on off'***. Значение по умолчанию — ***off*** (выключен).

Важно: перед обновлением кластера утилитами ***qhb_upgrade*** следует выключить параметр ***logon_jobs,*** выставив его в ***off***, и перезапустить СУБД. Иначе стабильность процесса обновления кластера не гарантируется.

Важно: перед обновлением кластера утилитами ***qhb_upgrade*** следует выключить параметр ***integrity_checks,*** выставив его в ***off***, и перезапустить СУБД. Иначе стабильность процесса обновления кластера не гарантируется.

6.9.10 Параметры модуля безопасного хранения QSS

Важно: параметры, описанные в данном разделе, имеют смысл, если QHB использует возможности QSS, который должен быть предварительно установлен. В противном случае использование шифрования будет приводить к ошибкам. При этом

без установленного QSS СУБД остается полнофункциональной и работоспособной, если при этом не производится попыток обращения к возможностям QSS.

qss_mode (integer)

Значение ***0*** (по умолчанию) означает отказ от использования возможностей QSS. При значении ***2*** СУБД может использовать возможности QSS. Значение ***1*** является зарезервированным и устаревшим.

qss_encrypt_statistic (bool)

Шифровать ли статистики по данным таблиц. При включении параметра будут зашифрованы статистики по всем таблицам, а не только по шифрованным. Если **qss_mode** выключено, то параметр игнорируется.

Значение по умолчанию ***false***.

После изменения значения параметра нужно пересобрать статистику всех таблиц, подробнее смотрите в разделе по модулю QSS.

qss_shmem_size_cap (integer)

Устанавливает память, зарезервированную для связи с сервером QSS. Использование памяти в любом случае начинается с 8 Кб, но может вырасти до этого предела. Каждый бэкэнд имеет собственную память. Объекты большего размера могут быть зашифрованы, но для обработки каждого из них память будет выделяться и освобождаться заново.

.

7 КОНФИГУРАЦИЯ СЕРВЕРА: ПОДКЛЮЧЕНИЯ И АУТЕНТИФИКАЦИЯ

7.1 Параметры подключения

7.1.1 listen_addresses (string)

Задает адреса TCP/IP, по которым сервер должен принимать подключения от клиентских приложений. Значение имеет форму разделенного запятыми списка имен хостов и/или числовых IP-адресов. Специальная запись ***** соответствует всем доступным IP-интерфейсам. Запись **0.0.0.0** позволяет прослушивать все адреса IPv4, а **::** — все адреса IPv6. Если список пуст, сервер не прослушивает ни один IP-интерфейс, и в этом случае для подключения к нему можно использовать только сокеты домена Unix. Значением по умолчанию является ***localhost***, что позволяет устанавливать только локальные «петлевые» подключения по TCP/IP. Хотя аутентификация клиента позволяет скрупулезно управлять доступом к серверу, **listen_addresses** определяет, через какие именно интерфейсы будут приниматься соединения, что может помочь предотвратить злонамеренные попытки подключения через незащищенные сетевые интерфейсы. Этот параметр можно задать только при запуске сервера.

7.1.2 port (integer)

TCP-порт, который прослушивает сервер; по умолчанию это **5432**. Обратите внимание, что этот номер порта используется для всех IP-адресов, через которые сервер принимает подключения. Этот параметр можно задать только при запуске сервера.

7.1.3 max_connections (integer)

Определяет максимальное количество одновременных подключений к серверу базы данных. По умолчанию обычно это 100 подключений, но может быть и меньше, если настройки ядра не будут это поддерживать (что определяется в процессе ***initdb***). Этот параметр можно задать только при запуске сервера.

При запуске резервного сервера следует установить для этого параметра значение большее или равное значению на главном сервере. В противном случае на резервном сервере не будут разрешены запросы.

7.1.4 superuser_reserved_connections (integer)

Определяет количество «слотов» подключений, которые зарезервированы для суперпользователей СУБД «Квант-Гибрид 1.5». В большинстве случаев подключения max_connections могут быть активными одновременно. Каждый раз, когда число активных одновременных подключений больше или равно **max_connections** минус **superuser_reserved_connections**, новые подключения будут приниматься только от суперпользователей, а подключения для репликации приниматься не будут.

По умолчанию резервируется три соединения. Это значение должно быть меньше значения **max_connections**. Этот параметр можно задать только при запуске сервера.

7.1.5 unix_socket_directories (string)

Задает каталог сокетов Unix-домена, через которые сервер должен принимать подключения от клиентских приложений. Можно создать несколько сокетов, перечислив несколько каталогов, разделенных запятыми. Пробелы между записями игнорируются; если в имени каталога требуются пробелы или запятые, заключите его в кавычки. Пустое значение указывает, что сервер не принимает подключения через сокеты домена Unix, и в этом случае к нему можно подключиться только по TCP/IP. Значением по умолчанию обычно является ***/tmp***, но его можно изменить во время сборки. Этот параметр можно задать только при запуске сервера.

В дополнение к самому файлу сокета, который называется **.s.PGSQL.***nnnn*, где **nnnn** — это номер порта сервера, в каждом из каталогов **unix_socket_directories** будет создан обычный файл с именем **.s.PGSQL.***nnnn***.lock**. Эти файлы ни в коем случае нельзя удалять вручную.

7.1.6 unix_socket_group (string)

Задает группу-владельца сокетов Unix-домена (пользователем-владельцем сокетов всегда является пользователь, запускающий сервер). В сочетании с

параметром **unix_socket_permissions** этот параметр можно использовать в качестве дополнительного механизма управления доступом для подключений через сокеты Unix-домена. По умолчанию это пустая строка, что делает владельцем группу по умолчанию пользователя сервера. Этот параметр можно задать только при запуске сервера.

7.1.7 unix_socket_permissions (integer)

Задает права доступа к сокетам Unix-домена. Для этих сокетов применяется стандартный набор разрешений файловой системы Unix. Ожидается, что значение параметра будет иметь числовой вид, указанный в формате, принимаемом системными функциями **chmod** и **umask** (для использования обычного восьмеричного формата число должно начинаться с 0 (нуля)).

Разрешения по умолчанию — **0777**, при которых подключаться к сокету могут все. Разумными альтернативами являются **0770** (доступ имеет только пользователь и группа) и **0700** (доступ имеет только пользователь). Обратите внимание, что для сокетов Unix-домена имеет значение только право на запись, поэтому нет смысла устанавливать или отзывать права на чтение или выполнение.

Этот параметр можно задать только при запуске сервера.

Этот параметр не применим к системам, которые полностью игнорируют разрешения для сокетов. Там схожего эффекта можно достичь, задав в параметре **unix_socket_directories** каталог, доступ к которому ограничен желаемой аудиторией.

7.1.8 bonjour (boolean)

Включает оповещение о существование сервера посредством Bonjour. По умолчанию выключен. Этот параметр можно задать только при запуске сервера.

7.1.9 bonjour_name (string)

Задает имя службы в среде Bonjour. Если для этого параметра задана пустая строка, '' (это значение по умолчанию), то в качестве имени службы используется имя компьютера. Этот параметр игнорируется, если сервер был скомпилирован без поддержки Bonjour. Этот параметр можно задать только при запуске сервера.

7.1.10 tcp_keepalives_idle (integer)

Задает время бездействия сети, по истечении которого операционная система должна отправлять клиенту сигнал TCP о поддержании активного состояния. Если это значение указано без единиц измерения, оно считается заданным в секундах. При значении 0 (по умолчанию) выбирается значение, принятое по умолчанию в операционной системе. Этот параметр поддерживается только в системах, поддерживающих **TCP_KEEPIDLE** или равнозначный ему параметр сокета; в других системах он должен быть равен нулю. В сеансах, подключенных через сокеты Unix-домена, этот параметр игнорируется и всегда считается равным нулю.

7.1.11 tcp_keepalives_interval (integer)

Задает время, по истечении которого следует повторно передать сигнал TCP о поддержании активного состояния, если получение предыдущего сигнала не было подтверждено клиентом. Если это значение указано без единиц измерения, оно считается заданным в секундах. При значении 0 (по умолчанию) выбирается значение, принятое по умолчанию в операционной системе. Этот параметр поддерживается только в системах, поддерживающих **TCP_KEEPINTVL** или равнозначный ему параметр сокета; в других системах он должен быть равен нулю. В сеансах, подключенных через сокеты домена Unix, этот параметр игнорируется и всегда считается равным нулю.

7.1.12 tcp_keepalives_count (integer)

Задает количество сигналов TCP о поддержании активного состояния, которые могут быть потеряны до того, как соединение сервера с клиентом будет считаться прерванным. При значении 0 (по умолчанию) выбирается значение, принятое по умолчанию в операционной системе. Этот параметр поддерживается только в системах, поддерживающих **TCP_KEEPCNT** или равнозначный ему параметр сокета; в других системах он должен быть равен нулю. В сеансах, подключенных через сокеты домена Unix, этот параметр игнорируется и всегда считается равным нулю.

7.1.13tcp_user_timeout (integer)

Задает время, в течение которого передаваемые данные могут оставаться неподтвержденными, прежде чем TCP-соединение будет принудительно закрыто. Если это значение указано без единиц измерения, оно считается заданным в миллисекундах. При значении 0 (по умолчанию) выбирается значение, принятое по умолчанию в операционной системе. Этот параметр поддерживается только в системах, поддерживающих **TCP_USER_TIMEOUT** или равнозначный ему параметр сокета; в других системах он должен быть равен нулю. В сеансах, подключенных через сокеты домена Unix, этот параметр игнорируется и всегда считается равным нулю.

7.2 Аутентификация

7.2.1 authentication_timeout (integer)

Максимальное время, за которое должна завершиться аутентификация клиента. Если потенциальный клиент не завершил протокол аутентификации за это время, сервер закрывает соединение. Это не дает зависшим при подключении клиентам занимать соединение неограниченно долго. Если это значение указано без единиц измерения, оно считается заданным в секундах. Значение по умолчанию составляет одну минуту (1m). Этот параметр можно задать только в файле **qhb.conf** или в командной строке сервера.

7.2.2 password_encryption (enum)

Если в CREATE ROLE или ALTER ROLE указан пароль, этот параметр определяет алгоритм, которым тот будет зашифрован. Значением по умолчанию является ***md5***, то есть пароль сохраняется в виде хэша MD5 (также в качестве псевдонима для ***md5*** допускается значение ***on***). При значении ***scram-sha-256*** пароль зашифруется алгоритмом SCRAM-SHA-256.

Обратите внимание, что старые клиенты могут не поддерживать механизм аутентификации SCRAM и, следовательно, не будут работать с паролями, зашифрованными алгоритмом SCRAM-SHA-256.

7.2.3 krb_server_keyfile (string)

Задает путь к файлу ключей Kerberos для данного сервера. Если в качестве значения задается пустая строка, то используется значение по умолчанию, зависящее от системы. Этот параметр можно задать только в файле **qhb.conf** или в командной строке сервера.

7.2.4 krb_caseins_users (boolean)

Определяет, должны ли имена пользователей GSSAPI обрабатываться без учета регистра. По умолчанию установлено значение ***off*** (регистр учитывается). Этот параметр можно задать только в файле **qhb.conf** или в командной строке сервера.

7.2.5 db_user_namespace (boolean)

Этот параметр позволяет относить имена пользователей к базам данных. По умолчанию установлено значение ***off*** (выключен). Этот параметр можно задать только в файле **qhb.conf** или в командной строке сервера.

Если этот параметр включен, имена пользователей следует создавать в виде **имя_пользователя@база_данных**. Когда подключающийся клиент передает **имя_пользователя**, к нему добавляются **@** и имя базы данных, и сервер ищет пользователя по этому специфическому для базы данных имени. Обратите внимание, что при создании в среде SQL пользователей с именами, содержащими **@**, потребуется заключать эти имена в кавычки.

Если этот параметр включен, вы все равно можете создавать обычных глобальных пользователей. Для этого достаточно добавить **@** при указании имени пользователя в клиенте, например ***joe@***. Символ **@** будет удален раньше, чем сервер найдет имя пользователя.

db_user_namespace вызывает расхождения в представлении имени пользователя на стороне клиента и сервера. Проверка подлинности всегда выполняется с именем на стороне сервера, поэтому методы аутентификации должны настраиваться по серверному имени пользователя, а не клиентскому. Поскольку метод аутентификации **md5** использует имя пользователя в качестве соли как на стороне клиента, так и на стороне сервера, при включенном параметре **db_user_namespace** этот метод использовать нельзя.

7.3 SSL

7.3.1 ssl (boolean)

Разрешает подключения SSL. Этот параметр можно задать только в файле **qhb.conf** или в командной строке сервера. Значение по умолчанию — ***off*** (выключен).

7.3.2 ssl_ca_file (string)

Задает имя файла, содержащего сертификаты центров сертификации (ЦС) для SSL- сервера. Относительные пути рассматриваются от каталога данных. Этот параметр можно задать только в файле **qhb.conf** или в командной строке сервера. По умолчанию этот параметр пуст, то есть файл ЦС не загружается, и проверка клиентских сертификатов не выполняется.

7.3.3 ssl_cert_file (string)

Задает имя файла, содержащего сертификат SSL-сервера. Относительные пути рассматриваются от каталога данных. Этот параметр можно задать только в файле **qhb.conf** или в командной строке сервера. Значение по умолчанию — **server.crt**.

7.3.4 ssl_crl_file (string)

Задает имя файла, содержащего список отзыва сертификатов (CRL, Certificate Revocation List) для SSL-сервера. Относительные пути рассматриваются от каталога данных. Этот параметр можно задать только в файле qhb.conf или в командной строке сервера. По умолчанию этот параметр пуст, то есть файл CRL не загружается.

7.3.5 ssl_key_file (string)

Задает имя файла, содержащего закрытый ключ SSL-сервера. Относительные пути рассматриваются от каталога данных. Этот параметр можно задать только в файле **qhb.conf** или в командной строке сервера. Значение по умолчанию — **server.key**.

7.3.6 ssl_ciphers (string)

Задает список наборов шифров SSL, которые разрешено использовать для SSL-соединений. Синтаксис этого параметра и список поддерживаемых значений можно найти на странице руководства по шифрам в пакете OpenSSL. Этот параметр можно

задать только в файле **qhb.conf** или в командной строке сервера. Значение по умолчанию — ***HIGH:MEDIUM:+3DES:!aNULL***. Обычно это разумный выбор, если у вас нет особых требований к безопасности.

Объяснение значения по умолчанию:

- **HIGH* — наборы шифров, в которых используются шифры из группы HIGH (например, AES, Camellia, 3DES)*
- **MEDIUM* — наборы шифров, в которых используются шифры из группы MEDIUM (например, RC4, SEED)*
- **+3DES* — порядок OpenSSL по умолчанию для HIGH проблематичен, потому что он ставит 3DES выше, чем AES128. Это неправильно, поскольку 3DES менее безопасен, чем AES128, и работает гораздо медленнее. Включение *+3DES* меняет этот порядок, размещая 3DES после всех других шифров групп HIGH и MEDIUM.*
- **!aNULL* — выключает наборы анонимных шифров, не требующие аутентификации. Такие наборы уязвимы для атак через посредника, и поэтому не должны использоваться.*

Доступные наборы шифров и их свойства зависят от версии OpenSSL. Чтобы увидеть фактическую информацию о них для текущей установленной версии OpenSSL, используйте команду openssl ciphers -v 'HIGH:MEDIUM:+3DES:!aNULL'. Обратите внимание, что этот список фильтруется во время выполнения, в зависимости от типа ключа сервера.

7.3.7 ssl_prefer_server_ciphers (boolean)

Определяет, следует ли отдавать предпочтение шифрам SSL-сервера, а не клиента. Этот параметр можно задать только в файле **qhb.conf** или в командной строке сервера. Значение по умолчанию — ***on*** (включен).

Обычно лучше использовать шифры сервера, так как в его конфигурации менее вероятны ошибки.

7.3.8 ssl_ecdh_curve (string)

Задает имя кривой для использования при обмене ключами ECDH. Эту кривую должны поддерживать все подключающиеся клиенты. Это необязательно должна

быть та же кривая, с которой был получен ключ эллиптической кривой сервера. Этот параметр можно задать только в файле **qhb.conf** или в командной строке сервера. Значение по умолчанию — ***prime256v1***.

Имена OpenSSL для наиболее распространенных кривых: ***prime256v1*** (NIST P-256), ***secp384r1*** (NIST P-384), ***secp521r1*** (NIST P-521). Полный список доступных кривых можно получить с помощью команды openssl ecparam -list_curves. Однако не все из них пригодны для TLS.

7.3.9 ssl_min_protocol_version (enum)

Задает минимальную версию протокола SSL/TLS, которую можно использовать. Допустимые на данный момент значения: ***TLSv1***, ***TLSv1.1***, ***TLSv1.2***, ***TLSv1.3***. Старые версии библиотеки OpenSSL поддерживают не все значения; при выборе неподдерживаемой версии возникнет ошибка. Версии протокола до TLS 1.0, а именно SSL v.2 и v.3, в любом случае не работают.

Значение по умолчанию — ***TLSv1***, в основном для поддержки старых версий библиотеки OpenSSL. Возможно, имеет смысл установить более высокое значение, если все применяемые у вас программные компоненты могут поддерживать более новые версии протокола.

Этот параметр можно задать только в файле **qhb.conf** или в командной строке сервера.

7.3.10 ssl_max_protocol_version (enum)

Задает максимальную версию протокола SSL/TLS, которую можно использовать. Допустимые значения такие же, как у параметра ssl_min_protocol_version, плюс пустая строка, означающая, что допускается любая версия протокола. По умолчанию разрешена любая версия. Установка максимальной версии протокола полезна прежде всего для тестирования или в случаях, когда у некоторых компонентов возникают проблемы при работе с более новым протоколом.

Этот параметр можно задать только в файле **qhb.conf** или в командной строке сервера.

7.3.11 ssl_dh_params_file (string)

Задает имя файла, содержащего параметры алгоритма Диффи-Хеллмана, используемые для так называемого эфемерного семейства DH-шифров SSL. По умолчанию этот параметр пуст, то есть используются параметры DH, скомпилированные по умолчанию. Применение нестандартных параметров DH повышает защиту, если злоумышленнику удается взломать хорошо известные скомпилированные параметры DH. Собственный файл с параметрами DH можно создать с помощью команды openssl dhparam -out dhparams.pem 2048.

Этот параметр можно задать только в файле **qhb.conf** или в командной строке сервера.

7.3.12 ssl_passphrase_command (string)

Задает внешнюю команду, которая будет вызываться, когда потребуется получить парольную фразу для расшифровки SSL-файла, например закрытого ключа. По умолчанию этот параметр пуст, то есть используется встроенный механизм запроса.

Эта команда должна вывести парольную фразу на устройство стандартного вывода и завершиться с кодом 0. В значении параметра ***%p*** заменяется строкой приглашения. (Напишите %%, чтобы вывести литерал %). Обратите внимание, что строка приглашения, вероятно, будет содержать пробелы, поэтому потребуется заключить ее в кавычки. Если в конце имеется один перевод строки, он будет удален при выводе.

Команда не обязана запрашивать у пользователя парольную фразу. Она может прочитать ее из файла, получить из системной связки ключей или другими подобными способами. Ответственность за надлежащую безопасность выбранного механизма несет пользователь.

Этот параметр можно задать только в файле **qhb.conf** или в командной строке сервера.

7.3.13 ssl_passphrase_command_supports_reload (boolean)

Этот параметр определяет, будет ли заданная параметром **ssl_passphrase_command** команда запроса пароля также вызываться при

перезагрузке конфигурации, если для файла ключа нужна парольная фраза. Если этот параметр выключен (по умолчанию), то **ssl_passphrase_command** будет игнорироваться при перезагрузке, и конфигурация SSL не будет перезагружаться, если требуется парольная фраза. Этот параметр подходит для команды, которой для ввода пароля требуется терминал TTY, который может быть недоступен во время работы сервера. Включение этого параметра может быть целесообразно, если, например, пароль считывается из файла.

Этот параметр можно задать только в файле **qhb.conf** или в командной строке сервера.

8 РЕЗЕРВНОЕ КОПИРОВАНИЕ И ВОССТАНОВЛЕНИЕ

Базы данных СУБД «Квант-Гибрид 1.5» должны регулярно подвергаться резервному копированию. Существует три принципиально разных подхода к резервному копированию данных СУБД «Квант-Гибрид 1.5»:

– SQL-дамп.
– Резервное копирование на уровне файловой системы.
– Непрерывное архивирование.

8.1 SQL-дамп

Метод дампа состоит в том, чтобы создать файл с SQL-командами, которые при применении на сервере воссоздают базу данных в том же состоянии, в котором она находилась во время создания дампа. СУБД «Квант-Гибрид 1.5» предоставляет для этой цели служебную программу ***qhb_dump***. Простейшее использование этой программы выглядит следующим образом:

```
qhb_dump dbname > dumpfile
```

qhb_dump записывает свой результат в стандартный вывод. В то время как вышеуказанная команда создаёт текстовый файл, ***qhb_dump*** может создавать файлы в других форматах, которые обеспечивают параллелизм и более детальный контроль над восстановлением объектов.

qhb_dump — это клиентское приложение СУБД «Квант-Гибрид 1.5». Это означает, что можно выполнить процедуру резервного копирования с любого удалённого хоста, который имеет доступ к базе данных. ***qhb_dump*** не работает с использованием специальных привилегий. В частности, требуется доступ на чтение ко всем таблицам, для которых выполняется резервное копирование, поэтому для резервного копирования всей базы данных почти всегда требуется запуск с правами суперпользователя базы данных.

Если у вас недостаточно прав для резервного копирования всей базы данных, вы всё равно можете создавать резервные копии тех частей базы данных, к которым у вас есть доступ, используя такие параметры, как -n schema или -t table.

Чтобы указать, к какому серверу баз данных должен обращаться ***qhb_dump***, используйте параметры командной строки -h host и -p port. Хост по умолчанию — это локальный хост или то значение, которое указано в переменной среды PGHOST. Точно так же порт по умолчанию указывается переменной окружения PGPORT или, если она не задана, значением, принятым по умолчанию.

qhb_dump по умолчанию соединяется с базой данных используя имя пользователя, которое совпадает с текущим именем пользователя операционной системы. Чтобы переопределить это, либо укажите опцию -U либо установите переменную окружения PGUSER. Помните, что соединения ***qhb_dump*** подчиняются обычным механизмам аутентификации клиента.

Важное преимущество ***qhb_dump*** перед другими методами резервного копирования, описанными ниже, заключается в том, что выходные данные ***qhb_dump***, как правило, могут загружаться в более новые версии СУБД «Квант-Гибрид 1.5», тогда как резервные копии на уровне файлов и непрерывное архивирование являются исключительно специфичными для версии сервера. ***qhb_dump*** также является единственным методом, который будет работать при переносе базы данных на другую архитектуру компьютера, например при переходе с 32-разрядного на 64-разрядный сервер.

Дампы, созданные ***qhb_dump***, являются внутренне непротиворечивыми, то есть дамп представляет собой снимок базы данных во время запуска ***qhb_dump***. ***qhb_dump*** не блокирует другие операции с базой данных во время работы.

Исключением являются те операции, которым требуется полная блокировка, например, большинство форм ALTER TABLE.

8.1.1 Восстановление дампа

Текстовые файлы, созданные ***qhb_dump***, предназначены для чтения программой ***psql***. Общая форма команды для восстановления дампа:

```
psql dbname < dumpfile
```

где dumpfile — это файл, содержащий вывод команды ***qhb_dump***. Эта команда не будет создавать базу данных dbname, поэтому вы должны создать её самостоятельно из template0 перед выполнением ***psql*** (например, с помощью createdb

-T template0 dbname). ***Psql*** поддерживает параметры, аналогичные ***qhb_dump***, для указания сервера базы данных, к которому нужно подключиться, и имени пользователя для использования. Дампы нетекстовых файлов восстанавливаются с помощью утилиты ***qhb_restore***.

Перед восстановлением дампа SQL все пользователи, которым принадлежали объекты или права на них в выгруженной базе данных, уже должны существовать. Если их нет, при восстановлении не удастся воссоздать объекты с первоначальным владельцем и / или правами.

По умолчанию сценарий ***psql*** будет продолжать выполняться после возникновения ошибки SQL. Если запустить ***psql*** с переменной ON_ERROR_STOP установленной для изменения этого поведения, то ***psql*** завершится кодом 3, если возникает ошибка SQL:

```
psql --set ON_ERROR_STOP=on dbname < dumpfile
```

Будет только частично восстановленная база данных. В качестве альтернативы можно указать, что весь дамп должен быть восстановлен за одну транзакцию, поэтому восстановление будет либо полностью завершено, либо полностью отменено. Этот режим может быть указан путём передачи параметров командной строки -1 или --single-transaction в ***psql***. При использовании этого режима даже небольшая ошибка может послужить причиной отката восстановления, которое уже выполняется в течение многих часов. Это может быть предпочтительнее, чем ручная очистка сложной базы данных после частично восстановленного дампа.

Возможность ***qhb_dump*** и ***psql*** использовать каналы ввода/вывода позволяет скопировать базу данных напрямую с одного сервера на другой, например:

qhb_dump -h host1 dbname | psql -h host2 dbname

Необходимо обратить особое внимание: дампы, создаваемые ***qhb_dump***, относятся к template0. Это означает, что любые языки, процедуры и т. д., добавленные через template1, также будут выгружены ***qhb_dump***. В результате при восстановлении, если вы используете настроенный template1, вы должны создать пустую базу данных из template0, как в примере выше.

После восстановления резервной копии целесообразно запустить ANALYZE для каждой базы данных, чтобы оптимизатор запросов имел статистику.

8.1.2 Использование qhb_dumpall

qhb_dump одновременно создаёт дамп только одной базы данных, который не содержит информацию о ролях или табличных пространствах (потому что они относятся к экземпляру, а не к базе данных). Для создания удобного дампа всего содержимого экземпляра базы данных предусмотрена программа ***qhb_dumpall***. ***qhb_dumpall*** выполняет резервное копирование каждой базы данных в данном кластере, а также сохраняет данные для всего кластера, такие как определения ролей и табличных пространств. Простой вариант использования этой программы:

```
qhb_dumpall > dumpfile
```

Полученный дамп можно восстановить с помощью ***psql***:

```
psql -f dumpfile qhb
```

Дополнительная информация.

Можно указать любое существующее имя базы данных для запуска, но если загружать в пустой экземпляр, то целесообразно использовать qhb.

Восстановление дампа ***qhb_dumpall*** необходимо производить с правами суперпользователя, поскольку это требуется для восстановления информации о ролях и табличных пространствах. Если вы используете табличные пространства, убедитесь, что их пути в дампе соответствуют новой среде.

qhb_dumpall работает, выполняя команды для воссоздания ролей, табличных пространств и пустых баз данных, затем вызывая ***qhb_dump*** для каждой базы данных. Это означает, что хотя каждая база данных будет внутренне согласованной, снимки разных баз данных не синхронизируются.

Данные всего экземпляра могут быть выгружены отдельно, используя опцию --globals-only. Это необходимо для полного резервного копирования экземпляра при выполнении команды ***qhb_dump*** в отдельных базах данных.

8.1.3 Обработка больших баз данных

Некоторые операционные системы имеют ограничения по максимальному размеру файла, которые вызывают проблемы при создании больших выходных

файлов ***qhb_dump***. ***qhb_dump*** может записывать в стандартный вывод, поэтому вы можете использовать стандартные инструменты Unix для решения этой потенциальной проблемы. Есть несколько возможных методов:

Используйте сжатые дампы. Вы можете использовать вашу предпочитаемую программу сжатия, например, gzip:

```
qhb_dump dbname | gzip > filename.gz
```

Восстановление дампа с помощью:

```
gunzip -c filename.gz | psql dbname
```

или:

```
cat filename.gz | gunzip | psql dbname
```

Команда split позволяет разбить вывод на более мелкие файлы, приемлемые по размеру для базовой файловой системы. Например, чтобы сделать куски по 1 мегабайту:

```
qhb_dump dbname | split -b 1m - filename
```

Восстановление дампа с помощью:

```
cat filename* | psql dbname
```

Используйте специальный формат дампа ***qhb_dump***. Если СУБД «Квант-Гибрид 1.5» был собран в системе с установленной библиотекой сжатия zlib, пользовательский формат дампа будет сжимать данные по мере их записи в выходной файл. Это приведет к размеру файла дампа, аналогичному использованию gzip, но у него есть дополнительное преимущество: таблицы могут восстанавливаться выборочно. Следующая команда создаёт дамп базы данных с использованием специального формата дампа:

```
qhb_dump -Fc dbname > filename
```

Дамп специального формата не является сценарием для ***psql***, а должен быть восстановлен с помощью ***qhb_restore***, например:

```
qhb_restore -d dbname filename
```

Для очень больших баз данных вам может потребоваться объединить split с одним из двух других подходов.

– Используйте функцию параллельного дампа ***qhb_dump***. Чтобы ускорить дамп большой базы данных, вы можете использовать параллельный

режим ***qhb_dump***. Этот режим позволит сбросить несколько таблиц одновременно. Можно контролировать количество потоков с помощью параметра -j. Параллельные дампы поддерживаются только для архива в формате каталога.

```
qhb_dump -j num -F d -f out.dir dbname
```

– Можно использовать qhb_restore -j для параллельного восстановления дампа. Это будет работать для любого архива, созданном в специальном формате или в формате каталога, независимо от того, был ли он создан с помощью qhb_dump -j.

8.2 Резервное копирование на уровне файловой системы

Альтернативная стратегия резервного копирования заключается в прямом копировании файлов, которые СУБД «Квант-Гибрид 1.5» использует для хранения данных в базе данных; Вы можете использовать любой метод для резервного копирования файловой системы; например:

```
tar -cf backup.tar /var/lib/qhb/data
```

Однако есть два ограничения, которые делают этот метод непрактичным или, по крайней мере, уступают методу ***qhb_dump***:

– Сервер базы данных должен быть выключен, чтобы получить правильную резервную копию. Промежуточные меры, такие как запрещение всех подключений, работать не будут (отчасти потому, что tar и подобные инструменты не делают атомарный снимок состояния файловой системы, но также из-за внутренней буферизации внутри сервера). Необходимо выключить сервер перед восстановлением данных.

– Если вы вникли в детали структуры файловой системы базы данных, у вас может возникнуть соблазн попытаться выполнить резервное копирование или восстановление только определенных отдельных таблиц или баз данных из их соответствующих файлов или каталогов. Это не будет работать, потому что информация, содержащаяся в этих файлах, не может быть использована без файлов журнала транзакций pg_xact/*,

которые содержат статус фиксации всех транзакций. Файл таблицы может использоваться только с этой информацией. Конечно, также невозможно восстановить только таблицу и связанные с pg_xact данные, потому что это сделает все остальные таблицы в экземпляре базы данных бесполезными. Таким образом, резервные копии файловой системы работают только для полного резервного копирования и восстановления всего экземпляра базы данных.

Альтернативный подход к резервному копированию файловой системы заключается в создании «согласованного снимка» каталога данных, если файловая система поддерживает эту функцию (и вы уверены, что она реализована правильно). Типичная процедура - сделать «замороженный снимок» тома, содержащего базу данных, затем скопировать весь каталог данных (не только части, см. выше) из моментального снимка на устройство резервного копирования, а затем освободить замороженный снимок. Это будет работать даже во время работы сервера базы данных. Однако созданная таким образом резервная копия сохраняет файлы базы данных в таком состоянии, как если бы сервер базы данных был неправильно остановлен; поэтому, когда вы запускаете сервер базы данных с резервной копией данных, он будет считать, что предыдущий экземпляр сервера завершился аварийно, и применит данные журналов WAL.

Важно: обязательно включите файлы WAL в свою резервную копию.

Рекомендуется выполнить CHECKPOINT перед созданием снимка, чтобы сократить время восстановления.

Если база данных распределена по нескольким файловым системам, то способа получить строго одновременно замороженные снимки всех томов может не оказаться. Например, если файлы данных и журнал WAL находятся на разных дисках, или если табличные пространства находятся в разных файловых системах, может оказаться невозможным использование резервного копирования «моментального снимка», поскольку моментальные снимки должны быть одновременными. Внимательно прочитайте документацию по файловой системе, прежде чем доверять технологии согласованных снимков в таких ситуациях.

Если согласованные снимки невозможны, один из вариантов - отключить сервер базы данных на достаточно длительное время, чтобы скопировать все замороженные снимки. Другим вариантом является выполнение непрерывного архивирования (создание базовой резервной копии), поскольку такие резервные копии не пострадают от изменений файловой системы во время резервного копирования. Это требует работы непрерывного архивирования на время процесса резервного копирования. Восстановление выполняется с использованием непрерывного восстановления архива.

Другой вариант – использовать rsync для резервного копирования файловой системы. Для этого сначала нужно запустить rsync во время работы сервера базы данных, а затем завершить работу сервера базы данных на время, достаточное, чтобы выполнить rsync --checksum. (--checksum необходим, потому что rsync различает время с точностью до секунды). Второй запуск rsync отработает быстрее, чем первый, потому что ему останется относительно мало данных для передачи, и конечный результат будет согласованным, поскольку сервер был выключен. Этот метод позволяет выполнять резервное копирование файловой системы с минимальным временем простоя.

Обратите внимание, что резервная копия файловой системы обычно больше, чем дамп SQL. (Например, qhb_dump не нужно записывать содержимое индексов, только команды для их восстановления). Однако создание резервной копии файловой системы может выполняться быстрее.

8.3 Непрерывное архивирование и восстановление на момент времени (PITR)

СУБД «Квант-Гибрид 1.5» поддерживает журнал упреждающей записи (WAL) в подкаталоге pg_wal/ каталога данных экземпляра. В журнал записываются все изменения, внесённые в файлы данных базы. Этот журнал существует главным образом в целях обеспечения безопасности при сбоях: в случае сбоя системы базы данных могут быть восстановлены до состояния согласованности путём «воспроизведения» записей журнала, созданных с момента последней контрольной точки. При этом, наличие журнала позволяет использовать третью стратегию

резервного копирования баз данных: можно объединить резервное копирование на уровне файловой системы с резервным копированием файлов WAL. Если требуется восстановление, то восстанавливается резервная копия файловой системы, а затем воспроизводятся изменения из резервных копий файлов WAL, чтобы привести систему к актуальному состоянию. Этот подход сложнее в администрировании, чем любой из предыдущих подходов, но он имеет некоторые существенные преимущества:

- не требуется идеально согласованное резервное копирование файловой системы в качестве отправной точки. Любое внутреннее несоответствие в резервной копии будет исправлено путём воспроизведения журнала (это существенно не отличается от того, что происходит во время восстановления после сбоя). Поэтому не нужна возможность создания снимков файловой системы, просто tar или аналогичный инструмент архивирования.
- Поскольку можно комбинировать неограниченно длинную последовательность файлов WAL для воспроизведения, непрерывное резервное копирование может быть достигнуто простым продолжением архивирования файлов WAL. Это особенно ценно для больших баз данных, где не всегда удобно делать полное резервное копирование.
- Нет необходимости воспроизводить записи WAL до самого конца. Можно остановить воспроизведение в любой момент и получить согласованный снимок базы данных на заданный момент времени. Таким образом, этот метод поддерживает восстановление на определенный момент времени: восстановление базы данных до её состояния возможно в любое время с момента создания резервной копии базы.
- Если непрерывно передавать серии файлов WAL на другой компьютер, который был загружен с одними и теми же базовыми данными резервной копии, то появляется система тёплого резервирования: в любой момент можно запустить второй сервер, и у него будет практически текущая копия база данных.

qhb_dump и ***qhb_dumpall*** не создают резервные копии на уровне файловой системы и не могут использоваться как часть решения для непрерывного архивирования. Это логические дампы, и они не содержат достаточно информации для использования при воспроизведении WAL.

Как и в случае простого метода резервного копирования файловой системы, этот метод может поддерживать только восстановление всего экземпляра базы данных, но не его части. Кроме того, для этого требуется большое архивное хранилище: базовая резервная копия может быть громоздкой, а занятая система будет генерировать много мегабайт трафика WAL, который необходимо архивировать. Тем не менее, это предпочтительный метод резервного копирования во многих ситуациях, когда требуется высокая надёжность.

Для успешного восстановления с использованием непрерывного архивирования (также называемого «оперативным резервным копированием» многими разработчиками баз данных) нужна непрерывная последовательность архивированных WAL-файлов история которой, по крайней мере, начинается со времени начала резервного копирования.

Для начала необходимо настроить и протестировать процедуру архивации файлов WAL, прежде чем делать первую базовую резервную копию. Механизм архивирования файлов WAL рассмотрен в разделе ниже.

8.3.1 Настройка архивирования WAL

В абстрактном смысле работающая система СУБД «Квант-Гибрид 1.5» создаёт бесконечно длинную последовательность записей WAL. Система физически делит эту последовательность на файлы сегментов WAL, которые обычно имеют размер 16 МБ (хотя размер сегмента можно изменить во время создания нового кластера базы данных через параметры initdb). Сегменты получают числовые имена, которые отражают их положение в абстрактной последовательности WAL. Когда архивация WAL не используется, система обычно создаёт только несколько файлов сегментов, а затем «перезаписывает» их, меняя имена ставших ненужными файлов WAL на новые, с большими номерами. Предполагается, что файлы сегментов, содержимое

которых предшествует последней контрольной точке, больше не представляют интереса и могут быть использованы заново.

При архивировании данных WAL нужно захватить содержимое каждого файла сегмента после его заполнения и сохранить эти данные где-то перед тем, как файл сегмента будет повторно использован. В зависимости от приложения и доступного оборудования, может быть много разных способов «сохранения данных где-то»: есть возможность скопировать файлы сегментов в каталог, смонтированный NFS на другом компьютере, записать на ленточный накопитель (гарантируя, что есть способ идентификации исходного имени каждого файла), или собрать вместе и записать на компакт-диски, или что-то ещё. Чтобы предоставить администратору базы данных гибкость, СУБД «Квант-Гибрид 1.5» старается не делать никаких предположений о том, как будет осуществляться архивирование. Вместо этого СУБД «Квант-Гибрид 1.5» позволяет администратору указать команду оболочки, которая будет выполняться, чтобы скопировать заполненный файл сегмента в место назначения. Команда может быть простой, например cp, или может вызывать сложный сценарий оболочки - все зависит от администратора.

Чтобы включить архивирование WAL, установите для параметра конфигурации wal_level значение replica или выше, для параметра archive_mode – значение on и укажите команду оболочки для использования в параметре конфигурации archive_command. Эти настройки всегда будут помещаться в файл qhb.conf. В archive_command %p заменяется путём к файлу для архивирования, а %f заменяется только именем файла. (Путь указывается относительно текущего рабочего каталога, т. е. каталога данных экземпляра). Используйте %% если вам нужно вставить символ % в команду. Самая простая полезная команда выглядит примерно так:

```
archive_command = 'test ! -f /mnt/server/archivedir/%f && cp %p
/mnt/server/archivedir/%f'  # Unix
```

она скопирует архивируемые сегменты WAL в каталог /mnt/server/archivedir. После замены параметров %p и %f действительная команда может выглядеть следующим образом:

```
test ! -f /mnt/server/archivedir/000000010000000A900000065 && cp
qhb_wal/000000010000000A900000065
/mnt/server/archivedir/000000010000000A900000065
```

Аналогичная команда будет сгенерирована для каждого нового файла, подлежащего архивированию.

Команда архивирования будет выполняться под управлением того же пользователя, на котором работает сервер СУБД «Квант-Гибрид 1.5». Поскольку серия архивируемых файлов WAL содержит практически всё, что есть в базе данных, необходимо быть уверенным, что заархивированные данные защищены от посторонних глаз. Например, архивировать в каталог, с ограниченными правами доступа на чтение для группы и других пользователей.

Важно, чтобы команда архивирования возвращала нулевой статус выхода, только если копирование произошло успешно. Получив нулевой результат, СУБД «Квант-Гибрид 1.5» предположит, что файл был успешно заархивирован, и удалит или переиспользует его. Ненулевой статус указывает СУБД «Квант-Гибрид 1.5», что файл не был заархивирован и будут периодически предприниматься попытки его архивации заново, пока это не удастся.

Обычно команда архивирования должна предотвращать перезапись любого ранее существующего архивного файла. Это важная функция безопасности для сохранения целостности вашего архива в случае ошибки администратора (например, отправка вывода двух разных серверов в один и тот же каталог архива).

Рекомендуется протестировать предложенную вами команду архивирования, чтобы убедиться, что она действительно не перезаписывает существующий файл, а если это так, то возвращает ненулевой статус. Приведённый выше пример команды для Unix обеспечивает это путём включения отдельного шага test. На некоторых платформах Unix в cp есть такие флаги, как -i, которые можно использовать для выполнения тех же операций менее явно, но на них нельзя полагаться, не убедившись, что возвращён правильный код статуса. В частности, GNU cp вернет нулевой код статуса, когда используется -i и целевой файл уже существует, что не является желаемым поведением.

При разработке вашей конфигурации архивации следует учесть, что произойдёт, если команда архивирования не будет выполнена повторно. Потому что при определенных обстоятельствах могут потребоваться вмешательства оператора, или архиву может не хватить места. Чтобы спорная ситуация могла быть разрешена достаточно быстро, необходимо убедиться в том, что о любой ошибке или запросе, пользователю приходят оповещения. В противном случае, pg_wal/ будет продолжать заполняться файлами сегментов WAL, пока ситуация не будет разрешена. Если файловая система, содержащая pg_wal/ заполняется, СУБД «Квант-Гибрид 1.5» завершит работу аварийно. Никакие зафиксированные транзакции не будут потеряны, но база данных останется в автономном режиме, пока вы не освободите некоторое пространство.

Скорость команды архивации не имеет значения, пока она может соответствовать средней скорости, с которой ваш сервер генерирует данные WAL. Нормальная работа СУБД продолжается, даже если процесс архивирования немного отстаёт. Если архивирование значительно отстаёт, это увеличит объём данных, которые могут быть потеряны в случае аварии. Это также будет означать, что pg_wal/ будет содержать большое количество ещё не заархивированных файлов сегментов, которые в конечном итоге могут превысить доступное дисковое пространство. Рекомендуется следить за процессом архивации, чтобы убедиться, что он работает так, как вы рассчитываете.

