



Разработка серверной части приложений PostgreSQL Базовый курс

Преподаватель:
Валентин Степанов

День 4

- PL/pgSQL (продолжение)
 - Триггеры
 - Отладка
- Разграничение доступа
 - Обзор разграничения доступа
- Резервное копирование
 - Логическое резервирование

PL/pgSQL

Триггеры

Триггеры и функции

- Триггер
 - объект базы данных – список обрабатываемых событий
 - при возникновении события вызывается триггерная функция и ей передается контекст
- Триггерная функция
 - объект базы данных – код обработки события
 - выполняется в той же транзакции, что и основная операция
 - соглашение: функция не принимает параметры, возвращает значение псевдотипа trigger (фактически record)
 - может использоваться в нескольких триггерах

События

- INSERT, UPDATE, DELETE

- | | | |
|-----------------|------------------------------|------------------|
| • таблицы | before/after
before/after | statement
row |
| • представления | before/after
instead of | statement
row |

- TRUNCATE

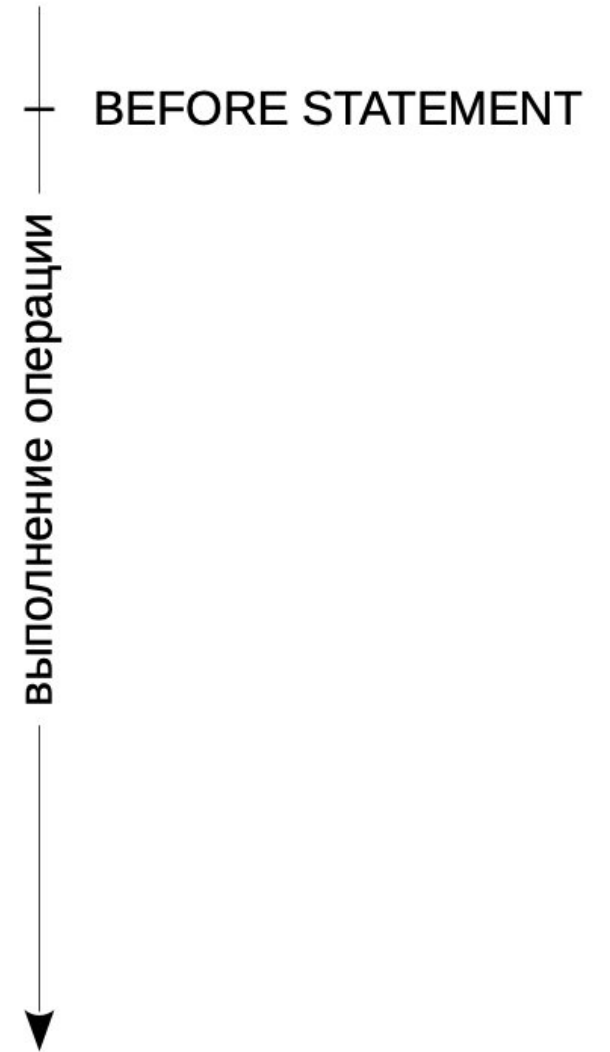
- | | | |
|-----------|--------------|-----------|
| • таблицы | before/after | statement |
|-----------|--------------|-----------|

- Условие WHEN

- устанавливает дополнительный фильтр

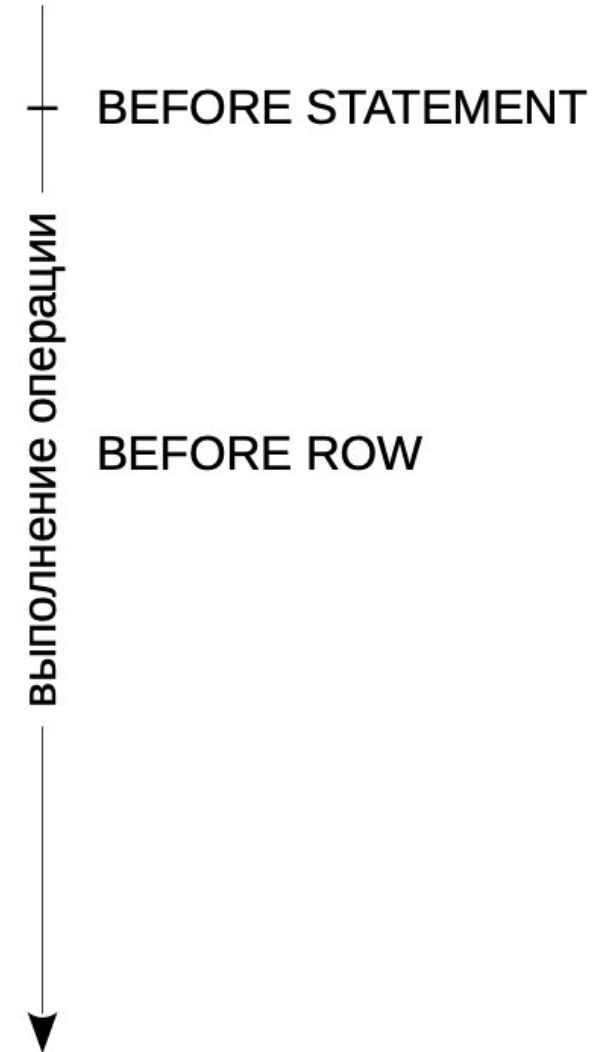
Before statement

- Срабатывает до операции
- Возвращаемое значение игнорируется
- Контекст TG-переменные



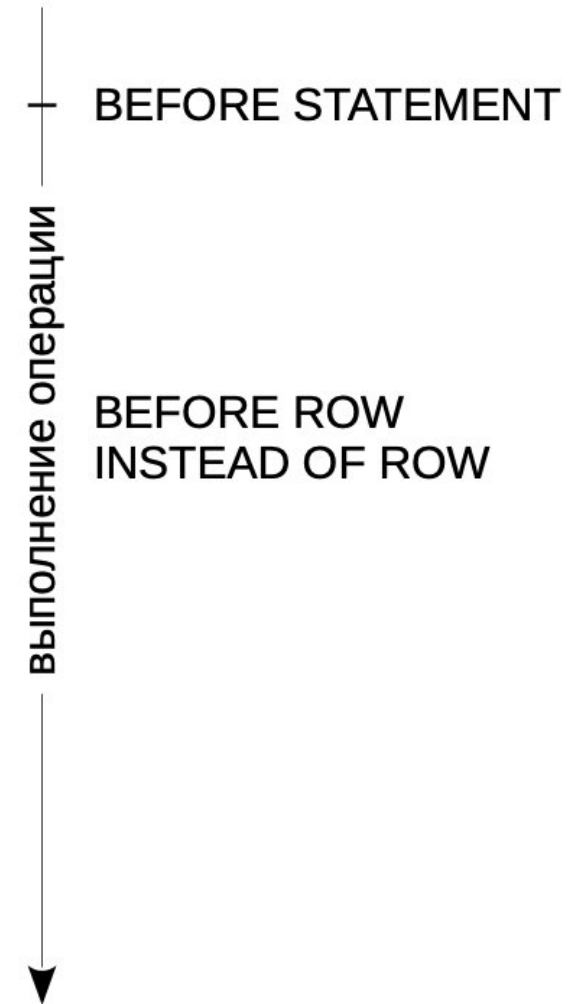
Before row

- Срабатывает перед действием со строкой в процессе выполнения операции
- Возвращаемое значение
 - строка (возможно, измененная)
 - null отменяет действие
- Контекст
 - OLD update, delete
 - NEW insert, update
 - TG-переменные



