УТВЕРЖДАЮ
Генеральный директор
ООО «Кейсистемс»
_____ А. А. Матросов
«___» _____ 2022 г.

ПРОГРАММНЫЙ КОМПЛЕКС «СУБД-КС»

Руководство администратора

ЛИСТ УТВЕРЖДЕНИЯ

Р.КС.09050-01 34 02-ЛУ

СОГЛАСОВАНО
Заместитель генерального директора
ООО «Кейсистемс»
_____ Е. В. Фёдоров
«___» _____ 2022 г.
Системный архитектор
ООО «Кейсистемс»
_____ И. П. Попов
«___» _____ 2022 г.
Руководитель ДПиРСИБ

_____ Д. В. Галкин
«___» _____ 2022 г.

2022

Литера А

Подп и дата	
Инв.N дубл	
Взам.инв.N	
Подп и дата	
Инв.N подл	

УТВЕРЖДЕНО
Р.КС.09050-01 34 02-ЛУ

ПРОГРАММНЫЙ КОМПЛЕКС «СУБД-КС»

Руководство администратора

Р.КС.09050-01 34 02

Листов 79

Подп и дата	
Инв.N дубл	
Взам.инв.N	
Подп и дата	
Инв.N подл	

2022

Литера А

АННОТАЦИЯ

Настоящий документ является частью руководства пользователя программного комплекса «СУБД-КС» (далее – «программный комплекс») по выполнению всех основных функций, связанных с администрированием.

Руководство актуально для указанной версии и для последующих версий вплоть до выпуска обновления руководства.

Порядок выпуска обновлений руководства

Выход новой версии программного комплекса сопровождается обновлением руководства пользователя только в случае наличия в версии значительных изменений режимов, описанных в руководстве, добавления новых режимов или изменения общей схемы работы. Если таких изменений версия не содержит, то остается актуальным руководство пользователя от предыдущей версии с учетом изменений, содержащихся в новой версии.

Перечень изменений версии программного комплекса содержится в сопроводительных документах к версии. Информация об изменениях руководства пользователя публикуется на сайте разработчика в разделе «Документация».

Информация о разработчике ПК «СУБД-КС»

ООО «Кейсистемс»

Адрес: 428000, Чебоксары, Главпочтамт, а/я 172

Телефон: (8352) 323-323

Факс: (8352) 571-033

http://www.keysystems.ru

E-mail: info@keysystems.ru

СОДЕРЖАНИЕ

ВВЕДЕНИЕ **5**
1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ **8**
1.1. НАЗНАЧЕНИЕ ПРОГРАММНОГО КОМПЛЕКСА 8
1.2. УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММНОГО КОМПЛЕКСА 8
2. ОПИСАНИЕ ОПЕРАЦИЙ **9**
2.1. УСТАНОВКА СУБД-КС В LINUX 9
2.1.1. Поддерживаемые дистрибутивы Linux 9
2.1.2. Установка и проверка 9
2.2. НАСТРОЙКА И ЭКСПЛУАТАЦИЯ СЕРВЕРА 9
2.2.1. Регистрация учетной записи пользователя 9
2.2.2. Создание кластера базы данных 10
2.2.2.1. Использование вторичных файловых систем 11
2.2.2.2. Использование сетевых файловых систем 11
2.2.3. Запуск сервера базы данных 11
2.2.3.1. Сбои при запуске сервера 13
2.2.3.2. Проблемы с клиентским подключением 13
2.2.4. Завершение работы сервера 13
2.3. КОНФИГУРАЦИЯ СЕРВЕРА 14
2.3.1. Настройка параметров 14
2.3.1.1. Имена и значения параметров 14
2.3.1.2. Взаимодействие параметров через файл конфигурации 15
2.3.1.3. Взаимодействие с параметрами через SQL 16
2.3.1.4. Управление содержимым файла конфигурации 17
2.3.2. Соединение и аутентификация 17
2.3.2.1. Настройки соединения 17
2.3.2.2. Аутентификация 18
2.3.3. Потребление ресурсов 18
2.3.3.1. Память 18
2.3.3.2. Диск 20
2.3.3.3. Использование ресурсов ядра 21
2.3.3.4. Вакуумная задержка на основе стоимости 21
2.3.3.5. Write Ahead Log 22
2.3.3.6. Асинхронное поведение 23
2.3.4. Журнал опережающей записи 26
2.3.4.1. Настройки 26
2.3.4.2. Контрольные точки (Checkpoints) 30
2.3.4.3. Архивирование 32
2.3.5. Планирование запросов 33
2.3.5.1. Конфигурация метода планирования 33
2.3.5.2. Константы стоимости для планировщика 35
2.3.5.3. Генетический оптимизатор запросов 38
2.3.5.4. Другие параметры планировщика 39
2.3.6. Отчеты об ошибках и ведение журнала 42
2.3.6.1. Место протоколирования 42
2.3.6.2. Время протоколирования 45
2.3.6.3. Протоколируемая информация 47
2.3.6.4. Заголовок процесса 52
2.3.7. Статистика выполнения 52
2.3.7.1. Сборщик статистики запросов и индексов 52
2.3.7.2. Мониторинг статистики 53
2.3.8. Автоматическая очистка 54
2.3.9. Клиентские соединения по умолчанию 56
2.3.9.1. Поведение команд 56
2.3.9.2. Языковая среда и форматирование 62
2.3.9.3. Предварительная загрузка разделяемых библиотек 64
2.3.9.4. Другие значения по умолчанию 66
2.3.10. Управление блокировками 66

2.3.11. Совместимость версии и платформы ... 68
2.3.11.1. Совместимость платформы и клиента ... 68
2.3.12. Обработка ошибок ... 68
2.3.13. Предустановленные параметры ... 69
2.3.14. Индивидуальные параметры ... 71
2.3.15. Параметры разработчика ... 71
2.3.16. Короткие параметры ... 75
Перечень сокращений ... 77
Предметный указатель ... 78
Лист регистрации изменений ... 79

ВВЕДЕНИЕ

Настоящее руководство содержит описание комплекса задач пользователя в части администрирования баз данных СУБД-КС. Задачи включают в себя установку программного обеспечения, настройку и настройку сервера, управление пользователями и базами данных, а также задачи по обслуживанию.

Функциональные возможности

Основные функциональные возможности, предоставляемые при использовании СУБД-КС:

– **Транзакционность** обеспечивается следующими возможностями:

  – полное соответствие принципам **ACID** - атомарность, непротиворечивость, изолированность, сохранность данных.

  – Atomicity - транзакция рассматривается как единая логическая единица, все ее изменения или сохраняются целиком, или полностью откатываются.

  – Consistency - транзакция переводит базу данных из одного непротиворечивого состояния (на момент старта транзакции) в другое непротиворечивое состояние (на момент завершения транзакции). Непротиворечивым считается состояние базы, когда выполняются все ограничения физической и логической целостности базы данных, при этом допускается нарушение ограничений целостности в течение транзакции, но на момент завершения все ограничения целостности, как физические, так и логические, должны быть соблюдены.

  – Isolation - изменения данных при конкурентных транзакциях изолированы друг от друга на основе системы версионности

  – Durability - СУБД-КС заботится о том, что результаты успешных транзакций гарантировано сохраняются на жесткий диск вне зависимости от сбоев аппаратуры.

– **Многоверсионность** (Multiversion Concurrency Control,MVCC) используется для поддержания согласованности данных в конкурентных условиях, в то время как в традиционных базах данных используются блокировки. MVCC означает, что каждая транзакция видит копию данных (версию базы данных) на время начала транзакции, несмотря на то, что состояние базы могло уже измениться. Это защищает транзакцию от несогласованных изменений данных, которые могли быть вызваны (другой) конкурентной транзакцией, и обеспечивает изоляцию транзакций. Основной выигрыш от использования MVCC по сравнению с блокировкой заключается в том, что блокировка, которую ставит MVCC для чтения не конфликтует с блокировкой на запись, и поэтому чтение никогда не блокирует запись и наоборот.

– наличие **Write Ahead Logging** (WAL) - общепринятого механизма протоколирования всех транзакций, что позволяет восстановить систему после возможных сбоев. Основная идея WAL состоит в том, что все изменения должны записываться в файлы на диск только после того, как эти записи журнала, описывающие эти изменения, будут гарантированно записаны на диск. Это позволяет не сбрасывать страницы данных на диск после фиксации каждой транзакции, так как мы уверены, что сможем всегда восстановить базу данных, используя журнал транзакций.

– **Point in Time Recovery** (PITR) - возможность восстановления базы данных (используя WAL) на любой момент в прошлом, что позволяет осуществлять непрерывное резервное копирование кластера СУБД-КС.

– **Репликация**.

- **Целостность данных** является основной характеристикой СУБД-КС. Помимо MVCC, СУБД-КС поддерживает целостность данных на уровне схемы - это внешние ключи (foreign keys), ограничения (constraints).
- **Производительность** СУБД-КС основывается на использовании индексов, интеллектуальном планировщике запросов, тонкой системы блокировок, системой управления буферами памяти и кэширования, превосходной масштабируемости при конкурентной работе.
- **Поддержка индексов**:
  - Стандартные индексы - B-tree, hash, R-tree, GiST (обобщенное поисковое дерево);
  - Частичные индексы (partial indices);
  - Функциональные индексы.
- **Планировщик запросов** основывается на стоимости различных планов, учитывая множество факторов. Он предоставляет возможность пользователю отлаживать запросы и настраивать систему.
- **Система блокировок** поддерживает блокировки на нижнем уровне, что позволяет сохранять высокий уровень конкурентности при защите целостности данных. Блокировка поддерживается на уровне таблиц и записей. На нижнем уровне, блокировка для общих ресурсов оптимизирована под конкретную ОС и архитектуру.
- **Управление буферами и кэширование** используют сложные алгоритмы для поддержания эффективности использования выделенных ресурсов памяти.
- **Tablespaces** (табличные пространства) позволяют гибкое использование дискового пространства для хранения объектов системы, что также повышает производительность и масштабируемость.
- **Расширяемость** СУБД-КС означает, что пользователь может настраивать систему путем определения новых функций, агрегатов, типов, языков, индексов и операторов. Объектно-ориентированность СУБД-КС позволяет перенести логику приложения на уровень базы данных, что сильно упрощает разработку клиентов, так как вся бизнес логика находится в базе данных. Функции в СУБД-КС однозначно определяются названием, количеством и типами аргументов.

Уровень подготовки пользователя

Для успешного освоения материала, изложенного в руководстве пользователя, и формирования навыков работы в программном комплексе с описанными режимами к пользователю предъявляются следующие требования:

- наличие опыта работы с персональным компьютером на базе операционных систем Linux на уровне квалифицированного пользователя;
- умение свободно осуществлять базовые операции в стандартных приложениях Linux.

Перечень эксплуатационной документации

В *таблице 1* представлен список пользовательской документации в части описания комплекса задач «СУБД-КС».

Таблица 1. Перечень эксплуатационной документации

№ п/п	Код документа	Наименование документа
1	2	3
1	Р.КС.09050-XX 34 01	Установка программного комплекса
2*	Р.КС.09050-XX 34 02	Руководство администратора
3	Р.КС.09050-XX 34 03	Руководство пользователя
* настоящее руководство		

Условные обозначения

В документе используются следующие условные обозначения:

Уведомление _ Важные сведения о влиянии текущих действий пользователя на выполнение других функций, задач программного комплекса.

Предупреждение _ Важные сведения о возможных негативных последствиях действий пользователя.

Предостережение _ Критически важные сведения, пренебрежение которыми может привести к ошибкам.

Замечание _ Полезные дополнительные сведения, советы, общеизвестные факты и выводы.

[Выполнить] _ Функциональные экранные кнопки.

<F1> _ Клавиши клавиатуры.

«Чек» _ Наименования объектов обработки (режимов).

Статус _ Названия элементов пользовательского интерфейса.

ОКНА => НАВИГАТОР _ Навигация по пунктам меню и режимам.

п. 2.1.1
рисунок 5 _ Ссылки на структурные элементы, рисунки, таблицы текущего документа.
[1] _ Ссылки на документы из перечня ссылочных документов.

1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ

1.1. Назначение программного комплекса

СУБД-КС - предназначена для хранения, обработки и анализа данных в качестве сервера СУБД прикладных программных продуктов ООО «Кейсистемс», представляющую собой объектно-реляционную систему управления, реализованную в структуре баз данных и поддерживающей язык SQL запросов.

1.2. Условия применения программного комплекса

Использование в качестве сервера СУБД.

2. ОПИСАНИЕ ОПЕРАЦИЙ

В данном разделе рассмотрим основные операции в рамках комплекса задач пользователя в части администрирования баз данных СУБД-КС.

2.1. Установка СУБД-КС в Linux

Как и любую серверную службу, доступную для внешнего мира, рекомендуется запускать СУБД-КС под отдельной учетной записью пользователя. Эта учетная запись пользователя должна владеть только данными, которыми управляет сервер, и не должна использоваться совместно с другими службами. Не рекомендуется устанавливать исполняемые файлы, принадлежащие этому пользователю, потому что скомпрометированные системы могут затем изменить свои собственные двоичные файлы.

2.1.1. Поддерживаемые дистрибутивы Linux

Двоичные пакеты установки СУБД-КС предоставляются для следующих систем на базе Linux:

- Astra Linux (1.7)
- Debian (10)

2.1.2. Установка и проверка

Инструкции по установке для различных дистрибутивов Linux:

Дистрибутив	Ссылка на инструкцию
Astra Linux 1.7	https://keysystems.ru/files/subd_ks/14.6/docum/RI_09050_Install_SUBD-KS.pdf

2.2. Настройка и эксплуатация сервера

2.2.1. Регистрация учетной записи пользователя

Как и любую серверную службу, доступную для внешнего мира, рекомендуется запускать СУБД-КС под отдельной учетной записью пользователя. Эта учетная запись пользователя должна владеть только данными, которыми управляет сервер, и не должна использоваться совместно с другими службами. Не рекомендуется устанавливать исполняемые файлы, принадлежащие этому пользователю, потому что скомпрометированные системы могут затем изменить свои собственные двоичные файлы.

Чтобы добавить учетную запись пользователя Unix в вашу систему, нужно использовать команду **useradd** или **adduser**.

2.2.2. Создание кластера базы данных

Прежде всего, необходимо инициализировать область хранения базы данных на диске (кластер БД). В стандарте SQL используется термин «кластер». После инициализации кластер БД будет содержать БД с именем postgres, которая является БД по умолчанию для использования утилитами, пользователями и сторонними приложениями. Сам сервер базы данных не требует существования БД, но многие внешние утилиты предполагают, что она существует. Другая БД, созданная в каждом кластере во время инициализации, называется template1, как следует из названия, это будет использоваться в качестве шаблона для впоследствии созданных баз данных; это не должно использоваться для фактической работы.

С точки зрения файловой системы, кластер БД - это один каталог, в котором будут храниться все данные. Это каталог данных или область данных. Размещение каталога в определенном месте:

/var/lib/pgsqlks/data или /var/lib/pgsqlks-xx.x/data , где xx.x это версия СУБД.

Чтобы инициализировать кластер базы данных в свой каталог, используйте команду **initdb**. Желаемое расположение файловой системы вашего кластера базы данных можно указать, например, с помощью опции, приведенной ниже.

```
$ initdb -D /usr/local/pgsqlks/data
```

Обратите внимание, что эту команду необходимо выполнить при входе в учетную запись пользователя СУБД-КС, которая описана в *п. 2.2.1*.

В качестве альтернативы указанной опции допустимо использовать переменную окружения PGDATA.

Также доступен запуск **initdb** с помощью программы **pg_ctl** следующим образом:

```
$ pg_ctl -D /usr/local/pgsqlks/data initdb
```

pg_ctl - команда, используемая для управления экземпляром сервера БД.

Initdb попытается создать указанную вами директорию, если она еще не существует. Для того чтобы создание было успешным, у учетной записи пользователя должны быть права на запись в родительский каталог. Поэтому рекомендуется, чтобы пользователь СУБД-КС владел не только каталогом данных, но и своим родительским каталогом. Если требуемый родительский каталог также не существует, сначала нужно его создать, используя привилегии **root**. Данный процесс может выглядеть так:

```
root # mkdir /usr/local/pgsqlks
root # chown postgres /usr/local/pgsqlks
root # su postgres
postgres $initdb -D /usr/local/pgsqlks/data
```

Если каталог данных существует и уже содержит файлы; **initdb** откажется запускаться. Это предотвращает случайную перезапись существующей установки.

Поскольку каталог данных содержит все данные, хранящиеся в базе данных, важно, чтобы он был защищен от несанкционированного доступа. Поэтому **initdb** отзывает права доступа у всех, кроме пользователя СУБД-КС и, при необходимости, группы. Групповой доступ, если он включен, доступен только для чтения. Это позволяет непривилегированному пользователю в той же группе, что и владелец кластера, создавать резервную копию данных кластера или выполнять другие операции, для которых требуется только чтение.

Обратите внимание, что включение или отключение группового доступа в существующем кластере требует выключения кластера и установки соответствующего режима для всех каталогов и файлов перед перезапуском СУБД-КС. В противном случае в каталоге данных может существовать сочетание режимов. Для кластеров, которые разрешают доступ только владельцу, соответствующие режимы предназначены 0700 для каталогов и 0600 для файлов. Для кластеров, которые также разрешают чтение по группе, соответствующие режимы предназначены 0750 для каталогов и 0640 для файлов.

2.2.2.1. Использование вторичных файловых систем

Многие инсталляторы создают свои кластеры БД в каталогах, отличных от корневого каталога компьютера. Если вы решите сделать это, не рекомендуется использовать самый верхний каталог в качестве каталога БД. Рекомендуется создать папку в каталоге точек монтирования, который принадлежит пользователю СУБД-КС, а затем создать в нем каталог данных. Это позволяет избежать проблем с разрешениями, особенно для таких операций, как pg_upgrade, а также гарантирует чистые сбои, если вторичный том отключен.

2.2.2.2. Использование сетевых файловых систем

Многие инсталляторы создают свои кластеры базы данных в сетевых каталогах. Иногда это делается через NFS или с помощью устройства сетевого хранилища (NAS), которое использует NFS для внутреннего использования. СУБД-КС не делает исключения для файловых систем NFS, то есть предполагает, что NFS ведет себя точно так же, как локально подключенные диски. Если реализация NFS клиента или сервера не обеспечивает стандартную семантику файловой системы, это может вызвать проблемы с надежностью. В частности, отложенные (асинхронные) записи в NFS Сервер могут спровоцировать повреждение данных. Рекомендуется монтировать файловую систему NFS синхронно (без кэширования), чтобы избежать этой опасности. Сети хранения данных (SAN) обычно используют протоколы связи, отличные от NFS, и не подвергаются подобным опасностям. Рекомендуется изучить документацию поставщика в части гарантий согласованности данных. Надежность СУБД-КС напрямую зависит от надежности файловой системы, которую она использует.

2.2.3. Запуск сервера базы данных

Прежде чем предоставить пользователям доступ к БД, необходимо запустить сервер БД. Исполняемый модуль программы называется postgres. Программе требуется указать размещение данных, которые предполагается использовать. Это делается с помощью дополнительной опции. Таким образом, самый простой способ запустить сервер выполнить след команду:

```
$ postgres -D /usr/local/pgsqlks/data
```

Это должно быть сделано при входе в учетную запись пользователя СУБД-КС. Иначе сервер будет пытаться использовать каталог данных, названный переменной среды PGDATA. Если эта переменная также не указана, произойдет сбой.

Обычно лучше стартовать СУБД-КС в фоновом режиме.

```
$ postgres -D /usr/local/pgsqlks/data >logfile 2>&1 &
```

Важно где-то хранить выходные данные сервера stdout и stderr, как показано выше. Это поможет для целей аудита и диагностики проблем.

Программа также имеет ряд других параметров командной строки. Для получения дополнительной информации см. *п. 2.3*.

Запуск можно осуществить также через программу pg_ctl:

```
pg_ctl start -l logfile
```

Сервер будет запущен в фоновом режиме, и поместит вывод в именованный файл журнала. Для остановки сервера СУБД-КС также используются программы postgres и pg_ctl.

Запуск сервера базы данных при загрузке компьютера. Сценарии автозапуска зависят от операционной системы. Примеры можно найти в каталоге contrib/start-scripts. Установка любого из них потребует прав администратора.

Одним из вариантов запуска сервера СУБД при перезагрузке хоста использование утилиты system. Для этого необходимо создать файл сервиса (например, at /etc/systemd/system/pgsqlks.service):

```
[Unit]
Description = СУБД-КС
After=syslog.target
After=network.target

#Documentation = man: Postgres (1)
[Service]
Type=notify
User=postgres
ExecStart=/opt/pgsqlks/bin/postgres -D /var/lib/pgsqlks/data
ExecReload=/usr/bin/kill -HUP $ MAINPID
KillMode=mixed
KillSignal=SIGINT
TimeoutSec = 300
[Install]
WantedBy = multi-user.target
```

Пока сервер работает, его идентификатор процесс (pid) хранится в файле postmaster.pid в каталоге данных. Это используется для предотвращения запуска нескольких экземпляров сервера в одном каталоге данных, а также может использоваться для выключения сервера.

