КОМПЬЮТЕРНЫЕ НАУКИ 2019. Т. 29. Вып. 3. С. 422–437.

УДК 004.423, 004.657, 004.91

© *Е. А. Белых, Ю. В. Гольчевский*

ПОДХОД К ПРОЕКТИРОВАНИЮ ЯЗЫКА ПОДСТАНОВОК ДЛЯ ГЕНЕРАЦИИ ЭЛЕКТРОННЫХ ДОКУМЕНТОВ, СОДЕРЖАЩИХ СЛОЖНЫЕ ТАБЛИЦЫ

В представленной работе описывается способ проектирования языка подстановок для генерации электронных документов на основе содержимого баз данных и файлов. Проектируемый язык предполагает возможность работы как с одной базой данных, так и с большим числом однотипных баз, имеет модульную структуру, при которой для сложных элементов документа используются шаблоны на отдельных вспомогательных языках. Один из таких вспомогательных языков — язык для генерации таблиц, имеющих сложную структуру с вложенными подтаблицами и расширенными ячейками. Описываемый язык позволяет группировать в удобном для чтения виде большие объемы разнообразных данных. Также предполагается, что язык подстановок и его вспомогательные конструкции не будут привязаны к каким-либо форматам входных и выходных данных, что позволяет использовать любые подходящие форматы путем написания соответствующего модуля для интерпретатора.

Ключевые слова: язык подстановок, языки программирования, генерация документов, шаблоны, электронные таблицы.

DOI: 10.20537/vm190311

§ 1. Введение

Представленная работа посвящена изучению проблемы генерации удобного для чтения и анализа человеком документа на основе данных, получаемых из базы данных или файла. Решением данной проблемы занимаются достаточно давно, однако на сегодняшний день IT-рынком практически так и не предлагается удобного и простого инструмента. Одним из инструментов, который можно было использовать для решения описываемой проблемы, являлся Apache Cocoon — программный каркас для разработки веб-приложений, ориентированный на использование XML, что давало возможность компоновать и публиковать контент в различных форматах, включая XML и PDF, и позднее использовавшийся как фреймворк при создании системы управления контентом Apache Lenya [1, 2]. На его основе предлагались системы автоматической динамической генерации документов резюме в академической и исследовательской среде [3–6], приложения для генерации презентаций [7] и другие. Необходимость генерации сложных документов в том или ином виде встречается при решении совершенно разных, казалось бы, задач. Например, при работе с электронными медицинскими картами [8], при представлении клинических данных пациентов из медицинских баз данных в виде удобных для анализа документов [9], в криминалистической деятельности при извлечении и обработке большого количества документов разных типов и форматов и составления полицейских отчетов [10], при оптимизации процессов, связанных с разработкой и тестированием программного обеспечения [11, 12], при работе с любыми персонализированными документами [13], при автоматизированном формировании отчетов и т. д. Достаточно востребованы операции по интеллектуальному извлечению данных из сложных корпоративных баз данных, чему посвящено множество работ, в которых предлагаются различные модели и подходы. Однако мало просто извлечь данные, из них далее нужно сформировать практически полезные документы.

