Обработка потоков данных

на основе онтологической модели
и правил логического вывода

Сергей Горшков

serge@trinidata.ru

trinidata.ru

ТриниДата

onto pro

АрхиГраф.MDM

АрхиГраф.СУЗ

Системы, управляемые онтологической моделью

В моделе-ориентированных системах структура данных и логика их обработки определяется онтологической моделью.

Цель создания таких систем – позволить менять алгоритмы работы системы по ходу ее функционирования.

Для этого как можно бóльшая часть логики переносится из программного кода в онтологическую модель.

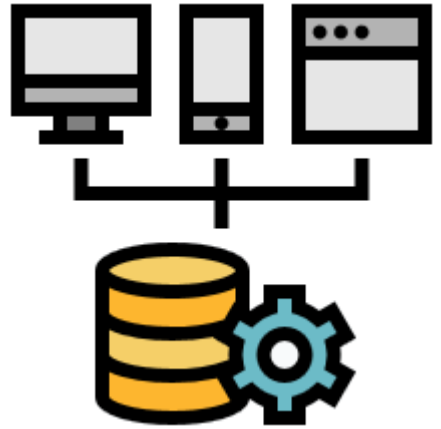
Концептуальная модель предметной области

Онтологическая модель

Интерфейс и алгоритмы работы приложения

В отличие от конфигурационных настроек, модель определяет структуру данных и логику их обработки в терминах концептуальной модели предметной области, а не промежуточной метамодели, связанной с программной структурой самой системы (в терминах «справочников», «документов» и т.д.).

Обработка потоков данных



Задачи обработки потоков данных (документов, сообщений от какого-либо оборудования и др.) часто возникает в корпоративных автоматизированных системах и интегрированных инфраструктурах.

Часто она решается при помощи шин ESB (Enterprise Service Bus), которые позволяют управлять логикой обработки потоковых данных на уровне настроек, а не кода.

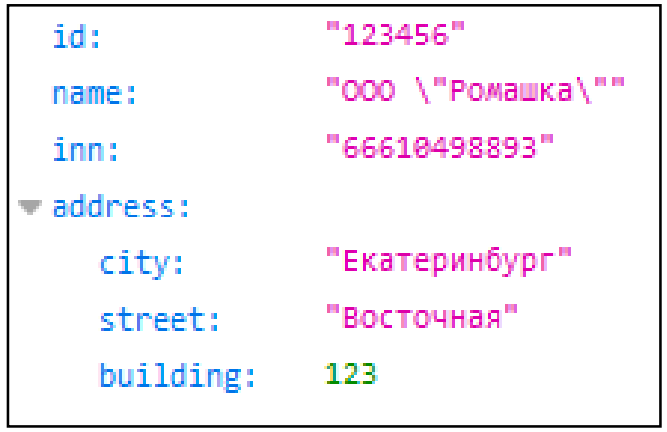
Однако при этом конвейеры обработки на шине жестко связаны с фиксированными форматами обмена, параметрами веб-сервисов интегрируемых систем.

Использование единой онтологической модели позволяет достичь цели **управления структурой и алгоритмами передачи и обработки информации** через единый интерфейс, связанный с единым репозиторием, доступным как пользователям-аналитикам, так и программным компонентам.

Преобразуем структуру данных

Для достижения поставленной цели необходимо использовать единый репозиторий онтологической модели и правил – обычно он уже существует в рамках моделе-управляемых систем и представляет собой точку доступа SPARQL.