2.2.3.1. Сбои при запуске сервера

Существует несколько распространенных причин, по которым сервер может не запуститься. Проверьте файл журнала сервера или запустите его вручную (без перенаправления стандартного вывода или стандартной ошибки) и посмотрите, какие сообщения об ошибках появляются. Ниже более подробно описаны некоторые из наиболее распространенных сообщений об ошибках.

```
LOG: could not bind IPv4 address "127.0.0.1": Address already in use
HINT: Is another postmaster already running on port 5432? If not, wait a few
seconds and retry.
FATAL: could not create any TCP/IP sockets
```

Обычно это означает только то, что вы попытались запустить другой сервер на том же порту, на котором он уже запущен. Однако если сообщение об ошибке не появляется с сообщением о том, что адрес уже используется, возможна другая проблема.

2.2.3.2. Проблемы с клиентским подключением

Возможные ошибки на стороне клиента довольно разнообразны и зависят от приложения, но некоторые из них могут быть напрямую связаны с тем, как был запущен сервер. Условия, отличные от указанных ниже, должны быть задокументированы в соответствующем клиентском приложении.

```
psql: could not connect to server: Connection refused
 Is the server running on host "server.joe.com" and accepting
 TCP/IP connections on port 5432?
```

Это общая ошибка «Не найден сервер». Возникает при неудачной попытке соединения TCP / IP. Способ решения - требуется настроить сервер для разрешения соединений TCP / IP.

2.2.4. Завершение работы сервера

Есть несколько способов выключить сервер БД. Вы можете управлять типом отключения, указывая необходимые сигналы процессу postgres.

SIGTERM

Это режим Smart Shutdown. После получения команды SIGTERM сервер запрещает новые подключения, но позволяет существующим сеансам нормально завершать свою работу. Он отключается только после завершения всех сеансов. Если сервер находится в режиме резервного копирования в онлайн-хранилище, он дополнительно ожидает, пока режим резервного копирования в онлайн-режиме больше не будет активным. При активном режиме резервного копирования новые подключения будут по-прежнему разрешены, но только для администраторов (это исключение позволяет администратору подключаться для завершения режима онлайн-резервного копирования). Если сервер находится в состоянии восстановления, когда запрашивается интеллектуальное завершение работы, восстановление и потоковая репликация будут остановлены только после завершения всех обычных сеансов.

SIGINT

Это режим быстрого отключения. Сервер запрещает новые подключения и отправляет все существующие серверные процессы SIGTERM, что заставит их прервать свои текущие транзакции и быстро завершить работу. Затем он ожидает завершения всех серверных процессов и, наконец, завершает работу. Если сервер находится в режиме резервного копирования в режиме онлайн, режим резервного копирования будет прерван.

SIGQUIT

Это режим немедленного отключения. Сервер отправит SIGQUIT всем дочерним процессам и будет ждать их завершения. Если они не прекратятся в течение 5 секунд, они будут отправлены SIGKILL. Процесс главного сервера завершается, как только завершаются все дочерние процессы, без выполнения обычной обработки завершения работы базы данных. Это приведет к восстановлению (путем воспроизведения журнала WAL) при следующем запуске. Способ рекомендуется только в чрезвычайных ситуациях.

Программа pg_ctl предоставляет удобный интерфейс для отправки сигналов на завершение работы сервера. Кроме того, вы можете отправить сигнал напрямую, используя системы, отличные от Windows. PID в postgres-процессе можно найти с помощью ps-программы, или из файла postmaster.pid в каталоге данных. Например, чтобы сделать быстрое отключение:

```
$ kill -INT `head -1 /usr/local/pgsqlks/data/postmaster.pid`
```

2.3. Конфигурация сервера

2.3.1. Настройка параметров

2.3.1.1. Имена и значения параметров

Все имена параметров нечувствительны к регистру. Каждый параметр принимает значение одного из пяти типов: логическое, строковое, целое, с плавающей запятой или перечислимое (перечисление). Тип определяет синтаксис для установки параметра:

- *Boolean:* Значения могут быть записаны в виде on,off,true,false,yes,no,1,0 (все без учета регистра) или какой-либо однозначный префикс одного из них.
- *String:* в общем случае заключайте значение в одинарные кавычки, удваивая любые одинарные кавычки внутри значения. Однако кавычки обычно можно опустить, если значением является простое число или идентификатор.
- *Numeric (integer and floating point):* десятичная точка разрешена только для параметров с плавающей запятой.
- *Numeric with Unit:* Некоторые числовые параметры имеют неявную единицу измерения, поскольку они описывают количество памяти или времени. Единицей может быть байты, килобайты, блоки (обычно восемь килобайт), миллисекунды, секунды или минуты. Неукрашенное числовое значение для одной из этих настроек будет использовать единицу измерения по умолчанию, которую можно узнать из pg_settings. unit, для удобства настройки могут быть заданы с единицей измерения, указанной явно,

например,'120 ms' для значения времени, и они будут преобразованы в любую фактическую единицу измерения параметра. Обратите внимание, что значение должно быть записано в виде строки (с кавычками), чтобы использовать эту функцию. Имя устройства чувствительно к регистру, и между числовым значением и единицей может быть пробел.

– Допустимые единицы памяти: B(байты), kB(килобайты), MB(мегабайты), GB(гигабайты) и TB(терабайты). Множитель для блоков памяти равен 1024, а не 1000.

– Допустимые единицы времени: ms(миллисекунды), s(секунды), min(минуты), h(часы) и d(дни).

– *Numerated:* параметры перечислимого типа записываются так же, как строковые параметры, но имеют ограниченный набор значений. Значения, допустимые для такого параметра, можно найти из pg_settings. Enumvals. Перечисленные значения параметров не чувствительны к регистру.

2.3.1.2. Взаимодействие параметров через файл конфигурации

Самый фундаментальный способ установить эти параметры - отредактировать файл конфигурации СУБД-КС postgresql.conf, который обычно хранится в каталоге данных. Копия по умолчанию создается при инициализации каталога кластера базы данных. Пример того, как может выглядеть этот файл:

```
# This is a comment
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 незначительны (кроме как в значении параметра в кавычках), а пустые строки игнорируются. Хеш-метки (#) обозначают оставшуюся часть строки как комментарий. Значения параметров, которые не являются простыми идентификаторами или числами, должны быть заключены в одинарные кавычки. Чтобы вставить одинарную кавычку в значение параметра, напишите две кавычки (предпочтительно) или обратную косую черту.

Параметры, установленные таким образом, предоставляют значения по умолчанию для кластера. Настройки, видимые активными сеансами, будут этими значениями, если они 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 его pg_ctl reload из командной строки или вызвав функцию SQL pg_reload_conf(). Процесс основного сервера также передает этот сигнал всем запущенным в данный момент процессам сервера, так что существующие сеансы также принимают новые значения (это произойдет после того, как они завершат выполнение любой выполняемой в данный момент клиентской команды). В качестве альтернативы можно отправить сигнал на один серверный процесс напрямую. Некоторые параметры могут быть установлены только при запуске сервера; любые изменения их записей в файле конфигурации будут игнорироваться до перезапуска сервера. Неверные настройки параметров в файле конфигурации также игнорируются (но регистрируются) во время обработки SIGHUP.

СУБД-КС каталог данных содержит файл postgresql.auto.conf, который имеет тот же формат, postgresql.conf, но никогда не должен редактироваться вручную. Этот файл содержит настройки, предоставленные с помощью команды ALTER SYSTEM.

Системное представление pg_file_settings может быть полезно для предварительного тестирования изменений в файле конфигурации или для диагностики проблем, если сигнал SIGHUP не дал желаемых результатов.

2.3.1.3. Взаимодействие с параметрами через SQL

СУБД-КС предоставляет три команды SQL для установки параметров конфигурации по умолчанию. Команда ALTER SYSTEM предоставляет доступные для SQL средства изменения глобальных значений по умолчанию; это функционально эквивалентно редактированию pgsqlks.conf. Кроме того, есть две команды, которые позволяют устанавливать значения по умолчанию для каждой базы данных или для каждой роли:

Команда ALTER DATABASE позволяет переопределять глобальные параметры для каждой базы данных.

Команда ALTER ROLE позволяет как глобальным настройкам, так и настройкам для каждой базы данных быть переопределенными пользовательскими значениями.

Значения, установленные с 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 составляют значения по умолчанию для оставшейся части сеанса. Обратите внимание, что некоторые настройки нельзя изменить после запуска сервера, поэтому их нельзя установить с помощью этих команд (или перечисленных ниже).

После подключения клиента к базе данных СУБД-КС предоставляет две дополнительные команды SQL (и эквивалентные функции) для взаимодействия с локальными настройками конфигурации сеанса.

Команда SHOW позволяет проверить текущее значение всех параметров. Соответствующая функция называется 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 есть set_config(setting_name, new_value, is_local).

Кроме того, системное представление pg_settings может использоваться для просмотра и изменения локальных значений сеанса:

Запрос аналогичен использованию SHOW ALL, но предоставляет больше возможностей, так как можно задавать условия фильтрации или объединяться с другими отношениями.

Использование UPDATE в этом представлении, в частности обновление setting столбца, является эквивалентом выдачи SET-команд. Например, эквивалент

```
SET configuration_parameter TO DEFAULT;
```

является таким:

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

2.3.1.4. Управление содержимым файла конфигурации

Основной файл конфигурации СУБД-КС содержит в себе директиву include с указание на каталог include_dir = 'conf.d' с единственным подфайлом конфигурации pgsqlks.conf. Настоятельно рекомендуется устанавливать дополнительные настройки только в этом подфайле конфигурации или создавать новые файлы конфигурации в каталоге conf.d.

Неабсолютные имена каталогов берутся относительно каталога, содержащего основной файл конфигурации postgresql.conf. Несколько файлов в каталоге include обрабатываются в порядке имен файлов (в соответствии с правилами языка C, т.е. с цифрами перед буквами и заглавными буквами перед строчными).

2.3.2. Соединение и аутентификация

2.3.2.1. Настройки соединения

Перечислим основные настройки.

```
listen_addresses(string)
```

Указывает адрес (а) TCP / IP, на котором (ых) сервер должен прослушивать соединения от клиентских приложений. Значение принимает 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 и IPv6. Если список пуст, сервер вообще не прослушивает ни один IP-интерфейс, и в этом случае для подключения к нему можно использовать только сокеты Unix. Значением по умолчанию является *, что соответствует всем доступным IP-интерфесам.

```
port(integer)
```

TCP-порт, который прослушивает сервер; по умолчанию имеет значение 5434. Обратите внимание, что один и тот же номер порта используется для всех IP-адресов, которые прослушивает сервер. Этот параметр можно установить только при запуске сервера.

```
max_connections(integer)
```

Определяет максимальное количество одновременных подключений к серверу базы данных. По умолчанию обычно используется 100 соединений. Этот параметр можно установить только при запуске сервера.

```
tcp_keepalives_idle(integer)
```

Указывает количество секунд бездействия, по истечении которых TCP должен отправить клиенту сообщение поддержки активности. Значение «0» использует системное значение по умолчанию. Этот параметр поддерживается только в системах, которые поддерживают TCP_KEEPINTVL.

```
tcp_keepalives_interval(integer)
```

Задает количество секунд, по истечении которых сообщение подтверждения активности которое не было подтверждено клиентом, должно быть передано повторно. Значение 0 использует системное значение по умолчанию. Этот параметр поддерживается только в системах, которые поддерживают TCP_KEEPINTVL.

```
tcp_keepalives_count(integer)
```

Задает количество сообщений поддержки активности TCP, которые могут быть потеряны до того, как соединение сервера с клиентом будет считаться прерванным. Значение 0 использует системное значение по умолчанию. Этот параметр поддерживается только в системах, которые поддерживают TCP_KEEPCNT.

2.3.2.2. Аутентификация

```
authentication_timeout(integer)
```

Максимальное время для завершения аутентификации клиента, в секундах. Если потенциальный клиент не завершил протокол аутентификации за это время, сервер закрывает соединение. Это не позволяет зависшим клиентам занимать соединение бесконечно. Значение по умолчанию - одна минута (1m). Этот параметр можно установить только в файле конфигурации или в командной строке сервера.

```
password_encryption(enum)
```

Если в CREATE ROLE или ALTER ROLE указан пароль, этот параметр определяет алгоритм, который будет использоваться для шифрования пароля. Значение по умолчанию: md5 - пароль хранится в виде хэша MD5 (также принимается как псевдоним для md5).

2.3.3. Потребление ресурсов

2.3.3.1. Память

```
shared_buffers(integer)
```

Устанавливает объем памяти, который сервер базы данных использует для буферов общей памяти. Значение по умолчанию обычно составляет 128 мегабайт (128MB), но может быть

меньше, если настройки ядра не будут его поддерживать (как определено во время initdb). Этот параметр должен быть не менее 128 килобайт. Для хорошей производительности обычно требуются настройки, значительно превышающие минимум. Этот параметр можно установить только при запуске сервера.

Если у вас есть выделенный сервер базы данных с 1 ГБ или более ОЗУ, разумным начальным значением для shared_buffers вашей системы будет 25% памяти. Существуют некоторые рабочие нагрузки, в которых эффективны даже большие параметры shared_buffers, но поскольку СУБД-КС также опирается на кэш операционной системы, маловероятно, что выделение более 40% оперативной памяти shared_buffersбудет работать лучше, чем меньшее. Большие настройки shared_buffers обычно требуют соответствующего увеличения max_wal_size, чтобы распространить процесс записи большого количества новых или измененных данных на более длительный период времени.

В системах с менее чем 1 ГБ ОЗУ необходим меньший процент ОЗУ, чтобы оставить достаточно места для операционной системы.

```
temp_buffers(integer)
```

Устанавливает максимальное количество временных буферов, используемых каждым сеансом базы данных. Это локальные буферы сеанса, используемые только для доступа к временным таблицам. По умолчанию используется восемь мегабайт (8MB). Параметр может быть изменен в отдельных сеансах, но только перед первым использованием временных таблиц в сеансе; последующие попытки изменить значение не будут влиять на этот сеанс.

Сессия будет распределять временные буферы по мере необходимости до предела, заданного temp_buffers. Стоимость установки большого значения в сеансах, которым на самом деле не нужно много временных буферов, составляет всего лишь дескриптор буфера, или около 64 байтов, на приращение temp_buffers. Однако если буфер фактически используется, для него будут использованы дополнительные 8192 байта.

```
max_prepared_transactions(integer)
```

Устанавливает максимальное количество транзакций, которые могут одновременно находиться в «подготовленном» состоянии. Установка этого параметра в ноль (по умолчанию) отключает функцию подготовленной транзакции. Этот параметр можно установить только при запуске сервера.

Если вы не планируете использовать подготовленные транзакции, этот параметр следует установить на ноль, чтобы предотвратить случайное создание подготовленных транзакций. Если вы используете подготовленные транзакции max_prepared_transactions, вы, вероятно, захотите иметь размер не менее max_connections, чтобы у каждого сеанса могла быть подготовленная транзакция в ожидании.

При запуске резервного сервера вы должны установить для этого параметра то же или более высокое значение, чем на главном сервере. В противном случае запросы не будут разрешены на резервном сервере.

```
work_mem(integer)
```

Задает объем памяти, который будет использоваться внутренними операциями сортировки и хэш-таблицами перед записью во временные файлы на диске. Значение по умолчанию составляет четыре мегабайта (4MB). Обратите внимание, что для сложного запроса несколько операций сортировки или хеширования могут выполняться параллельно; каждой операции будет разрешено использовать столько памяти, сколько указано в этом значении, прежде чем она начнет записывать данные во временные файлы. Кроме того, несколько запущенных сеансов могут выполнять такие операции одновременно. Следовательно, общая используемая память может во много раз превышать значение work_mem; этот факт необходимо учитывать при выборе значения. Операции сортировки используются для ORDER BY, DISTINCT и слияние объединений. Хеш-таблицы используются в хеш-соединениях, агрегации на основе хеша и обработке IN подзапросов на основе хеша.

```
maintenance_work_mem(integer)
```

Задает максимальный объем памяти для использования операций по техническому обслуживанию, таких как VACUUM, CREATE INDEX и ALTER TABLE ADD FOREIGN KEY. По умолчанию это 64 мегабайта (64MB). Поскольку только одна из этих операций может быть выполнена за один раз сеансом базы данных, и у установки обычно нет многих из них, работающих одновременно, можно безопасно установить это значение значительно больше, чем work_mem. Большие настройки могут повысить производительность для очистки и восстановления дампов базы данных.

Обратите внимание, что при запуске autovacuum, до времени autovacuum_max_workers на это может быть выделена память, поэтому будьте осторожны, чтобы не установить слишком высокое значение по умолчанию. Это может быть полезно для контроля, отдельно устанавливая autovacuum_work_mem.

```
autovacuum_work_mem(integer)
```

Задает максимальный объем памяти, который будет использоваться каждым рабочим процессом автоочистки. По умолчанию используется значение -1, указывающее, что вместо этого следует использовать значение maintenance_work_mem. Настройка не влияет на поведение VACUUM при запуске в других контекстах.

2.3.3.2. Диск

```
temp_file_limit(integer)
```

Задает максимальный объем дискового пространства, который процесс может использовать для временных файлов, таких как временные файлы сортировки и хэширования, или файл хранилища для удерживаемого курсора. Транзакция, пытающаяся превысить этот лимит, будет отменена. Значение указывается в килобайтах, и -1(по умолчанию) означает отсутствие ограничений. Изменение данного параметра доступно только администратору.

Этот параметр ограничивает общее пространство, используемое в любой момент всеми временными файлами, используемыми данным процессом СУБД-КС. Следует отметить, что

дисковое пространство, используемое для явных временных таблиц, в отличие от временных файлов, используемых за кулисами при выполнении запросов, не учитывается в этом ограничении.

2.3.3.3. Использование ресурсов ядра

```
max_files_per_process(integer)
```

Устанавливает максимальное количество одновременно открытых файлов, разрешенное для каждого подпроцесса сервера. По умолчанию это тысяча файлов. Если ядро применяет безопасный лимит для каждого процесса, вам не нужно беспокоиться об этом параметре. Но на некоторых платформах (особенно в большинстве систем BSD) ядро позволяет отдельным процессам открывать гораздо больше файлов, чем может реально поддерживать система, если все процессы пытаются открыть такое количество файлов. Если вы обнаружите, что видите слишком много открытых файлов, попробуйте уменьшить этот параметр. Этот параметр можно установить только при запуске сервера.

2.3.3.4. Вакуумная задержка на основе стоимости

Во время выполнения команд VACUUM и ANALYZE система поддерживает внутренний счетчик, который отслеживает оценочную стоимость различных выполняемых операций ввода/вывода. Когда накопленная стоимость достигает предела (указанного с помощью vacuum_cost_limit), процесс, выполняющий операцию, будет находиться в режиме ожидания в течение короткого периода времени, как указано в vacuum_cost_delay. Затем он сбросит счетчик и продолжит выполнение.

Цель этой функции - позволить администраторам снизить влияние ввода/вывода этих команд на одновременную работу базы данных. Есть много ситуаций, когда важно, чтобы команды технического обслуживания, как VACUUM и ANALYZE отработали быстро, без оказания значительного влияния на способность системы выполнять другие операции с базой данных. Задержка вакуума, основанная на затратах, позволяет администраторам достичь этого.

Эта функция по умолчанию отключена для VACUUM команд, введенных вручную. Чтобы включить его, установите для vacuum_cost_delay переменной ненулевое значение.

```
vacuum_cost_delay(integer)
```

Интервал времени в миллисекундах, в течение которого процесс будет находиться в спящем режиме после превышения предела стоимости. Значение по умолчанию равно нулю. Обратите внимание, что во многих системах эффективное разрешение задержек спящего режима составляет 10 миллисекунд; установка vacuum_cost_delay значения, не кратного 10, может иметь те же результаты, что и установка следующего значения, кратного 10.

```
vacuum_cost_page_hit(integer)
```

Ориентировочная стоимость очистки буфера в общем буферном кеше. Он представляет собой стоимость блокировки пула буферов, поиска общей хеш-таблицы и сканирования содержимого страницы. Значением по умолчанию является один.

```
vacuum_cost_page_miss(integer)
```

Ориентировочная стоимость очистки буфера, который должен быть прочитан с диска. Это представляет собой попытку заблокировать пул буферов, найти общую хеш-таблицу, прочитать нужный блок с диска и просканировать его содержимое. Значение по умолчанию 10.

