PL/pgSQL
Отладка

Темы

Способы отладки

Служебные сообщения

Трассировка

Способы отладки

Использование отладчика

Служебные сообщения в коде

PL Debugger

Состав

- расширение pldbgapi
- графический интерфейс: PgAdmin, пр.

Возможности

- установка точек сохранения
- пошаговое выполнение
- проверка и установка значений переменных
- не требуется изменение кода
- отладка работающих приложений

Служебные сообщения

Отладка кода

Мониторинг длинных процессов

Журнал приложения

Служебные сообщения

Подходы к реализации

- RAISE
- межпроцессное взаимодействие
- запись в таблицу
- запись в файл

Учесть при реализации

- нетранзакционные сообщения
- одновременная работа
- доступ к журналу
- очистка журнала

RAISE

1. Добавление в код отладочных сообщений

2. Вызов функции

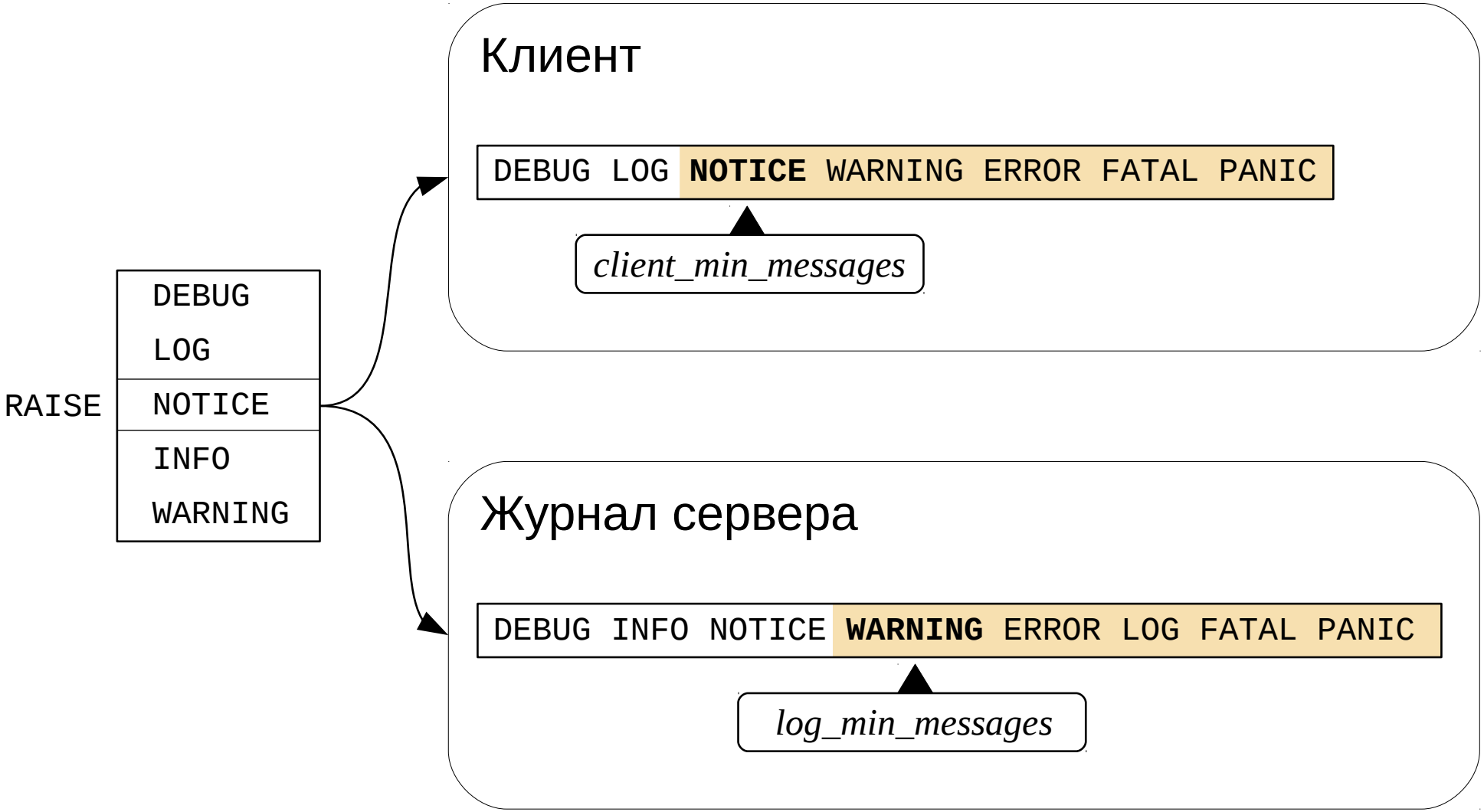
3. Асинхронные сообщения по ходу выполнения

```
CREATE FUNCTION get_count
    (tabname text) RETURNS bigint
AS $$
DECLARE
    cmd text;
    retval bigint;
BEGIN
    cmd := 'SELECT COUNT(*) FROM '
        || quote_ident(tabname);
    RAISE NOTICE 'cmd: %', cmd;
    EXECUTE cmd INTO retval;
    RETURN retval;
END;
$$ LANGUAGE plpgsql;
```

```
psql=> SELECT get_count('pg_class');

NOTICE:  cmd: SELECT COUNT(*) FROM pg_class
 get_count
-----------
       334
(1 row)
```

