Расширенный курс для разработчиков PostgreSQL

Преподаватель:

Валентин Степанов

Слабоструктурированные (semi-structured) данные

- Наиболее известные примеры: XML и JSON
- Схема данных определяется самим документом
  - а не хранится отдельно от данных
  - документ содержит разметку для выделения элементов и определения иерархии
- Данные не соответствуют реляционной модели
  - конкретный документ можно представить в виде таблиц
  - документы общего вида тоже можно, но практически — неудобно

Применение

- Интерфейсный формат
  - между приложением и внешней системой
  - между клиентской и серверной частями приложения
- Внутри базы данных как атомарное значение
  - только хранение и извлечение
  - не нарушает 1NF, достаточно обычных средств SQL
- Внутри базы данных как неатомарное значение
  - операции с отдельными частями документа
  - SQL не достаточно; требуется специальный язык запросов и набор дополнительных операций
  - гибкость, когда данные плохо укладываются в реляционную модель

Операции

- Преобразование документа к реляционному виду
  - сохранение транспортного документа в базу данных
  - использование средств SQL для обработки
- Преобразование реляционных данных к виду документа
  - выгрузка данных в транспортный формат
- Выделение части документа
  - специализированный язык запросов
- Индексирование документов
  - поддержка операций специализированного языка запросов

Формат XML

- eXtensible Markup Language
  - универсальный язык разметки, первый стандарт 1998 года
  - форматы на основе XML: XHTML, FB2, RSS и Atom, SVG, SOAP...
- Структура
  - вложенные элементы
  - отделяются тегами, могут иметь атрибуты, содержат текст
- Инструментарий
  - описание схемы (DTD, XML Schema), языки запросов XPath и XQuery, язык преобразования XSLT
  - PostgreSQL: XPath 1.0

Формат JSON

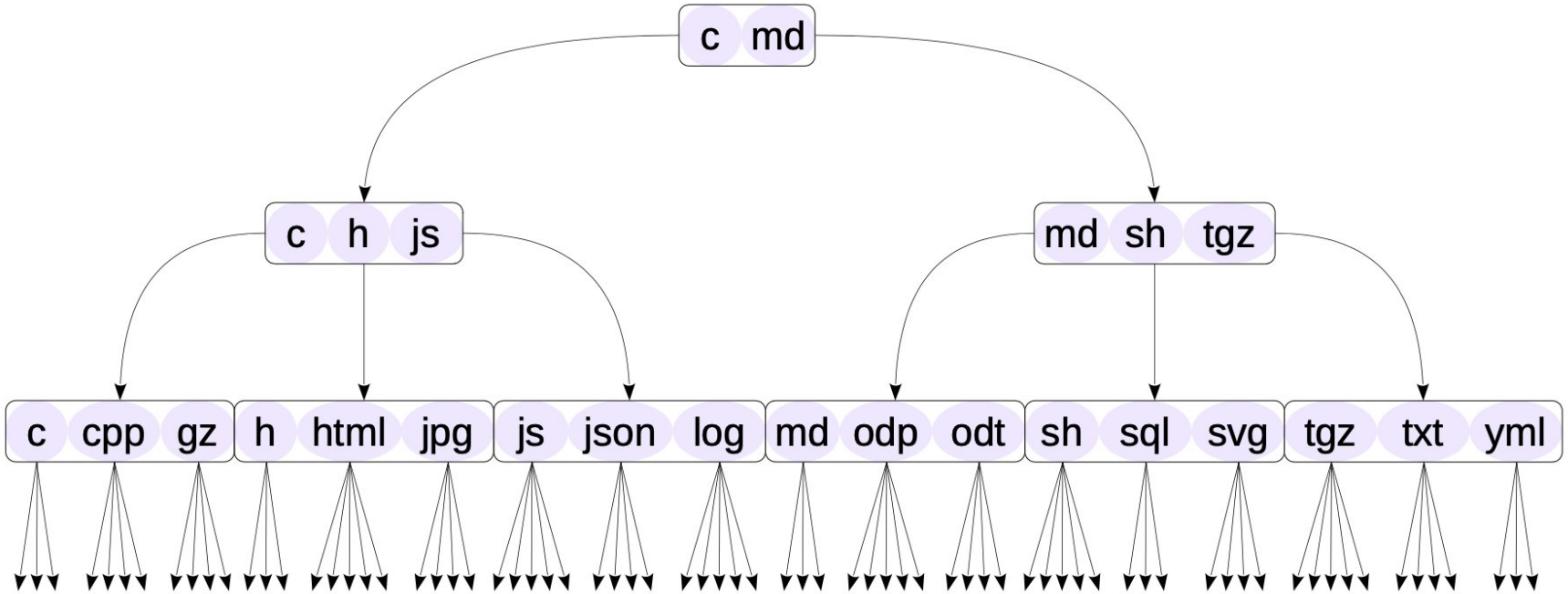
- JavaScript Object Notation
  - простой язык разметки, стандарт 1999 года
  - появился в JavaScript, но распространен повсеместно
- Структура
  - объекты (пары «ключ: значение»), массивы значений
  - значения текстовые, числовые, даты, логические
- Инструментарий
  - PostgreSQL: язык запросов JSONPath, неполная пока поддержка стандарта SQL/JSON, индексирование

Типы данных для JSON

- Json
  - хранение в виде обычного текста + синтаксическая проверка
  - сохраняется исходное форматирование
  - необходимость повторного разбора при каждом запросе
- Jsonb
  - внутренний формат, исключающий повторный разбор
  - поддержка SQL/JSON (язык запросов JSONPath)
  - поддержка индексирования

Метод доступа GIN

- Инвертированный индекс
  - метод применим для сложносоставных значений (массив, текст)
  - API метода доступа определяет, как выделять элементы значения



Итоги

- PostgreSQL поддерживает слабоструктурированные типы
- Частичная поддержка XML
  - формат теряет популярность
- Полноценная поддержка JSON
  - направление активно развивается
- Индексирование на основе метода доступа GIN

Практика

- Часть сведений о книгах (ISBN, аннотация и другие) выводятся приложением только на отдельной странице. Сейчас эти данные хранятся в таблице books, но не передаются приложению. Измените функцию, реализующую API приложения, так, чтобы сведения передавались в виде одного объекта JSON. Проверьте реализацию в приложении.
- Многочисленные столбцы, хранящие дополнительные сведения о книгах, никак не используются в серверной части приложения. Замените их на один столбец, в котором сведения будут храниться в формате JSON. Сделайте это прозрачно для приложения.

Практика

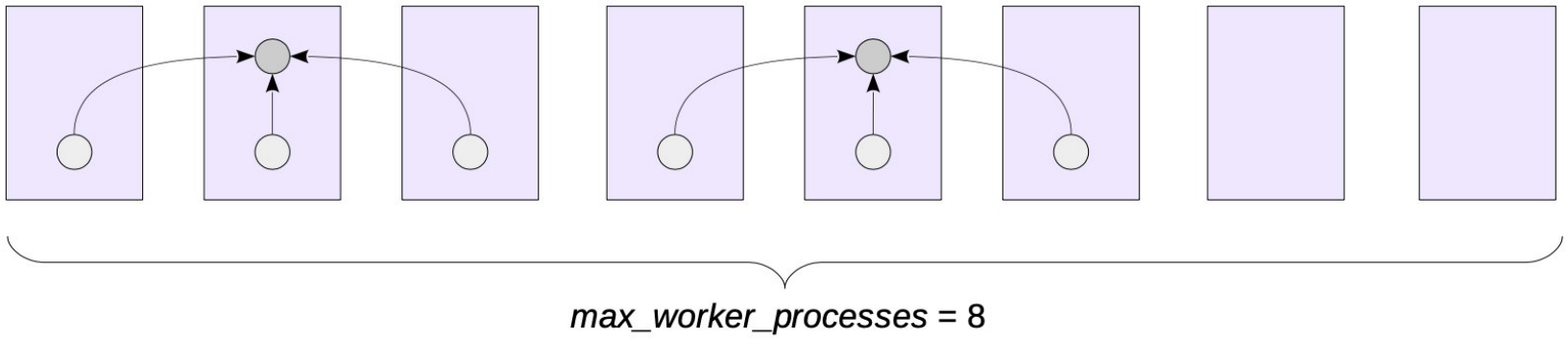
- В базе данных имеются таблицы пользователей и их заказов, связанные отношением «один ко многим».
  Напишите функцию, по идентификатору пользователя возвращающую документ JSON, содержащий все сведения о пользователе вместе с массивом его заказов.
- В пользовательском интерфейсе необходимо выводить названия городов, хранящиеся в базе данных, на выбранном языке. Один из способов — хранить все переводы названий в объекте JSON в виде пар «код языка — название».
  Реализуйте такой способ. Для удобства создайте представление, показывающее название городов на языке, заданном в конфигурационном параметре сервера.

Фоновые процессы

- Фиксированный набор служебных процессов
  - postmaster
  - walwriter
  - checkpointer
  - autovacuum
  - и другие
- Динамически порождаемые фоновые процессы
  - контролируются процессом postmaster
  - имеют доступ к разделяемой памяти
  - могут устанавливать внутреннее соединение с базами данных
  - реализуются на языке C

Использование

- Параллельное выполнение запросов
  - max_parallel_workers = 8
- Параллельное выполнение служебных команд
  - max_parallel_maintenance_workers = 2
- Логическая репликация
  - max_logical_replication_workers = 4



Прикладные задачи

- Некоторые идеи
  - планировщик заданий внутри СУБД
  - асинхронная обработка событий
  - распараллеливание сложной обработки данных
  - автономные транзакции
  - и другое
- Но пользовательские задачи неудобно писать на языке C

Расширение dblink

- Назначение
  - выполнение произвольных SQL-команд на удаленном сервере
  - в качестве удаленного сервера может выступать и локальный
- Плюсы
  - стандартное расширение
- Минусы
  - фоновые процессы не используются
  - устанавливается новое соединение

Расширение pg_background

- Назначение
  - возможность выполнить команду SQL в фоновом процессе
  - команда может вызвать подпрограмму на любом серверном языке
- Плюсы
  - используются фоновые процессы
- Минусы
  - стороннее расширение
- https://github.com/vibhorkum/pg_background

Итоги

- Сервер предоставляет механизм фоновых процессов
- Фоновые процессы используются для внутренних целей, но могут быть использованы и для прикладных задач
- Расширение pg_background позволяет реализовывать фоновые процессы на процедурных языках

Практика

- Реализуйте расчет рейтинга книг на основе данных о голосах пользователей «за» и «против».
  Чтобы не допускать разрастание таблицы, выполняйте обновление пакетами по N строк. Между частичными обновлениями запускайте очистку.
  Оформите обновление в виде хранимой процедуры, принимающей N как параметр.
- Выполните процедуру обновления в фоновом режиме.

Практика

- Фоновый процесс не будет запущен, если запуск приведет к превышению числа допустимых процессов. Напишите функцию-обертку для pg_background_launch, которая пытается запустить процесс N раз с интервалом в одну секунду.
- Сравните накладные расходы на порождение процесса расширений dblink и pg_background, многократно выполняя простой запрос.