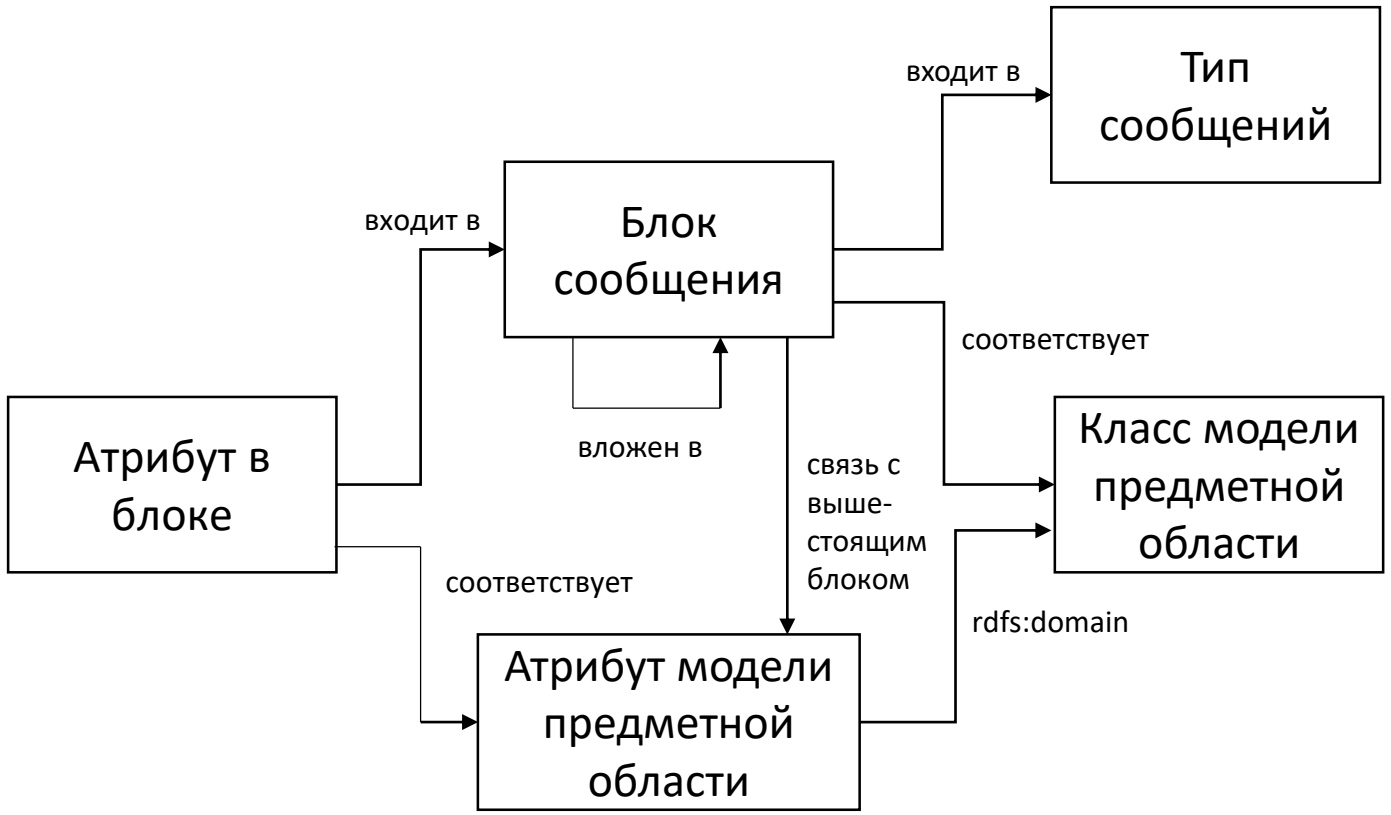
Далее нужно создать в модели описание структуры входящих сообщений, которое сопоставит их элементы с элементами модели предметной области.
В качестве примера рассмотрим соответствие элементов json-сообщения классам и атрибутам онтологии:



Класс	**demo:Клиент**
Атрибут	demo:ИдентификаторВИсточнике
Атрибут	demo:Название
Атрибут	demo:ИНН
Класс	**demo:Адрес**
Атрибут	demo:Город
Атрибут	demo:Улица
Атрибут	demo:НомерДома

Метамодель описания структуры сообщения

Конкретный вид метамодели для описания структуры блоков сообщений и их соответствия элементам модели предметной области зависит от условий задачи. Для примера с обработкой JSON-сообщений основа метамодели может быть такой:



На практике структура сложнее, т.к. элементы сообщений и атрибуты модели могут соответствовать не 1:1 – тогда необходимы преобразования, логику которых также нужно описать в модели.