RAISE


RAISE
DEBUG
LOG
NOTICE
INFO
WARNING
Клиент
DEBUG LOG NOTICE WARNING ERROR FATAL PANIC
client_min_messages
Журнал сервера
DEBUG INFO NOTICE WARNING ERROR LOG FATAL PANIC
log_min_messages

Процесс → процесс

NOTIFY / LISTEN

- встроенные команды сервера
- транзакционные команды

Статус сеанса

- параметр *application_name* виден в представлении pg_stat_activity, можно вывести в журнал сервера

Процесс → таблица

Расширение dblink

- входит в состав сервера
- накладные расходы на создание соединения

Автономные транзакции

- коммерческий дистрибутив (Postgres Pro Enterprise)

Процесс → файл

Расширение adminpack

- входит в состав сервера
- файлы внутри PGDATA

Недоверенные языки

- например, PL/PerlU

Трассировка

Трассировка — журналирование выполняемых на сервере команд SQL

Команды записываются в журнал сервера

Трассировка

Накладные расходы на запись в журнал

Большой размер журнала

Инструменты для работы с журналом

Доступ к журналу (безопасность)

Трассировка

Настройки

время выполнения длинных команд	log_min_duration_statement
какие команды включать	log_statement
время выполнения команд	log_duration
длинные ожидания	log_lock_waits
использование временных файлов	log_temp_files
контекст сообщения	log_line_prefix
…	

Способы задания параметров

все сеансы	конфигурационные файлы
отдельные сеансы	ALTER DATABASE … SET ALTER ROLE … SET
во время выполнения	SET, set_config()