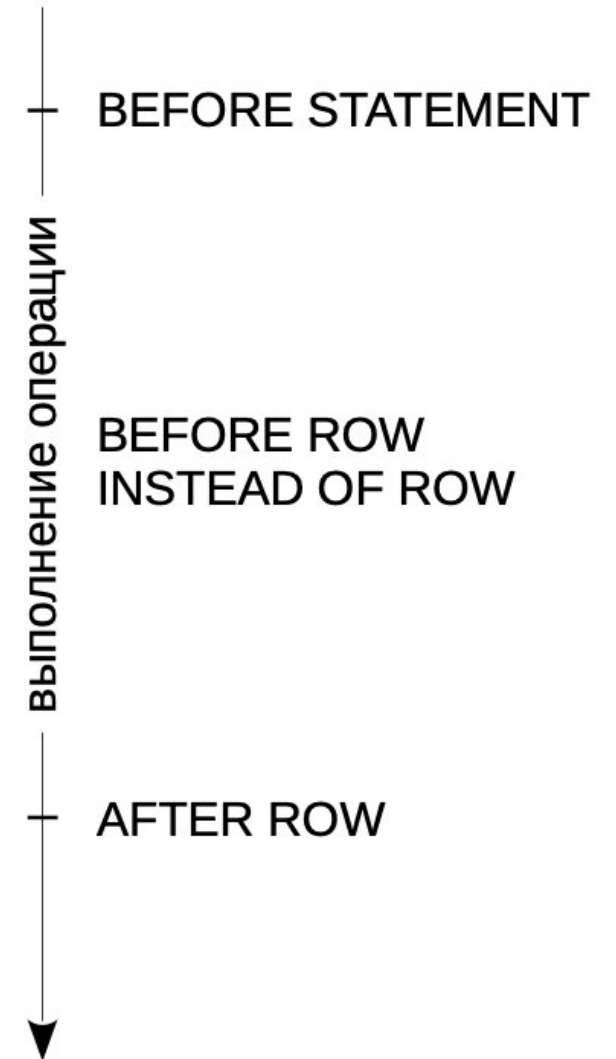
Instead of row

- Срабатывает вместо действия со строкой для представлений
- Возвращаемое значение
 - строка (возможно, измененная) – будет видна в RETURNING
 - null отменяет действие
- Контекст
 - OLD update, delete
 - NEW insert, update
 - TG-переменные



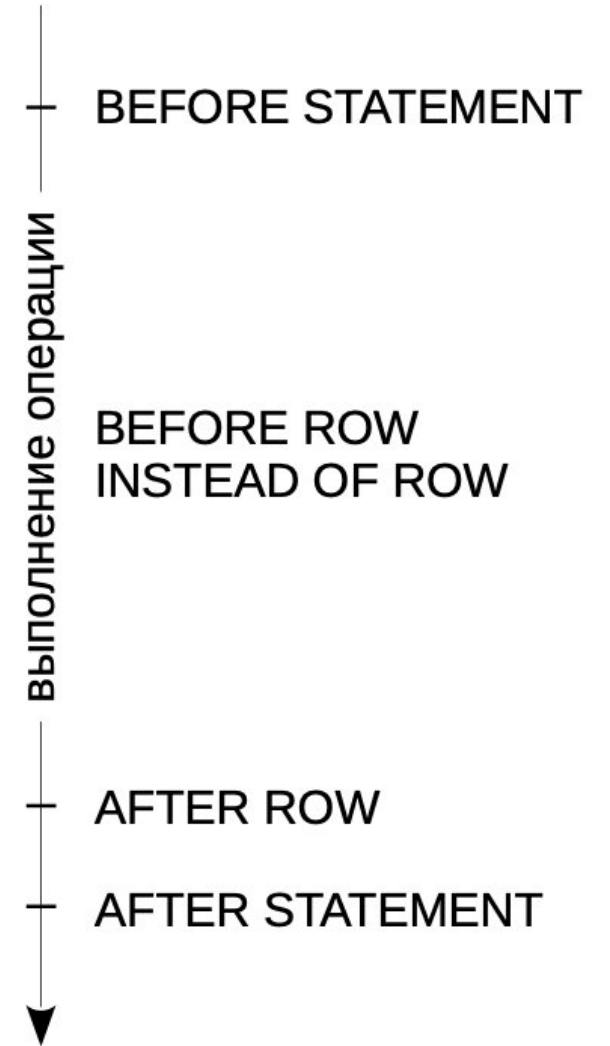
After row

- Срабатывает
 - после выполнения операции
 - очередь из прошедших условие WHEN
- Возвращаемое значение игнорируется
- Контекст
 - OLD, OLD TABLE update, delete
 - NEW, NEW TABLE insert, update
 - TG-переменные