Процесс трансформации данных

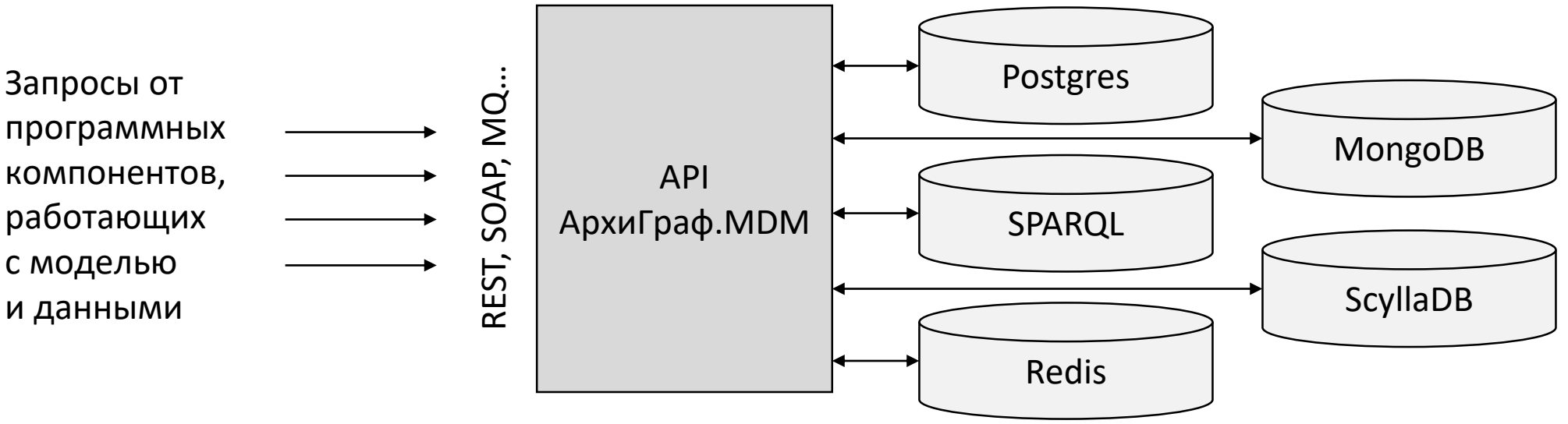
В составе автоматизированной системы реализуется компонент, который считывает метамодель из репозитория и обрабатывает входящие сообщения, поступающие, например, через очередь обмена сообщениями MQ.
Этот компонент преобразует поступившие данные в объекты классов онтологической модели, обладающие значениями атрибутов и связанные между собой.

Если задача состоит только в архивации потока данных, то хранилище результатов обработки может быть любой noSQL или SQL- базой данных.
Но в нашей задаче необходимо выполнить валидацию и смысловое преобразование данных с помощью правил логического вывода, поэтому хранилище представляет собой временный граф.

При изменении модели предметной области или правил соответствия структуры данных они вступают в силу немедленно! Не нужно даже перезапускать обработчики очередей.

Экосистема АрхиГраф

В экосистеме АрхиГраф (программное обеспечение компании ТриниДата) доступ к любым данным осуществляется через API АрхиГраф.MDM.
Этот промежуточный слой обеспечивает работу с классами, атрибутами и индивидуальными объектами онтологических моделей, абстрагируя приложение от деталей их хранения. Физически они могут располагаться в любой СУБД.
В целях обработки потоковых данных временный граф целесообразно разместить в in-memory хранилище, например Redis.



В настройках MDM указано, в какое хранилище помещать объекты каждого класса онтологической модели

Правила логического вывода

Итак, содержимое входящего сообщения помещено во временный граф, который физически расположен в in-memory хранилище.
Теперь к нему нужно применить правила логического вывода, созданные в конструкторе логических правил АрхиГраф.СУЗ.

С помощью правил логического вывода можно:

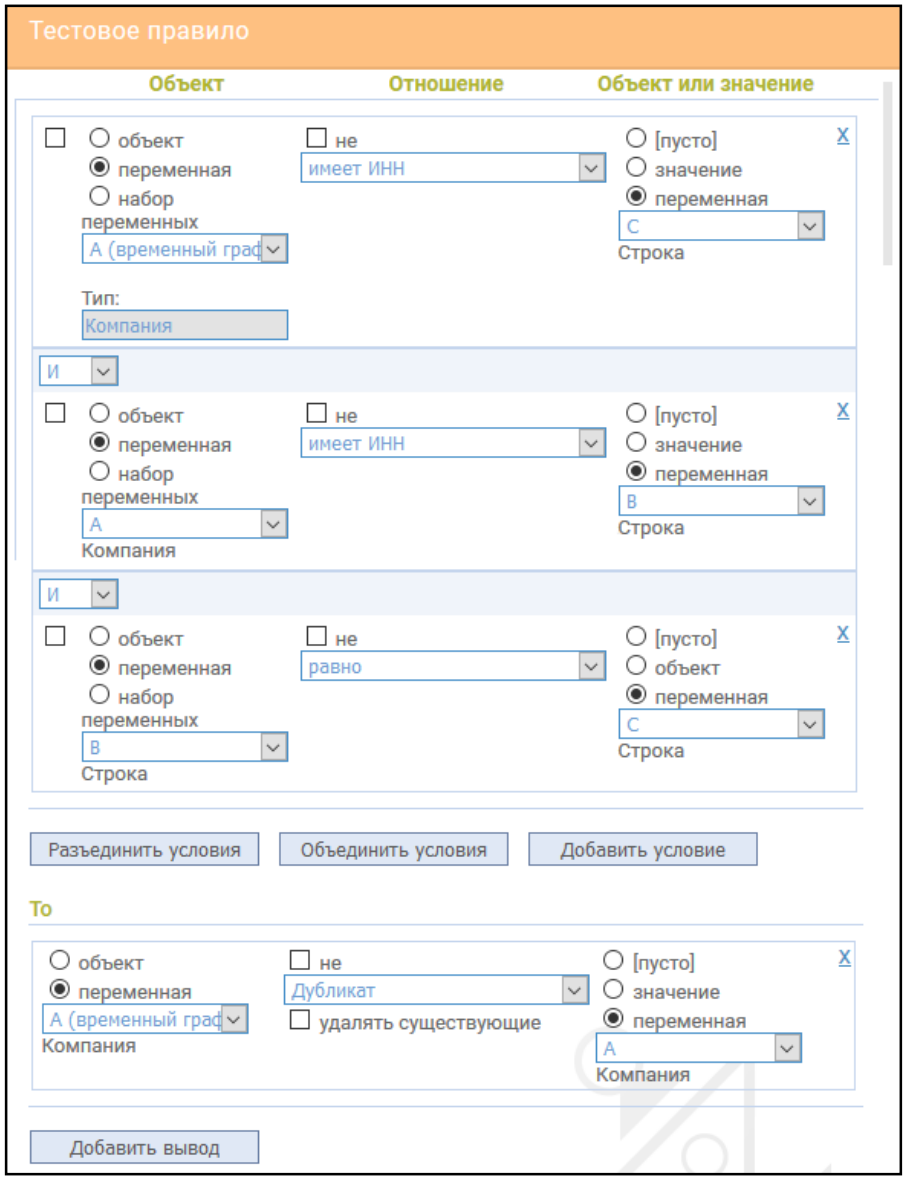
- валидировать свойства и связи объектов;
- искать в постоянных хранилищах копии пришедших объектов (по ключевым полям), создавать или обновлять их;
- создавать произвольные новые объекты в постоянных хранилищах, используя свойства поступивших объектов;
- выполнять арифметические вычисления над свойствами объектов;
- … и многое другое (функционал правил АрхиГраф.СУЗ аналогичен SPIN, но реализует и ряд дополнительных функций).