Асинхронная обработка

- Разнесение во времени возникновения события и его обработки
- Соображения производительности
  - клиенту не требуется ждать ответа
  - возможность управлять ресурсами для обработки
- Реализация
  - очередь сообщений
  - более сложный вариант: модель публикация – подписка

Внешние системы

- RabbitMQ, ActiveMQ, ZeroMQ и т. п.
- Плюсы
  - эти системы работают
  - следование стандартам (AMQP, JMP, STOMP, MQTT...)
  - гибкость, масштабирование, производительность
- Возможные минусы
  - отдельная система (включающая отдельную СУБД) со своими особенностями настройки, администрирования, мониторинга
  - все сложности построения распределенных систем (отсутствие глобальных транзакций)

Очередь внутри базы: PgQ

- Плюсы
  - эта система работает
- Возможные минусы
  - мало гибкости, например, исключительно пакетная обработка
  - плохо документирована
  - внешняя программа-демон
  - избыточно сложная реализация в расчете на старые версии PostgreSQL
- https://github.com/pgq

Очередь своими руками

- Возможные плюсы
  - не требуются внешние зависимости
  - простые требования — простая реализация
- Минусы
  - требуется отладка и тестирование
  - при усложнении требований готовая система может обойтись дешевле

Вспоминаем про горизонт

- Что получилось: одна большая транзакция

```
take_message();
--обработка
complete_message();

take_message();
--обработка
complete_message();

take_message();
--обработка
complete_message();

COMMIT;
```

Вспоминаем про горизонт

- Что надо: каждое событие в отдельной транзакции

```
take_message();
--обработка
complete_message();

COMMIT;
```

```
take_message();
--обработка
complete_message();

COMMIT;
```

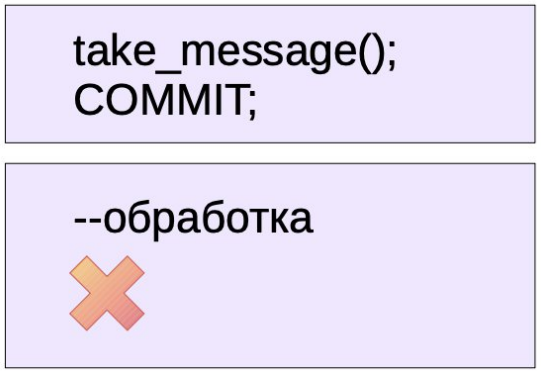
```
take_message();
--обработка
complete_message();

COMMIT;
```

Вспоминаем про горизонт

- Еще лучше: позволить обработке события состоять из нескольких транзакций

Чего не хватает

- Зависшие сообщения
  - остаются в статусе «в работе» при аварийном завершении обработчика



- Решение
  - проверять существование процесса-обработчика, указанного в таблице
  - при отсутствии возвращать сообщение в статус «новый» (возможно)

Чего не хватает

- Корректная обработка исключительных ситуаций
- Сохранение результатов обработки
- Решение
  - не удалять обработанные сообщения, а помечать отдельным статусом
  - потребуется правильный индекс
  - потребуется периодическая очистка исторических данных
  - обращаем внимание на автоочистку

Итоги

- Асинхронная обработка полезна во многих случаях
- Внешние системы имеет смысл использовать, если
  - они вписываются в общую архитектуру информационной системы
  - предъявляются серьезные требования
- Очередь сообщений в базе данных — простое решение для простых задач
  - важна правильная реализация:
    эффективное получение очередного события (SKIP LOCKED),
    избегание долгих транзакций
  - чем больше требований, тем сложнее будет реализация
  - обратить внимание на настройку автоочистки для таблицы очереди

Практика

- В приложении предусмотрен механизм фоновых заданий, но серверная часть обработки очереди отсутствует. Напишите недостающие функции:
  - take_task – получает очередное задание из очереди;
  - complete_task – завершает обработку задания;
  - process_tasks – основной цикл обработки заданий.
- Запустите процедуру обработки очереди заданий в фоновом режиме. Проверьте, что фоновые задания, поставленные в очередь в приложении, выполняются, а результаты их работы доступны для просмотра.

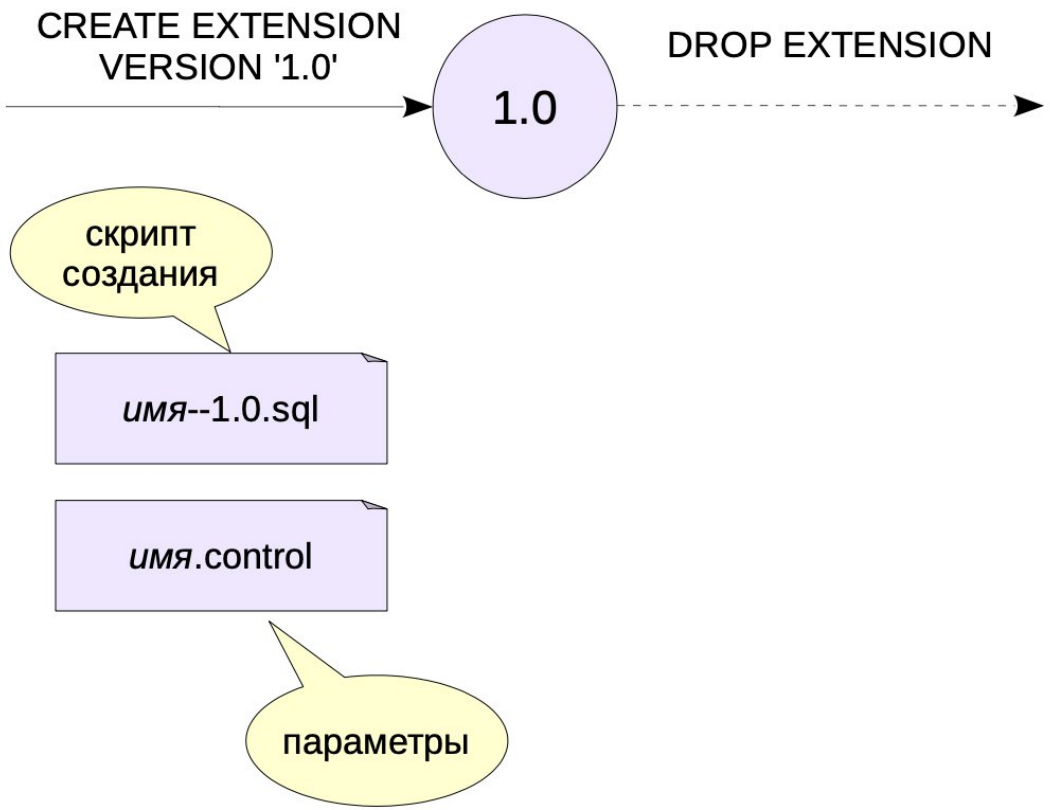
Практика

- Напишите тест, проверяющий, что обработка очереди, показанная в демонстрации, работает корректно при выполнении в несколько потоков.
  Убедитесь, что тест не проходит, если убрать предложение FOR UPDATE SKIP LOCKED.
- Добавьте в реализацию проверку «зависших» сообщений. Если такая ситуация будет обнаружена, зависшее сообщение должно быть снова принято в работу.

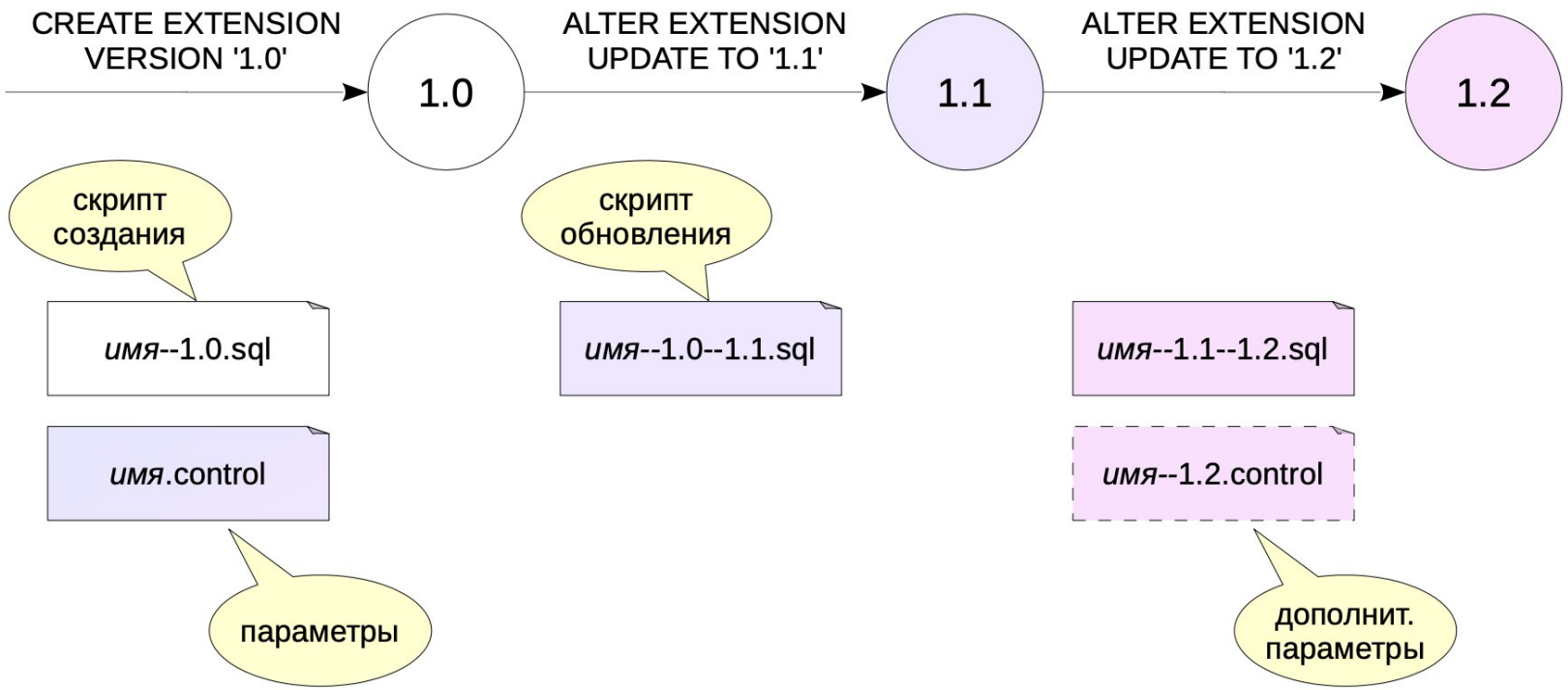
Расширения

- Группа взаимосвязанных объектов БД
  - установка всех объектов одной командой
  - невозможность удалить отдельный объект
  - сохранение связи при выгрузке с помощью pg_dump
  - инструменты для перехода на новую версию
- Источники
  - в составе дистрибутива (contrib)
  - внешние расширения
  - возможность создания собственных расширений

Создание расширения



Обновление расширения



Итоги

- Расширения — упаковка взаимосвязанных объектов БД, предназначенных для решения какой-либо задачи
- Расширения упрощают использование функционала
- Имеются средства для разработки собственных расширений

Практика

- Создайте расширение bookfmt, содержащее все необходимое для работы с типом данных для формата изданий.
  Установите расширение, чтобы объединить имеющиеся в базе данных разрозненные объекты.
- Воспользуйтесь стандартным расширением isn для того чтобы проверить корректность кодов ISBN у имеющихся в магазине книг.

Практика

- Создайте расширение с функцией для подготовки текста к публикации. Функция должна применить к текстовому параметру последовательность правил, выполняющих замену по регулярному выражению, и вернуть результат.
  Правила должны храниться в таблице и применяться в порядке их вставки. К предопределенным правилам пользователь может добавлять собственные.
  Работа функции не должна зависеть от настройки пути поиска, но должна позволять выбрать схему при установке.
- Установите расширение в схему typo, добавьте в таблицу пользовательское правило. Корректно ли выгружает копию базы данных утилита pg_dump? Проверьте возможность добавить правило после восстановления из резервной копии.