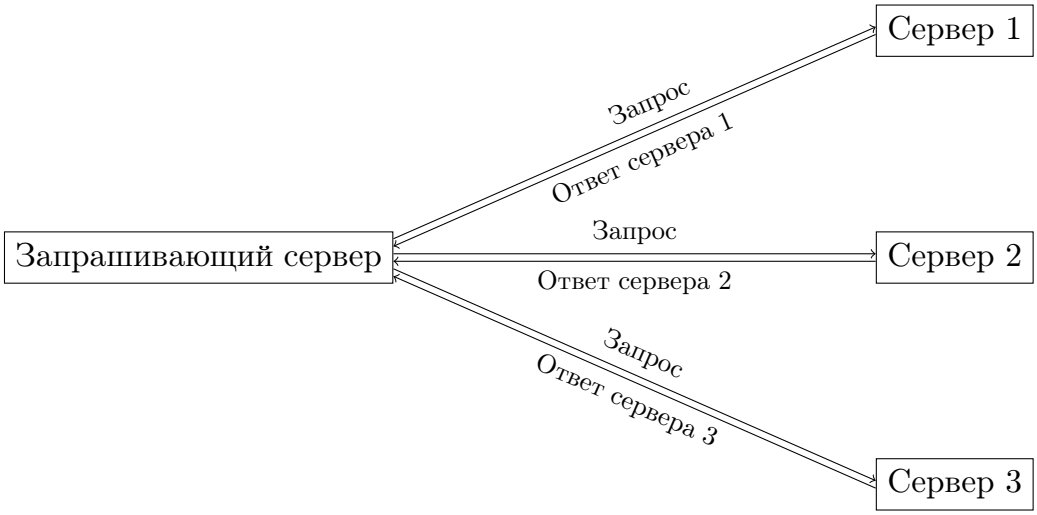

Рис. 1. Схема запросов к нескольким базам данных на разных серверах

Описанная проблема усложняется, если вместо одной базы данных необходимо делать выборки из нескольких однотипных баз. В таком случае для построения документа всем базам данных отправляется одинаковый набор запросов, на который каждая дает свой набор ответов, на основе которых далее строится итоговый документ. Схема такой работы может быть описана так, как показано на рис. 1.

В качестве запросов могут использоваться запросы, созданные с применением языка *SQL*. Это обусловлено тем, что именно SQL на современном этапе используется чаще всего для получения данных из баз, а также тем, что на данный момент существует большое количество инструментов для перевода разнообразных данных в реляционную модель. Как пример можно привести инструмент, описанный в статье, где приведен способ преобразования предназначенных для чтения научных записей в структурированную реляционную базу данных [14].

Помимо запросов на SQL также может использоваться произвольный *shell-сценарий*, запускаемый на серверах и выводящий необходимые данные.

На практике чаще используется SQL. Ответом на запрос будет набор значений, разделенный на столбцы и строки, где столбцы соответствуют выбранным при запросе полям, а строки — найденным записям. В случае отличных от SQL языков или shell-сценариев, можно использовать форматы вывода, совместимые с таким представлением данных.

Генерируемый итоговый документ может состоять из разных элементов: произвольного текста, таблиц, графиков. Его задача — представить полученные данные в максимально удобном для анализа человеком виде. Часто такой документ содержит одну или несколько таблиц, сгенерированных на основе полученных из ответов данных и окруженных произвольным текстом, например, заголовками и разными уточняющими надписями. Упрощенный фрагмент такого документа приведен на рис. 2. В дальнейшем он будет использоваться для пояснения действий.

Данные, полученные в результате применения описанной выборки, могут иметь достаточно сложную структуру. Поэтому и содержащие их таблицы могут иметь сложную структуру с ячейками, расширенными на соседние, и вложенными друг в друга подтаблицами. При этом количество строк и столбцов заранее неизвестно.

Как правило, структура генерируемого документа изменяется не так часто, а данные, которые он представляет — постоянно. Это обуславливает то, что структуру документа удобно описывать некоторым *шаблоном* независимо от данных. Он должен позволять де-

Васильеву Василию Васильевичу
от Иванова Ивана Ивановича

Отчёт о количестве документов, пришедших из других организаций

Ниже приведено количество и тип документов, пришедщих на каждый сервер в период с 01.01.2018 по 31.01.2018:

Сервер 1		Сервер 2		Сервер 3	
Тип документа	Количество	Тип документа	Количество	Тип документа	Количество
Документ 12	27	Документ 5	45	Документ 13	37
Документ 7	14	Документ 18	67	Документ 9	29
Документ 6	31			Документ 11	31
Документ 14	19				

Рис. 2. Пример фрагмента генерируемого документа

лать текстовые вставки на основе данных, а также генерировать описанные выше сложные таблицы [15].