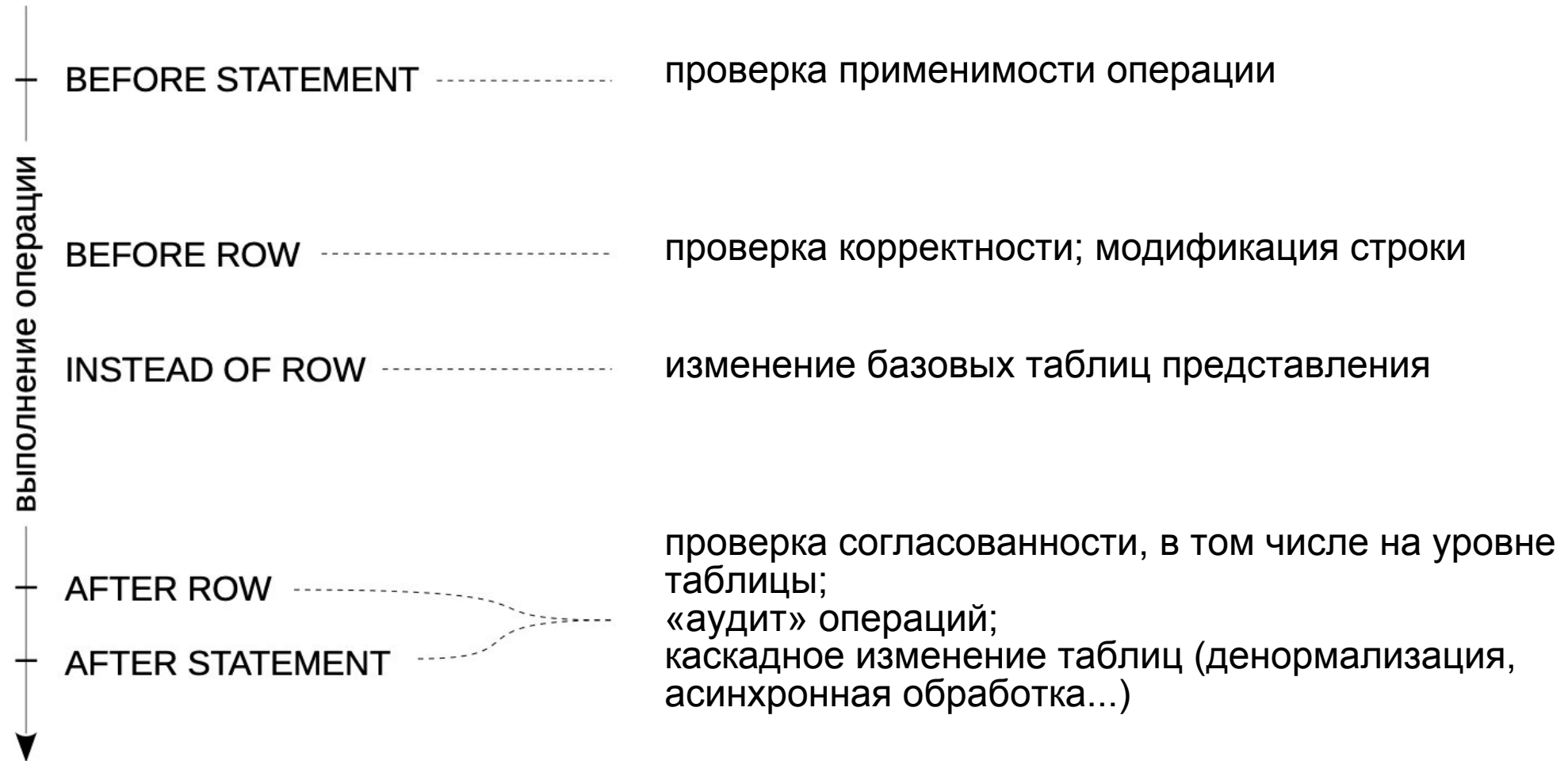


After statement

- Срабатывает после операции (даже если не затронута ни одна строка)
- Возвращаемое значение игнорируется
- Контекст
 - OLD TABLE update, delete
 - NEW TABLE insert, update
 - TG-переменные



Возможное использование



Сложности

- Код вызывается неявно
 - сложно отследить логику выполнения
- Правила видимости изменчивой триггерной функции
 - виден результат триггеров BEFORE ROW или INSTEAD OF ROW
- Порядок вызова триггеров для одного события
 - триггеры отрабатывают в алфавитном порядке
- Не предотвращается зацикливание
 - триггер может вызвать срабатывание других триггеров
- Можно нарушить ограничения целостности
 - например, исключив из обработки строки, которые должны удалиться

Событийные триггеры

- Событийный триггер
 - похож на обычный «табличный» триггер, но другой объект
- Триггерная функция
 - соглашение: функция не принимает параметры, возвращает значение псевдотипа `event_trigger`
 - для получения контекста служат специальные функции
- События
 - `DDL_COMMAND_START` перед выполнением команды
 - `DDL_COMMAND_END` после выполнения команды
 - `TABLE_REWRITE` перед перезаписью таблицы
 - `SQL_DROP` после удаления объектов

Итоги

- Триггер – способ отреагировать на возникновение события
- С помощью триггера можно отменить операцию, изменить ее результат или выполнить дополнительные действия
- Триггер выполняется как часть транзакции; ошибка в триггере приводит к откату транзакции
- Использование триггеров AFTER ROW и переходных таблиц удорожает обработку
- Все хорошо в меру: сложную логику трудно отлаживать из-за неявного выполнения триггеров

Практика №26

1. Создайте триггер, обрабатывающий обновление поля `onhand_qty` представления `catalog_v`.

Проверьте, что в «Каталоге» появилась возможность заказывать книги.

2. Обеспечьте выполнение требования согласованности: количество книг на складе не может быть отрицательным (нельзя купить книгу, которой нет в наличии).

Внимательно проверьте правильность реализации, учитывая, что с приложением могут одновременно работать несколько пользователей.

Практика №27

1. Напишите триггер, увеличивающий счетчик (поле version) на единицу при каждом изменении строки. При вставке новой строки счетчик должен устанавливаться в единицу. Проверьте правильность работы.

2. Даны таблицы заказов (orders) и строк заказов (lines). Требуется выполнить денормализацию: автоматически обновлять сумму заказа в таблице orders при изменении строк в заказе.

Создайте необходимые триггеры с использованием переходных таблиц для минимизации операций обновления.

PL/pgSQL

Отладка

Проверки корректности

- Проверки времени компиляции и времени выполнения
 - plpgsql.extra_warnings
 - plpgsql.extra_errors
 - дополнительные проверки в расширении plpgsql_check
- Проверки в коде
 - команда ASSERT
- Тестирование