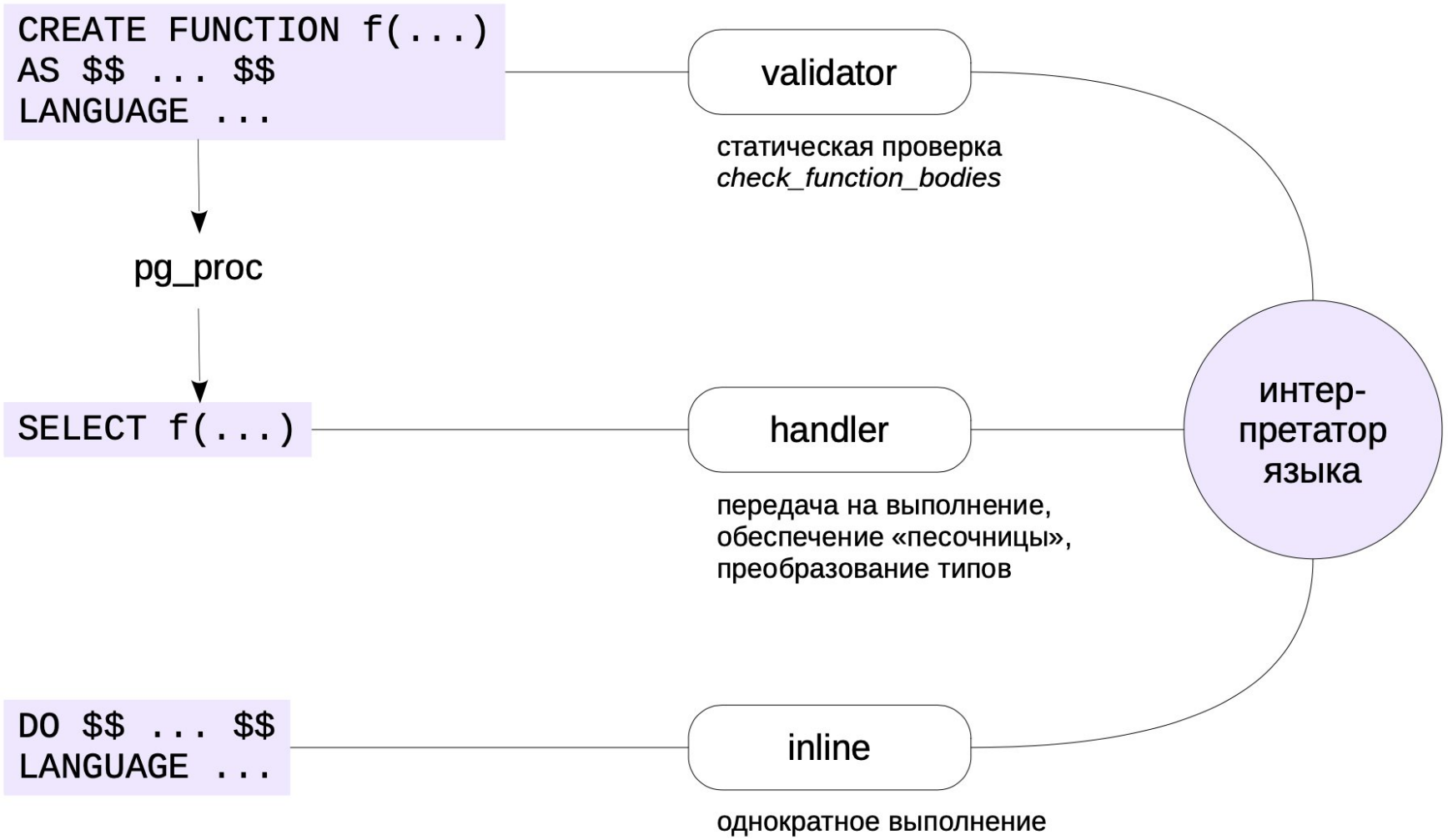
Языки программирования

- «Встроенные»
  - C
  - SQL
- Стандартные (поддержка сообщества)
  - PL/pgSQL
  - PL/Perl, PL/Python, PL/Tcl
- Сторонние
  - PL/Java, PL/V8, PL/R, PL/Lua, ...

Доверенные языки

- Доверенные
  - гарантируют работу пользователя в рамках выданного доступа
  - ограничено взаимодействие с окружением и внутренностями СУБД, язык должен позволять работать в «песочнице»
  - по умолчанию доступ для всех пользователей
- Недоверенные
  - доступна полная функциональность языка
  - доступ только у суперпользователей
  - обычно к названию языка добавляют «u»: plperl → plperlu

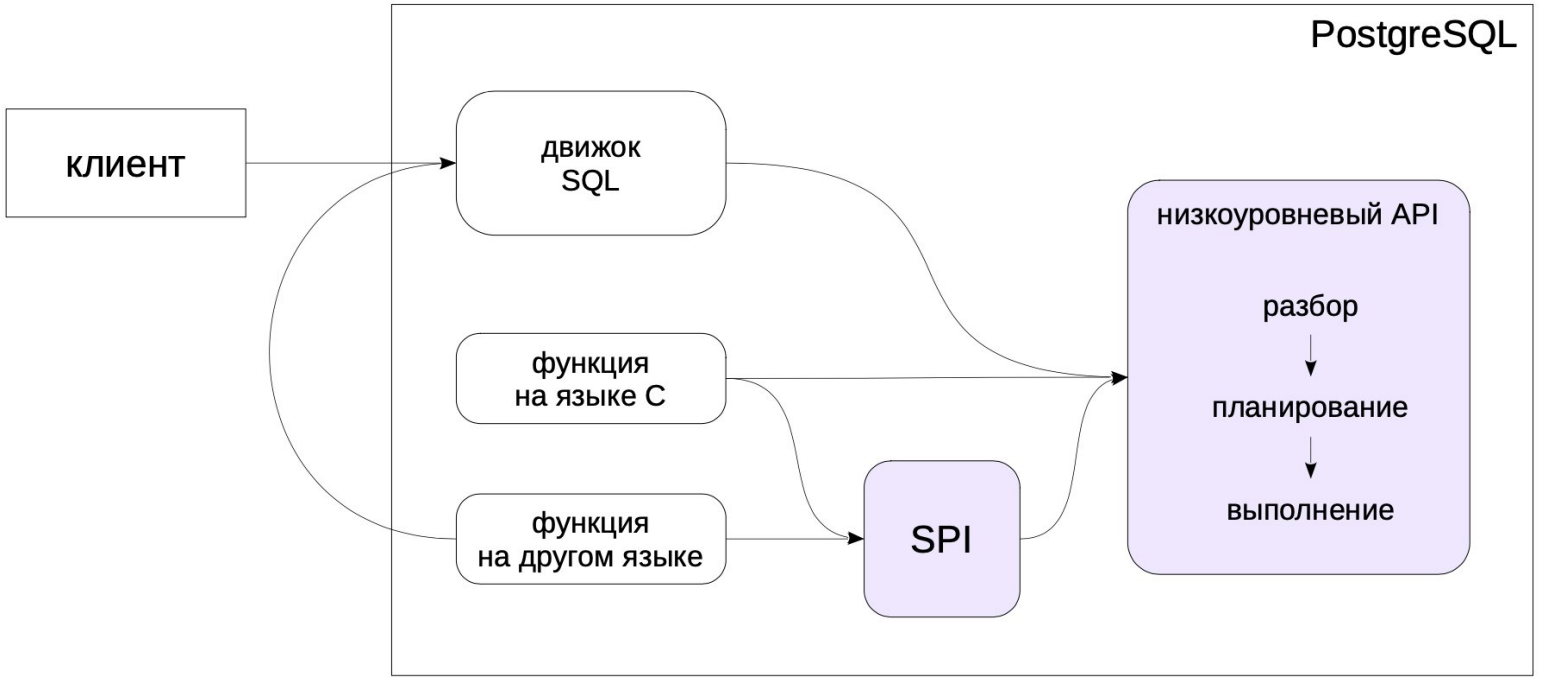
Подключение нового языка



Задачи

- Обработка информации, хранимой в базе данных
  - подпрограмма всегда выполняется в контексте подключения к БД
  - PL/pgSQL интегрирован с SQL
  - для остальных — интерфейс серверного программирования (SPI)
- Вычисления, не связанные с базой данных
  - возможности PL/pgSQL сильно ограничены
  - эффективность
  - удобство использования
  - наличие готовых библиотек
  - специализированные задачи

Интерфейс SPI



Из чего выбирать

- Интенсивная работа с данными
  - SQL, PL/pgSQL
- Проверенные, часто используемые
  - PL/Perl, PL/Python, PL/V8, PL/Java
- Другие языки общего назначения
  - PL/Go, PL/Lua, ...
- Специальные задачи
  - PL/R (статистика), PL/Proxy (удаленные вызовы), ...
- Максимальная эффективность
  - C

Итоги

- PostgreSQL позволяет подключать любые языки
- Богатый выбор языков программирования позволяет решать любые задачи на стороне сервера

Практика

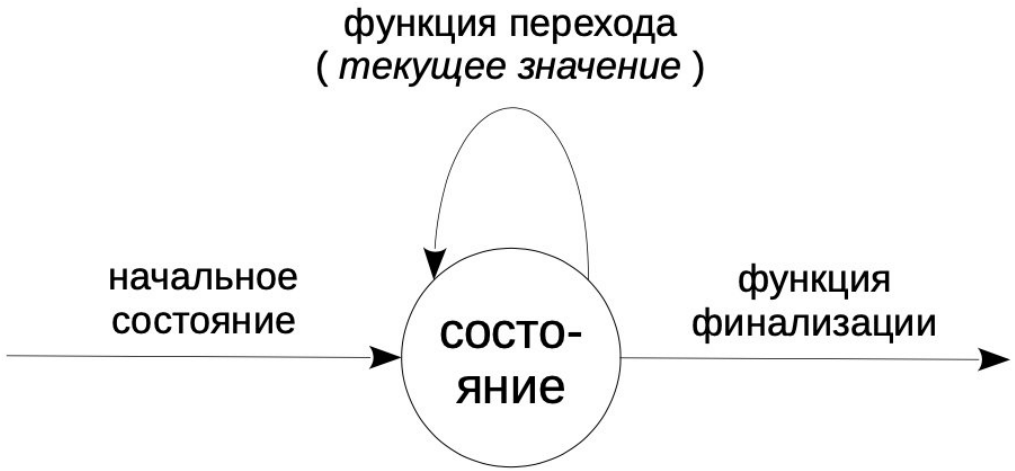
- Функция empapi.run, написанная на языке Python, принимает параметр типа jsonb. Добавьте трансформацию, чтобы избежать преобразований в текстовый вид и обратно.
- Отправляйте пользователю, совершившему покупку в магазине, письмо-подтверждение с указанием суммы.
  В виртуальной машине настроен локальный почтовый сервер, пересылающий любую исходящую почту в локальный ящик пользователя student. В PostgreSQL нет встроенной функции для отправки писем, но ее можно реализовать на каком-либо недоверенном языке.
  Убедитесь, что письмо не может быть отправлено до того, как транзакция покупки будет завершена.