Правила логического вывода: пример

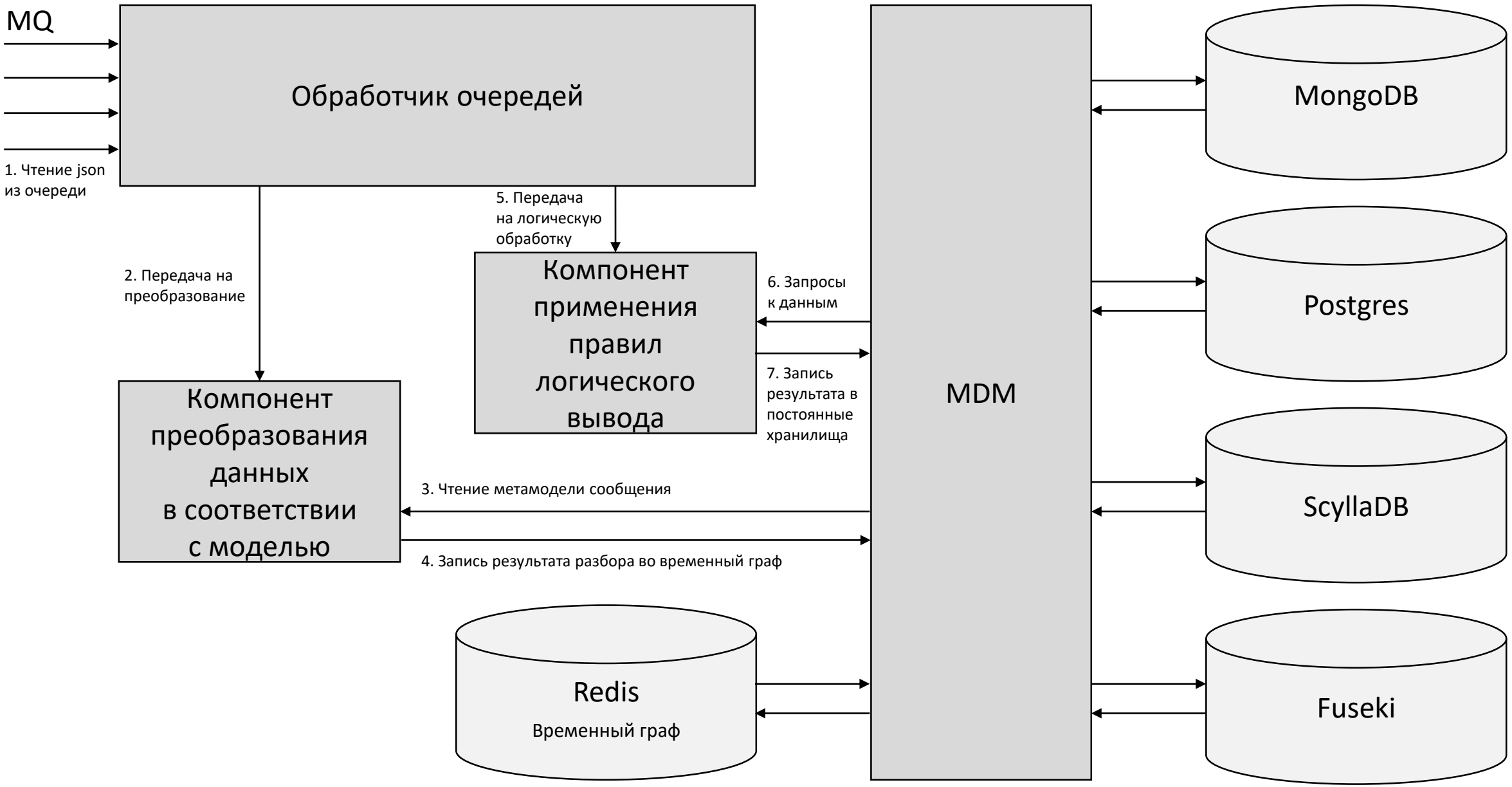
Допустим, в поступившем json-объекте содержится информация о клиенте и сделке с ним.

Правила логического вывода могут:

1) Найти клиента по ИНН в постоянном хранилище;
2) Если он отсутствует – создать его;
3) Проверить статус клиента, чтобы узнать, возможно ли заключение сделки с ним;
4) Найти сделку по ее параметрам;
5) Если отсутствует – создать новую сделку.