```
vacuum_cost_page_dirty(integer)
```

Ориентировочная стоимость взимается, когда очистка модифицирует блок, который был ранее чистым. Он представляет собой дополнительный ввод / вывод, необходимый для повторной очистки грязного блока на диске. Значением по умолчанию является 20.

```
vacuum_cost_limit(integer)
```

Накопленная стоимость, которая заставит процесс засыпать. Значение по умолчанию составляет 200.

2.3.3.5. Write Ahead Log

Существует отдельный серверный процесс, называемый фоновым модулем записи WAL (фоновый писатель), функция которого заключается в записи «грязных» (новых или измененных) общих буферов на диск. Он записывает общие буферы так, чтобы серверным процессам, которые обрабатывают пользовательские запросы, не приходилось ждать, пока произойдет запись или чтобы это ожидание было минимальным. Тем не менее, средство записи в фоновом режиме вызывает общее увеличение нагрузки ввода/вывода, так как запись производится каждый раз, когда меняется одна и та же страница, хотя ее изменение можно было записать один раз в момент контрольной точки. Параметры, обсуждаемые в этом подразделе, могут использоваться для настройки поведения для конкретных ситуаций.

```
bgwriter_delay(integer)
```

Определяет задержку между раундами активности для фонового писателя. В каждом раунде модуль записи выдает записи для некоторого количества «грязных» буферов (управляемых следующими параметрами). Затем bgwriter_delay спит в течение миллисекунд и повторяется. Когда в пуле буферов нет «грязных» буферов, он переходит в более длительный режим ожидания, независимо от bgwriter_delay. Значение по умолчанию составляет 200 миллисекунд (200ms). Во многих системах эффективное разрешение задержек сна составляет 10 миллисекунд; установка bgwriter_delay значения, не кратного 10, может иметь те же результаты, что и установка следующего значения, кратного 10. Этот параметр может быть установлен только в файле конфигурации или в командной строке сервера.

```
bgwriter_lru_maxpages(integer)
```

В каждом раунде фоновый писатель будет записывать не более этого количества буферов. Установка этого значения в ноль отключает фоновую запись (обратите внимание, что контрольные точки, которые управляются отдельным выделенным вспомогательным процессом, не затрагиваются). Значение по умолчанию - 100 буферов. Этот параметр может быть установлен только в файле конфигурации или в командной строке сервера.

```
bgwriter_lru_multiplier(floating point)
```

Количество «грязных» буферов, записанных в каждом раунде, основано на количестве новых буферов, которые были необходимы серверным процессам во время последних раундов. Средняя потребность за последнее время умножается bgwriter_lru_multiplier на получение оценки количества буферов, которые понадобятся в следующем раунде. «Грязные» буферы записываются до тех пор, пока не появится необходимое количество «чистых», многоразовых буферов (тем не менее, bgwriter_lru_maxpages за один раунд запишет не больше буферов). Таким образом, настройка 1.0 представляет политику «точно в срок», позволяющую записать именно то количество буферов, которое, по прогнозам, потребуется. Большие значения обеспечивают некоторую защиту от всплесков спроса, в то время как меньшие значения преднамеренно оставляют записи для серверных процессов. По умолчанию это 2.0. Этот параметр может быть установлен только в файле конфигурации или в командной строке сервера.

```
bgwriter_flush_after(integer)
```

Всякий раз, когда фоновый модуль записи записывает больше чем bgwriter_flush_after байтов, сервер заставляет ОС поместить эти записи в основное хранилище. Это ограничит количество «грязных» данных в кэше страниц ядра, уменьшая вероятность зависаний при fsync в конце контрольной точки, или когда ОС записывает данные большими пакетами в фоновом режиме. Зачастую это приводит к значительному снижению задержки транзакций, но также есть некоторые случаи, особенно с рабочими нагрузками, которые больше, чем shared_buffers, но меньше, чем кэш страниц ОС, где производительность может снизиться. Этот параметр может не повлиять на некоторые платформы. Параметр можно установить только в файле конфигурации СУБД-КС или в командной строке сервера.

Меньшие значения bgwriter_lru_maxpages и bgwriter_lru_multiplier позволяют уменьшить дополнительную нагрузку ввода / вывода, вызванные фоновым модулем записи, но серверные процессы будут получать приоритет, задерживая интерактивные запросы.

2.3.3.6. Асинхронное поведение

```
effective_io_concurrency(integer)
```

Устанавливает количество одновременных операций дискового ввода/вывода, которые, как ожидает СУБД-КС, могут выполняться одновременно. Увеличение этого значения приведет к увеличению числа операций ввода/вывода, которые каждый отдельный сеанс СУБД-КС пытается инициировать параллельно. Допустимый диапазон: от 1 до 1000 или ноль для отключения выдачи асинхронных запросов ввода/вывода. В настоящее время этот параметр влияет только на сканирование по битовой карте.

Для магнитных дисков хорошей отправной точкой для этого параметра является количество отдельных дисков, содержащих массив RAID 0 или зеркало RAID 1, используемое для базы данных. Однако если база данных часто занята несколькими запросами, выполняемыми в параллельных сеансах, более низкие значения параметра могут привести к полной загрузке дискового массива. Значение выше, чем необходимо, при полной загрузке дисков, приведет только к дополнительной загрузке ЦП. SSD-накопители и другие хранилища на основе памяти часто могут обрабатывать много одновременных запросов, поэтому оптимальным значением является несколько сотен.

Асинхронный ввод-вывод зависит от эффективной posix_fadvise функции, которой нет в некоторых операционных системах. Если функция отсутствует, то установка этого параметра в любое значение, кроме нуля, приведет к ошибке.

В поддерживаемых системах по умолчанию установлено значение 1, в противном случае - 0.

```
max_worker_processes(integer)
```

Устанавливает максимальное количество фоновых процессов, которые может поддерживать система. Этот параметр можно установить только при запуске сервера. По умолчанию это 8.

При запуске резервного сервера вы должны установить для этого параметра то же или более высокое значение, чем на главном сервере. В противном случае запросы не будут разрешены на резервном сервере.

Изменяя это значение, рассмотрите также настройку max_parallel_workers, max_parallel_maintenance_workers и max_parallel_workers_per_gather.

```
max_parallel_workers_per_gather(integer)
```

Устанавливает максимальное количество подключений, которое может быть запущено одним узлом Gather или Gather Merge. Параллельные рабочие процессы берутся из пула процессов, установленных max_worker_processes, ограниченного max_parallel_workers. Обратите внимание, что запрошенное количество рабочих процессов может быть недоступно во время выполнения. Если это произойдет, план будет работать с меньшим количеством процессов, чем ожидалось, что может быть неэффективным. Значение по умолчанию - 2. Установка этого значения в 0 отключает параллельное выполнение запроса.

Обратите внимание, что параллельные запросы могут потреблять значительно больше ресурсов, чем непараллельные запросы, поскольку каждый рабочий процесс - это совершенно отдельный процесс, который оказывает примерно то же влияние на систему, что и дополнительный сеанс пользователя. Это следует учитывать при выборе значения для этого параметра, а также при настройке других параметров, управляющих использованием ресурсов, таких как work_mem. Лимиты ресурсов work_mem, которые применяются индивидуально для каждого процесса, означают, что общее использование может быть намного выше для всех процессов, чем обычно для любого отдельного процесса.

```
max_parallel_maintenance_workers(integer)
```

Устанавливает максимальное количество параллельных рабочих процессов, которые могут быть запущены одной служебной командой. В настоящее время единственная служебная команда, которая поддерживает использование параллельных рабочих, есть CREATE INDEX и только при построении индекса B-дерева. Значение по умолчанию - 2. Установка этого значения в 0 отключает использование параллельных рабочих процессов служебными командами.

Обратите внимание, что параллельные служебные команды не должны потреблять больше памяти, чем эквивалентные непараллельные операции. Эта стратегия отличается от стратегии параллельного запроса, где ограничения ресурсов обычно применяются для каждого рабочего процесса. Параллельные служебные команды рассматривают ограничение ресурса maintenance_work_mem как ограничение, которое применяется ко всей служебной команде независимо от количества параллельных рабочих процессов. Однако параллельные служебные команды могут по-прежнему потреблять значительно больше ресурсов ЦП и пропускной способности ввода/вывода.

```
max_parallel_workers(integer)
```

Устанавливает максимальное количество рабочих процессов, которое система может поддерживать для параллельных операций. Значение по умолчанию - 8. При увеличении или уменьшении этого значения также следует рассмотреть возможность настройки max_parallel_maintenance_workers и max_parallel_workers_per_gather.

```
backend_flush_after(integer)
```

При выполнении backend_flush_after желательно настроить, размещение записей в основном хранилище. Это ограничит количество «грязных» данных в кэше страниц ядра, уменьшая вероятность зависаний, когда ОС записывает данные большими пакетами в фоновом режиме. Зачастую это приводит к значительному снижению задержки транзакций, но также есть некоторые случаи, особенно с рабочими нагрузками, которые больше, чем shared_buffers, но меньше, чем кэш страниц ОС, где производительность может снизиться. Допустимый диапазон, который отключает принудительную обратную запись, находится между 0 и 2MB. По умолчанию это 0 не принудительная обратная запись. (Если BLCKSZ не 8 КБ, максимальное значение масштабируется пропорционально ему.)

```
old_snapshot_threshold(integer)
```

Задает минимальное время, в течение которого снимок можно использовать без риска возникновения ошибки snapshot too old при использовании снимка. Этот параметр можно установить только при запуске сервера.

Чтобы предотвратить неверные результаты из-за очистки данных, генерируется ошибка, когда снимок сделан раньше достижения порога переполнения данных, и снимок используется для чтения страницы, которая была изменена с момента создания снимка.

Значение -1отключает эту функцию и является значением по умолчанию. Полезные значения для быстродействия, могут варьироваться от небольшого количества часов до нескольких дней. Настройка будет приведена к степени детализации минут, а небольшие числа

(например, 0 или 1min) допустимы только потому, что иногда они могут быть полезны для тестирования.

2.3.4. Журнал опережающей записи

2.3.4.1. Настройки

```
wal_level(enum)
```

wal_level определяет, сколько информации записывается в WAL. Значение по умолчанию – replica это запись достаточного количества данных для поддержки архивации и репликации WAL, включая выполнение запросов только для чтения на резервном сервере. Minimal удаляет все журналы, кроме информации, необходимой для восстановления после сбоя или немедленного выключения. Наконец, logical добавляет информацию, необходимую для поддержки логического декодирования. Каждый уровень включает информацию, зарегистрированную на всех более низких уровнях. Этот параметр можно установить только при запуске сервера.

На minimal уровне, WAL-протоколирование некоторых массовых операций может быть безопасно пропущено, что может значительно ускорить операции. Операции, в которых может применяться эта оптимизация, включают в себя:

- CREATE TABLE AS
- CREATE INDEX
- CLUSTER
- COPY (в таблицы, которые были созданы или усечены в одной транзакции)

Но минимальный WAL не содержит достаточно информации для восстановления данных из базовой резервной копии и журналов WAL, поэтому replica для включения архивации WAL (archive_mode) и потоковой репликации необходимо использовать более высокий уровень.

На logical уровне регистрируется та же информация, что и при добавлении информации replica, необходимой для извлечения наборов логических изменений из WAL. Использование уровня logical увеличит объем WAL, особенно если много таблиц настроены на REPLICA IDENTITY FULL, и многие инструкции UPDATE и DELETE выполняются.

```
fsync(boolean)
```

Если этот параметр включен, сервер СУБД-КС будет пытаться убедиться, что обновления физически записываются на диск, выполняя системные вызовы fsync() или различные эквивалентные методы. Это гарантирует, что кластер базы данных сможет вернуться в согласованное состояние после сбоя операционной системы или оборудования.

Хотя отключение fsync часто приводит к повышению производительности, это может привести к необратимому повреждению данных в случае сбоя питания или сбоя системы. Таким образом, рекомендуется отключить его только в том случае, если вы можете легко воссоздать всю базу данных из внешних данных.

Примеры безопасных обстоятельств для отключения fsync включают начальную загрузку нового кластера базы данных из файла резервной копии, использование кластера базы данных для

обработки пакета данных, после которого база данных будет выброшена и воссоздана, или для клона базы данных «только для чтения», который часто воссоздается и не используется для отработки отказа. Одно только высококачественное оборудование не является достаточным основанием для отключения fsync.

Для надежного восстановления при включении fsync необходимо принудительно установить все измененные буферы в ядре в долговременное хранилище. Это можно сделать, когда кластер fsync выключен или включен, запустив initdb --sync-only, запустив sync, размонтировав файловую систему или перезагрузив сервер.

Во многих ситуациях отключение synchronous_commit для некритических транзакций может обеспечить большую часть потенциального выигрыша в производительности при отключении fsync без сопутствующего риска повреждения данных.

Fsync может быть установлен только в файле конфигурации или в командной строке сервера. Если вы отключите этот параметр, также рассмотрите возможность отключения full_page_writes.

`synchronous_commit(enum)`

Указывает, будет ли фиксация транзакции ожидать записи WAL на диск, прежде чем команда вернет клиенту указание «успех». Допустимые значения on, remote_apply, remote_write, local, и off. По умолчанию установлена безопасная настройка on. Когда настройка в положении off, возможна задержка между тем, когда клиенту сообщается об успехе, и, когда транзакция действительно гарантированно защищена от сбоя сервера (максимальная задержка в три раза больше wal_writer_delay). В отличие от fsync, установите для этого параметра значение off – это не создает риска несогласованности базы данных: сбой операционной системы или базы данных может привести к потере некоторых недавних совершенных транзакций, но состояние базы данных будет таким же, как если бы эти транзакции были аккуратно прерваны. Таким образом, отключение synchronous_commit может быть полезной альтернативой, когда производительность важнее, чем точная уверенность в долговечности транзакции.

Если synchronous_standby_names не является пустым, этот параметр также контролирует, будет ли фиксация транзакции ожидать репликацию своих записей WAL на резервный сервер. Когда установлено значение on, коммиты будут ждать, пока ответы текущего синхронного режима ожидания не укажут, что они получили запись фиксации транзакции, а также сбросили ее на диск. Это гарантирует, что транзакция не будет потеряна, если как основной, так и все синхронные резервные серверы не повредят хранилище своей базы данных. Когда установлено значение remote_apply, коммиты будут ждать, пока ответы из текущих синхронных резервов не укажут, что они получили запись фиксации транзакции и применили ее, так что она стала видимой для запросов в резервах. Когда установлена настройка remote_write, коммиты будут ждать до тех пор, пока ответы из текущего синхронного резерва не укажут, что они получили запись фиксации транзакции и записали ее в свою операционную систему. Этот параметр достаточен для обеспечения сохранения данных даже в случае сбоя резервного экземпляра СУБД-КС, но не в случае сбоя в режиме ожидания на уровне операционной системы, поскольку данные не достигли стабильного хранилища в режиме ожидания. Наконец, параметр local заставляет коммиты ждать локального сброса на диск, но не репликации. Синхронная репликация предоставляется для полноты.

Если synchronous_standby_names пусто, то настройки on, remote_apply, remote_write и local все обеспечивают одинаковый уровень синхронизации: при фиксировании транзакции ждать только сохранение данных на диск.

Этот параметр может быть изменен в любое время; Поведение любой транзакции определяется настройкой, действующей при ее фиксации. Поэтому возможно и полезно, чтобы некоторые транзакции выполнялись синхронно, а другие - асинхронно.

```
wal_sync_method(enum)
```

Метод, используемый для принудительного обновления WAL на диск. Если параметр fsync выключен, то этот параметр не имеет значения, поскольку обновления файла WAL не будут принудительно отменены. Возможные значения:

```
open_datasync(сохранение файлов WAL открывать их функцией open()с опцией O_DSYNC)
```

- open_datasync (сохранение файлов WAL открывать их фугкцией open() с опцией O_DSYNC)
- fdatasync (вызывать fdatasync() при каждом фиксировании)
- fsync (вызывать fsync() при каждом фиксировании)
- fsync_writethrough (вызывать fsync() при каждом фиксировании, форсируя write-through of any disk запись кэша)
- open_sync (сохранение файлов WAL открывать функцией open() с опцией O_SYNC)

В open_* опции также допустимо использовать O_DIRECT. По умолчанию используется первый метод в приведенном выше списке. Этот параметр можно установить только в файле конфигурации или в командной строке сервера.

```
full_page_writes(boolean)
```

Когда этот параметр включен, СУБД-КС сервер записывает все содержимое каждой страницы диска в WAL во время первой модификации этой страницы после контрольной точки. Это необходимо, потому что запись страницы, которая выполняется во время сбоя операционной системы, может быть завершена только частично, что приводит к несогласованным данным в странице на диске, которая содержит смесь старых и новых данных. Данные об изменениях на уровне строк, обычно хранящиеся в WAL, не будут достаточны для полного восстановления такой страницы во время восстановления после сбоя. Хранение полного изображения страницы гарантирует, что страница может быть восстановлена правильно, но за счет увеличения объема данных, которые должны быть записаны в WAL. Поскольку воспроизведение WAL всегда начинается с контрольной точки, это достаточно сделать при первом изменении каждой страницы после контрольной точки. Поэтому одним из способов снижения стоимости полностраничных записей является увеличение параметров интервала контрольной точки.

Отключение этого параметра ускоряет работу, но может привести к неисправимому повреждению данных без вывода сообщений после сбоя системы. Риски аналогичны отключению fsync.

Отключение этого параметра не влияет на использование архивации WAL для восстановления на определенный момент времени (PITR).

Этот параметр можно установить только в файле конфигурации СУБД-КС или в командной строке сервера. По умолчанию имеет значение on.

```
wal_log_hints(boolean)
```

Когда этот параметр имеет значение on, сервер СУБД-КС записывает все содержимое каждой страницы диска в WAL во время первой модификации этой страницы после контрольной точки, даже для некритических модификаций так называемых битов подсказок.

Если контрольные суммы данных включены, обновления битов подсказок всегда регистрируются в WAL, и этот параметр игнорируется. Вы можете использовать этот параметр, чтобы проверить, сколько дополнительных журналов WAL произойдет, если в вашей базе данных будут включены контрольные суммы данных.

Этот параметр можно установить только при запуске сервера. Значением по умолчанию является off.

```
wal_compression(boolean)
```

Когда этот параметр имеет значение on, сервер СУБД-КС сжимает полный образ страницы, записанный в WAL, когда full_page_writes включен или во время основного резервного копирования. Сжатый образ страницы будет распакован во время воспроизведения WAL. Значением по умолчанию является off. Только администратор может изменить этот параметр.

Включение этого параметра может уменьшить объем WAL без увеличения риска неустранимого повреждения данных, но за счет некоторого дополнительного ресурса ЦП, затрачиваемого на сжатие во время ведения журнала WAL и на декомпрессию во время воспроизведения WAL.

```
wal_buffers(integer)
```

Объем общей памяти, используемой для данных WAL, которые еще не были записаны на диск. Значение по умолчанию, равное -1, выбирает размер, равный 1/32 (около 3%) общих буферов, но 64kB обычно не меньше и не больше, чем размер одного сегмента WAL 16MB. Это значение может быть установлено вручную, но любое положительное значение меньше, чем 32kB будет рассматриваться как 32kB. Этот параметр можно установить только при запуске сервера.

Содержимое буферов WAL записывается на диск при каждой фиксации транзакции, поэтому чрезвычайно большие значения вряд ли обеспечат значительное преимущество. Однако установка этого значения, по крайней мере, в несколько мегабайт может улучшить производительность записи на занятом сервере, где одновременно фиксируются многие клиенты. Автонастройка, по умолчанию выбранная равной -1, должна давать приемлемые результаты в большинстве случаев.

```
wal_writer_delay(integer)
```

Определяет, как часто модуль записи WAL сбрасывает WAL. После сброса WAL wal_writer_delay спит в течение миллисекунд, если только он не проснулся от асинхронной фиксации транзакции. Если последний сброс wal_writer_delay произошел менее миллисекунды назад и wal_writer_flush_afterс тех пор было произведено меньше байтов WAL, то WAL записывается только в операционную систему, а не на диск. Значение по умолчанию составляет 200 миллисекунд (200ms). Обратите внимание, что во многих системах эффективное разрешение задержек сна составляет 10 миллисекунд; установка wal_writer_delay значения, не кратного 10, может иметь те же результаты, что и установка следующего значения, кратного 10. Этот параметр может быть установлен только в файле конфигурации СУБД-КС или в командной строке сервера.

```
wal_writer_flush_after(integer)
```

Определяет, как часто фоновый писатель WAL сбрасывает данные на диск. Если последний сброс wal_writer_delay произошел менее миллисекунды назад, и wal_writer_flush_after с тех пор было произведено меньше байтов WAL, то WAL записывается только в операционную систему, а не на диск. Если значение wal_writer_flush_after установлено равным 0, то данные WAL сбрасываются немедленно. По умолчанию это 1MB. Этот параметр можно установить только в файле конфигурации СУБД-КС или в командной строке сервера.