Практика

- Языки Python и Perl имеют удобный тип данных — ассоциативный массив, — который отсутствует в PL/pgSQL. В Python это dict (словарь), в Perl — hash (хеш-таблица). Напишите на одном из этих языков функцию, получающую на вход текстовую строку и возвращающую таблицу из слов, которые встречаются в этой строке, с указанием количества вхождений.
  Решается ли эта задача на языке SQL?
- Напишите функцию, которая по имени файла определяет и выводит его MIME-тип, аналогично команде shell
  file --brief --mime-type

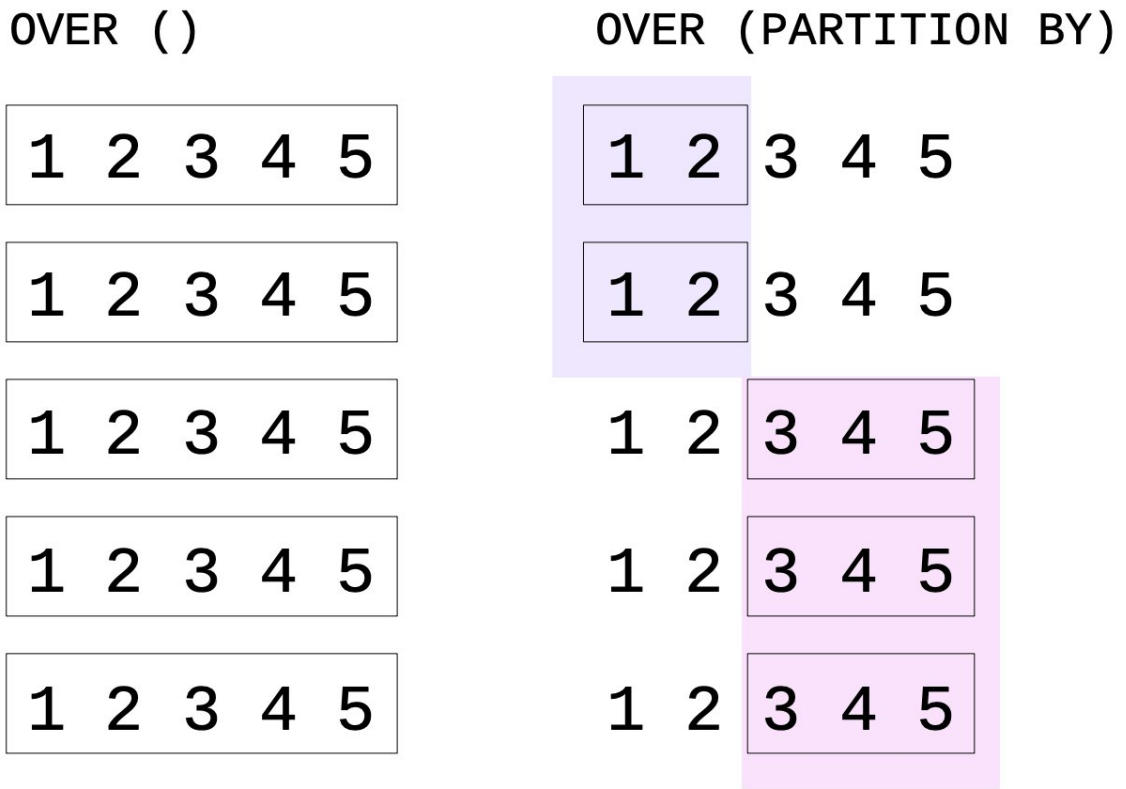
Агрегатные функции

- Построчная обработка агрегируемой выборки
  - состояние
  - функции перехода и финализации

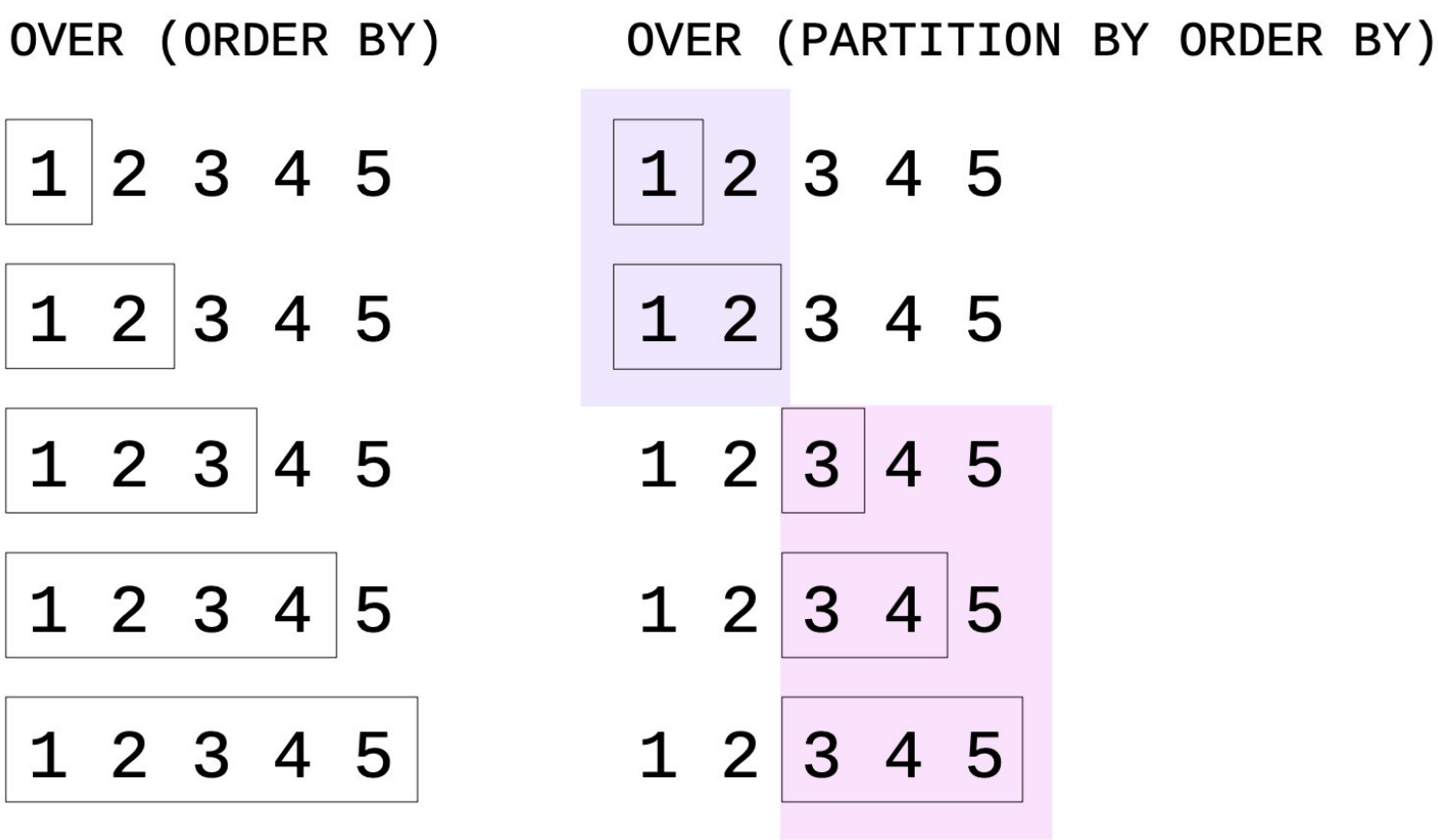


Оконные функции

- Окно определяет агрегируемую выборку для каждой строки



Нарастающий итог



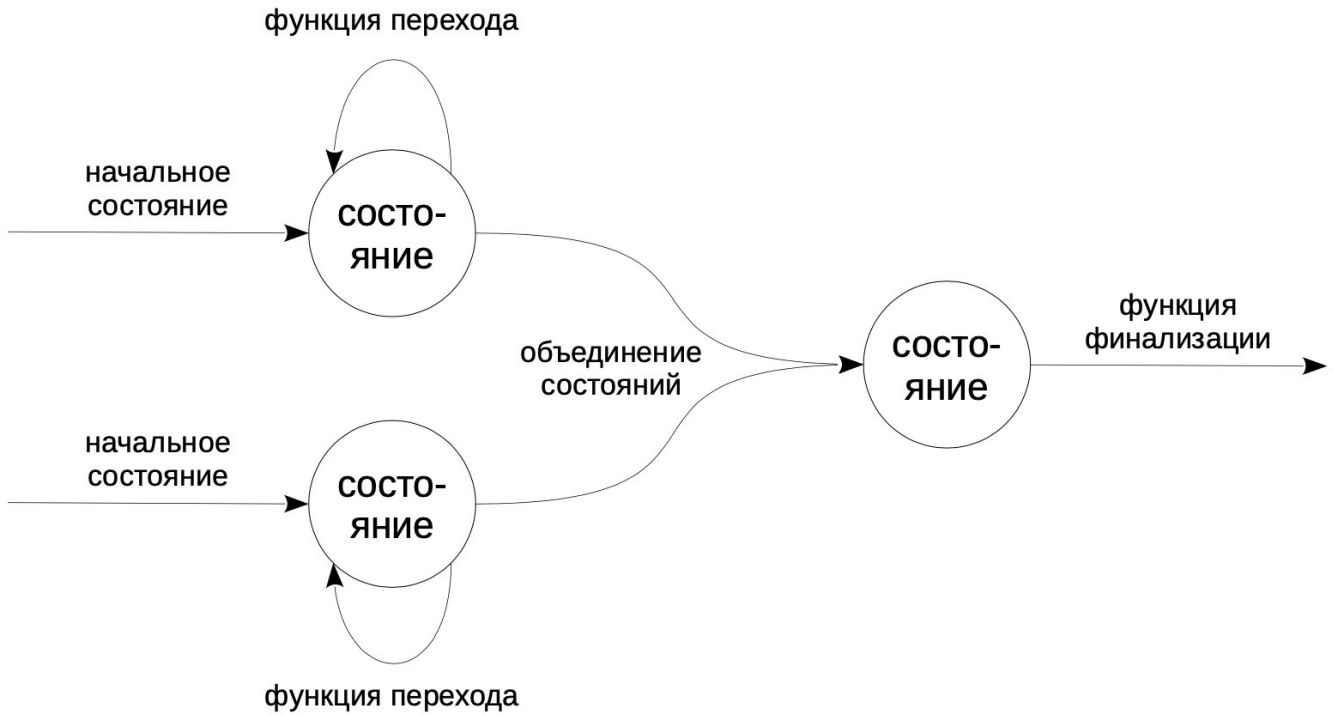
Скользящая рамка

- Могут двигаться «голова» и «хвост» рамки одновременно

```
OVER (ROW BETWEEN 2 PRECEDING AND CURRENT ROW)
```

Параллелизм

- Объединение состояний параллельных процессов



Итоги

- PostgreSQL позволяет создавать агрегатные и оконные функции
- Агрегатные и оконные функции дают возможность использовать процедурную обработку в стиле SQL

Практика

- Напишите отчет, выводящий складские остатки по каждой книге в денежном выражении (используя закупочную, а не розничную цену). Оформите отчет как фоновое задание. Учтите, что поступления происходят разными партиями по разной цене, а продажи никак не привязаны к партиям. Поэтому считайте, что в первую очередь продаются книги из более старых партий.
- Дополните расширение bookfmt функциями min и max для формата издания. Обновите версию расширения и убедитесь, что функции появились в базе данных.

Практика

- В банкомат загружено некоторое количество купюр определенных достоинств.
  Напишите функцию, возвращающую минимальный набор купюр, которыми банкомат может выдать указанную сумму.
- Бизнес-центр сдает офисы компаниям. В конце месяца администрации БЦ приходит общий счет за электроэнергию, который надо распределить между арендаторами пропорционально площади занимаемых помещений. Выставляемые арендаторам счета необходимо округлить до копеек, но так, чтобы их сумма совпала со значением, указанным в общем счете.