Общая архитектура решения

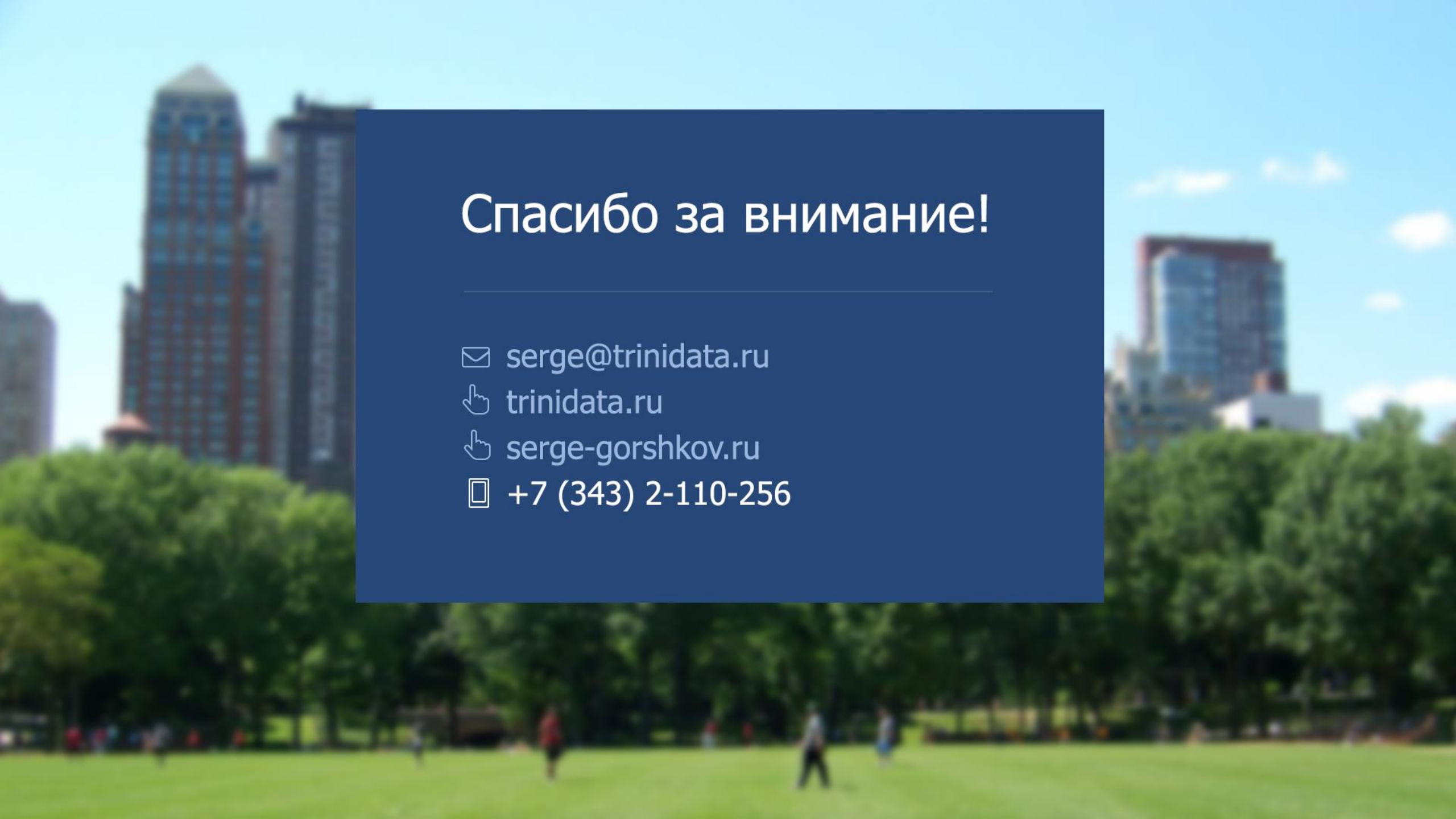


Результаты

Мы построили систему обработки потоковых данных на платформе АрхиГраф, которая:

1) Неограниченно масштабируется за счет подключения новых хранилищ (объекты разных классов можно распределять по различным хранилищам) и шардирования / кластеризации самих хранилищ.
2) Позволяет запускать любое количество обработчиков входящих данных разных типов.
3) Дает возможность менять структуру хранимой информации во время работы системы.
4) Позволяет по мере изменения требований менять места хранения объектов.
5) Дает возможность менять правила разбора входящих сообщений по ходу функционирования системы без ее остановки.
6) Дает возможность редактировать правила логической обработки данных по ходу функционирования системы без ее остановки.
7) Позволяет разграничивать доступ разных программных систем к объектам классов.

Производительность работы системы – десятки сообщений в секунду в одном потоке исполнения, сотни сообщений в секунду при использовании 4-8 потоков.



Спасибо за внимание!

serge@trinidata.ru
trinidata.ru
serge-gorshkov.ru
+7 (343) 2-110-256