Трассировка

Расширение auto_explain

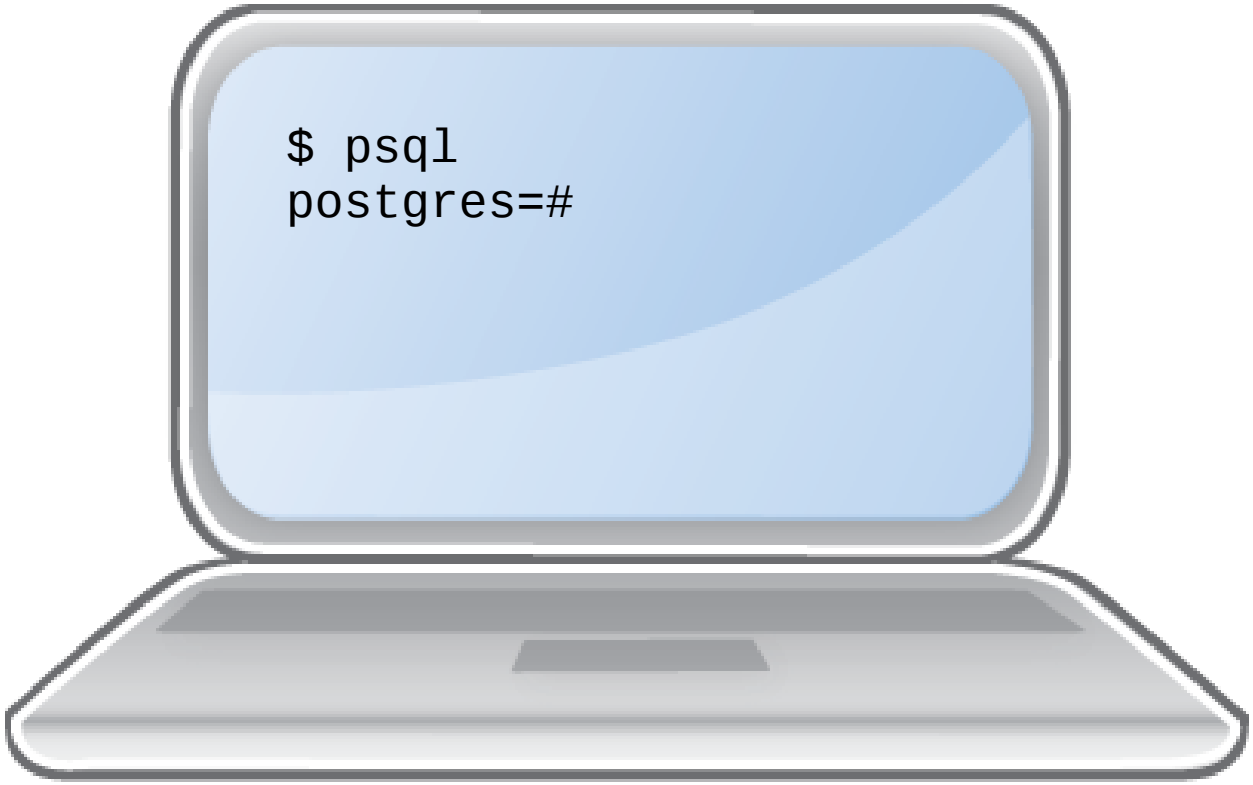
Задачи

- включение в журнал планов выполнения
- включение в журнал вложенных запросов

Настройки

планы длинных команд	auto_explain.log_min_duration
вложенные запросы	auto_explain.log_nested_statements
…	

Демонстрация



Итоги

PL Debugger — отладчик PL/pgSQL,
API поддерживается сообществом

Служебные сообщения в коде — доступны разные подходы

Гибкие настройки журнала сервера для трассировки сеансов

Практика

1. Измените функцию get_catalog так, чтобы текст динамически формируемого запроса записывался в журнал сервера.
2. В приложении выполните несколько раз поиск, заполняя разные поля, и убедитесь, что команды SQL формируются правильно.
3. Включите трассировку команд в файле postgresql.conf и перечитайте конфигурацию.
4. Поработайте в приложении и проверьте, какие команды попадают в журнал сервера.

Авторские права
© Postgres Professional, 2017 год.
Авторы: Егор Рогов, Павел Лузанов

Использование материалов курса
Некоммерческое использование материалов курса (презентации, демонстрации) разрешается без ограничений. Коммерческое использование возможно только с письменного разрешения компании Postgres Professional. Запрещается внесение изменений в материалы курса.

Обратная связь
Отзывы, замечания и предложения направляйте по адресу:
edu@postgrespro.ru

Отказ от ответственности
Компания Postgres Professional не несет никакой ответственности за любые повреждения и убытки, включая потерю дохода, нанесенные прямым или непрямым, специальным или случайным использованием материалов курса. Компания Postgres Professional не предоставляет каких-либо гарантий на материалы курса. Материалы курса предоставляются на основе принципа «как есть» и компания Postgres Professional не обязана предоставлять сопровождение, поддержку, обновления, расширения и изменения.

Темы

Способы отладки

Служебные сообщения

Трассировка

Способы отладки

Использование отладчика

Служебные сообщения в коде

Отладка представляет собой процесс поиска причин, по которым код работает не так, как ожидается. В большинстве случаев речь идет о поиске ошибок в коде. Но не следует исключать ситуации, когда не верны наши ожидания, а код работает как и планировалось.