При написании команды архивирования вы должны исходить из того, что имена файлов, подлежащих архивированию, могут быть длиной до 64 символов и могут содержать любую комбинацию букв ASCII, цифр и точек. Нет необходимости сохранять исходный относительный путь (%p), но необходимо сохранить имя файла (%f).

Обратите внимание, что несмотря на то, что архивация WAL позволяет восстановить любые изменения, внесенные в данные в базе данных СУБД «Квант-Гибрид 1.5», она не восстановит изменения, внесённые в файлы конфигурации (qhb.conf, qhb_hba.conf и qhb_ident.conf), так как они редактируются вручную, а не с помощью операций SQL.

Команда архивирования вызывается только для завершённых сегментов WAL. Следовательно, если сервер генерирует только небольшой трафик WAL (или имеет периоды простоя, когда это происходит), может возникнуть длительная задержка между завершением транзакции и ее безопасной записью в архивное хранилище. Чтобы установить ограничение на срок хранения не архивированных данных, вы можете установить archive_timeout, чтобы заставить сервер переключаться на новый файл сегмента WAL так часто, насколько это необходимо. Обратите внимание, что архивные файлы, которые архивируются раньше из-за принудительного переключения, имеют ту же длину, что и полностью заполненные файлы. Поэтому неразумно устанавливать очень короткое значение archive_timeout — это приведёт к переполнению вашего архивного хранилища.

Параметр archive_timeout установленный на минутный промежуток или около того обычно бывает корректен.

Кроме того, можно принудительно переключить сегмент с помощью pg_switch_wal если вы хотите, чтобы только что завершённая транзакция была заархивирована как можно скорее. Другие функции утилит, относящиеся к управлению WAL, перечислены в таблице ниже.

Таблица 3. Функции управления резервным копированием

Имя	**Тип ответа**	**Описание**
pg_create_restore_point(name text)	*pg_lsn*	*Создать именованную точку для выполнения восстановления**
pg_current_wal_flush_lsn()	*pg_lsn*	*Получить текущее местоположение сброса журнала предзаписи.*
pg_current_wal_insert_lsn()	*pg_lsn*	*Получить текущее местоположение вставки журнала предзаписи.*
pg_current_wal_lsn()	*pg_lsn*	*Получить текущее местоположение записи в журнал предзаписи.*

Имя	Тип ответа	Описание
pg_start_backup(label text [, fast boolean [, exclusive boolean]])	*pg_lsn*	*Подготовление к выполнению резервному копированию**
pg_stop_backup()	*pg_lsn*	*Завершить выполнение монопольного резервного копирования**
pg_stop_backup(exclusive boolean [, wait_for_archive boolean])	*set of record*	*Завершить выполнение монопольного резервного копирования**
pg_is_in_backup()	*bool*	*Истинно, если монопольное резервное копирование все еще выполняется.*
pg_backup_start_time()	*timestamp with time zone*	*Получить время запуска монопольного резервного копирования.*
pg_switch_wal()	*pg_lsn*	*Принудительное переключение на новый файл журнала с опережением записи**
pg_walfile_name(lsn pg_lsn)	*text*	*Преобразовать местоположение журнала предзаписи в имя файла.*
pg_walfile_name_offset(lsn pg_lsn)	*text, integer*	*Преобразовать местоположение журнала предзаписи в имя файла и десятичное смещение байта в нём.*
pg_wal_lsn_diff(lsn pg_lsn, lsn pg_lsn)	*numeric*	*Рассчитать разницу между двумя местоположениями журнала записи.*

Дополнительная информация.

* - (по умолчанию ограничено суперпользователями, но другим пользователям может быть предоставлено разрешение EXECUTE для запуска функции).

Когда wal_level – minimal некоторые команды SQL оптимизированы, чтобы избежать ведения журнала WAL. Если архивация или потоковая репликация были включены во время выполнения одного из этих операторов, WAL не будет содержать достаточно информации для восстановления архива (на восстановление после

аварийного завершения это не распространяется). По этой причине wal_level может быть изменен только при запуске сервера. Тем не менее, параметр archive_command может быть изменен с перезагрузкой файла конфигурации. Если есть необходимость временно остановить архивирование, один из способов сделать это - установить для archive_command пустую строку (’ ’). Это приведет к тому, что файлы WAL будут накапливаться в pg_wal/ пока не будет восстановлена рабочая команда archive_command.

8.3.2 Создание базовой резервной копии

Самый простой способ выполнить базовое резервное копирование - использовать инструмент qhb_basebackup. Он может создавать базовую резервную копию в виде обычных файлов или в виде архива tar. Альтернативой этому инструменту служит ***qbackup***. Эта программа позволяет сохранять ваши резервные копии в структурированный каталог, при этом поддерживается инкрементальное копирование и сжатие, а также параллельный режим работы.

Если требуется больше гибкости, чем могут обеспечить эти программы, то вы также можете сделать базовую резервную копию, используя низкоуровневый API.

Не стоит беспокоиться о количестве времени, необходимого для создания базовой резервной копии. Однако если вы обычно запускаете сервер с выключенными full_page_writes, вы можете заметить падение производительности во время резервного копирования, так как full_page_writes включается автоматически в режиме резервного копирования.

Чтобы в последующем использовать резервную копию, вам необходимо сохранить все файлы сегментов WAL, созданные во время и после резервного копирования файловой системы. Чтобы помочь вам в этом, процесс базового резервного копирования создаёт файл истории резервного копирования, который немедленно сохраняется в области архивации WAL. Этот файл назван как первый файл сегмента WAL, который необходим для резервного копирования файловой системы. Например, если начальный файл WAL - 000000010000123400055CD файл истории резервного копирования будет иметь имя, например, 000000010000123400055CD.007C9330.backup. Вторая часть имени файла обозначает

точную позицию в файле WAL и обычно может игнорироваться. После того, как архивации резервной копии файловой системы и файлы сегментов WAL, использованные во время резервного копирования (как указано в истории резервного копирования файла), все архивные сегменты WAL с численно меньшими именами больше не нужны для восстановления резервной копии файловой системы и могут быть удалены. При этом целесообразно сохранить несколько наборов резервных копий, чтобы быть уверенным, что возможно восстановить нужные данные.

Файл истории резервного копирования — это небольшой текстовый файл. Он содержит строку метки, которую вы дали qhb_basebackup, а также начальное и конечное время и имена начального и конечного сегментов WAL, относящихся к резервной копии. Если была использована метка для идентификации связанного файла дампа, то заархивированного файла истории достаточно, чтобы указать, какой файл дампа нужно восстановить.

Поскольку целесообразно хранить все архивные файлы WAL начиная с первой базовой резервной копии, интервал между повторными резервными копированиями копиями следует выбирать исходя из того, сколько памяти планируется потратить на архивные файлы WAL. Следует учитывать, сколько времени необходимо будет потратить на восстановление, так как система должна будет воспроизвести все эти сегменты WAL, что может занять некоторое время, если прошло много времени с момента последнего резервного копирования базы.

8.3.3 Создание базовой резервной копии с использованием API низкого уровня

Процедура создания базовой резервной копии с использованием низкоуровневых API-интерфейсов содержит на несколько шагов больше, чем метод qhb_basebackup или qbackup, но относительно проста. Эти шаги должны выполняться последовательно, и чтобы каждый шаг был успешен перед переходом к следующему шагу.

Резервные копии низкого уровня могут быть сделаны немонопольным способом.

8.3.3.1 Создание немонопольной резервной копии на низком уровне

Немонопольное резервное копирование низкого уровня позволяет запускать другие параллельные резервные копии (через API-интерфейс или qhb_basebackup).

– Убедитесь, что архивация WAL включена и работает.
– Подключитесь к серверу (не имеет значения, к какой базе данных) как пользователь с правами на запуск pg_start_backup (суперпользователь или пользователь, которому предоставлена привилегия EXECUTE для функции) и выполните команду:

```
SELECT pg_start_backup('label', false, false);
```

Где label - любая строка, которую вы хотите использовать в качестве идентификатора этой операции резервного копирования. Соединение, вызывающее pg_start_backup должно поддерживаться до конца резервного копирования, иначе резервное копирование будет автоматически прервано.

По умолчанию pg_start_backup может занять много времени до завершения. Это связано с тем, что он выполняет контрольную точку, а операции ввода-вывода, требуемые для этого, распределяются в интервале времени, равного по умолчанию половине вашего интервала между контрольными точками. Обычно это то, что и требуется, потому что это минимизирует влияние на обработку запросов. Если необходимо начать резервное копирование как можно скорее, измените второй параметр на true, который немедленно выдаст контрольную точку с использованием максимально возможного количества операций ввода-вывода.

Третий параметр false указывает pg_start_backup инициировать немонопольную базовую резервную копию.

– Выполните резервное копирование, используя любой удобный инструмент для резервного копирования файловой системы, такой как tar или cpio (не ***qhb_dump*** или ***qhb_dumpall***). Допустимо не останавливать нормальную работу базы данных, пока вы делаете это.

– В том же соединении, что и раньше, введите команду:

```
SELECT * FROM pg_stop_backup(false, true);
```

Это прекращает режим резервного копирования. На ведущем сервере также выполняется автоматическое переключение на следующий сегмент WAL. В режиме ожидания невозможно автоматическое переключение сегментов WAL, поэтому вы можете запустить pg_switch_wal, чтобы выполнить ручное переключение. Цель переключения заключается в том, чтобы последний файл сегмента WAL, записанный в течение интервала резервного копирования, был готов к архивированию.

pg_stop_backup вернет одну строку с тремя значениями. Второе из этих полей должно быть записано в файл с именем backup_label в корневом каталоге резервной копии. Третье поле должно быть записано в файл с именем tablespace_map, если поле не пустое. Эти значения важны для резервного копирования и должны быть записаны без изменений.

– Как только файлы сегментов WAL, активные во время резервного копирования, будут заархивированы, всё готово. Файл, идентифицируемый первым возвращаемым значением pg_stop_backup является последним сегментом, который требуется для формирования полного набора файлов резервных копий. Если параметр archive_mode включен и параметр wait_for_archive функции pg_stop_backup равен true, pg_stop_backup не выполнится до тех пор, пока не будет заархивирован последний сегмент. На ведомом сервере параметр archive_mode при этом должен иметь значение always. Архивирование этих файлов происходит автоматически, так как вы уже настроили archive_command. В большинстве случаев это происходит быстро, но рекомендуется следить за системой архивирования, чтобы убедиться в отсутствии задержек. Если процесс архивирования отстал из-за сбоев команды архивирования, он будет повторять попытки до тех пор, пока архив не будет успешно завершен и резервное копирование не будет завершено. Если необходимо установить ограничение по времени выполнения pg_stop_backup,

установите соответствующее значение statement_timeout. При этом необходимо учитывать, что, если pg_stop_backup завершит работу из-за этого, резервная копия может потерять целостность.

Если процесс резервного копирования отслеживает и гарантирует, что все файлы сегментов WAL, необходимые для резервного копирования, успешно заархивированы, то для параметра wait_for_archive (по умолчанию устанавливается значение true) можно установить значение false, чтобы pg_stop_backup завершался, как только запись остановки резервного копирования записывается в WAL. По умолчанию pg_stop_backup будет ждать, пока все WAL будут заархивированы, что может занять некоторое время. Эту опцию следует использовать с осторожностью: если архивирование WAL не контролируется должным образом, резервная копия может не включать все файлы WAL и, следовательно, будет неполной и не сможет быть восстановлена.

8.3.3.2 Создание монопольной резервной копии на низком уровне

Монопольный метод резервного копирования устарел и обычно его следует избегать.

Процесс для монопольной резервной копии в основном такой же, как и для немонопольной, но отличается в нескольких ключевых шагах. Этот тип резервного копирования может выполняться только на первичном сервере и не допускает одновременного резервного копирования. Более того, поскольку он создает файл метки резервной копии, как описано ниже, он может заблокировать автоматический перезапуск главного сервера после сбоя. С другой стороны, ошибочное удаление этого файла из резервной копии является распространенной ошибкой, которая может привести к серьезному повреждению данных. Если необходимо использовать этот метод, могут быть выполнены следующие шаги.

- Убедитесь, что архивация WAL включена и работает.
- Подключитесь к серверу как пользователь с правами на запуск pg_start_backup (суперпользователь или пользователь, которому предоставлена привилегия EXECUTE для функции) и выполните команду:

```
SELECT pg_start_backup('label');
```

где label - любая строка, которую вы хотите использовать для уникальной идентификации этой операции резервного копирования.

pg_start_backup создает файл метки резервной копии с именем backup_label в каталоге кластера с информацией о вашей резервной копии, включая время начала и строку метки.

Функция также создает файл карты табличных пространств, называемый tablespace_map, в каталоге кластера с информацией о символических ссылках табличных пространств в pg_tblspc/, если присутствует одна или несколько таких ссылок.

По умолчанию вызов pg_start_backup может занять довольно продолжительное время. Это связано с тем, что при этом выполняется контрольная точка, и ввод-вывод, необходимый для выполнения контрольной точки, будет распределен в течение значительного периода времени (по умолчанию в течение половины установленного интервала между контрольными точками). Если вы хотите начать резервное копирование как можно скорее, используйте:

```
SELECT pg_start_backup ('label', true);
```

Это заставляет контрольную точку выполниться как можно быстрее.

– Выполните резервное копирование, используя любой удобный инструмент для резервного копирования файловой системы, такой как tar или cpio (не ***pg_dump*** или ***pg_dumpall***).

  Как отмечалось выше, если во время резервного копирования происходит сбой сервера, перезапуск может оказаться невозможным, пока файл backup_label не будет удален вручную из каталога PGDATA. Обратите внимание: нельзя удалять файл backup_label при восстановлении резервной копии, поскольку это приведет к повреждению.

– Снова подключитесь к базе данных как пользователь с правами на запуск pg_stop_backup (суперпользователь или пользователь, которому

предоставлена привилегия EXECUTE для этой функции) и выполните команду:

```
SELECT pg_stop_backup();
```

Эта функция завершает режим резервного копирования и выполняет автоматический переход к следующему сегменту WAL. Цель переключения заключается в том, чтобы последний сегмент WAL, записанный в течение интервала резервного копирования, был бы готов к архивированию.

– Как только файлы сегментов WAL, активные во время резервного копирования, будут заархивированы, процедура резервирования будет завершена. WAL-файл, идентифицируемый результатом выполнения pg_stop_backup, является последним сегментом, который требуется для формирования полного набора файлов резервных копий. Если опция archive_mode включена, pg_stop_backup не завершается, пока не будет заархивирован последний сегмент. Архивирование этих файлов происходит автоматически, поскольку уже должна быть настроена команда archive_command. В большинстве случаев это происходит быстро, но рекомендуется следить за системой архивирования, чтобы убедиться в отсутствии задержек. Если процесс архивирования отстал из-за сбоев команды архивирования, он будет повторять попытки до тех пор, пока архивация не будет успешно завершена, и только в этом случае резервное копирование закончится.

При использовании монопольного режима резервного копирования необходимо убедиться, что pg_stop_backup успешно завершится в конце резервного копирования. Даже в случае сбоя самой резервной копии, например, из-за недостатка дискового пространства, сбой вызова pg_stop_backup оставит сервер в режиме резервного копирования на неопределенное время, что приведет к сбою будущих резервных копий и увеличит риск сбоя перезапуска сервера при остающемся файле backup_label.

8.3.3.3 Резервное копирование каталога данных

Некоторые инструменты резервного копирования файловой системы выдают предупреждения или ошибки, если файлы, которые они пытаются скопировать, изменяются в процессе копирования. При создании базовой резервной копии активной базы данных это нормальная ситуация, а не ошибка. Однако необходимо убедиться, что вы можете отличить логи такого рода от реальных ошибок. Например, некоторые версии rsync возвращают отдельный код завершения для «исчезнувших исходных файлов», и вы можете написать скрипт, чтобы принять этот код завершения как случай, не связанный с ошибкой. Кроме того, некоторые версии GNU tar возвращают код ошибки, неотличимый от фатальной ошибки, если файл был усечён во время его копирования через tar. GNU tar версии 1.16 и выше завершается с кодом 1, если файл был изменен во время резервного копирования, и 2 для других ошибок.

В GNU tar версии 1.23 и более поздних версиях вы можете использовать параметры предупреждений --warning=no-file-changed, --warning=no-file-removed, чтобы скрыть соответствующие предупреждающие сообщения.

Убедитесь, что ваша резервная копия содержит все файлы из каталога экземпляра базы данных (например, ***/var/lib/qhb/data***). Если вы используете табличные пространства, которые не находятся под этим каталогом, будьте осторожны и включайте их (убедитесь, что ваша резервная копия архивирует символические ссылки в виде ссылок, иначе восстановление повредит ваши табличные пространства).

Необходимо исключить из резервной копии файлы в подкаталоге экземпляра ***pg_wal/***. Эта небольшая корректировка имеет смысл, поскольку снижает риск ошибок при восстановлении. Это легко организовать, если ***pg_wal/*** является символической ссылкой, указывающей куда-то за пределы каталога экземпляра, что в любом случае является обычной настройкой из соображений производительности. Вы также можете исключить qhbmaster.pid и postmaster.opts, которые записывают информацию о работающем qhbmaster, а не о qhbmaster, который в конечном итоге будет использовать эту резервную копию. Эти файлы могут запутать ***qhb_ctl***.

Также стоит исключать из резервной копии каталог **pg_replslot/** кластера, чтобы слоты репликации, существующие на главном сервере, не попадали в копию. В противном случае последующее использование резервной копии на резервном сервере может привести к неопределенному сроку хранения файлов WAL в резервной системе и, возможно, к раздутию на главном сервере, если включена обратная связь с горячим резервом, поскольку клиенты, использующие эти слоты репликации, все равно будут подключаться и обновлять слоты на мастере, а не на резервном сервере. Даже если резервная копия предназначена только для использования при создании нового мастера, копирование слотов репликации не будет особенно полезным, поскольку содержимое этих слотов, вероятно, будет сильно устаревшим к тому времени, когда новый мастер войдет в сеть.

Содержимое каталогов pg_dynshmem/, pg_notify/, pg_serial/, pg_snapshots/, pg_stat_tmp/ и pg_subtrans/ (но не сами каталоги) может быть исключено из резервной копии, поскольку оно будет инициализировано при запуске qhbmaster. Если stats_temp_directory установлен и указывает на подкаталог внутри каталога данных, то содержимое этого каталога также может быть опущено.

Любой файл или каталог, начинающийся с pgsql_tmp может быть опущен из резервной копии. Эти файлы удаляются при запуске qhbmaster, а каталоги восстанавливаются по мере необходимости.

Файлы pg_internal.init могут быть исключены из резервной копии всякий раз, когда найден файл с таким именем. Эти файлы содержат данные кэша отношений, которые всегда восстанавливаются при восстановлении.

Файл метки резервной копии содержит строку метки, которую вы задали во время вызова pg_start_backup, а также имя начального файла WAL. Файл карты табличных пространств содержит имена символических ссылок, поскольку они существуют в каталоге pg_tblspc/ и полный путь каждой символической ссылки.

Также возможно сделать резервную копию, когда сервер остановлен, например с помощью qbackup. В этом случае вы не можете использовать pg_start_backup или pg_stop_backup, и поэтому должны использовать собственные устройства для отслеживания того, какая резервная копия является какой, и как далеко назад идут

связанные файлы WAL (qbackup делает это автоматически). Как правило, лучше следовать процедуре непрерывного архивирования, описанной выше.

8.3.4 Восстановление с помощью непрерывного архива

Процедура восстановления из резервной копии:

1. Остановите сервер, если он работает.
2. При наличии места для этого, скопируйте весь каталог данных экземпляра и все табличные пространства во временную папку на случай, если они понадобятся вам позже. Необходимо проверить, что в вашей системе достаточно свободного места для хранения двух копий вашей базы данных. Если у вас недостаточно места, вы должны как минимум сохранить содержимое подкаталога экземпляра pg_wal, так как он может содержать журналы, которые не были заархивированы до завершения работы системы.
3. Удалите все существующие файлы и подкаталоги в каталоге данных экземпляра и в корневых каталогах всех используемых вами табличных пространств.
4. Восстановите файлы базы данных из резервной копии вашей файловой системы. Убедитесь, что они восстановлены с правами владельца (пользователя системы баз данных, а не root) и с правами доступа. Если вы используете табличные пространства, вы должны убедиться, что символические ссылки в pg_tblspc/ были правильно восстановлены.
5. Удалите все файлы, присутствующие в pg_wal/, которые были получены из резервной копии, потому что, вероятно, они устарели и не являются актуальными. Если вы не архивировали pg_wal/, то заново создайте его с надлежащими разрешениями, соблюдая осторожность, чтобы убедиться, что вы восстановили его в качестве символической ссылки, если вы его так настроили ранее.
6. Если у вас есть разархивированные файлы сегментов WAL, которые вы сохранили на шаге 2, скопируйте их в pg_wal/. Лучше всего их

копировать, а не перемещать, т.к. у вас остаются неизмененные файлы, и, если возникает проблема, и придется начать заново.

7. Задайте параметры конфигурации восстановления в qhb.conf и создайте файл recovery.signal в каталоге данных экземпляра. Вы также можете временно изменить qhb_hba.conf, чтобы обычные пользователи не могли подключаться, пока вы не убедитесь, что восстановление прошло успешно.
8. Запустите сервер. Сервер перейдет в режим восстановления и продолжит чтение нужных ему архивированных файлов WAL. Если восстановление будет прервано из-за внешней ошибки, сервер можно просто перезапустить, он продолжит восстановление. По завершении процесса восстановления сервер удалит recovery.signal (для предотвращения случайного повторного входа в режим восстановления позже), а затем начнет обычные операции с базой данных.
9. Проверьте содержимое базы данных, чтобы убедиться, что вы восстановились до нужного состояния. Если нет, вернитесь к шагу 1. Если все в порядке, разрешите пользователям подключаться, восстановив qhb_hba.conf в нормальном состоянии.

Ключевой частью всего этого является настройка конфигурации восстановления, которая описывает, как вы хотите восстановить и как далеко должно пройти восстановление. Единственное, что необходимо указать, это restore_command, которая сообщает СУБД «Квант-Гибрид 1.5», как извлекать заархивированные сегменты файла WAL. Как и archive_command, это командная строка оболочки. Она может содержать %f, который заменяется именем нужного файла журнала, и %p, который заменяется путём, по которому нужно скопировать файл журнала (путь указывается относительно текущего рабочего каталога, т.е. каталога данных экземпляра). Напишите %%, если вам нужно вставить фактический символ % в команду. Самая простая команда выглядит примерно так:

```
restore_command = 'cp /mnt/server/archivedir/%f %p'
```

который скопирует ранее заархивированные сегменты WAL из каталога /mnt/server/archivedir. Возможно использовать что-то гораздо более сложное, например, сценарий оболочки, который просит оператора смонтировать соответствующую ленту. qbackup делает это автоматически, используя встроенную подкоманду.

Важно, чтобы команда возвращала ненулевой статус выхода при ошибке. Эта команда будет вызвана так же для запроса файлов, которых нет в архиве, она должна вернуть ненулевое значение, если их нет. Однако это не должно считаться за ошибку. Исключением является, если команда была прервана сигналом (отличным от SIGTERM, который используется для отключения сервера базы данных) или ошибкой оболочки (например, команда не найдена), то восстановление будет прервано, и сервер не запустится.

Не все запрошенные файлы будут файлами сегментов WAL; также следует ожидать запросов на файлы с суффиксом .history. Также помните, что базовое имя пути %p будет отличаться от %f, не ожидайте, что они будут взаимозаменяемыми.

Сегменты WAL, которые нельзя найти в архиве, будут искать в pg_wal/; это позволяет использовать последние неархивированные сегменты. Однако сегменты, доступные из архива, будут использоваться вместо файлов в pg_wal/, так как имеют приоритет над ними.

Обычно восстановление выполняется через все доступные сегменты WAL, тем самым восстанавливая базу данных до текущего момента времени (или максимально близко, учитывая доступные сегменты WAL). Поэтому нормальное восстановление заканчивается сообщением «файл не найден», точный текст сообщения об ошибке зависит от выбранного вами параметра restore_command. Вы можете увидеть сообщение об ошибке в начале восстановления для файла с именем что-то вроде 00000001.history. Это нормально, и не указывает на проблему в простых ситуациях восстановления.

Если необходимо восстановить данные до некоторого предыдущего момента времени, укажите необходимую точку остановки. Вы можете указать точку остановки, известную как «целевая точка восстановления», либо по дате/времени,

названной точки восстановления, либо по завершению транзакции с определенным идентификатором.

Точка остановки должна быть указана после времени окончания основного резервного копирования, то есть времени окончания pg_stop_backup. Нельзя использовать базовую резервную копию для восстановления до того времени, когда эта резервная копия выполнилась. Чтобы восстановиться до такого времени, вы должны вернуться к своей предыдущей базовой резервной копии и начать восстановление оттуда.

В команде restore в qbackup существуют опции, позволяющие выбрать эти настройки. Ими можно будет воспользоваться только если резервная копия была сделана при помощи backup

Если восстановление обнаружит поврежденные данные WAL, восстановление будет остановлено в этот момент, и сервер не запустится. В таком случае процесс восстановления может быть перезапущен с самого начала с указанием «цели восстановления» до точки повреждения, чтобы восстановление могло завершиться нормально. Если восстановление завершится неудачно по внешней причине, такой как сбой системы или если архив WAL стал недоступен, то восстановление можно перезапустить, и оно будет перезапущено почти с того места, где произошел сбой. Перезапуск восстановления работает так же, как контрольная точка в обычной работе: сервер периодически передает все свое состояние на диск, а затем обновляет файл pg_control, чтобы указать, что уже обработанные данные WAL не нужно снова сканировать.

8.3.5 Временная шкала

В СУБД «Квант-Гибрид 1.5» есть понятие временных шкал. Когда восстановление архива завершается, создаётся новая временная шкала для определения последовательности записей WAL, созданной после этого восстановления. Идентификационный номер шкалы времени является частью имён файлов сегментов WAL, поэтому новая шкала времени не перезаписывает данные WAL, созданные предыдущими временными шкалами. Фактически возможно архивировать много различных временных шкал.

Когда создаётся временная шкала, СУБД «Квант-Гибрид 1.5» создаёт файл «историю временной шкалы», который показывает, с какой шкалы времени он вышел и когда, от какой временной шкалы он ответвился и когда. Эти файлы истории необходимы, чтобы позволить системе выбирать правильные файлы сегментов WAL при восстановлении из архива, содержащего несколько временных шкал. Поэтому они архивируются в область архива WAL точно так же, как файлы сегментов WAL. Файлы истории — это небольшие текстовые файлы, поэтому их целесообразно хранить на протяжении долгого времени (в отличие от больших файлов сегментов). При необходимости, можно добавлять комментарии в файлы истории, чтобы записывать свои собственные заметки о том, как и почему была создана эта конкретная временная шкала.

По умолчанию восстановление происходит на ту же временную шкалу, которая была текущей, когда была сделана базовая резервная копия. Если вам необходимо восстановить какую-либо дочернюю линию времени (то есть необходимо вернуться к некоторому состоянию, которое само было получено после попытки восстановления), нужно указать целевой идентификатор временной шкалы в recovery_target_timeline.

По умолчанию, восстановление происходит до последней временной шкалы, найденной в архиве. Если вы хотите восстановить временную шкалу, которая была текущей на момент создания базовой резервной копии, или определенную дочернюю временную шкалу (то есть вы хотите вернуться к некоторому состоянию, которое само было сгенерировано после попытки восстановления), вам необходимо указать опцию current или идентификатор целевой временной шкалы в recovery_target_timeline.

8.4 Рекомендации и примеры

В данном разделе приведены рекомендации по настройке непрерывного архивирования.

8.4.1 Автономные горячие резервные копии

Для создания автономных оперативных резервных копий можно использовать средства резервного копирования СУБД «Квант-Гибрид 1.5». Это резервные копии, которые нельзя использовать для восстановления на определенный момент времени, но, как правило, ими гораздо быстрее выполнять резервное копирование и восстановление, чем дампы ***qhb_dump***. Они намного больше, чем дампы ***qhb_dump***, поэтому в некоторых случаях преимущество в скорости может быть сведено на нет.

Как и в случае базовых резервных копий, самый простой способ создать автономное горячее резервное копирование — использовать инструмент qhb_basebackup. Если при вызове задать параметр -X, весь журнал предварительной записи, необходимый для использования резервной копии, будет автоматически включен в резервную копию, и никаких специальных действий для её восстановления не требуется.

Если требуется больше гибкости при копировании файлов резервных копий, процесс более низкого уровня можно использовать и для автономных оперативных резервных копий. Чтобы подготовиться к низкоуровневым автономным горячим резервным копиям, убедитесь, что для wal_level задано значение replica или выше, для параметра archive_mode значение on, и настройте команду archive_command, которая выполняет архивирование только при наличии файла переключателя. Например:

```
archive_command = 'test ! -f /var/lib/qhb/backup_in_progress ||
(test ! -f /var/lib/qhb/archive/%f && cp %p
/var/lib/qhb/archive/%f)'
```

В qbackup есть подкоманда backup-wal, которую можно использовать для этих целей. Эта команда выполняет архивирование, когда существует /var/lib/qhb/backup_in_progress, и в противном случае без вывода сообщений возвращает нулевое состояние выхода, что позволяет СУБД «Квант-Гибрид 1.5» переиспользовать ненужный файл WAL.

Создание резервной копии можно выполнить с помощью сценария:

```
touch /var/lib/qhb/backup_in_progress
psql -c "select pg_start_backup('hot_backup');"
```

```
tar -cf /var/lib/qhb/backup.tar /var/lib/qhb/data/
psql -c "select pg_stop_backup();"
rm /var/lib/qhb/backup_in_progress
tar -rf /var/lib/qhb/backup.tar /var/lib/qhb/archive/
```

Файл-переключатель /var/lib/qhb/backup_in_progress создаётся первым, что позволяет архивировать заполненные файлы WAL. После резервного копирования файл переключателя удаляется. Архивные файлы WAL затем добавляются в резервную копию, чтобы как базовая резервная копия, так и все необходимые файлы WAL были частью одного и того же tar-файла. Необходимо добавлять обработку ошибок в скрипты резервного копирования.

Существует альтернативный подход к горячему резервному копированию в виде утилиты qbackup. Эта утилита комбинирует архивирование WAL-записей и постраничное инкрементальное резервное копирование, что позволяет добиться более компактных резервных копий за счёт сохранения только тех данных, которые изменились с момента предыдущей автономной резервной копии.

8.4.2 Сжатие архивных журналов

Если размер архивного хранилища имеет значение, вы можете использовать gzip для сжатия архивных файлов:

```
archive_command = 'gzip < %p > /var/lib/qhb/archive/%f'
```

Затем нужно использовать gunzip во время восстановления:

```
restore_command = 'gunzip < /mnt/server/archivedir/%f > %p'
```

8.4.3 Скрипты archive_command

Существует возможность использовать сценарии для определения своей команды archive_command, их запись в qhb.conf выглядит так:

```
archive_command = 'local_backup_script.sh "%p" "%f"'
```

Использование отдельного файла сценария целесообразно тогда, когда вы хотите использовать более одной команды в процессе архивирования. Это позволяет управлять всей сложностью скрипта, который может быть написан на популярном языке сценариев, таком как bash или perl.

Примеры задач, которые могут быть решены в сценарии:

– Копирование данных для безопасного хранения данных за пределами площадки.
– Пакетное копирование WAL, так что они передаются каждые три часа, а не по одному.
– Взаимодействие с другим программным обеспечением для резервного копирования и восстановления.
– Взаимодействие с программным обеспечением для мониторинга ошибок.

При использовании сценария archive_command рекомендуется включить logging_collector. Любые сообщения, записанные в сценарий stderr, будут затем отображаться в журнале сервера базы данных, что позволяет легко диагностировать сложные конфигурации в случае сбоев.

Возможно использовать qbackup backup-wal для этих целей.

8.4.4 Пример настройки синхронной реплики

Внимание! В данном разделе приводится конечный пример настройки кластера баз данных с синхронной репликацией. Обратите внимание, что здесь приводятся некоторые параметры (например, ip адреса хостов или каталоги баз данных), которые могут не соответствовать вашим.

8.4.4.1 Начальные условия

172.31.101.70 — IP сервера, назначенного основным (мастер).

172.31.101.71 — IP резервного сервера (реплика).

Каталог размещения БД — **PG_DATA = /u01/qhb/db**.

/usr/local/qhb/ — путь установки QHB по умолчанию.

Сервисы ***systemd*** сконфигурированы с учетом вышеперечисленного.

8.4.4.2 Настройка основного сервера

1) Создаем пользователя QHB для репликации:

```
create user repluser replication password 'repluser';
```

2) Устанавливаем необходимые параметры в файле конфигурации **vi /u01/qhb/db/qhb.conf**:

```
listen_addresses = '*'
wal_level = hot_standby
```

```
max_wal_senders = 2
max_replication_slots = 2
hot_standby = on
hot_standby_feedback = on
logging_collector = on
log_directory = './log'
synchronous_commit = on
synchronous_standby_names = '*'
```

3) Устанавливаем параметры доступа к базе данных с машины резервного сервера в файле конфигурации

```
vi /u01/qhb/db/qhb_hba.conf

host replication repluser 172.31.101.70/32 md5
host replication repluser 172.31.101.71/32 md5
```

Внимание! Существующую запись ***host replication*** не изменяем.

4) Производим рестарт сервиса СУБД

```
systemctl restart qhb && systemctl status qhb
```

8.4.4.3 Настройка резервного сервера

1) Останавливаем сервис СУБД на сервере реплики

```
systemctl stop qhb && systemctl status qhb
```

2) Удаляем директорию с данными базы данных

```
rm -rf /u01/qhb/db/* # Полное удаление данных
```

Внимание! Удаляя директорию, вы должны полностью осознавать необратимость данного действия и тот факт, что все данные, файлы конфигурации и логи будут утеряны без возможности восстановления.

3) В случае если ситуация позволяет, выполняем восстановление через бэкап.

Если объем данных в базе данных большой, то рекомендуется воспользоваться копией данных, при ее наличии.

Восстановление через утилиту бэкапирования запускается от системного пользователя QHB:

```
sudo -u qhb \#
```

```
/usr/local/qhb/bin/qhb_basebackup \#Утилита бэкапа
-h 172.31.101.70 \ # IP адрес мастер сервера
-p 5432 \ # Порт по умолчанию; рекомендуется сменить, если конфигурация была изменена
-U repluser \ # Пользователь, созданный ранее на основном сервере, с правами репликации
-D /u01/qhb/db/ \ # Директория размещения СУБД
-Fp -Xs -P -R # Набор ключей, необходимых для репликации
```

Если копия делается при помощи qhb_basebackup, параметры подключения для реплики будут указаны в файле **qhb.auto.conf**. Следует удостовериться в корректности заданной строки подключения.

Если копия базы данных была развернута через файл (или иной корректный способ, не указанный в данной инструкции), то **до старта** необходимо скорректировать несколько параметров:

– запуск сервера СУБД в резервном режиме. Для этого требуется создать файл **standby.signal** в директории кластера базы данных (рядом с **qhb.conf**).

```
sudo -u qhb touch /u01/qhb/db/standby.signal
ls -la standby.signal
#-rw------- 1 qhb qhb     0 Nov 25 15:34 standby.signal
```

– в файле конфигурации **qhb.conf** необходимо указать строку подключения к основному серверу:

```
primary_conninfo    =    'user=repluser    password=repluser host=172.31.101.70 port=5432 sslmode=disable sslcompression=0 gssencmode=disable                          krbsrvname=postgres target_session_attrs=any'
```

Примечание: строка подключения длинная, и некоторые консольные редакторы могут применить опцию **word-wrap**, нарушая визуальную целостность строки.

4) В обоих случаях параметры СУБД в файле конфигурации будут скопированы с мастера, и следует изменить некоторые значения на соответствующие реплике.

Параметры на реплике указываются следующие:

```
vi /u01/qhb/db/qhb.conf

listen_addresses = '*'
logging_collector = on
log_directory = 'log' # Удостоверьтесь в существовании директории и наличии необходимых прав.
```

5) Теперь и только теперь можно запустить СУБД на втором сервере, реплике.

```
systemctl start qhb && systemctl status qhb
```

8.4.4.4 Проверка

1) Проверяем на мастере:

```
/usr/local/qhb/bin/psql -x -c "select * from pg_stat_replication"
```

Должен высветится один слот:

```
pid              | 15566
usesysid         | 16385
usename          | repluser
application_name | walreceiver
client_addr      | 172.31.101.71
client_hostname  |
client_port      | 40516
backend_start    | 2020-11-25 09:45:22.249802-03
backend_xmin     |
state            | streaming
sent_lsn         | 0/30001F8
write_lsn        | 0/30001F8
flush_lsn        | 0/30001F8
replay_lsn       | 0/30001F8
write_lag        |
flush_lag        |
replay_lag       |
sync_priority    | 1
sync_state       | sync
                   ^^^^^^^
reply_time       | 2020-11-25 09:47:22.710523-03
```

Причем **sync_state** должен быть ***sync*** (репликация синхронная).

2) Создадим таблицу на мастере:

```
CREATE TABLE rtest (vname VARCHAR(40));
```

3) Заполним случайными значениями на миллион строк:

```
insert into rtest select substr(md5(random()::text), 0, 40)
from generate_series(1,1000000);
```

4) На ведомом сервере проверяем чтение из таблицы:

```
qhb=# select count(*) from rtest;
  count
---------
 1000000
(1 row)
```

8.4.4.5 Дополнительные настройки

В некоторых вариантах рекомендовано прописать явное указание слота репликации.

На реплике в файл **qhb.auto.conf** добавить параметр primary_slot_name = 'standby_slot2_qhb'.

На мастере выполнить команду SQL:

```
SELECT
pg_create_physical_replication_slot('standby_slot2_qhb');
```

Запрос по активным слотам репликации должен показывать на мастере нашу реплику

```
SELECT * FROM pg_replication_slots WHERE active=true;
```

8.5 Предостережения

Существует несколько ограничений техники непрерывного архивирования.

– Если команда CREATE DATABASE выполняется во время создания базовой резервной копии, а затем шаблонная база данных, которую скопировала CREATE DATABASE изменяется, пока базовая резервная копия все ещё выполняется, возможно, что восстановление приведёт к распространению этих изменений на созданную базу данных. Чтобы

избежать этого риска, лучше не изменять шаблонные базы данных при создании базовой резервной копии.

– Команды CREATE TABLESPACE регистрируются в WAL с буквальным абсолютным путём и поэтому будут воспроизведены как создания табличного пространства с тем же абсолютным путём. Это может быть нежелательно, если журнал воспроизводится на другом компьютере. Это может быть опасно, даже если журнал воспроизводится на том же компьютере, но в новом каталоге данных: воспроизведение все равно перезапишет содержимое исходного табличного пространства. Чтобы избежать потенциальных ошибок такого рода, рекомендуется создавать новую базовую резервную копию после создания или удаления табличных пространств.

Формат WAL по умолчанию довольно громоздкий, поскольку включает в себя множество снимков страницы диска. Эти снимки страниц предназначены для поддержки восстановления после сбоя, поскольку может потребоваться исправить частично записанные страницы диска. В зависимости от аппаратного и программного обеспечения системы риск частичной записи может быть достаточно мал, чтобы его можно было игнорировать, и в этом случае вы можете значительно уменьшить общий объём архивированных журналов, выключив снимки страниц с помощью параметра full_page_writes. Прочтите примечания и предупреждения к WAL перед тем, как сделать это. Выключение настройки снимков страницы не мешает использованию журналов для операций PITR. Возможность для последующего улучшения — сжатие архивных данных WAL путём удаления ненужных копий страниц, даже если установлен параметр full_page_writes. Может быть необходимость уменьшить количество снимков страниц, включенных в WAL, максимально увеличив параметры интервала контрольных точек.

9 РОЛИ В БАЗЕ ДАННЫХ

СУБД «Квант-Гибрид 1.5» управляет правами доступа к базе данных, используя концепцию ролей. Роль может рассматриваться как пользователь базы данных или группа пользователей базы данных, в зависимости от того, как установлена роль. Роли могут владеть объектами базы данных (например, таблицами и функциями) и могут назначать права для этих объектов другим ролям, чтобы контролировать, кто имеет доступ к каким объектам. Кроме того, можно предоставить членство в роли другой роли, что позволяет роли участника использовать права, назначенные другой роли.

Понятие ролей объединяет понятия «пользователи» и «группы». Любая роль может выступать в роли пользователя, группы или обоих.

9.1 Роли базы данных

Роли базы данных концептуально полностью отделены от пользователей операционной системы. На практике может быть удобно ввести соответствие, но это не обязательно. Роли базы данных являются глобальными для установленного экземпляра базы данных (а не для отдельной базы данных). Чтобы создать роль, используйте команду SQL CREATE ROLE:

```
CREATE ROLE name;
```

name соответствует правилам для идентификаторов SQL: без украшений без специальных символов или в двойных кавычках (на практике обычно требуется добавить в команду дополнительные параметры, такие как LOGIN). Чтобы удалить существующую роль, используйте аналогичную команду DROP ROLE:

```
DROP ROLE name;
```

Для удобства программы createuser и dropuser предоставляются в качестве оберток вокруг этих команд SQL, которые можно вызывать из командной строки оболочки:

```
createuser name
dropuser name
```

Чтобы определить набор существующих ролей, изучите системный каталог ***pg_roles***, например

```
SELECT rolname FROM pg_roles;
```

Метакоманда \du программы psql полезна для просмотра существующих ролей.

Для начальной загрузки системы базы данных, недавно инициализированная система всегда содержит одну предопределенную роль. Эта роль всегда является «суперпользователем», и по умолчанию (если она не изменена при запуске ***initdb***) она будет иметь то же имя, что и пользователь операционной системы, который инициализировал экземпляр базы данных. Обычно эта роль будет называться ***qhb***. Чтобы создать больше ролей, сначала необходимо подключиться с этой начальной ролью к БД.

Каждое соединение с сервером базы данных выполняется с использованием имени определенной роли, и эта роль определяет начальные права доступа для команд, выполненных в этом соединении. Имя роли, которое будет использоваться для конкретного подключения к базе данных, указывается клиентом, который инициирует запрос на подключение в зависимости от приложения. Например, программа ***psql*** использует параметр командной строки ***-U*** чтобы указать роль для подключения. Многие приложения по умолчанию принимают имя текущего пользователя операционной системы (включая ***createuser*** и ***psql***). Поэтому часто удобно поддерживать соответствие имен между ролями и пользователями операционной системы.

