Сравнительный анализ способов извлечения данных из xml файлов

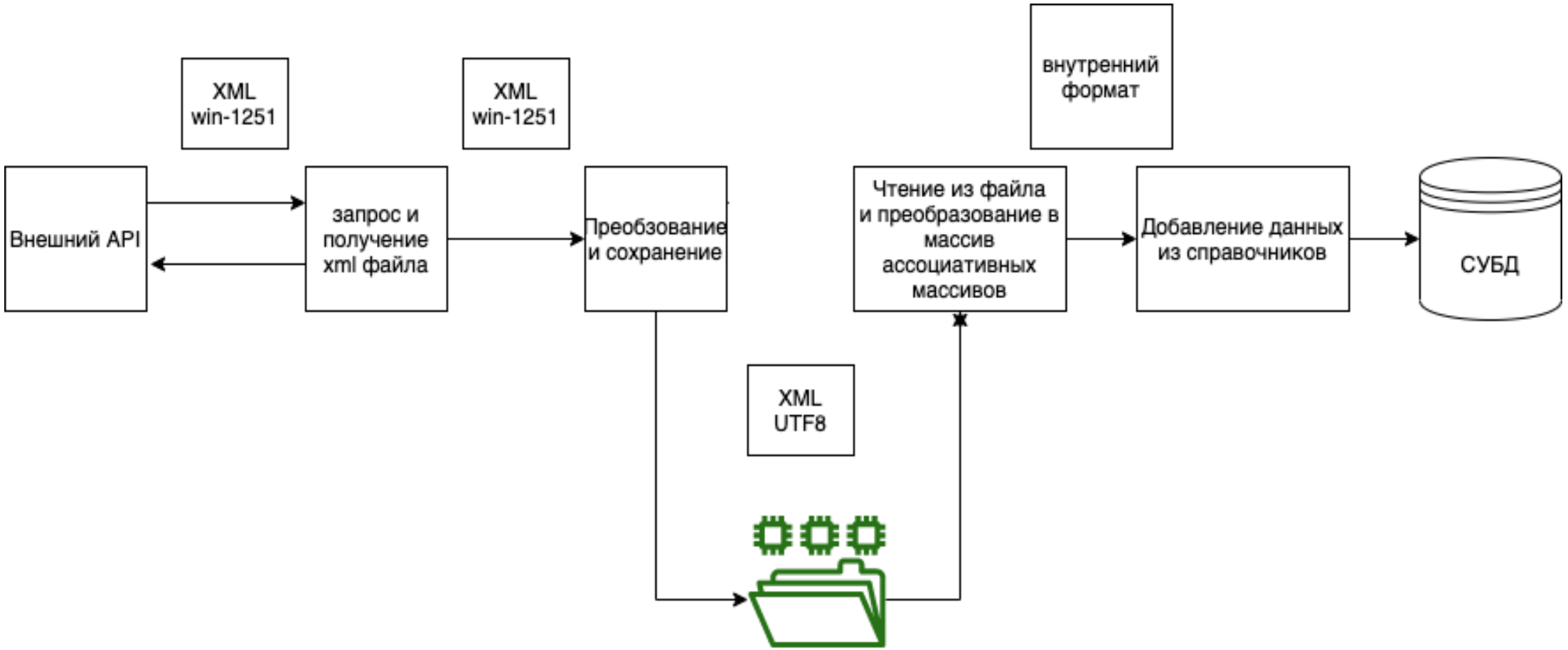
Власов Евгений

Постановка задачи

По API от внешнего сервиса запрашиваются файлы данных в формате xml-документов. Формально в файле содержится массив записей одинаковой структуры. После получения записи обрабатываются и сохраняются во внешней СУБД.

Формат файлов стабилен, но периодически в структуру записей добавляются новые атрибуты. Для сохранения в СУБД необходимо установить соответствие между атрибутами записи и полями в таблице БД,. Поэтому сделать преобразование xml->СУБД без дополнительной обработки невозможно.

Схема обработки файлов



Точки оптимизации

Получение данных от внешнего сервера, добавление данных из справочников и сохранение в СУБД по очевидным причинам оптимизировать невозможно. Поэтому для оптимизации времени выполнения выбраны этапы преобразования и сохранения файла и чтение из файла и преобразование в массив ассоциативных массивов.

Варианты обработки

1) Ruby + XML Parser
2) Ruby XLST преобразование + XML->Ruby Hash
3) Рукотворный xml2json конвертер на C++ +Ruby JSON->Ruby Hash

Ruby + XML Parser

Используется библиотека Nokogiri, которая базируется на libxml2.

Входной xml файл считывается парсером, далее в цикле происходит обход узлов xml-документа, считывание данных каждого из них и преобразование в ruby hash.

Ruby XLST преобразование + XML->Ruby Hash

Используется библиотека Nokogiri, в которая базируется на libxml2. В состав библиотеки входит модуль XSTL, позволяющий выполнять xslt-преобразование xml-документов.

Для преобразования xml документов разработан набор XLST правил, которые переводят xml->json.

После преобзования полученый json-файл считывается ruby json, который на выходе выдает ruby hash.

Рукотворный xml2json конвертер на C++ +Ruby JSON->Ruby Hash

Разработана программа на языке C++ с использованием библиотеки pugixml, которая осуществляет чтение xml-файла и преобразование в JSON формат.

После преобразования полученный json-файл считывается ruby json, который на выходе выдает ruby hash.

Входные данные

Файл данных xml.

Размер: 43 мб

Количество элементов в массиве: 43780

Количество атрибутов в одной xml узле: 63

Измерительный стенд

MacBook Pro (15-inch, 2018)

Процессов 2,6 GHz 6-ядерный процессор Intel Core i7

Память 16 ГБ 2400 MHz DDR4

Таблица результатов

Способ	Чтение(с)	Преобразование(с)	Итого(с)
Ruby XML	6.2351	> 60	> 66
Ruby XLST	9,067	2.5533	11,6203
xml2json	3,575	2.5539	6.1289

Выводы

Изначальный вариант ruby xml работал более 60 секунд, поэтому накладывались циклы обработки и это приводило к перегрузке очереди сообщений для обработчиков.

Принятое решение позволило сделать выбор между двумя вариантами xlst и самописный xml2json преобразователь. Каждый из этих вариантов сокращает время преобразования более чем в 6 и 10 раз соответственно.

Для выбора из конкретного варианта необходимо сделать дополнительно оценку целесообразности использования C++.