Текстовый поиск

- Средства SQL — LIKE, регулярные выражения
  - нет морфологического поиска
  - нет возможности ранжирования результатов
  - нет индексной поддержки
- Внешние поисковые системы
  - сложно синхронизировать с базой данных
  - отсутствие транзакционности
  - нет доступа к метаданным
  - сложности с разграничением доступа

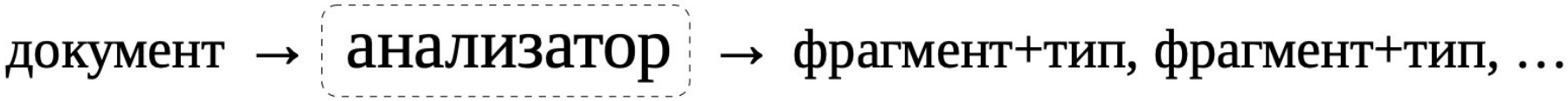
Документы и запросы

- Документ
  - произвольный текст, который можно разобрать на слова (лексемы)
  - docx, pdf и т. п. надо предварительно преобразовать в текст
  - внутреннее компактное представление tsvector
- Запрос
  - одна или несколько лексем, соединенных логическими связками:
    - & и
    - | или
    - ! не
    - <-> предшествует (фразовый поиск)
    - ( ) для изменения приоритета
  - внутреннее представление tsquery

Соответствие

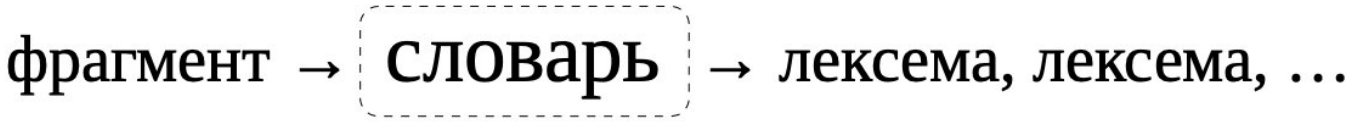
- Соответствие документа запросу
  - оператор @@
- Релевантность
  - веса лексем (A, B, C, D в порядке убывания важности)
  - ts_rank — по частоте найденных лексем
  - ts_rank_cd — также учитывает близость лексем
  - оператор расстояния <=> (расширение rum)

Анализатор



- Выделяет в тексте документа фрагменты и назначает фрагментам типы
- Штатный анализатор default
  - 23 типа фрагментов (слова, числа, адреса, теги XML и т. п.)
- Другие анализаторы
  - расширения
  - набор функций на языке Си

Словари

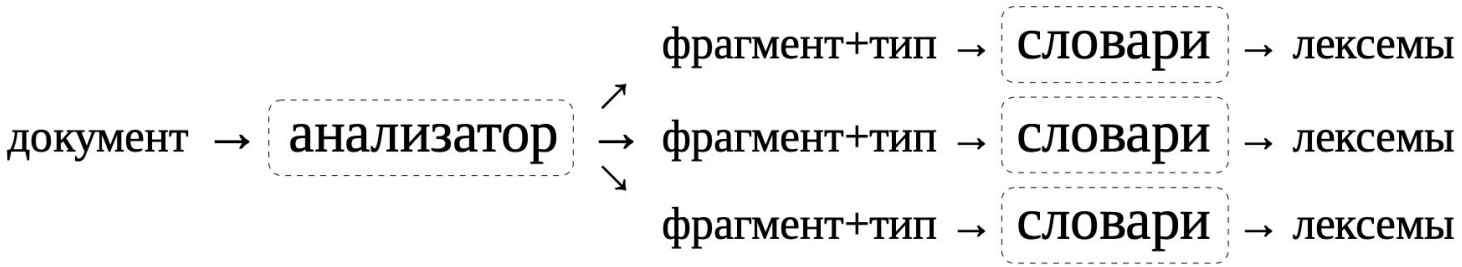


- Словарь превращает фрагмент в лексему
  - убрать стоп-слова
  - привести буквы к одному регистру
  - привести словоформы к общему виду
  - привести синонимы к одному варианту
  - и т. п.
- Штатные словари
  - simple — нижний регистр и стоп-слова
  - стемминг для 21 языка

Шаблоны словарей

- Словарь — параметризованный шаблон
  - набор функций на языке Си
- Штатные шаблоны
  - стеммер snowball
  - словарь ispell
  - синонимы: приведение синонимов к одному виду
  - тезаурус: приведение фраз к одному виду

Конфигурация



- Своя цепочка словарей для каждого типа фрагмента
  - первый словарь, распознавший фрагмент, возвращает лексему
  - фильтрующий словарь передает лексему дальше по цепочке

Индексная поддержка

- GiST
  - неточное представление с обязательной перепроверкой по таблице
  - эффективность уменьшается при увеличении количества слов
  - быстрое обновление (зависит от числа документов)
- GIN
  - точное и компактное представление
  - эффективность не теряется при увеличении количества слов
  - медленное обновление (зависит от числа слов в документах)
- RUM (расширение на основе GIN)
  - дополнительная информация в индексе (позиции лексем и др.)
  - выдача результатов сразу в порядке релевантности

Итоги

- Интегрированный полнотекстовый поиск обеспечивает
  - актуальность результатов
  - транзакционность
  - учет языковых особенностей
  - фразовый поиск
  - ранжирование результатов
  - индексную поддержку
- Поиск охватывает любую информацию, приводимую к текстовому виду
- Возможна тонкая настройка под конкретные нужды

Практика

- Сейчас поиск в книжном магазине работает только по названиям книг.
  Замените его на полнотекстовый поиск по названиям книг, полным именам авторов и аннотациям.
  Проверьте сделанные изменения в приложении. Убедитесь, что поиск не зависит от формы слов в запросе.

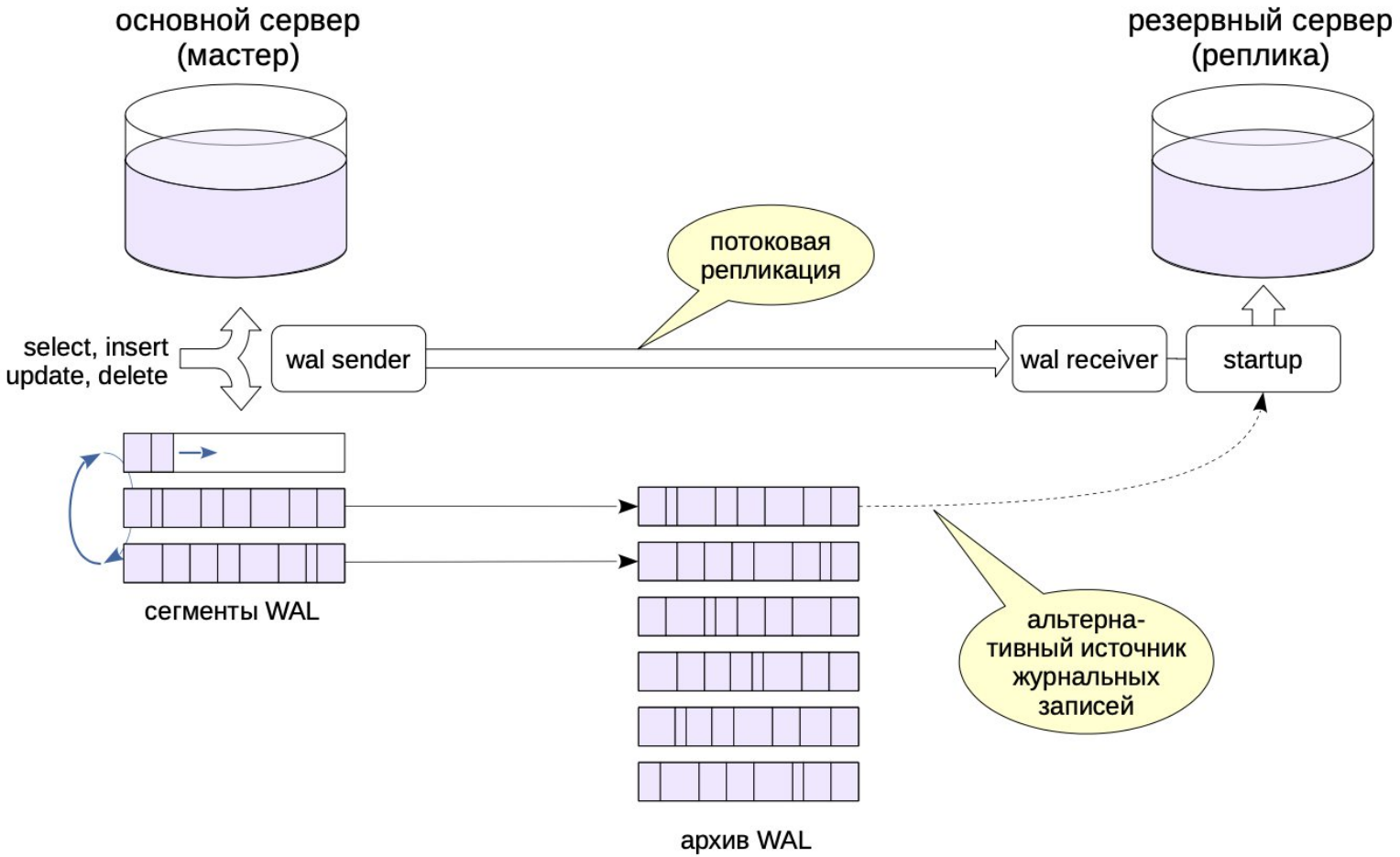
Практика

- Восстановите базу данных с архивом рассылок из резервной копии.
  Добавьте в таблицу mail_messages столбец search_vector и заполните его так же, как в демонстрации.
- Сколько документов содержат одновременно фразы «vacuum full» и «index page»?
  Сравните скорость поиска без индекса и с индексами разных типов (GiST, GIN, RUM).
- Сколько документов содержит слово «vacuuming» именно в такой форме?

Физическая репликация

- Синхронизация копий кластера на нескольких серверах
- Основные задачи
  - высокая доступность, отказоустойчивость
  - масштабируемость
- Механизм
  - один сервер транслирует журнальные записи на другой сервер, и тот проигрывает полученные записи
- Особенности
  - мастер-реплика: поток данных только в одну сторону
  - требуется двоичная совместимость серверов
  - возможна репликация только всего кластера

Физическая репликация



Уровни журнала

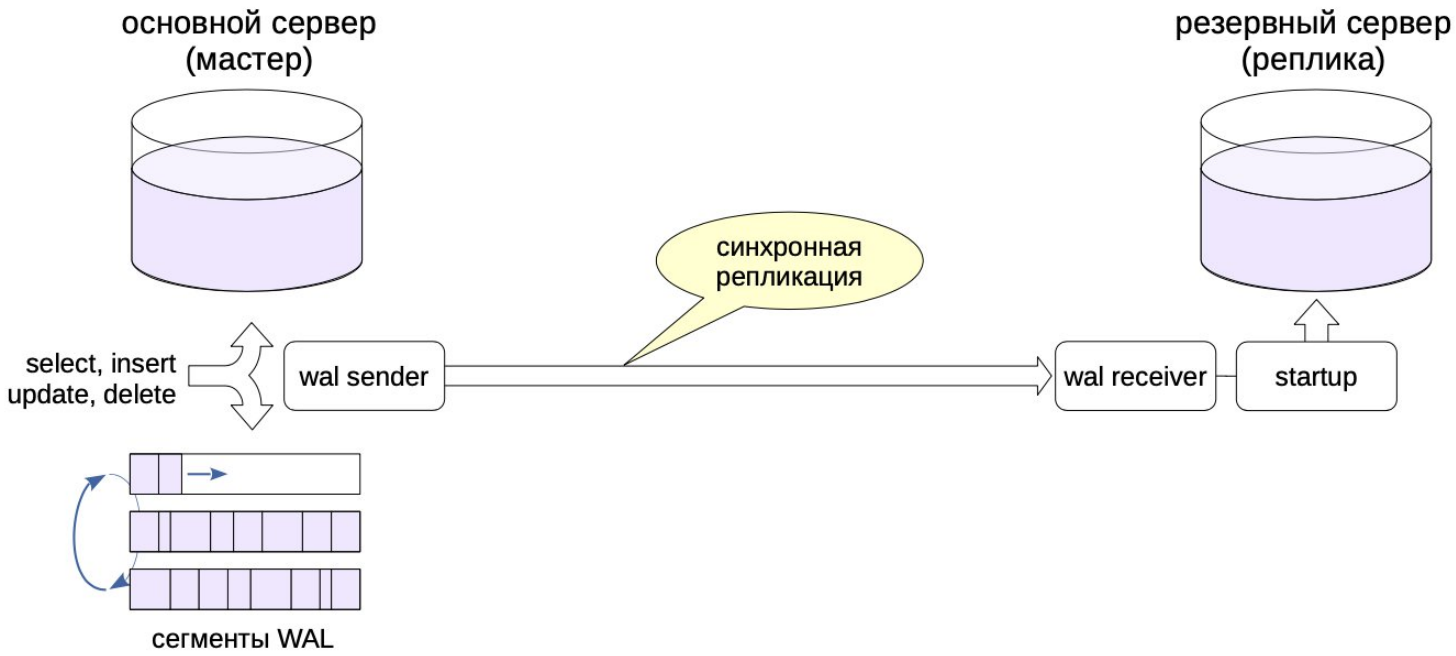
- Параметр wal_level
  - minimal
    - восстановление после сбоя
  - < replica
    - восстановление после сбоя
    - восстановление из резервной копии, репликация