Набор ролей базы данных, к которым может подключаться данное клиентское соединение, определяется настройкой аутентификации клиента (таким образом, клиент не ограничен подключением в качестве роли, соответствующей пользователю операционной системы так же, как имя пользователя не обязательно должно совпадать с его или ее настоящим именем). Поскольку идентификация роли определяет набор привилегий, доступных подключенному клиенту, важно тщательно настроить права при настройке многопользовательской среды.

9.2 Атрибуты ролей

Роль в базе данных может иметь ряд атрибутов, которые определяют ее права и взаимодействуют с системой аутентификации клиента.

Таблица 4. Атрибуты ролей

Атрибут	**Описание**
LOGIN (вход в систему)	Только роли с атрибутом LOGIN могут использоваться в качестве начального имени роли для подключения к базе данных. Роль с атрибутом LOGIN может рассматриваться как «пользователь базы данных». Чтобы создать роль с правами входа в систему, используйте: CREATE ROLE name LOGIN; CREATE USER name; CREATE USER эквивалентен CREATE ROLE за исключением того, что CREATE USER по умолчанию включает LOGIN, а CREATE ROLE - нет)
SUPERUSER (статус суперпользователя)	Суперпользователь базы данных обходит все проверки разрешений, кроме права на вход. Это опасная привилегия, и ее не следует использовать небрежно. Лучше всего выполнять большую часть своей работы в роли, которая не является суперпользователем. Чтобы создать нового суперпользователя базы данных, используйте CREATE ROLE name SUPERUSER. Необходимо делать это под ролью, которая уже является суперпользователем.
CREATEDB (создание базы данных)	Роль должна быть явно предоставлена разрешение на создание баз данных (за исключением суперпользователей, поскольку они обходят все проверки разрешений). Чтобы создать такую роль, используйте CREATE ROLE name CREATEDB.

Атрибут	Описание
CREATEROLE (создание ролей)	Роли должно быть явно предоставлено разрешение на создание большего количества ролей (за исключением суперпользователей, так как они обходят все проверки разрешений). Чтобы создать такую роль, используйте CREATE ROLE name CREATEROLE. Роль с привилегией CREATEROLE может изменять и CREATEROLE другие роли, а также предоставлять или отзывать членство в них. Однако для создания, изменения, удаления или изменения принадлежности к роли суперпользователя требуется статус суперпользователя; CREATEROLE недостаточно для этого.
REPLICATION (инициирование репликации)	Роль должна быть явно предоставлена разрешение на инициирование потоковой репликации (за исключением суперпользователей, так как они обходят все проверки разрешений). Роль, используемая для потоковой репликации, также должна иметь разрешение LOGIN. Чтобы создать такую роль, используйте CREATE ROLE name REPLICATION LOGIN.
PASSWORD (пароль)	Пароль имеет значение только в том случае, если метод аутентификации клиента требует от пользователя ввода пароля при подключении к базе данных. password и методы аутентификации md5 используют пароли. Пароли базы данных отделены от паролей операционной системы. Укажите пароль при создании роли с помощью CREATE ROLE name PASSWORD ’ string ’.

Рекомендуется создать роль с привилегиями CREATEDB и CREATEROLE, которая не является суперпользователем, а затем использовать эту роль для всего

рутинного управления базами данных и ролями. Такой подход позволяет избежать опасностей работы в качестве суперпользователя для задач, которые на самом деле этого не требуют.

Атрибуты роли могут быть изменены после создания с помощью ***ALTER ROLE***.

Роль также может иметь специфичные для роли значения «по умолчанию» для многих параметров конфигурации настроек сервера. Например, если по какой-то причине необходимо выключить сканирование индекса при каждом подключении, можно использовать:

```
ALTER ROLE myname SET enable_indexscan TO off;
```

Это сохранит настройку (но не установит ее сразу). В последующих соединениях с этой ролью будет выглядеть, как будто SET enable_indexscan TO off был выполнен непосредственно перед началом сеанса. Вы все еще можете изменить этот параметр во время сеанса; это будет только по умолчанию. Чтобы удалить настройку по умолчанию для конкретной роли, используйте ALTER ROLE rolename RESET varname. Обратите внимание, что специфичные для роли значения по умолчанию, прикрепленные к ролям без права ***LOGIN***, довольно бесполезны, поскольку они никогда не будут вызваны.

9.3 Ролевая модель управления доступом

Часто удобно группировать пользователей, чтобы упростить управление привилегиями: таким образом, права могут быть предоставлены или отменены для группы в целом. В СУБД «Квант-Гибрид 1.5» это достигается созданием роли, представляющей группу, а затем предоставлением членства в роли группы отдельным ролям пользователей.

Чтобы настроить групповую роль, сначала создайте роль:

```
CREATE ROLE name;
```

Обычно роль, используемая в качестве группы, не имеет атрибута LOGIN, хотя вы можете установить ее, если хотите.

Когда роль группы существует, вы можете добавлять и удалять участников, используя команды GRANT и REVOKE:

```
GRANT group_role TO role1, ... ;
REVOKE group_role FROM role1, ... ;
```

Возможно предоставить членство и другим групповым ролям (нет различий между групповыми ролями и не групповыми ролями). База данных не позволит вам настроить циклические циклы членства. Кроме того, не разрешено предоставлять членство в роли для ***PUBLIC***.

Члены групповой роли могут использовать права роли двумя способами. Во-первых, каждый член группы может явно выполнить SET ROLE, чтобы временно «стать» групповой ролью. В этом состоянии сеанс базы данных имеет доступ к привилегиям роли группы, а не к исходной роли входа, и любые созданные объекты базы данных считаются принадлежащими роли группы, а не роли входа. Во-вторых, роли участников, имеющие атрибут ***INHERIT*** автоматически используют права ролей, членами которых они являются, включая любые права, унаследованные этими ролями.

Пример:

```
CREATE ROLE joe LOGIN INHERIT;
CREATE ROLE admin NOINHERIT;
CREATE ROLE wheel NOINHERIT;
GRANT admin TO joe;
GRANT wheel TO admin;
```

Сразу после подключения в качестве роли *joe* сеанс базы данных будет использовать права, предоставленные непосредственно *joe*, а также любые права, предоставленные *admin*, поскольку *joe* «наследует» права *admin*. Однако права, предоставленные *wheel*, ему недоступны, потому что, хотя *joe* косвенно является членом *wheel*, членство осуществляется через роль *admin*, которая имеет атрибут *NOINHERIT*. После:

```
SET ROLE admin;
```

сеанс будет использовать только те права, которые предоставлены *admin*, а не те, которые предоставлены *joe*. После:

```
SET ROLE wheel;
```

сеанс будет использовать только те права, которые предоставлены *wheel*, а не те, которые предоставлены либо *joe* либо *admin*. Исходное состояние привилегий может быть восстановлено любым способом из:

```
SET ROLE joe;
SET ROLE NONE;
RESET ROLE;
```

Команда *SET ROLE* всегда позволяет выбрать любую роль, в которую прямо или косвенно входит исходная роль входа. Таким образом, в приведенном выше примере нет необходимости становиться *admin*, прежде чем стать *wheel*.

В стандарте SQL существует четкое различие между пользователями и ролями, и пользователи не наследуют права автоматически, в то время как роли делают. Такое поведение может быть получено в СУБД «Квант-Гибрид 1.5», если для ролей, используемых в качестве ролей SQL, используется атрибут *INHERIT*, а для ролей, используемых в качестве пользователей SQL, - атрибут *NOINHERIT*. В СУБД «Квант-Гибрид 1.5» по умолчанию всем ролям предоставляет атрибут *INHERIT*.

Атрибуты роли *LOGIN*, *SUPERUSER*, *CREATEDB* и *CREATEROLE* могут рассматриваться как специальные права, но они никогда не наследуются, как обычные права для объектов базы данных. Вы должны на самом деле установить роль для конкретной роли, имеющей один из этих атрибутов, чтобы использовать этот атрибут. Продолжая приведенный выше пример, мы можем предоставить *CREATEDB* и *CREATEROLE* роль *admin*. Тогда сеанс, соединяющийся как роль *joe*, не будет иметь этих привилегий сразу, а только после выполнения SET ROLE admin.

Чтобы уничтожить групповую роль, используйте *DROP ROLE*:

```
DROP ROLE name;
```

Любое членство в групповой роли автоматически отменяется (но роли участников не затрагиваются иным образом).

9.4 Удаление ролей

Поскольку роли могут владеть объектами базы данных и могут иметь права для доступа к другим объектам, удаление роли часто является не просто вопросом

быстрого *DROP ROLE*. Любые объекты, принадлежащие этой роли, должны быть сначала сброшены или переназначены другим владельцам; и любые разрешения, предоставленные этой роли, должны быть аннулированы.

Право собственности на объекты может передаваться по одному с помощью команд *ALTER*, например:

```
ALTER TABLE user1_table OWNER TO user2;
```

В качестве альтернативы, можно использовать команду REASSIGN OWNED для переназначения владения всеми объектами, принадлежащими роли, которая должна быть удалена, другой роли. Поскольку *REASSIGN OWNED* не может получить доступ к объектам в других базах данных, необходимо запускать его в каждой базе данных, содержащей объекты, принадлежащие этой роли (обратите внимание, что первый такой *REASSIGN OWNED* изменит владельца любых совместно используемых между базами данных объектов, то есть баз данных или табличных пространств, которые принадлежат роли, подлежащей удалению).

Как только любые ценные объекты были переданы новым владельцам, любые оставшиеся объекты, принадлежащие подлежащей удалению роли, могут быть удалены с помощью команды DROP OWNED. Эта команда не может получить доступ к объектам в других базах данных, поэтому необходимо запускать ее в каждой базе данных, содержащей объекты, принадлежащие этой роли. Кроме того, *DROP OWNED* не удаляет целые базы данных или табличные пространства, поэтому это необходимо делать вручную, если роли принадлежат какие-либо базы данных или табличные пространства, которые не были переданы новым владельцам.

DROP OWNED также занимается удалением любых привилегий, предоставленных целевой роли для объектов, которые ей не принадлежат. Поскольку REASSIGN OWNED не касается таких объектов, как правило, необходимо запустить как *REASSIGN OWNED* и *DROP OWNED* (в таком порядке!), чтобы полностью удалить зависимости роли, которую необходимо удалить.

Итого, самый общий рецепт удаления роли, которая использовалась для владения объектами:

```
REASSIGN OWNED BY doomed_role TO successor_role;
```

```
DROP OWNED BY doomed_role;
-- repeat the above commands in each database of the cluster
DROP ROLE doomed_role;
```

Когда не все принадлежащие объекты должны быть переданы одному и тому же владельцу-преемнику, лучше всего обработать исключения вручную, а затем выполнить описанные выше шаги для полной очистки.

Если попытка *DROP ROLE* выполняется, пока зависимые объекты все еще остаются, она выдаст сообщения, определяющие, какие объекты необходимо переназначить или отбросить.

9.5 Роли по умолчанию

СУБД «Квант-Гибрид 1.5» предоставляет набор ролей по умолчанию, которые обеспечивают доступ к определенным, часто необходимым, привилегированным возможностям и информации. Администраторы могут предоставлять эти роли пользователям и/или другим ролям в их среде, предоставляя этим пользователям доступ к указанным возможностям и информации.

Роли по умолчанию описаны в таблице ниже (Таблица 5. Роли по умолчанию). Обратите внимание, что конкретные разрешения для каждой роли по умолчанию могут измениться в будущем при добавлении дополнительных возможностей. Администраторы должны отслеживать заметки о выпуске на предмет изменений.

Таблица 5. Роли по умолчанию

Роль	**Разрешенный доступ**
pg_read_all_settings	Прочитайте все переменные конфигурации, даже те, которые обычно видны только суперпользователям.
pg_read_all_stats	Прочитайте все представления pg_stat_ * и используйте различные статистические расширения, даже те, которые обычно видны только суперпользователям.

Роль	Разрешенный доступ
pg_stat_scan_tables	Выполните функции мониторинга, которые могут блокировать ACCESS SHARE для таблиц, возможно, в течение длительного времени.
pg_monitor	Читать / выполнять различные виды мониторинга и функции. Эта роль является членом групп pg_read_all_settings, pg_read_all_stats и pg_stat_scan_tables.
pg_database_owner	Никакого. Членство неявно включает в себя текущего владельца базы данных.
pg_signal_backend	Подать сигнал другому бэкенду, чтобы отменить запрос или завершить сеанс.
pg_read_server_files	Разрешить чтение файлов из любого места, к которому база данных может получить доступ на сервере с помощью COPY и других функций доступа к файлам.
pg_write_server_files	Разрешить запись в файлы в любом месте, к которому база данных может получить доступ на сервере с помощью COPY и других функций доступа к файлам.
pg_execute_server_program	Разрешить выполнение программ на сервере базы данных как пользователь, база данных запускается так же, как с COPY и другими функциями, которые позволяют выполнять программу на стороне сервера.
pg_checkpoint	Разрешить выполнение команды CHECKPOINT.

У роли pg_database_owner имеется только один неявный, определяемый сложившейся ситуацией член, а именно — владелец текущей базы данных.

Изначально эта роль не предоставляет никаких прав. Как и любая роль, она может владеть объектами или получать права доступа. Следовательно, как только pg_database_owner приобретет права в базе-шаблоне, каждый владелец базы данных, созданной из этого шаблона, получит эти права. Роль pg_database_owner не может быть членом какой-либо роли и не может явно включать членов. Эта роль владеет схемой public, так что каждый владелец базы данных управляет локальным использованием этой схемы.

pg_monitor, ***pg_read_all_settings***, ***pg_read_all_stats*** и ***pg_stat_scan_tables*** предназначены для того, чтобы администраторы могли легко настроить роль для мониторинга сервера базы данных. Они предоставляют набор общих привилегий, позволяющих роли читать различные полезные параметры конфигурации, статистику и другую системную информацию, обычно доступную только суперпользователям.

Роль ***pg_signal_backend*** предназначена для того, чтобы администраторы могли включать доверенные, но не суперпользовательские роли, для отправки сигналов другим бэкендам. В настоящее время эта роль позволяет отправлять сигналы для отмены запроса на другом сервере или завершения его сеанса. Однако пользователь, которому предоставлена эта роль, не может отправлять сигналы бэкенду, принадлежащему суперпользователю.

pg_read_server_files, ***pg_write_server_files*** и ***pg_execute_server_program*** предназначены для того, чтобы администраторы могли иметь доверенные, но не суперпользовательские роли, которые могут получать доступ к файлам и запускать программы на сервере базы данных в качестве пользователя базы данных. Поскольку эти роли могут получить доступ к любому файлу в файловой системе сервера, они пропускают все проверки разрешений на уровне базы данных при непосредственном доступе к файлам и могут использоваться для получения доступа на уровне суперпользователя, поэтому при предоставлении этих ролей следует соблюдать особую осторожность.

При предоставлении этих ролей следует соблюдать осторожность, чтобы гарантировать, что они используются только там, где это необходимо, и при том понимании, что эти роли предоставляют доступ к конфиденциальной информации.

Администраторы могут предоставить доступ к этим ролям пользователям с помощью команды GRANT, например:

```
GRANT pg_signal_backend TO admin_user;
```

9.6 Роли ИБ

В Системе реализованы две роли информационной безопасности: ***Роль Администратора СУБД*** и ***Роль Администратора БД***, которые реализованы в виде разделяемых каталогов (таблиц схемы **pg_catalog**):

- ***qhb_dbms_admin*** для администраторов СУБД;
- ***qhb_db_admin*** для администраторов конкретных Баз Данных.

9.6.1 Информация о ролях Администраторов ИБ

Наличие в системе ролей Администраторов СУБД и Администраторов конкретных БД позволяет назначать Администраторов Информационной Безопасности не включая атрибут Суперпользователя у соответствующих записей в таблице каталога пользователей и ролей ***pg_authid***.

9.6.1.1 Привилегии Администраторов ИБ

Таблица 6. Привилегии Администраторов ИБ

Объект в СУБД	**Администратор СУБД**	**Администратор БД**
Таблица в БД	Права по умолчанию	Полные права в управляемой БД: Вставка, Чтение, Редактирование, Удаление
Последовательность	Права по умолчанию	Полные права в управляемой БД: Использование, Чтение, Редактирование
База Данных	Имеет право создавать БД	Полные права в управляемой БД: Создание объектов, Логин, Работа с планировщиком

Объект в СУБД	**Администратор СУБД**	**Администратор БД**
Foreign Data Wrapper	Права по умолчанию	Право использования в управляемой БД
Функции/процедуры	Права по умолчанию	Право использования в управляемой БД
Язык процедур	Права по умолчанию	Право использования в управляемой БД
pg_largeobject	Права по умолчанию	Полные права в управляемой БД: Чтение и изменение
ALTER SYSTEM	Полные права в СУБД	Права по умолчанию
Схема	Права по умолчанию	Полные права в управляемой БД: Использование и создание
Табличное пространство	Полные права на создание	Права по умолчанию
Тип данных	Права по умолчанию	Полные права в управляемой БД, включая создание

9.6.1.2 Особенности реализации

В данной реализации назначение ролей Администратора СУБД и Администратора конкретных БД осуществляется индивидуально каждому пользователю, минуя иерархию ролей в СУБД. Назначить роли Администраторов ИБ групповым ролям СУБД нельзя.

9.6.1.3 Назначение ролей ИБ

Суперпользователь имеет полномочия назначать Администраторов СУБД и Администраторов конкретных БД с помощью встроенных процедур.

9.6.1.3.1 Назначение Администратора СУБД

Встроенная процедура ***qhb_make_dbms_admin*** позволяет назначить пользователя одним из Администраторов СУБД, что наделит его дополнительными полномочиями.

```
SELECT qhb_make_dbms_admin(<user_oid>);
```

Для передачи идентификатора пользователя в виде параметра в функцию ***qhb_make_dbms_admin*** следует найти значение ***oid*** в каталоге **pg_authid**, соответствующее имени требуемого пользователя. Например, с помощью такого запроса SQL:

```
SELECT oid FROM pg_authid WHERE rolname LIKE '<user_name>';
```

Права для назначения пользователя Администратором СУБД есть у суперпользователей.

Права для назначения пользователя Администратором СУБД отсутствуют:

- у обычных пользователей;
- у Администраторов СУБД и БД, не являющихся суперпользователями.

9.6.1.3.2 Назначение Администратора БД

Встроенная процедура ***qhb_make_db_admin*** позволяет назначить пользователя одним из Администраторов заданной БД, что наделит его дополнительными полномочиями в отношении выбранной БД и ее объектов.

```
SELECT qhb_make_db_admin(<user_oid>, <database_oid>);
```

Для передачи идентификатора пользователя в виде параметра в функцию ***qhb_make_db_admin*** следует найти значение ***oid*** в каталоге **pg_authid**, соответствующее имени требуемого пользователя. Например, с помощью такого запроса SQL:

```
SELECT oid FROM pg_authid WHERE rolname LIKE '<user_name>';
```

Аналогично, идентификатор БД для передачи в виде параметра можно найти с помощью следующего запроса SQL:

```
SELECT oid FROM pg_database WHERE datname LIKE
'<database_name>';
```

Права для назначения пользователя Администратором БД есть:

- у Суперпользователей;

– у Администраторов СУБД.

Отсутствуют права для назначения пользователя Администратором БД:

– у обычных пользователей;

– у Администраторов БД, не являющихся суперпользователями.

9.6.2 Просмотр текущих Администраторов СУБД и БД

Текущие Администраторы ИБ отображены в таблицах разделяемого каталога qhb_dbms_admin и qhb_db_admin.

С помощью следующих запросов, использующих встроенные процедуры, можно определить текущий статус пользователя:

```
SELECT qhb_is_dbms_admin(<user_oid>);
```

и

```
SELECT qhb_is_db_admin(<user_oid>, <database_oid>);
```

9.6.3 Удаление ролей Администраторов ИБ

Для отмены назначения ролей Администратора СУБД и Администратора БД следует воспользоваться процедурами ***qhb_drop_dbms_admin*** и ***qhb_drop_db_admin***.

Для удаления назначения пользователя Администратором СУБД можно использовать запрос:

```
SELECT qhb_drop_dbms_admin(<user_oid>);
```

Для удаления назначения пользователя Администратором конкретной БД можно использовать запрос:

```
SELECT qhb_drop_db_admin(<user_oid>, <database_oid>);
```

Для передачи идентификатора пользователя в виде параметра в функции ***qhb_drop_dbms_admin*** и ***qhb_drop_db_admin*** следует найти значение ***oid*** в каталоге **pg_authid**, соответствующее имени требуемого пользователя. Например, с помощью такого запроса SQL:

```
SELECT oid FROM pg_authid WHERE rolname LIKE '<user_name>';
```

Аналогично, идентификатор БД для передачи в виде параметра можно найти с помощью следующего запроса SQL:

```
SELECT oid FROM pg_database WHERE datname LIKE
'<database_name>';
```

9.7 Профили безопасности

Профили безопасности позволяют производить групповую настройку политик безопасности СУБД в отношении пользователей.

9.7.1 Описание таблиц

Связь пользователя с определенным профилем безопасности хранится в таблице ***pg_catalog.qhb_auth_profile_authid***, где столбец **role_id** указывает на **Oid** пользователя СУБД, расположенного в каталоге ***pg_catalog.pg_authid***, а столбец **profile_id** указывает на сам профиль, расположенный в таблице ***pg_catalog.qhb_auth_profile***.

По умолчанию, доступен один начальный профиль безопасности (единственная запись в **qhb_auth_profile**) с идентификатором **Oid** равным ***9352***. Удаление этого профиля может повредить стабильности работы системы, так как он используется для всех пользователей, кому еще не был назначен администратором СУБД выделенный профиль.

Информация о заблокированных пользователях хранится в таблице ***pg_catalog.qhb_user_lockout***, причем блокировки применяются только при включенном параметре **logon_jobs = on** конфигурации СУБД в **qhb.conf**. По умолчанию этот параметр выключен, то есть **logon_jobs = off**.

Перед обновлением кластера утилитами **qhb_upgrade** следует выключить параметр **logon_jobs** выставив его в ***off*** и перезапустить СУБД. Иначе стабильность процесса обновления кластера не гарантируется.

9.7.2 Атрибутивный состав таблиц

Для таблицы ***qhb_user_lockout*** доступны следующие столбцы:

- Oid oid - идентификатор записи, первичный ключ;
- Oid role_id - идентификатор пользователя из таблицы pg_catalog.pg_authid;
- int4 invalid_attempts - счетчик ошибочных попыток ввода пароля, необходим для своевременной блокировки;
- bool locked - признак блокировки пользователя role_id;

– timestamp locked_since - дата и время начала блокировки;
– timestamp last_activity - дата и время последнего захода в СУБД.

Для таблицы ***qhb_auth_profile*** доступны следующие столбцы:

– Oid oid - идентификатор записи, первичный ключ;
– int4 max_failed_attempts - разрешенное данным профилем число ошибочных попыток ввода пароля;
– bool auto_unlock - признак автоматической разблокировки пользователей данного профиля безопасности;
– int4 lockout_duration_sec - длительность блокировки в секундах, при включенной автоматической разблокировке;
– int4 inactivity_lockout_sec - опция блокировки по таймауту неактивности пользователей данного профиля, значение в секундах;
– bool password_check - признак проверки задаваемых паролей на соответствие политикам безопасности профиля;
– int4 password_len - опция задания минимальной длины пароля, значение в количестве символов;
– int4 alphabet_size - опция задания минимальной мощности алфавита пароля.

Для таблицы ***qhb_auth_profile_authid*** доступны следующие столбцы:

– Oid role_id - идентификатор пользователя из таблицы pg_catalog.pg_authid, первичный ключ;
– Oid profile_id идентификатор профиля из таблицы pg_catalog.qhb_auth_profile.

9.7.3 Процедуры SQL для управления профилями

– Разблокировать всех пользователей для данного профиля

```
qhb_auth_profile_unlock_by_profile ( id_profile )
```

Где id_profile (тип - Oid) - идентификатор профиля.

Пример:

```
SELECT qhb_auth_profile_unlock_by_profile(9352);
```

–Заблокировать всех пользователей для данного профиля

```
qhb_auth_profile_lockout_by_profile ( id_profile )
```

Где id_profile (тип - Oid) - идентификатор профиля.

Пример:

```
SELECT qhb_auth_profile_lockout_by_profile(9352);
```

– Установить минимальную мощность алфавита пароля для данного профиля

```
qhb_auth_profile_set_password_alphabet_size ( id_profile,
total_char)
```

Где id_profile (тип - Oid) - идентификатор профиля. total_char (тип - INTEGER) - мощность алфавита (общее количество символов) или NULL для отключения проверки.

Пример:

```
SELECT qhb_auth_profile_set_password_alphabet_size(9352,
60);
```

– Установить минимальную длину пароля для данного профиля

```
qhb_auth_profile_set_password_len ( id_profile,
pass_length)
```

Где id_profile (тип - Oid) - идентификатор профиля. pass_length (тип - INTEGER) - минимальная длина пароля или NULL для отключения проверки.

Пример:

```
SELECT qhb_auth_profile_set_password_len(9352, 6);
```

– Отключить проверку сложности пароля в данном профиле

```
qhb_auth_profile_disable_password_check ( id_profile )
```

Где id_profile (тип - Oid) - идентификатор профиля.

Пример:

```
SELECT qhb_auth_profile_disable_password_check(9352);
```

– Включить проверку сложности паролей в данном профиле

```
qhb_auth_profile_enable_password_check ( id_profile )
```

Где id_profile (тип - Oid) - идентификатор профиля.

Пример:

```
SELECT qhb_auth_profile_enable_password_check(9352);
```

– Заблокировать данного пользователя

```
qhb_auth_profile_lockout_user ( id_user )
```

Где id_user (тип - Oid) - идентификатор пользователя.

Пример:

```
SELECT qhb_auth_profile_lockout_user(10);
```

– Разблокировать данного пользователя

```
qhb_auth_profile_unlock_user ( id_user )
```

Где id_user (тип - Oid) - идентификатор пользователя.

Пример:

```
SELECT qhb_auth_profile_unlock_user(10);
```

– Включить либо выключить автоматическую разблокировку пользователей данного профиля

```
qhb_auth_profile_set_auto_unlock ( id_profile,  status)
```

Где id_profile (тип - Oid) - идентификатор профиля. Status (тип - BOOLEAN) - значение параметра.

Пример:

```
SELECT qhb_auth_profile_set_auto_unlock(9352, true);
```

– Установить допустимое число ошибок при вводе пароля для пользователей данного профиля

```
qhb_auth_profile_set_max_failed_attempts ( id_profile,
num_errors)
```

Где id_profile (тип - Oid) - идентификатор профиля. num_errors (тип - INTEGER) - допустимое число ошибок при вводе пароля.

Пример:

```
SELECT qhb_auth_profile_set_max_failed_attempts(9352, 10);
```

– Убрать проверку на максимальное число ошибок при вводе пароля для пользователей данного профиля

```
qhb_auth_profile_set_unlimited_attempts ( id_profile )
```

Где id_profile (тип - Oid) - идентификатор профиля.

Пример:

```
SELECT qhb_auth_profile_set_unlimited_attempts(9352);
```

– Назначить данному пользователю профиль по умолчанию

```
qhb_auth_profile_unset ( id_user )
```

Где id_user (тип - Oid) - идентификатор пользователя.

Пример:

```
SELECT qhb_auth_profile_unset(10);
```

– Назначить данному пользователю данный профиль безопасности

```
qhb_auth_profile_set ( id_user, id_profile )
```

Где id_user (тип - Oid) - идентификатор пользователя. id_profile (тип - Oid) - идентификатор профиля.

Пример:

```
SELECT qhb_auth_profile_set(10, 9352);
```

– Удалить профиль безопасности

```
qhb_auth_profile_delete ( id_profile )
```

Где id_profile (тип - Oid) - идентификатор профиля.

Пример:

```
SELECT qhb_auth_profile_delete(1);
```

– Создать новый профиль безопасности по данному шаблону

```
qhb_auth_profile_create ( id_profile )
```

Где id_profile (тип - Oid) - идентификатор профиля-образца.

Пример:

```
SELECT qhb_auth_profile_create(9352);
```

ВНИМАНИЕ! В случае ручного управления таблицами каталога, при выполнении запросов SQL напрямую к таблицам ***pg_authid***, ***qhb_auth_profile***, ***qhb_user_lockout***, ***qhb_auth_profile_authid***, возможно нарушение стабильности работы подсистемы профилей безопасности и СУБД в целом. Эти таблицы являются таблицами разделяемого каталога, и поэтому не поддерживают триггеры для отслеживания изменений в них.

9.7.4 Проверка паролей на соответствие политике профиля

Проверка паролей на сложность, требуемую политикой профиля, происходит только при установке пароля при включенном значении **password_check** в соответствующем профиле. При использовании администратором СУБД процедур

для управления политиками сложности пароля, следует также задать новые пароли для всех пользователей, относящихся к профилю, и разблокировать их для успешного входа. При ослаблении параметра сложности пароля блокировки не происходит, и пользователь может зайти со своим прежним паролем.

9.7.5 Интеграция с ролями ИБ

Для пользователей, назначенных Администраторами СУБД (qhb_dbms_admin) доступно создание, изменение, удаление и назначение профилей безопасности. Также они имеют возможность подвергнуть других пользователей блокировке и разблокированию (за исключением других суперпользователей и Администраторов СУБД).

Для пользователей, назначенных Администраторами конкретных БД (qhb_db_admin) не предоставляется дополнительных полномочий, таких как создание, изменение, удаление и назначение профилей безопасности.

9.8 Функции безопасности

Функции, триггеры и политики безопасности на уровне строк позволяют пользователям вставлять код на внутренний сервер, который другие пользователи могут выполнять непреднамеренно. Следовательно, эти механизмы позволяют создавать «троянских коней» относительно легко. Самая сильная защита - жесткий контроль над тем, кто может определять объекты. Там, где это невозможно, пишите запросы, относящиеся только к объектам, имеющим доверенных владельцев. Удалите из ***search_path*** общедоступную схему и любые другие схемы, которые позволяют не доверенным пользователям создавать объекты.

Функции выполняются внутри процесса внутреннего сервера с разрешениями операционной системы демона сервера баз данных. Если язык программирования, используемый для функции, допускает неконтролируемый доступ к памяти, можно изменить внутренние структуры данных сервера. Следовательно, среди прочего, такие функции могут обойти любые системы контроля доступа. Языки функций, которые разрешают такой доступ, считаются «ненадежными», а СУБД «Квант-

Гибрид 1.5» позволяет только суперпользователям создавать функции, написанные на этих языках.

10 ПОЛНОМОЧИЯ

При создании объекта БД ему назначается владелец. Обычно владельцем является роль/пользователь, выполнившая команду создания объекта. Для большинства типов объектов начальное состояние таково, что только владелец (или суперпользователь) может что-либо делать с объектом. Чтобы другие роли могли его использовать, необходимо предоставить права.

Право изменять или уничтожать объект всегда является правом только владельца.

Объект может быть назначен новому владельцу с помощью команды ALTER соответствующего вида для объекта, например:

```
ALTER TABLE table_name OWNER TO new_owner;
```

Суперпользователи всегда могут это сделать, обычные же роли могут делать это только в том случае, если они являются одновременно текущим владельцем объекта (или членом роли-владельца) и членом новой роли-владельца.

Для назначения прав используется команда GRANT.

10.1 Предоставление права доступа к объектам базы данных

Предоставление права осуществляется выполнением команды, имеющей общий вид:

```
GRANT { { Название_права }[, ...] | ALL [ PRIVILEGES ] }
    ON { [ тип_объекта ] имя_объекта [, ...]  }
    TO указание_роли [, ...] [ WITH GRANT OPTION ]
```

Право доступа может быть предоставлено к таблице, столбцу, представлению, сторонней таблице, последовательности, базе данных, обертке сторонних данных, стороннему серверу, функции, процедуре, процедурному языку, схеме или табличному пространству.

Доступные права: SELECT, INSERT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER, CREATE, CONNECT, TEMPORARY, EXECUTE и USAGE.

Объекты, к которым может быть применено право представлены в таблице ниже.

Таблица 7. Права и объекты

Право	**Применимо к типам объектов**
SELECT	LARGE OBJECT, SEQUENCE, TABLE (и табличные объекты), столбец таблицы
INSERT	TABLE, столбец таблицы
UPDATE	LARGE OBJECT, SEQUENCE, TABLE, столбец таблицы
DELETE	TABLE
TRUNCATE	TABLE
REFERENCES	TABLE, столбец таблицы
TRIGGER	TABLE
CREATE	DATABASE, SCHEMA, TABLESPACE
CONNECT	DATABASE
TEMPORARY	DATABASE
EXECUTE	FUNCTION, PROCEDURE
USAGE	DOMAIN, FOREIGN DATA WRAPPER, FOREIGN SERVER, LANGUAGE, SCHEMA, SEQUENCE, TYPE

Детальное описание прав представлено в таблице ниже.

Таблица 8. Детальное описание прав

Право	**Описание**
SELECT	Позволяет получать значения из всех или определенного столбца (столбцов) таблицы, представления, материализованного представления или другого табличного объекта. Также позволяет использовать команду COPY TO. Также право необходимо для ссылки на столбцы в конструкциях *UPDATE* или *DELETE*. Для последовательностей право позволяет использовать функцию *currval*. Для больших объектов позволяет объекту быть прочитанным.

Право	**Описание**
INSERT	Разрешает вставку новой строки в таблицу, представление и т. д. Может быть предоставлено на определенные столбцы, в этом случае в команде *INSERT* могут быть перечислены только эти столбцы (а остальные столбцы будут получать значения по умолчанию). Также позволяет использовать команду *COPY FROM*
UPDATE	Разрешает обновление всех столбцов или определенного столбца (столбцов) таблицы, представления и т. д. (На практике любая нетривиальная команда *UPDATE* потребует и права *SELECT*, поскольку она часто ссылается на столбцы таблицы, чтобы определить, какие строки обновлять, и / или для вычисления новых значений столбцов). Конструкции *SELECT ... FOR UPDATE* и *SELECT ... FOR SHARE* также требуют это право, как минимум, для одного столбца в дополнение к праву *SELECT*. Для последовательностей это право позволяет использовать функции *nextval* и *setval*. Для больших объектов это право позволяет писать или очищать (TRUNCATE) объект.
DELETE	Позволяет удалить строки из таблицы, представления и т. д. (На практике любой нетривиальной команде *DELETE* также потребуется право *SELECT*, поскольку она должна ссылаться на столбцы таблицы, чтобы определить, какие строки следует удалить).
TRUNCATE	Позволяет очистить таблицы, представления и т. д.
REFERENCES	Позволяет создать ограничение внешнего ключа, ссылающегося на таблицу или определенные столбцы таблицы.
TRIGGER	Позволяет создать триггер для таблицы, представления и т. д.

Право	**Описание**
CREATE	Для баз данных позволяет создавать новые схемы и публикации в базе данных. Для схем позволяет создавать новые объекты внутри схемы. Для табличных пространств позволяет создавать таблицы, индексы и временные файлы в табличном пространстве и позволяет создавать базы данных, в которых табличное пространство является табличным пространством по умолчанию.
CONNECT	Позволяет пользователю получившему грант подключаться к базе данных. Это право проверяется при создании соединения (в дополнение к проверке любых ограничений прописанным в конфигурационном файле qhb_hba.conf).
TEMPORARY	Позволяет создавать временные таблицы.
EXECUTE	Позволяет вызывать функции или процедуры, включая использование любых операторов, которые реализованы поверх функции. Это единственный тип полномочий, применимый к функциям и процедурам.
USAGE	Для процедурных языков позволяет использовать язык для создания функций на этом языке. Это единственный тип привилегий, применимый к процедурным языкам. Для схем разрешает доступ к объектам, содержащимся в схеме (при условии, что собственные требования к правам объектов также выполнены). Это позволяет получившему право «искать» объекты в схеме. Без этого разрешения можно увидеть имена объектов, например, путем запроса системных каталогов. Кроме того, после отзыва этого разрешения существующие сеансы могут иметь операторы, которые ранее выполняли этот поиск, так что это не является полностью безопасным способом

Право	Описание
	предотвращения доступа к объекту. Для последовательностей позволяет использовать функции *currval* и *nextval*. Для типов и доменов позволяет использовать тип или домен при создании таблиц, функций и других объектов схемы. (Обратите внимание, что это право не управляет всем «использованием» типа, например значениями типа, появляющимися в запросах. Она только предотвращает создание объектов, зависящих от типа. Основная цель этого права - контролировать, какие пользователи могут создать зависимости от типа, которые могут помешать владельцу изменить тип позже). Для оболочек сторонних данных позволяет создавать новые серверы с использованием *foreign-data wrapper. Для сторонних серверов позволяет создавать сторонние таблицы с использованием сервера. Получатели могут также создавать, изменять или удалять свои собственные сопоставления пользователей, связанные с этим сервером.

Запись ALL (или ALL PRIVILEGES) вместо определенного права предоставляет все права, которые могут быть применены к объекту.

Предоставление права для специальной роли PUBLIC указывает, что права должны быть предоставлены всем ролям, включая те, что могут быть созданы позже. PUBLIC может рассматриваться как неявно определенная группа, которая всегда включает все роли.

Таблица 9. Права для объектов

Тип объекта	ALL	PUBLIC
DATABASE	CREATE, TEMPORARY, CONNECT	TEMPORARY, CONNECT
DOMAIN	USAGE	USAGE

Тип объекта	ALL	PUBLIC
FUNCTION или PROCEDURE	EXECUTE	EXECUTE
FOREIGN DATA WRAPPER	USAGE	none
FOREIGN SERVER	USAGE	none
LANGUAGE	USAGE	USAGE
LARGE OBJECT	SELECT, UPDATE	none
SCHEMA	USAGE, CREATE	none
SEQUENCE	SELECT, UPDATE, USAGE	none
TABLE (и табличные объекты)	INSERT, SELECT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER	none
Столбец таблицы	INSERT, SELECT, UPDATE, REFERENCES	none
TABLESPACE	CREATE	none
TYPE	USAGE	USAGE

Если указано WITH GRANT OPTION, получатель права может в свою очередь предоставить его другим лицам. Без этого указания получатель не может этого сделать. Группе PUBLIC право передачи права дать нельзя.

Владелец объекта имеет все права доступа по умолчанию (владельцем по умолчанию является тот, кто создал объект, но право владения может быть передано самим владельцем или суперпользователем (однако в целях обеспечения безопасности владелец может отказаться от некоторых своих прав).

Право удалить объект или каким-либо образом изменить его определение в качестве предоставляемого права не рассматривается; оно присуще владельцу и не может быть предоставлено или отозвано. (Однако аналогичный эффект можно получить, предоставив или отозвав членство в роли, которой принадлежит объект; см.

ниже.) Также владелец неявно имеет право распоряжения всеми правами для своего объекта.

Существует также возможность предоставления прав на все объекты одного типа в рамках одной или нескольких схем. В настоящее время эта функция поддерживается только для таблиц, последовательностей, функций и процедур:

```
ALL TABLES IN SCHEMA имя_схемы [, ...],
ALL SEQUENCES IN SCHEMA  имя_схемы [, ...]
ALL { FUNCTIONS | PROCEDURES | ROUTINES } IN SCHEMA имя_схемы
[, ...]
```

10.2 Предоставление права для ролей

```
GRANT имя_роли [, ...] TO указание_роли [, ...]
    [ WITH ADMIN OPTION ]
    [ GRANTED BY указание_роли ]
```

Этот вариант предоставляет членство в роли для одной или нескольких других ролей. Членство в роли имеет важное значение, поскольку оно передает предоставленные роли права каждому из ее членов.

Участник, получивший членство в роли с указанием WITH ADMIN OPTION, может, в свою очередь, предоставлять членство в роли другим пользователям, а также отменять членство в роли. Без WITH ADMIN OPTION обычные пользователи не могут этого сделать. Роль не имеет права WITH ADMIN OPTION для самой себя, но может предоставить или отозвать членство в роли в сеансе базы данных, в котором пользователь сеанса соответствует роли. Суперпользователи баз данных могут предоставлять или отменять членство в любой роли кому угодно. Роли с правом CREATEROLE могут предоставлять или отменять членство в любой роли, которая не является суперпользователем.

При указании GRANTED BY предоставление права записывается как совершенное конкретной ролью. Этой возможностью могут пользоваться только суперпользователи, за исключением тех случаев, когда она называет ту же роль при выполнении команды.

В отличие от прав, членство в роли не может быть предоставлено группе PUBLIC. Обратите также внимание, что эта форма команды не допускает избыточное слово GROUP в указании роли.

Для отмены прав доступа используется команда REVOKE.

Понятия пользователей и групп были объединены в единый вид сущности, называемый ролью. Поэтому больше нет необходимости использовать ключевое слово GROUP, чтобы определить, является ли участник пользователем или группой. Слово GROUP по-прежнему разрешено в команде, но лишено смысловой нагрузки.

Пользователь может выполнить команды SELECT, INSERT и подобные им со столбцом таблицы, если обладает этим правом для конкретного столбца или для всей таблицы. Предоставление права на уровне таблицы, а затем отмена права для одного из столбцов этой таблицы не даст желаемого эффекта: операция с правами на уровне столбцов не затронет право на уровне таблицы.

Если пользователь, не являющийся владельцем объекта, попытается назначить право доступа к объекту (с помощью GRANT), то эта команда завершится ошибкой, если у пользователя нет никаких прав доступа к объекту. Если же пользователь имеет некоторые права, команда будет выполнена, но предоставит только те права, которые даны ему с правом передачи. Формы GRANT ALL PRIVILEGES выдадут предупреждающее сообщение, если у пользователя вообще нет никаких прав передачи, тогда как другие формы выдадут предупреждение, если пользователь не имеет прав распоряжаться конкретными правами, указанными в команде. (В принципе, эти утверждения применимы и к владельцу объекта, но поскольку владелец всегда рассматривается как обладатель всех прав, такие ситуации невозможны.)