```
commit_delay(integer)
```

commit_delay добавляет задержку, измеряемую в микросекундах, до начала сброса WAL. Это может улучшить пропускную способность групповой фиксации, позволяя большему количеству транзакций фиксироваться через один сброс WAL, если загрузка системы достаточно высока, чтобы дополнительные транзакции были готовы к фиксации в течение заданного интервала. Однако это также увеличивает задержку commit_delay до микросекунд для каждого сброса WAL. Поскольку задержка просто теряется, если никакие другие транзакции не готовы к фиксации, задержка выполняется только в том случае, если, по крайней мере, активны commit_siblings других транзакций. Кроме того, никакие задержки не выполняются, если fsync отключено. По умолчанию commit_delay равно нулю (без задержки). Только администраторы могут изменять этот параметр.

```
commit_siblings(integer)
```

Минимальное количество одновременных открытых транзакций, необходимых для выполнения задержки commit_delay. Большее значение повышает вероятность того, что хотя бы еще одна транзакция будет готова к фиксации в течение интервала задержки. По умолчанию используется пять транзакций.

2.3.4.2. Контрольные точки (Checkpoints)

```
checkpoint_timeout(integer)
```

Максимальное время между автоматическими контрольными точками WAL, в секундах. Допустимый диапазон составляет от 30 секунд до одного дня. По умолчанию это пять минут (5min). Увеличение этого параметра может увеличить время, необходимое для восстановления

после сбоя. Этот параметр можно установить только в файле конфигурации СУБД-КС или в командной строке сервера.

```
checkpoint_completion_target(floating point)
```

Определяет цель завершения контрольной точки, как часть общего времени между контрольными точками. По умолчанию это 0,5. Этот параметр можно установить только в файле конфигурации СУБД-КС или в командной строке сервера.

```
checkpoint_flush_after(integer)
```

Всякий раз, когда во время выполнения контрольной точки было записано больше байтов чем checkpoint_flush_after, сервер заставляет ОС поместить эти записи в основное хранилище. Это ограничит количество «грязных» данных в кеше страниц ядра, уменьшит вероятность зависаний при fsync в конце контрольной точки или когда ОС записывает данные большими пакетами в фоновом режиме. Зачастую это приводит к значительному снижению задержки транзакций. Этот параметр можно установить только в файле конфигурации СУБД-КС или в командной строке сервера.

```
checkpoint_warning(integer)
```

Записать сообщение в журнал сервера, если контрольные точки, вызванные заполнением файлов сегментов WAL, выполняются быстрее, чем через заданное значение секунд (что указывает на необходимость повышения значения max_wal_size). По умолчанию 30 секунд (30s). Ноль отключает предупреждение. Предупреждения не будут генерироваться, если checkpoint_timeout меньше checkpoint_warning. Этот параметр можно установить только в файле конфигурации СУБД-КС или в командной строке сервера.

```
max_wal_size(integer)
```

Максимальный размер, позволяющий WAL расти между автоматическими контрольными точками WAL. Это мягкий предел; размер WAL может превышать max_wal_size при особых обстоятельствах, например, при большой нагрузке, сбое archive_command или высоких wal_keep_segments настройках. По умолчанию это 1 ГБ. Увеличение этого параметра может увеличить время, необходимое для восстановления после сбоя. Этот параметр можно установить только в файле конфигурации СУБД-КС или в командной строке сервера.

```
min_wal_size(integer)
```

Пока использование диска WAL остается ниже этого параметра, старые файлы WAL всегда перерабатываются для использования в будущем в контрольной точке, а не удаляются. Это можно использовать для гарантии того, что достаточно места WAL зарезервировано для обработки пиков при использовании WAL, например, при выполнении больших пакетных заданий. По умолчанию установлено значение 80 МБ. Этот параметр можно установить только в файле конфигурации СУБД-КС или в командной строке сервера.

2.3.4.3. Архивирование

```
archive_mode(enum)
```

Когда этот параметр archive_mode включен, заполненные сегменты WAL отправляются в архивное хранилище с помощью параметра archive_command. В дополнение к off-отключению есть два режима: on и always. Во время нормальной работы нет никакой разницы между этими двумя режимами, но при установке always на WAL архиватор включается также во время восстановления архива или в режиме ожидания. В always режиме все файлы, восстановленные из архива или переданные с потоковой репликацией, будут заархивированы (снова).

archive_mode и archive_command являются отдельными переменными, так что archive_command их можно изменять, не выходя из режима архивирования. Этот параметр можно установить только при запуске сервера. archive_mode не может быть включен, когда wal_levelустановлен на minimal.

```
archive_command(string)
```

Команда локальной оболочки, вызывается для архивирования завершенного сегмента файла WAL. Любой знак %p в строке заменяется на путь к файлу для архивирования, а любой знак %f заменяется только именем файла. Путь указывается относительно рабочего каталога сервера, т.е. каталога данных кластера.) Необходимо использовать двойной знак %% для встраивания фактического символа % в команду. Для команды важно возвратить нулевой статус выхода, если она выполнена успешно.

Этот параметр можно установить только в файле конфигурации СУБД-КС или в командной строке сервера. Он игнорируется, если archive_mode не был включен при запуске сервера. Если в archive_command задана пустая строка (по умолчанию), а archive_mode архивация WAL временно отключена, то сервер продолжает накапливать файлы сегментов WAL в ожидании, что в ближайшее время будет предоставлена команда. Установка archive_command команды, которая ничего не делает, но возвращает значение true, эффективно отключает архивирование, но также нарушает цепочку файлов WAL, необходимых для восстановления архива, поэтому ее следует использовать только в особых случаях.

```
archive_timeout(integer)
```

Команда archive_command вызывается только для завершенных сегментов WAL. Следовательно, если ваш сервер генерирует небольшой трафик WAL (или имеет периоды простоя, когда это происходит), между завершением транзакции и ее безопасной записью в архивном хранилище может пройти длительная задержка. Чтобы ограничить возраст не архивированных данных, вы можете установить archive_timeout и заставить сервер периодически переключаться на новый файл сегмента WAL. Если этот параметр больше нуля, сервер переключается на новый файл сегмента по истечении этого количества секунд с момента последнего переключения файла сегмента, и когда-либо выполнялась какая-либо операция с базой данных, включая одну контрольную точку (контрольные точки пропускаются, если нет активность базы данных). Обратите внимание, что архивированные файлы, которые закрываются рано из-за принудительного переключения, имеют ту же длину, что и полностью заполненные файлы. Следовательно, неразумно использовать очень короткие archive_timeout - это приведет к переполнению вашего архивного хранилища. Значение настройки archive_timeout минута или

около того, как правило, является оптимальной. Этот параметр можно установить только в файле конфигурации СУБД-КС или в командной строке сервера.

2.3.5. Планирование запросов

2.3.5.1. Конфигурация метода планирования

Эти параметры конфигурации обеспечивают грубый метод воздействия на планы запросов, выбранные оптимизатором запросов. Если план по умолчанию, выбранный оптимизатором для определенного запроса, не является оптимальным, временное решение состоит в том, чтобы использовать один из этих параметров конфигурации, чтобы оптимизатор выбрал другой план. Более эффективные способы улучшения качества планов, выбранных оптимизатором, включают настройку констант стоимости планировщика, ручной запуск ANALYZE, увеличение значения параметра конфигурации default_statistics_target и увеличение объема статистики, собираемой для определенных столбцов, используя ALTER TABLE SET STATISTICS.

```
enable_bitmapscan(boolean)
```

Включает или отключает использование планировщиком запросов типов планов по битовой карте. По умолчанию имеет значение on.

```
enable_gathermerge(boolean)
```

Включает или отключает использование планировщиком запросов типов планов слияния. По умолчанию имеет значение on.

```
enable_hashagg(boolean)
```

Включает или отключает использование планировщиком запросов типов планов хэшированной агрегации. По умолчанию имеет значение on.

```
enable_hashjoin(boolean)
```

Включает или отключает использование планировщиком запросов типов планов хэш-соединения. По умолчанию имеет значение on.

```
enable_indexscan(boolean)
```

Включает или отключает использование планировщиком запросов типов планов сканирования индекса. По умолчанию имеет значение on.

```
enable_indexonlyscan(boolean)
```

Включает или отключает использование планировщиком запросов типов планов проверки только для индекса. По умолчанию имеет значение on.

```
enable_material(boolean)
```

Включает или отключает использование материализации в планировщике запросов. Невозможно полностью подавить материализацию, но отключение этой переменной не позволяет планировщику вставить узлы материализации, за исключением случаев, когда это требуется для корректности. По умолчанию имеет значение on.

```
enable_mergejoin(boolean)
```

Включает или отключает использование планировщиком запросов типов планов объединения слиянием. По умолчанию имеет значение on.

```
enable_nestloop(boolean)
```

Включает или отключает использование плановиком запросов планов соединения с вложенными циклами. Невозможно полностью исключить соединения с вложенными циклами, но отключение этой переменной не позволяет планировщику использовать ее, если есть другие доступные методы. По умолчанию имеет значение on.

```
enable_parallel_append(boolean)
```

Включает или отключает использование планировщиком запросов типов планов добавления с параллельной поддержкой. По умолчанию имеет значение on.

```
enable_parallel_hash(boolean)
```

Включает или отключает использование планировщиком запросов типов планов хеш-соединения с параллельным хешем. Не имеет эффекта, если планы хеш-соединения также не включены. По умолчанию имеет значение on.

```
enable_partition_pruning(boolean)
```

Включает или отключает возможность планировщика запросов исключать разделы многораздельной таблицы из планов запросов. Это также контролирует способность планировщика генерировать планы запросов, которые позволяют исполнителю запросов удалять (игнорировать) разделы во время выполнения запроса. По умолчанию имеет значение on.

```
enable_partitionwise_join(boolean)
```

Включает или отключает использование планировщиком запросов разделения по частям, что позволяет выполнять соединение между разделенными таблицами путем объединения соответствующих разделов. В настоящее время объединение по частям применяется только в том случае, если в условия объединения включены все ключи разделов, которые должны иметь одинаковый тип данных и иметь точно совпадающие наборы дочерних разделов. Поскольку при

планировании объединения по частям во время планирования может использоваться значительно больше процессорного времени и памяти, по умолчанию используется значение off.

```
enable_partitionwise_aggregate(boolean)
```

Включает или отключает использование планировщиком запросов групповой группировки или агрегации по секциям, что позволяет группировать или агрегировать по секционированным таблицам, выполняемым отдельно для каждой секции. Если GROUP BY предложение не включает ключи раздела, только частичное агрегирование может быть выполнено для каждого раздела, а финализация должна быть выполнена позже. Поскольку групповое группирование или агрегация могут использовать значительно больше процессорного времени и памяти во время планирования, по умолчанию используется off.

```
enable_seqscan(boolean)
```

Включает или отключает использование планировщиком запросов типов планов последовательного сканирования. Невозможно полностью отключить последовательное сканирование, но отключение этой переменной не позволяет планировщику использовать его, если есть другие доступные методы. По умолчанию имеет значение on.

```
enable_sort(boolean)
```

Включает или отключает использование планировщиком запросов явных шагов сортировки. Невозможно полностью исключить явные сортировки, но отключение этой переменной не позволяет планировщику использовать ее, если есть другие доступные методы. По умолчанию имеет значение on.

```
enable_tidscan(boolean)
```

Включает или отключает использование планировщиком запросов типов планов сканирования TID. По умолчанию имеет значение on.

2.3.5.2. Константы стоимости для планировщика

Переменные стоимости, описанные в этом разделе, оцениваются в произвольном масштабе. Только их относительные значения имеют значение, следовательно, масштабирование их всех вверх или вниз на один и тот же коэффициент не приведет к изменению выбора планировщика. По умолчанию эти переменные стоимости основаны на стоимости последовательных выборок страниц; то есть seq_page_cost обычно устанавливается в значение, 1.0, а другие переменные стоимости устанавливаются со ссылкой на данную переменную. Допустимо использовать другой масштаб, если необходимо получить, к примеру, фактическое время выполнения в миллисекундах на конкретной машине.

К сожалению, не существует четко определенного метода определения идеальных значений для переменных стоимости. Их лучше всего рассматривать как средние значения для всего набора запросов, которые получит конкретная установка. Это означает, что менять их на основании всего лишь нескольких экспериментов очень

рискованно.

```
seq_page_cost(floating point)
```

Устанавливает плановую оценку стоимости выборки страницы на диске, которая является частью серии последовательных выборок. По умолчанию это 1.0. Это значение можно переопределить для таблиц и индексов в определенном табличном пространстве, установив параметр табличного пространства с тем же именем.

```
random_page_cost(floating point)
```

Устанавливает оценку планировщика стоимости непоследовательной выборки страницы диска. По умолчанию это 4.0. Это значение можно переопределить для таблиц и индексов в определенном табличном пространстве, установив параметр табличного пространства с тем же именем.

Уменьшение этого значения относительно seq_page_cost приведет к тому, что система предпочтет сканирование индекса; его увеличение сделает просмотр индекса относительно более дорогим. Вы можете увеличить или уменьшить оба значения вместе, чтобы изменить важность затрат дискового ввода-вывода по сравнению со стоимостью процессора, которые описываются следующими параметрами.

Произвольный доступ к механическому дисковому хранилищу обычно намного дороже, чем четырехкратный последовательный доступ. Однако используется более низкий уровень по умолчанию (4.0), поскольку предполагается, что большинство случайных обращений к диску, таких как индексированные чтения, находятся в кэше. Значение по умолчанию можно рассматривать как моделирование произвольного доступа как в 40 раз медленнее, чем последовательное, при этом ожидается, что 90% случайных чтений будут кэшироваться.

Если вы считаете, что кэш-память в 90% является неверным допущением для вашей рабочей нагрузки, вы можете увеличить random_page_cost, чтобы лучше отражать истинную стоимость случайных операций чтения из хранилища. Соответственно, если ваши данные, скорее всего, будут полностью в кэше, например, когда база данных меньше общей памяти сервера, может быть целесообразно уменьшение random_page_cost. Хранилище с низкой стоимостью случайного чтения относительно последовательных, например, твердотельных накопителей, также может быть лучше смоделировано с более низким значением random_page_cost.

Хотя система позволит вам установить значение random_page_cost меньше, чем seq_page_cost, это физически нецелесообразно. Однако установка их равными имеет смысл, если база данных полностью кэшируется в ОЗУ, поскольку в этом случае издержки несущественны. Кроме того, в сильно кэшированной базе данных вы должны понизить оба значения относительно параметров ЦП, поскольку стоимость извлечения страницы уже в ОЗУ намного меньше, чем обычно.

```
cpu_tuple_cost(floating point)
```

Устанавливает плановую оценку стоимости обработки каждой строки во время запроса. По умолчанию это 0,01.

```
cpu_index_tuple_cost(floating point)
```

Устанавливает плановую оценку стоимости обработки каждой записи индекса во время сканирования индекса. По умолчанию это 0,005.

```
cpu_operator_cost(floating point)
```

Устанавливает плановую оценку стоимости обработки каждого оператора или функции, выполняемой во время запроса. По умолчанию это 0,0025.

```
parallel_setup_cost(floating point)
```

Устанавливает плановую оценку стоимости запуска параллельных рабочих процессов. По умолчанию это 1000.

```
parallel_tuple_cost(floating point)
```

Устанавливает плановую оценку стоимости переноса одного кортежа из параллельного рабочего процесса в другой процесс. По умолчанию это 0,1.

```
min_parallel_table_scan_size(integer)
```

Устанавливает минимальный объем табличных данных, которые необходимо сканировать, чтобы можно было рассмотреть параллельное сканирование. При параллельном последовательном сканировании объем проверенных данных таблицы всегда равен размеру таблицы, но при использовании индексов количество проверенных данных таблицы обычно будет меньше. По умолчанию установлено 8 мегабайт (8MB).

```
min_parallel_index_scan_size(integer)
```

Устанавливает минимальный объем индексных данных, которые должны быть отсканированы, чтобы можно было рассмотреть параллельное сканирование. Обратите внимание, что параллельное сканирование индекса обычно не затрагивает весь индекс; это число страниц, которое, по мнению плановика, будет действительно затронуто сканированием, которое имеет отношение к делу. По умолчанию это 512 килобайт (512kB).

```
effective_cache_size(integer)
```

Устанавливает предположение планировщика об эффективном размере дискового кэша, доступного для одного запроса. Это учитывается при оценке стоимости использования индекса; чем выше значение, тем выше вероятность того, что будет использоваться индексное сканирование, чем ниже значение, тем больше вероятность того, что будет использовано последовательное сканирование. При установке этого параметра вы должны учитывать как общие буферы СУБД-КС, так и часть дискового кэша ядра, которая будет использоваться для файлов данных СУБД-КС, хотя некоторые данные могут существовать в обоих местах. Кроме того, примите во внимание ожидаемое количество одновременных запросов к разным таблицам, так как им придется совместно использовать доступное пространство. Этот параметр не влияет на размер разделяемой памяти, выделяемой СУБД-КС и не резервирует кэш диска на ядре; он используется

только для целей оценки. Система также не предполагает, что данные остаются в кэше диска между запросами. По умолчанию это 4 гигабайта (4GB).

```
jit_above_cost(floating point)
```

Задает стоимость запроса, выше которой активируется JIT-компиляция, если она включена. Выполнение JIT тратит время на планирование, но может ускорить выполнение запроса. Установка этого параметра -1отключает компиляцию JIT. По умолчанию это 100000.

```
jit_inline_above_cost(floating point)
```

Устанавливает стоимость запроса, выше которой JIT-компиляция пытается встроить функции и операторы. Встраивание добавляет время планирования, но может улучшить скорость выполнения. Не имеет смысла устанавливать это значение меньше, чем jit_above_cost. Установка этого параметра -1отключает встраивание. По умолчанию это 500000.

```
jit_optimize_above_cost(floating point)
```

Устанавливает стоимость запроса, выше которой JIT-компиляция применяет дорогостоящие оптимизации. Такая оптимизация добавляет время планирования, но может улучшить скорость выполнения. Не имеет смысла устанавливать это значение меньше чем jit_above_cost, и вряд ли будет выгодно устанавливать это значение больше чем jit_inline_above_cost. Установка этого параметра -1отключает дорогостоящие оптимизации. По умолчанию это 500000.

2.3.5.3. Генетический оптимизатор запросов

Оптимизатор генетических запросов (GEQO) - это алгоритм, который выполняет планирование запросов с использованием эвристического поиска. Это сокращает время планирования сложных запросов (тех, которые объединяют многие отношения) за счет создания планов, которые иногда уступают тем планам, которые получены обычным перебором.

```
geqo(boolean)
```

Включает или отключает генетическую оптимизацию запросов, включено по умолчанию. Обычно лучше не выключать его в производстве; переменная geqo_threshold обеспечивает более точное управление GEQO.

```
geqo_threshold(integer)
```

Используйте генетическую оптимизацию запросов, чтобы планировать запросы, по крайней мере, с таким количеством FROM элементов (обратите внимание, что FULL OUTER JOIN конструкция считается только одним FROM элементом) По умолчанию используется значение 12. Для более простых запросов обычно лучше использовать обычный планировщик полного перебора, но для запросов со многими таблицами полный перебор занимает слишком много времени, часто дольше, чем время на выполнение неоптимального плана. Таким образом,

пороговое значение для размера запроса является удобным способом управления использованием GEQO.

```
geqo_effort(integer)
```

Управляет компромиссом между временем планирования и качеством плана запроса в GEQO. Эта переменная должна быть целым числом в диапазоне от 1 до 10. Значением по умолчанию является пять. Большие значения увеличивают время, затрачиваемое на планирование запросов, но также повышают вероятность выбора эффективного плана запросов.

geqo_effort на самом деле ничего не делает напрямую; он используется только для вычисления значений по умолчанию для других переменных, которые влияют на поведение GEQO (описано ниже). Вы также можете установить другие параметры вручную.

```
geqo_pool_size(integer)
```

Контролирует размер пула, используемого GEQO, то есть количество особей в генетической популяции. Оно должно быть не менее двух, а полезные значения обычно составляют от 100 до 1000. Если оно установлено на ноль (настройка по умолчанию), то подходящее значение выбирается на основе geqo_effort количества таблиц в запросе.

```
geqo_generations(integer)
```

Управляет количеством поколений, используемых GEQO, то есть количеством итераций алгоритма. Должно быть хотя бы одно, а полезные значения находятся в том же диапазоне, что и размер пула. Если он установлен на ноль (настройка по умолчанию), то подходящее значение выбирается на основе geqo_pool_size.

```
geqo_selection_bias(floating point)
```

Управляет смещением выбора, используемым GEQO. Смещение выбора - это избирательное давление в популяции. Значения могут быть от 1,50 до 2,00; последний по умолчанию.