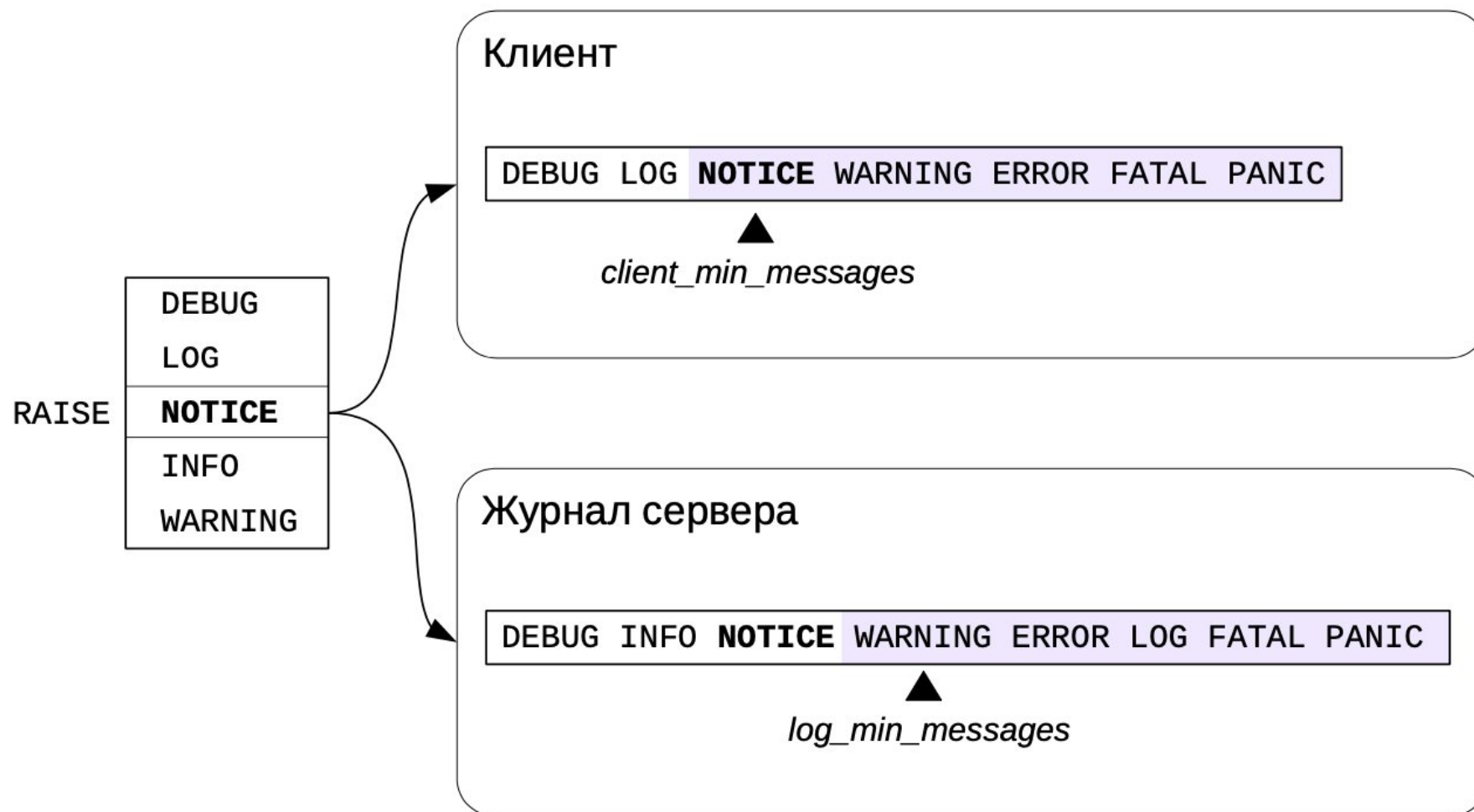
PL/pgSQL Debugger

- Интерфейс
 - расширение `pldbgar` предоставляет API
 - пользовательский интерфейс в некоторых графических средах
- Возможности
 - установка точек прерывания
 - пошаговое выполнение
 - проверка и установка значений переменных
 - не требуется изменение кода
 - отладка работающих приложений

Служебные сообщения

- Не только отладка кода
 - мониторинг долго выполняющихся процессов
 - ведение журнала приложения
- Подходы к реализации
 - вывод на консоль или в журнал сервера
 - запись в таблицу или в файл
 - передача информации другим процессам

Команда RAISE



Процесс → процесс (IPC)

- NOTIFY → LISTEN
 - команды SQL
 - транзакционное выполнение, неудобно для отладки
- Статус сеанса
 - параметр `application_name`
 - виден в представлении `pg_stat_activity` и выводе команды `ps`
 - можно использовать в журнальных сообщениях

Процесс → таблица

- Расширение dblink
 - входит в состав сервера
 - накладные расходы на создание соединения

Процесс → файл

- Расширение adminpack
 - входит в состав сервера
 - в том числе позволяет записывать текстовые файлы
- Недоверенные языки
 - например, PL/Perl

Трассировка SQL

- Стандартная трассировка в журнал сообщений
 - накладные расходы на запись в журнал
 - большой размер файла журнала
 - требуются инструменты для анализа
 - нужен доступ к журналу (безопасность)
- Настройки
 - долго выполняющиеся команды `log_min_duration_statement`
 - какие команды записывать `log_statement`
 - контекст сообщения `log_line_prefix`
 - ...

Трассировка SQL

- Расширение `auto_explain`
 - запись в журнал планов выполнения запросов
 - трассировка вложенных запросов
- Настройки
 - планы долгих команд `auto_explain.log_min_duration`
 - вложенные запросы `auto_explain.log_nested_statements`
 - ...

Трассировка PL/pgSQL

- Расширение plpgsql_check
 - накладные расходы на запись сообщений
 - большой объем выдачи
- Основные настройки
 - включение трассировки
 - plpgsql_check.enable_tracer
 - plpgsql_check.tracer
 - уровень сообщений
 - plpgsql_check.tracer_errlevel

Итоги

- PL/pgSQL Debugger – API отладчика, используется в графических средах разработки
- Служебные сообщения – вывод на консоль, запись в журнал сообщений сервера, в таблицу или в файл, передача другим процессам
- Возможность трассировки сеансов

Практика №28

1. Измените функцию `get_catalog` так, чтобы динамически формируемый текст запроса записывался в журнал сообщений сервера.

В приложении выполните несколько раз поиск, заполняя разные поля, и убедитесь, что команды SQL формируются правильно.

2. Включите трассировку команд SQL на уровне сервера.

Поработайте в приложении и проверьте, какие команды попадают в журнал сообщений.

Выключите трассировку.

Практика №29

1. Включите трассировку PL/pgSQL-кода средствами расширения `plpgsql_check` и проверьте ее работу на примере нескольких подпрограмм, вызывающих одна другую.
2. При выводе отладочных сообщений из PL/pgSQL-кода удобно понимать, к какой подпрограмме они относятся. В демонстрации имя функции выводилось вручную. Реализуйте функционал, автоматически добавляющий к тексту сообщений имя текущей функции или процедуры.