Использование реплики

- Допускаются
  - запросы на чтение данных (SELECT, COPY TO, курсоры)
  - установка параметров сервера (SET, RESET)
  - управление транзакциями (BEGIN, COMMIT, ROLLBACK...)
  - создание резервной копии (pg_basebackup)
- Не допускаются
  - любые изменения (INSERT, UPDATE, DELETE, TRUNCATE, nextval...)
  - блокировки, предполагающие изменение (SELECT FOR UPDATE...)
  - команды DDL (CREATE, DROP...), в т. ч. создание временных таблиц
  - команды сопровождения (VACUUM, ANALYZE, REINDEX...)
  - управление доступом (GRANT, REVOKE...)
  - не срабатывают триггеры и рекомендательные блокировки

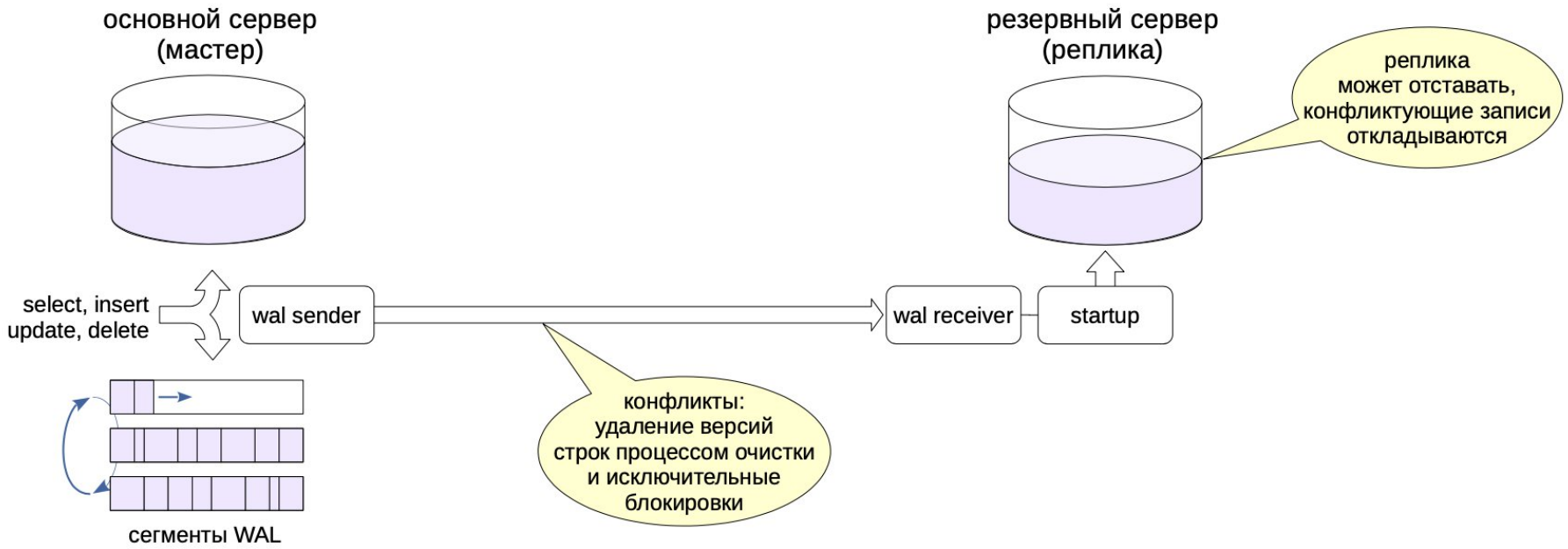
Использование реплики

- надежность хранения данных



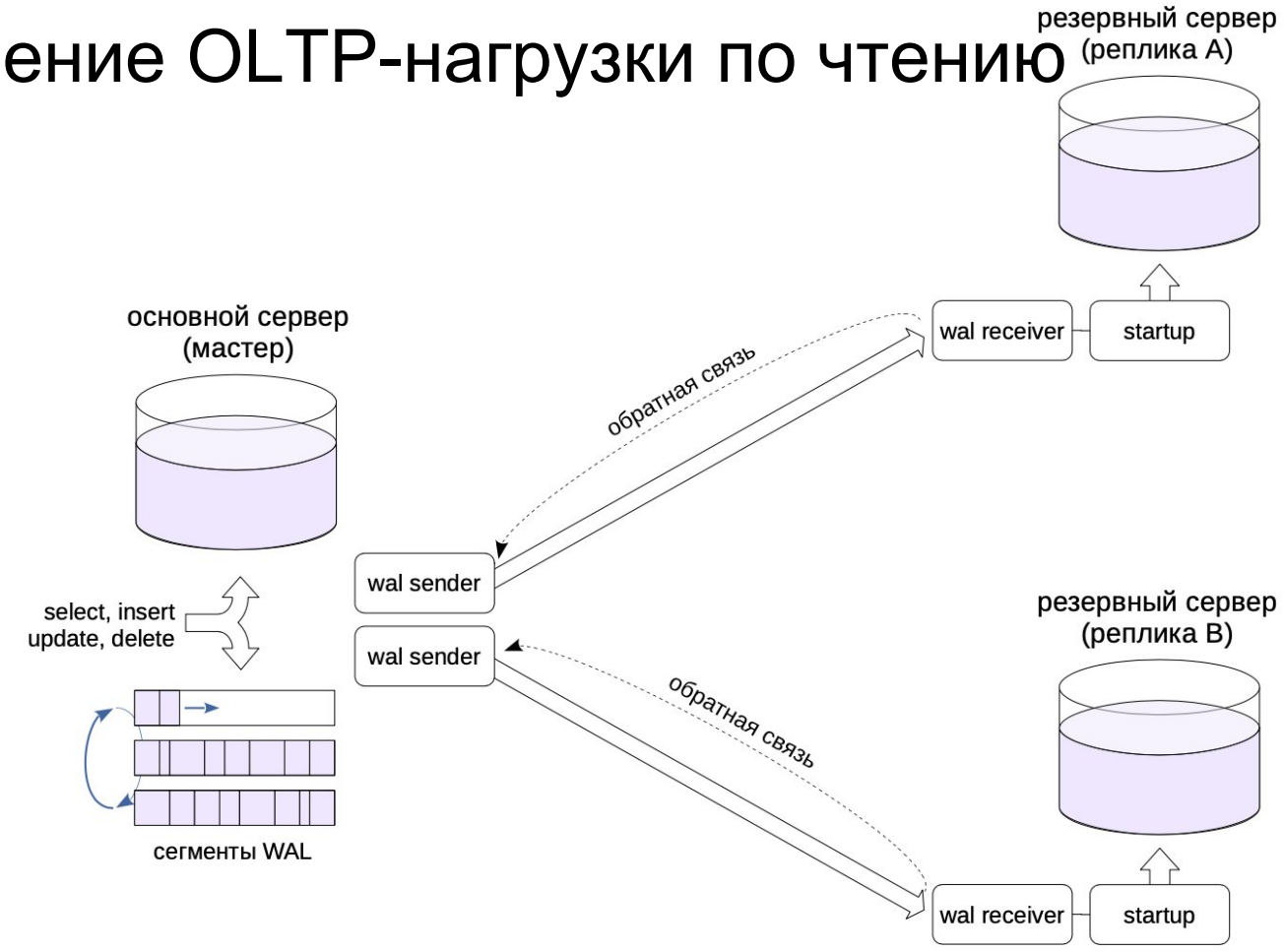
Использование реплики

- выполнение долгих аналитических запросов (отчетов)



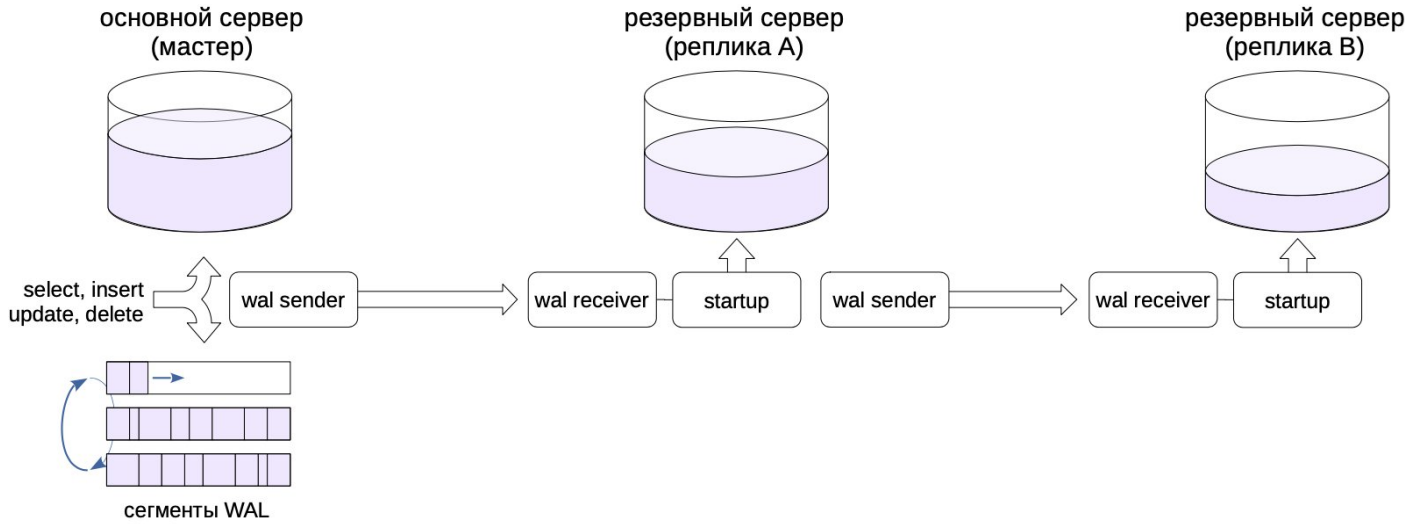
Несколько реплик

- распределение OLTP-нагрузки по чтению



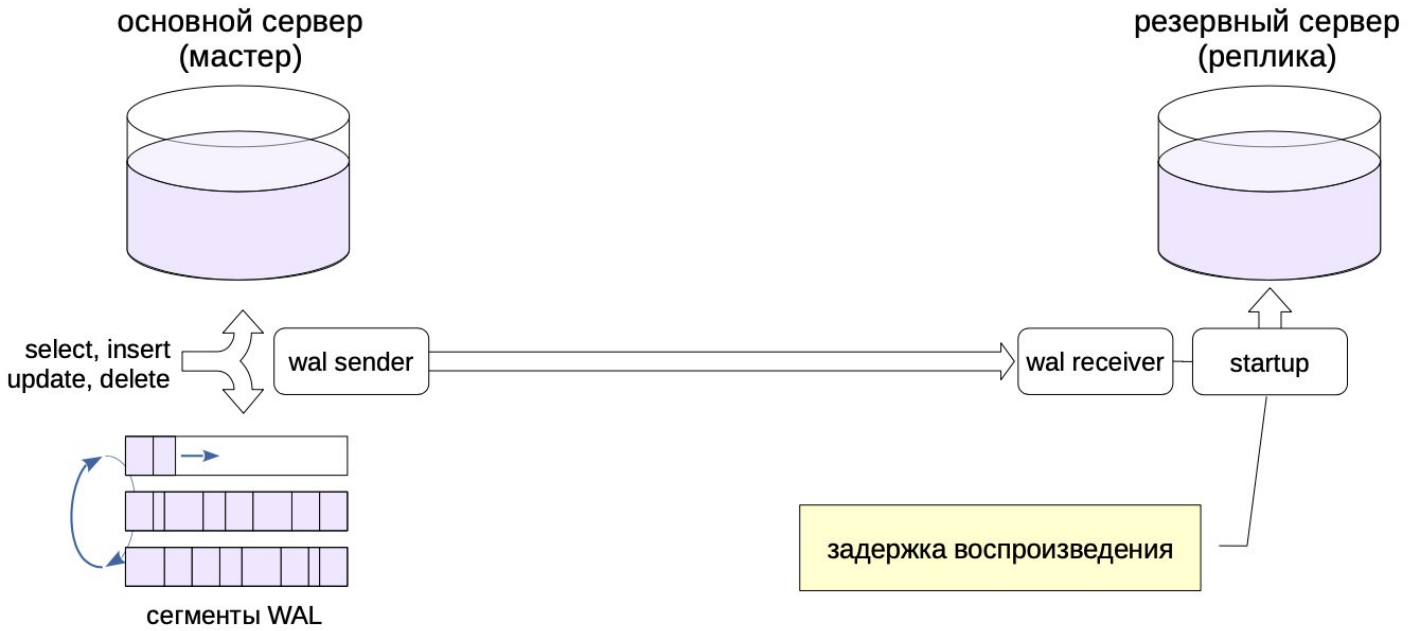
Каскадная репликация

- уменьшение нагрузки на мастер и перераспределение сетевого трафика



Отложенная репликация

- «машина времени»
  и возможность восстановления на определенный момент
  без архива