```
geqo_seed(floating point)
```

Управляет начальным значением генератора случайных чисел, используемого GEQO для выбора случайных путей в пространстве поиска порядка соединения. Значение может варьироваться от нуля (по умолчанию) до единицы. Изменение значения изменяет набор исследуемых путей соединения и может привести к тому, что будет найден худший путь.

2.3.5.4. Другие параметры планировщика

```
default_statistics_target(integer)
```

Устанавливает цель статистики по умолчанию для столбцов таблицы без цели для конкретного столбца, установленной с помощью ALTER TABLE SET STATISTICS. Большие

значения увеличивают время, необходимое для выполнения ANALYZE, но могут улучшить качество оценок планировщика. Значение по умолчанию - 100.

```
constraint_exclusion(enum)
```

Управляет использованием в планировщике запросов ограничений таблицы для оптимизации запросов. Допустимые значения constraint_exclusion: on (проверять ограничения для всех таблиц), off (никогда не проверять ограничения) и partition (проверять ограничения только для дочерних таблиц и подзапросов наследования UNION ALL). Partition - это настройка по умолчанию. Он часто используется с традиционными деревьями наследования для повышения производительности.

Когда этот параметр разрешает его для определенной таблицы, планировщик сравнивает условия запроса с ограничениями проверок таблицы и пропускает таблицы сканирования, для которых условия противоречат ограничениям. Например:

```
CREATE TABLE parent(key integer,...);
CREATE TABLE child1000(check (key between 1000 and 1999)) INHERITS(parent);
CREATE TABLE child2000(check (key between 2000 and 2999)) INHERITS(parent);
...
SELECT * FROM parent WHERE key = 2400;
```

При включенном исключении ограничений сканирование SELECT не будет выполняться для таблицы child 1000 вообще, что повысит производительность.

В настоящее время исключение ограничений по умолчанию включено только для случаев, которые часто используются для реализации разбиения таблиц через деревья наследования. Включение этого параметра для всех таблиц приводит к дополнительным затратам на планирование, что заметно при простых запросах и чаще всего не дает никаких преимуществ для простых запросов. Если у вас нет таблиц, которые разделены с использованием традиционного наследования, вы можете отключить его полностью (обратите внимание, что эквивалентная функция для секционированных таблиц управляется отдельным параметром enable_partition_pruning).

```
cursor_tuple_fraction(floating point)
```

Устанавливает оценку планировщика доли строк курсора, которые будут получены. По умолчанию это 0,1. Меньшие значения этого параметра склоняют планировщик в сторону использования планов « быстрого запуска » для курсоров, которые будут быстро извлекать первые несколько строк, и, возможно, потребуется много времени для извлечения всех строк. Большие значения делают больший акцент на общее расчетное время. При максимальном значении 1,0 курсоры планируются точно так же, как и обычные запросы, с учетом только общего расчетного времени, а не того, как скоро могут быть доставлены первые строки.

```
from_collapse_limit(integer)
```

Планировщик объединит подзапросы в верхние запросы, если результирующий список FROM будет содержать не более этого количества элементов. Меньшие значения сокращают время планирования, но могут привести к ухудшению планов запросов. По умолчанию восемь.

Установка этого значения в geqo_threshold или более может инициировать использование планировщика GEQO, что приведет к неоптимальным планам.

```
jit(boolean)
```

Определяет, может ли компиляция JIT использоваться СУБД-КС, если она доступна. По умолчанию это off.

```
join_collapse_limit(integer)
```

Планировщик будет переписывать явные JOIN онструкции (кроме FULL JOINs) в списки FROM элементов всякий раз, когда будет получен список, содержащий не более этого количества элементов. Меньшие значения сокращают время планирования, но могут привести к ухудшению планов запросов.

По умолчанию эта переменная установлена так же, как from_collapse_limit, что подходит для большинства применений. Установка его в 1 предотвращает любое изменение порядка явных JOINs. Таким образом, явный порядок соединения, указанный в запросе, будет фактическим порядком, в котором соединяются отношения. Поскольку планировщик запросов не всегда выбирает оптимальный порядок соединения, опытные пользователи могут временно установить для этой переменной значение 1, а затем явно указать желаемый порядок соединения. Установка этого значения в geqo_threshold или более может инициировать использование планировщика GEQO, что приведет к неоптимальным планам.

```
parallel_leader_participation(boolean)
```

Позволяет основному процессу выполнять план запроса в узлах Gather и Gather Merge узлах, не ожидая рабочих процессов. По умолчанию имеет значение on. Установка его в значение off уменьшает вероятность того, что процессы будут заблокированы, потому что основной процесс не читает кортежи достаточно быстро, но требует, чтобы основной процесс ожидал запуска рабочих процессов, прежде чем будут созданы первые кортежи. Степень, в которой основной процесс может помочь или снизить производительность, зависит от типа плана, количества рабочих процессов и продолжительности запроса.

```
force_parallel_mode(enum)
```

Позволяет использовать параллельные запросы для целей тестирования даже в тех случаях, когда повышение производительности не ожидается. Допустимые значения force_parallel_mode: off (использовать параллельный режим только тогда, когда ожидается повышение производительности), on (принудительный параллельный запрос для всех запросов, для которых он считается безопасным) и regress (например, on, но с дополнительными изменениями поведения, как описано ниже).

Точнее говоря, установка этого значения on добавит Gather узел в начало любого плана запроса, для которого это кажется безопасным, так что запрос выполняется внутри параллельного процесса. Даже если параллельный процесс недоступен или не может использоваться, такие операции, как запуск субтранзакции, которая будет запрещена в контексте параллельного запроса, будут запрещены, если только планировщик не считает, что это приведет к сбою запроса. Если при установке этого параметра возникают сбои или неожиданные результаты, возможно,

некоторые функции, используемые запросом, необходимо пометить PARALLEL UNSAFE (или, возможно, PARALLEL RESTRICTED).

Установка значения regress имеет все те же эффекты, что и его установка, on плюс некоторые дополнительные эффекты, предназначенные для облегчения автоматического регрессионного тестирования. Обычно сообщения от параллельного процесса включают контекстную строку, указывающую на это, но настройка regress подавляет эту строку, так что выходные данные такие же, как при не параллельном выполнении. Кроме того, узлы Gather, добавленные в планы с помощью этого параметра, скрыты в выходных данных EXPLAIN, поэтому выходные данные соответствуют тому, что будет получено, если этот параметр был изменен на off.

2.3.6. Отчеты об ошибках и ведение журнала

2.3.6.1. Место протоколирования

```
log_destination(string)
```

СУБД-КС поддерживает несколько методов регистрации сообщений сервера: stderr, csvlog и syslog. Для этого параметра указывается список желаемых пунктов размещения журнала, разделенных запятыми. По умолчанию размещение только в stderr. Этот параметр можно установить только в конфигурационном файле или в командной строке сервера.

```
log_directory(string)
```

Когда параметр logging_collector включен, этот параметр определяет каталог, в котором будут создаваться файлы журналов. Его можно указать как абсолютный путь или относительно каталога данных кластера. Этот параметр можно установить только в файле конфигурации или в командной строке сервера.

```
log_filename(string)
```

Когда параметр logging_collector включен, этот параметр устанавливает имена созданных файлов журнала. Значение рассматривается как strftime шаблон, поэтому %-escapes можно использовать для указания изменяющихся во времени имен файлов (обратите внимание, что при наличии каких-либо зависимых от %часового пояса -escapes вычисление выполняется в зоне, указанной в log_timezone). Поддерживаемые %-escapes аналогичны тем, которые перечислены в спецификации strftime открытой группы. Обратите внимание, что система strftime не используется напрямую, поэтому специфичные для платформы (нестандартные) расширения не работают.

Если вы указываете имя файла без экранирования, вы должны запланировать использование утилиты ротации журналов, чтобы в конечном итоге не заполнить весь диск.

Если вывод в формате CSV включен log_destination, *.csv будет добавлен к имени файла журнала с меткой времени для создания имени файла для вывода в формате CSV.

Этот параметр можно установить только в файле конфигурации или в командной строке сервера.

```
log_file_mode(integer)
```

Этот параметр устанавливает разрешения для файлов журнала, если logging_collector on включен. Значением параметра будет числовой режим, заданный в формате, понятном для системных вызовов chmod и umask (для использования обычного восьмеричного формата число должно начинаться с 0(нуля)).

Разрешения по умолчанию – 0600 означает, что только владелец сервера может читать или записывать файлы журнала. Другой обычно полезный параметр – 0640 разрешить членам группы владельца читать файлы. Однако обратите внимание, что для использования такой настройки вам нужно изменить log_directory, чтобы хранить файлы где-то за пределами каталога данных кластера. В любом случае неразумно делать файлы журналов общедоступными, поскольку они могут содержать конфиденциальные данные.

Этот параметр можно установить только в файле конфигурации или в командной строке сервера.

```
log_rotation_age(integer)
```

Когда logging_collector этот параметр включен, этот параметр определяет максимальное время жизни отдельного файла журнала. По истечении этого количества минут будет создан новый файл журнала. Установите на ноль, чтобы отключить создание новых файлов журнала на основе времени. Этот параметр можно установить только в конфигурационном файле или в командной строке сервера.

```
log_rotation_size(integer)
```

Когда параметр logging_collector включен, он определяет максимальный размер отдельного файла журнала. После того, как эти многие килобайты будут записаны в файл журнала, будет создан новый файл журнала. Установите на ноль, чтобы отключить создание новых файлов журнала на основе размера. Этот параметр можно установить только в файле конфигурации или в командной строке сервера.

```
log_truncate_on_rotation(boolean)
```

Когда параметр logging_collector включен, СУБД-КС будет обрезать (перезаписывать), а не добавлять к любому существующему файлу журнала с таким же именем. Однако усечение будет происходить только при открытии нового файла из-за ротации по времени, а не при запуске сервера или ротации по размеру. Если этот параметр отключен, во все случаи будут добавляться уже существующие файлы. Этот параметр можно установить только в файле конфигурации или в командной строке сервера.

Пример

Чтобы хранить 7 дней журналов, один файл журнала в день с именем server_log.Mon и server_log.Tue и т. д., а также с возможностью автоматически перезаписывать журнал прошлой недели с журналом этой недели, нужно установить:

log_filename=server_log.%a

log_truncate_on_rotation=on

log_rotation_age=1440.

Пример

Чтобы хранить журналы в течение 24 часов, один файл журнала в час, но при этом с возможностью переключения файла, если файл журнала превышает 1 ГБ, установите значение:

log_filename=server_log.%H%M

log_truncate_on_rotation=on

log_rotation_age=60

log_rotation_sizeв=1000000.

Добавление %M в log_filename позволит при переключении по размеру указать другое имя файла в пределах одного часа, отличное от начального имени файла часа.

```
syslog_facility(enum)
```

Если включен syslog, то это параметр определяет значение «facility». Допустимые значения LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, LOCAL7. По умолчанию используется LOCAL0.

```
syslog_ident(string)
```

Когда включена запись в системный журнал, этот параметр определяет имя программы, используемое для идентификации сообщений СУБД-КС в журналах системного журнала. По умолчанию это postgres. Этот параметр можно установить только в файле конфигурации или в командной строке сервера.

```
syslog_sequence_numbers(boolean)
```

Если включен syslog, каждое сообщение будет иметь префикс возрастающего порядкового номера (например, [2]). Это позволяет обойти подавление « --- последнее сообщение повторено N раз --- », которое многие реализации системного журнала выполняют по умолчанию. В более современных реализациях системного журнала можно настроить повторное подавление сообщений (например, $RepeatedMsgReductionв rsyslog), поэтому в этом может не быть необходимости. Кроме того, вы можете отключить это, если вы действительно хотите подавить повторяющиеся сообщения.

Этот параметр можно установить только в файле конфигурации или в командной строке сервера.

```
syslog_split_messages(boolean)
```

Если включен syslog, этот параметр определяет способ доставки сообщений в системный журнал. При включении (по умолчанию) сообщения разделяются на строки, а длинные строки разделяются так, что они помещаются в 1024 байта, что является типичным пределом размера для традиционных реализаций системного журнала. Когда он выключен, сообщения журнала сервера СУБД-КС доставляются в службу системного журнала как есть, и служба системного журнала должна обрабатывать потенциально громоздкие сообщения.

Если в конечном итоге системный журнал регистрируется в текстовом файле, то эффект будет одинаковым в любом случае, и лучше оставить этот параметр включенным, поскольку большинство реализаций системного журнала либо не могут обрабатывать большие сообщения, либо их необходимо специально настраивать для их обработки. Но если системный журнал в конечном итоге записывает данные на какой-либо другой носитель, может оказаться необходимым или более полезным хранить сообщения логически вместе.

Этот параметр можно установить только в файле конфигурации или в командной строке сервера.

```
event_source(string)
```

Когда включена запись в журнал событий, этот параметр определяет имя программы, используемое для идентификации сообщений СУБД-КС в журнале. По умолчанию это СУБД-КС. Этот параметр можно установить только в файле конфигурации или в командной строке сервера.

2.3.6.2. Время протоколирования

```
log_min_messages(enum)
```

Управляет тем, какие уровни сообщений записываются в журнал сервера. Допустимые значения DEBUG5, DEBUG4, DEBUG3, DEBUG2, DEBUG1, INFO, NOTICE, WARNING, ERROR, LOG, FATAL, и PANIC. Каждый уровень включает в себя все уровни, которые следуют за ним. Чем позже уровень, тем меньше сообщений отправляется в журнал. По умолчанию это WARNING. Обратите внимание, что здесь LOG имеет другой ранг, чем в client_min_messages. Только администраторы могут изменять этот параметр.

```
log_min_error_statement(enum)
```

Управляет тем, какие операторы SQL, которые вызывают ошибку, записываются в журнал сервера. Текущий оператор SQL включен в запись журнала для любого сообщения указанной серьезности или выше. Допустимые значения DEBUG5, DEBUG4, DEBUG3, DEBUG2, DEBUG1, INFO, NOTICE, WARNING, ERROR, LOG, FATAL, и PANIC. Значение по умолчанию: ERROR - операторы, вызывающие ошибки, сообщения журнала, фатальные ошибки или паники. Чтобы эффективно отключить запись неудачных операторов, установите для этого параметра значение PANIC. Только администраторы могут изменять этот параметр.

```
log_min_duration_statement(integer)
```

Заставляет регистрировать продолжительность каждого выполненного оператора, если запрос выполнялся по крайней мере указанное количество миллисекунд. Установка этого значения

в ноль регистрирует продолжительность всех команд. Значение -1 (по умолчанию) отключает запись продолжительности команд. Включение этого параметра может быть полезно для отслеживания неоптимизированных запросов. Только администраторы могут изменять этот параметр.

Таблица 2 объясняет уровни серьезности сообщений, используемые СУБД-КС.

Таблица 2. Уровни серьезности сообщений

Колонка	Уровни серьезности сообщений			
	Строгость	Использование	Системный журнал	Журнал событий
1	2	3	4	5
1	DEBUG1..DEBUG5	Предоставляет последовательно-более подробную информацию для использования разработчиками.	DEBUG	INFORMATION
2	INFO	Предоставляет информацию, неявно запрашиваемую пользователем, например, вывод из VACUUM VERBOSE.	INFO	INFORMATION
3	NOTICE	Предоставляет информацию, которая может быть полезна пользователям, например, уведомление об усечении длинных идентификаторов.	NOTICE	INFORMATION
4	WARNING	Предоставляет предупреждения о возможных проблемах, например, COMMITвне блока транзакции.	NOTICE	WARNING
5	ERROR	Сообщает об ошибке, которая привела к прерыванию текущей команды.	WARNING	ERROR
6	LOG	Сообщает информацию, представляющую интерес для администраторов, например, действия контрольных точек.	INFO	INFORMATION
	FATAL	Сообщает об ошибке, которая привела к прерыванию текущего сеанса.	ERR	ERROR
	PANIC	Сообщает об ошибке,	CRIT	ERROR

Колонка	Уровни серьезности сообщений			
	Строгость	Использование	Системный журнал	Журнал событий
1	2	3	4	5
		которая привела к прерыванию всех сеансов базы данных.		

2.3.6.3. Протоколируемая информация

```
application_name(string)
```

application_name может быть любая строка менее NAMEDATALEN символов (64 символа в стандартной сборке). Обычно устанавливается приложением при подключении к серверу. Имя будет отображаться в виде pg_stat_activity и включаться в записи журнала CSV. Его также можно включить в обычные записи журнала с помощью параметра log_line_prefix. В значении application_name могут использоваться только печатные символы ASCII. Другие символы будут заменены на вопросительные знаки (?).

```
debug_print_parse(boolean)
debug_print_rewritten(boolean)
debug_print_plan(boolean)
```

Эти параметры позволяют выводить различные выходные данные отладки. Когда они установлены, они печатают результирующее дерево разбора, выходные данные программы переписывания запросов или план выполнения для каждого выполненного запроса. Эти сообщения отправляются на уровне LOG - сообщений, поэтому по умолчанию они отображаются в журнале сервера, но не отправляются клиенту. Вы можете изменить это, настроив client_min_messages и / или log_min_messages. Эти параметры по умолчанию отключены.

```
debug_pretty_print(boolean)
```

При установке отступы сообщения debug_pretty_print, произведенном debug_print_parse, debug_print_rewritten или debug_print_plan. Это приводит к более удобному для восприятия, но намного более длинному выводу, чем «компактный» формат, используемый, когда он выключен. Параметр включен по умолчанию.

```
log_checkpoints(boolean)
```

Заставляет контрольные точки и точки перезапуска регистрироваться в журнале сервера. Некоторая статистика включена в сообщения журнала, включая количество записанных буферов и время, потраченное на их запись. Этот параметр можно установить только в файле конфигурации или в командной строке сервера. По умолчанию выключено.

```
og_connections(boolean)
```

Приводит к регистрации каждой попытки подключения к серверу, а также успешного завершения проверки подлинности клиента. Только администраторы могут изменять этот параметр при запуске сеанса, и его нельзя изменить вообще в течение сеанса. По умолчанию это off.

Некоторые клиентские программы, такие как psql, пытаются подключиться дважды, определяя, требуется ли пароль, поэтому повторяющиеся сообщения «соединение получено» не обязательно указывают на проблему.

```
log_disconnections(boolean)
```

Заставляет завершения сеанса регистрироваться. Вывод журнала предоставляет информацию, аналогичную log_connections, плюс продолжительность сеанса. Только администраторы могут изменять этот параметр при запуске сеанса, и его нельзя изменить вообще в течение сеанса. По умолчанию это off.

```
log_duration(boolean)
```

Вызывает продолжительность каждого выполненного оператора для регистрации. По умолчанию это off. Только администраторы могут изменять этот параметр.

```
log_error_verbosity(enum)
```

Управляет количеством сведений, записываемых в журнал сервера для каждого зарегистрированного сообщения. Допустимые значения TERSE, DEFAULT и VERBOSE, каждый из которых добавляет больше полей для отображаемых сообщений. TERSE исключает регистрацию DETAIL, HINT, QUERY и CONTEXT информации об ошибках. Выходные данные VERBOSE включают код ошибки SQLSTATE и имя файла исходного кода, имя функции и номер строки, которая вызвала ошибку. Только администраторы могут изменять этот параметр.

```
log_hostname(boolean)
```

По умолчанию в сообщениях журнала подключений отображается только IP-адрес подключающегося хоста. Включение этого параметра также приводит к регистрации имени хоста. Обратите внимание, что в зависимости от настройки разрешения имени вашего хоста это может привести к значительному снижению производительности. Этот параметр можно установить только в файле конфигурации или в командной строке сервера.