Разграничение доступа

Обзор разграничения доступа


Роли и атрибуты

- Роль – пользователь СУБД
 - роль не связана с пользователем ОС
- Свойства роли определяются атрибутами
- LOGIN возможность подключения
 - SUPERUSER суперпользователь
 - CREATEDB возможность создавать базы данных
 - CREATEROLE возможность создавать роли
 - REPLICATION использование протокола репликации
 - и другие

Подключение

1. Строки `pg_hba.conf` просматриваются сверху вниз
2. Выбирается первая запись, которой соответствуют параметры подключения (тип, база, пользователь и адрес)

#	TYPE	DATABASE	USER	ADDRESS	METHOD
	local	all	postgres		peer
	local	all	all		peer
	host	all	all	127.0.0.1/32	md5
	host	all	all	:::1/128	md5
<hr/>					
	local — сокет		all — любая роль		
	host — TCP/IP		имя роли		
<hr/>					
		all — любая БД			
		имя БД			
<hr/>					
				all — любой IP	
				IP/маска	
				доменное имя	



Подключение

3. Выполняется аутентификация указанным методом

4. Удачно — доступ разрешается, иначе — запрещается
(если не подошла ни одна запись — доступ запрещается)

#	TYPE	DATABASE	USER	ADDRESS	METHOD
	local	all	postgres		peer
	local	all	all		peer
	host	all	all	127.0.0.1/32	md5
	host	all	all	:::1/128	md5

trust — верить

reject — отказать

scram-sha-256 и md5 — запросить пароль

peer — спросить ОС

Парольная аутентификация

- На сервере
 - пароль устанавливается при создании роли или позже
 - пользователю без пароля будет отказано в доступе
 - пароль хранится в системном каталоге pg_authid
- Ввод пароля на клиенте
 - вручную
 - из переменной окружения PGPASSWORD
 - из файла ~/.pgpass (строки в формате узел:порт:база:роль:пароль)

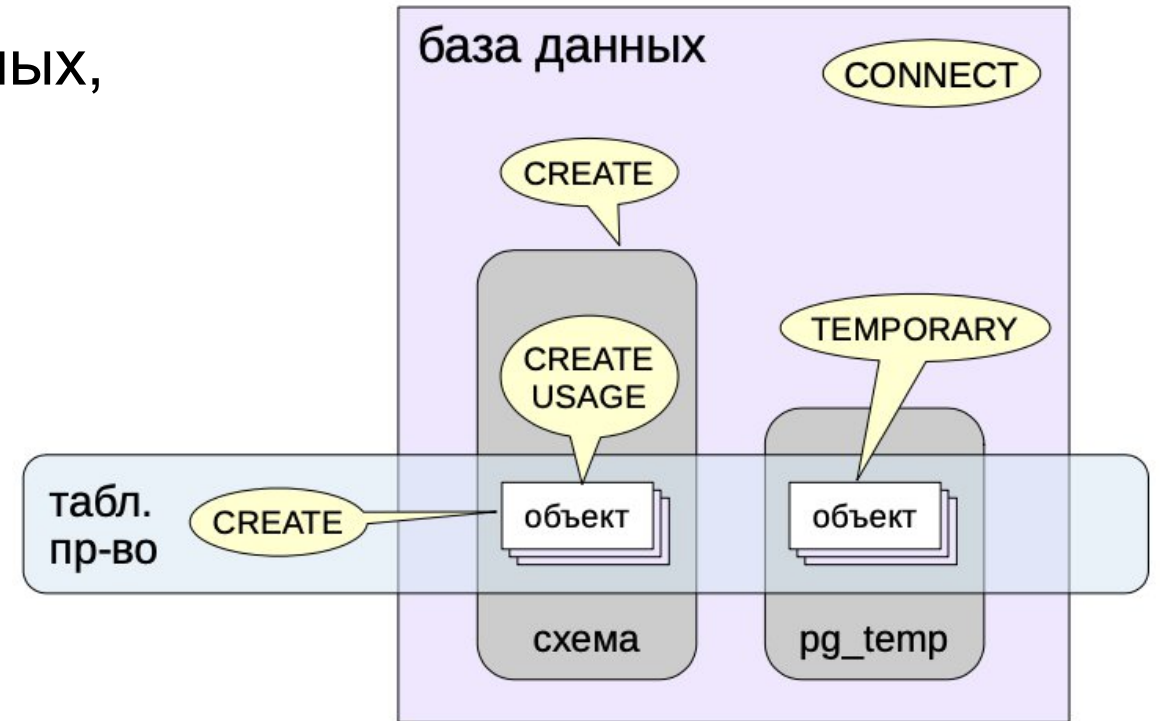
Привилегии

- Привилегии определяют права доступа ролей к объектам
- Таблицы
 - SELECT чтение данных
 - INSERT вставка строк
 - UPDATE изменение строк
 - REFERENCES внешний ключ
 - DELETE удаление строк
 - TRUNCATE опустошение таблицы
 - TRIGGER создание триггеров
- Представления
 - SELECT чтение данных
 - TRIGGER создание триггеров

} можно на уровне столбцов

Привилегии

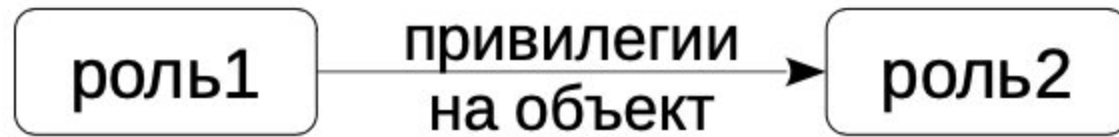
- Табличные пространства, базы данных, схемы
- Последовательности
 - SELECT currval
 - UPDATE nextval setval
 - USAGE currval nextval



Управление привилегиями

- Выдача привилегий

- роль1: GRANT привилегии ON объект TO роль2;



- Отзыв привилегий

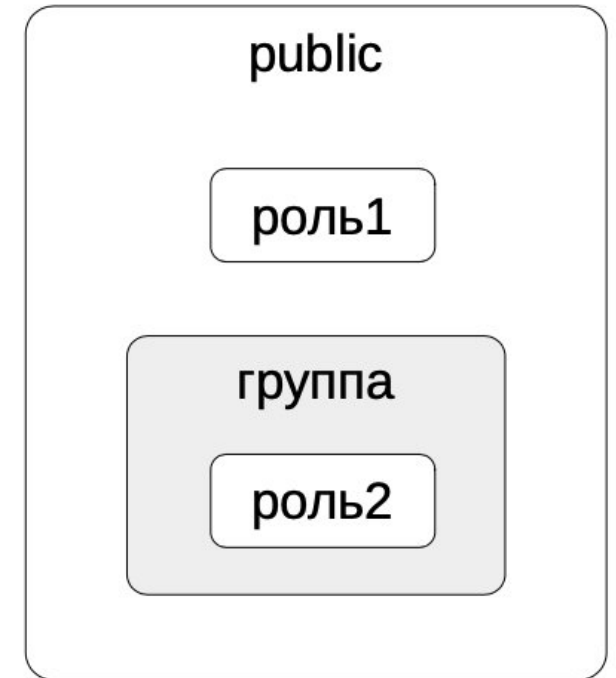
- роль1: REVOKE привилегии ON объект FROM роль2;

Групповые роли

- Включение роли в группу
 - роль1: GRANT группа TO роль2;



- псевдороль public неявно включает в себя все остальные роли
- Исключение роли из группы
 - роль1: REVOKE группа FROM роль2;



Суперпользователи

- Кто входит в категорию
 - роли с атрибутом SUPERUSER
- Права
 - полный доступ ко всем объектам – проверки не выполняются

Владелец объекта

- Кто входит в категорию
 - изначально – роль, создавшая объект (можно сменить)
 - а также роли, включенные в роль владельца
- Права
 - изначально все привилегии для объекта (можно отозвать)
 - действия со своими объектами, не регламентированные привилегиями, например: удаление объекта, выдача и отзыв привилегий и т. п.