Обычно выделяют два способа отладки:

- Использование отладчика — специальной программы, обычно с пользовательским интерфейсом. Отладчики позволяют пошагово выполнять код программ, устанавливать точки останова, проверять и устанавливать значения переменных.
- Добавлять в критические места кода вывод диагностических сообщений, которые содержат информацию о текущем контексте выполнения (значения переменных, стек вызова и пр.)

Для отладки хранимых функций на PL/pgSQL доступны оба способа.

PL Debugger

Postgres PROFESSIONAL

Состав

- расширение pldbgapi
- графический интерфейс: PgAdmin, пр.

Возможности

- установка точек сохранения
- пошаговое выполнение
- проверка и установка значений переменных
- не требуется изменение кода
- отладка работающих приложений

PL Debugger — это отладчик для PL/pgSQL. Представляет собой расширение pldbgapi, которое официально поддерживается разработчиками PostgreSQL.

Расширение pldbgapi — это набор функций для сервера PostgreSQL, которые позволяют устанавливать точки останова, пошагово выполнять код программ, проверять и устанавливать значений переменных.

Некоторые GUI-инструменты, например PgAdmin, имеют встроенный интерфейс для работы с этими функциями.

Исходный код отлаживаемых программ изменять не требуется, что дает возможность выполнять отладку работающих приложений. Т. е. процесс отладки не обязательно запускать отдельно, можно подключиться к работающему сеансу и начать отладку.

Исходный код отладчика доступен по ссылке:

https://git.postgresql.org/gitweb/?p=pldebugger.git;a=summary

Служебные сообщения

Отладка кода

Мониторинг длинных процессов

Журнал приложения

Второй способ отладки предполагает добавление в важные места кода служебных сообщений, содержащих текущий контекст. Анализируя выдачу отладочных сообщений можно понять, что именно пошло не так.

Помимо собственно отладки, служебные сообщения могут выполнять и другие функции.

Например, если какой-то процесс выполняется достаточно долго, то служебные сообщения помогут определить, на каком этапе выполнения находится программа.

Так же, как и в журнал сервера записывается важная информация о работе СУБД, так и приложение, при помощи служебных сообщений, может записывать в некий журнал важную информацию о своей работе. Например, в журнал приложения можно записывать информацию о запуске отчетов: название отчета, пользователь, когда, с какими параметрами и т. д. Такая информация может сильно облегчить работу специалистов поддержки.

Служебные сообщения

Подходы к реализации

- RAISE
- межпроцессное взаимодействие
- запись в таблицу
- запись в файл

Учесть при реализации

- нетранзакционные сообщения
- одновременная работа
- доступ к журналу
- очистка журнала

Можно выделить несколько подходов к реализации служебных сообщений в PL/pgSQL.

Помимо уже знакомой команды RAISE, можно отправлять сообщения другому процессу, записывать сообщения в таблицу или в файл.

При выборе того или иного подхода нужно обращать внимание на следующее.

1. Являются ли сообщения транзакционными?
   - а) отправка сообщений не дожидаясь окончания транзакции;
   - б) отправка независимо от статуса завершения транзакции.
2. Одновременное использование в нескольких сеансах
3. Доступ к журналу сообщений.
4. Средства для очистки журнала.

RAISE

```
CREATE FUNCTION get_count
    (tabname text) RETURNS bigint
AS $$
DECLARE
    cmd text;
    retval bigint;
BEGIN
    cmd := 'SELECT COUNT(*) FROM '
        || quote_ident(tabname);
    RAISE NOTICE 'cmd: %', cmd;
    EXECUTE cmd INTO retval;
    RETURN retval;
END;
$$ LANGUAGE plpgsql;
```

1. Добавление в код отладочных сообщений

2. Вызов функции

```
psql=> SELECT get_count('pg_class');

NOTICE:  cmd: SELECT COUNT(*) FROM pg_class
 get_count
-----------
       334
(1 row)
```