```
log_line_prefix(string)
```

Это строка в стиле printf, которая выводится в начале каждой строки журнала, начиная с «escape-последовательности» которая заменяется информацией о состоянии, как указано ниже. Нераспознанные ветки игнорируются. Другие символы копируются прямо в строку журнала. Информация о состоянии может быть выравнена влево или вправо, указав числовой литерал % после и перед параметром. Отрицательное значение приведет к тому, что информация о состоянии будет дополнена справа пробелами, чтобы дать ей минимальную ширину, тогда как

положительное значение будет дополнено слева. Заполнение может быть полезно для облегчения восприятия человеком файлов журнала. Этот параметр можно установить только в файле конфигурации или в командной строке сервера. По умолчанию в '%m [%p]' записывается метка времени и идентификатор процесса.

Таблица 3. Значения идентификаторов

Значение	Эффект	Только сессия
1	2	3
%a	Имя приложения	да
%u	Имя пользователя	да
%d	Имя базы данных	да
%r	Имя удаленного хоста или IP-адрес, а также удаленный порт	да
%h	Имя удаленного хоста или IP-адрес	да
%p	Идентификатор процесса	нет
%t	Отметка времени без миллисекунд	нет
%m	Отметка времени с миллисекундами	нет
%n	Отметка времени с миллисекундами (как эпоха Unix)	нет
%i	Командный тег: тип текущей команды сеанса	да
%e	Код ошибки SQLSTATE	нет
%c	Идентификатор сеанса: см. Ниже	нет
%l	Номер строки журнала для каждого сеанса или процесса, начиная с 1	нет
%s	Отметка времени начала процесса	нет
%v	Идентификатор виртуальной транзакции (backendID / localXID)	нет
%x	ID транзакции (0, если не назначен)	нет
%q	Не выдает никаких выходных данных, но указывает несессионным процессам остановиться в этой точке строки; игнорируется сессионными процессами	нет
%%	буквальный %	нет

Возвращенное значение %c выводит псевдоуникальный идентификатор сеанса, состоящий из двух 4-байтовых шестнадцатеричных чисел (без ведущих нулей), разделенных точкой. Числа - это время начала процесса и идентификатор процесса, поэтому %c также можно использовать как способ экономии места для печати этих элементов. Например, чтобы сгенерировать идентификатор сеанса из pg_stat_activity, используйте этот запрос:

```
SELECT to_hex (trunc (EXTRACT (EPOCH FROM backend_start)) :: integer) || '' ||
 to_hex (PID)
FROM pg_stat_activity;
```

```
log_lock_waits(boolean)
```

Создает сообщения журнала, когда сеанс ожидает получение блокировки дольше, чем deadlock_timeout. Это полезно при определении того, вызывает ли ожидание блокировки низкую

производительность. По умолчанию это off. Только администраторы могут изменять этот параметр.

```
log_statement(enum)
```

Управляет тем, какие операторы SQL регистрируются. Допустимые значения none (в выключенном состоянии), ddl, mod и all (все операторы). DDL регистрирует все операторы определения данных, такие как CREATE, ALTER и DROP. DML регистрирует все манипуляции с данными, таких как INSERT, UPDATE, DELETE, TRUNCATE, и COPY FROM. Операторы PREPARE, EXECUTE и EXPLAIN ANALYZE также регистрируются, если содержащиеся в них команды имеют соответствующий тип.

По умолчанию имеет значение none. Только администраторы могут изменять этот параметр.

Операторы, содержащие простые синтаксические ошибки, не регистрируются даже с помощью параметра log_statement= all, поскольку сообщение журнала отправляется только после того, как был выполнен базовый анализ для определения типа оператора. В случае протокола расширенного запроса этот параметр также не регистрирует операторы, которые не достигли фазы выполнения (т.е. находятся на этапе анализа или планирования). Установите log_min_error_statement на ERROR (или ниже), чтобы найти такие операторы.

```
log_replication_commands(boolean)
```

Заставляет каждую команду репликации регистрироваться в журнале сервера. Значением по умолчанию является off. Только администраторы могут изменять этот параметр.

```
log_temp_files(integer)
```

Управляет регистрацией временных имен и размеров файлов. Временные файлы могут быть созданы для сортировки, хэшей и результатов временных запросов. Запись журнала создается для каждого временного файла при его удалении. Нулевое значение записывает всю информацию временного файла, в то время как положительные значения регистрируют только файлы, размер которых больше или равен указанному количеству килобайт. Значением по умолчанию является -1, что запрещает такую регистрацию. Только администраторы могут изменять эту настройку.

```
log_timezone(string)
```

Устанавливает часовой пояс, используемый для отметок времени, записанных в журнале сервера. В отличие от TimeZone, это значение распространяется на весь кластер, поэтому все сеансы будут сообщать метки времени единообразно. Встроенное значение по умолчанию есть GMT, но обычно оно переопределяется файлом конфигурации; initdb установит параметр, соответствующий его системной среде. Этот параметр можно установить только в файле конфигурации или в командной строке сервера.

```
CREATE TABLE postgres_log
(
 log_time timestamp(3) with time zone,
```

```
 user_name text,
 database_name text,
 process_id integer,
 connection_from text,
 session_id text,
 session_line_num bigint,
 command_tag text,
 session_start_time timestamp with time zone,
 virtual_transaction_id text,
 transaction_id bigint,
 error_severity text,
 sql_state_code text,
 message text,
 detail text,
 hint text,
 internal_query text,
 internal_query_pos integer,
 context text,
 query text,
 query_pos integer,
 location text,
 application_name text,
 PRIMARY KEY (session_id, session_line_num)
);
```

Чтобы импортировать файл журнала в эту таблицу, используйте COPY FROM команду:

```
COPY postgres_log FROM '/full/path/to/logfile.csv' WITH csv;
```

Есть несколько вещей, которые нужно сделать, чтобы упростить импорт файлов журналов CSV:

Установите log_filename и log_rotation_age и составьте согласованную схему именования для ваших файлов журнала. Это позволит вам предсказать, каким будет имя файла, и узнать, когда отдельный файл журнала будет завершен и, следовательно, готов к импорту.

Установите log_rotation_size в значение 0, чтобы отключить ротацию журнала по размеру, поскольку это затрудняет прогнозирование имени файла журнала.

Перевод log_truncate_on_rotation в значение on определяет настройку так, что старые данные журнала не смешивается с новым в том же файле.

Приведенное выше определение таблицы содержит спецификацию первичного ключа. Это полезно для защиты от случайного импорта одной и той же информации дважды. Команда COPY фиксирует все импортируемые данные за один раз, поэтому любая ошибка приведет к сбою всего импорта. Если вы импортируете частичный файл журнала, а затем снова импортируете файл после его завершения, нарушение первичного ключа приведет к сбою импорта. Дождитесь завершения и закрытия журнала, прежде чем импортировать. Эта процедура также защитит от случайного импорта неполной строки, которая была написана не полностью, что также может привести COPY к сбою.

2.3.6.4. Заголовок процесса

Эти параметры определяют, как изменяются заголовки процессов сервера. Названия процессов обычно просматриваются с помощью таких программ, как ps.

```
cluster_name(string)
```

Задает имя кластера, которое отображается в заголовке процесса для всех процессов сервера в этом кластере. Имя может быть любой строкой длиной NAMEDATALEN, 64 символа в стандартной сборке. В значении cluster_name могут использоваться только печатные символы ASCII. Другие символы будут заменены на вопросительные знаки (?). Имя не отображается, если для этого параметра задана пустая строка ' ' (по умолчанию). Этот параметр можно установить только при запуске сервера.

```
update_process_title(boolean)
```

Включает обновление заголовка процесса каждый раз, когда сервер получает новую команду SQL. Этот параметр по умолчанию установлен в значение on. Только администраторы могут изменять этот параметр.

2.3.7. Статистика выполнения

2.3.7.1. Сборщик статистики запросов и индексов

Эти параметры управляют функциями сбора статистики на уровне сервера. Когда сбор статистики включен, производимые данные могут быть доступны через pg_stat и pg_statio - семейства системных представлений.

```
track_activities(boolean)
```

Включает сбор информации о текущей выполняемой команде каждого сеанса, а также время, когда эта команда начала выполнение. Этот параметр включен по умолчанию. Обратите внимание, что даже когда эта опция включена, эта информация видна не всем пользователям, а только администратору и пользователю, которому принадлежит сеанс, о котором сообщается, поэтому она не должна представлять угрозу безопасности. Только администраторы могут изменять эту настройку.

```
track_activity_query_size(integer)
```

Задает количество байтов, зарезервированных для отслеживания текущей выполняемой команды для каждого активного сеанса, для pg_stat_activity. queryполе. Значение по умолчанию - 1024. Этот параметр можно установить только при запуске сервера.

```
track_counts(boolean)
```

Включает сбор статистики о работе базы данных. Этот параметр включен по умолчанию, поскольку службе autovacuum требуется собранная информация. Только администраторы могут изменять эту настройку.

```
track_io_timing(boolean)
```

Включает синхронизацию вызовов ввода-вывода базы данных. Этот параметр по умолчанию отключен, поскольку он будет периодически запрашивать у операционной системы текущее время, что может вызвать значительные издержки на некоторых платформах. Вы можете использовать инструмент pg_test_timing для измерения затрат времени в вашей системе. Информация о времени ввода / вывода отображается в pg_stat_database, в выходных данных EXPLAIN, когда опция BUFFERS используется, и в pg_stat_statements. Только администраторы могут изменять этот параметр.

```
track_functions(enum)
```

Включает отслеживание количества вызовов функций и использованного времени. Укажите pl, чтобы отслеживать только функции процедурного языка, а all – чтобы также отслеживать функции языка SQL и C. Значение по умолчанию – none - отключение отслеживания статистики функций. Только администраторы могут изменять этот параметр.

Функции языка SQL, которые достаточно просты для «встраивания» в вызывающий запрос, отслеживаться не будут, независимо от этого параметра.

```
stats_temp_directory(string)
```

Устанавливает каталог для хранения временных данных статистики. Это может быть путь относительно каталога данных или абсолютный путь. По умолчанию это pg_stat_tmp. Указание этого значения на файловую систему на основе ОЗУ снизит требования к физическому вводу/выводу и может привести к повышению производительности. Этот параметр можно установить только в файле конфигурации или в командной строке сервера.

2.3.7.2. Мониторинг статистики

```
log_statement_stats(boolean)
log_parser_stats(boolean)
log_planner_stats(boolean)
log_executor_stats(boolean)
```

Для каждого запроса выводит статистику производительности соответствующего модуля в журнал сервера. Это грубый инструмент профилирования, похожий на средство getrusage() операционной системы Unix. Опция log_statement_stats сообщает общую статистику выписок, а остальные - статистику по модулям. Опция log_statement_stats не может быть включена вместе с любой из опций для модуля. Все эти параметры по умолчанию отключены. Только администраторы могут изменять эти настройки.

2.3.8. Автоматическая очистка

Настройки, описанные в данном пункте, управляют поведением функции автоочистки. Обратите внимание, что многие из этих настроек могут быть переопределены для каждой таблицы отдельно.

```
autovacuum(boolean)
```

Определяет условия, при которых сервер должен запускать службу автоочистки. По умолчанию включено; однако для работы автоочистки также необходимо включить track_counts. Этот параметр может быть установлен только в файле конфигурации или в командной строке сервера; однако автоматическая очистка может быть отключена для отдельных таблиц путем изменения параметров хранения таблиц.

Обратите внимание, что даже если этот параметр отключен, система будет запускать процессы автоочистки, если это необходимо, чтобы предотвратить обход идентификатора транзакции.

```
log_autovacuum_min_duration(integer)
```

Заставляет регистрировать каждое действие, выполненное автоочисткой, если оно выполняется по крайней мере указанное количество миллисекунд. Установка нулевого значения регистрирует все действия автоочистки. Значение -1 (по умолчанию) отключает ведение журнала действий автоочистки. Например, если вы установите это значение, 250ms будут регистрироваться все сеансы автоматической очистки и анализа, которые выполняются в течение 250 мс или более. Кроме того, если для этого параметра установлено любое значение, кроме -1, сообщение будет записано в журнал, если действие автоочистки пропущено из-за конфликтующей блокировки или одновременного удаления отношения. Включение этого параметра может быть полезным для отслеживания активности автоочистки. Этот параметр может быть установлен только в файле конфигурации или в командной строке сервера; но параметр может быть переопределен для отдельных таблиц путем изменения параметров хранения таблиц.

```
autovacuum_max_workers(integer)
```

Задает максимальное количество процессов автоочистки (кроме запуска автоочистки), которые могут выполняться одновременно. По умолчанию три. Этот параметр можно установить только при запуске сервера.

```
autovacuum_naptime(integer)
```

Определяет минимальную задержку между запусками автоочистки в любой заданной базе данных. В каждом интервале запуска служба проверяет базу данных и выдает команды VACUUM и ANALYZE команды, необходимые для таблиц в этой базе данных. Задержка измеряется в секундах и по умолчанию составляет одну минуту (1min). Этот параметр можно установить только в файле конфигурации или в командной строке сервера.

```
autovacuum_vacuum_threshold(integer)
```

Задает минимальное количество обновленных или удаленных кортежей, необходимое для запуска одной таблицы VACUUM. По умолчанию используется 50 кортежей. Этот параметр может быть установлен только в файле конфигурации или в командной строке сервера; но параметр может быть переопределен для отдельных таблиц путем изменения параметров хранения таблиц.

```
autovacuum_analyze_threshold(integer)
```

Задает минимальное количество вставленных, обновленных или удаленных кортежей, необходимое для запуска ANALYZE одной таблицы. По умолчанию используется 50 кортежей. Этот параметр может быть установлен только в файле конфигурации или в командной строке сервера; но параметр может быть переопределен для отдельных таблиц путем изменения параметров хранения таблиц.

```
autovacuum_vacuum_scale_factor(floating point)
```

Определяет часть размера таблицы, которую нужно добавить autovacuum_vacuum_threshold при принятии решения о запуске VACUUM. По умолчанию используется значение 0,2 (20% от размера таблицы). Этот параметр может быть установлен только в файле конфигурации или в командной строке сервера; но параметр может быть переопределен для отдельных таблиц путем изменения параметров хранения таблиц.

```
autovacuum_analyze_scale_factor(floating point)
```

Определяет часть размера таблицы, которую нужно добавить autovacuum_analyze_threshold при принятии решения о запуске ANALYZE. По умолчанию используется значение 0,1 (10% от размера таблицы). Этот параметр может быть установлен только в файле конфигурации или в командной строке сервера; но параметр может быть переопределен для отдельных таблиц путем изменения параметров хранения таблиц.

```
autovacuum_freeze_max_age(integer)
```

Указывает максимальный возраст (в транзакциях) для таблицы pg_class. Поле Relfrozenxid может быть достигнуто до того, как операция VACUUM будет вынуждена предотвратить обход идентификатора транзакции в таблице. Обратите внимание, что система будет запускать процессы автоочистки, чтобы предотвратить циклическое изменение, даже если автоочистка отключена.

Очистка также позволяет удалять старые файлы из подкаталога pg_xact, поэтому по умолчанию это относительно низкие 200 миллионов транзакций. Этот параметр можно установить только при запуске сервера, но его можно уменьшить для отдельных таблиц, изменив параметры хранения таблиц.

```
autovacuum_multixact_freeze_max_age(integer)
```

Указывает максимальный возраст (в мультитранзакциях) для таблицы pg_class. Поле Relminmxid может быть достигнуто до того, как операция VACUUM будет вынуждена

предотвратить многофакторный перенос идентификатора в таблице. Обратите внимание, что система будет запускать процессы автоочистки, чтобы предотвратить циклическое изменение, даже если автоочистка отключена.

```
autovacuum_vacuum_cost_delay(integer)
```

Задает значение задержки стоимости, которое будет использоваться в автоматических операциях очистки. Если указано значение -1, будет использоваться обычное значение autovacuum_vacuum_cost_delay. Значение по умолчанию составляет 20 миллисекунд. Этот параметр может быть установлен только в файле конфигурации или в командной строке сервера, или в командной строке сервера; но параметр может быть переопределен для отдельных таблиц путем изменения параметров хранения таблиц.

```
autovacuum_vacuum_cost_limit(integer)
```

Указывает предельное значение стоимости, которое будет использоваться в автоматических операциях VACUUM. Если указано значение -1 (которое используется по умолчанию), будет использоваться обычное значение uum_cost_limit. Обратите внимание, что это значение пропорционально распределяется между работающими процессами автоочистки, если их больше одного, поэтому сумма пределов для каждого процесса не превышает значения этой переменной. Этот параметр может быть установлен только в файле конфигурации или в командной строке сервера; но параметр может быть переопределен для отдельных таблиц путем изменения параметров хранения таблиц.

2.3.9. Клиентские соединения по умолчанию

2.3.9.1. Поведение команд

```
client_min_messages(enum)
```

Управляет тем, какие уровни сообщений отправляются клиенту. Допустимые значения DEBUG5, DEBUG4, DEBUG3, DEBUG2, DEBUG1, LOG, NOTICE, WARNING, и ERROR. Каждый уровень включает в себя все уровни, которые следуют за ним. Чем позже уровень, тем меньше сообщений отправляется. По умолчанию это NOTICE. Обратите внимание, что здесь LOG имеет другой ранг, чем в log_min_messages .

Сообщения INFO уровня всегда отправляются клиенту.

```
search_path(string)
```

Эта переменная указывает порядок поиска схем, когда на объект (таблицу, тип данных, функцию и т.д.) Ссылается простое имя без указания схемы. Когда в разных схемах есть объекты с одинаковыми именами, используется тот, который был найден первым в пути поиска. На объект, который не входит ни в одну из схем в пути поиска, можно ссылаться только путем указания его содержащей схемы с квалифицированным (пунктирным) именем.

Значение для search_path должно быть разделенным запятыми списком имен схем. Любое имя, которое не является существующей схемой или является схемой, для которой пользователь не имеет USAGE разрешения, игнорируется.

Если один из элементов списка является специальным именем $user, то схема, имя которой возвращается, CURRENT_USER заменяется, если такая схема существует, и у пользователя есть разрешение USAGE на нее (если нет, $user игнорируется).

Схема системного каталога pg_catalog всегда ищется независимо от того, упоминается она в пути или нет. Если она упомянута в пути, то она будет найдена в указанном порядке. Если схемы pg_cataloge нет в пути, она будет найдена перед поиском любого из элементов пути.

Аналогично, схема временной таблицы текущего сеанса всегда ищется, если она существует. Он может быть явно указан «в пути», используя псевдоним pg_temp_nnn pg_temp. Если он не указан «в пути», то он ищется первым (даже раньше pg_catalog). Однако во временной схеме ищутся только имена отношений (таблица, представление, последовательность и т.д.) и имена типов данных. Никогда не ищется имя функции или оператора.

Когда объекты создаются без указания конкретной целевой схемы, они будут помещены в первую действительную схему с именем in search_path. Сообщение об ошибке появляется, если путь поиска пуст.

Значением по умолчанию для этого параметра является "$user", public. Этот параметр поддерживает совместное использование базы данных (где ни у одного пользователя нет частных схем, и у всех есть общее использование public), частных схем для каждого пользователя и их комбинаций. Другие эффекты можно получить, изменив настройку пути поиска по умолчанию, либо глобально, либо для каждого пользователя.

Конфигурация по умолчанию подходит только в том случае, если в БД работает один пользователь или несколько доверяющих друг другу пользователей.

Текущее эффективное значение пути поиска можно проверить с помощью SQL функции current_schemas. Это не совсем то же самое, что изучение значения search_path, поскольку current_schemas показывает, как были разрешены элементы, появляющиеся в search_path.

```
row_security(boolean)
```

Эта переменная определяет, будет ли возникать ошибка вместо применения политики безопасности строк. Когда установлено значение on, политики применяются корректно. Когда установлено значение off, запросы не выполняются, что в противном случае применило бы по крайней мере одну политику. По умолчанию установлено значение on.

```
default_tablespace(string)
```

Эта переменная указывает табличное пространство по умолчанию, в котором создаются объекты (таблицы и индексы), когда команда CREATE явно не указывает табличное пространство.

Значением является либо имя табличного пространства, либо пустая строка, указанная с использованием табличного пространства по умолчанию для текущей базы данных. Если значение не соответствует имени какого-либо существующего табличного пространства, СУБД-КС

автоматически будет использовать табличное пространство по умолчанию для текущей базы данных. Если указано табличное пространство не по умолчанию, то у пользователя должна быть привилегия на CREATE в этом пространстве, иначе попытки создания приведут к ошибкам.

Эта переменная не используется для временных таблиц; для них вместо этого используется temp_tablespaces.

Эта переменная также не используется при создании баз данных. По умолчанию новая база данных наследует настройки табличного пространства из базы данных шаблонов, из которой она скопирована.

```
temp_tablespaces(string)
```

Эта переменная определяет табличные пространства, в которых создаются временные объекты (временные таблицы и индексы для временных таблиц), когда команда CREATE явно не указывает табличное пространство. Временные файлы для таких целей, как сортировка больших наборов данных, также создаются в этих табличных пространствах.

Значение представляет собой список имен табличных пространств. Когда в списке более одного имени, СУБД-КС выбирает случайную запись списка каждый раз, когда создается временный объект; за исключением того, что внутри транзакции последовательно созданные временные объекты размещаются в последовательных табличных пространствах из списка. Если выбранный элемент списка представляет собой пустую строку, СУБД-КС автоматически будет использовать табличное пространство по умолчанию для текущей базы данных.

Когда temp_tablespaces задано интерактивно, указание несуществующего табличного пространства является ошибкой, равно как и указание табличного пространства, для которого у пользователя нет привилегий CREATE. Однако при использовании ранее установленного значения несуществующие табличные пространства игнорируются, как и табличные пространства, для которых у пользователя нет привилегий CREATE. В частности, это правило применяется при использовании значения, установленного в файле конфигурации.

Значением по умолчанию является пустая строка, в результате чего все временные объекты создаются в табличном пространстве по умолчанию текущей базы данных.