Важно: суперпользователи баз данных могут обращаться ко всем объектам, независимо от настроек прав объекта. Это сравнимо с правами пользователя root в системе Unix. Как и в случае croot, неразумно работать под суперпользователем, за исключением тех случаев, когда это абсолютно необходимо. Если суперпользователь решает выполнить команду GRANT или REVOKE, команда выполняется так, как если бы она выполнялась владельцем объекта воздействия. В частности, права,

предоставленные с помощью такой команды, проявятся как права, назначенные владельцем объекта. Для членства в роли членство будет представлено как назначенное самой ролью.

GRANT и REVOKE также могут выполняться ролью, которая не является владельцем объекта воздействия, но является членом роли, которая владеет объектом, или членом роли, которая обладает правами доступа WITH GRANT OPTION для этого объекта. В этом случае права будут записаны как предоставленные ролью, которая фактически владеет объектом или владеет правами доступа WITH GRANT OPTION. Например, если таблица t1принадлежит роли g1, членом которой является u1, то u1 может предоставлять права на t1 роли u2, но эти права будут предоставлены как назначенные непосредственно ролью g1. Любой другой член роли g1 сможет позже отозвать их.

Если роль, выполняющая команду GRANT, получила на это необходимые права косвенно через несколько путей членства в роли, то не будет определено, какая именно роль назначила право. В таких случаях рекомендуется использовать SET ROLE, чтобы переключиться на конкретную роль, которую вы хотите назначить в качестве выполняющей команду GRANT.

Предоставление прав на таблицу не означает автоматического расширения прав на любые используемые таблицей последовательности, включая те из них, что связаны со столбцами SERIAL. Разрешения на последовательности должны быть установлены отдельно.

Существует команда SCHED, которая расширяет привилегии и позволяет запускать задания на конкретной базе данных.

11 РЕКОМЕНДУЕМЫЕ НАСТРОЙКИ БЕЗОПАСНОСТИ СРЕДЫ ФУНКЦИОНИРОВАНИЯ

11.1 Ограничения программной среды

В данном подразделе описывается реализация требований пункта 1.6.8 «Ограничения программной среды в СУБД «Квант-Гибрид 1.5»» документа ВЕМР.00207-01 94 01 «Технические условия».

СУБД «Квант-гибрид» предназначена для работы в сертифицированных защищенных операционных системах (перечень операционных систем приведен в разделе 1.3). Данные операционные системы реализуют ограничения программной среды при эксплуатации СУБД «Квант-гибрид» в части безопасности функционирования, что выявляет и блокирует загрузку в адресное пространство СУБД программного обеспечения, не включенного в перечень (список) программного обеспечения, разрешенного для выполнения.

Защищенные операционные системы не допускают загрузку стороннего программного обеспечения в адресное пространство любого из запущенных процессов, в том числе и в адресное пространство СУБД «Квант-гибрид». За поддержку данного функционала отвечает механизм виртуальной памяти. В виртуальной памяти каждый процесс работает со своим виртуальным адресным пространством и не может обратиться непосредственно к физической памяти или адресным пространствам других процессов.

Устройство сертифицированных операционных систем, в которых функционирует СУБД, позволяет размещать код СУБД в области памяти, не доступной одновременно для записи и исполнения. Выделение памяти в этих операционных системах происходит автоматически при запуске СУБД (код СУБД и связанных библиотек загружается в раздел оперативной памяти без права на запись).

11.1.1 Запрет на создание процедур, хранимых в базах данных, пользователям баз данных

Функционал запрета на создание процедур, хранимых в базах данных, пользователям баз данных обеспечивают следующие механизмы:

1) Средства идентификации и аутентификации пользователей, позволяющие запретить обращение к базам данных (и самой СУБД) незарегистрированным и неидентифицированным пользователям.
2) Управление доступом к ресурсам, осуществляемое в двух направлениях одновременно:
   - Система ролей и прав, присваиваемых этим ролям/группам ролей. Подробную информацию см. в разделе «Роли в базе данных» настоящего документа.
   - Разграничение доступа, применяемое к объектам доступа, т. е. присвоение данным объектам атрибутов доступа, определяющих, какая операция может (или не может) над ними выполняться. Подробную информацию см. в разделе «Полномочия» настоящего документа.

Примечание: Управление доступом осуществляется по принципу минимально возможного, т. е. по умолчанию ролям/группам ролей предоставляется самый минимальный уровень доступа из имеющихся, а для объектов доступа аналогично по умолчанию указывается минимальный атрибут доступа из имеющихся.

11.1.2 Выявление и блокировка загрузки в адресное пространство СУБД программного обеспечения, целостность которого нарушена

Механизм контроля целостности реализован с помощью фонового процесса integrity_checker. Этот фоновый процесс осуществляет проверку исполняемых модулей и утилит на целостность бинарного файла. Также фоновый процесс осуществляет проверку целостности программного кода хранимых процедур самой СУБД «Квант-гибрид». Проверка состоит в удостоверении корректности контрольных сумм, подсчитанных при сборке установочного пакета СУБД «Квант-гибрид» и после авторизованных изменений. Запускается данный процесс при установке соответствующих параметров для его запуска в файле конфигурации qhb.conf. При этом фоновый процесс integrity_checker запускается вместе со стартом СУБД и продолжает выполняться вплоть до завершения ее работы.

Объектами контроля целостности являются поставляемые бинарные утилиты и программный код хранимых процедур, как встроенных, так и добавленных пользователем в процессе работы СУБД «Квант-гибрид». Для проверки процедур необходимо также, чтобы на кластере были включены контрольные суммы. Для целостности бинарных утилит проверка хэш-сумм происходит в соответствии с файлами, расположенными в каталоге integrity и имеющими расширение .sha256.

Дополнительная информация по данному вопросу, включающая настройку СУБД, содержится в подразделах «Контроль целостности» и «Включение фоновых процессов целостности и безопасности» настоящего документа.

11.2 Межсетевые экраны

Сервер СУБД «Квант-Гибрид 1.5» должен быть полностью изолированным и не допускать никаких входящих подключений, SSH или psql.

Для повышения безопасности сервера базы данных следует заблокировать доступ к узлу, на котором работает база данных, на уровне порта, с помощью межсетевого экрана. По умолчанию, СУБД «Квант-Гибрид 1.5» прослушивает TCP-порт 5432. В операционной системе ОС Альт 8СП можно использовать iptables - наиболее доступную утилиту для управления межсетевым экраном:

```
# Make sure not to drop established connections.
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j
ACCEPT

# Allow SSH.
iptables -A INPUT -p tcp -m state --state NEW --dport 22 -j
ACCEPT

# Allow QHB.
iptables -A INPUT -p tcp -m state --state NEW --dport 5432 -j
ACCEPT

# Allow all outbound, drop everything else inbound.
iptables -A OUTPUT -j ACCEPT
iptables -A INPUT -j DROP
```

```
iptables -A FORWARD -j DROP
```

Примечание: при обновлении правил iptables рекомендуется использовать iptables-apply, который автоматически откатывает изменения в случае, если вы случайно заблокируете себя.

Приведенное выше правило для СУБД «Квант-Гибрид 1.5» позволит любому подключаться к порту 5432. Его можно сделать более строгим, принимая подключения только с определенных IP-адресов или подсетей:

```
# Only allow access to QHB port from the local subnet.
iptables -A INPUT -p tcp -m state --state NEW --dport 5432 -s
192.168.1.0/24 -j ACCEPT
```

Возможность полностью предотвратить входящие подключения к порту 5432 потребует своего рода локального агента, который поддерживает постоянное исходящее соединение с клиентским узлом\ узлами и имеет возможность передавать трафик на локальный экземпляр СУБД «Квант-Гибрид 1.5».

Такой метод называется «обратным туннелированием». Его можно продемонстрировать при помощи удаленного перенаправления портов. Возможно открыть обратный туннель, выполнив следующую команду с узла, на котором работает СУБД «Квант-Гибрид 1.5»:

```
ssh -f -N -T -R 5432:localhost:5432 user@<client-host>
```

При этом, <client-host> должен быть доступным из узла СУБД «Квант-Гибрид 1.5» и иметь запущенного SSH-демона. Команда перенаправит порт 5432 на сервере базы данных на порт 5432 на клиентском компьютере, и вы сможете подключиться к базе данных через туннель:

```
psql "host=localhost port=5432 user=qhb dbname=qhb"
```

11.3 Прослушивание адресов

Целесообразно ограничить адреса, которые прослушивает сервер для клиентских подключений, с помощью параметра listen_addresses файла конфигурации СУБД «Квант-Гибрид 1.5» (qhb.conf). Если узел, на котором работает СУБД «Квант-Гибрид 1.5», имеет несколько сетевых интерфейсов, используйте этот

параметр, чтобы убедиться, что сервер прослушивает только те интерфейсы, через который клиенты будут подключаться к нему:

```
listen_addresses = 'localhost, 192.168.0.1'
```

Если клиенты, подключающиеся к базе данных, всегда находятся на одном и том же узле, отключение прослушивания сокетов TCP может полностью исключить сеть из рассматриваемой картины. Запись пустой строки в параметр прослушиваемых адресов заставляет сервер принимать только соединения сокетов домена ОС:

```
listen_addresses = ''
```

11.4 Безопасность на транспортном уровне

СУБД «Квант-Гибрид 1.5» поддерживает SSL и позволяет использовать его для аутентификации как сервера, так и клиента.

11.5 Серверный SSL

Для аутентификации сервера сначала необходимо получить сертификат, который сервер будет предоставлять подключающимся клиентам. Для этого можно использовать openssl CLI:

```
# Make a self-signed server CA.
openssl req -sha256 -new -x509 -days 365 -nodes \
    -out server-ca.crt \
    -keyout server-ca.key

# Generate server CSR. Put the hostname you will be using to
connect to# the database in the CN field.
openssl req -sha256 -new -nodes \
    -subj "/CN=qhb.example.com" \
    -out server.csr \
    -keyout server.key

# Sign a server certificate.
openssl x509 -req -sha256 -days 365 \
    -in server.csr \
    -CA server-ca.crt \
    -CAkey server-ca.key \
```

```
    -CAcreateserial \
    -out server.crt
```

В производственной среде необходимо убедиться, что эти сертификаты обновлены вовремя.

11.6 Клиентский SSL

Аутентификация клиента по сертификату позволяет серверу проверить личность подключающегося, подтверждая, что сертификат, представленный клиентом, подписан доверенным центром сертификации (CA).

Рекомендуется использовать разные центры сертификации для выдачи сертификатов клиенту и серверу, поэтому ниже приведен пример создания клиентского CA и использования его для подписи сертификата клиента:

```
# Make a self-signed client CA.
openssl req -sha256 -new -x509 -days 365 -nodes \
    -out client-ca.crt \
    -keyout client-ca.key

# Generate client CSR. CN must contain the name of the
database role you# will be using to connect to the database.
openssl req -sha256 -new -nodes \
    -subj "/CN=alice" \
    -out client.csr \
    -keyout server.key

# Sign a client certificate.
openssl x509 -req -sha256 -days 365 \
    -in client.csr \
    -CA client-ca.crt \
    -CAkey client-ca.key \
    -CAcreateserial \
    -out client.crt
```

Обратите внимание, что поле CommonName (CN) сертификата клиента должно содержать имя учетной записи базы данных, к которой подключается клиент. Сервер СУБД «Квант-Гибрид 1.5» будет использовать его для идентификации клиента.

11.7 Конфигурация SSL

Учитывая подразделы выше возможно настроить сервер СУБД «Квант-Гибрид 1.5» (qhb.conf) для приема соединений **SSL**:

```
ssl = on
ssl_cert_file = '/path/to/server.crt'
ssl_key_file = '/path/to/server.key'
ssl_ca_file = '/path/to/client-ca.crt'# This setting is on by
default but it’s always a good idea to# be explicit when it
comes to security.
ssl_prefer_server_ciphers = on

# TLS 1.3 will give the strongest security and is advised
when# controlling both server and clients.
ssl_min_protocol_version = 'TLSv1.3'
```

Последняя часть настройки — обновить файл host-based аутентификации сервера СУБД «Квант-Гибрид 1.5» (qhb_hba.conf), чтобы требовать SSL для всех подключений и аутентифицировать клиентов с помощью сертификатов:

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
hostssl all             all             ::/0                    cert
hostssl all             all             0.0.0.0/0               cert
```

Теперь пользователи, подключающиеся к серверу, должны будут предоставить действительный сертификат, подписанный клиентским CA:

```
psql "host=qhb.example.com \
      user=alice \
      dbname=qhb \
      sslmode=verify-full \
      sslrootcert=/path/to/server-ca.crt \
      sslcert=/path/to/client.crt \
      sslkey=/path/to/client.key"
```

11.8 Безопасность на уровне базы данных

11.8.1 Роли

Выше было рассмотрено, как защитить сервер базы данных СУБД «Квант-Гибрид 1.5» от неавторизованных сетевых подключений, использовать шифрование и убедиться, что сервер и клиенты могут доверять друг другу с помощью взаимной аутентификации TLS. Еще одна часть защиты — выяснить, что пользователи могут делать и к чему у них есть доступ после подключения к базе данных и подтверждения своей личности.

Более подробно, описание ролей представлено в разделе Роли в базе данных.

СУБД «Квант-Гибрид 1.5» имеет комплексную систему прав пользователя, основанную на концепции ролей. В СУБД «Квант-Гибрид 1.5» «роль» является синонимом «пользователя», поэтому любое имя учетной записи базы данных, которое вы используете, скажем, с psql (например, «user = user1»), на самом деле является ролью с LOGIN атрибутом, который позволяет ей подключаться к базе данных. Фактически, следующие команды SQL эквивалентны:

```
CREATE USER user1;
CREATE ROLE user1 LOGIN;
```

Помимо возможности входа в систему, роли могут иметь иные атрибуты, позволяющие им обходить все проверки прав доступа (SUPERUSER), создавать базы данных (CREATEDB), создавать другие роли (CREATEROLE) и другие.

Помимо атрибутов, ролям могут быть предоставлены права доступа, которые можно разделить на две категории: членство в других ролях и привилегии на объекты базы данных.

11.8.2 Предоставление роли прав доступа

Пример отслеживания инвентаризацию серверов:

```
CREATE TABLE server_inventory (
    id            int PRIMARY KEY,
    description   text,
    ip_address    text,
    environment   text,
```

```
    owner         text,
);
```

По умолчанию, конфигурация СУБД «Квант-Гибрид 1.5» включает роль суперпользователя («qhb»), используемую для начальной загрузки базы данных. Использование этой роли для всех операций с базой данных было бы эквивалентно постоянному использованию «root» в ОС Альт 8СП. Следует создавать непривилегированную роль и при необходимости назначать ей права доступа, следуя принципу наименьших полномочий.

Вместо того, чтобы назначать полномочия/роли каждому новому пользователю индивидуально, можно создать «групповую роль» и предоставить другим ролям (сопоставление с отдельными пользователями) членство в этой группе. Например, для просмотра списка серверов, требуется:

```
-- Create a group role that doesn't have ability to login by
itself and-- grant it SELECT privileged on the server
inventory table.CREATE ROLE developer;
GRANT SELECT ON server_inventory TO developer;

-- Create two user accounts which will inherit "developer"
permissions upon-- logging into the database.CREATE ROLE
user1 LOGIN INHERIT;
CREATE ROLE user2 LOGIN INHERIT;

-- Assign both user account to the "developer" group
role.GRANT developer TO user1, user2;
```

Теперь при подключении к базе данных, пользователи унаследуют привилегии групповой роли «разработчик» ("developer") и смогут выполнять запросы к таблице инвентаризации сервера.

Привилегия SELECT распространяется на все столбцы таблицы по умолчанию. Чтобы разрешить своим пользователям просматривать общую информацию об инвентаризации серверов, не позволяя им подключаться, скрывая IP-адрес, необходимо:

```
CREATE ROLE intern;
```

```
GRANT SELECT (id, description) ON server_inventory TO intern;
CREATE ROLE user3 LOGIN INHERIT;
GRANT intern TO user3;
```

Другие наиболее часто используемые привилегии объектов базы данных — INSERT, UPDATE, DELETE и TRUNCATE аналогичны соответствующим SQL выражениями. Также возможно назначить права для подключения к определенным базам данных, создания новых схем или объектов в схеме, выполнения функции и так далее.

11.8.3 Безопасность на уровне строк

Одной из наиболее продвинутых функций системы привилегий СУБД «Квант-Гибрид 1.5» является безопасность на уровне строк, которая позволяет предоставлять полномочия подмножеству строк в таблице. Сюда входят как строки, которые могут быть запрошены с помощью оператора SELECT, так и строки, которые могут быть результатами операторов INSERT, UPDATE и DELETE.

Чтобы начать использовать безопасность на уровне строк, необходимы две вещи: включить ее для таблицы и определить политику, которая будет контролировать доступ на уровне строк.

Основываясь на предыдущем примере, предположим, что вы хотите разрешить пользователям обновлять только свои собственные серверы. Сначала включите RLS в таблице:

ALTER TABLE server_inventory ENABLE ROW LEVEL SECURITY;

Без какой-либо определенной политики СУБД «Квант-Гибрид 1.5» по умолчанию использует политику «запретить», это означает, что никакая роль (кроме владельца таблицы, который обычно является ролью, создавшей таблицу) не имеет к ней доступа.

Политика безопасности строк — это логическое выражение, которое СУБД «Квант-Гибрид 1.5» будет применять для каждой строки, которая должна быть возвращена или обновлена. Строки, возвращаемые оператором SELECT, проверяются на соответствие выражению, указанному в разделе USING, в то время

как строки, обновленные операторами INSERT, UPDATE или DELETE, проверяются на соответствие с WITH CHECK выражением.

Ниже представлено определение настроек, которые позволяют пользователям видеть все серверы, но обновлять только свои собственные, определенные полем «владелец»:

```
CREATE POLICY select_all_servers
    ON server_inventory FOR SELECT
    USING (true);

CREATE POLICY update_own_servers
    ON server_inventory FOR UPDATE
    USING (current_user = owner)
    WITH CHECK (current_user = owner);
```

Обратите внимание, что только владелец таблицы может создавать или обновлять для нее политику безопасности строк.

11.8.4 Аудит

Ведение подробного контрольного журнала — одна из опций безопасности системы, которое часто упускается из виду. Мониторинг доступа на уровне сети или на уровне узла для сервера базы данных выходит за рамки этого руководства. Рассмотрим какие есть возможности, когда дело касается самого сервера СУБД «Квант-Гибрид 1.5».

Самое простое, что можно сделать для улучшения видимости того, что происходит в базе данных, — это включить подробное ведение журнала. Добавьте следующие параметры в файл конфигурации сервера, чтобы включить ведение журнала всех попыток подключения и всех выполненных операторов SQL:

```
Log successful and unsuccessful connection attempts.
log_connections = on

Log terminated sessions.
log_disconnections = on

Log all executed SQL statements.
```

```
log_statement = all
```

Для более детализированного аудита СУБД «Квант-Гибрид 1.5» можно использовать сторонние решения и штатные возможности ОС.

12 РАБОТА С РАСШИРЕНИЯМИ И ДОПОЛНИТЕЛЬНЫМИ МОДУЛЯМИ

12.1 Модуль безопасного хранения

Модуль безопасного хранения СУБД «Квант-Гибрид» (Quantum Secure Storage или QSS), позволяет создавать таблицы, которые шифруются с помощью криптоалгоритма «Кузнечик» (ГОСТ 34.1-2015) при записи на диск.

12.1.1 Описание

Шифрование осуществляет отдельное программное обеспечение — средство криптографической защиты (СКЗИ) Quantum Secure Storage (QSS).

Примечание: данный продукт приобретается и лицензируется отдельно.

Сервер QSS можно устанавливать как одновременно с СУБД, так и после, т. е. QSS можно установить и начать использовать, когда СУБД уже находится в эксплуатации.

Сервер QSS следует запускать на том же хосте, что и сервер СУБД, в противном случае может наблюдаться существенное снижение производительности. Клиентские библиотеки, необходимые для взаимодействия СУБД с QSS, устанавливаются в том же пакете, что и сам сервер QSS. Пользователя, от имени которого запускается СУБД, нужно добавить в специальную группу *qss-client* (группа создается автоматически при установке QSS). Также следует создать в QSS рабочий ключ, который будет использоваться для шифрования.

Подробную информацию о том, как устанавливать, настраивать QSS и создавать в нем ключи, см. в документации по QSS.

Дополнительные требования при использовании QSS:

- при необходимости защиты информации сохраняемой в СУБД в соответствии с законодательством РФ, необходимо использовать СКЗИ соответствующего класса для передачи защищаемой информации по каналам связи;
- при использовании СУБД совместно с сертифицированным СКЗИ QSS необходимо выполнять требования по безопасности приведенные в ПП на СКЗИ в п. 4, 5 и 6;

– при использовании СУБД совместно с сертифицированным СКЗИ QSS необходимо использовать модуль доверенной загрузки, который имеет сертификат соответствия ФСБ России по классу защиты 2 и выше.

12.1.2 Использование

12.1.2.1 Возможности

Возможности СУБД при использовании QSS:

1) Шифрование отдельных таблиц. Шифрование происходит на уровне отдельных таблиц; включение шифрования таблицы включает также шифрование индексов этой таблицы, связанной таблицы TOAST и сообщений WAL о действиях в этой таблице. Столбцы определенных типов, хранящиеся не в основном файле таблицы и не в TOAST (например, большие объекты (LOB), тип ***rbytea***), будут незашифрованными. Информация о том, как включить шифрование Rbytea, описана ниже в разделе по настройке Rbytea для QSS.
2) Шифрование отдельных столбцов отсутствует.
3) Шифрование материализованного представления. Рекомендуется использовать шифрование материализованного представления, если в составе есть шифрованные данные.

   Важно: автоматически шифрование данных при создании шифрованного материального представления не включается.
4) Нельзя включить шифрование таблиц каталога (например, шифровать тексты хранимых процедур на SQL).

   Важно: если у зашифрованной таблицы есть DEFAULT CONSTRAINT, то значение по умолчанию хранится в каталоге в незашифрованном виде. То же самое и для текста триггеров, заполняющих шифрованную таблицу.
5) Нельзя включить шифрование партиционированных таблиц.
6) Данные на диске и в кэше операционной системы зашифрованы (и это является преимуществом по сравнению с шифрованием на уровне диска),

данные в памяти СУБД не зашифрованы, так как к ним нужен оперативный доступ.

7) Не поддерживается копирование шифрованной таблицы из базы-шаблона в новую базу во время CREATE DATABASE; в частности, нельзя создавать шифрованные таблицы в базе ***template1***.

12.1.2.2 Конфигурационные параметры

Настройки QSS находятся в отдельном конфигурационном файле ***qss2_config.toml***; пример файла генерируется при инициализации кластера (при помощи расширения ***initdb***):

key = "key_name" # идентификатор ключа, существующего в QSS, которым будет производиться шифрование;

shmem = true # будет ли шифрование проводиться через разделяемую память;

provider = "compressed-append" # "compressed-append" или "two-phase" - способ хранения атрибутов nonce, tag шифрованных страниц в файлах СУБД;

socket_address = "/run/qss/client.socket" # для связи с сервером qss; при установке QSS по умолчанию менять нет необходимости.

Параметры ***shmem*** и ***provider*** не рекомендуется менять без консультации с техподдержкой, поскольку значения по умолчанию обеспечивают наилучшую производительность. Параметр ***provider*** **нельзя** менять, когда уже имеются зашифрованные данные.

В основном конфигурационном файле использование QSS включается параметром:

```
qss_mode = 2 #
```

Это целочисленный параметр, допустимые значения 2 и 0:

- При запуске СУБД с **qss_mode = 2** проверяется корректность настройки QSS, и если обнаруживаются какие-либо проблемы, то QHB не запускается.
- Запуск СУБД с **qss_mode = 0** при наличии зашифрованных таблиц разрешен, но обратиться к зашифрованным таблицам будет невозможно.

12.1.2.3 Создание таблицы

При создании зашифрованной таблицы или материализованного представления нужно указать предложение ***using qss***:

```
create table tab_qss(c1 int, c2 text) using qss;


create materialized view mat_view_qss using qss as
    select c1, string_agg(c2, ','), count(1) from tab_qss
group by c1;
```

Корректно превратить уже созданную таблицу в зашифрованную или наоборот (с помощью `ALTER TABLE`) нельзя.

12.1.2.4 RBytea

Модуль RBytea позволяет создавать столбцы типа RBytea. Шифрование всех столбцов типа RBytea регулируется единым параметром, а не по отдельности для каждой таблицы. Однако допускается иметь одновременно шифрованные и не шифрованные значения RBytea. Каждое отдельное значение шифруется (или не шифруется) в зависимости от значения параметра в конкретный момент.

Таким образом, возможно включать и выключать шифрование RBytea. Но включение не зашифрует старые значения RBytea.

Параметр:

```
rbytea.filesystem_qss_mode = 2
```

При qss_mode = 0 значение этого параметра игнорируется.

Подробную информацию о параметрах конфигурирования RBytea см. в разделе Параметры конфигурации расширения Rbytea.

Примечание

При шифровании RBytea невозможен такой режим эксплуатации значений этого типа, когда данные типа RBytea хранятся в единственном экземпляре для нескольких узлов кластера на сетевом хранилище, т. к. разные узлы будут использовать разные ключи QSS.

12.1.2.5 Шифрование статистик

Статистики по данным таблицы могут содержать чувствительную информацию, такую как наиболее частотные значения в каждой колонке. Статистики

по шифрованным таблицам желательно тоже шифровать. Для этого надо дополнительно включить параметр **qss_encrypt_statistic**.

После этого статистики помещаются вместо таблиц каталога **pg_statistic**, **pg_statistic_ext_data** в аналогичные шифрованные таблицы **qss_statistic**, **qss_statistic_ext_data**. Статистики нешифрованных таблиц тоже будут храниться там.

После переключения статистики будут недоступны в моменте, что может приводить к плохим планам выполнения запросов. У вас есть следующие варианты, как их заполнить:

- запустить VACUUM ANALYZE;
- подождать, пока тоже самое сделает автовакуум (т.е. если ничего не делать, то статистики соберутся "сами");
- скопировать имеющиеся статистики запросом, чтобы не ждать VACUUM ANALYZE:

```
delete from qss_statistic;
insert into qss_statistic select * from pg_statistic;
delete from qss_statistic_ext_data;
insert into qss_statistic_ext_data select * from
pg_statistic_ext_data;
```

(При выключении **qss_encrypt_statistic** копирование в обратную сторону.)

После включения шифрования статистик рекомендуется удалить нешифрованные:

```
delete from pg_statistic;
delete from pg_statistic_ext_data;
```

ВНИМАНИЕ: таблицу **pg_statistic_ext** трогать не надо ни в каких сценариях!

12.1.2.6 Обслуживание

В этом разделе в ряде сценариев предлагается «расшифровать таблицу»/«зашифровать таблицу» подразумевается выгрузка содержимого таблицы и последующее ее пересоздание.

Это можно сделать разными способами, например, перенести данные в другую таблицу в той же базе. При этом вам может понадобиться удалить и заново воссоздать ограничения, индексы и т. д.

Примечание: к данным операциям применимы общие рекомендации по ETL (Extract, Transform, Load — извлечение, преобразование, загрузка).

Важно: нельзя загружать данные в зашифрованную таблицу с помощью QDL (с незашифрованной таблицей этот модуль использовать можно).

12.1.2.7 Смена ключа

ВНИМАНИЕ: не меняйте ключ, если у вас есть зашифрованные данные!

ВНИМАНИЕ: смена ключа возможна только при наличии текущего активного ключа. Сменить ключ после того, как у ключа, на котором зашифрованы данные, истек срок действия, будет невозможно. Узнать срок действия ключа можно в административном приложении QSS.

1. Расшифруйте все данные.
2. Убедитесь в отсутствии зашифрованных таблиц.
3. Если использовались зашифрованные статистики, то эти данные надо удалить: выключить qss_encrypt_statistic, потом выполнить:

   ```
   truncate table qss_statistic;
   truncate table qss_statistic_ext_data;
   ```

   (truncate table можно делать шифрованным таблицам и при выключенном QSS).
4. Создайте другой ключ в QSS, запишите этот ключ в файл qss2_config.toml и перезапустите СУБД.
5. Снова зашифруйте таблицы.
6. Опционально включить шифрование статистик.

Важно: Не допускайте истечения срока действия ключа, иначе прочесть данные будет невозможно. Узнать срок действия ключа можно в административном приложении QSS.

12.1.2.8 Создание резервной копии

Резервную копию можно будет восстановить только на тот же хост, поскольку ключи шифрования привязаны к хосту и не могут быть перенесены на другую машину (это основное правило использование QSS, связанное с требованиями безопасности). Если хост, на котором ранее располагались QSS и QHB, более не может использоваться по каким-либо причинам (например, техническая авария), то возможно восстановление QHB на другой машине при условии восстановления из резервной копии ключей QSS (для этого должна быть резервная копия файла storage.bin).

Резервную копию рекомендуется создавать с помощью утилиты **qbackup**. При этом резервная копия создается обычным способом, однако не рекомендуется использовать потоковое резервное копирование с указанием ***wal-method=stream***. Для формирования зашифрованной потоковой резервной копии требуется наличие непрерывной архивации WAL и потокового резервного копирования в режиме ***wal-method=fetch*** или ***wal-method=none***.

Также нельзя использовать автоматическое переключение узла кластера, с которого проводится резервное копирование. Файлы резервной копии, взятые с разных хостов, не должны попадать в один каталог резервных копий, поэтому автоматическое переключение мастера не должно приводить к переключению резервного копирования на другой узел.

12.1.2.9 Репликация

На каждом узле необходим свой сервер QSS. У каждого узла должен быть свой ключ, при этом имена ключей могут совпадать. Это позволит иметь на всех узлах идентичные файлы **qss2_config.toml**.

Допустимы следующие варианты репликации:

- потоковая репликация (синхронная и асинхронная);
- логическая репликация;
- репликация основной-резервный на основе триггеров;
- репликация на основе SQL в ПО промежуточного слоя (через репитер запросов и т. п.).

Варианты репликации, которые **не** будут работать совместно с **QSS**:

- на разделяемых дисках (сетевой диск или репликация на уровне файловой системы);
- доставка журналов упреждающей записи.

Важно: на каждом узле должно быть задано одинаковое значение параметра qss_mode. При включении и настройке шифрования на мастере обязательно требуется включение и настройка шифрования на реплике **до начала** заполнения шифрованной таблицы данными. В противном случае репликационный узел прекратит функционирование.

Важно: подключать узел к потоковой репликации можно, только когда нет зашифрованных таблиц. В том числе не должно быть включено шифрование статистик, таблицы **qss_statistic** и **qss_statistic_ext_data пустые**.

12.1.3 Особенности и ограничения шифрования в QHB:

- Отсутствие возможности зашифровать и расшифровать (с помощью команды `ALTER TABLE`) уже созданную таблицу.
- При шифровании RBytea нельзя хранить данные этого типа в единственном экземпляре для нескольких узлов кластера на сетевом хранилище.
- Перешифрование возможно только путем расшифрования и повторного шифрования.
- Отсутствие возможности загружать данные в зашифрованные таблицы с помощью модуля QDL.
- Восстановление резервной копии возможно только на тот же хост, с которого она была сделана.
- Все узлы кластера должны быть созданы до начала шифрования. Подключить новый узел потоковой репликации после начала шифрования нельзя.
- Отсутствие возможности зашифровать партиционированные таблицы.
- Отсутствие возможности использовать для шифрованных таблиц параметр **APPEND_ONLY**.

– Отсутствие возможности шифрования таблиц каталога (например, pg_proc).
– Отсутствие возможности шифрования отдельных столбцов.
– База, в которой есть шифрованные таблицы, не может быть использована как шаблон в CREATE DATABASE.

12.2 Модули лицензирования

12.2.1 qhb-serial

Утилита идентификации аппаратуры хоста ***qhb-serial*** поставляется в составе одноименного пакета ***qhb-serial***.

При установке она размещается в общедоступном каталоге, обычно в **/usr/bin/**.

Утилита используется для запроса идентификаторов аппаратуры хоста или виртуальной машины, куда устанавливается QHB, для передачи их вендору или дилеру QHB. По этой информации генерируется лицензионный файл, привязанный именно к этому хосту или виртуальной машине.

Запуск утилиты:

```
qhb-serial
```

Параметры:

запуск без параметров —вывести ***serial number*** текущего хоста, если он не является виртуальной машиной.

```
-u
--uuid
```

Вывести вместо ***serial number*** текущего хоста ***system-uuid*** виртуальной машины.

```
-s
--cpus
```

Вывести количество ядер хоста или виртуальной машины, которое будет учитываться при проверке лицензии.

```
-r
--report
```

Сформировать полный вывод параметров хоста или виртуальной машины для передачи вендору и формирования файла лицензии.

```
--help
```

Показать справку об аргументах командной строки qhb-serial и завершиться.

Пример запуска утилиты:

```
qhb-serial --report
```

12.2.2 qhb-license

Утилита просмотра файла лицензии **qhb-license** поставляется в составе пакета ***qhb-license-bin***.

При установке она размещается в общедоступном каталоге, обычно в **/usr/bin/**.

Утилита используется для просмотра параметров лицензионного файла, полученного от вендора или дилера QHB.

Пример запуска утилиты:

```
qhb-license --license-path /usr/local/qhb/license
```

Пример вывода утилиты:

```
Идентификатор лицензии: fa4764bf-5fd8-4579-b8cb-d90b7259f62d
Идентификатор аппаратного обеспечения (HID):
37245DA9E484489EDD71F263E0CDF9F8
Идентификатор системы (UUID):
B5AD08BC368E984DA7AD7A407D7F705D
Дата начала действия лицензии: 2023-03-15
Дата начала срока тех. поддержки: 2023-01-10
Дата окончания срока тех. поддержки: 2023-11-11
Имя пользователя: USERNAME
Код пользователя: USERCODE
Имя продукта: СУБД Квант-Гибрид
Код продукта: QHB-std1
Код сделки: BARGAINCODE
Номер релиза: 1.5
Редакция: FULL
Тип лицензии: TEST
Список лицензионных метрик
        CONNECTIONS: 2
```

```
        PROC: 8
Тип продукта: DBMS
Дата окончания действия лицензии: 2023-11-11
```

Для получения краткой справки запустите утилиту с флагом ***--help***.

```
qhb-license --help
```

12.2.2.1 Получение файла лицензии

Клиент получает лицензионный файл, который необходим для использования QHB. Вместе с файлом лицензии клиент получает номер лицензии, по которому его можно идентифицировать. Данный номер также необходим для обращения в техническую поддержку.

12.2.3Применение лицензионных параметров

При запуске QHB ожидает наличия лицензионного файла **license** в каталоге, куда он установлен, например **/usr/local/qhb/license**.

Для проверки лицензии требуется произвести идентификацию аппаратуры хоста пользователя, которая выполняется пакетом *qhb-serial*. Данная утилита также необходима для запуска QHB и утилит, использование которых предусматривает наличие лицензионного файла.

Примечание: утилита qhb-serial может не требоваться для запуска таких утилит, как qcp и qdl, если в лицензионном файле идентификация аппаратуры хоста не заполняется.

По проверке соответствия параметров лицензирования, указанных в лицензионном файле текущего дистрибутива QHB, выводятся следующие сообщения:

- первоначально при выводе версии печатается краткая информация о сроке действия лицензии;
- далее в обычный журнал сообщений выводятся найденные ошибки загрузки и валидации лицензии.

12.2.3.1 Запрос лицензионных параметров во время работы

Для удобства администратора СУБД предоставляется SQL-функция, которая позволяет получить детали текущей используемой лицензии.

Таблица 10. Параметры функции

Допустимые значения	Описание
lid	Идентификатор лицензии
hid	Идентификатор аппаратного обеспечения
system_uuid	Универсальный уникальный идентификатор системы
license_start	Дата начала действия лицензии
support_start	Дата начала действия техподдержки
support_stop	Дата окончания действия техподдержки
valid_thru	Дата окончания действия лицензии
user_name	Имя пользователя
user_code	Код пользователя
release	Текущий релиз
product_name	Название продукта
product_code	Код продукта
bargain_code	Код сделки
license_type	Тип лицензии
edition	Редакция
product_type	Тип продукта

Например:

```
# select qhb_license_info('product_code');
-- QHB

# select qhb_license_info('release');
```

```
-- 1.3
```

Дополнительно предоставляется SQL-функция, которая позволяет получить сразу все детали текущей используемой лицензии в виде таблицы.