Переключение на реплику

- Плановое переключение
  - останов основного сервера для технических работ без прерывания обслуживания
  - ручной режим
- Аварийное переключение
  - переход на реплику из-за сбоя основного сервера
  - ручной режим,
    но можно автоматизировать с помощью дополнительного кластерного ПО

Итоги

- Механизм репликации основан на передаче журнальных записей на реплику и их применении
  - трансляция потока записей или файлов WAL
- Физическая репликация создает точную копию всего кластера
  - однонаправленная, требует двоичной совместимости
  - базовый механизм для решения целого ряда задач

Практика

- Разверните реплику.
  Проверьте, как работает приложение, если переключить его на использование реплики.
- Добавьте к механизму фоновых заданий возможность выполнять задания на реплике с помощью расширения dblink.
  Убедитесь, что долгое задание не приведет к появлению такой же долгой транзакции на основном сервере.

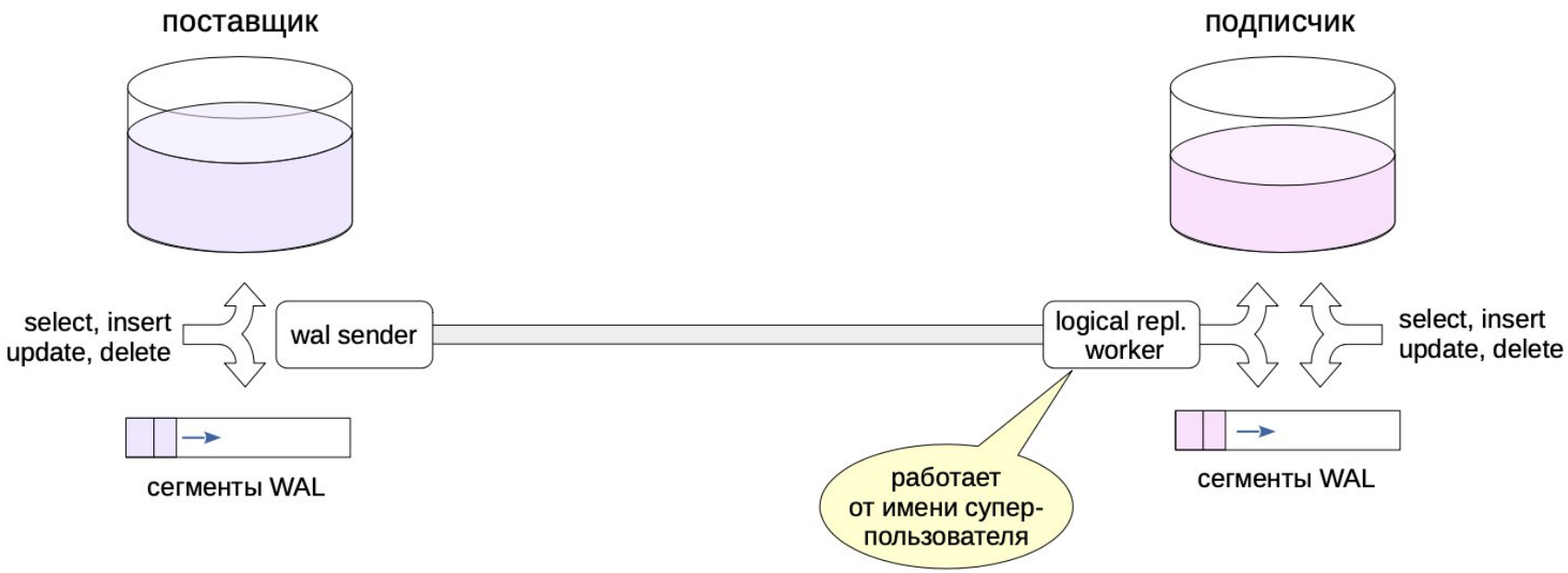
Практика

- Настройте физическую потоковую репликацию между двумя серверами в синхронном режиме. Проверьте работу репликации. Убедитесь, что при остановленной реплике фиксация не завершается.
- По умолчанию применение конфликтующих записей на реплике откладывается максимум на 30 секунд. Отключите откладывание применения и убедитесь, что долгий запрос, выполняющийся на реплике, будет прерван, если необходимые ему версии строк удаляются и очищаются на мастере.
  Включите обратную связь и убедитесь, что теперь запрос не прерывается из-за того, что на мастере откладывается очистка.

Логическая репликация

- Механизм
  - публикующий сервер читает собственные журнальные записи, декодирует их в изменения строк данных и отправляет подписчикам
  - сервер-подписчик применяет изменения к своим таблицам
- Особенности
  - публикация-подписчик: поток данных возможен в обе стороны
  - требуется совместимость на уровне протокола
  - возможна выборочная репликация отдельных таблиц

Логическая репликация



Уровни журнала

- Параметр wal_level
  - minimal
    - восстановление после сбоя
  - replica
    - восстановление после сбоя
    - восстановление из резервной копии, репликация
  - logical
    - восстановление после сбоя
    - восстановление
    - из резервной копии, репликация
    - логическая репликация

Публикации и подписчики

- Публикующий сервер
  - публикация включает несколько таблиц одной базы данных
  - изменения данных выдаются построчно в порядке фиксации транзакций (реплицируются строки, а не команды SQL)
- Подписчики
  - получают и применяют изменения
  - возможна начальная синхронизация данных
  - без разбора, трансформаций и планирования — сразу выполнение
  - возможны конфликты с локальными данными

Конфликты

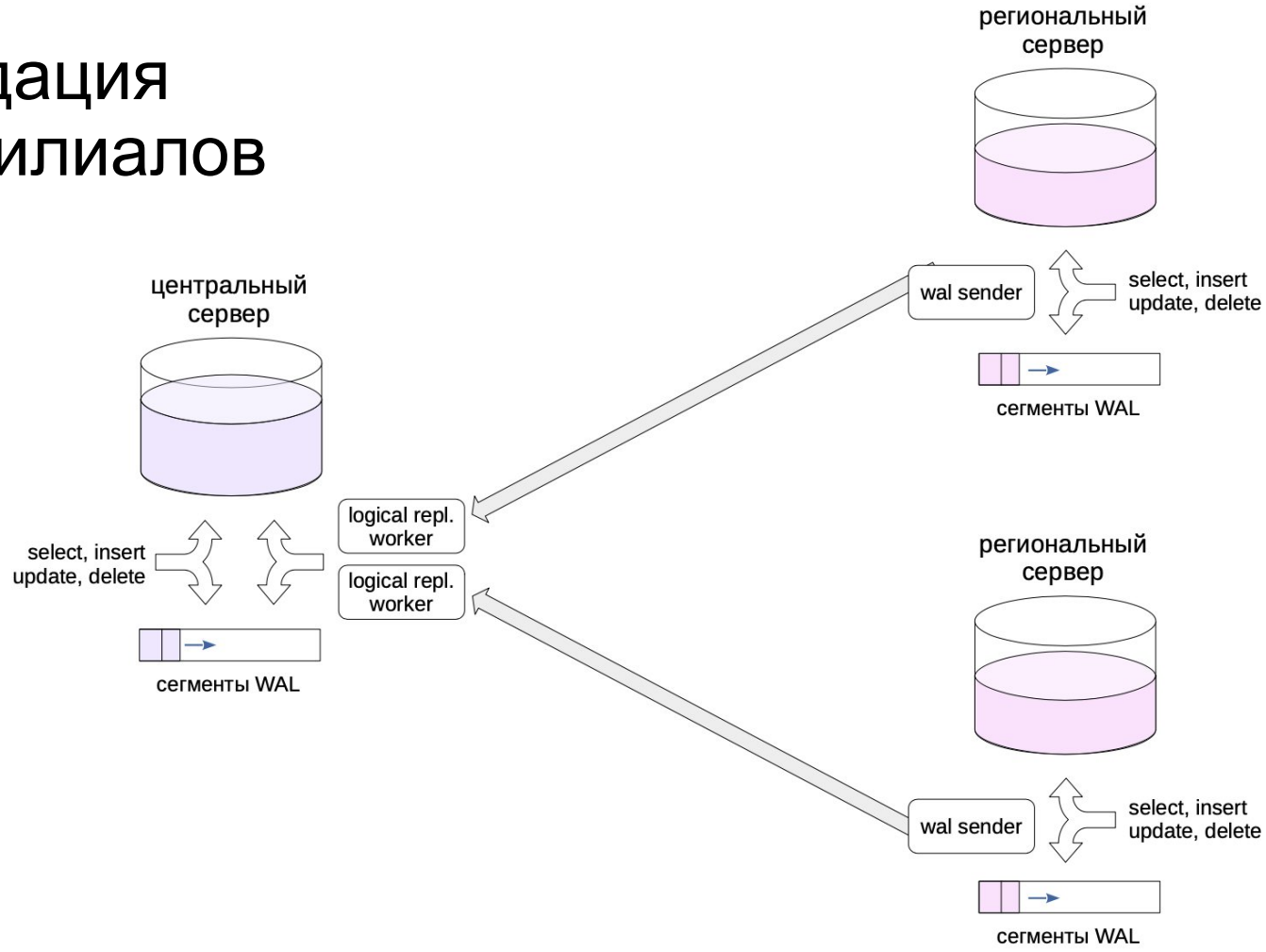
- Режимы идентификации для изменения и удаления
  - столбцы первичного ключа (по умолчанию)
  - столбцы указанного уникального индекса с ограничением NOT NULL
  - все столбцы
  - без идентификации (по умолчанию для системного каталога)
- Конфликты — нарушение ограничений целостности
  - репликация приостанавливается до устранения конфликта вручную
  - либо исправление данных,
    либо пропуск конфликтующей транзакции

Триггеры на подписчике

- При обычной работе
  - в обслуживающих процессах (session_replication_role = origin или local)
  - ENABLE TRIGGER
    ENABLE ALWAYS TRIGGER
- При применении реплицированных изменений
  - в процессе logical replication worker (session_replication_role = replica)
  - ENABLE REPLICA TRIGGER
  - ENABLE ALWAYS TRIGGER
  - только на уровне строк (за исключением начальной синхронизации)