Остальные роли

- Кто входит в категорию
 - все остальные (не суперпользователи и не владельцы)
- Права
 - доступ в рамках выданных привилегий
 - обычно наследуют привилегии групповых ролей (но атрибут NOINHERIT требует явного переключения роли)

Подпрограммы

- Единственная привилегия для функций и процедур
 - EXECUTE выполнение
- Характеристики безопасности
 - SECURITY INVOKER выполняется с правами вызывающего (по умолчанию)
 - SECURITY DEFINER выполняется с правами владельца

Привилегии по умолчанию

- Привилегии псевдороди public
 - подключение к любой базе данных
 - доступ к схеме public и создание в ней объектов
 - доступ к системному каталогу
 - выполнение любых подпрограмм
 - привилегии выдаются автоматически для каждого нового объекта
- Настраиваемые привилегии по умолчанию
 - возможность дополнительно выдать или отозвать привилегии для только что созданного объекта

Политики защиты строк

- Дополнение к системе привилегий для разграничения доступа к таблицам на уровне строк
- Политика применяется
 - к определенным ролям
 - к определенным командам (SELECT, INSERT, UPDATE, DELETE)
- Политика определяет условие доступности строки
 - разрешительная: позволяет видеть строку, если условие выполнено
 - ограничительная: запрещает видеть строку, если условие не выполнено
 - отдельные условия (предикаты) для существующих и для новых строк

Итоги

- Роли, привилегии и политики – гибкий механизм, позволяющий по-разному организовать работу
 - можно легко разрешить все всем
 - можно строго разграничить доступ, если это необходимо
- При создании новых ролей надо позаботиться о возможности их подключения к серверу

Практика №30

1. Создайте две роли (пароль должен совпадать с именем):

- employee – сотрудник магазина,

- buyer – покупатель.

Убедитесь, что созданные роли могут подключиться к БД.

2. Отзовите у роли public права выполнения всех функций и подключения к БД.

3. Разграничьте доступ таким образом, чтобы:

- сотрудник мог только заказывать книги, а также – добавлять авторов и книги,

- покупатель мог только приобретать книги.

Проверьте выполненные настройки в приложении.

Практика №31

Подпрограммы, объявленные как выполняющиеся с правами владельца (SECURITY DEFINER), могут использоваться, чтобы предоставить обычным пользователям возможности, доступные только суперпользователю.

1. Создайте обычного непривилегированного пользователя и проверьте, что он не может изменять значение параметра `log_statement`.
2. Напишите подпрограмму для включения и выключения трассировки SQL-запросов так, чтобы созданная роль могла ей воспользоваться.

Резервное копирование

Логическое резервирование

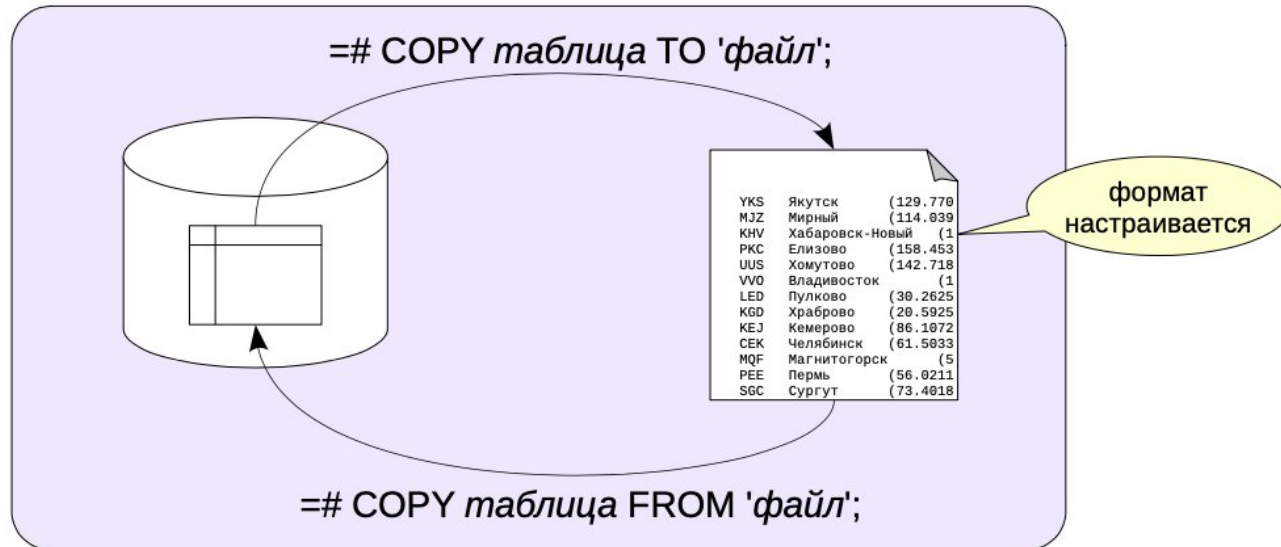
Логическая копия

- Команды SQL для создания объектов и наполнения данными
 - + можно сделать копию отдельного объекта или отдельной базы
 - + можно восстановиться на другой версии или архитектуре (не требуется двоичная совместимость)
 - + простота использования
 - невысокая скорость работы
 - восстановление только на момент создания резервной копии