Цель данной работы — предложить подход к созданию универсального инструмента, позволяющего решать описанные выше проблемы и генерировать документы с таблицами независимо от их сложности, используя запросы к одной базе данных, множеству однотипных баз, а также получая данные из файлов. При этом важным условием является отсутствие привязки к определенным форматам входных и выходных данных. Вместо этого инструмент должен иметь возможность добавления *модулей* для работы с разными форматами.

§ 2. Возможные решения

Обычно в общем виде поставленную задачу не решают, а ограничиваются частными случаями. Опишем распространенные способы решения.

Часто для решения задачи используются офисные пакеты, такие как *Microsoft Office* и *OpenOffice*. Такой подход удобен, когда генерируемый документ имеет простую структуру, а запрашиваемых данных немного, и они находятся в одной базе данных.

Однако создание более сложных документов может вызвать затруднения из-за ограниченных возможностей скриптовых языков этих пакетов.

Помимо этого, формат итогового документа будет ограничен форматами используемого офисного пакета, что не всегда удобно с точки зрения практического применения.

Другой способ решения — это использование форматов *XML* и *XSLT*. С их помощью можно создавать более сложные шаблоны, а описывающий эти шаблоны код будет оставаться достаточно простым.

Основная проблема такого подхода в том, что формат XML предназначен для представления сложных иерархических данных, в то время, как генерируемый документ имеет

более простую структуру. Поэтому в качестве основного языка для шаблона документа формат XML является избыточным.

Проблема заключается и в том, что формат XSLT просто описывает схему трансляции XML-файла одной структуры в XML-файл другой структуры, что накладывает ряд ограничений на создаваемый документ. Это особенно сильно проявляется, если он должен иметь сложные таблицы.

Как и в первом случае, при таком подходе накладываются ограничения на формат итогового документа — совместимый с XML формат.

Пример такого подхода описан в работе [16], где XML вместе с XSLT используется для генерации электронных медицинских карт, которые в отличие от поставленной в данной работе задачи, имеют более простую структуру.

Еще один пример приведен в работе [13], в которой на основе *doc-шаблона*, создаваемого в *Microsoft Word*, и XSL-преобразования генерируется необходимый документ. Работа интересна тем, что в ней комбинируются XSL-преобразования с офисными пакетами. Однако из-за этого выходной формат ограничивается документами в word-совместимом формате.

Сложность языка XML обсуждается и в работе [17]. Для решения этой проблемы был представлен язык *BonXai*, упрощающий многие конструкции языка XML. Важной особенностью этого языка является возможность свободных преобразований его в XML и обратно. Предположительно, такой подход может помочь решить проблему, но, как уже упоминалось, если использовать XML-подобные форматы, то и форматы входных-выходных данных должны быть XML-совместимыми, что является ограничением, которое можно избежать.

Третий способ решения подразумевает использование языков программирования общего назначения например *AWK*, *PHP* или *C*. В этом случае каждый шаблон будет представлять собой полноценную программу. Главные плюсы такого подхода — это возможность генерировать документы любой сложности и отсутствие ограничений в выборе формата итогового документа.

Предлагаются и другие решения. Например, в работе [18] была предложена математическая модель автоматического формирования документа путем интеграции Visual Basic for Application, ActiveX Data Object и Extensible Markup Language (XML). Она построена на использовании классификации основных типов данных документа, создании правил хранения файлов конфигурации, настройки шаблонов документов и применении алгоритма автоматической генерации документов с использованием некоторых критериев отбора.

Однако такие способы сложно назвать удобным. Даже если разработать специализированный программный интерфейс на используемом языке, код шаблона будет содержать достаточно много команд, не имеющих прямого отношения к формированию документа или процедура будет перегружена достаточно большой предварительной работой по созданию нужных конфигураций и правил. Все это будет несколько затруднять доработку существующих и создание новых шаблонов.

Первые два способа из описанных выше могут использоваться только для частных случаев задачи. Третий способ позволяет решать задачу в общем виде, но при этом требует избыточных усилий.