```
# select attname, attvalue from qhb_license();
```

При невозможности обработать указанный параметр функция ***qhb_license_info*** возвращает пустую строку в виде значения типа ***text***.

Возвращаемые данные соответствуют значениям, указанным в самой лицензии (лицензионный файл), и могут отличаться от значений, соответствующих сборке СУБД.

12.2.3.2 Действия клиента в случае ошибки загрузки лицензии

При обычной проверке лицензии (при запуске СУБД) могут возникать два типа ошибок:

- ошибки загрузки лицензии;
- ошибки валидации лицензии относительно текущей сборки/релиза QHB.

В случае ошибки загрузки лицензии следует удостовериться, что:

- лицензия расположена по ожидаемому продуктом пути: в каталоге установки, например **/usr/local/qhb/license**;
- файл лицензии не был изменен после получения от коммерческого отдела (файл подписан криптографически, и любые правки делают его недействительным);
- у пользователя QHB есть права на чтение лицензионного файла.

В случае соблюдения этих условий и повторного возникновения ошибки следует обратиться в техническую поддержку.

12.2.3.3 Действия клиента в случае ошибки валидации лицензии

В случае ошибки валидации лицензии детальная информация об ошибке доступна в начале файла журнала СУБД, сформированного при запуске.

Сообщение включает в себя результат валидации (каждый некорректный параметр выводится на новой строке) и краткое указание для дальнейших действий, локализованное для текущих настроек ОС (**LC_ALL=ru/en**).

12.2.3.4 Лицензирование QDL и QCP

При запуске данные утилиты ожидают наличие лицензионного файла ***qdl-license*** или ***qcp-license*** (аналогично QHB) в директории местонахождения исполняемого файла.

Сообщения об ошибках загрузки и валидации лицензии выводятся в лог соответствующих утилит (в зависимости от настроек это могут быть стандартные потоки ввода/вывода либо файл журнала).

Возможность запросить лицензионные параметры для утилит во время их работы в данном релизе не предусмотрена.

В случае ошибки валидации лицензии детальная информация об ошибке доступна в виде результата валидации (каждый некорректный параметр выводится на новой строке) и краткого указания для дальнейших действий (вывод зависит от текущих настроек).

12.3 Асинхронный пул соединений

Асинхронный пул соединений (Quantum Connection Pool, QCP) представляет собой балансировщик сетевой нагрузки предназначенный для оптимального использования серверных подключений. Этот модуль обеспечивает подключение и поддержку связи с удаленными клиентами.

12.3.1 Общий принцип работы

QCP принимает входящие подключения от удаленных клиентов по адресу, указанному в параметре **listening_address** (по умолчанию – 0.0.0.0:8080), и перенаправляет их трафик серверам баз данных, перечисленным в разделе **servers**. Соединения с серверами устанавливаются (и завершаются) автоматически, по мере необходимости.

В зависимости от режима работы, задаваемого параметром **relay_mode**, перенаправление трафика данных от клиентов к серверам осуществляется одним из следующих способов:

1) **relay_mode: Session**: при первом обращении к серверу клиенту выделяется уникальное соединение с базой данных, которое возвращается в пул только при отключении клиента.
2) **relay_mode: Smart**: при обращении к серверу клиенту выделяется уникальное соединение с базой данных, которое возвращается в пул тогда и только тогда, когда в ответе от сервера будет стоять флаг **Idle** (транзакция неактивна (вне блока транзакции)).

С полным описанием настроек можно ознакомиться в примере конфигурационного файла ***qcp/config-example.yaml***.

12.3.2 Запуск и работа

Запуск осуществляется с помощью утилиты ***qcp*** (`qcp --help` для списка параметров), остановка – утилитой ***qcp-ctrl*** (`qcp-ctrl --help` для списка параметров). Например, чтобы остановить запущенный экземпляр QCP, необходимо выполнить команду `qcp-ctrl quit`.

Вывод записей журнала в процессе работы QCP контролируется параметром **log_output** (см. ***qcp/config-example.yaml***), при этом уровень протоколирования задается параметром **log_level**. Например, чтобы выводить записи журнала уровня *Info* и выше в файл ***/tmp/qcp.log***, необходимо задать следующие параметры в файле конфигурации:

```
log_level: Info
log_output:
    file: /tmp/qcp.log
```

12.3.3 Потребление памяти

При запуске программы единовременно выделяется количество памяти, указанное в разделе **arena** файла конфигурации:

```
# Конфигурация памяти (необязательно)
arena:
    chunk_size: 65 KB # Размер одного блока памяти;
                      # поддерживаются суффиксы B, KB, MB, GB
    # Общее количество потребляемой памяти можно указать,
```

```
        # либо используя параметр «количество блоков памяти»:
        chunks_count: 3150 # Количество таких блоков
        # либо указав общее количество напрямую:
        total_size: 3.1 GB # Поддерживаются суффиксы B, KB, MB, GB
```

12.3.4 Подключение к СУБД

QCP может подключаться либо к одиночному серверу СУБД, либо к кластеру СУБД.

Для подключения к серверу СУБД в разделе **server:** необходимо указать его адрес, например:

```
    server:
        # Одиночный сервер
        address: # Тип и адрес подключения (TCP или сокет домена Unix)
          Tcp: "127.0.0.1:1234"
          #          Unix:          "/home/mexus/devel/optim/qhb-with-
rust/build/dbsockets/.s.PGSQL.5432"
```

Независимо от того, осуществляется ли подключение к кластеру или к одной СУБД, для описания подключений используются следующие необязательные параметры:

- **inactive_timeout**: временной интервал, после которого неактивное соединение с сервером будет закрыто при условии наличия минимального количества подключений.
- **lifetime**: примерное максимальное время жизни соединения к серверу, по истечении которого оно будет закрыто.
- **heart_beat_interval**: интервал проверки допустимости соединения с сервером в период, когда подключение не ассоциировано ни с одним клиентом.

12.3.5 Многопользовательский режим

QCP поддерживает возможность подключения к кластеру СУБД, используя различные пары пользователь-база данных.

Для каждой из таких пар создается отдельный пул подключений. Когда клиент подключается к QCP, он указывает логин пользователя и имя базы данных, и, в соответствии с этими параметрами, QCP помещает его в нужный пул.

При попытке подключения к QCP с парой пользователь-база данных, которая отсутствует в конфигурации, клиент получит ошибку:

```
    FATAL:  Unknown user name, database name, or invalid combination of
user and database
    -- КРИТИЧНО: Неизвестное имя пользователя, имя базы данных или
недопустимое сочетание пользователя и базы данных
```

При этом в логах QCP появится примерно следующее сообщение:

```
    [qcp::relay_helper] [WARN] Client ... startup sequence failed:
Unknown user-database pair: user "user", db "db"
    --  (сбой  в  последовательности  запуска:  неизвестная  пара
пользователь-БД: пользователь "user", БД "db")
```

Пары пользователь-база данных перечисляются в секции **connections:** раздела **server:**, например:

```
    # Первый пул подключений.
    - username: user # Имя пользователя
      password: something # Необязательный параметр: пароль для
                          # аутентификации
      database: utilities # Необязательный параметр: имя базы данных.
                          #  Если  не  указана,  в  качестве  имени  базы
будет
                          # использоваться имя пользователя.
      backends: # Количество подключений с данным именем пользователя
и
                # базой
          initial: 1 # .. при старте пула
          min: 1     # .. минимальное
          max: 3     # .. максимальное
      # Необязательная секция. Описывает то, как клиент должен быть
      #  аутентифицирован  на  QCP.  Если  секция  не  задана,  то  для
авторизации
```

```
        # пользователя с данным именем пользователя для данной базы на
QCP
        # будет использоваться тот же пароль, какой задан для сервера,
если
        # он задан.
        client_auth:
            # Метод аутентификации: trust или md5.
            method: md5
            # Пароль для авторизации. Если не задан, будет использоваться
            # тот же, что и для подключения QCP к базе.
            password: lol2
        options: # Необязательный параметр: дополнительные параметры
                 # подключения, передаваемые базе данных
            option_name: option_value
    # Второй пул подключений.
    - username: qhb # Имя пользователя
      backends: # Количество подключений с данным именем пользователя
и
                # базой
            initial: 2 # .. при старте пула
            min: 2     # .. минимальное
            max: 5     # .. максимальное
```

Для описания каждого пула подключений для пары пользователь-база данных требуется указать имя пользователя (**username**) и имя базы данных (**database**). Поле **database** можно пропустить, тогда оно примет то же значение, что и **username**.

Помимо этого, необходимо указать ограничения на количество подключений к СУБД в данном пуле, в секции **backends:**:

- **initial** указывает, сколько будет открыто подключений к СУБД с данной парой пользователь-база данных при старте QCP,
- **min** – ограничение на минимальное количество подключений к СУБД, может быть меньше, чем **initial**.
- **max** – максимальное количество подключений к СУБД. Должно быть не меньше, чем **initial** и **min**.

Если для подключения к СУБД с данными пользователем и базой данных требуется пароль, то его можно указать в поле **password**.

Так же доступна секция **options:**, позволяющая передать произвольные аргументы командной строки для сервера в виде ***имя: значение***.

12.3.6 Авторизация клиентов на QCP

По умолчанию для каждой пары пользователь-база данных для авторизации пользователей QCP использует такие же настройки, как и для авторизации QCP на СУБД: если СУБД требует пароль, то и QCP будет требовать от пользователя его же, если СУБД не требует пароль, то и QCP не будет.

Данное поведение можно переопределить, используя секцию **client_auth:**:

- поле **method** позволяет переопределить метод авторизации,
- поле **password** позволяет задать пароль, отличный от пароля, используемого для подключения к СУБД.

12.3.7 Применение параметров без перезапуска QCP

Некоторые параметры конфигурации можно изменить, не перезапуская QCP. Такой способ называется «перезагрузка файла конфигурации».

Для этого необходимо изменить требуемые параметры в конфигурационном файле, после чего выполнить

```
sudo systemctl reload qcp
```

Если QCP работает не под управлением ***systemd***, то необходимо определить идентификатор процесса QCP (обратитесь к системному администратору, чтобы узнать, как это сделать на вашей ОС), после чего выполнить следующую команду, подставляя найденный идентификатор вместо **<*pid*>**:

```
sudo kill -USR1 <pid>
```

Для того чтобы обновленная конфигурация применилась, необходимо, чтобы конфигурационный файл был допустимым. В противном случае в логах QCP отобразится невозможность применения обновленной конфигурации, и QCP продолжит работать в прежнем режиме. Частичное применение конфигурации в таком случае исключено.

Если применение возможно, в логах появится соответствующая запись:

```
[INFO] Reloaded configuration file /путь/к/config.yaml
-- [ИНФОРМАЦИЯ] перезагружен файл конфигурации /путь/к/config.yaml
```

Обратите внимание, что для применения обновленной конфигурации требуется некоторое время, которое может достигать нескольких секунд при высокой нагрузке.

Применение параметров подключения для каждой пары пользователь-база данных сопровождается записями в логе:

```
[INFO] Updating connection configuration for user "qhb", db "qhb"
[INFO] Applied new connection parameters for user "qhb", db "qhb"
-- [ИНФОРМАЦИЯ] Обновление конфигурации подключения для пользователя "qhb", БД "qhb"
-- [ИНФОРМАЦИЯ] Добавлены новые параметры подключения для пользователя "qhb", БД "qhb"
```

Применение параметров входящих подключений сопровождается записью в логе:

```
[INFO] Incoming parameters updated
-- [ИНФОРМАЦИЯ] Обновлены параметры входящих подключений
```

Изменение следующих параметров не может быть осуществлено без перезапуска QCP, поэтому они будут проигнорированы QCP при перезагрузке файла конфигурации:

- **arena**
- **log_level**
- **log_output**
- **log_colouring**
- **log_timestamps**
- **worker_threads**

Следующие параметры будут обновлены во время следующего их использования:

- **accept_clients_no_backends**: параметр будет обновлен при подключении следующего клиента, либо же когда появится доступное подключение к СУБД, если прием новых подключений от клиентов был приостановлен.
- в секции **servers_retry_back_off** параметры **min_wait**, **max_wait** и **max_wait_at** будут обновлены при следующей попытке установить подключение к СУБД.
- в секции **server** параметры **inactive_timeout**, **lifetime**, **check_interval** и **heart_beat_interval** будут применены к подключениям к СУБД, установленным после перезагрузки файла конфигурации.

Изменение типа подключения к СУБД приведет к отключению всех текущих клиентов и разрыву всех текущих подключений к СУБД.

Удаление подключения из секции **connections:** раздела **server:** приведет к отключению всех клиентов, использующих такие подключения, а также к разрыву всех таких подключений к СУБД.

Изменение имени пользователя (**username**), имени базы данных (**database**), пароля (**password**) или параметров командной строки сервера (**options**) у подключения приведет к отключению всех клиентов, использующих такие подключения, а также к разрыву всех таких подключений к СУБД и последующему переподключению к СУБД с обновленными параметрами.

Изменение поля **client_auth** у подключения будет применено для всех последующих подключений к QCP.

12.3.8 Инструкция по установке и использованию QCP

В данной инструкции представлены простые шаги по установке QCP, а также базовая настройка сервиса ***systemd***.

Чтобы начать использовать QCP, необходимо установить поставляемый пакет в систему. Если вы устанавливаете пакеты из репозитория СУБД «Квант-гибрид», можно воспользоваться пошаговой инструкцией по начальной загрузке, установке и запуску. После подключения к репозиторию установить пакет, например, для Ред ОС, можно следующим образом:

```
su yum install qcp
```

После установки будет создан файл с примером конфигурации в ***/etc/qcp/config-example.yaml***. Его необходимо скопировать в ***/etc/qcp/config.yaml*** и отредактировать под свои нужды, выставив необходимые параметры подключения к СУБД.

Управление сервисом осуществляется через поставляемый юнит ***systemd***.

Перед первым запуском, после установки QCP, необходимо выполнить перезагрузку демона ***systemd***, чтобы изменения вступили в силу:

```
systemctl daemon-reload
```

Запуск QCP:

```
sudo systemctl start qcp
```

Остановка QCP:

```
sudo systemctl stop qcp
```

Для того чтобы QCP автоматически запускался при старте системы, выполните следующую команду:

```
sudo systemctl enable qcp
```

При изменении конфигурации необходимо выполнить следующую команду, чтобы QCP перечитал файл конфигурации:

```
sudo systemctl reload qcp
```

12.3.9 Настройка QCP после запуска контейнеров

По умолчанию сервис QCP выключен. Для его запуска необходимо в файле ***docker-compose.yaml*** убрать (закомментировать) запись

```
entrypoint: ["echo", "Service qcp disabled"]
```

из раздела **services: -> qcp:**, а также настроить сервис, в частности, параметры подключения к серверу.

Настройка QCP осуществляется посредством редактирования файла ***./qcp/config.yaml***.

Обязательно замените в разделе **servers**:

– *%USER%* – на имя пользователя, от имени которого запускается `docker-compose`

– *%DATABASE%* – на имя базы данных; в простейшем случае база создается с именем, совпадающим с именем пользователя.

В дальнейшем, если вы создадите другую базу данных, этот параметр может потребоваться изменить. По умолчанию сервис QCP работает на порту 8080, при необходимости этот параметр можно переопределить (значения параметров применяются только при запуске).

Отредактируйте файл ***docker-compose.yaml*** следующим образом:

В разделе **services: -> qhb: -> volumes:** замените

```
- ./qhb/pgdata:/qhb-data/
```

на

```
- /путь/до/pgdata:/qhb-data/
```

где ***/путь/до/pgdata*** – абсолютный путь до каталога базы данных с существующей базой данных.

В разделе **services: -> qhb: -> environment:** замените *- USER* на *- USER=USERNAME*.

Запустите контейнер с помощью команды

```
docker-compose -e USER="USERNAME" up
```

где ***USERNAME*** – имя пользователя, владеющего каталогом ***/путь/до/pgdata***.

Пример отредактированного файла ***docker-compose.yaml***:

```
version: "3.3"
services:
  qhb:
    build: qhb
    image: qhb
    network_mode: "host"
    environment:
      - USER=qhb
    volumes:
      - ./qhb/pgdata:/qhb-data/

  qcp:
    build: qcp
    image: qcp
```

```
        network_mode: "host"
        entrypoint: ["echo", "Service qcp disabled"]  # Закомментируйте
эту строку, если будете использовать QCP
        volumes:
          - ./qcp/config.yaml:/config.yaml:ro
```

12.3.10 Метрики пула соединений QCP

Ниже представлены метрики пула соединений QCP, характеризующие его работу. Сбор метрик проводится при операциях выполнения запроса.

Таблица 11. Метрики пула соединений QCP

Имя	**Описание**	**Тип**
qcp.queue	Количество запросов в очереди на данный момент	gauge
qcp.obtain_backend	Время ожидания назначения обслуживающего процесса для исполнения запроса клиента	timer
qcp.obtain_backend_failed	Превышено максимальное время ожидания назначения обслуживающего процесса для исполнения запроса клиента	timer
<адрес субд>.in_use	Количество используемых подключений (обслуживающих процессов) к СУБД	gauge

12.4 Сервер метрик

Сервер метрик используется в инфраструктуре СУБД «Квант-гибрид» для сбора, агрегации и пересылки метрик в различные системы:

- систему мониторинга *Graphite*;
- базу данных временных рядов *Prometheus*;
- агент по сбору метрик и журналов *Vector*;

– для сохранения данных метрик в CSV-файлах.

Список и особенности формирования метрик СУБД «Квант-гибрид» содержатся в разделе «Метрики СУБД “Квант-гибрид”» данного документа.

Сервер метрик должен быть установлен и настроен на каждой машине, где работают компоненты СУБД «Квант-гибрид» (сам сервер баз данных или QCP).

При необходимости агрегатор сервера метрик можно разместить на отдельном хосте. Подробную информацию см. в подразделе «Работа сервера метрик в режиме разделенного коллектора и агрегатора» настоящего документа.

Полную версию описания и настроек можно посмотреть в технической документации по СУБД «Квант-Гибрид» 1.5 на ресурсе разработчика.

12.4.1 Настройка сервера метрик

Пример файла конфигурации устанавливается по пути ***/etc/metricsd/config-example.yaml***.

Для работы сервера необходимо скопировать его в файл ***/etc/metricsd/config.yaml*** и скорректировать необходимые параметры. Особого внимания требует раздел **aggregation → backends**. Настройка сервера метрик под разных потребителей представлена в подразделах ниже.

Для автоматического запуска сервера при старте системы активируйте соответствующий сервис ***systemd***:

```
$ sudo systemctl enable --now metricsd.service
```

В случае потери связи между сервером метрик и СУБД «Квант-гибрид» (происходит в случае перезагрузки сервиса сервера метрик), в результате чего снижается количество поступающих метрик и появляются записи журнала «Failed to open a metrics sender» (не удалось открыть передатчик метрик), администратор для повторного подключения сервера метрик к базе должен выполнить либо перезапуск СУБД «Квант-гибрид1», либо вызов следующей функции в СУБД «Квант-гибрид»:

```
SELECT metrics_reset();
```

После выполнения команды необходимо убедиться, что количество поступающих метрик вернулось к изначальному объему (восстановление должно произойти в течении 1 минуты).

В конфигурации базы нужно изменить строчку:

```
#metrics_collector_path = 'путь/к/metrics`
```

на

```
metrics_collector_path = '@metrics-collector'
```

А затем перезапустить базу данных в следующем порядке:

– запустить ***metricsd***;

– запустить базу данных;

– запустить систему для приема метрик (*Prometheus, Graphite* и т.п).

12.4.1.1 Конфигурация сервера метрик для Graphite

Тема установки и настройки *Graphite* выходит за рамки данной документации. Пожалуйста, обратитесь к документации *Graphite* по адресу https://graphite.readthedocs.io/en/latest/.

```
# Конфигурация серверов. Должен быть сконфигурирован хотя бы
# один сервер.
backends:
  # Конфигурация сервера graphite
  - graphite:
    # Адрес терминала TCP Graphite для текстового протокола. Порт
по
    # умолчанию: 2003. Доступен только протокол TCP, поэтому если
    # Graphite не принимает подключение к этому порту, это вызовет
    # ошибку!
    address: "graphite:2003"
    # Префикс, добавляемый к имени каждой метрики. Необязательный
    # параметр; по умолчанию – пустая строка.
    prefix: ""
    # Время ожидания подключения. Необязательный параметр;
    # по умолчанию – 30 секунд.
    connection_timeout: "30 sec"
    # Время ожидания отправки данных. Необязательный параметр;
    # по умолчанию – 5 секунд.
    send_timeout: "5 sec"
```

Измените значение параметра **address** на реальный адрес сервера *Graphite* в вашей сети. Также рекомендуется изменить значение параметра **prefix** на, например, имя машины, на которой запущен сервер. Этот префикс будет добавляться ко всем генерируемым метрикам.

12.4.1.2 Конфигурация сервера метрик для формирования CSV-файлов

```
# Конфигурация серверов. Должен быть сконфигурирован хотя бы
# один сервер.
backends:
  # Конфигурация CSV-сервера (в случае необходимости копирования
  # данных о метриках в CSV-файлы)
  - csv:
    # Каталог, содержащий CSV-файлы.
    directory: "/var/lib/qhb/csv_create"
    # Системный идентификатор экземпляра СУБД.
    qhb_instance: "instance_name"
    # Интервал переключения CSV-файлов (необязательно).
    rotation_age: "1 h"
    # Настройка времени следующего переключения после запуска
    # (необязательно).
    rotation_offset: 2022-03-01T18:00:00
```

Параметры конфигурации сервера CSV необходимы в случае, если нужно сохранять данные метрик в CSV-файлы. Используются следующие параметры:

- **directory** указывает место расположения CSV-файлов.
- **qhb_instance** – системный идентификатор экземпляра; не может быть пустым и содержать кавычки, в чем-то аналогичен параметру **prefix**, но его значение при загрузке данных метрик заполняет отдельный столбец в таблице метрик, не связанный с именем метрик. Может быть полезен при наличии нескольких экземпляров СУБД на одном сервере.
- **rotation_age** – период переключения CSV-файлов. Единицы измерения можно задавать, например, в виде секунд (*s*), минут (*m*), часов (*h*), дней (*d*).
- **rotation_offset** – этот параметр задает время переключения на очередной CSV-файл после запуска сервера метрик. Время указывается для часового

пояса UTC. Параметр удобно использовать для выравнивания начала очередного периода, например, по началу следующего часа.

12.4.1.3 Конфигурация сервера метрик для Vector

Тема установки и настройки сборщика журналов *Vector* выходит за рамки данной документации. Пожалуйста, обратитесь к документации *Vector* по адресу https://vector.dev/docs/reference/configuration/sources/http_server/.

Пример настройки конфигурации источника данных *Vector* для сервера метрик СУБД «Квант-гибрид»:

```
sources:
  qhb_metricsd:
    type: "http_server"
    address: "metricsd:80"
    acknowledgements: false
    encoding: "ndjson"
```

Настройка параметров сервера метрик:

```
# Конфигурация серверов. Должен быть сконфигурирован хотя бы
# один сервер.
backends:
  # Конфигурация сервера Vector
  - vector.dev:
      # URL-адрес слушателя vector типа "http_server", настроенного
      # на получение данных в формате "ndjson".
      url: "metricsd:80"
```

Исправьте в параметрах **address** и **url** значение *metricsd* на реальный адрес сервера метрик в вашей сети.

12.4.1.4 Конфигурация сервера метрик для Prometheus

Тема установки и настройки *Prometheus* выходит за рамки данной документации. Пожалуйста, обратитесь к документации *Prometheus* по адресу https://prometheus.io/docs/introduction/overview/.

Пример изменений в конфигурации *Prometheus* для сервера метрик СУБД «Квант-гибрид»:

```
    scrape_configs:
      # Имя операции (job_name) добавляется в виде метки `job=< job_name
>`
      # во все временные ряды, извлекаемые из этой конфигурации.
      - job_name: 'metricsd'
         # metrics_path по умолчанию принимает значение '/metrics'
         # scheme по умолчанию принимает значение 'http'.
      static_configs:
        - targets: ['metricsd:8080']
```

Настройка параметров сервера метрик:

```
    # Конфигурация серверов. Должен быть сконфигурирован хотя бы
    # один сервер.
    backends:
      # Конфигурация сервера Prometheus:
      - prometheus:
          # Адрес, который будет выставлен для сканирования сервером
          # Prometheus.
          listening_address: "metricsd:8080"
```

Измените в параметрах **targets** и **listening_address** значение *metricsd* на реальный адрес сервера метрик в вашей сети.

Важно: при отправке метрик в *Prometheus* все точки, присутствующие в названии метрики, сервером метрик заменяются на «_».

12.4.2 Работа сервера метрик в режиме разделенного коллектора и агрегатора

Сервер метрик состоит из двух компонентов:

- **Коллектор**. Эта часть отвечает за непосредственный обмен метриками между приложениями (например, СУБД «Квант-гибрид») и сервером метрик. Отправка метрик осуществляется через разделяемую память, которой управляет сервер метрик. Данный компонент не занимается обработкой метрик, лишь их сбором из множества источников. Далее метрики передаются во второй компонент – **агрегатор**.
- **Агрегатор**. Как следует из названия, этот компонент агрегирует, то есть накапливает метрики и делает статистические вычисления: усреднение,

вычисление перцентилей, максимальных значений и т. д. Также этот компонент занимается отправкой агрегированных метрик дальше, например, в *Graphite*, или записывает их в CSV-файл.

По умолчанию сервер метрик запускается в режиме, который объединяет в себе и **коллектор**, и **агрегатор**. Такая архитектура позволяет минимизировать затраты на передачу данных, поскольку она осуществляется через оперативную память.

Можно запускать сервер метрик в режиме, когда активен только один из компонентов, причем они могут быть разнесены на разные физические устройства. В таком случае собранные метрики будут отправляться в **агрегатор**, используя протокол UDP. Обратите внимание, что UDP не гарантирует доставку пакетов. Этот вопрос выходит за границы данного руководства.

Важно: Сервер метрик в режиме **коллектора** должен работать на том же хосте, что и исследуемое приложение, так как обмен метриками осуществляется через разделяемую память. Техническая возможность вынести **коллектор** на отдельный хост отсутствует.

12.4.2.1 Возможные причины для переноса агрегатора сервера метрик на отдельный хост

Разделение **коллектора** и **агрегатора** не является необходимым, но в ряде случаев их разделение может быть полезно, а именно:

- Конечный получатель агрегированных метрик (например, *Graphite*) может быть недоступен с хоста, где запускается исследуемое приложение.
- Исследуемое приложение разнесено на разные хосты, и есть желание агрегировать метрики по всем хостам «насквозь», не учитывая этого различия.
- Для снижения нагрузки на хост, где запущен сервер метрик. Однако следует отметить, что агрегация метрик не отнимает много ресурсов, поэтому такая оптимизация сомнительна.

Данный перечень не является исчерпывающим, но призван описать возможные применения.

12.4.2.2 Настройка агрегатора сервера метрик на отдельном хосте

Запуск сервера метрик в режиме **коллектора** осуществляется одним из следующих способов, от более предпочтительных к менее:

1) Удалите или закомментируйте **целиком** раздел **aggregation:** в конфигурационном файле ***/etc/metricsd/config.yaml***.
2) `/usr/bin/metricsd --only-collector –config /etc/metricsd/config.yaml`, где ***/etc/metricsd/config.yaml*** – путь к конфигурационному файлу.
3) С помощью дополнительного файла ***/etc/systemd/system/metricsd.service.d/override.conf*** (вместо ***override.conf*** можно выбрать любое другое имя, заканчивающееся на *.conf*) со следующим содержимым:

```
[Service]
ExecStart=
ExecStart=/usr/bin/metricsd --only-collector --config /etc/metricsd/config.yaml
```

4) Посредством изменения файла ***metricsd.service*** и добавлением флага **--only-collector** к вызову ***metricsd*** в поле *ExecStart=*. Данный способ не рекомендуется к применению. Используйте его, только если точно знаете, что делаете, на свой страх и риск.

Коллектор в данном случае является просто каналом для передачи метрик на другой хост к **агрегатору**. **Агрегатор** настраивается на подключение к соответствующему серверу (*Prometheus*, *Graphite* и др.).

Сервер метрик, работающий в режиме агрегатора, устанавливается на отдельном сервере.

Запуск сервера метрик в режиме **агрегатора** осуществляется одним из следующих способов, от более предпочтительных к менее:

1) Удалите или закомментируйте **целиком** раздел **collection:** в конфигурационном файле ***/etc/metricsd/config.yaml***.
2) `/usr/bin/metricsd --only-aggregator --config /etc/metricsd/config.yaml`, где ***/etc/metricsd/config.yaml*** – путь к конфигурационному файлу.
3) С помощью дополнительного файла ***/etc/systemd/system/metricsd.service.d/override.conf*** (вместо **override.conf** можно выбрать любое другое имя, заканчивающееся на *.conf*) со следующим содержимым:

```
[Service]
ExecStart=
ExecStart=/usr/bin/metricsd --only-aggregator --config /etc/metricsd/config.yaml
```

4) Посредством изменения файла ***metricsd.service*** и добавлением флага **--only-aggregator** к вызову ***metricsd*** в поле *ExecStart=*. Данный способ не рекомендуется к применению.

Важно! Убедитесь в том, что:

– в разделе «aggregation» конфигурации сервера метрик, работающего в режиме **агрегатора**, указан такой адрес **listen_address**, получение данных на который будет доступно с сервера метрик, работающего в режиме **коллектора**;
– в секции «collection» конфигурации сервера метрик, работающего в режиме **коллектора**, указан верный адрес агрегатора **aggregator_address**, соответствующий адресу из предыдущего пункта;
– во избежание фрагментирования пакетов UDP, в разделе «collection» конфигурации сервера метрик, работающего в режиме **коллектора**, указан подходящий размер допустимого пакета **udp_payload**. Обратитесь к вашему системному/сетевому администратору для уточнения подходящего размера;
– ваш сетевой администратор не запретил передачу пакетов UDP по сети.

Детали конфигурации сети выходят за рамки данного руководства. При возникновении вопроса обратитесь к вашему администратору сети.

12.4.2.2.1 Пример файла конфигурации коллектора

```
# Уровень детализации на сервере.
verbosity: "warn"

# Конфигурация сбора метрик.
#
# Если вы не хотите запускать сбор метрик, этот раздел можно
# пропустить.
collection:
  # Адрес и порт (UDP) агрегатора метрик.
  # Необязательны, если агрегация метрик выполняется на том же
  # экземпляре сервера; в противном случае необходимы.
  aggregator_address: "127.0.0.1:5400"

  # Unix-адрес квитирующего сервера коллектора.
  # Символ `@` в начале означает абстрактный сокет (т. е. в файловой
  # системе никакого сокета не создается).
  bind_addr: "@metrics-collector"

  # Путь, по которому создаются запросы spsc.
  queue_path: "/metrics-collector-queues"

  # Вместимость приемника запросов.
  # Необязательно. Значение по умолчанию – 1024.
  queue_capacity: 1024

  # Количество обработанных элементов, после которого обновляются
  # разделяемые счетчики.
  # Необязательно. Значение по умолчанию – до 10.
  batch_size: 10

  # Количество обрабатываемых потоков.
  # Необязательно. Значение по умолчанию – до 2.
```

```
  threads: 2

  # Объем полезных данных UDP.
  # Необязательно. Значение по умолчанию – 1024.
  udp_payload: 1024

  # Максимальный интервал между передачами.
  # Необязательно. Значение по умолчанию – "1s" (1 с).
  send_interval: "1s"

  # Количество потоков рабочих процессов.
  # Необязательно. Значение по умолчанию – 2.
  workers: 2
```

12.4.2.2.2 Пример файла конфигурации агрегатора

```
# Уровень детализации на сервере.
verbosity: "info"

# Конфигурация агрегации метрик.
#
# Если вы не хотите запускать агрегацию метрик, этот раздел можно
# пропустить.
aggregation:
  # Адрес прослушивания агрегатора
  # Необязателен, если сбор метрик выполняется на том же экземпляре
  # сервера; в противном случае необходим.
  listen_address: "0.0.0.0:5400"

  # Интервал между каждой агрегацией
  send_interval: 10s

  # Список используемых перцентилей для величин синхронизации.
  # Значения по умолчанию – [50, 90, 95, 99, 999].
  # Значения больше 1000 не поддерживаются.
  percentiles_list: [50, 90, 95, 99, 999]
```

```
      # Срок существования «индикаторных» (gauge) значений. Необязательно;
      # по умолчанию 5 минут.
      gauge_lifetime: 5min

      # Конфигурация серверов. Должен быть сконфигурирован как минимум
      # один сервер.
      backends:
        # Конфигурация сервера graphite
        - graphite:
          # Адрес конечной точки TCP Graphite для текстового протокола.
          # Порт по умолчанию 2003.
          # Доступен только протокол TCP, поэтому если Graphite не ожидает
          # передачи данных на этом порту, вы получите ошибку!
          address: "Address:2003"
          # Префикс, стоящий в начале имени всех метрик. Необязательно;
          # по умолчанию это пустая строка.
          prefix: "Address"
          # Максимальное время ожидания подключения. Необязательно;
          # по умолчанию 30 секунд.
          connection_timeout: "30 sec"
          # Максимальное время ожидания передачи данных. Необязательно;
          # по умолчанию 5 секунд.
          send_timeout: "5 sec"
```

12.4.3 Метрики СУБД «Квант-гибрид»

12.4.3.1 Общие сведения

В настоящее время метрики в СУБД «Квант-гибрид» собираются на уровне всего кластера баз данных и относятся ко всему набору процессов СУБД «Квант-гибрид» независимо от того, являются ли они фоновыми или поддерживающими пользовательские подключения. Часть метрик собирается на уровне баз данных и на уровне бэкендов. Данные метрик отсылаются на **коллектор** и **агрегатор** метрик ***metricsd***, краткое описание которого можно найти в подразделе «Сервер метрик»

выше. Далее метрики агрегируются и записываются в *Graphite*, в качестве интерфейса к которому выступает Grafana.

12.4.3.2 Типы метрик

- **Gauge** – неотрицательное целое значение. Может быть установлено, увеличено или уменьшено на заданное число.
- **Counter** – неотрицательное целое значение. Может быть увеличено на заданное число.
- **Timer** – неотрицательное целое значение, длительность в наносекундах. Значение может быть только записано; во время агрегации вычисляется ряд статистических характеристик:
  - *sum* – сумма значений
  - *count* – количество записанных значений
  - *max* – максимальное из записанных значений
  - *min* – минимальное из записанных значений
  - *mean* – среднее арифметическое
  - *median* – медиана
  - *std* – стандартное квадратичное отклонение
  - при необходимости список перцентилей

12.4.3.3 Группы метрик

12.4.3.3.1 sys

К группе *sys* относятся различные системные метрики.

Сбор системных метрик регулируется двумя глобальными константами:

- **qhb_os_monitoring** – логическая константа, по умолчанию значение *off* (выключена). При значении *on* выполняется периодический сбор метрик.
- **qhb_os_stat_period** – период сбора системной статистики в секундах, по умолчанию 30 секунд.

Таблица 12. Метрики и загрузки ЦП

Имя	Описание	Тип
sys.load_average.1min	*load average* за последнюю минуту	gauge
sys.load_average.5min	*load average* за последние 5 минут	gauge
sys.load_average.15min	*load average* за последние 15 минут	gauge
sys.cpu.1min	Процент загрузки процессоров за последнюю минуту	gauge
sys.cpu.5min	Процент загрузки процессоров за последнюю минуту за последние 5 минут	gauge
sys.cpu.15min	Процент загрузки процессоров за последнюю минуту за последние 15 минут	gauge

Примечание. Значения метрик *load average* содержат значения с точностью до сотых, однако при передаче данных в силу технических особенностей исходные значения умножаются на 100 и передаются как целые числа. Поэтому при выводе данных следует учитывать эту особенность и делить значения на 100.

load average – усредненное количество исполняемых и ожидающих потоков за заданный интервал времени (1, 5 и 15 минут). Обычно соответствующие значения выводятся с помощью команд `uptime`, `top` или `cat /proc/loadavg`.

Если значение этого показателя за последнюю минуту больше, чем за последние 5 и 15 минут, нагрузка растет, если меньше – падает. Однако этот показатель важен не сам по себе, а относительно общего числа процессоров. Дополнительные и производные от *load average* метрики по загрузке процессоров *sys.cpu.Nmin* показывают приблизительный процент загрузки процессоров с учетом их количества и рассчитываются по следующей упрощенной формуле:

```
sys.cpu.<N>min = sys.load_average.<N>min / cpu_count * 100
```

где **cpu_count** – количество процессоров в системе, а ***N*** принимает значения 1, 5 или 15.

Количество процессоров рассчитывается как произведение количества физических сокетов, ядер на сокет и нитей на ядро. Команда `lscpu` выводит все необходимые данные. Например:

```
Thread(s) per core:              2
Core(s) per socket:              4
Socket(s):                       1
```

В данном случае *cpu_count* = 2 * 4 * 1 = 8.

Альтернативным и более простым методом может быть получение этого значения через команду `nproc`.

Таким образом, в данном случае загрузка в 100% будет достигнута, если величина *load average* будет стремиться к 8. Однако эти расчеты и значения будут иметь довольно приблизительный и даже условный характер, что показывает приведенная выше по ссылке статья.

Таблица 13. Метрики использования памяти

Имя	**Описание**	**Тип**
sys.mem.total	Общий размер установленной памяти RAM	gauge
sys.mem.used	Используемая память	gauge
sys.mem.free	Неиспользуемая память	gauge
sys.mem.available	Память, доступная для запускаемых приложений (не включая swap, но учитывая потенциально освобождаемую память, занимаемую страничным кэшем)	gauge
sys.swap.total	Общий размер файла подкачки	gauge

Имя	Описание	Тип
sys.swap.free	Неиспользуемая память файла подкачки	gauge

Значения метрик соответствуют полям из вывода утилиты ***free*** (значения отображаются в килобайтах, т. е. соответствуют выводу `free -k`).

Таблица 14. Соответствие значений метрик полям вывода утилиты free

Метрика	Поле утилиты *free*
sys.mem.total	*Mem:total*
sys.mem.used	*Mem:used*
sys.mem.free	*Mem:free*
sys.mem.available	*Mem:available*
sys.swap.total	*Swap:total*
sys.swap.free	*Swap:free*

Величину, соответствующую выводимому в утилите ***free*** значению *Mem:buff/cache*, можно рассчитать по формуле:

```
Mem:buff/cache = Mem:total - Mem:used - Mem:free
```

Таким образом, в Grafana можно, используя функцию *diffSeries*, рассчитывать и выводить это значение на основании других имеющихся данных.

Значение *Mem:shared* (данные виртуальной файловой системы tmpfs) через метрики не выводится.

Значение *Swap:used* можно рассчитать по формуле:

```
Swap:used = Swap:total - Swap:free
```

Это значение также можно выводить в Grafana в виде рассчитываемой величины через функцию *diffSeries*.

Более подробное описание этих показателей можно получить через справочную систему операционной системы для утилиты ***free*** (для этого вызовите *man free*).

Таблица 15. Метрики использования дискового пространства

Имя	**Описание**	**Тип**
sys.disk_space.total	Объем дисковой системы, на которой находится каталог с данными (в байтах)	gauge
sys.disk_space.free	Свободное пространство в дисковой системе, на которой находится каталог с данными (в байтах)	gauge

Метрики относятся к дисковой системе, на которой расположен каталог с файлами базы данных. Этот каталог определяется параметром командной строки **-D** при запуске базы данных либо переменной среды **$PGDATA**. Параметр **data_directory** в файле конфигурации ***qhb.conf*** может переопределять расположение каталога с данными.

Таблица 16. Другие системные метрики

Имя	**Описание**	**Тип**
sys.processes	Общее количество запущенных в системе процессов	gauge
sys.uptime	Количество секунд, прошедших с начала запуска системы	gauge

12.4.3.3.2 db_stat

К группе *db_stat* относятся метрики чтения и записи блоков на уровне экземпляра СУБД «Квант-гибрид».

Таблица 17. Метрики чтения и записи блоков

Имя	Описание	Тип
qhb.db_stat.numbackends	Количество активных серверных процессов	gauge
qhb.db_stat.blocks_fetched	Количество блоков, полученных при чтении	counter
qhb.db_stat.blocks_hit	Количество блоков, найденных в кэше при чтении	counter
qhb.db_stat.blocks_read_time	Время чтения блоков, в миллисекундах	counter
qhb.db_stat.blocks_write_time	Время записи блоков, в миллисекундах	counter
qhb.db_stat.conflicts.tablespace	Количество запросов, отмененных из-за удаленных табличных пространств	counter
qhb.db_stat.conflicts.lock	Количество запросов, отмененных из-за тайм-аутов блокировок	counter
qhb.db_stat.checksum_failures	Количество несовпадений контрольной суммы страницы данных	counter

Имя	Описание	Тип
qhb.db_stat.conflicts.snapshot	Количество запросов, отмененных из-за устаревших снимков состояния	counter
qhb.db_stat.conflicts.bufferpin	Количество запросов, отмененных из-за закрепленных буферов	counter
qhb.db_stat.conflicts.startup_deadlock	Количество запросов, отмененных из-за взаимных блокировок	counter
qhb.db_stat.tuples.returned	Количество строк, полученных при последовательном сканировании	counter
qhb.db_stat.tuples.fetched	Количество строк, полученных при сканировании по индексу	counter
qhb.db_stat.tuples.inserted	Количество строк, добавленных в базу данных	counter
qhb.db_stat.tuples.updated	Количество строк, измененных в базе данных	counter

Имя	Описание	Тип
qhb.db_stat.tuples.deleted	Количество строк, удаленных из базы данных	counter

Также генерируются версии этих метрик по базам данных и отдельным бэкендам. В имена метрик включается идентификатор базы данных, например: *qhb.db_stat.db*_16384.numbackends, или (для некоторых метрик) номер процесса, например: *qhb.db_stat.34567.blocks_fetched*. Данная группа метрик может генерировать большие объемы данных, поэтому администратор должен решить, действительно ли необходимо оставлять эту группу метрик включенной.

На основании метрик *qhb.db_stat.blocks_fetched* и *qhb.db_stat.blocks_hit* рассчитывается коэффициент попадания в кэш:

```
k = blocks_hit / blocks_fetched * 100%
```

Хорошим уровнем обычно считается значение более 90%. Если значение коэффициента существенно ниже этой отметки, желательно рассмотреть возможность увеличения объема буферного кэша.

12.4.3.3.3 bgwr

К группе *bgwr* относятся метрики фонового процесса записи.

Таблица 18. Метрики фонового процесса записи

Имя	Описание	Тип
qhb.bgwr.checkpoints_timed	Количество запланированных контрольных точек, которые были выполнены	counter

Имя	Описание	Тип
qhb.bgwr.checkpoints_req	Количество запрошенных контрольных точек, выполненных вне очереди запланированных	counter
qhb.bgwr.checkpoint_write_time	Время, потраченное на этап обработки контрольной точки, где файлы записываются на диск, в миллисекундах	counter
qhb.bgwr.checkpoint_sync_time	Время, потраченное на этап обработки контрольной точки, где файлы синхронизируются с диском, в миллисекундах	counter
qhb.bgwr.buffers_checkpoint	Количество буферов, записанных во время контрольных точек	counter
qhb.bgwr.buffers_clean	Количество буферов, записанных фоновым процессом записи	counter
qhb.bgwr.maxwritten_clean	Сколько раз фоновый процесс записи останавливал сброс грязных страниц, поскольку записывал слишком много буферов	counter

Имя	Описание	Тип
qhb.bgwr.buffers_backend	Количество буферов, записанных непосредственно обслуживающим процессом	counter
qhb.bgwr.buffers_backend_fsync	Сколько раз обслуживающий процесс вызывал *fsync* сам (обычно их обрабатывает фоновый процесс записи, даже когда обслуживающий процесс выполняет запись самостоятельно)	counter
qhb.bgwr.buffers_alloc	Количество выделенных буферов	counter

Данные по перечисленным метрикам отображаются в разделе «Контрольные точки и операции с буферами» информационной панели СУБД «Квант-гибрид». Обычно при выполнении запланированных контрольных точек сначала происходит запись информации о начале контрольной точки, затем в течение некоторого времени идет сброс блоков на диск и по окончании контрольной точки фиксируется информация о продолжительности записи и синхронизации данных. В случае обработки запланированной контрольной точки запись блоков равномерно распределяется во времени согласно параметрам настройки, чтобы снизить влияние этого процесса на общий ввод/вывод. При контрольных точках, запрошенных через команду, сброс блоков происходит сразу, без искусственной задержки.

Также генерируются версии этих метрик по базам данных. В имена метрик включается идентификатор базы данных, например: *qhb.db_stat.db_16384.numbackends*.

12.4.3.3.4 mem_stat

К группе *mem_stat* относятся метрики для отслеживания размера памяти *work area*.

Память *work area* может быть приблизительно описана как память, используемая внутренними операциями сортировки и хэш-таблицами. Размер этой памяти контролируется глобальным параметром конфигурации **work_mem**: при превышении лимита вместо оперативной памяти начинают использоваться временные файлы на диске. Параметр **work_mem** ограничивает память, используемую единичной операцией.

В группу *mem_stat* входит три разновидности метрик с общим именем *work_area*:

- *qhb.mem_stat.work_area* – показывает общий размер *work area* в байтах всего кластера
- *qhb.mem_stat.db_<номер>.work_area* – показывает размер *work area* в байтах для одной базы данных
- *qhb.mem_stat.<pid>.work_area* – показывает размер *work area* в байтах для одного рабочего процесса

Метрики второй и третьей разновидностей создаются динамически.

Все метрики типа *gauge*, то есть их значения могут уменьшаться по мере освобождения ранее занятой памяти.

12.4.3.3.5 temp_stat

К группе *temp_stat* относятся метрики по временным файлам и таблицам СУБД «Квант-гибрид».

Таблица 19. Метрики по временным файлам и таблицам

Имя	**Описание**	**Тип**
qhb.temp_stat.temp_count	Количество временных файлов, созданных за период агрегации	counter

Имя	Описание	Тип
qhb.temp_stat.temp_bytes	Общий объем временных файлов, созданных за период агрегации, в байтах	counter
qhb.temp_stat.temp_tables	Количество временных таблиц, созданных за период агрегации	counter
qhb.temp_stat.temp_table_bytes	Общий объем временных таблиц, созданных за период агрегации, в байтах	counter

Данные собираются за период агрегации коллектора метрик. Обновление данных происходит по окончании соответствующей операции, например, при создании временной таблицы с указанием **DROP ON COMMIT** обновление статистики произойдет после выполнения команды `COMMIT` при удалении временной таблицы. В некоторых случаях объем, занимаемый временными файлами и таблицами, может быть весьма существенным.

Также генерируются версии этих метрик по базам данных. В имена метрик включается идентификатор базы данных, например: *qhb.db_stat.db_16384.numbackends*.

12.4.3.3.6 wal

К группе *wal* относятся метрики процесса архивации журнала упреждающей записи.

Таблица 20. Метрики процесса архивации WAL

Имя	Описание	Тип
qhb.wal.archived	Количество успешно выполненных операций архивации файлов журнала упреждающей записи	counter

Имя	Описание	Тип
qhb.wal.failed	Количество попыток архивации, завершившихся сбоем	counter
qhb.wal.archive_time	Время, потраченное на копирование файлов журналов, в наносекундах	counter

Эти метрики работают в том случае, если настроена архивация журнала упреждающей записи. Для этого необходимо установить параметр **archive_mode** в значение *on* (включен) и определить команду архивации в параметре **archive_command**.

12.4.3.3.7 transaction

К группе *transaction* относятся метрики по транзакциям.

Таблица 21. Метрики по транзакциям

Имя	Описание	Тип
qhb.transaction.commit	Количество фиксаций транзакций	counter
qhb.transaction.rollback	Количество откатов транзакций	counter
qhb.transaction.deadlocks	Количество взаимоблокировок	counter

Данные метрик собираются непосредственно при выполнении команд завершения и отмены транзакций на уровне всего кластера баз данных, на уровне отдельных баз данных и на уровне бэкендов.

Коэффициент подтверждения транзакций рассчитывается как процентное отношение фиксаций транзакций к сумме фиксаций и откатов транзакций:

```
k = commit/(commit + rollback)*100%
```

Обычно значение стремится к 100%, поскольку чаще всего транзакции завершаются успешно. Существенная доля откатившихся транзакций может говорить о том, что в системе существуют проблемы.

Взаимоблокировки возникают, когда различные сеансы взаимно ожидают освобождения заблокированных данных. В этом случае после автоматического определения взаимоблокировки происходит откат одной из транзакций.

12.4.3.3.8 wait

К группе *wait* относятся метрики событий ожидания.

Данный набор метрик полностью соответствует набору стандартных событий ожидания. Метрики имеют префикс *qhb.wait*. Далее в наименовании идет класс события ожидания и через точку название события ожидания. В текущем релизе имена метрик ограничены в размере и имеют в наименовании максимум 31 символ. Все метрики по событиям ожидания имеют тип *counter*, однако значение отражает промежуток времени в микросекундах, который эти ожидания заняли в совокупности во всех сеансах за период сбора.

Примечание: если в течение периода наблюдения работало множество пользовательских подключений, которые находились в состоянии ожидания, суммарное значение времени ожидания может многократно превысить этот период. Например, если в течение 10 секунд 1000 сеансов провели в ожиданиях по одной секунде, суммарное время ожидания составит 1000 секунд.

Метрики с типами событий ожидания *Lock*, *LWLock*, *IO* и *IPC* отображаются в разделе «События ожидания» информационной панели СУБД «Квант-гибрид». Значения метрик выводятся в микросекундах (автоматически переводясь в другие единицы при увеличении значений). На существующих графиках выводятся не все события ожиданий, а только по пять самых значимых по величине на каждом графике. Разные события ожиданий могут иметь сильно отличающиеся по величине продолжительности. Значительные колебания значений могут отражать возникающие проблемы.

Таблица 22. Наиболее значимые события ожидания

Имя события ожидания	**Описание**
Lock.extend	Ожидание расширения отношения. Становится заметным при активном росте таблиц. Необходимость выделения новых блоков приводит к некоторым задержкам, которые отражаются в этой метрике.
Lock.transactionid	Ожидание завершения транзакции. Событие ожидания возникает в том случае, если транзакция вынуждена ждать окончания обработки предыдущих транзакций, которые также получают подтверждение своего окончания.
Lock.tuple	Ожидание получения блокировки для кортежа. Возникает в случае одновременной работы с теми же данными нескольких транзакций.
LWLock.WALWriteLock	Ожидание записи буферов WAL на диск. Часто является лидером среди событий ожиданий этого типа, так как операции с диском являются наиболее медленными в этой группе событий ожидания.
LWLock.wal_insert	Ожидание вставки WAL в буфер памяти.
LWLock.buffer_content	Ожидание чтения или записи страницы данных в памяти. Возникает и становится существенным при интенсивном вводе/выводе.
LWLock.buffer_mapping	Ожидание связывания блока данных с буфером в буферном пуле.

Имя события ожидания	Описание
LWLock.lock_manager	Ожидание добавления или проверки блокировок для обслуживающих процессов. Событие становится значимым при частых транзакциях.

12.4.3.3.9 bufmgr

К группе *bufmgr* относятся метрики буферного менеджера, касающиеся механизмов управления памятью. Сбор метрик проводится при операциях чтения данных.

Таблица 23. Метрики буферного менеджера

Имя	Описание	Тип	Агрегат
qhb.bufmgr.BufferAlloc	Сколько раз проводился поиск буфера	timer	count
qhb.bufmgr.BufferAlloc	Общее время, потраченное на поиск буфера	timer	sum
qhb.bufmgr.happy_path	Сколько раз буфер нашелся сразу	timer	count
qhb.bufmgr.happy_path	Общее время, потраченное на поиск, когда буфер нашелся сразу	timer	sum

Имя	Описание	Тип	Агрегат
qhb.bufmgr.cache_miss	Количество промахов кэша буферов	timer	count
qhb.bufmgr.cache_miss	Общее время, потраченное на обработку промахов кэша буферов	timer	sum
qhb.bufmgr.disk_read	Количество чтений страницы с диска (асинхронно)	timer	count
qhb.bufmgr.flush_dirty	Количество выгрузок страницы на диск (асинхронно)	timer	count
qhb.bufmgr.retry_counter	Количество повторных обработок промаха	counter	
qhb.bufmgr.strategy_pop_cnt	Сколько раз срабатывала специальная стратегия получения или	counter	

Имя	Описание	Тип	Агрегат
	вытеснения буфера		
qhb.bufmgr.strategy_reject_cnt	Количество забракованных буферов, предложенных специальной стратегией	counter	
tarq_cache.allocate	Сколько раз проводился поиск в TARQ	timer	count
tarq_cache.allocate	Общее время, потраченное на поиск в TARQ	timer	sum
tarq_cache.allocate_new	Сколько раз выбирался исключаемый блок в TARQ	timer	count
tarq_cache.rollback	Количество откатов вытеснения в TARQ	timer	count
tarq_cache.rollback	Общее время, потраченное на откаты	timer	sum

Имя	Описание	Тип	Агрегат
	вытеснения в TARQ		
tarq_cache.touch	Общее время, потраченное на учет популярных страниц в TARQ	timer	sum

12.4.3.3.10 queryid

К группе *queryid* относятся метрики, предоставляющие возможность отслеживать статистику планирования и выполнения сервером всех (почти) операторов SQL.

Эта группа отличается тем, что входящие в нее метрики создаются динамически при начале обработки движком нового оператора. В имена новых метрик добавляется уникальное имя или ID запроса. Например, метрика *qhb.queryid.c92e145f160e7b9e.exec_calls* отражает количество исполнений некоторого оператора SQL. Текст самого оператора можно получить из новой системной таблицы ***qhb_queryid*** запросом вида `SELECT * FROM qhb_queryid WHERE qid = 'c92e145f160e7b9e'`.

Подробнее настройка метрик группы *queryid* описана в технической документации по СУБД «Квант-Гибрид» на ресурсе разработчика.

12.4.3.3.11 Метрики пула соединений QCP

Ниже представлены метрики пула соединений, характеризующие работу пула. Сбор метрик проводится при операциях выполнения запроса.

Таблица 24. Метрики пула соединений QCP

Имя	Описание	Тип
qcp.queue	Количество запросов в очереди на данный момент	gauge

Имя	Описание	Тип
qcp.obtain_backend	Время ожидания назначения обслуживающего процесса для исполнения запроса клиента	timer
qcp.obtain_backend_failed	Превышено максимальное время ожидания назначения обслуживающего процесса для исполнения запроса клиента	timer
<адрес субд>.in_use	Количество используемых подключений (обслуживающих процессов) к СУБД	gauge

12.4.4 Включение и выключение записи значений для групп метрик

В СУБД «Квант-гибрид» имеется механизм, который позволяет включать и выключать отправку групп метрик. В каталоге данных расположен файл конфигурации ***metrics_config.toml***, в котором каждая строка соответствует группе метрик:

```
[is_enabled]
default = true
sys = true
db_stat = true
bgwr = true
wal = true
transaction = true
wait = true
bufmgr = true
mem_stat = true
temp_stat = true
queryid = false
```

Переменные принимают значения *true* (отправка метрик группы разрешена) или *false* (отправка запрещена).

Таблица 25. Настройка записи значений для групп метрик

Группа метрик	**Описание**	**Значение по умолчанию**
default	Группа по умолчанию (различные метрики, не выделенные в отдельные группы)	true
sys	Метрики операционной системы	true
db_stat	Метрики чтения и записи блоков	true
bgwr	Метрики фонового процесса записи	true
mem_stat	Метрики размера памяти «work area»	true
temp_stat	Метрики по данным временных файлов и таблиц	true
wal	Метрики архивации файлов WAL	true
transaction	Метрики транзакций	true
wait	Метрики событий ожидания	true
bufmgr	Метрики механизмов управления памятью	true
queryid	Метрики планирования и выполнения сервером операторов SQL	false

Примечание: для сбора метрик операционной системы необходимо также установить в параметре **qhb_os_monitoring** значение *on* (включен).

Для просмотра списка групп метрик и их текущего состояния можно воспользоваться SQL-функцией *metrics_config_show*. Пример вывода этой функции:

```
SELECT * FROM metrics_config_show();
 group_name  | enabled
-------------+---------
 bgwr        | t
```

```
 bufmgr      | t
 db_stat     | t
 default     | t
 mem_stat    | t
 queryid     | f
 sys         | t
 temp_stat   | t
 transaction | t
 wait        | t
 wal         | t
(11 rows)
```

Помимо задания значений настроек через файл параметров, имеется возможность менять эти значения через функции. Изменения сразу же отображаются в файле конфигурации.

С помощью параметра **respect_transient_settings** в функциях *metrics_config_enable_group* (включает отправку значений для метрик заданной группы) и *metrics_config_disable_group* (выключает отправку значений для метрик заданной группы) можно указать, отдавать ли приоритет параметрам отправки групп метрик, установленным на уровне отдельных обслуживающих процессов.

С помощью функции *metrics_config_transient_set* можно включить или выключить отправку значений заданной группы метрик для конкретного обслуживающего процесса. Настройка будет действовать в течение всего времени жизни обслуживающего процесса либо до вызова функции *metrics_config_enable_group* или *metrics_config_disable_group* со вторым параметром **respect_transient_settings = false**, который в данном случае распространит действие команды также на те обслуживающие процессы, в которых устанавливалось собственное значение параметров.

При запуске обслуживающего процесса с идентификатором, использовавшимся ранее, для данного идентификатора обнуляются все временные настройки.

Примечание: при использовании SQL-функций для управления метриками и аннотациями метрики записываются всегда.

12.4.5 Информационные панели метрик СУБД «Квант-гибрид» для Grafana

Информационные панели (dashboards) СУБД «Квант-гибрид» для Grafana расположены в репозитории.

СУБД «Квант-гибрид» поставляется совместно с сервером метрик, который записывает данные метрик в *Graphite*. Интерфейсом к этим метрикам служит Grafana. Текущий набор информационных панелей для Grafana поставляется в качестве самодокументируемых образцов, на основе которых пользователи при необходимости могут самостоятельно создать панели, более соответствующие их потребностям. Поставляемые панели можно использовать и в исходном виде.

12.4.6 Импорт информационных панелей

Экспорт JSON-описания информационных панелей выполнен в Grafana 6.7.2.

Перед импортом JSON-описания необходимо решить, будут ли названия метрик содержать в качестве префикса имя хоста. Именно таким образом устроены наименования метрик внутри панелей, и этот вариант рекомендуется оставить. В начале имен метрик добавлена переменная **$server_name**, по умолчанию для нее выбрано значение *your_host_name*. Перед импортом можно заменить в JSON-файлах это значение на наименование одного из хостов. В дальнейшем в этой переменной через интерфейс Grafana можно будет добавить через запятую все имена хостов, с которых будут собираться метрики. Это позволит быстро переключаться при просмотре метрик с одного хоста на другой. Если такая схема использоваться не будет (в случае, если будут просматриваться метрики с единственного хоста), можно удалить в файлах JSON во всех именах метрик префикс **$server_name** до проведения импорта описания JSON. Это более трудоемкий вариант и его выбирать не рекомендуется.

Для импорта описаний информационных панелей необходимо выполнить следующие шаги:

1) В меню **Dashboards** вашего сайта Grafana выбрать пункт **Manage**.
2) В открывшемся списке папок и панелей выбрать существующую или создать новую папку.

3) Находясь в выбранной папке, выбрать в правой верхней части страницы пункт **Import**.
4) На открывшейся странице можно либо нажать справа вверху кнопку **Upload json file** и загрузить файл, либо вставить содержимое JSON-файла в поле под заголовком **Or paste JSON** и нажать кнопку **Load**.
5) После этого нужно заполнить необходимые параметры и выполнить загрузку JSON-описания.
6) При необходимости, отредактируйте наименование источника данных в модели JSON, заменив в описаниях информационных панелей значение поля `datasource` с `Graphite` на корректное значение. Более простым вариантом может быть смена имени источника данных на `Graphite`. Такой вариант будет предпочтительным, если источник данных до этого нигде больше не использовался.

Перечисленные выше пункты нужно повторить для каждой импортируемой панели.

12.4.7 Информационная панель «Операционная система»

Информационная панель представляет основные системные показатели:

- Время работы экземпляра СУБД «Квант-гибрид»;
- Средняя загрузка;
- Использование ОЗУ;
- Использование памяти;
- Использование дисковой системы, на которой расположен каталог баз данных.

12.4.8 Информационная панель «QHB»

Информационная панель содержит несколько разделов:

- Транзакции;
- Чтение и запись блоков;
- События ожидания;
- Контрольные точки и операции с буферами;

– Архивация WAL.

В каждом разделе представлены наборы тематических панелей, отражающие основные показатели.

Примечание. По умолчанию теги различных событий не заданы, поэтому необходимо самостоятельно настроить вывод дополнительных аннотаций:

– добавить необходимые комментарии к данным метрик функцией *qhb_annotation* (см. подраздел «Аннотации» ниже);

– настроить вывод доступных аннотаций в информационной панели Grafana.

12.4.9 QHB-мониторинг для Zabbix.

Пример использования сервера Zabbix 6.2 для мониторинга узла сети с СУБД «Квант-гибрид» и агентом Zabbix.

Необходимые файлы:

– ***template_db_qhb.yaml*** – файл шаблона «QHB by Zabbix agent», который нужно импортировать на сервер Zabbix 6.2;

– ***template_db_qhb.conf*** – файл с пользовательскими параметрами агента Zabbix для опроса СУБД «Квант-гибрид»;

– каталог ***qhb/***, содержащий SQL-файлы, к которым обращаются пользовательские параметры.

Архив для Zabbix 6.2 или выше расположен в репозитории.

12.4.10 Установка

Примечание: более подробную информации см. в документации Zabbix по работе с шаблонами агента (ознакомиться можно по адресу https://www.zabbix.com/documentation/6.2/en/manual/config/templates_out_of_the_box/zabbix_agent)

1) Установите Zabbix-агент на узле сети с СУБД «Квант-гибрид»
2) Скопируйте каталог ***qhb/*** в домашний каталог Zabbix-агента ***/var/lib/zabbix***. Если в ***/var/lib/*** отсутствует каталог ***zabbix/***, то его необходимо создать. Каталог ***qhb/*** содержит SQL-файлы, необходимые для получения метрик из СУБД «Квант-гибрид1»:

```
# mkdir -p /var/lib/zabbix/qhb
# cd /var/lib/zabbix/qhb
# wget <сервер>/zabbix/qhb/qhb.tar
# tar -xvf qhb.tar
# chmod -R 707 /var/lib/zabbix/qhb
# rm -rf qhb.tar
```

3) Скопируйте файл ***template_db_qhb.conf*** в конфигурационную директорию Zabbix-агента ***/etc/zabbix/zabbix_agentd.d/***:

```
# wget <сервер>/zabbix/template_db_qhb.conf
```

4) Создайте пользователя *zbx_monitor* с правами «только чтение» и доступом к экземпляру СУБД «Квант-гибрид»:

```
CREATE USER zbx_monitor WITH PASSWORD '<PASSWORD>' INHERIT;
GRANT pg_monitor TO zbx_monitor;
```

5) Отредактируйте файл ***qhb_hba.conf***, чтобы разрешить подключение к Zabbix. Для этого откройте его в текстовом редакторе nano и вставьте в него следующие строки:

```
host all zbx_monitor 127.0.0.1/32 trust
```

6) Перезагрузите СУБД «Квант-гибрид» и Zabbix-агент:

```
# systemctl restart qhb
# systemctl restart zabbix_agentd
```

7) Импортируйте файл ***template_db_qhb.yaml*** шаблона «QHB by Zabbix agent» на сервере Zabbix. Подробнее об импорте шаблонов см. в документации Zabbix по работе с импортом шаблонов (ознакомиться можно по адресу https://www.zabbix.com/documentation/6.2/en/manual/xml_export_import/templates?hl=Template%2Cimport%2Ctemplate#importing).

8) Установите параметры макроса **{$PG.HOST}**, **{$PG.PORT}**, **{$PG.USER}**, **{$PG.PASSWORD}**, **{$PG.DB}** для узла сети с СУБД «Квант-гибрид ».

9) Присоедините шаблон «QHB by Zabbix agent» к узлу сети с СУБД «Квант-гибрид».

12.4.11 Собираемые параметры и триггеры

Триггеры и собираемые параметры описаны в технической документации по СУБД «Квант-Гибрид» на ресурсе разработчика.

12.4.12 Настройка сбора метрик

Для того чтобы информационной панели отображали данные метрик, необходимо выполнить некоторые настройки.

12.4.12.1 Настройка сервера метрик

Рекомендуется в параметре **prefix** конфигурационного файла ***/etc/metricsd/config.yaml*** сервера метрик прописать имя хоста, на котором он работает. Если сделать это для каждого сервера, все метрики будут организованы иерархически, и первый уровень иерархии будет уровнем серверов. В именах метрик в предлагаемых панелях для этих целей присутствует переменная **$server_name**. Подразумевается, что на хосте работает только один кластер баз данных.

12.4.12.2 Настройка параметров базы данных

При настройке СУБД «Квант-гибрид» используется параметр **metrics_collector_path**, который по умолчанию имеет значение *@metrics-collector* (представляет собой путь к сокету домена Unix); сервер метрик по умолчанию запускается именно на этом адресе.

Для настройки отправки аннотаций необходимо в **qhb.conf** прописать следующие параметры:

- **grafana.address** – адрес Grafana, например http://localhost:3000
- **grafana.token** – необходимо указать токен, полученный в Grafana по адресу http://localhost:3000/org/apikeys

Пример настроек в ***qhb.conf*** для отправки метрик и аннотаций:

```
    metrics_collector_path = '@metrics-collector'
    grafana.address = 'http://localhost:3000'
    grafana.token                                                    =
'eyJrIjoiNGxTaloxMUNTQkFUMTN0blZqUTN6REN6OWI5YjM1MzMiLCJuIjoidGVzdCIsI
mlkIjoxfQ=='
```

Примечание. При необходимости записи данных метрик в CSV-файлы в качестве значения параметра **metrics_collector_path** нужно указать путь к файлу сокета домена Unix, например */tmp/metrics-collector.sock*. Это же значение нужно указать в параметре **bind_addr** раздела **collection** в настройках сервера метрик (***/etc/metricsd/config.yaml***).

Для сбора системных метрик (информационная панель «Операционная система») необходимо установить параметр **qhb_os_monitoring** в значение *on* (включен). Можно также задать период сбора системной статистики **qhb_os_stat_period** (значение по умолчанию – 30 секунд). Не рекомендуется задавать для этого параметра слишком низкое значение, поскольку сбор системной статистики требует некоторых затрат.

В файле параметров можно прописать:

```
qhb_os_monitoring = on
qhb_os_stat_period = 60 # если период по умолчанию в 30 секунд
                        # не устраивает
```

Либо выполнить команды:

```
ALTER SYSTEM SET qhb_os_monitoring = ON;
ALTER SYSTEM SET qhb_os_stat_period = 60;
SELECT pg_reload_conf();
```

12.4.12.3 Примеры использования метрик в функциях SQL

Помимо встроенных метрик пользователи могут использовать свои метрики через функции SQL.

12.4.12.3.1 Тип метрик Timer

Используется при фиксации промежутка времени, единицы измерения – наносекунды.

```
SELECT qhb_timer_report('qhb.timer.nano',10000000000 /* 10 секунд
в наносекундах */);
```

12.4.12.3.2 Тип метрик Counter

Используется, когда нужно зафиксировать количество произошедших за промежуток времени событий.

```
    SELECT qhb_counter_increase_by('qhb.example.counter',10);
```

12.4.12.3.3 Тип метрик Gauge

Используется, когда нужно установить некий статичный показатель в определенное значение или изменить его.

```
    SELECT   qhb_gauge_update('qhb.gauge_example.value',   10);   /*
Установка значения */
    SELECT  qhb_gauge_add('qhb.gauge_example.value',1);  /*  Увеличение
значения */
    SELECT  qhb_gauge_sub('qhb.gauge_example.value',1);  /*  Уменьшение
значения */
```

12.4.12.3.4 Аннотации

Используются, если нужно добавить комментарий к данным метрик. Первый параметр функции – текст комментария, последующие параметры – теги.

```
    SELECT      qhb_annotation('Начало      выполнения      теста',
'test','billing'); /* Текст аннотации и два тега */
```

12.5 Инструмент для управления резервным копированием Qbackup

qbackup – инструмент для управления резервным копированием и восстановлением кластеров баз данных СУБД «Квант-гибрид».

Каталог копий – директория с резервными копиями. Внутри нее могут содержаться директории с резервными копиями, информация о них в файле каталога ***catalog.json*** и директория ***wal***. Директорию каталога нужно создать до начала резервного копирования.

Каталог данных – директория с кластером баз данных.

Следующие поля в файле каталога копий являются служебными, но могут быть полезны. Они меняются с каждой минорной версией СУБД «Квант-гибрид»:

– Версия кластера (*cluster_version*) – бинарная версия СУБД «Квант-гибрид». Это номер сборки базы данных.

– Версия ***qbackup*** (*qbackup_version*).

12.5.1 Синтаксис

```
qbackup help [команда]

qbackup list -B каталог_копий [параметр...]

qbackup backup -B каталог_копий [-D каталог_данных] [--incremental]
        [--compress] [--direct] [параметр...]

qbackup replica -D каталог_данных [параметр...]

qbackup restore -B каталог_копий -D каталог_данных
        -i идентификатор_копии [--connection] [параметр...]

qbackup remove -B каталог_копий -i идентификатор_копии ...
        [--single] [параметр...]

qbackup validate -B каталог_копий -i идентификатор_копии
        [-D каталог_данных] [-t тип_проверки] [параметр...]

qbackup backup-wal -B каталог_копий -D каталог_данных [--compress]
-p путь_к_файлу_wal
        -f имя_файла_wal [параметр...]

qbackup restore-wal -p путь_к_файлу_wal
        -f путь_к_файлу_wal [параметр...]
```

12.5.2 Обзор

По сравнению с другими средствами резервного копирования ***qbackup*** имеет следующие преимущества, полезные для реализации различных стратегий резервного копирования и работы с базами данных большого объема:

– Постраничное инкрементальное копирование: позволяет сэкономить место на диске и создавать копии быстрее, чем при полном копировании. Восстановление инкрементальных копий также осуществляется быстрее, чем воспроизведение файлов WAL.

– Реализация создания кумулятивной инкрементальной резервной копии с возможностью выбора родителя для инкрементального бэкапа.
– Параллельное выполнение: выполнение внутренних процессов команд `backup` и `restore` в несколько параллельных потоков.
– Сжатие: хранение копируемых данных в сжатом состоянии для экономии дискового пространства.
– Каталогизация резервных копий: получение списка резервных копий и соответствующей метаинформации в виде простого текста или JSON.
– Расширенные возможности работы с удаленным резервным копированием.
– Для потокового (**stream**) резервного копирования используется протокол репликации.
– Возможность удаленного восстановления из резервной копии.
– Расширенные возможности валидации резервных копий и восстановленных кластеров.

Для управления резервными копиями ***qbackup*** создает *каталог резервных копий*. В этом каталоге сохраняются все файлы резервных копий с дополнительной метаинформацией, а также архивы WAL, необходимые для восстановления на момент времени.

Используя ***qbackup*** в режиме потокового (**stream**) резервного копирования, вы можете выполнять полное или инкрементальное резервное копирование с локального кластера, а также с удаленного:

– Полные резервные копии содержат все файлы, необходимые для восстановления кластера баз данных с нуля.
– Инкрементальные копии создаются на уровне страниц и включают только те данные, которые изменились со времени последнего копирования. Это позволяет сэкономить место на диске и создавать копии быстрее, чем при полном копировании. Восстановление инкрементальных копий также осуществляется быстрее, чем воспроизведение файлов WAL.

qbackup в режиме прямого (**direct**) резервного копирования может производить полное копирование как работающего, так и остановленного

экземпляра. Для копирования WAL используются внутренние механизмы СУБД «Квант-гибрид».

12.5.3 Ограничения

В настоящее время ***qbackup*** имеет следующие ограничения:

- ***qbackup*** работает с серверами СУБД «Квант-гибрид» не ниже версии 1.5.
- На сервере СУБД «Квант-гибрид», где была сделана копия, и на сервере, где она будет восстанавливаться, должны быть одинаковые значения параметров **block_size** и **wal_block_size** и одинаковая основная версия. В зависимости от конфигурации кластера, СУБД «Квант-гибрид» может накладывать дополнительные ограничения, например, по архитектуре процессора и версии libc/libicu.
- Прямое (**direct**) резервное копирование предусмотрено только для создания локальных полных резервных копий. Для инкрементального бэкапирования используется потоковое (**stream**) резервное копирование как для локального, так и для удаленного хоста.
- Невозможно сделать потоковую (**stream**) резервную копию на остановленном сервере. Для этого предусмотрено резервное копирование в режиме **direct**, которое не требует обязательного подключения к базе.
- Инкрементальные резервные копии невозможно базировать на резервных копиях, созданных на остановленном сервере.
- После включении CDC для инкрементального бэкапирования необходимо предварительно сделать полную резервную копию для корректного последующего захвата изменений данных.
- Ограничения, связанные с копированием данных, защищенных с помощью модуля безопасного хранения (см. подраздел Модуль безопасного хранения, а также блоки «Важно» раздела 12.5, касающиеся работы с шифрованными данными).

12.5.4 Установка и подготовка

Установив ***qbackup***, выполните следующие действия:

– Создайте каталог копий.
– Настройте кластер баз данных для использования qbackup.
– Настройте SSH для выполнения удаленного восстановления из резервных копий, если вы планируете использовать данный функционал.

12.5.4.1 Создание каталога копий

qbackup хранит резервные копии в каталоге копий, путь к которому указывается в переменной среды **BACKUPS_DIR** или в аргументе командной строки **-B** (**--backups-dir**). Каталог копий должен быть создан пользователем перед началом работы с утилитой и не должен содержать никаких файлов. Ручное редактирование файлов в каталоге копий не допускается. Пользователь, запускающий ***qbackup***, должен иметь полный доступ к ***каталогу_копий*** и, как минимум, доступ на чтение всего содержимого ***каталога_данных***.

12.5.4.2 Настройка кластера баз данных

Несмотря на то, что ***qbackup*** можно использовать от имени суперпользователя, рекомендуется создать отдельную роль с минимальными правами, необходимыми для выбранной стратегии копирования. В этом руководстве примером такой роли служит роль *backup*.

Программа ***qbackup*** должна читать непосредственно файлы кластера, поэтому запускать ***qbackup*** (или подключаться к нему удаленно) нужно от имени пользователя ОС, который имеет доступ на чтение всех файлов и каталогов внутри каталога данных кластера (***PGDATA***), подлежащего копированию.

12.5.4.2.1 Настройки для использования qbackup в режиме прямого (direct) резервного копирования

Если планируется использоваться бэкап в режиме **stream** для потокового резервного копирования, то данный раздел можно пропустить и сразу настраивать кластер с учетом требований к потоковому резервному копированию.

Для выполнения физического резервного копирования на активной базе данных роль *backup* должна иметь следующие разрешения на сервере СУБД «Квант-гибрид» (только в базе данных, **к которой производится подключение**):

```
BEGIN;
CREATE ROLE backup WITH LOGIN;
GRANT USAGE ON SCHEMA pg_catalog TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.qhb_start_backup(text, boolean, boolean) TO backup;
GRANT EXECUTE ON FUNCTION pg_catalog.qhb_stop_backup(boolean, boolean) TO backup;
COMMIT;
```

В файле ***qhb_hba.conf*** разрешите подключение к кластеру баз данных пользователю с именем *backup*.

Для возможности прямого резервного копирования рабочего экземпляра СУБД «Квант-гибрид» необходимо дополнительно настроить архивацию WAL, как указано в подразделе «Настройка непрерывного архивирования WAL» настоящего документа.

12.5.4.2.2 Настройки для использования qbackup в режиме потокового (stream) резервного копирования

Программа ***qbackup*** в режиме **stream** позволяет получить копию кластера базы данных с удаленного или локального сервера по сети, используя протокол репликации. Если вы используете ***qbackup*** через отдельного пользователя (например, *backup*), не забудьте дать ему роль **REPLICATION** в базе данных и соответствующие права в ***qhb_hba.conf***.

Пример:

```
CREATE USER backup REPLICATION PASSWORD 'backup';
```

В файле ***qhb_hba.conf*** разрешите выполнение репликации для роли *backup*:

```
# qhb_hba.conf
host replication backup <master_ip> md5
host replication backup <standby_ip> md5
host all backup <master_ip> md5
```

```
host all backup <standby_ip> md5
```

В файле ***qhb.conf*** установите в параметре **wal_level** значение выше *minimal*.

Пример:

```
# qhb.conf
wal_level='replica'
```

Для потокового резервного копирования предусмотрены полное и инкрементальное бэкапирование (в том числе удаленное) на включенной СУБД. Для инкрементальных резервных копий реализован механизм CDC – захват изменения данных (Change Data Capture). Для использования CDC необходимо настроить параметр **qcdc_registry_size**. Он принимает размер в мегабайтах. После перезапуска базы данных в ее корне появится файл ***qhb-cdc.map*** с указанным размером. Размер может быть любым в указанных далее пределах; для оптимальной работы следует выбирать его исходя из правила: предполагаемый размер кластера ÷ 1000. Максимальное значение этого параметра – 10 Гб (10240 Мб), минимальное – 1 Мб. При значении 0 CDC считается выключенным.

Пример:

```
# qhb.conf
# Предполагаемый размер кластера – 30GB
qcdc_registry_size=30MB
```

После включении CDC необходимо сделать полную резервную копию для корректного последующего захвата изменений данных при инкрементальном создании резервной копии.

В режиме потокового резервного копирования все необходимые файлы WAL будут автоматически архивированы (см. описание параметра **--wal-method** ниже).

Важно! При наличии шифрованных таблиц в базе данных не рекомендуется использовать потоковое резервное копирование с указанием **wal_method=stream**. Для формирования зашифрованной потоковой резервной копии требуется наличие непрерывной архивации WAL и потокового резервного копирования в режиме **wal_method=fetch** или **wal_method=none**.

Если вы планируете восстановление на момент времени, то необходимо дополнительно настроить архивацию WAL, как указано в подразделе «Настройка непрерывного архивирования WAL» ниже.

12.5.4.3 Настройка непрерывного архивирования WAL

Для выполнения локального резервного копирования, а также для восстановления на момент времени должно осуществляться непрерывное архивирование WAL.

Чтобы настроить непрерывное архивирование через программу ***qbackup***, модифицируйте файл конфигурации ***qhb.conf***:

– Установите в параметре **wal_level** значение выше *minimal*.

– Установите в параметре **archive_mode** значение *on* или *always*.

– Установите параметр **archive_command**:

```
archive_command='путь_установки/qbackup backup-wal
-B каталог_копий -f %f -p %p'
```

Здесь ***путь_установки*** – путь к каталогу установленной версии ***qbackup***, которую вы хотите использовать.

В результате команды `qbackup backup-wal` все сегменты WAL будут располагаться в директории ***каталог_копий/wal***. Подкаталог ***/wal*** будет сформирован автоматически после начала работы.

Примечание: вместо команды `qbackup backup-wal` можно воспользоваться любым другим средством при условии, что в процессе непрерывного архивирования сегменты WAL будут попадать в каталог ***каталог_копий/wal***.

Пример возможной конфигурации для копирования работающего кластера:

```
# qhb.conf
wal_level='replica'
archive_mode='on'
archive_command='/usr/bin/qbackup backup-wal -B /opt/backup_dir -f
%f -p %p'
```

Установка **archive_command** необходима для прямого (direct) резервного копирования работающего кластера. Для копирования остановленного кластера этот

параметр устанавливать не требуется. Для потокового резервного копирования или при использовании подкоманды `replica` параметры **archive_mode** и **archive_command** можно не настраивать. Все необходимые файлы WAL будут автоматически архивированы (возможны варианты архивации).

12.5.4.4 Настройка копирования с резервного сервера

Утилита qbackup может копировать данные с резервного сервера. Для этого требуется дополнительная настройка:

- на основном сервере в ***qhb.conf*** установите для параметра **full_page_writes** значение *on* и **ненулевое** значение для параметра **archive_timeout**;
- на резервном сервере в ***qhb.conf*** установите для параметра **archive_mode** значение *always*, для параметра **hot_standby** – *on*.

Пример:

```
# qhb.conf
full_page_writes = on
hot_standby = on
# Следующий параметр обязателен для резервного сервера, если предполагается архивация WAL
archive_mode = always
```

Для получения самодостаточных копий с резервного сервера выполните действия, описанные в подразделе «Настройки для использования qbackup в режиме потокового (stream) резервного копирования» настоящего документа.

Для получения архивных копий с резервного сервера выполните действия, описанные в подразделе «Настройка непрерывного архивирования WAL» настоящего документа.

После этих подготовительных действий вы можете делать резервные копии с резервного сервера в режимах **stream** и **direct**.

Копирование с резервного сервера имеет следующие особенности настройки и ограничения:

1) Если резервный сервер повышается до ведущего во время копирования, копирование прерывается с ошибкой.

2) На основном сервере необходимо установить ненулевой параметр **archive_timeout**. Это связано с тем, что утилита qbackup на резервном сервере ожидает переключения сегмента WAL на основном сервере для возможности завершить архивацию на резервном сервере. В противном случае выводится уведомление:

```
    Warning! Still waiting for all required wal segments to be archived.
    # Предупреждение! Все еще ожидаю завершения архивации всех требуемых сегментов WAL.
```

3) Все необходимые для резервной копии записи WAL должны содержать полные страницы. Для этого требуется включить режим **full_page_writes** на основном сервере и отказаться от использования в **archive_command** утилит, удаляющих полные страницы из файлов WAL.

4) Чтобы qbackup на резервном сервере смогла отправить команду переключения сегмента WAL на основной сервер, необходимо предоставить права на переключение сегмента пользователю, который отвечает за репликацию, например пользователю *repluser*. Какой пользователь отвечает за репликацию, можно увидеть в ***qhb.auto.conf***.

```
    GRANT execute ON function pg_switch_wal TO repluser
```

12.5.4.5 Настройка SSH для выполнения удаленного восстановления

Программа qbackup поддерживает удаленный режим восстановления. В этом режиме каталог резервных копий находится в локальной системе, а целевой экземпляр СУБД «Квант-гибрид» работает на удаленном компьютере. В настоящее время удаленное восстановление поддерживается с использованием только одного сетевого протокола – SSH.

Если вы хотите использовать qbackup для восстановления резервной копии в удаленном режиме (применяя SSH), выполните следующие действия:

1) Установите qbackup в обеих системах: ***backup_host*** (сервер резервного копирования) и ***db_host*** (сервер баз данных). На удаленном компьютере

qbackup должна находиться в таком месте, чтобы она могла запускаться без указания пути к программе

2) Для организации связи между узлами настройте SSH-соединение без пароля для подключения пользователя (например, *backup*) на компьютере ***backup_host*** к серверу ***db_host*** под именем *qhb*:

```
[backup@backup_host] ssh-keygen -t rsa
[backup@backup_host]  ssh-copy-id  -i  ~/.ssh/id_rsa.pub qhb@db_host
```

При необходимости может потребоваться запустить ssh-агент вручную:

```
[backup@backup_host] eval `ssh-agent`
[backup@backup_host] ssh-add
```

3) Проверьте верность настроек с помощью попытки удаленного запуска на ***backup_host*** команды:

```
[backup@backup_host] ssh qhb@db_host qbackup apply-inc --help
```

Здесь:

– *backup_host* – система, в которой находится каталог копий.

– *db_host* – система, в которой функционирует экземпляр СУБД «Квант-гибрид».

– *backup* – пользователь в системе *backup_host*, от имени которого запускается qbackup.

– *qhb* – пользователь в системе *db_host*, от имени которого запускается экземпляр СУБД «Квант-гибрид».

В удаленном режиме восстановления qbackup работает следующим образом:

– Основной процесс запускается в системе ***backup_host*** и подключается к системе ***db_host***.

– В процессе работы на ***db_host*** копируются файлы резервной копии, а файлы инкрементальных изменений накладываются путем вызова на ***db_host*** программы qbackup.

– Если ***qbackup*** на ***db_host*** сталкивается с ошибкой, основной процесс qbackup выдает информацию о возникшей ошибке и завершает работу.
– Распаковка сжатой резервной копии всегда осуществляется в системе ***backup_host***.

12.5.5 Использование

12.5.5.1 Создание резервной копии

12.5.5.1.1 Создание прямой (direct) резервной копии

Для создания прямой резервной копии с локального хоста выполните следующую команду:

```
qbackup backup -B каталог_копий -D каталог_данных --direct [--compress]
    [параметры подключения...] [параметр...]
```

Флаг **--compress** значительно уменьшает размер резервной копии, но может привести к замедлению процесса копирования.

12.5.5.1.2 Создание потоковой (stream) резервной копии

Для создания потоковой резервной копии выполните следующую команду:

```
qbackup backup -B каталог_копий [--compress] [--incremental] [--from идентификатор_копии]
    [параметры подключения...] [параметр...]
```

qbackup в режиме **stream** (по умолчанию) поддерживает два основных типа создания резервных копий:

– Полные копии. Содержимое ***каталога_данных*** копируется в ***каталог_копий***, за исключением необязательных файлов. В ***каталоге_копий*** может содержаться любое количество независимых резервных копий.
– Инкрементальные копии. При передаче флага **--incremental** qbackup запрашивает у сервера данные об изменившихся файлах и на основе этих данных создает резервную копию. В *инкрементальной копии* хранятся только те файлы (или части файлов), которые изменились с момента

создания *полной копии*. Это значительно сокращает размер резервной копии, но может потребовать больше времени для выполнения операций сравнения файлов. (Это экспериментальная функция, но, несмотря на это, резервные копии всегда будут согласованными.) Обратите внимание, что инкрементальные резервные копии могут создаваться продолжительное время и иметь большой размер.

При восстановлении кластера из инкрементальной копии qbackup использует родительскую полную копию и все инкрементальные копии между ними, в совокупности образующие «цепочку копий». Таким образом, прежде чем делать инкрементальные копии, необходимо сделать как минимум одну полную, причем после включения CDC.

Флаг **--compress** значительно уменьшает размер резервной копии, но может привести к замедлению процесса копирования.

Флаг **--from** применяется для создания дополнительной цепи резервных копий. Применяется только для инкрементальных резервных копий для выбора на основе какой из копий (любых полных или инкрементальных) будет сделан инкремент. Также этот флаг необходим, если восстановлению подлежала не последняя резервная копия из цепочки; в таком случае для продолжения инкрементального бэкапирования необходимо указать **–from *id_восстановленного_бэкапа***. Последующие инкрементальные резервные копии могут быть созданы без дополнительного параметра **--from**. Подробнее можно ознакомиться в подразделе «Дополнительные параметры для команды backup» настоящего документа.

При указании удаленного хоста можно получить копию кластера базы данных с удаленного сервера по сети, используя протокол репликации. Подробнее о параметрах репликации можно прочитать ниже в подразделе «Дополнительные параметры копирования».

qbackup поддерживает протокол репликации для получения резервных копий с удаленного сервера.

12.5.5.2 Создание реплики кластера

Для создания реплики выполните следующую команду на хосте предполагаемой реплики, используя параметры подключения к основному серверу:

```
qbackup replica -D каталог_данных [--compress] [параметры подключения...]
    [параметры репликации...] [параметр...]
```

Эта подкоманда полезна для создания резервного сервера.

Флаг **--compress** значительно уменьшает размер резервной копии, но может привести к замедлению процесса копирования.

12.5.5.3 Восстановление кластера

Чтобы восстановить кластер баз данных из резервной копии, выполните команду `restore`:

```
qbackup restore -B каталог_копий -D каталог_данных
  -i идентификатор_копии [--tablespace-mapping старый_путь=новый_путь] [--connection] [--wal-directory] [параметры...]
```

Здесь:

- ***каталог_копий*** – каталог, в котором хранятся все файлы резервных копий и метаданные.
- ***каталог_данных*** – каталог, в котором будут храниться данные восстановленного кластера. Этот каталог должен быть создан перед выполнением команды, быть пустым и доступным для записи.
- ***идентификатор_копии*** определяет, из какой резервной копии будет восстановлен кластер. Если вы выбираете для восстановления инкрементальную копию, qbackup автоматически восстанавливает нижележащую полную копию и затем последовательно применяет все необходимые добавления.

Параметр **--connection** используется при восстановлении на удаленный хост. См. описание параметра в подразделе «Параметры подключения для удаленного восстановления» настоящего документа.

Параметр **--wal-directory** используется для переопределения директории расположения архива WAL файлов, если она отличалась от ***каталога_копий***. См. описание параметра **--wal-directory** ниже.

См. описание параметра **--tablespace-mapping** ниже.

По окончании работы команды `restore` запустите службу базы данных, используя ***qhb_ctl***.

12.5.5.4 Выполнение восстановления на момент времени (PITR)

Возможно восстановить состояние кластера на любой момент времени (до заданной точки восстановления), используя с командой `restore` параметры точки восстановления.

– Чтобы восстановить состояние кластера на определенный момент времени, укажите это время в параметре **--target-time**, в формате *timestamp*. Например:

```
qbackup restore -B каталог_копий -D каталог_данных
  -i идентификатор_копии --target-time='2017-05-18
14:18:11+03'
```

– Чтобы восстановить состояние кластера до определенной транзакции, воспользуйтесь параметром **--target-xid**. Например:

```
qbackup restore -B каталог_копий -D каталог_данных
  -i идентификатор_копии --target-xid=687
```

– Чтобы восстановить состояние кластера до определенной позиции в журнале (LSN), воспользуйтесь параметром **--target-lsn**. Например:

```
qbackup restore -B каталог_копий -D каталог_данных
  -i идентификатор_копии --target-lsn=16/B374D848
```

– Чтобы восстановить состояние кластера до заданной именованной точки восстановления, воспользуйтесь параметром **--target-name**. Например:

```
qbackup restore -B каталог_копий -D каталог_данных
  -i идентификатор_копии --target-
name='before_app_upgrade'
```

– Чтобы восстановить самое раннее из возможных согласованное состояние кластера, укажите флаг **--immediate**:

```
qbackup restore -B каталог_копий -D каталог_данных
  -i идентификатор_копии --immediate
```

В дополнение к параметрам, устанавливающим точку восстановления, можно также указать параметры **--action**, **--timeline** и **--inclusive**.

12.5.5.5 Проверка резервных копий

Команда `validate` позволяет проверить (и в некоторых случая восстановить) целостность резервной копии, а также восстановленного кластера. Существует три вида проверки:

12.5.5.5.1 checksum (режим по умолчанию)

По умолчанию по окончании резервного копирования qbackup подсчитывает контрольную сумму всех файлов внутри директории резервной копии, что позволяет впоследствии проверять их согласованность. Этот вариант позволяет сравнить эту контрольную сумму с той, которая будет вычислена в процессе проверки.

Если контрольные суммы не совпадают, то следует перейти к режиму **qhb-checksum** для выявления поврежденного файла таблицы. Если проверка прошла успешна, это означает, что повреждены второстепенные файлы базы.

12.5.5.5.2 qhb-checksum

Позволяет проверить контрольные суммы файлов таблиц баз данных. Эта функция аналогична ***qhb_checksums*** за исключением того, что данный режим работает с каталогизированными qbackup резервными копиями. Если резервная копия была сделана с включенными контрольными суммами, то данный вариант позволяет сверить их. Если контрольные суммы были выключены в кластере на момент резервного копирования, то флаг **-e –enable** позволяет включить их. Может использоваться исключительно на полных несжатых резервных копиях в режиме **direct**.

12.5.5.5.3 database

Этот режим позволяет двоично сравнить файлы внутри резервной копии с файлами кластера. Проверка должна производится на выключенном сервере. Если будут найдены поврежденные файлы, то будет предложено их восстановить. Этот

вариант полезен для проверки только что восстановленной базы данных, а также тех файлов таблиц, которые не должны были измениться с момента восстановления. Кроме того, в этом режиме можно проверить только что созданную/восстановленную не архивированную полную резервную копию.

Пример:

```
    $ qbackup backup -B /tmp/backup -h localhost -u user1 -fc # сетевая полная сжатая резервная копия ID: QZWGEI
    $ qbackup backup -B /tmp/backup -h localhost -u user1 -fi # сетевая инкрементальная резервная копия ID: QZWGFJ

    $ qbackup validate -B /tmp/cat -t checksum -i QZWGFJ # Проверка типа checksum
    $ qbackup validate -B /tmp/cat -t qhbchecksum -i QZWGEI # Проверка типа qhb-checksum

    $ qbackup restore -B /tmp/backup -i QZWGFJ -D /tmp/qhb-data/ # Восстановление

    # Предположительное повреждение файла `/tmp/qhb-data/base/12677/175`

    $ qbackup validate -B /tmp/backup -i QZWGFJ -D /tmp/qhb-data/ -t database # Проверка типа database
    You are going to restore
    /tmp/cat/QZWH7I/base/12677/175 into /tmp/qhb-data/base/12677/175
    You are going to restore
    /tmp/cat/QZWH7I/base/12677/175 into /tmp/qhb-data/base/12677/175
     yes

    $ qbackup validate -B /tmp/backup -i QZWGFJ -D /tmp/qhb-data/ -t database # Проверка типа database (успешная)
```

При использовании режима проверки **database** можно использовать флаг **-y --yes**, позволяющий автоматически согласиться на восстановление

Примечание. В каких ситуациях восстановить файл не получится:

– повреждены одинаковые файлы внутри резервной копии и внутри кластера;

– файл были изменен с помощью СУБД;

– файл был полностью удален;

– в файле поврежден заголовок (информация о последнем изменении).

Примечание: если вы видите предупреждение о том, что файл не найден, то, скорее всего, за время работы СУБД были созданы новые файлы таблиц. Если это не так, значит, был потерян файл в резервной копии.

12.5.5.6 Запуск qbackup в параллельных потоках

Команды `backup` и `restore` могут выполняться в несколько параллельных потоков. Это может существенно ускорять работу qbackup при наличии достаточных ресурсов (ядер процессора, производительности дисковой подсистемы и сети).

Параллельным выполнением управляет ключ командной строки **--threads**. Например, чтобы запустить резервное копирование в четыре параллельных потока, выполните:

```
qbackup backup -B каталог_копий [-D каталог_данных] [--direct] [--compress] [--incremental] -j 4
```

По умолчанию или если заданное число потоков равно 0, qbackup автоматически выбирает число потоков, которое обеспечит максимальную производительность копирования (в ущерб другим операциям).

12.5.5.7 Управление каталогом резервных копий

С помощью qbackup можно управлять резервными копиями в командной строке:

– просматривать имеющиеся резервные копии;

– удалять резервные копии.

12.5.5.7.1 Просмотр информации о резервных копиях

Чтобы просмотреть список существующих копий для каждого экземпляра, выполните команду `list`:

```
qbackup list -B каталог_копий
```

qbackup выводит список всех имеющихся резервных копий.

Для каждой копии выдаются следующие сведения:

ID – уникальный идентификатор копии.

status – состояние резервной копии. Возможные варианты:

– *Done* – резервная копия выполнена и готова к использованию;

– *In Progress* – резервное копирование еще выполняется;

– *Error* – резервное копирование завершилось с ошибкой;

– *Cancelled* – резервное копирование отменено.

size – размер резервной копии (в несжатом виде).

compressed size – размер сжатой резервной копии (только для сжатых копий).

kind – тип резервной копии. Возможные варианты:

– *Full (Stream)* – полная потоковая резервная копия;

– *Incremental (Stream)* – инкрементальная потоковая резервная копия;

– *Full (Direct)* – полная прямая резервная копия;

is compressed – сжата ли копия.

start lsn – LSN (Log Sequence Number, последовательный номер в журнале WAL) в момент начала копирования. Отсутствует для копий остановленного кластера.

parent – идентификатор предыдущей копии для инкрементальных копий;

start time – системное время начала резервного копирования;

end time – системное время окончания резервного копирования.

Также можно получить информацию о резервных копиях в формате JSON:

```
qbackup list -B каталог_копий --json
```

Более подробную системную информацию можно получить в файле ***каталог_копий/catalog.json***.

12.5.5.7.2 Удаление резервных копий

Для удаления ставшей ненужной резервной копии выполните команду `remove`:

```
qbackup remove -B каталог_копий -i идентификатор_копии
```

Эта команда удалит резервную копию с заданным ***идентификатором_копии*** вместе со всеми инкрементальными копиями, которые от нее зависят (если таковые имеются). Таким образом вы можете удалить некоторые последние инкрементальные

копии из «цепочки копий». В параметре **-i** можно передавать несколько идентификаторов через пробел. Перед удалением программа выводит информацию об удаляемых резервных копиях и требует дополнительного подтверждения; это можно выключить с помощью флага **-y**.

Подробнее о расширенных параметрах можно прочитать в подразделе «Дополнительные параметры для команды remove» настоящего документа.

12.5.6 Работа в командной строке

12.5.6.1 Команды

В этом подразделе описываются команды qbackup. Необязательные параметры этих команд заключаются в квадратные скобки. Подробно все параметры описываются ниже в подразделе «Параметры».

12.5.6.1.1 help

```
qbackup help [команда]
```

Выдает справку по командам qbackup. Если в параметрах задается одна из команд qbackup, выводит подробную информацию по параметрам, которые принимает эта команда.

Аналогичную роль выполняет флаг **--help**, доступный для любой из команд.

12.5.6.1.2 list

```
qbackup list -B каталог_копий [--json] [--help]
```

Показывает содержимое каталога копий.

По умолчанию содержимое каталога представляется в виде обычного текста. Можно передать флаг **--json**, чтобы получить результат в формате JSON.

Более подробно использование этой команды описывается в подразделе «Управление каталогом резервных копий» настоящего документа.

12.5.6.1.3 backup

```
qbackup backup -B каталог_копий [-D каталог_данных] [--direct]
[--compress] [--incremental]
[--help] [-j число_потоков] [--progress] [параметры репликации...]
```

Создает резервную копию экземпляра СУБД «Квант-гибрид».

Более подробно использование этой команды описывается в подразделе «Создание резервной копии» настоящего документа.

Подробнее о расширенных параметрах этой команды можно прочитать в подразделе «Дополнительные параметры для команды backup» настоящего документа.

12.5.6.1.4 restore

```
qbackup restore -B каталог_копий -D каталог_данных -i идентификатор_копии
[--help]
[-j число_потоков] [--progress] [--connection] [--wal-directory]
[параметры точки восстановления...]
```

Восстанавливает экземпляр СУБД «Квант-гибрид» из резервной копии, расположенной в ***каталоге_копий***, на локальный или удаленный хост.

Более подробно использование этой команды описывается в подразделе «Восстановление кластера» настоящего документа. Подробную информацию по параметрам точки восстановления см. в подразделе «Выполнение восстановления на момент времени (PITR)» настоящего документа.

12.5.6.1.5 replica

```
qbackup replica -D каталог_данных
[--compress] [--help] [--progress] [параметры репликации...]
```

Создает реплику экземпляра СУБД «Квант-гибрид».

Более подробно использование этой команды описывается в подразделе «Создание реплики кластера» настоящего документа.

12.5.6.1.6 remove

```
qbackup remove -B каталог_копий -i идентификатор_копии [идентификатор_копии_2]... [--single]
[--help] [--progress]
```

Удаляет копию с заданным ***идентификатором_копии*** и все инкрементальные копии, которые от нее зависят. Если передан флаг **--single**, удаляет только указанную копию.

Подробнее о расширенных параметрах этой команды можно прочитать в подразделе «Дополнительные параметры для команды remove» настоящего документа.

Более подробно использование этой команды описывается в подразделе «Удаление резервных копий» настоящего документа.

12.5.6.1.7 validate

```
qbackup validate -B каталог_копий -i идентификатор_копии [-D
каталог_данных] [-t тип_проверки]
[--help] [--progress] [параметр...]
```

Проверяет корректность резервной копии с заданным ***идентификатором_копии***. Существует три типа проверки: **checksum**, **qhb-checksum** и **database**; подробную информацию см в подразделе «Проверка резервных копий» настоящего документа.

12.5.6.1.8 backup-wal

```
qbackup backup-wal -B каталог_копий [--compress] --wal-file-
name=имя_файла_wal --wal-file-path=путь_к_файлу_wal
[--help]
```

Копирует файлы WAL в соответствующий подкаталог каталога копий. При необходимости для расположения WAL можно использовать любую директорию, отличную от ***каталога_копий***, и учитывать это при выполнении `restore-wal`.

Флаг **--compress** значительно уменьшает размер архивированного WAL, но может привести к замедлению процесса копирования.

Команду `backup-wal` можно указать как значение параметра **archive_command** СУБД «Квант-гибрид» при настройке непрерывного архивирования WAL. Подробную информацию см. в подразделе «Настройка непрерывного архивирования WAL» настоящего документа.

12.5.6.1.9 restore-wal

```
qbackup restore-wal --destination-wal-file-
path=путь_и_имя_файла_wal --source-wal-file-
path=путь_к_исходному_файлу_wal
```

```
[--help]
```

Копирует файлы WAL в соответствующий подкаталог с нужным именем.

Команда `restore-wal` автоматически используется как `restore_command` в ***qbackup.conf*** после восстановления.

Пример использования:

```
restore_command='qbackup restore-wal -f /tmp/catalog/wal/%f -p %p'
```

12.5.6.2 Параметры

12.5.6.2.1 Общие параметры

Общие параметры поддерживаются большей частью команд qbackup.

```
-j
--threads
```

Позволяет указать количество потоков для операций, которые поддерживают параллельное выполнение (`backup` и `restore`).

По умолчанию, если значение параметра не установлено или равно 0, используется оптимальное число потоков для данной системы.

```
-P
--progress
```

Выводит интерактивную информацию о процессе выполнения операции. Предназначено для отображения пользователям. При работе потокового резервного копирования также получает информацию о размере табличных пространств.

```
-v
--verbose
```

Выводит более подробную информацию о процессе выполнения операции.

12.5.6.2.2 Параметры подключения

12.5.6.2.2.1 Параметры подключения для резервного копирования

Для подключения к базе необходимо указать ряд параметров. Обязательными для успешного подключения являются только два из них: имя пользователя и имя хоста базы. Необходимость остальных параметров определяется настройками кластера.

Параметры подключения используются только в команде `backup` и только для резервного копирования работающего кластера.

```
-u
--user
```

Имя пользователя базы, от имени которого выполняется копирование. Значение по умолчанию *qhb*.

Если установлена переменная среды **QBACKUP_USER**, используется ее значение. Параметр, указанный в командной строке, имеет приоритет над переменной среды.

```
-h
--host
```

Указывает имя хоста компьютера, на котором работает сервер. Если значение начинается со слэша, оно используется в качестве каталога для сокета домена Unix. Множественные имена хоста можно перечислить через запятую. Каждое из имен будет опробовано последовательно. Значение по умолчанию *localhost*.

Если установлена переменная среды **QBACKUP_HOST**, используется ее значение. Параметр, указанный в командной строке, имеет приоритет над переменной среды.

```
--db-name
```

Имя базы данных для подключения. По умолчанию соответствует имени пользователя.

Если установлена переменная среды **QBACKUP_DBNAME**, используется ее значение. Параметр, указанный в командной строке, имеет приоритет над переменной среды.

```
-p
--port
```

Порт для подключения к базе. Несколько портов можно перечислить через запятую. Число указанных портов может быть равно 1, (в этом случае для всех хостов используется один и тот же порт), либо должно соответствовать указанному число хостов. Значение по умолчанию равно *5432*.

Если установлена переменная среды **QBACKUP_PORT**, используется ее значение. Параметр, указанный в командной строке, имеет приоритет над переменной среды.

```
--password
```

Пароль пользователя для подключения к базе.

Если установлена переменная среды **QBACKUP_PASSWORD**, используется ее значение. Параметр, указанный в командной строке, имеет приоритет над переменной среды. Если пароль не указан, но он потребовался в процессе работы программы, он будет запрошен в диалоговом режиме. После первой неверной попытки будет выведено сообщение об ошибке аутентификации.

```
-w
--no-password
```

Никогда не входить в диалоговый режим для запроса пароля. Этот параметр может быть полезен в скриптах автоматизации.

12.5.6.2.2.2 Параметры подключения для удаленного восстановления

```
-C
--connection
```

Параметр подключения к удаленному хосту через ssh для восстановления из резервной копии.

Задается в формате *-C user@ip:port* при настроенном ssh-соединении между хостами. См. подраздел «Настройка SSH для выполнения удаленного восстановления» настоящего документа.

12.5.6.2.3 Параметры точки восстановления

Для команды `restore` можно указать дополнительные параметры, регулирующие восстановление до определенной точки (PITR).

Используются только в команде `restore`.

```
--target-time=момент_времени
```

Восстановление до определенного момента времени в формате *timestamp*.

Значение этого параметра задается в том же формате, что принимается типом данных *timestamp with time zone*, за исключением того, что в нем нельзя использовать

сокращенное название часового пояса. Поэтому рекомендуется задавать числовое смещение от UTC или записывать название часового пояса полностью, например, *Europe/Helsinki* (но не EEST).

```
--target-xid=xid_транзакции
```

Восстановление на определенный идентификатор транзакции.

Имейте в виду, что числовое значение идентификатора отражает последовательность именно старта транзакций, а фиксироваться они могут в ином порядке. Восстановлению будут подлежать все транзакции, которые были зафиксированы до указанной (и, возможно, включая ее, в зависимости от значения параметра **--inclusive**).

```
--target-lsn=[ latest | LSN ]
```

Восстановление на определенный LSN. Этот параметр принимает значение системного типа данных *pg_lsn*. С ключевым словом *latest* восстанавливаться будет до последнего LSN в архиве.

```
--target-name=имя_точки_восстановления
```

Восстановление на определенную именованную точку восстановления, созданную с помощью функции *pg_create_restore_point()*.

```
--immediate
```

Данный параметр указывает, что процесс восстановления должен завершиться, как только будет достигнуто целостное состояние, т. е. как можно раньше. При восстановлении из оперативной резервной копии, это будет точкой, в которой завершился процесс резервного копирования.

```
--action=[ pause | promote | shutdown ]
```

Указывает, какое действие должен предпринять сервер после достижения цели восстановления. Вариант по умолчанию – *pause*, что означает приостановку восстановления. Второй вариант, *promote*, означает, что процесс восстановления завершится, и сервер начнет принимать подключения. Наконец, с вариантом *shutdown* сервер остановится, как только целевая точка восстановления будет достигнута.

Вариант *pause* позволяет выполнить запросы к базе данных и убедиться в том, что достигнутая цель оказалась желаемой точкой восстановления. Для снятия с паузы нужно вызвать функцию *pg_wal_replay_resume()*, что в итоге приведет к завершению

восстановления. Если окажется, что желаемая точка восстановления еще не достигнута, нужно остановить сервер, установить более позднюю цель и перезапустить сервер для продолжения восстановления.

Вариант *shutdown* полезен для получения готового экземпляра сервера в желаемой точке. При этом данный экземпляр сможет воспроизводить дополнительные записи WAL (а при перезапуске ему придется воспроизводить записи WAL после последней контрольной точки).

Внимание. Так как ***recovery.signal*** не переименовывается, когда в **--action** выбран вариант *shutdown*, при последующем запуске будет происходить немедленная остановка, пока вы не измените конфигурацию или не удалите файл ***recovery.signal*** вручную.

Этот параметр не действует, если целевая точка восстановления не установлена.

```
--timeline=[ current | latest | id_временной_шкалы ]
```

Указывает временную шкалу для восстановления. Значение может задаваться числовым идентификатором временной шкалы (***id_временной_шкалы***) или ключевым словом. С ключевым словом *current* восстанавливается та временная шкала, которая была активной при создании базовой резервной копии. С ключевым словом *latest* восстанавливаться будет последняя временная шкала, найденная в архиве, что полезно для резервного сервера. По умолчанию подразумевается *latest*.

```
--inclusive=[ on | off ]
```

Прекратить восстановление в момент достижения заданной точки (*on*) или непосредственно перед ее достижением (*off*). По умолчанию установлено *on*.

```
--wal-directory=каталог_архива_wal
```

Переопределить директорию с архивом WAL. Требуется при указании директории в **archive_command**, отличной от ***каталога_копий***. По умолчанию параметр соответствует **-B** ***каталог_копий***.

12.5.6.2.4 Дополнительные параметры копирования

```
-R
--write-recovery-conf
```

Записать строку подключения к основному серверу в ***qhb.auto.conf***. Записывает параметры **primary_slot_name** и **primary_conninfo**. Это параметр по умолчанию; при ошибке подключения к основному серверу нужно отредактировать эти параметры вручную.

```
-r
--max-rate=максимальная_скорость
```

Ограничить скорость передачи файлов. Принимает суффиксы *k* – КБ и *M* – МБ. Принимает значения в пределах от 32 КБ/с до 1024 МБ/с.

```
-X
--wal-method=[ stream | fetch | none ]
```

Способ архивирования WAL-файлов во время репликации.

Здесь:

- *stream* – архивировать WAL в отдельном потоке параллельно. Это значение по умолчанию. Файлы архивируются в ***$PGDATA/pg_wal/***.
- *fetch* – архивировать WAL вместе с файлами кластера. Они будут расположены внутри директории ***$PGDATA/pg_wal/***.
- *none* – не архивировать файлы WAL.

Важно: при выборе метода *none* файлы WAL **не** будут архивироваться! Вам следует архивировать файлы WAL вручную, иначе резервная копия будет недействительна!

Важно: при наличии шифрованных таблиц потоковый метод (**wal-method=stream**) использовать не рекомендуется.

```
-S
--slotname=имя_слота
```

Создать слот репликации для архивирования WAL-файлов. По умолчанию создается временный слот репликации, а этот параметр создает постоянный слот репликации.

Примечание: Имя слота репликации может включать в себя **только** английские буквы нижнего регистра (a-z), цифры (0-9) и символ нижнего подчеркивания (**_**)**.**

```
-T
```

```
--tablespace-mapping старый_путь=новый_путь
```

Выполняет адресацию табличных пространств. Адресация – изменение оригинального пути директории табличного пространства на новый. Этот параметр можно указывать несколько раз.

Здесь:

– ***старый_путь*** – оригинальный путь табличного пространства.

– ***новый_путь*** – новый путь (путь адресации) табличного пространства.

12.5.6.2.5 Прочие параметры

12.5.6.2.5.1 Дополнительные параметры для команды backup

```
-i
--incremental
```

Позволяет создавать инкрементальные резервные копии.

```
-c
--compress
```

Позволяет уменьшить размер резервной копии.

```
-f
--fast-checkpoint
```

Запрашивает выполнение быстрой контрольной точки вместо обычной. Применяется при резервном копировании с горячего сервера или при репликации.

```
--skip-broken
```

Позволяет базировать новые инкрементальные копии на последней **успешной** копии, пропуская ошибочные без их непосредственного удаления.

```
-C
--comment
```

Позволяет добавить комментарий для резервной копии.

```
--no-checksum
```

Позволяет выключить подсчет контрольной суммы резервной копии. Применяйте, только если уверены, что осознаете все последствия.

```
--from
```

Позволяет выбрать родительскую резервную копию для создания от него инкрементальной копии. Применяется для создания дополнительной цепи резервных копий. Для указания используется идентификатор резервной копии.

12.5.6.2.5.2 Дополнительные параметры для команды remove

```
-y
--yes
```

Позволяет удалить резервные копии без дополнительного предостережения.

```
--prune
```

Позволяет удалить все резервные копии.

```
--single
```

Позволяет удалить одну копию без удаления цепочки.

Важно: после удаления резервной копии из цепочки все копии, сделанные позже нее, становятся недействительными!

12.6 Модуль прямой загрузки данных

Модуль прямой загрузки данных (Quantum Direct Loader, QDL) – утилита, позволяющая осуществить загрузку из файла формата CSV в таблицу формата СУБД «Квант-гибрид» согласно конфигурации, минуя обычные механизмы базы данных. Полученный файл можно скопировать в качестве файла таблицы в каталог базы данных. Скорость работы при этом существенно выше, чем при выполнении команд `INSERT` и `COPY`, за счет использования оптимизированного многопоточного кода, отсутствия блокировок таблицы и обхода транзакционного ядра. Это дополнительный модуль, который устанавливается отдельно.

12.6.1 Поддерживаемые типы полей

Таблица 26. Поддерживаемые типы полей

Типы	**Названия**
Целочисленные	*smallint, int2, integer, int4, int, bigint, int8*
Числовые с плавающей запятой	*float4, real, float8, double precision*
Последовательности	*smallserial, serial, bigserial*

Типы	Названия
Текстовые	*character, char, varchar, character varying, text*
Числа с указанной точностью	*decimal, numeric, dec*
Логические	*bool, boolean*
Идентификаторы	*uuid, oid*
Временные	*timestamp, date*
Двоичные	*bytea*

Примечание. В текущей версии QDL допускается использование значений NaN (Not a Number) в таких типах данных, как *float* (числовые с плавающей запятой) и *numeric* (числа с указанной точностью). В дополнение к обычным числовым значениям и NaN *float* также имеют специальные значения: *infinity* и *-infinity*. Обратите внимание, что все значения, находящиеся в файле формата CSV, не заключаются в кавычки.

Идентификаторы *UUID* записываются в виде последовательности шестнадцатеричных цифр в нижнем регистре, разделенных знаками минуса на несколько групп, в таком порядке: группа из 8 цифр, за ней три группы из 4 цифр и наконец группа из 12 цифр, что в сумме составляет 32 цифры. Пример *UUID* в этом стандартном виде: *a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11*.

Временные типы поддерживают такие специальные значения как *infinity*, *-infinity* и *epoch*.

12.6.2 Синтаксис

```
qdl [флаги] [параметры] [субкоманды]
```

12.6.2.1 Флаги

```
-h
--help
```

Выводит справочную информацию об аргументах командной строки ***qdl***.

```
-v
--verbose
```

Выводит отладочную информацию ***qdl***.

```
-V
--version
```

Выводит информацию о версии ***qdl***.

12.6.2.1.1 Параметры

```
-c
--config <файл_конфигурации>
```

Путь к файлу конфигурации ***qdl***.

```
-d
--data <файл_данных>
```

Путь к исходным данным. Используйте '**-**', чтобы читать из stdin.

12.6.2.1.2 Субкоманды

```
create_table
```

Генерирует SQL-скрипт для создания таблицы.

```
help
```

Выводит список доступных субкоманд или справку по заданным субкомандам.

```
insert_values
```

Заполняет файл с таблицей значениями.

```
validate
```

Проверяет правильность файла CSV с данными.

12.6.3 Команда create_table

Генерирует SQL-скрипт, создающий таблицу СУБД согласно конфигурации, а также заполняет ее первыми тремя кортежами CSV-таблицы. Это требуется для создания структуры таблицы, которая будет заполнена данными. Также это позволяет получить пути расположения файлов и их названия (OID, требуемый для генерирования командой `insert_values`).

`qdl --config <`***`файл_конфигурации`***`> --data <`***`файл_данных`***`> create_table [`**`флаги`**`]`

12.6.3.1 Флаги

```
-h
--help
```

Выводит справочную информацию об аргументах командной строки `create_table`.

```
-V
--version
```

Выводит информацию о версии `create_table`.

Примечание. В формате CSV все символы являются значимыми. Значение в таблице, дополненное пробелами или любыми другими символами, кроме *разделителя*, будет включать эти символы. Это может приводить к ошибкам при заполнении таблицы данными из системы, дополняющей строки CSV пробельными символами до некоторой фиксированной ширины. В случае возникновения такой проблемы необходимо обработать файл CSV и удалить из него замыкающие пробельные символы, прежде чем загружать данные из него в СУБД «Квант-гибрид».

12.6.3.1.1 Пример использования

```
qdl --config config.yml
    --data data.csv
    create_table
```

12.6.4 Команда insert_values

Производит загрузку из файла в формате CSV в таблицу формата, используемого СУБД «Квант-гибрид». Данная команда выполняется многопоточно; количество ядер, отводимое под параллельную обработку, указывается в файле конфигурации. Пользователь может установить пороговый процент срабатывания механизма очистки избыточно выделенной памяти.

Параметр **--out-dir <*выходной_каталог*>** является обязательным для ввода из командной строки.

Также рекомендуется предварительно удостовериться, что версия компоновки страницы СУБД «Квант-гибрид» равна 4. Сделать это можно, например, при помощи расширения ***pageinspect::page_header***.

```
    qdl --config <файл_конфигурации> --data <файл_данных> insert_values
[параметры] --out-dir <выходной_каталог> [флаги]
```

12.6.4.1 Флаги

```
    -h
    --help
```

Выводит справочную информацию об аргументах командной строки `insert_values`.

```
    -V
    --version
```

Выводит информацию о версии `insert_values`.

12.6.4.2 Параметры

```
    -o
    --out-dir <выходной_каталог>
```

Выходной каталог.

```
    -r
    --ram-usage-percent <%_оперативной_памяти>
```

Процент используемой оперативной памяти, после которой запускается сборщик мусора для внутренних буферов (от 1 до 99; значение по умолчанию — 80).

12.6.4.3 Пример использования

```
    qdl --config config.yml
        --data data.csv
        insert_values
        --out-dir output/
```

12.6.5 Команда validate

Проверяет логическую целостность CSV-файла и согласованность структуры данных с файлом конфигурации, сигнализируя об ошибках соответствующим кодом завершения.

```
qdl --config <файл_конфигурации> --data <файл_данных> validate [флаги]
```

12.6.5.1 Флаги

```
    -h
```

```
    --help
```

Выводит справочную информацию об аргументах командной строки `validate`.

```
    -V
    --version
```

Выводит информацию о версии `validate`.

12.6.5.2 Пример использования

```
    qdl --config config.yml
        --data data.csv
        validate
```

12.6.6 Файл конфигурации

Файл конфигурации имеет следующие разделы:

1) ***general*** – базовая конфигурация приложения. Содержит поля:
   - **threads** (целочисленное) – количество потоков-обработчиков сырых данных.
   - **chunk_size** (целочисленное, необязательное) – размер внутреннего буфера передачи данных; данный параметр может меняться с целью оптимизации в нестандартных аппаратных конфигурациях.
   - **checksum** (логическое, необязательное, по умолчанию *true*) – задает расчет контрольных сумм.
2) ***data_source*** – выбор источника данных. В данной версии единственным поддерживаемым форматом является *csv*. Содержит поля:
   - **delimiter** – разделитель столбцов для CSV.
   - **double_quote** – если *true*, то две кавычки подряд воспринимаются как одна экранированная кавычка.
   - **escape** – символ, используемый для экранирования.
   - **comment** – символ, отмечающий начало комментария.
3) ***output*** – конфигурация выходных данных. Содержит поля:
   - **segment_size** (целочисленное, необязательное) – максимальное количество страниц в сегменте базы данных.

- **oid** (целочисленное) – идентификатор основной таблицы.
- **toast_oid** (целочисленное, необязательное) – идентификатор таблицы TOAST.

4) ***table*** – конфигурация таблицы. Содержит поля:
   - **name** (строковое) – имя таблицы.
   - **fields** (повторяющиеся *field*) – описание каждого из полей (например, *field_1 : varchar* и необязательная пометка *[nullable]*).

12.6.6.1 Пример файла конфигурации

```
    # Конфигурация системы:
    general:
        # Количество потоков, выделяемых для анализа и сериализации
строк.
        # Обратите внимание, что помимо этих потоков обязательно
выделяются
        # еще два: поток чтения и поток записи в файл.
        threads: 2
        checksum: true

    # Конфигурация исходных данных:
    data_source:
        csv:
            # Разделитель столбцов для CSV:
            delimiter: ";"

    # Конфигурация получаемых данных:
    output:
        # Идентификатор объекта. Определяет целочисленный идентификатор
для
        # группы файлов целевой таблицы.
        oid: 16385
        # Идентификатор таблицы для хранения больших данных
        toast_oid: 16386
```

```
# Конфигурация целевой таблицы:
table:
    # Имя таблицы. Требуется для задания имени создаваемой таблицы.
    name: "t_qdl"
    # Столбцы в формате "имя" : "тип"
    fields:
        id : integer
        social_id : integer [nullable]
        first_name : varchar
        last_name : varchar
        city : varchar (30)
        profession : varchar (30) [nullable]
        salary : double precision
        description : varchar [nullable]
```

12.6.7 Сценарий использования

Важно. Во избежание проблем с правами доступа, все последующие операции требуется проводить от имени пользователя, который запускает экземпляр базы данных.

Шаг 0 (необязательный)

Проверить CSV файл на согласованность с конфигурацией. Это может сэкономить значительное количество времени в случаях, когда конвертируется CSV-файл большого объема, так как если в ходе работы команды `insert_values` будет обнаружена ошибка, то ее выполнение придется повторить после исправления причин ее возникновения.

```
qdl --config config.yml --data data.csv validate
```

Шаг 1

Сгенерировать SQL-скрипт для создания таблицы, в которую планируется загрузить данные, а впоследствии использовать ее в СУБД. Также этот скрипт получает OID (идентификатор объекта; потребуется для файла конфигурации `insert_values`) и пути файлов базы данных. Эти операции могут быть произведены вручную, но во избежание возможных ошибок рекомендуется использовать скрипт.

```
qdl --config config.yml --data data.csv create_table
```

Шаг 2

Исполнить SQL-скрипт. OID, полученный в ходе выполнения, требуется разместить в файле конфигурации ***config.yml***.

Шаг 3

Загрузить таблицу. Этот процесс может занять значительное время, в зависимости от объема загружаемых данных. Потоки, выполняющие чтение/конвертацию/запись, имеют определенные названия; это позволит системному администратору оценить загруженность каждого отдельно взятого потока ядра и определить их количеством таким образом, чтобы добиться оптимальной производительности. Внутренние буфера ***qdl*** не имеют ограничения размера, а значит, при **очень** медленной записи на диск возможна ситуация, когда эти буферы займут всю оперативную память.

```
    qdl --config config.yml --data data.csv insert_values --out-dir
./output_dir/
```

Шаг 4

Заменить файлы СУБД сгенерированными при помощи ***qdl*** – копируем файлы при помощи команд операционной системы. После этого нужно выполнить следующий запрос:

```
    SELECT qhb_attach_qdl_data('имя_таблицы');
```

При этом сбрасывается кэш данных таблицы и соответствующей ей таблицы TOAST (если она имеется), происходит переподключение файлов данных таблицы, переиндексация таблицы TOAST, регистрация данных для функционала CDC (Change Data Capture).

Примечание: для чтения из stdin нужно использовать вертикальную черту (|). Примеры использования команд:

Команда `create_table`:

```
    cat data.csv | qdl --config config.yml --data - create_table
```

Команда `validate`:

```
    cat data.csv | qdl --verbose --config config.yml --data - validate
```

Команда `insert_values`:

```
    cat data.csv | qdl --config config.yml --data - insert_values --
out-dir output/
```

Вызов справочной информации:

```
    qdl -h
```

12.7 Модуль прямой загрузки данных метрик

Для загрузки данных метрик в базу данных используется модуль ***QDLM*** (Quantum Direct Loader for Metrics), который, в свою очередь, основан на функциональности прямой загрузки данных (предварительная установка ***QDL*** для работы ***QDLM*** не требуется). Функциональность прямой загрузки данных работает в обход стандартного механизма и позволяет в разы сократить время, необходимое для загрузки данных. При такой загрузке данные не попадают в стандартные файлы WAL, и данные не появятся при восстановлении базы из резервной копии и последующего воспроизведения WAL. Но при использовании инкрементального резервного копирования эти данные будут скопированы автоматически.

12.7.1 Настройка загрузки данных метрик в таблицу базы данных

В настоящее время можно организовать как локальную обработку CSV-файлов метрик на том же экземпляре СУБД «Квант-гибрид 1.5», так и централизованную обработку CSV-файлов из нескольких источников. Во втором случае подразумевается либо запись файлов в каталог CSV-файлов от нескольких серверов метрик, расположенных на одном хосте с несколькими экземплярами СУБД «Квант-гибрид 1.5», либо копирование CSV-файлов с удаленных хостов любыми средствами операционной системы. Возможна также комбинация этих вариантов.

12.7.2 Установка QDLM

Установка ***QDLM*** осуществляется аналогично другим модулям СУБД «Квант-гибрид 1.5».

12.7.2.1 Настройка параметров в файле параметров QDLM

В ***/etc/qdlm/config.yaml*** нужно прописать значения для следующих параметров:

```
    # максимальный период переключения csv-файлов (в минутах),
поступающих
    # от различных серверов метрик, задается для проверки
сформированных
    # ранее csv-файлов, которые могли быть пропущены при обработке,
    # и пропуска тех, которые могут быть записаны еще не до конца
    max_rotation_age: 60

    # каталог CSV-файлов
    csv_directory: "/var/lib/qhb/csv_files"

    # параметры соединения с базой данных
    qhb_connection: "host=localhost port=5432 user=qhb dbname=qhb"

    # файл параметров прямой загрузки данных
    qdl_config: "/etc/qdlm/qdl.yml"

    # каталог данных
    qhb_data: "/var/lib/qhb/data"
```

Параметры соединения описываются в технической документации по СУБД «Квант-Гибрид 1.5» на ресурсе разработчика. Особенностью ***QDLM*** является возможность подключения к базе данных только на локальном хосте. В текущем релизе это обусловлено копированием сформированных файлов таблиц напрямую в каталог базы данных в локальной файловой системе. В случае необходимости использования отдельной базы данных (например, *metrics*) и **отдельного пользователя для базы данных метрик** (например, *metrics_user*) можно выполнить от имени суперпользователя следующий набор команд из ***psql***:

```
    CREATE DATABASE metrics;
    CREATE USER metrics_user WITH login;
    GRANT all ON database metrics to metrics_user;
    \c metrics;
    GRANT usage ON schema pg_toast to metrics_user;
    GRANT all privileges ON all tables IN schema pg_toast to
metrics_user;
```

После этого в строке соединения можно прописать соответствующие имена пользователя и базы данных:

```
    # параметры соединения с базой данных
    qhb_connection:   "host=localhost   port=5432   user=metrics_user
dbname=metrics"
```

12.7.2.2 Файл параметров прямой загрузки данных метрик

Предварительно настроенный файл параметров прямой загрузки данных метрик копируется при установке ***QDLM*** и не требует изменений. Путь к нему по умолчанию: ***/etc/qdlm/qdl.yml***.

12.7.2.3 Запуск утилиты QDLM

Запуск прямой загрузки данных метрик производится следующей командой:

```
    qdlm load -c /etc/qdlm/config.yml
```

Запуск прямой загрузки данных метрик с параметром **-D /<PATH>/<QHB-DATA>** переназначает каталог данных.

При запуске утилита проверяет каталог, указанный в параметре **csv_directory** и начинает обработку csv-файлов, которые были обновлены не позднее, чем **max_rotation_age** минут назад. Если в каталог записываются файлы нескольких серверов метрик, в качестве **max_rotation_age** администратором устанавливается максимальное значение в минутах из всех значений параметра **rotation_age** соответствующих серверов метрик. Благодаря такому ограничению все свежие файлы, в которые может происходить запись, не будут обрабатываться. После обработки накопившихся файлов утилита переходит в режим ожидания события окончания записи очередного файла в каталоге, после чего новый файл обрабатывается и его данные заносятся в новую партицию таблицы *metric_archive*. В текущем релизе схема таблицы предзадана и не меняется. После этого происходит проверка, нет ли файлов, события по которым могли быть случайно пропущены (например, такое возможно при слишком большой загрузке системы). Если такие файлы обнаруживаются, они обрабатываются, после чего утилита снова переходит в режим ожидания получения события об окончании записи очередного csv-файла. Обработанные csv-файлы перемещаются в подкаталог ***qdlm_swap*** каталога,

указанного в **csv_directory**. В дальнейшем обработанные csv-файлы можно удалить или переместить в некий архив.

12.7.2.4 Результаты обработки

После запуска обработки CSV-файлов первым шагом проверяется, созданы ли партиционированная таблица *metric_archive* и последовательность *seq_metric_chunk* в схеме public базы данных, указанной в параметрах. Если эти объекты еще не созданы, они создаются автоматически.

Таблица 27. Metric_archive

Столбец	**Тип**	**Описание**
instance_id	*text*	Значение, указанное в параметре **qhb_instance** сервера метрик. В общем случае данные могут поступать из различных источников и относиться к разным базам данных.
metric_dt	*timestamp*	Дата и время формирования метрики
metric_type_id	*smallint*	Тип метрики: 0 – Counter (счетчик), 1 – Gauge (уровень), 2 – Timer (время)
metric_name	*text*	Название метрики
metric_name_ext	*text*	Наименования агрегатов (*std*, *max*, *min*, *sum*, *count*, *median* и перцентили) для метрик типа Timer
metric_value	*double precision*	Значение метрики

Данные каждого CSV-файла обрабатываются аналогично тому, как обрабатываются данные при загрузке таблицы через QDL. После обработки очередного файла в базе данных появляется таблица *metric_archive_**N***, где ***N*** – номер из последовательности *seq_metric_chunk*. По окончании обработки таблица

становится партицией партиционированной таблицы *metric_archive*. Партиционирование идет по двум столбцам: *instance_id* и *metric_dt*. При присоединении таблицы в качестве партиции задаются минимальная и максимальная границы значений столбца *metric_dt*. В дальнейшем при обращении к таблице *metric_archive* при указании условий по *instance_id* и *metric_dt* будет автоматически происходить выбор соответствующих партиций.

12.8 Расширение Rbytea

Rbytea— внешнее хранение двоичных данных.

12.8.1 Тип данных rbytea

Тип данных ***rbytea*** предназначен для хранения двоичных данных. Он аналогичен типу ***bytea***, с той лишь разницей, что сами данные хранятся не в табличном пространстве базы, а во внешнем хранилище. В качестве внешнего хранилища в текущей версии QHB выступает файловая система. Это может быть смонтированный в определенную точку файловой системы сервера внешний том или символическая ссылка.

Основной целью расширения является вынос двоичных данных из таблиц базы данных в нетранзакционное хранилище, тем самым разгрузив базу данных (зачастую двоичные данные имеют большой объем, который занимает значительный процент от общего размера базы, усложняя администрирование и обслуживание).

Тип данных ***rbytea*** в записи базы данных оставляет только небольшой заголовок, в котором содержатся служебные поля и ссылка на файл во внешнем хранилище. В качестве ссылки используется тип данных ***uuid***, а генерация случайных идентификаторов использует возможности модуля ***pgcrypto***.

12.8.2 Установка расширения

Установка расширения производится командой CREATE EXTENSION:

```
CREATE EXTENSION rbytea CASCADE;
```

Установку должен запускать суперпользователь баз данных. Эту команду следует запускать в той базе данных, в которой предполагается использовать модуль.

Для работы фонового процесса необходимо обеспечить предварительную загрузку разделяемой библиотеки расширения, указав к конфигурации параметр:

```
shared_preload_libraries = 'librbytea'
```

Описание параметров расширения приведены в подразделе Параметры конфигурации расширения Rbytea.

12.8.3 Функции для работы с типом rbytea

Функции для работы с типом rbytea приведены в таблице ниже.

Таблица 28. Функции для работы с типом rbytea

Имя	Тип результата	Описание
uuid(rbytea)	***uuid***	Возвращает идентификатор данных
len(rbytea)	***bigint***	Возвращает длину данных в байтах
qss_mode(rbytea)	***bigint***	Возвращает признак зашифрования данных или его отсутствия
md5(rbytea)	***text***	Возвращает md5 сумму данных
sha256(rbytea)	***text***	Возвращает sha256 сумму данных
ext_file_dir(rbytea)	***text***	Возвращает каталог хранения данных (абсолютный или от $PGDATA)
ext_file_path(rbytea)	***text***	Возвращает полное имя файла хранения данных (путь абсолютный или от $PGDATA)
trash_file_dir(rbytea)	***text***	Возвращает каталог хранения удалённых данных (абсолютный или от $PGDATA)

Имя	Тип результата	Описание
trash_file_path(rbytea)	***text***	Возвращает полное имя файла хранения удалённых данных (путь абсолютный или от $PGDATA)
txid_created(rbytea)	***bigint***	Возвращает идентификатор транзакции, во время которой были созданы данные
rvacuum()	***bigint***	Выполняет очистку устаревших данных в хранилище

12.8.4 Фоновый процесс очистки устаревших копий

Поскольку на файловую систему не распространяется транзакционность базы данных, во внешнем хранилище могут оставаться данные полей таблиц типа rbytea, которые были удалены или сохранены в незаконченных, отмененных транзакциях. Поэтому периодически запускается фоновый процесс очистки, который проходит по некоторому диапазону транзакций, очищая данные. При этом файлы перемещаются в каталог TRASH, который создается для каждой базы данных, указанной в параметре rbytea.databases_for_vacuuming.

После каждого запуска максимальный номер транзакции запоминается и при следующем запуске используется как нижняя граница диапазона сканирования. В качестве верхней границы диапазона сканирования используется последняя завершенная транзакция.

12.8.5 Особенности функционирования расширения rbytea

12.8.5.1 Rbytea в кластере и при использовании шифрования

При работе расширения в кластере существуют некоторые ограничения.

Каталог (точки монтирования файловой системы/тома) для сохранения образов данных, задаваемый в параметре **rbytea.filesystem_storage_path**, см. Параметры конфигурации расширения Rbytea, может:

– указывать на каталог в рамках одного локального хоста и использоваться исключительно этим одним хостом кластера,
– или на некоторый каталог сетевого файлового хранилища, используемого сразу всеми хостами кластера.

В первом случае, образы данных будут доступны только на данном конкретном хосте, где они и были созданы, и обращение к данным на других хостах кластера будет приводить к ошибке. Но, в данном случае, возможно использование шифрование данных с использованием QSS.

Во втором случае, образы данных будут доступны на всех хостах кластера, но использование шифрования в текущей версии QHB и расширения ***rbytea*** запрещено, поскольку каждый хост кластера обязан использовать свои рабочие ключи QSS, и зашифрованные одним ключом данные будут недоступны на других хостах (попытка чтения будет приводить к ошибке).

12.8.5.2 Обновление расширения с предыдущих версий

Версия расширения ***rbytea*** версии 1.2 существенно отличается от предыдущих версий расширения. Из-за этого нет возможности провести обновление расширения с ранних версий на текущую с использованием команды

```
ALTER EXTENSION rbytea UPDATE TO '1.2';
```

Возможно только удаление расширения предыдущей версии и создание уже с текущей версией командой

```
CREATE EXTENSION rbytea CASCADE;
```

При этом необходимо позаботиться о сохранении образов данных, если нет цели их полного пересоздания с потерей.

Это можно сделать миграцией

– через тип данных bytea,
– через сохранение образов на диске, где они размещались до миграции (если они не были зашифрованы),
– или через дамп, если образы данных были зашифрованы.

Подробные шаги такой миграции остаются на усмотрение администратора БД.

12.8.6 Параметры конфигурации расширения

Для работы расширения нужно установить несколько параметров в конфигурационном файле (подробную информацию см. в подразделе Параметры конфигурации расширения Rbytea).

12.9 Расширение QHB_audit

Данное расширение описано в подразделе QHB_audit (главы Протоколирование и регистрация событий).

12.10 Расширение Qbim

Расширение Qbim представляет собой конструктор для разработки ERM на темпоральной основе, то есть, моделируемые объекты представляются временными срезами, последовательностями своих состояний во времени. Основное внимание и акцент в данном расширении сделан не на "декомпозиции до конца", а на поддержании темпоральности данных. Расширение реализует краткий набор логики хранения (создания, изменения) объектов в реляционной СУБД (на примере qhb), реализованный в слое хранимых процедур.

Подробную информацию, включая пример использования, можно посмотреть в технической документации по СУБД «Квант-Гибрид» 1.5 на ресурсе разработчика.

Установка расширения производится командой CREATE EXTENSION. Поскольку есть расширения, от которых зависит расширение **Qbim**, рекомендуется использовать установку с предложением **CASCADE**. Расширение **Qbim** переносимое, поэтому его можно установить в различные схемы базы данных, используя **SCHEMA имя_схемы**.

Начиная с версии расширения 1.2 есть возможность использовать различные опции таблицы ***action***.

При обновлении расширения до версии 1.2 с предыдущих версий используйте предложение ***UPDATE*** команды ALTER EXTENSION.

Таблица 29. Таблица Локальных параметров для управления созданием таблицы action

Параметр	**Назначение**	**Значение по-умолчанию**
qbim.action.use_partitions	Создавать таблицу партиционированную по полю ***fct_dt***.	'true'
qbim.action.partition_period	- при создании партиционированной таблицы - размер секции, которые будут созданы при создании. Возможны варианты: 'day', 'week', 'month', 'year'.	'month'
qbim.action.partition_count	- при создании партиционированной таблицы - количество создаваемых секций, не считая двух секций для "самых старых" и "самых новых" значений.	5
qbim.action.use_unlogged	Создавать таблицу с параметром ***unlogged***. Внимательно отнеситесь к этому параметры, его лучше не использовать при работе в кластере.	'false'
qbim.action.use_appendonly	Создавать таблицу как неизменяемую (append_only). Возможно только при включённом TARQ.	'true'
qbim.action.upgrade_exists	Мигрировать данные из уже существующей таблицы ***action***, при обновлении с ранних версии расширения ***Qbim***, до версии 1.2. Если	'true'

Параметр	Назначение	Значение по-умолчанию
	расширение устанавливается впервые, параметр не имеет смысла.	

Параметры нужно установить перед вызовом команд CREATE EXTENSION или ALTER EXTENSION.

Примеры установки:

```
create extension qbim cascade;
set qbim.action.use_partitions='true';
set qbim.action.partition_period='year';
set qbim.action.partition_count=2;

create extension qbim schema my_schema cascade;
```

12.11 Расширение Uref

Тип данных ***uref*** используется в расширении **Qbim** как ссылочный тип на различные экземпляры **объектов**.

В настоящей реализации тип ***uref*** используется в поле ***r_object*** таблицы ***action***, но также может использоваться в ссылочных полях **объектов**.

Тип данных uref состоит из двух частей:

- идентификатора объекта, представленного типом bigint (obj.id),
- идентификатора экземпляра объекта, представленного типом bigint или uuid («локальным» или «глобальным» соответственно).

Более подробную информацию, включая методы работы, можно посмотреть в технической документации по СУБД «Квант-Гибрид» 1.5 на ресурсе разработчика.

12.12 Расширение Qbayes

Qbayes — методы статистического анализа на основе условных вероятностей Байеса.

Расширение QHB Qbayes — это созданный командой разработчиков СУБД инструментарий для задач расчета вероятностных моделей с применением сетей Байеса.

Тестовые сценарии для расширения можно посмотреть в технической документации по СУБД «Квант-Гибрид» 1.5 на ресурсе разработчика.

Исходные данные для различных вычислений могут храниться в таблицах и представлениях внутри базы данных СУБД. К таким данным относятся исторические факты (срабатывание или несрабатывание гипотез) и соответствующие им причины (параметры). В качестве параметров рассматриваются логические (бинарные) значения или скалярные величины, которые методами расширения приводятся к дискретным значениям и разбиваются на диапазоны, определяемые пользователем. Значения гипотез всегда предполагаются логическими (бинарными).

После предварительной статистической обработки исторических фактов (обучения), модель позволяет рассчитывать условные вероятности тех же самых гипотез при новых условиях (новых значениях параметров).

Более подробную информацию по расширению можно посмотреть в технической документации по СУБД «Квант-Гибрид» 1.5 на ресурсе разработчика.

12.13 Расширение Part_Import

Part_Import — частичный импорт файлов данных из внешней базы данных. Данное расширение содержит две функции:

- ***partial_import(*строка_подключения, имена_таблиц*)*** — основная функция расширения, которая служит для копирования файлов, содержащих данные таблиц, из внешней базы данных в локальную. Под таблицами локальной базы данных подразумеваются таблицы, доступные в текущем сеансе подключения, под таблицами внешней — те, доступ к которым мы получаем, воспользовавшись **строкой_подключения**. Имена таблиц для копирования передаются в параметре **имена_таблиц**. Этот параметр имеет тип строкового массива, где каждая строка содержит имя одной таблицы. Имя может быть как полным — т. е. включать пространство имен, так и неполным — т.е. без пространства имен. В

последнем случае сервер должен понимать по неполному имени, о какой конкретно таблице идет речь.

– ***table_script(*oid*)*** — функция, возвращающая SQL-скрипт, позволяющий создать таблицу, OID которой передан в параметре. Скрипт содержит индексы, внешние индексы, ограничения, последовательности, комментарии.

12.13.1 Partial_import

Особенности работы частичного импорта:

1 Блокировки на внешние таблицы не накладываются, блокировки на локальные таблицы накладываются только перед переключением на «новые» файлы с данными; блокировки предельно короткие. Перенесенные данные нескольких таблиц согласованы между собой.

2 Расширение следит за корректностью файлов, для чего сверяет метаданные таблиц внешней и локальной баз данных. В процессе сверки необходимо убедиться, что названия, типы и порядок полей, включая удаленные столбцы, полностью совпадают. Должны совпадать ограничения столбцов, тип хранения столбца. Таблицы TOAST должны быть либо у обеих таблиц, либо ни у одной из них. Должно совпадать количество индексов, их наименования и состав столбцов. Не имеет значение, если не совпадает табличное пространство. Допустим также импорт материализованных табличных представлений, не должно быть проблем и с переносом партиций.

3 В случае наличия ограничения внешнего ключа будут предложены скрипты по их удалению в локальной базе данных и восстановлению по окончании импорта. Если при импорте будет переноситься не только таблица с внешним ключом, но и таблица, на которую внешний ключ ссылается, никаких ограничений не возникнет; при этом данные будут согласованы.

4 Ввиду отсутствия блокировок внешних таблиц теоретически возможно выполнение операций DDL с внешней таблицей в период между

завершением сверки метаданных и началом переноса файлов с данными таблиц. В этом случае результат работы непредсказуем.

5 В процессе работы переносятся только файлы с данными, поэтому рекомендуется пересобрать статистику по импортированными таблицам по окончанию работы импорта.

6 Хотя импорт и инициирует создание контрольной точки на локальной базы данных по окончании своей работы, теоретически возможно аварийное завершение работы базы данных в период между переключением на новые файлы и завершением создания контрольной точки — в этом случае процедура аварийного восстановления с данными импортированных таблиц может дать непредсказуемые результаты.

12.13.2 table_script

Эта функция является адаптацией кода утилиты **pg_dump**. Взята часть, отвечающая исключительно за генерацию скриптов таблиц, индексов, ограничений и последовательностей. Вся остальная функциональность, посвященная подготовке содержимого таблиц к выгрузке, а также сама выгрузка отброшены.

12.14 Oid2name

oid2name — это программа-утилита, помогающая администраторам изучать файловую структуру СУБД. Дополнительную информацию по расширению можно посмотреть в технической документации по СУБД «Квант-Гибрид» 1.5 на ресурсе разработчика.

12.15 Vacuumlo

vacuumlo — удаляет потерянные большие объекты из базы данных СУБД.

12.15.1 Синтаксис

```
vacuumlo [параметр...] имя_бд...
```

12.15.2 Описание

Vacuumlo — это простая программа-утилита, которая удалит все «потерянные» большие объекты из базы данных СУБД. Потерянным большим объектом (БО)

считается любой БО, OID которого не фигурирует ни в одном столбце данных ***oid*** или ***lo*** в базе данных.

Дополнительно обратитен внимание на наличие триггера ***lo_manage*** в модуле **lo**. ***lo_manage*** полезен тем, что пытается предотвратить образование потерянных БО.

Обрабатываются все базы данных, перечисленные в командной строке.

12.15.3 Параметры

Vacuumlo принимает следующие аргументы командной строки:

-l *предел*

--limit=*предел*

Удалять в одной транзакции не более заданного **предела** больших объектов (количество по умолчанию — 1000). Поскольку сервер запрашивает блокировку для каждого удаляемого БО, удаление слишком большого количества больших объектов в одной транзакции может привести к превышению лимита **max_locks_per_transaction**. Если вы хотите удалить все в одной транзакции, установите этот предел равным нулю.

-n

--dry-run

Не удалять ничего, просто показать предполагаемые операции.

-v

--verbose

Выводить множество сообщений о прогрессе.

-V

--version

Вывести версию **vacuumlo** и завершиться.

-?

--help

Показать справку об аргументах командной строки **vacuumlo** и завершиться.

Кроме того, в качестве параметров подключения **vacuumlo** принимает следующие аргументы командной строки:

-h хост

--host=хост

Имя хост-компьютера, на котором работает сервер баз данных.

-p *порт*

--port=*порт*

Порт сервера баз данных.

-U *имя_пользователя*

--username=*имя_пользователя*

Имя пользователя, под которым производится подключение.

-w

--no-password

Не запрашивать ввод пароля. Если серверу требуется аутентификация по паролю, и пароль недоступен с помощью иных средств, таких как файл **.pgpass**, попытка подключения завершится неудачно. Этот параметр может быть полезен в пакетных заданиях и скриптах, где нет пользователя, чтобы ввести пароль.

-W

--password

Принудительно запрашивать пароль перед подключением к базе данных.

Это несущественный параметр, так как **vacuumlo** автоматически запросит пароль, если сервер требует аутентификацию по паролю. Для этого **vacuumlo** потребуется дополнительная попытка подключения к серверу. В некоторых случаях имеет смысл ввести **-W**, чтобы исключить эту лишнюю попытку.

12.15.4 Переменные среды

PGHOST

PGPORT

PGUSER

Параметры подключения по умолчанию.

Эта утилита, как и большинство других утилит СУБД, также использует переменные среды, поддерживаемые libpq. Дополнительную информацию по переменным средам можно посмотреть в технической документации по СУБД «Квант-Гибрид» 1.5 на ресурсе разработчика.

Переменная среды **PG_COLOR** указывает, использовать ли цвет в диагностических сообщениях. Возможные значения: ***always*** (всегда), ***auto*** (автоматически) и ***never*** (никогда).

12.15.5 Примечания

Утилита **vacuumlo** работает следующим образом: сначала она создает временную таблицу, содержащую все OID больших объектов в выбранной базе данных. Затем она сканирует все столбцы в базе данных, имеющие тип ***oid*** или ***lo***, и удаляет соответствующие записи из временной таблицы. (Примечание: рассматриваются только типы именно с этими именами; в частности, домены на их базе не рассматриваются.) Оставшиеся записи во временной таблице указывают на потерянные большие объекты, которые и удаляются.

12.16 Специализированные модули

В данном разделе представлена краткая информация по специализированным модулям. Расширенную информацию по каждому из описанных в данном разделе модулей можно посмотреть в технической документации по СУБД «Квант-Гибрид» 1.5 на ресурсе разработчика.

Специализированные модули:

- **adminpack** — предоставляет ряд вспомогательных функций, которыми могут пользоваться pgAdmin и другие средства администрирования и управления для предоставления дополнительной функциональности;
- **amcheck** — предоставляет функции, позволяющие проверять логическую целостность структуры отношений;
- **auth_delay** — заставляет сервер сделать небольшую паузу перед тем, как выдать сообщение об ошибке аутентификации, чтобы усложнить атаки методом прямого подбора паролей к базам данных;
- **auto_explain** — предоставляет возможность автоматического протоколирования планов выполнения медленных операторов, без необходимости выполнения EXPLAIN вручную;

- **bloom** — предоставляет индексный метод доступа, основанный на фильтрах Блума;
- **btree_gin** — предоставляет типовые классы операторов GIN, реализующие поведение, схожее с тем, что реализуют классы B-дерева, для различных типов данных;
- **btree_gist** — предоставляет классы операторов индексов GiST, реализующие поведение, схожее с тем, что реализуют классы B-дерева, для различных типов данных;
- **citext** — предоставляет тип данных для строк символов, не учитывающих регистр, ***citext***;
- **cube** — реализует тип данных ***cube*** для представления многомерных кубов;
- **dblink** — обеспечивает подключения к другим базам данных QHB из сеанса базы данных:
  - ***dblink_connect*** — открывает постоянное подключение к удаленной базе данных;
  - ***dblink_connect_u*** — открывает постоянное подключение к удаленной базе данных, небезопасно;
  - ***dblink_disconnect*** — закрывает постоянное подключение к удаленной базе данных;
  - ***dblink*** — выполняет запрос в удаленной базе данных;
  - ***dblink_exec*** — выполняет команду в удаленной базе данных;
  - ***dblink_open*** — открывает курсор в удаленной базе данных;
  - ***dblink_fetch*** — возвращает строки из открытого курсора в удаленной базе данных;
  - ***dblink_close*** — закрывает курсор в удаленной базе данных;
  - ***dblink_get_connections*** — возвращает имена всех открытых именованных подключений **dblink**;
  - ***dblink_error_message*** — выдает сообщение последней ошибки для именованного подключения;

  - ***dblink_send_query*** — передает асинхронный запрос в удаленную базу данных;
  - ***dblink_is_busy*** — проверяет, не занято ли подключение асинхронным запросом;
  - ***dblink_get_notify*** — выдает асинхронные уведомления подключения;
  - ***dblink_get_result*** — получает результат асинхронного запроса;
  - ***dblink_cancel_query*** — отменяет любой активный запрос в именованном подключении;
  - ***dblink_get_pkey*** — возвращает позиции и имена полей первичного ключа отношения;
  - ***dblink_build_sql_insert*** — формирует оператор INSERT из локального кортежа, заменяя значения полей первичного ключа переданными альтернативными значениями;
  - ***dblink_build_sql_delete*** — формирует оператор DELETE со значениями, передаваемыми для полей первичного ключа;
  - ***dblink_build_sql_update*** — формирует оператор UPDATE из локального кортежа, заменяя значения полей первичного ключа переданными альтернативными значениями;
- **dict_int** — представляет собой пример дополнительного шаблона словаря для полнотекстового поиска, который был создан для управления индексацией целых чисел (со знаком и без);
- **dict_xsyn** — представляет собой пример дополнительного шаблона словаря для полнотекстового поиска, который заменяет слова группами их синонимов, тем самым позволяя находить слово по одному из его синонимов;
- **earthdistance** — предоставляет два разных способа вычисления ортодромии (расстояния по дуге большого круга; кратчайшего расстояния между двумя точками на поверхности Земли);

- **file_fdw** — предоставляет обертку сторонних данных ***file_fdw***, с помощью которой можно обращаться к файлам данных в файловой системе сервера или выполнять программы на сервере и читать их вывод;
- **fuzzystrmatch** — предоставляет несколько функций для определения схожести и расстояния между строками;
- **hstore** — реализует тип данных ***hstore*** для хранения наборов пар ключ/значение внутри одного значения QHB;
- **intagg** — предоставляет агрегатор и нумератор целых чисел;
- **intarray** — предоставляет ряд полезных функций и операторов для работы с массивами целых чисел без NULL;
- **isn** — предоставляет типы данных для следующих международных стандартов нумерации товаров: EAN13, UPC, ISBN (книги), ISMN (музыка) и ISSN (серийные номера);
- **lo** — предоставляет поддержку управления большими объектами;
- **ltree** — реализует тип данных ***ltree*** для представления меток данных, хранящихся в древовидной иерархической структуре;
- **old_snapshot** — позволяет ознакомиться с состоянием сервера, которое используется в реализации **old_snapshot_threshold**;
- **pageinspect** — предоставляет функции, позволяющие просматривать содержимое страниц базы данных на низком уровне;
- **passwordcheck** — проверяет пароли пользователей, задаваемые командами CREATE ROLE или ALTER ROLE;
- **pg_buffercache** — предоставляет возможность проверить, что происходит в общем кэше буферов в реальном времени;
- **pgcrypto** — предоставляет криптографические функции для СУБД:
  - стандартные функции хеширования;
  - функции хеширования пароля;
  - функции шифрования на основе PGP;
  - низкоуровневые функции шифрования;
  - функции получения случайных данных;

- **pg_freespacemap** — обеспечивает возможность исследовать карту свободного пространства;
- **pg_prewarm** — предоставляет оптимальный способ загружать данные отношений в кэш буферов операционной системы или в кэш буферов QHB;
- **pgrowlocks** — предоставляет функцию, показывающую информацию о блокировке строк для заданной таблицы;
- **pg_stat_statements** — предоставляет возможность отслеживать статистику планирования и выполнения сервером всех операторов SQL;
- **pgstattuple** — предоставляет различные функции для получения статистики на уровне кортежей;
- **pg_surgery** — предоставляет различные функции для проведения операций с поврежденными отношениями;
- **pg_trgm** — предоставляет функции и операторы для определения схожести буквенно-цифрового текста на основе соответствия триграмм, а также классы операторов индексов, поддерживающие быстрый поиск схожих строк;
- **pg_visibility** — предоставляет средства для исследования карты видимости и информации о видимости на уровне страниц для таблицы, а также функции для проверки целостности карты видимости и ее принудительного пересоздания;
- **postgres_fdw** — предоставляет обертку сторонних данных ***postgres_fdw***, которую можно использовать для обращения к данным, хранящимся на внешних серверах QHB;
- **seg** — реализует тип данных ***seg*** для представления отрезков или интервалов с плавающей запятой;
- **sepgsql** — поддерживает мандатное управление доступом с использованием меток, работающее на основе политик безопасности SELinux;

- **spi** — предоставляет несколько рабочих примеров использования Интерфейса программирования сервера (SPI) и триггеров:
  - ***refint*** — функции для реализации ссылочной целостности
  - ***autoinc*** — функции для автоувеличения полей
  - ***insert_username*** — функции для отслеживания пользователя, меняющего таблицу
  - ***moddatetime*** — функции для отслеживания времени последнего изменения
- **sslinfo** — предоставляет информацию о SSL-сертификате, который был предоставлен текущим клиентом при подключении к QHB;
- **tablefunc** — содержит ряд функций, возвращающих таблицы (то есть множества строк) ;
- **tcn** — предоставляет триггерную функцию, уведомляющую блоки прослушивания об изменениях в любой таблице, к которой она привязана
- **test_decoding** — представляет собой пример плагина вывода логического декодирования;
- **tsm_system_rows** — предоставляет метод взятия выборки из таблицы ***SYSTEM_ROWS***, который можно использовать в предложении **TABLESAMPLE** команды SELECT;
- **tsm_system_time** — предоставляет метод взятия выборки из таблицы ***SYSTEM_TIME***, который можно использовать в предложении **TABLESAMPLE** команды SELECT;
- **unaccent** — представляет собой словарь текстового поиска, убирающий надбуквенные (диакритические) знаки из лексем;
- **uuid-ossp** — предоставляет функции для генерирования универсальных уникальных идентификаторов (UUID) по одному из нескольких стандартных алгоритмов;
- **xml2** — предоставляет функциональные возможности для выполнения запросов XPath и преобразований XSLT.

13 ОБЩИЕ КОДЫ ОШИБОК

Всем сообщениям, которые выдает сервер СУБД, назначены пятисимвольные коды ошибок, соответствующие кодам «SQLSTATE», описанным в стандарте SQL. Приложения, должны знать, какое условие ошибки имело место, обычно должны проверять код ошибки, а не обращаться к текстовому сообщению о ней. Не все коды ошибок, которые выдает СУБД, определены стандартом SQL; некоторые дополнительные коды ошибок для условий, не описанных стандартом, были добавлены.

Согласно стандарту, первые два символа кода ошибки обозначают класс ошибок, а последние три символа обозначают определенное условие в этом классе. Благодаря этому, приложение, не знающее значение определенного кода ошибки, все равно может понять, что делать, по классу ошибки.

В приложении к настоящему документу перечислены все коды ошибок, определенные в СУБД. Также показаны классы ошибок. Для каждого класса ошибок имеется «стандартный» код ошибки с последними тремя символами 000. Этот код выдается только для тех условий ошибок, которые относятся к некоторому классу, но не имеют более определенного кода.

Символ, указанный во втором столбце, — имя условия в PL/pgSQL. Имена условий могут записываться в верхнем или нижнем регистре. (Обратите внимание, что PL/pgSQL, в отличие от ошибок, не распознает предупреждения; то есть классы 00, 01 и 02.)

Для некоторых типов ошибок сервер сообщает имя объекта базы данных (таблицу, столбец таблицы, тип данных или ограничение), связанного с ошибкой; например, имя уникального ограничения, вызвавшего ошибку ***unique_violation***. Такие имена передаются в отдельных полях сообщения об ошибке, чтобы приложениям не пришлось извлекать их из возможно локализованного текста сообщения для человека.

14 ОРГАНИЗАЦИОННЫЕ МЕРЫ ЗАЩИТЫ ИНФОРМАЦИИ

1. В случае необходимости существует возможность запретить удаленное подключение к СУБД, используя параметры сервера, указанные в пункте «Параметры подключения» настоящего документа.
2. Запрещено использовать стандартные учетные записи операционной системы и СУБД – как минимум, необходимо изменить пароли учетных записей.
3. Запрещено использовать «пустые», простые или стандартные пароли.
4. Необходимо руководствоваться принципом разумной достаточности при назначении полномочий пользователям (избегать наличия избыточных полномочий).
5. Необходимо организовать процедуры соблюдения соответствия персонала пользователям ОС и СУБД, чтобы избежать накопления «лишних» пользователей при смене персонала.
6. Необходимо организовать периодическую инвентаризацию пользователей СУБД и ОС, и их прав (не реже 1 раза в год).
7. Необходимо организовать в защищенной среде возможность безопасного подключения к СУБД требуемого перечня приложений используя файл qhb_hba.conf и параметры сервера, указанные в пункте «Параметры подключения» настоящего документа.
8. Необходимо обеспечить наличие политик информационной безопасности в Вашей компании, которые регламентируют
    a. сложность паролей (рекомендовано не менее 8 знаков, состоящих из букв разного регистра, цифр и символов) и механизмов их смены (периодичность смены, совпадение с предыдущими паролями или с паролями в других системах), хранения и передачи;
    b. порядок ознакомления с эксплуатационными документами на используемое программное обеспечение (обязательно до использования данного программного обеспечения);

c. порядок журналирования выполнения организационных мер пользователями и администраторами.

9. Необходимо обеспечить выполнение политик информационной безопасности из предыдущего пункта.
10. Необходимо обеспечить отправку сведений о недостатках средства от потребителей средства по стандартным каналам связи на адрес электронной почты (qhb.support@quantom.info) и телефону (+7 (495) 642-97-42 доб. 538).

СОКРАЩЕНИЯ И ОПРЕДЕЛЕНИЯ

Сокращение, обозначение	**Расшифровка**
libpq	Набор библиотечных функций, которые позволяют клиентским программам передавать запросы на сервер PostgreSQL и получать результаты этих запросов. libpq также является базовым движком для нескольких других прикладных интерфейсов PostgreSQL, в том числе написанных для C++, Perl, Python, Tcl и ECPG.
PostgreSQL	Свободная объектно-реляционная система управления базами данных (СУБД). Версия PostgreSQL 14.2. Является базовой основой для разработки СУБД «Квант-Гибрид 1.5».
Postmaster (процесс)	Самый первый процесс экземпляра СУБД. Он запускает другие вспомогательные процессы и управляет ими, а также создает обслуживающие процессы по требованию.
QHB	Объектно-реляционная система управления базами данных «Квант-Гибрид 1.5»
QSS	«Quantum Secure Storage» (QSS), сертифицированный ФСБ России комплекс для криптографической защиты конфиденциальности и целостности информации «Quantum Secure Storage» (сертификат СФ/124-4721 от 15.01.2024). В Системе подключен в виде одноименного модуля.
SQL	Structured Query Language. Специализированный язык структурированных запросов.
Архивирование WAL (процесс)	Процесс сохранения копий файлов WAL в целях создания резервных дубликатов данных или для поддержания актуального состояния реплик.

Сокращение, обозначение	**Расшифровка**
База данных	Совокупность данных, которая рассматривается как модуль; основной целью базы данных является хранение и поиск связанной информации.
БО	Большой объект
Журнал WAL (Хранилище WAL)	Журнал упреждающей записи. Журнал, в котором отслеживаются изменения в кластере баз данных, производимые при выполнении операций пользователями и самой системой.
Записывание WAL (процесс)	Процесс, осуществляющий перенос записей WAL из общей памяти в файлы WAL.
Запись WAL	Низкоуровневое описание отдельного изменения данных. Оно содержит достаточно информации для повторного выполнения (воспроизведения) этого изменения в случае, если при сбое системы оно не зафиксировалось.
ИБ	Информационная безопасность
Кластер	11. Кластер баз данных. Набор баз данных и глобальных SQL-объектов, а также их общих статических и динамических метаданных. 12. Экземпляр СУБД.
Клиент (процесс)	Любой процесс, возможно, удаленный, который запускает сеанс путем подключения к экземпляру СУБД для взаимодействия с базой данных.
Обслуживающий процесс (backend)	Процесс экземпляра СУБД, который реализует сеанс клиента и обрабатывает его запросы.

Сокращение, обозначение	**Расшифровка**
Общая память	Область ОЗУ, совместно используемая процессами, относящимися к одному экземпляру СУБД. В нее отображаются фрагменты файлов базы данных, а также она служит временным хранилищем для записей WAL и содержит дополнительную общую информацию. Общая память относится ко всему экземпляру, а не к отдельной базе данных в нем.
ОС	Операционная система.
Основной (сервер)	Когда две и более базы данных связываются вместе посредством репликации, сервер, считающийся достоверным источником информации, называется *основным* или *главным*.
Пользователь базы данных	Пользователь, который взаимодействует с СУБД и выполняет операции на объектах, хранимых в пределах базы данных.
Резервное копирование	Процесс создания копии данных (англ. backup copy) на носителе (жёстком диске, дискете и т. д.), предназначенном для восстановления данных в оригинальном или новом месте их расположения в случае их повреждения или разрушения.
Реплика	1. База данных, связанная с основной базой и содержащая копию некоторых или всех данных последней. Прежде всего такие базы создаются для расширения возможностей доступа к этим данным, а также для обеспечения доступности данных в случае потери основного сервера. 2. Резервный сервер. Сервер, связанный с основным сервером и содержащий копию некоторых или всех данных последнего.

Сокращение, обозначение	**Расшифровка**
Репликация	Процесс переноса информации с одного сервера на другой называется репликацией. Она может реализовываться как *физическая репликация*, когда с одного сервера на другой точно копируются все изменения на уровне файлов, или как *логическая репликация*, когда изменения в определенном подмножестве данных передаются с помощью представления на более высоком уровне.
Сеанс	Состояние, в котором клиент может взаимодействовать с обслуживающим процессом, используя установленное подключение.
СУБД	Система управления базой данных.
Файл WAL	Один из последовательно нумеруемых файлов, служащих хранилищем WAL. Все эти файлы имеют одинаковый предопределенный размер и записываются по порядку. В файле WAL перемежаются изменения, производимые в нескольких параллельных сеансах. В случае сбоя системы эти файлы также по порядку считываются и все записанные в них изменения воспроизводятся, в результате чего восстанавливается состояние системы до сбоя.
Экземпляр СУБД	Группа обслуживающих и вспомогательных процессов, использующих общую область разделяемой памяти. Экземпляром СУБД управляет один процесс postmaster, а этот экземпляр управляет ровно одним кластером баз данных со всеми его базами. На одном сервере могут работать несколько экземпляров СУБД, если их TCP-порты не конфликтуют.

ПРИЛОЖЕНИЕ «КОДЫ ОШИБОК»

Код ошибки	**Имя условия**
Класс 00 — Успешное завершение	
00000	successful_completion
Класс 01 — Предупреждение	
01000	warning
0100C	dynamic_result_sets_returned
01008	implicit_zero_bit_padding
01003	null_value_eliminated_in_set_function
01007	privilege_not_granted
01006	privilege_not_revoked
01004	string_data_right_truncation
01P01	deprecated_feature
Класс 02 — Нет данных (это также класс предупреждений согласно стандарту SQL)	
02000	no_data
02001	no_additional_dynamic_result_sets_returned
Класс 03 — Оператор SQL еще не завершен	
03000	sql_statement_not_yet_complete
Класс 08 — Исключение, связанное с подключением	
08000	connection_exception
08003	connection_does_not_exist
08006	connection_failure
08001	sqlclient_unable_to_establish_sqlconnection
08004	sqlserver_rejected_establishment_of_sqlconnection
08007	transaction_resolution_unknown
08P01	protocol_violation
Класс 09 — Исключение с действием триггера	
09000	triggered_action_exception
Класс 0A — Неподдерживаемая функциональность	

Код ошибки	**Имя условия**
0A000	feature_not_supported
Класс 0B — Неверное начало транзакции	
0B000	invalid_transaction_initiation
Класс 0F — Исключение с указателем на данные	
0F000	locator_exception
0F001	invalid_locator_specification
Класс 0L — Неверный праводатель	
0L000	invalid_grantor
0LP01	invalid_grant_operation
Класс 0P — Неверное указание роли	
0P000	invalid_role_specification
Класс 0Z — Исключение диагностики	
0Z000	diagnostics_exception
0Z002	stacked_diagnostics_accessed_without_active_handler
Класс 20 — Case не найден	
20000	case_not_found
Класс 21 — Нарушение количества	
21000	cardinality_violation
Класс 22 — Исключение в данных	
22000	data_exception
2202E	array_subscript_error
22021	character_not_in_repertoire
22008	datetime_field_overflow
22012	division_by_zero
22005	error_in_assignment
2200B	escape_character_conflict
22022	indicator_overflow
22015	interval_field_overflow
2201E	invalid_argument_for_logarithm

Код ошибки	**Имя условия**
22014	invalid_argument_for_ntile_function
22016	invalid_argument_for_nth_value_function
2201F	invalid_argument_for_power_function
2201G	invalid_argument_for_width_bucket_function
22018	invalid_character_value_for_cast
22007	invalid_datetime_format
22019	invalid_escape_character
2200D	invalid_escape_octet
22025	invalid_escape_sequence
22P06	nonstandard_use_of_escape_character
22010	invalid_indicator_parameter_value
22023	invalid_parameter_value
22013	invalid_preceding_or_following_size
2201B	invalid_regular_expression
2201W	invalid_row_count_in_limit_clause
2201X	invalid_row_count_in_result_offset_clause
2202H	invalid_tablesample_argument
2202G	invalid_tablesample_repeat
22009	invalid_time_zone_displacement_value
2200C	invalid_use_of_escape_character
2200G	most_specific_type_mismatch
22004	null_value_not_allowed
22002	null_value_no_indicator_parameter
22003	numeric_value_out_of_range
2200H	sequence_generator_limit_exceeded
22026	string_data_length_mismatch
22001	string_data_right_truncation
22011	substring_error
22027	trim_error

Код ошибки	**Имя условия**
22024	unterminated_c_string
2200F	zero_length_character_string
22P01	floating_point_exception
22P02	invalid_text_representation
22P03	invalid_binary_representation
22P04	bad_copy_file_format
22P05	untranslatable_character
2200L	not_an_xml_document
2200M	invalid_xml_document
2200N	invalid_xml_content
2200S	invalid_xml_comment
2200T	invalid_xml_processing_instruction
22030	duplicate_json_object_key_value
22032	invalid_json_text
22033	invalid_sql_json_subscript
22034	more_than_one_sql_json_item
22035	no_sql_json_item
22036	non_numeric_sql_json_item
22037	non_unique_keys_in_a_json_object
22038	singleton_sql_json_item_required
22039	sql_json_array_not_found
2203A	sql_json_member_not_found
2203B	sql_json_number_not_found
2203C	sql_json_object_not_found
2203D	too_many_json_array_elements
2203E	too_many_json_object_members
2203F	sql_json_scalar_required
Класс 23 — Нарушение ограничения целостности	
23000	integrity_constraint_violation

Код ошибки	**Имя условия**
23001	restrict_violation
23502	not_null_violation
23503	foreign_key_violation
23505	unique_violation
23514	check_violation
23P01	exclusion_violation
Класс 24 — Неверное состояние курсора	
24000	invalid_cursor_state
Класс 25 — Неверное состояние транзакции	
25000	invalid_transaction_state
25001	active_sql_transaction
25002	branch_transaction_already_active
25008	held_cursor_requires_same_isolation_level
25003	inappropriate_access_mode_for_branch_transaction
25004	inappropriate_isolation_level_for_branch_transaction
25005	no_active_sql_transaction_for_branch_transaction
25006	read_only_sql_transaction
25007	schema_and_data_statement_mixing_not_supported
25P01	no_active_sql_transaction
25P02	in_failed_sql_transaction
25P03	idle_in_transaction_session_timeout
Класс 26 — Неверное имя оператора SQL	
26000	invalid_sql_statement_name
Класс 27 — Нарушение при изменении данных в триггере	
27000	triggered_data_change_violation
Класс 28 — Неверное указание авторизации	
28000	invalid_authorization_specification
28P01	invalid_password
Класс 2B — Зависимые описания прав все еще существуют	

Код ошибки	**Имя условия**
2B000	dependent_privilege_descriptors_still_exist
2BP01	dependent_objects_still_exist
Класс 2D — Неверное завершение транзакции	
2D000	invalid_transaction_termination
Класс 2F — Исключение в подпрограмме SQL	
2F000	sql_routine_exception
2F005	function_executed_no_return_statement
2F002	modifying_sql_data_not_permitted
2F003	prohibited_sql_statement_attempted
2F004	reading_sql_data_not_permitted
Класс 34 — Неверное имя курсора	
34000	invalid_cursor_name
Класс 38 — Исключение во внешней подпрограмме	
38000	external_routine_exception
38001	containing_sql_not_permitted
38002	modifying_sql_data_not_permitted
38003	prohibited_sql_statement_attempted
38004	reading_sql_data_not_permitted
Класс 39 — Исключение при вызове внешней подпрограммы	
39000	external_routine_invocation_exception
39001	invalid_sqlstate_returned
39004	null_value_not_allowed
39P01	trigger_protocol_violated
39P02	srf_protocol_violated
39P03	event_trigger_protocol_violated
Класс 3B — Исключение точки сохранения	
3B000	savepoint_exception
3B001	invalid_savepoint_specification
Класс 3D — Неверное имя каталога	

Код ошибки	**Имя условия**
3D000	invalid_catalog_name
Класс 3F — Неверное имя схемы	
3F000	invalid_schema_name
Класс 40 — Откат транзакции	
40000	transaction_rollback
40002	transaction_integrity_constraint_violation
40001	serialization_failure
40003	statement_completion_unknown
40P01	deadlock_detected
Класс 42 — Ошибка синтаксиса или нарушение правила доступа	
42000	syntax_error_or_access_rule_violation
42601	syntax_error
42501	insufficient_privilege
42846	cannot_coerce
42803	grouping_error
42P20	windowing_error
42P19	invalid_recursion
42830	invalid_foreign_key
42602	invalid_name
42622	name_too_long
42939	reserved_name
42804	datatype_mismatch
42P18	indeterminate_datatype
42P21	collation_mismatch
42P22	indeterminate_collation
42809	wrong_object_type
428C9	generated_always
42703	undefined_column
42883	undefined_function

Код ошибки	**Имя условия**
42P01	undefined_table
42P02	undefined_parameter
42704	undefined_object
42701	duplicate_column
42P03	duplicate_cursor
42P04	duplicate_database
42723	duplicate_function
42P05	duplicate_prepared_statement
42P06	duplicate_schema
42P07	duplicate_table
42712	duplicate_alias
42710	duplicate_object
42702	ambiguous_column
42725	ambiguous_function
42P08	ambiguous_parameter
42P09	ambiguous_alias
42P10	invalid_column_reference
42611	invalid_column_definition
42P11	invalid_cursor_definition
42P12	invalid_database_definition
42P13	invalid_function_definition
42P14	invalid_prepared_statement_definition
42P15	invalid_schema_definition
42P16	invalid_table_definition
42P17	invalid_object_definition
Класс 44 — Нарушение WITH CHECK OPTION	
44000	with_check_option_violation
Класс 53 — Нехватка ресурсов	
53000	insufficient_resources

Код ошибки	**Имя условия**
53100	disk_full
53200	out_of_memory
53300	too_many_connections
53400	configuration_limit_exceeded
Класс 54 — Превышение ограничения программы	
54000	program_limit_exceeded
54001	statement_too_complex
54011	too_many_columns
54023	too_many_arguments
Класс 55 — Объект не в требуемом состоянии	
55000	object_not_in_prerequisite_state
55006	object_in_use
55P02	cant_change_runtime_param
55P03	lock_not_available
55P04	unsafe_new_enum_value_usage
Класс 57 — Вмешательство оператора	
57000	operator_intervention
57014	query_canceled
57P01	admin_shutdown
57P02	crash_shutdown
57P03	cannot_connect_now
57P04	database_dropped
Класс 58 — Ошибка системы (ошибка, внешняя по отношению к СУБД)	
58000	system_error
58030	io_error
58P01	undefined_file
58P02	duplicate_file
Класс 72 — Ошибка снимка	
72000	snapshot_too_old

Код ошибки	**Имя условия**
Класс F0 — Ошибка файла конфигурации	
F0000	config_file_error
F0001	lock_file_exists
Класс HV — Ошибка обертки сторонних данных (SQL/MED)	
HV000	fdw_error
HV005	fdw_column_name_not_found
HV002	fdw_dynamic_parameter_value_needed
HV010	fdw_function_sequence_error
HV021	fdw_inconsistent_descriptor_information
HV024	fdw_invalid_attribute_value
HV007	fdw_invalid_column_name
HV008	fdw_invalid_column_number
HV004	fdw_invalid_data_type
HV006	fdw_invalid_data_type_descriptors
HV091	fdw_invalid_descriptor_field_identifier
HV00B	fdw_invalid_handle
HV00C	fdw_invalid_option_index
HV00D	fdw_invalid_option_name
HV090	fdw_invalid_string_length_or_buffer_length
HV00A	fdw_invalid_string_format
HV009	fdw_invalid_use_of_null_pointer
HV014	fdw_too_many_handles
HV001	fdw_out_of_memory
HV00P	fdw_no_schemas
HV00J	fdw_option_name_not_found
HV00K	fdw_reply_handle
HV00Q	fdw_schema_not_found
HV00R	fdw_table_not_found
HV00L	fdw_unable_to_create_execution

Код ошибки	**Имя условия**
HV00M	fdw_unable_to_create_reply
HV00N	fdw_unable_to_establish_connection
Класс P0 — Ошибка PL/pgSQL	
P0000	plpgsql_error
P0001	raise_exception
P0002	no_data_found
P0003	too_many_rows
P0004	assert_failure
Класс XX — Внутренняя ошибка	
XX000	internal_error
XX001	data_corrupted
XX002	index_corrupted