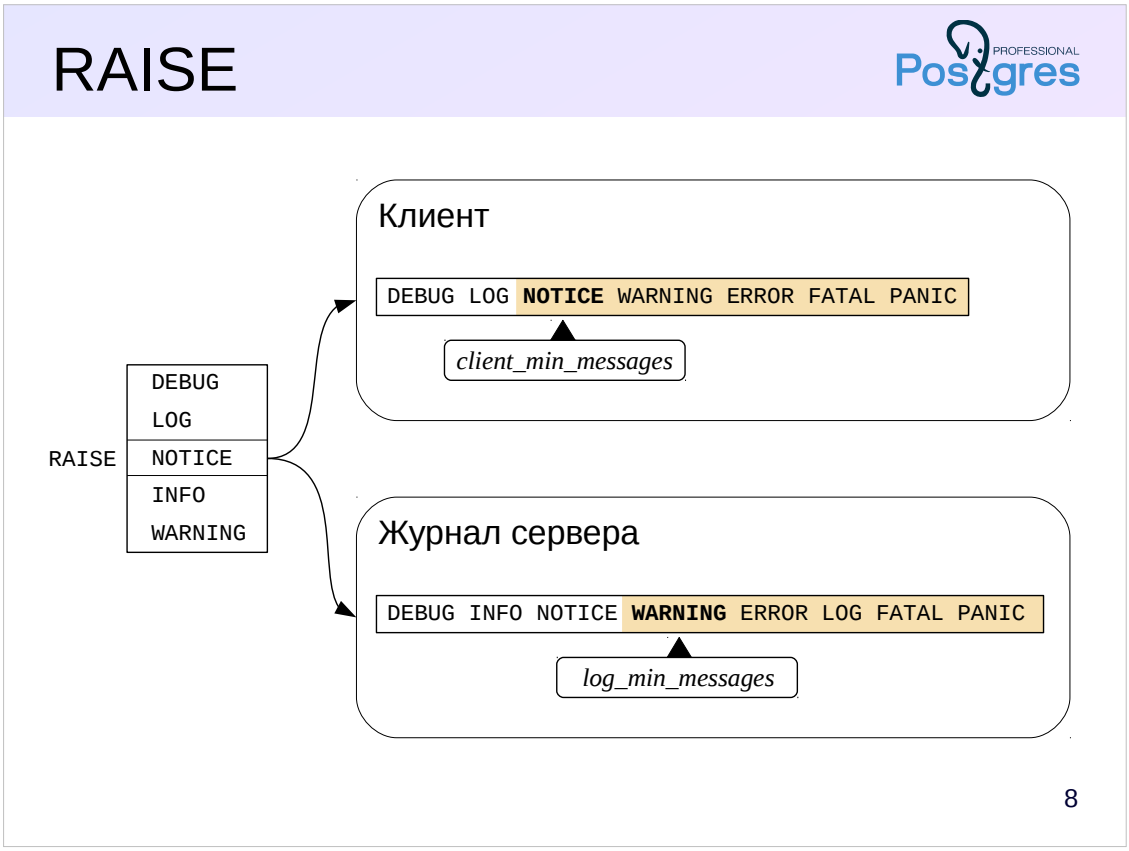
3. Асинхронные сообщения по ходу выполнения

Мы уже встречались с командой RAISE. В простом случае, для отладки нужно:

- добавить вызовы RAISE NOTICE в код функции;
- запустить функцию на выполнение, например в сеансе psql;
- анализировать получаемые сообщения по ходу выполнения программы.

Важно отметить, что сообщения RAISE нетранзакционные: отправляются асинхронно, не зависят от статуса завершения транзакции.

https://postgrespro.ru/docs/postgresql/9.6/plpgsql-errors-and-messages.html#plpgsql-statements-raise



Команда RAISE имеет два основных назначения.

Во-первых, она служит для вызова исключительных ситуаций, что подробно рассматривалось в теме «Обработка ошибок».

Во-вторых, команда используется для отправки сообщений. Причем сообщения можно не только отправлять клиенту, но и записывать в журнал сервера.

Для управления отправкой сообщений используются уровень сообщения (DEBUG, LOG, NOTICE, INFO, WARNING) команды RAISE и параметры client_min_messages, log_min_messages. От значений этих параметров зависит, будет ли сообщение отправлено клиенту (client_min_messages) и/или записано в журнал сервера (log_min_messages).