```
check_function_bodies(boolean)
```

Этот параметр обычно включен. Когда установлено значение off, оно отключает проверку строки тела функции во время оператора CREATE FUNCTION. Отключение проверки позволяет избежать побочных эффектов процесса проверки и избежать ложных срабатываний из-за проблем, таких как прямые ссылки. Установите этот параметр в off перед загрузкой функций от имени других пользователей; pg_dump делает это автоматически.

```
default_transaction_isolation(enum)
```

Каждая транзакция SQL имеет уровень изоляции, который может быть: «read uncommitted», «read committed», «repeatable read» или «serializable». Этот параметр контролирует уровень изоляции по умолчанию для каждой новой транзакции. По умолчанию «read committed».

```
default_transaction_read_only(boolean)
```

Транзакция SQL только для чтения не может изменять временные таблицы. Этот параметр контролирует состояние по умолчанию только для чтения каждой новой транзакции. По умолчанию имеет значение off (на чтение / запись).

```
default_transaction_deferrable(boolean)
```

При работе на уровне изоляции «serializable» отложенная транзакция SQL, доступная только для чтения, может быть отложена до того, как ей будет разрешено продолжить. Однако, как только он начинает выполняться, она не несет никаких накладных расходов, необходимых для обеспечения сериализуемости; поэтому у кода сериализации не будет причин принудительно прерывать его из-за одновременных обновлений, что делает эту опцию подходящей для длительных транзакций «только для чтения».

Этот параметр управляет отложенным статусом по умолчанию для каждой новой транзакции. В настоящее время это не влияет на транзакции «чтения-записи» или те, которые работают на уровнях изоляции ниже, чем serializable. По умолчанию имеет значение off.

```
session_replication_role(enum)
```

Управляет срабатыванием связанных с репликацией триггеров и правил для текущего сеанса. Установка этой переменной требует привилегий администратора и приводит к отмене любых ранее кэшированных планов запросов. Возможные значения origin (по умолчанию), replica и local.

Предполагаемое использование этого параметра заключается в том, что системы логической репликации устанавливают replica, когда они применяют реплицированные изменения. В результате триггеры и правила (которые не были изменены из конфигурации по умолчанию) не будут срабатывать на реплике.

СУБД-КС рассматривает origin и local настройки. Сторонние системы репликации могут использовать эти два значения для своих внутренних целей, например, local для обозначения сеанса, изменения которого не должны реплицироваться.

Поскольку внешние ключи реализованы как триггеры, установка параметра replica также отключает все проверки внешних ключей, которые могут привести к несогласованности данных в случае неправильного использования.

```
statement_timeout(integer)
```

Прервать любой оператор, который занимает больше указанного количества миллисекунд, начиная с момента поступления команды на сервер от клиента. Если log_min_error_statement установлено в значение ERROR или ниже, то заявление о тайм-ауте также будет зарегистрировано. Нулевое значение (по умолчанию) отключает это.

Установка statement_timeout в файле конфигурации не рекомендуется, потому что это повлияет на все сеансы.

```
lock_timeout(integer)
```

Прерывает любой оператор, который ожидает дольше указанного количества миллисекунд при попытке получить блокировку таблицы, индекса, строки или другого объекта базы данных. Ограничение по времени применяется отдельно для каждой попытки получения блокировки. Ограничение применяется как к явным запросам на блокировку (таким как LOCK TABLE, так и SELECT FOR UPDATE без NOWAIT) и к неявно полученным блокировкам. Если в log_min_error_statement установлено значение ERROR или ниже, то заявление об истечении времени ожидания будет зарегистрировано. Нулевое значение (по умолчанию) отключает это.

В отличие от statement_timeout, это время ожидания может происходить только в ожидании блокировки. Обратите внимание, что если значение statement_timeout ненулевое, устанавливать одно и то же или большее значение lock_timeout бессмысленно, так как время ожидания оператора всегда срабатывает первым.

Установка lock_timeout в файле конфигурации не рекомендуется, потому что это повлияет на все сеансы.

```
idle_in_transaction_session_timeout(integer)
```

Завершите любой сеанс открытой транзакцией, которая простаивала дольше указанной длительности в миллисекундах. Это позволяет снять любые блокировки, удерживаемые этим сеансом, и повторно использовать слот подключения; это также позволяет выполнить «вакуум» кортежей, видимых только для этой транзакции.

Значение по умолчанию 0 отключает эту функцию.

```
vacuum_freeze_table_age(integer)
```

Очистка выполняет агрессивное сканирование таблицы pg_class, если поле Relfrozenxid достигло возраста, указанного в этом параметре. Агрессивное сканирование отличается от обычной очистки тем, что оно посещает каждую страницу, которая может содержать «незамерзающие» XID или MXID, а не только те, которые могут содержать «мертвые» кортежи. По умолчанию имеет значение 150 миллионов транзакций. Хотя пользователи могут установить это значение в диапазоне от нуля до двух миллиардов, очистка будет незаметно ограничивать эффективное значение до 95% autovacuum_freeze_max_age, чтобы принудительная автоочистка выполнилась прежде чем для таблицы будет запущена автоочистка для предотвращения зацикливания.

```
vacuum_freeze_min_age(integer)
```

Указывает возраст ограничения (в транзакциях), который следует использовать очистке, чтобы решить, следует ли заморозить версии строк при сканировании таблицы. По умолчанию 50 миллионов транзакций. Хотя пользователи могут установить это значение в диапазоне от нуля до одного миллиарда, VACUUM будет незаметно ограничивать эффективное значение половиной значения autovacuum_freeze_max_age, чтобы между принудительными автоочистками не было необоснованно короткого времени.

```
vacuum_multixact_freeze_table_age(integer)
```

Очистка выполняет агрессивное сканирование таблицы pg_class, если поле Relminmxid достигло возраста, указанного в этом параметре. Агрессивное сканирование отличается от

обычной очистки тем, что оно посещает каждую страницу, которая может содержать «незамерзающие» XID или MXID, а не только те, которые могут содержать «мертвые кортежи». По умолчанию установлено 150 миллионов мультитранзакций. Хотя пользователи могут установить это значение в диапазоне от нуля до двух миллиардов, очистка будет незаметно ограничивать эффективное значение до 95% от autovacuum_multixact_freeze_max_age, чтобы принудительная автоочистка выполнилась прежде, чем для таблицы будет запущена автоочистка для предотвращения зацикливания.

```
vacuum_multixact_freeze_min_age(integer)
```

Указывает возраст ограничения (в мультифакторах), при котором VACUUM следует использовать, чтобы решить, заменять ли многофакторные идентификаторы более новым идентификатором транзакции или многофакторным идентификатором при сканировании таблицы. По умолчанию используется 5 миллионов мультифакторов. Хотя пользователи могут установить это значение в диапазоне от нуля до одного миллиарда, очистка будет незаметно ограничивать действующее значение половиной значения autovacuum_multixact_freeze_max_age, чтобы принудительная автоочистка не выполнялась слишком часто.

```
vacuum_cleanup_index_scale_factor(floating point)
```

Указывает долю от общего числа кортежей кучи, подсчитанных в предыдущем сборе статистики, которые можно вставить без сканирования индекса на этапе очистки VACUUM. Этот параметр в настоящее время применяется только к индексам B-дерева.

Если никакие кортежи не были удалены из кучи, индексы B-дерева все еще сканируются на этапе очистки VACUUM, когда выполняется хотя бы одно из следующих условий: статистика индекса устарела или индекс содержит удаленные страницы, которые могут быть переработаны во время очистки. Статистика индекса считается устаревшей, если число вновь вставленных кортежей превышает долю vacuum_cleanup_index_scale_factor от общего числа кортежей кучи, обнаруженных предыдущим сбором статистики. Общее количество кортежей кучи хранится на мета-странице индекса. Обратите внимание, что мета-страница не включает эти данные, пока очистка VACUUM не найдет мертвые кортежи, поэтому сканирование индекса B-дерева на этапе очистки может быть пропущено только в том случае, если второй и последующие VACUUM-циклы не обнаружат «мертвых» кортежей.

Значение может варьироваться от 0 до 10000000000. Когда vacuum_cleanup_index_scale_factor установлено в значение 0, сканирование индекса никогда не пропускается во время очистки VACUUM. Значением по умолчанию является 0.1.

```
bytea_output(enum)
```

Устанавливает выходной формат для значений типа bytea. Допустимые значения: hex (по умолчанию) и escape. См. Раздел 8.4 для получения дополнительной информации. Тип Bytea всегда принимает оба формата на входе, независимо от значения этого параметра.

```
xmlbinary(enum)
```

Устанавливает, как двоичные значения должны быть закодированы в XML. Это относится, например, когда bytea значения преобразуются в XML с помощью функций xmlelement или xmlforest. Возможны значения base64 и hex, которые определены в стандарте XML-схемы.

Фактический выбор здесь в основном зависит от вкуса, ограниченного только возможными ограничениями в клиентских приложениях. Оба метода поддерживают все возможные значения, хотя шестнадцатеричная кодировка будет несколько больше, чем кодировка base64.

```
xmloption(enum)
```

Устанавливает, является ли DOCUMENT или CONTENT неявным при преобразовании между значениями XML и символьных строк. Допустимые значения DOCUMENT и CONTENT. По умолчанию установлено CONTENT.

В соответствии со стандартом SQL, команда для установки этой опции:

```
SET XML OPTION { DOCUMENT | CONTENT }
```

Этот синтаксис также доступен в СУБД-КС.

```
gin_pending_list_limit(integer)
```

Устанавливает максимальный размер списка ожидания GIN, который используется, когда fastupdate включен. Если список становится больше этого максимального размера, он очищается путем массового перемещения записей в нем в основную структуру данных GIN. По умолчанию используется четыре мегабайта (4MB). Этот параметр можно переопределить для отдельных индексов GIN, изменив параметры хранения индекса.

2.3.9.2. Языковая среда и форматирование

```
DateStyle(string)
```

Устанавливает формат отображения значений даты и времени, а также правила интерпретации неоднозначных значений ввода даты. По историческим причинам, эта переменная содержит два независимых компонента: спецификации формата вывода (ISO, Postgres, SQL или German) и ввода - вывода спецификации / для заказа год месяц / день / (DMY, MDY или YMD). Они могут быть установлены отдельно или вместе. Ключевые слова Eurои European являются синонимами для DMY; ключевые слова US, NonEuro и NonEuropean являются синонимами MDY. Встроенное значегние по умолчанию есть ISO, MDY, но initdb инициализирует файл конфигурации с настройкой, соответствующей поведению выбранной lc_time - локали.

```
IntervalStyle(enum)
```

Устанавливает формат отображения для значений интервала. Значение sql_standard выдаст выходные данные, соответствующие литералам стандартного интервала SQL.

Параметр IntervalStyle также влияет на интерпретацию неоднозначного ввода интервала.

```
TimeZone(string)
```

Устанавливает часовой пояс для отображения и интерпретации меток времени. Встроенное значение по умолчанию есть GMT, но обычно оно переопределяется файлом конфигурации; initdb установит параметр, соответствующий его системной среде.

```
timezone_abbreviations(string)
```

Устанавливает коллекцию сокращений часовых поясов, которые будут приниматься сервером для ввода даты и времени. По умолчанию 'Default' - это коллекция, которая работает в большинстве стран мира.

```
extra_float_digits(integer)
```

Этот параметр регулирует количество цифр, отображаемых для чисел с плавающей запятой значений, в том числе float4, float8 и геометрических типов данных. Значение параметра добавляется к стандартному количеству цифр (FLT_DIG или DBL_DIG по мере необходимости). Значение может быть установлено до 3, чтобы включить частично значимые цифры; это особенно полезно для сброса данных с плавающей запятой, которые необходимо точно восстановить. Или это может быть установлено отрицательным, чтобы подавить нежелательные цифры.

```
client_encoding(string)
```

Устанавливает кодировку на стороне клиента (набор символов). По умолчанию используется кодировка БД.

```
lc_messages(string)
```

Устанавливает язык, на котором отображаются сообщения. Допустимые значения зависят от системы. Если для этой переменной задана пустая строка (которая является значением по умолчанию), то значение наследуется от среды выполнения сервера системно-зависимым способом.

Только администраторы могут изменять этот параметр, поскольку он влияет на сообщения, отправляемые в журнал сервера, а также клиенту.

```
lc_monetary(string)
```

Устанавливает языковой стандарт для форматирования денежных сумм, например, с семейством функций to_char.

```
lc_numeric(string)
```

Устанавливает локаль для форматирования чисел, например, с семейством функций to_char.

```
lc_time(string)
```

Устанавливает языковой стандарт для форматирования даты и времени, например, с помощью семейства функций to_char.

```
default_text_search_config(string)
```

Выбирает конфигурацию текстового поиска, которая используется теми параметрами функций текстового поиска, у которых нет явного аргумента, задающего конфигурацию.

2.3.9.3. Предварительная загрузка разделяемых библиотек

Для предварительной загрузки разделяемых библиотек в память сервера доступно несколько параметров. Значение этих параметров задается однотипно. Например, установка параметра '$libdir/mylib' будет загружать в память mylib.so из стандартного каталога установки сервера. Различия между настройками заключаются в том, когда они вступают в силу, и какие привилегии требуются для их изменения.

Библиотеки процедурного языка могут быть предварительно загружены с использованием синтаксиса, '$libdir/plXXX', где XXX есть pgsql, perl, tcl или python.

Только разделяемые библиотеки, специально предназначенные для использования с СУБД-КС, могут быть загружены таким образом. Каждая библиотека, поддерживаемая СУБД-КС, имеет «специальный блок», который проверяется для обеспечения совместимости. По этой причине не-СУБД-КС библиотеки не могут быть загружены таким образом. Вы можете для этого использовать средства операционной системы, например, LD_PRELOAD.

```
local_preload_libraries(string)
```

Эта переменная указывает одну или несколько разделяемых библиотек, которые должны быть предварительно загружены при запуске подключения. Он содержит разделенный запятыми список имен библиотек, где каждое имя обрабатывается командой LOAD . Пробелы между записями игнорируются; заключите имя библиотеки в двойные кавычки, если вам нужно включить в имя пробел или запятые. Значение параметра вступает в силу только в начале соединения. Последующие изменения не имеют никакого эффекта. Если указанная библиотека не найдена, попытка подключения завершится неудачно.

Эта опция может быть установлена любым пользователем. Из-за этого загружаемые библиотеки ограничиваются теми, которые появляются в plugins подкаталоге стандартного каталога библиотеки установки. (Администратор базы данных обязан убедиться, что установлены только «безопасные» библиотеки.) Записи в local_preload_libraries могут, например, явно указывать этот каталог $libdir/plugins/mylib или просто указывать имя библиотеки – mylib - будет иметь тот же эффект, что и в случае $libdir/plugins/mylib.

Назначение этой функции - позволить непривилегированным пользователям загружать библиотеки отладки или измерения производительности в конкретные сеансы, не требуя явной LOAD команды. Для этого было бы типично установить этот параметр с помощью PGOPTIONS переменной среды на клиенте или с помощью ALTER ROLE SET.

Однако если модуль специально не предназначен для такого использования пользователями, не являющимися администраторами, воспользуйтесь session_preload_libraries.

```
session_preload_libraries(string)
```

Эта переменная указывает одну или несколько разделяемых библиотек, которые должны быть предварительно загружены при запуске подключения. Он содержит разделенный запятыми список имен библиотек, где каждое имя обрабатывается командой LOAD. Пробелы между записями игнорируются; заключите имя библиотеки в двойные кавычки, если вам нужно включить в имя пробел или запятые. Значение параметра вступает в силу только в начале соединения. Последующие изменения не имеют никакого эффекта. Если указанная библиотека не найдена, попытка подключения завершится неудачно. Только администраторы могут изменять эту настройку.

Цель этой функции - позволить библиотекам отладки или измерения производительности загружаться в конкретные сеансы без явной команды LOAD. Например, auto_explain можно включить для всех сессий с данным именем пользователя, установив этот параметр с помощью ALTER ROLE SET. Кроме того, этот параметр можно изменить без перезапуска сервера (но изменения вступают в силу только при запуске нового сеанса), поэтому проще добавлять новые модули, даже если они должны применяться ко всем сеансам.

В отличие от shared_preload_libraries, нет большого преимущества в производительности при загрузке библиотеки при запуске сеанса, а не при его первом использовании. Однако при использовании пула соединений есть некоторое преимущество.

```
shared_preload_libraries(string)
```

Эта переменная указывает одну или несколько разделяемых библиотек, которые должны быть предварительно загружены при запуске сервера. Он содержит разделенный запятыми список имен библиотек, где каждое имя обрабатывается командой LOAD. Пробелы между записями игнорируются; заключите имя библиотеки в двойные кавычки, если вам нужно включить в имя пробел или запятые. Этот параметр можно установить только при запуске сервера. Если указанная библиотека не найдена, сервер не запустится.

Некоторые библиотеки должны выполнять определенные операции, которые могут выполняться только при запуске postmaster, такие как выделение общей памяти, резервирование легких блокировок или запуск рабочих фоновых процессов. Эти библиотеки должны быть загружены при запуске сервера через этот параметр. Смотрите документацию библиотеки для уточнения.

Также могут быть предварительно загружены другие разделяемые библиотеки. За счет предварительной загрузки общей библиотеки время запуска библиотеки сокращается при первом использовании библиотеки. Однако время запуска каждого нового серверного процесса может немного увеличиться, даже если этот процесс никогда не использует библиотеку. Поэтому этот параметр рекомендуется только для библиотек, которые будут использоваться в большинстве сеансов. Кроме того, изменение этого параметра требует перезапуска сервера, так что, скажем, это неправильная настройка для краткосрочных задач отладки. Вместо этого используйте для этого session_preload_libraries .

```
jit_provider(string)
```

Эта переменная является именем используемой библиотеки провайдера JIT. По умолчанию это llvmjit. Этот параметр можно установить только при запуске сервера.

Если задана несуществующая библиотека, JIT будет недоступен, но ошибки не возникнет. Это позволяет устанавливать поддержку JIT отдельно от основного пакета СУБД-КС.

2.3.9.4. Другие значения по умолчанию

```
dynamic_library_path(string)
```

Если необходимо открыть динамически загружаемый модуль и его имя, указанное в команде CREATE FUNCTION или LOAD, не имеет имени каталога (т.е. имя не содержит косую черту), система будет искать этот файл в данном пути.

Значение для dynamic_library_path должно быть списком абсолютных путей к каталогам, разделенных двоеточиями. Если элемент списка начинается со специальной строки $libdir, вместо него подставляется скомпилированный каталог библиотеки пакетов СУБД-КС; здесь устанавливаются модули, предоставляемые стандартным дистрибутивом СУБД-КС. (Используйте, pg_config –pkglibdir, чтобы узнать имя этого каталога.) Например:

```
dynamic_library_path = '/usr/local/lib/ pgsqlks-11.4:/home/my_project/lib: $ libdir'
```

Значением по умолчанию для этого параметра является '$libdir'. Если в качестве значения задана пустая строка, автоматический поиск пути отключается.

Этот параметр может быть изменен администраторами во время выполнения, но настройка, выполненная таким образом, будет сохраняться только до конца клиентского соединения, поэтому этот метод должен быть зарезервирован для целей разработки. Рекомендуемый способ установить этот параметр в файле конфигурации.

```
gin_fuzzy_search_limit(integer)
```

Мягкий верхний предел размера набора, возвращаемого при сканировании индекса GIN.

2.3.10. Управление блокировками

```
deadlock_timeout( integer)
```

Время ожидания блокировки (в миллисекундах), по истечении которого будет выполняться проверка состояния взаимоблокировки. Проверка на взаимоблокировку является относительно ресурсозатратной, поэтому сервер не запускает ее каждый раз, когда ожидает блокировки. Взаимные блокировки не распространены в рабочих приложениях. Увеличение этого значения сокращает время, затрачиваемое на ненужные проверки взаимоблокировок, но замедляет сообщение о реальных ошибках взаимоблокировок. По умолчанию одна секунда (1s), что, вероятно, является наименьшим значением, которое необходимо получить на практике. Рекомендуется, чтобы настройка превышала типичное время транзакции, чтобы повысить вероятность снятия блокировки до того, как запустится проверка. Только администраторы могут изменять эту настройку.

Когда установлен параметр log_lock_waits, этот параметр также определяет период ожидания, прежде чем будет выдано сообщение журнала об ожидании блокировки. Если вы пытаетесь исследовать задержки блокировки, вы можете установить более короткое значение deadlock_timeout, чем обычно.