Физическая копия

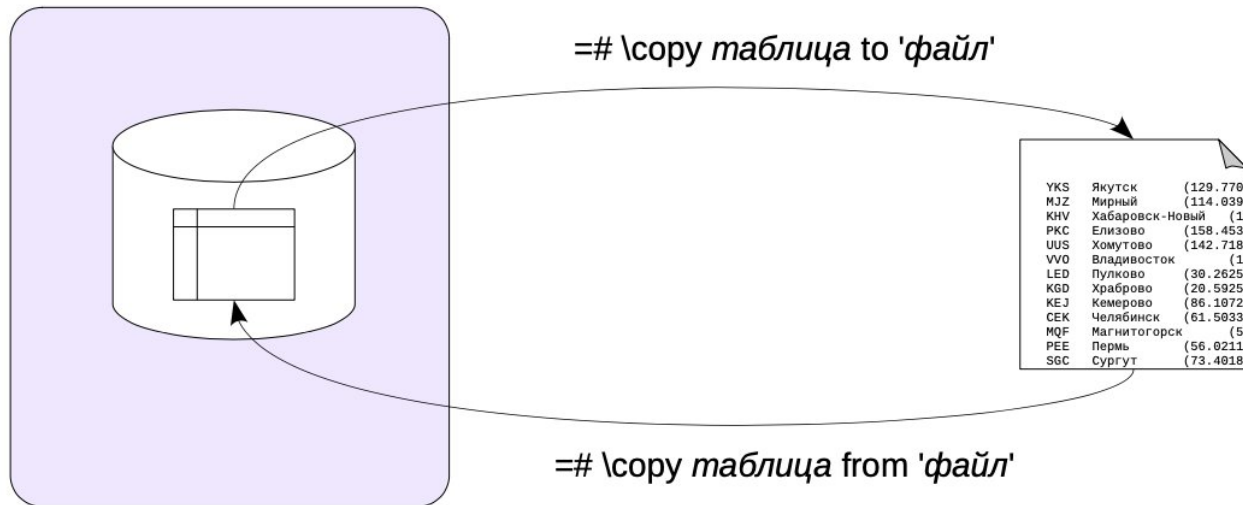
- Копия файловой системы кластера баз данных
 - + быстрее, чем логическое резервирование
 - + восстанавливается статистика
 - можно восстановиться только на совместимой системе
 - и на той же самой основной версии PostgreSQL
 - выборочная копия невозможна, копируется весь кластер
- Архив журнала предзаписи
 - + возможность восстановления на определенный момент времени

Копия таблицы в SQL



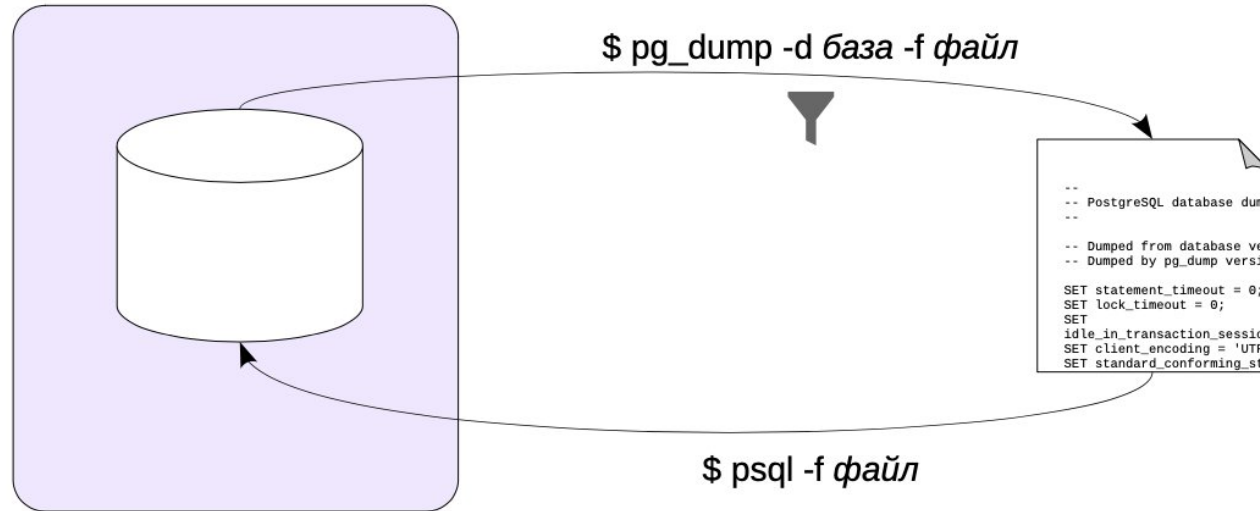
- файл в ФС сервера и доступен владельцу экземпляра PostgreSQL
- можно ограничить столбцы (или использовать произвольный запрос)
- при восстановлении строки добавляются к имеющимся в таблице

Копия таблицы в psql



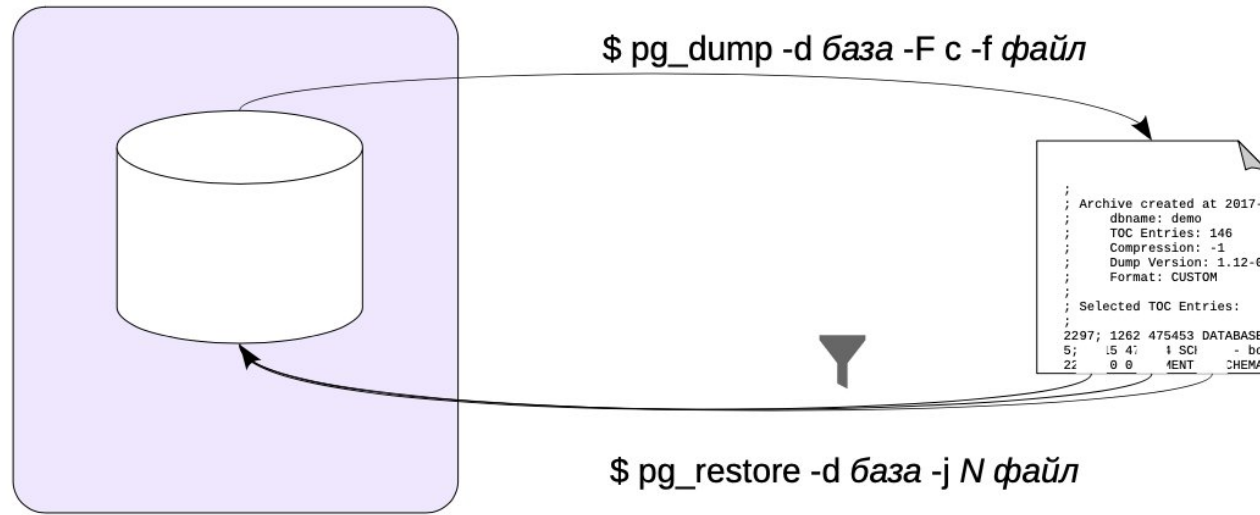
- файл в ФС клиента и доступен пользователю ОС, запустившему psql
- происходит пересылка данных между клиентом и сервером
- синтаксис и возможности аналогичны команде COPY