Оба параметра имеют упорядоченные списки значений. Сообщение будет отправлено клиенту и/или серверу, если уровень команды RAISE равен значению соответствующего параметра или находится правее.

Значения параметров по умолчанию настроены так, что сообщения с уровнем NOTICE отправляются только клиенту, с уровнем LOG — только в журнал, а с уровнем WARNING — и клиенту, и в журнал.

Сообщения с уровнем INFO всегда отправляются клиенту, их нельзя перехватить параметром client_min_messages.

Процесс → процесс

NOTIFY / LISTEN

- встроенные команды сервера
- транзакционные команды

Статус сеанса

- параметр *application_name* виден в представлении pg_stat_activity, можно вывести в журнал сервера

Серверные процессы в PostgreSQL могут обмениваться информацией между собой.

Среди встроенных решений можно отметить следующие.

- Использование команд NOTIFY для отправки сообщений в одном процессе и LISTEN для получения в другом. При использовании этих команд нужно помнить, что они являются транзакционными.
- Использование параметра application_name. Например, сеанс с долго выполняющимся процессом может периодически записывать статус выполнения в application_name. В другом сеансе администратор может опрашивать представление pg_stat_activity, содержащем подробную информацию, включая application_name, о всех выполняющихся сеансах. Значение application_name также можно записывать в журнал сервера (настраивая параметр log_line_prefix). Это облегчит поиск нужных строк в журнале.

Процесс → таблица

Расширение dblink

- входит в состав сервера
- накладные расходы на создание соединения

Автономные транзакции

- коммерческий дистрибутив (Postgres Pro Enterprise)

Еще один способ обмена сообщениями — запись сообщений в таблицу базы данных.

К плюсам данного подхода относится то, что параллельная работа и доступ журналу обеспечиваются средствами самой СУБД.

Однако, нужно позаботиться о том, чтобы вставка в таблицу была нетранзакционной.

Для этих целей можно использовать расширение dblink, идущее в составе сервера PostgreSQL. Это расширение позволяет открыть новое соединение к той же самой БД, поэтому вставка в таблицу выполняется в другой транзакции. К минусам данного подхода относится то, что открытие нового соединения требует дополнительных ресурсов сервера.

В коммерческих дистрибутивах, например Postgres Pro Enterprise, также доступен механизм автономных транзакций.

Процесс → файл

Расширение adminpack

- входит в состав сервера
- файлы внутри PGDATA

Недоверенные языки

- например, PL/PerlU

Вести запись сообщений можно и в файл операционной системы.

Например, с использованием расширения adminpack. Это расширение позволяет читать и писать файлы, находящиеся внутри каталога PGDATA.

Если требуется хранить журнал вне PGDATA, можно написать собственные функции для записи сообщений в журнал на любом недоверенном языке.

Трассировка

Трассировка — журналирование выполняемых на сервере команд SQL

Команды записываются в журнал сервера

Основная задача сервера базы данных — выполнять запросы пользователей. PostgreSQL можно настроить таким образом, чтобы запросы пользователей записывались в журнал сервера.

Трассировка

Накладные расходы на запись в журнал

Большой размер журнала

Инструменты для работы с журналом

Доступ к журналу (безопасность)

Такой подход к трассировке, когда запросы всех сеансов записываются в общий журнал, имеет ряд особенностей, которые необходимо учитывать:

- Высоконагруженное приложение может выполнять огромное число запросов. Их запись в файл журнала может оказывать влияние на производительность системы ввода/вывода.
- При этом размер самого журнала может быть огромен, что затруднит работу с ним.
- Журнал — это текстовый файл. И хотя его можно просматривать глазами, в большинстве случаев требуются специальные инструменты для анализа, например pgBadger. Журнал можно загружать и в таблицу базы данных для анализа средствами SQL.
- Журнал — это файл на сервере. И чтобы его посмотреть, к этому файлу нужен доступ, который не всегда есть у разработчиков приложений. К тому же, если речь идет о продуктивной системе, в журнале могут быть команды, содержащие информацию, доступ к которой должен быть ограничен. Например, значения отдельных полей в запросах.