```
max_locks_per_transaction( integer)
```

Таблица общих блокировок отслеживает блокировки на объектах max_locks_per_transaction * (max_connections + max_prepared_transactions) (например, на таблицах); следовательно, одновременно можно заблокировать не более указанного количества различных объектов. Этот параметр контролирует среднее количество блокировок объектов, выделенных для каждой транзакции; отдельные транзакции могут блокировать больше объектов, если блокировки всех транзакций помещаются в таблицу блокировок. Это не количество строк, которые могут быть заблокированы (этот параметр не ограничен). Значение по умолчанию, 64, исторически доказано достаточным, но вам может потребоваться увеличить это значение, если у вас есть запросы, которые касаются множества разных таблиц в одной транзакции, например, запрос родительской таблицы с множеством дочерних элементов. Этот параметр можно установить только при запуске сервера.

При запуске резервного сервера вы должны установить для этого параметра то же или более высокое значение, чем на главном сервере. В противном случае запросы не будут разрешены на резервном сервере.

```
max_pred_locks_per_transaction( integer)
```

Общая таблица блокировки предикатов отслеживает блокировки на объектах max_pred_locks_per_transaction* (max_connections + max_prepared_transactions) (например, на таблицах); следовательно, одновременно можно заблокировать не более указанного количества различных объектов. Этот параметр контролирует среднее количество блокировок объектов, выделенных для каждой транзакции; отдельные транзакции могут блокировать больше объектов, если блокировки всех транзакций помещаются в таблицу блокировок. Это не количество строк, которые могут быть заблокированы (этот параметр не ограничен). Стандартное значение 64 обычно достаточно для тестирования, но может потребоваться увеличить это значение, при наличии клиентов, которые касаются множества различных таблиц в одной сериализуемой транзакции. Этот параметр можно установить только при запуске сервера.

```
max_pred_locks_per_relation( integer)
```

Опция контролирует, сколько страниц или кортежей одного отношения может быть заблокировано предикатом, прежде чем блокировка будет повышена до полного отношения. Значения, большие или равные нулю, означают абсолютный предел, а отрицательные значения означают значение max_pred_locks_per_transaction, поделенное на абсолютное значение этого параметра. По умолчанию используется значение -2. Этот параметр можно установить только в файле конфигурации или в командной строке сервера.

```
max_pred_locks_per_page( integer)
```

Опция контролирует, сколько строк на одной странице может быть заблокировано предикатом, прежде чем блокировка будет переведена на всю страницу. Значение по умолчанию -

2. Этот параметр можно установить только в файле конфигурации или в командной строке сервера.

2.3.11. Совместимость версии и платформы

2.3.11.1. Совместимость платформы и клиента

```
transform_null_equals( boolean)
```

Когда этот параметр включён, проверки вида выражение = NULL (или NULL = выражение) воспринимаются как выражение IS NULL, то есть они истинны, если выражение даёт значение NULL, и ложны в противном случае. Согласно спецификации SQL, сравнение выражение = NULL должно всегда возвращать NULL (неизвестное значение). Поэтому по умолчанию этот параметр выключен (равен off).

2.3.12. Обработка ошибок

```
exit_on_error( boolean)
```

Если установлено значение true, любая ошибка прервет текущий сеанс. По умолчанию для этого параметра установлено значение false, поэтому только фатальные ошибки будут вызывать прерывание сеанса.

```
restart_after_crash( boolean)
```

Когда установлено значение true, которое является значением по умолчанию, СУБД-КС автоматически повторно инициализируется после сбоя. Установка значения true позволит максимизировать доступность базы данных. Однако в некоторых случаях, например, когда СУБД-КС вызывается кластерным программным обеспечением, может быть полезно отключить перезапуск, чтобы кластерное программное обеспечение могло получить контроль и предпринимать любые действия, которые оно сочтет целесообразными.

```
data_sync_retry( boolean)
```

Если установлено значение false, которое является значением по умолчанию, СУБД-КС выдаст ошибку уровня PANIC при невозможности сброса измененных файлов данных в файловую систему. Это приводит к сбою сервера базы данных. Этот параметр можно установить только при запуске сервера.

В некоторых операционных системах состояние данных в кеше страниц ядра после сбоя обратной записи неизвестно. Вторая попытка может быть объявлена успешной, когда фактически данные были потеряны. В этих обстоятельствах единственный способ избежать потери данных - это восстановление из WAL после того, как сообщается о любом сбое, предпочтительно после расследования основной причины сбоя и замены любого неисправного оборудования.

Если установлено значение true, СУБД-КС сообщит об ошибке, но продолжит работу, чтобы можно было повторить операцию сброса данных на более поздней контрольной точке. Установите значение true только после изучения обработки операционной системы буферизованных данных в случае сбоя обратной записи.

2.3.13. Предустановленные параметры

Следующие параметры доступны «только для чтения» и определяются при компиляции СУБД-КС или при его установке. Как таковые, они были исключены из базового файла конфигурации. Эти опции сообщают о различных аспектах поведения СУБД-КС, которые могут представлять интерес для определенных приложений, в частности, административных интерфейсов.

```
block_size( integer)
```

Сообщает размер блока диска. Это определяется значением BLCKSZ при построении сервера. Значение по умолчанию составляет 8192 байта. Значение некоторых переменных конфигурации (например, shared_buffers) зависит от block_size.

```
data_checksums( boolean)
```

Сообщает, включены ли контрольные суммы данных для этого кластера. Посмотрите контрольные суммы данных для получения дополнительной информации.

```
data_directory_mode( integer)
```

В системах Unix этот параметр сообщает о разрешениях каталога данных (data_directory), определенных при запуске. Смотрите групповой доступ для получения дополнительной информации.

```
lc_collate( string)
```

Сообщает локаль, в которой выполняется сортировка текстовых данных. Это значение определяется при создании базы данных.

```
lc_ctype( string)
```

Сообщает о локали, определяющей классификации символов. Это значение определяется при создании базы данных. Обычно это будет то же самое lc_collate, но для специальных приложений это может быть установлено по-другому.

```
max_function_args( integer)
```

Сообщает максимальное количество аргументов функции. Это определяется значением FUNC_MAX_ARGS при построении сервера. Значение по умолчанию составляет 100 аргументов.

```
max_identifier_length( integer)
```

Сообщает максимальную длину идентификатора. Он определяется на единицу меньше, чем значение NAMEDATALEN при построении сервера. Значение по умолчанию NAMEDATALEN-64; поэтому по умолчанию max_identifier_length используется 63 байта, что может быть меньше 63 символов при использовании многобайтовых кодировок.

```
max_index_keys( integer)
```

Сообщает максимальное количество индексных ключей. Это определяется значением INDEX_MAX_KEYS при построении сервера. Значение по умолчанию составляет 32 ключа.

```
segment_size( integer)
```

Сообщает количество блоков (страниц), которые могут быть сохранены в сегменте файла. Это определяется значением RELSEG_SIZE при построении сервера. Максимальный размер файла сегмента в байтах равен segment_size, умноженному на block_size; по умолчанию это 1 ГБ.

```
server_encoding( string)
```

Сообщает кодировку базы данных (набор символов). Определяется при создании базы данных. Обычно клиенты должны иметь дело только со значением client_encoding.

```
server_version( string)
```

Сообщает номер версии сервера. Это определяется значением PG_VERSION при построении сервера.

```
server_version_num( integer)
```

Сообщает номер версии сервера как целое число. Это определяется значением PG_VERSION_NUM при построении сервера.

```
wal_block_size( integer)
```

Сообщает размер блока диска WAL. Это определяется значением XLOG_BLCKSZ при построении сервера. Значение по умолчанию составляет 8192 байта.

```
wal_segment_size( integer)
```

Сообщает размер сегментов журнала записи вперед. Значение по умолчанию составляет 16 МБ.

2.3.14. Индивидуальные параметры

Эта функция была разработана для того, чтобы позволить параметрам, обычно не известным СУБД-КС, добавляться дополнительными модулями (такими как процедурные языки). Это позволяет настраивать модули расширения стандартными способами.

Имена пользовательских параметров иногда состоят из двух частей: имя расширения, затем точка, затем имя параметра, очень похожее на квалифицированные имена в SQL. Пример: plpgsql.variable_conflict.

Поскольку в процессах, которые не загрузили соответствующий модуль расширения, может потребоваться установка пользовательских параметров, СУБД-КС примет настройку для любого имени параметра, состоящего из двух частей. Такие переменные рассматриваются как заполнители и не имеют функции до тех пор, пока модуль, который их определяет, не будет загружен. Когда модуль расширения загружается, он добавляет свои определения переменных, преобразует любые значения параметров в соответствии с этими определениями и выдает предупреждения для любых нераспознанных параметров, начинающихся с его имени расширения.

2.3.15. Параметры разработчика

Следующие параметры предназначены для работы с исходным кодом СУБД-КС, а в некоторых случаях - для восстановления сильно поврежденных баз данных. Не должно быть никаких причин использовать их в производственной базе данных. Как таковые, они были исключены из базового файла конфигурации. Обратите внимание, что для многих из этих параметров требуются специальные флаги компиляции исходного кода.

```
allow_system_table_mods( boolean)
```

Позволяет модифицировать структуру системных таблиц. Это используется initdb. Этот параметр можно установить только при запуске сервера.

```
ignore_system_indexes( boolean)
```

Игнорировать системные индексы при чтении системных таблиц (но все же обновлять индексы при изменении таблиц). Это полезно при восстановлении с поврежденных системных индексов. Этот параметр нельзя изменить после начала сеанса.

```
post_auth_delay( integer)
```

Если он не равен нулю, задержка в это много секунд происходит при запуске нового серверного процесса после того, как он выполняет процедуру аутентификации. Это сделано для того, чтобы дать разработчикам возможность подключить к серверу процесс с помощью отладчика. Этот параметр нельзя изменить после начала сеанса.

```
pre_auth_delay( integer)
```

Если он не равен нулю, задается задержка после того, как новый процесс сервера был порожден, перед выполнением процедуры аутентификации. Это нужно для того, чтобы дать разработчикам возможность подключиться к процессу сервера с помощью отладчика для отслеживания неправильного поведения при аутентификации. Этот параметр можно установить только в файле конфигурации или в командной строке сервера.

```
trace_notify( boolean)
```

Формирует большое количество отладочных сообщений для LISTEN и NOTIFY команд. client_min_messages или log_min_messages должны быть DEBUG1 или ниже, чтобы отправить этот вывод в журналы клиента или сервера, соответственно.

```
trace_recovery_messages( enum)
```

Включает ведение журнала результатов отладки, связанных с восстановлением, которые нельзя получить другим способом. Этот параметр позволяет пользователю переопределить обычную настройку log_min_messages, но только для определенных сообщений. Это предназначено для использования при отладке горячего резервирования. Допустимые значения DEBUG5, DEBUG4, DEBUG3, DEBUG2, DEBUG1, и LOG. Значение по умолчанию - LOG - не влияет на запись этих сообщений в журнала. Другие значения приводят к тому, что связанные с восстановлением отладочные сообщения регистрируются так, как если бы они имели LOG-приоритет. Этот параметр можно установить только в файле конфигурации или в командной строке сервера.

```
trace_sort( boolean)
```

Включение настройки позволяет получать информацию об использовании ресурсов во время операций сортировки. Этот параметр доступен, только если TRACE_SORT макрос был определен при компиляции СУБД-КС (тем не менее, TRACE_SORT в настоящее время определяется по умолчанию).

```
trace_locks( boolean)
```

Если включено, то выдается информация об использовании блокировки. Регистрируемая информация включает в себя тип операции блокировки, тип блокировки и уникальный идентификатор заблокированного или разблокированного объекта. Также включены битовые маски для типов блокировки, уже предоставленных для этого объекта, а также для типов блокировки, ожидаемых для этого объекта. Для каждого типа блокировки также подсчитывается количество предоставленных блокировок и ожидающих блокировок, а также итоговые значения. Пример вывода файла журнала показан здесь:

```
LOG: LockAcquire: new: lock(0xb7acd844) id(24688,24696,0,0,0,1)
 grantMask(0) req(0,0,0,0,0,0,0)=0 grant(0,0,0,0,0,0,0)=0
 wait(0) type(AccessShareLock)
LOG: GrantLock: lock(0xb7acd844) id(24688,24696,0,0,0,1)
 grantMask(2) req(1,0,0,0,0,0,0)=1 grant(1,0,0,0,0,0,0)=1
 wait(0) type(AccessShareLock)
```

```
LOG: UnGrantLock: updated: lock(0xb7acd844) id(24688,24696,0,0,0,1)
 grantMask(0) req(0,0,0,0,0,0,0)=0 grant(0,0,0,0,0,0,0)=0
 wait(0) type(AccessShareLock)
LOG: CleanUpLock: deleting: lock(0xb7acd844) id(24688,24696,0,0,0,1)
 grantMask(0) req(0,0,0,0,0,0,0)=0 grant(0,0,0,0,0,0,0)=0
 wait(0) type(INVALID)
```

Детали сбрасываемой конструкции можно найти в src/include/storage/lock.h.

Этот параметр доступен, только если LOCK_DEBUG макрос был определен при компиляции СУБД-КС.

```
trace_lwlocks( boolean)
```

Если включено, то фиксируется информация об использовании легковесных блокировок. Облегченные блокировки предназначены главным образом для обеспечения взаимного исключения доступа к структурам данных с общей памятью.

Этот параметр доступен, только если LOCK_DEBUG макрос был определен при компиляции СУБД-КС.

```
trace_userlocks( boolean)
```

Если включено, регистрируется информация об использовании блокировки пользователя. Вывод такой же, как для trace_locks, только для консультативных блокировок.

Этот параметр доступен, только если LOCK_DEBUG макрос был определен при компиляции СУБД-КС.

```
trace_lock_oidmin( integer)
```

Если установлена данная опция, не отслеживаются блокировки для таблиц ниже этого OID. (используйте, чтобы избежать вывода на системные таблицы)

Этот параметр доступен, только если LOCK_DEBUG макрос был определен при компиляции СУБД-КС.

```
trace_lock_table( integer)
```

Безусловно фиксировать блокировки на этой таблице (OID).

Этот параметр доступен, только если макрос LOCK_DEBUG был определен при компиляции СУБД-КС .

```
debug_deadlocks( boolean)
```

Если установлено, выводит информацию обо всех текущих блокировках при возникновении взаимоблокировки.

Этот параметр доступен, только если макрос LOCK_DEBUG был определен при компиляции СУБД-КС.

```
log_btree_build_stats( boolean)
```

Если установлено, регистрирует статистику использования системных ресурсов (память и ЦП) для различных операций B-дерева.

Этот параметр доступен, только если макрос BTREE_BUILD_STATS был определен при компиляции СУБД-КС.

```
wal_consistency_checking( string)
```

Этот параметр предназначен для проверки ошибок в процедурах повторного воспроизведения WAL. Если этот параметр включен, полностраничные изображения любых буферов, измененных вместе с записью WAL, добавляются в запись. Если запись впоследствии воспроизводится, система сначала применяет каждую запись, а затем проверяет, соответствуют ли буферы, измененные записью, сохраненным образам. В некоторых случаях допускаются незначительные изменения, которые будут игнорироваться. Любые неожиданные различия приведут к фатальной ошибке, что приведет к прекращению восстановления.

Значением этого параметра по умолчанию является пустая строка, которая отключает эту функцию. Он может быть установлен для общей проверки всех записей или для списка имен менеджеров ресурсов, разделенных запятыми, чтобы проверять только записи, выдаваемых этими менеджерами ресурсов. В настоящее время поддерживаются менеджеры ресурсов heap, heap2, btree, hash, gin, gist, sequence, spgist, brin, и generic. Только администраторы могут изменять этот параметр.

```
wal_debug( boolean)
```

Если включено, выдает отладочную информацию, связанную с WAL. Этот параметр доступен, только если макрос WAL_DEBUG был определен при компиляции СУБД-КС.

```
ignore_checksum_failure( boolean)
```

Имеет эффект только если включён data checksums.

Обнаружение ошибки контрольной суммы во время чтения обычно заставляет СУБД-КС сообщать об ошибке и прерывает текущую транзакцию. При включении опции ignore_checksum_failure система игнорирует сбой (но все равно выдает предупреждение) и продолжает обработку. Такое поведение может вызвать сбои, распространение или скрытие ошибки, или скрытие других серьезных проблем. Тем не менее, это может позволить вам обойти ошибку и извлечь неповрежденные кортежи, которые все еще могут присутствовать в таблице, если заголовок блока все еще цел. Если заголовок поврежден, будет сообщено об ошибке, даже если эта опция включена. По умолчанию установлено значение off, и оно может быть изменено только администратором.

```
zero_damaged_pages( boolean)
```

Обнаружение поврежденного заголовка страницы обычно заставляет СУБД-КС сообщать об ошибке, прерывая текущую транзакцию. При установке значения zero_damaged_pageson система вместо этого выдает предупреждение, обнуляет поврежденную страницу в памяти и

продолжает обработку. Такое поведение уничтожит данные, а именно все строки на поврежденной странице. Однако он позволяет обойти ошибку и извлечь строки из любых неповрежденных страниц, которые могут присутствовать в таблице. Это полезно для восстановления данных, если произошло повреждение из-за аппаратной или программной ошибки. Не надо устанавливать этот параметр, пока не потеряна надежда на восстановление данных с поврежденных страниц таблицы. Страницы с нулевым удалением не переносятся на диск, поэтому рекомендуется заново создать таблицу или индекс, прежде чем снова отключить этот параметр. По умолчанию установлено значение off, и оно может быть изменено только администратором.

```
jit_debugging_support( boolean)
```

Если LLVM обладает необходимой функциональностью, то происходит регистрация генерированных функций в GDB. Это облегчает отладку. По умолчанию установлено значение off. Этот параметр можно установить только при запуске сервера.

```
jit_dump_bitcode( boolean)
```

Записывает сгенерированный IR LLVM в файловую систему внутри data_directory. Это полезно только для работы над внутренними компонентами реализации JIT. По умолчанию установлено значение off. Этот параметр может быть изменен только администратором.

```
jit_expressions( boolean)
```

Определяет, являются ли выражения JIT-компилируемыми, когда JIT-компиляция активирована. По умолчанию установлено значение on.

```
jit_profiling_support( boolean)
```

Если LLVM обладает требуемыми функциями, отправляет данные, необходимые для профилирования с помощью функций профиля, сгенерированных JIT. Выходные файлы записываются в каталог $HOME/.debug/jit/. По умолчанию установлено значение off. Этот параметр можно установить только при запуске сервера.

```
jit_tuple_deforming( boolean)
```

Определяет, будет ли трансформироваться кортеж JIT, если JIT-компиляция активирована. По умолчанию имеет значение on.

2.3.16. Короткие параметры

Для удобства предусмотрены также соответствующие однобуквенные параметры командной строки для некоторых параметров. Они описаны в таблице (*Таблица 4*).

Таблица 4. Краткий ключ параметра

Краткий параметр	Эквивалент
1	2
-B x	shared_buffers = x
-d x	log_min_messages = DEBUGx
-e	datestyle = euro
-fb, -fh, -fi, -fm, -fn, -fo, -fs,-ft	enable_bitmapscan = off, enable_hashjoin = off, enable_indexscan = off, enable_mergejoin = off, enable_nestloop = off, enable_indexonlyscan = off, enable_seqscan = off,enable_tidscan = off
-F	fsync = off
-h x	listen_addresses = x
-i	listen_addresses = '*'
-k x	unix_socket_directories = x
-l	ssl = on
-N x	max_connections = x
-O	allow_system_table_mods = on
-p x	port = x
-P	ignore_system_indexes = on
-s	log_statement_stats = on
-S x	work_mem = x
-tpa, -tpl,-te	log_parser_stats = on, log_planner_stats = on,log_executor_stats = on
-W x	post_auth_delay = x

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

Сокращение или термин	Термин
1	2
ПК	Программный комплекс
БД	База данных

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

Б

Блокировка (1 уровень), 66

Д

Дополнительные модули параметров (1 уровень), 71

З

Запись в журнал (1 уровень), 26
Запуск сервера (1 уровень), 11
Значения по умолчанию (1 уровень), 56

К

Кластер, 9
Конфигурация сервера (1 уровень), 14
Краткие эквиваленты ключей командной строки (1 уровень), 76

Н

Настройка соединения (1 уровень), 17

О

Очистка (1 уровень), 54

П

Планирование запросов (1 уровень), 33
Получение информации о выполнении запросов (1 уровень), 52
Потребление памяти, дискового пространства, ресурсов ядра (1 уровень), 18
Предустановленные параметры (1 уровень), 69
Проблемы на клиенте (1 уровень), 13

Р

Работа над ошибками (1 уровень), 68

С

Сбои при запуске сервера (1 уровень), 12
Сведения об ошибках (1 уровень), 42
Сетевые файловые системы (1 уровень), 11
Системные параметры (1 уровень), 71
Совместимость (1 уровень), 68

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Номер версии	Примечание	Дата	ФИО исполнителя
01	Начальная версия	25.11.2022	Котова И.В.