Копия базы данных



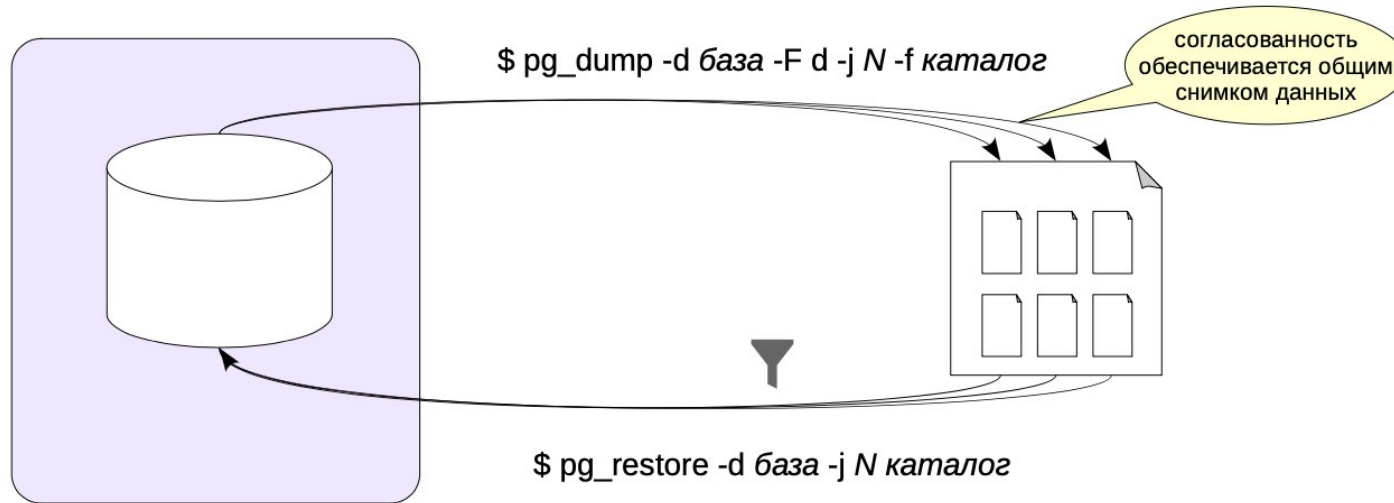
- формат: команды SQL
- при выгрузке можно выбрать отдельные объекты базы данных
- новая база должна быть создана из шаблона `template0`
- заранее должны быть созданы роли и табличные пространства
- после загрузки имеет смысл выполнить `ANALYZE`

Формат custom



- внутренний формат с оглавлением
- отдельные объекты базы данных можно выбрать на этапе восстановления
- возможна загрузка в несколько параллельных потоков

Формат directory

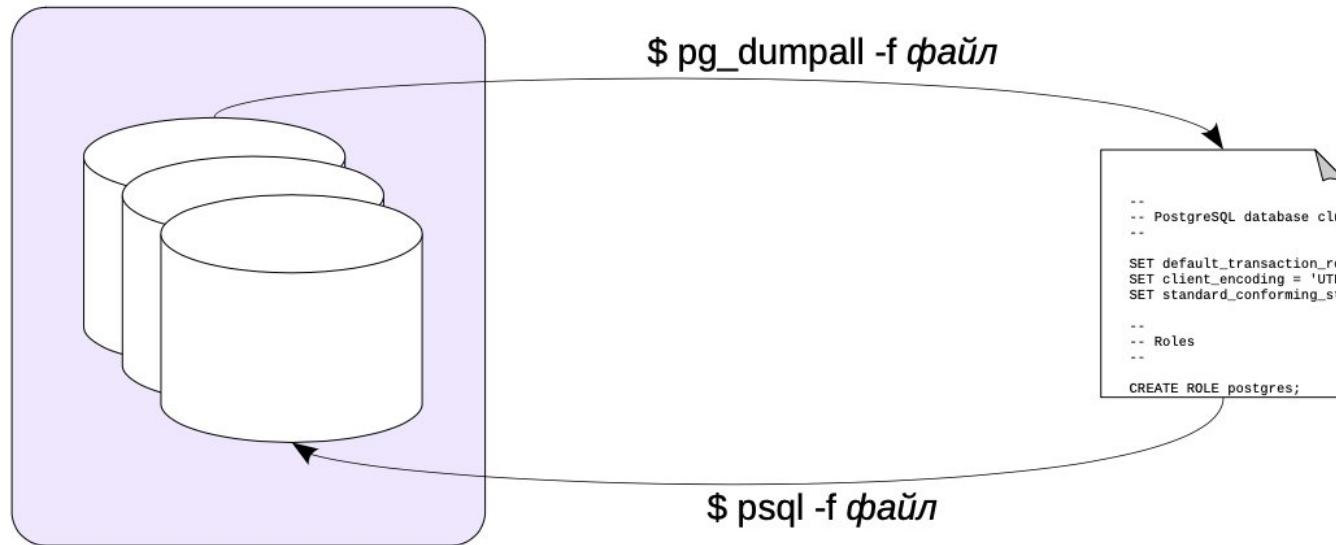


- каталог с оглавлением и отдельными файлами на каждый объект базы
- отдельные объекты базы данных можно выбрать на этапе восстановления
- и выгрузка, и загрузка возможны в несколько параллельных потоков

Сравнение форматов

	plain	custom	directory	tar
утилита для восстановления	psql	pg_restore		
сжатие	zlib			
выборочное восстановление		да	да	да
параллельное резервирование			да	
параллельное восстановление		да	да	

Копия кластера БД



- формат: команды SQL
- выгружает весь кластер, включая роли и табличные пространства
- пользователь должен иметь доступ ко всем объектам кластера
- не поддерживает параллельную выгрузку

Итоги

- Логическое резервирование позволяет сделать копию всего кластера, базы данных или отдельных объектов
- Хорошо подходит
 - для данных небольшого объема
 - для длительного хранения, за время которого меняется версия сервера
 - для миграции на другую платформу
- Плохо подходит
 - для восстановления после сбоя с минимальной потерей данных

Практика №32

1. Создайте резервную копию базы данных bookstore в формате custom.

«Случайно» удалите все записи из таблицы authorship. Проверьте, что приложение перестало отображать названия книг на вкладках «Магазин», «Книги», «Каталог».

Используйте резервную копию для восстановления потерянных данных в таблице.

Проверьте, что нормальная работа книжного магазина восстановилась.

Практика №33

1. Создайте таблицу с политикой, разрешающей чтение только части строк. Создайте непривилегированного пользователя для Алисы и предоставьте ей доступ к таблице.

В обязанности Алисы входит резервное копирование таблицы. Сможет ли она выполнять их, не являясь суперпользователем? Проверьте.

2. Команда `psql \copy` позволяет направить результат на вход произвольной программы. Воспользуйтесь этим, чтобы открыть результаты какого-нибудь запроса в электронной таблице LibreOffice Calc.