Трассировка

Postgres PROFESSIONAL

Настройки

время выполнения длинных команд	log_min_duration_statement
какие команды включать	log_statement
время выполнения команд	log_duration
длинные ожидания	log_lock_waits
использование временных файлов	log_temp_files
контекст сообщения	log_line_prefix
...	

Способы задания параметров

все сеансы	конфигурационные файлы
отдельные сеансы	ALTER DATABASE … SET ALTER ROLE … SET
во время выполнения	SET, set_config()

Для настройки трассировки имеется ряд конфигурационных параметров. Например, можно настроить запись в журнал всех выполняемых команд или только команд, которые выполняются дольше определенного времени. Также в журнал можно записывать информацию об ожиданиях, использовании временных файлов и пр.

Список всех параметров для управления журналом сервера: https://postgrespro.ru/docs/postgresql/9.6/runtime-config-logging.html

Самый простой способ задания параметров — записать их в конфигурационный файл. В этом случае параметры будут действовать на сеансы всех пользователей.

Чтобы ограничить действие параметров, их можно установить для отдельных сеансов при помощи команд ALTER DATABASE, ALTER ROLE.

Наконец, параметры можно включать/отключать прямо во время выполнения, командой SET или функцией set_config. По умолчанию, устанавливать параметры журналирования могут только суперпользователи. Однако это ограничение можно обойти, если использовать функции SECURITY DEFINER (это рассматривается в модуле «Разграничение доступа»).

Подробнее об установке параметров конфигурации: https://postgrespro.ru/docs/postgresql/9.6/config-setting.html

Трассировка

Расширение auto_explain

Задачи

- включение в журнал планов выполнения
- включение в журнал вложенных запросов

Настройки

планы длинных команд	auto_explain.log_min_duration
вложенные запросы	auto_explain.log_nested_statements
…	

При включении трассировки, в журнал попадают команды в том виде, в каком они отправлены на сервер. Если была вызвана функция PL/pgSQL, то в журнале будет записан только вызов функции, но не будет всех команд, которые выполняются внутри функции.

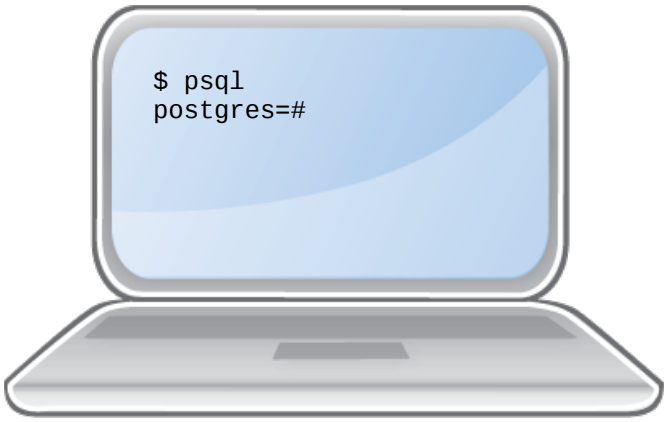
Если все-таки нужно иметь в журнале все выполняемые команды, включая вложенные в функции, то это можно сделать при помощи расширения auto_explain.

Как и следует из названия расширения, его основная задача записывать в журнал не только текст команды, но и план ее выполнения.

Подробнее о расширении auto_explain:

https://postgrespro.ru/docs/postgresql/9.6/auto-explain.html

Демонстрация



Итоги

PL Debugger — отладчик PL/pgSQL,
API поддерживается сообществом

Служебные сообщения в коде — доступны разные подходы

Гибкие настройки журнала сервера для трассировки сеансов

Практика

1. Измените функцию get_catalog так, чтобы текст динамически формируемого запроса записывался в журнал сервера.
2. В приложении выполните несколько раз поиск, заполняя разные поля, и убедитесь, что команды SQL формируются правильно.
3. Включите трассировку команд в файле postgresql.conf и перечитайте конфигурацию.
4. Поработайте в приложении и проверьте, какие команды попадают в журнал сервера.