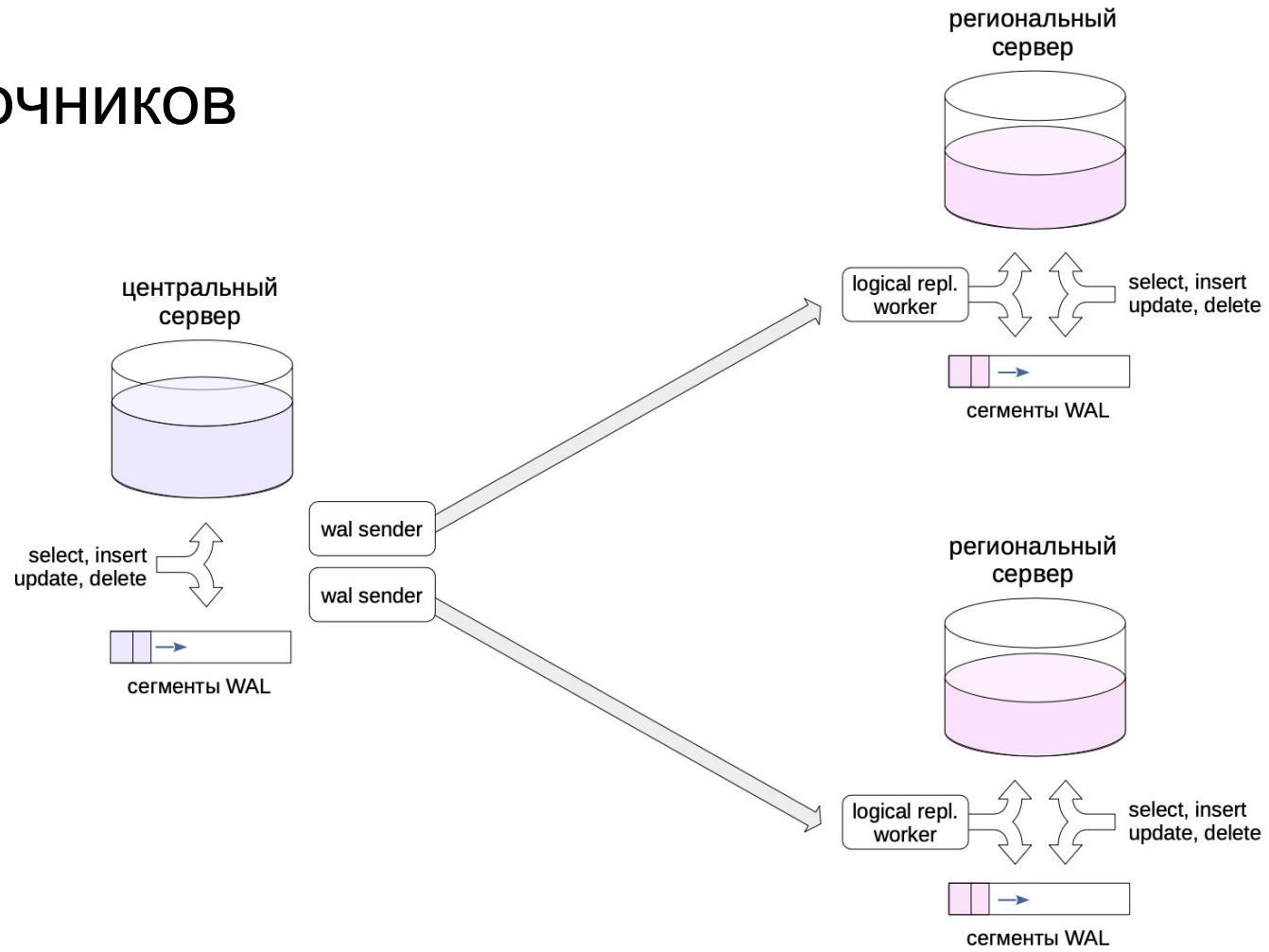
Консолидация

- получение и консолидация данных нескольких филиалов



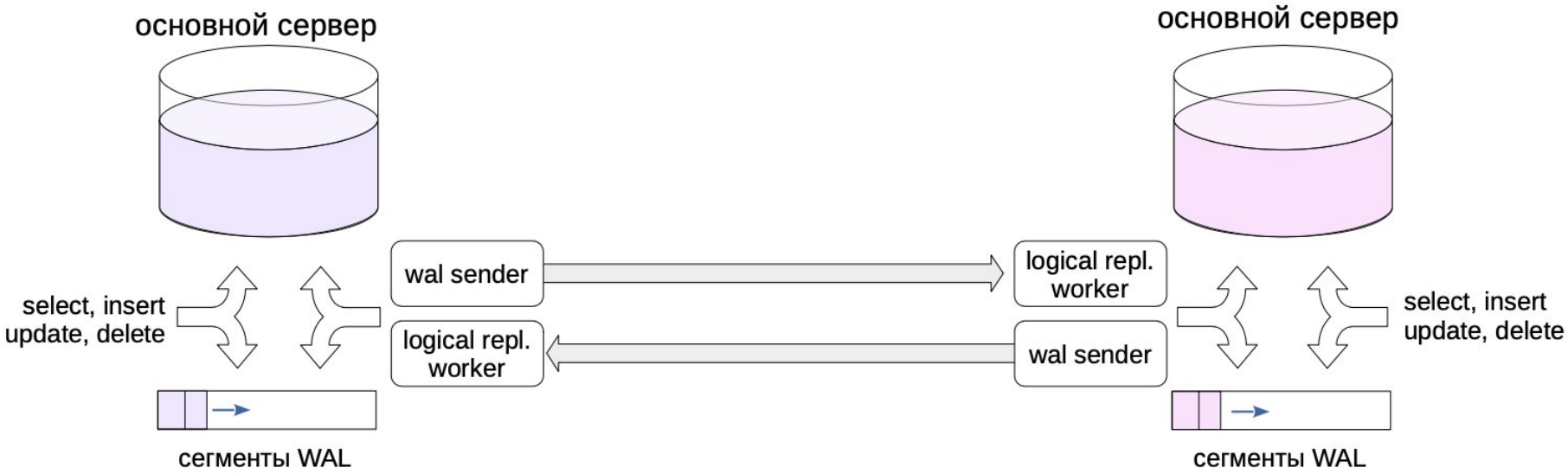
Общие справочники

- репликация справочников



Мастер-мастер

- кластер, в котором данные могут изменять несколько серверов (дело будущего)



Ограничения

- Не реплицируются
  - команды DDL
  - значения последовательностей
  - большие объекты (lo)
  - изменения материализованных представлений, сторонних таблиц, секционированных таблиц
- Могут вызвать проблемы
  - массовые изменения данных
  - изменения, сделанные долгими транзакциями
- Не поддерживаются
  - автоматическое разрешение конфликтов
  - двунаправленная репликация одной и той же таблицы

Итоги

- Логическая репликация передает изменения строк отдельных таблиц
  - разнонаправленная
  - совместимость на уровне протокола
- Следует учитывать имеющиеся ограничения

Практика

- Разверните второй сервер из резервной копии. Очистите таблицу операций.
- Организуйте репликацию справочников книг и авторов из основного магазина во второй. Предполагается, что эти таблицы могут меняться только на основном сервере. Убедитесь, что изменения названий книг и т. п. передаются на второй сервер, но изменения наличного количества происходят на двух серверах независимо.

Практика

- На двух серверах ведутся однотипные таблицы бухгалтерских транзакций. Требуется консолидировать все записи в ту же таблицу на одном из серверов (например, для целей построения общей отчетности). Предложите способ различить «свои» и «чужие» записи и не допустить конфликты при логической репликации. Реализуйте предложенную схему.
- Настройте логическую репликацию произвольной таблицы в другую на том же самом сервере.

Внешние данные

- Представление внешних данных как обычных таблиц
  - стандарт ISO/IEC 9075-9 (SQL/MED)
  - поддерживаются SELECT, INSERT, UPDATE, DELETE
  - триггеры
  - обычное разграничение доступа (GRANT, REVOKE)
- Способ миграции данных из других СУБД
- В будущем — механизм для шардинга
  - вместе с секционированием

Компоненты

1. Обертка сторонних данных
   - каков тип внешнего источника данных?
   - postgres_fdw: внешний сервер PostgreSQL

Компоненты

1. Обертка сторонних данных
   - каков тип внешнего источника данных?
2. Внешний сервер
   - как подключиться к внешнему источнику?
   - postgres_fdw: узел, порт, база данных

Компоненты

1. Обертка сторонних данных
   - каков тип внешнего источника данных?
2. Внешний сервер
   - как подключиться к внешнему источнику?
3. Сопоставление ролей
   - как локальные роли связаны с разграничением доступа, используемым внешним источником?
   - postgres_fdw: и там, и там — обычные роли PostgreSQL

Компоненты

1. Обертка сторонних данных
   - каков тип внешнего источника данных?
2. Внешний сервер
   - как подключиться к внешнему источнику?
3. Сопоставление ролей
   - как локальные роли связаны с ролями внешнего источника?
4. Внешние таблицы
   - какова структура внешних данных?
   - postgres_fdw: данные в обычных таблицах

Соединения и транзакции

- Управление соединениями
  - открывается при первом обращении к внешним данным
  - остается открытым до конца сеанса
- Управление удаленными транзакциями
  - фиксируется или прерывается автоматически, когда фиксируется или прерывается локальная транзакция
  - Repeatable Read для локальных транзакций уровней Read Committed и Repeatable Read
  - Serializable для локальных транзакций Serializable
  - согласованность не гарантируется (нет двухфазной фиксации)

Сравнение с dblink

	postgres_fdw	dblink
доступные команды SQL	SELECT, INSERT, UPDATE, DELETE	любые
совмещение в запросе локальных и внешних данных	да	сложно
управление соединением	автоматическое	ручное
управление транзакциями	автоматическое	ручное
глобальные транзакции	нет	нет
стандартизованный способ	да	нет

Расширение file_fdw

1. Обертка сторонних данных
   - тип внешнего источника — текстовый файл с разделителями
2. Внешний сервер
   - как подключиться — не требуется дополнительной информации
3. Сопоставление ролей
   - не имеет смысла, т. к. во внешнем источнике нет ролей
4. Внешние таблицы
   - структура данных — описание полей файла в виде столбцов таблицы

Другие доступные обертки

- Базы данных
  - ODBC, JDBC
  - различные реляционные и NoSQL СУБД
- Файлы различных форматов
- Геоданные и научные данные
- Веб-ресурсы
  - стандартные протоколы (RSS, IMAP, WWW...)
  - для различных сайтов
- Можно написать собственную обертку
  - на C или Python (с помощью расширения Multicorn)

Итоги

- Обертки сторонних данных — стандартный способ работы с данными любых внешних источников
  - доступ к PostgreSQL и файлам «из коробки»
  - есть реализации оберток для огромного числа систем
- Альтернатива — расширение dblink для выполнения произвольных запросов на удаленном сервере PostgreSQL

Практика

- Поставщик информирует магазин о новинках книжного рынка, предоставляя CSV-файл определенного формата. Загрузите информацию из файла 2020.csv, расположенного в домашнем каталоге пользователя student, в таблицы базы данных. Учтите, что авторы, указанные в файле, могут уже существовать в базе данных, но в файле могут быть допущены опечатки.
- Проверьте в приложении, что новые книги успешно переданы на логическую реплику, и для них работает полнотекстовый поиск.

2020.CSV

CSV-файл содержит следующие поля:

- фамилия автора; - имя автора;
- отчество автора; - название книги;
- формат издания;
- количество страниц;
- ISBN;
- аннотация;
- издательство;
- год выпуска;
- гарнитура;
- номер издания;
- серия;
- имя файла с обложкой (файл находится в каталоге covers).

Практика

- В двух разных базах данных находятся две таблицы с одинаковым набором столбцов. Выведите строки, которые присутствуют в первой таблице, но отсутствуют во второй. Решите задачу с помощью postgres_fdw и с помощью dblink и сравните два способа.
- Как можно проверить, какой уровень изоляции использует обертка postgres_fdw и как она управляет соединениями и транзакциями?