§ 3. Макроязык подстановок

Альтернативным решением может быть специальный язык подстановок для генерации документов на основе шаблона, что позволит избавиться от недостатков, отмеченных выше.

Предлагаемый язык подстановок может состоять из разных *конструкций*, а также подстановок этих конструкций. Конструкции задаются с помощью символа «#» и следующего за ним слова, обозначающего определяемую конструкцию. Далее перечисляются параметры, а затем располагается текст определяемой конструкции.

Текст конструкции может быть определен тремя способами: одной строкой, несколькими строками или как содержимое файла. В первом случае вся конструкция пишется одной строкой и текст определения находится сразу после параметров:

```
#entname arg1 ... argn text
```

где `#entname` — определяющее слово конструкции, `arg1`, `...`, `argn` — ее параметры, а `text` — текст конструкции.

Во втором случае первая строка содержит определяющее слово и параметры конструкции, а последующие строки — ее текст. Строка `#end` обозначает конец текста. Формально это можно записать так:

```
#entname arg1 arg2 ... argn
    string1
    string2
    ...
    stringn
#end
```

где `#entname` — определяющее слово конструкции, `arg1`, `...`, `argn` — ее параметры, а `string1`, `...`, `stringn` — строки, составляющие текст конструкции.

В третьем случае конструкция определяется почти так же, как и в первом, только вместо текста конструкции следует имя файла, содержащего этот текст:

```
#entname args : filename
```

где `#entname` — определяющее слово конструкции, `args` — ее параметры, а `filename` — имя файла, содержащего текст конструкции.

§ 4. Определения и источники данных. Подстановка определений

Определения — это статически определяемые конструкции, которые в дальнейшем вставляются в текст с помощью подстановки. Они обозначаются словом `def` и имеют один параметр — свое имя.

Источники данных похожи на определения, однако вставляемые при подстановке фрагменты документа получаются не из текста конструкции, а из внешних источников: SQL-запросов, *CSV*-файлов и т. д. Источники данных могут обозначаться словом `input`, и содержать два параметра: первый обозначает имя источника данных, а второй — его тип.

Внутрь определений могут быть включены подстановки других определений и источников данных. При формировании итогового документа выполняется рекурсивная подстановка всех определений, начиная с корневого, которое имеет специальное имя — *document*.

Подстановка определений может выполняться с помощью символа «*» и следующего за ним имени определения, например `*defname`, где `defname` — имя определения. Иногда сразу после имени определения идет текст, не отделенный от него пробелами. В этом случае имя определения при подстановке можно заключить между скобками «{» и «}»:

```
*{defname}
```

Например, ниже приведены фрагменты шаблона, генерирующие заголовок документа, изображенного на рис. 2:

```
#def namefrom_rp Иванов Иван Иванович
#def nameto_dp Васильеву Василию Васильевичу
#def datefrom 01.01.2018
#def dateto 31.12.2018

#def document
    ...
    *nameto_dp<br> от *namefrom_rp
    ...
    Отчет о количестве документов,
    пришедших из других организаций
    ...
    Ниже приведено количество и~тип документов,
    пришедших на каждый сервер в период с
    *datefrom по *dateto:
    ...
#end
```

В представленном выше шаблоне используется четыре определения: `namefrom_rp` и `nameto_dp`, которые при подстановке дают имя отправителя и имя получателя, а также `datefrom` и `dateto`, дающие даты начала и конца периода отчета.

Подстановки можно выполнять внутри любых конструкций, в том числе и источников данных. Ниже приведен SQL-запрос, используемый для генерации таблицы из примера на рис. 2:

```
#input request sql
    select
        a.name, count(\*)
    from
        doc d
        join agent a on a.id = d.agent_id
    where
        d.date between *datefrom and *dateto
    group by
        a.name, d.doctype
#endinput
```

В этом запросе используются подстановки определений *datefrom и *dateto для того, чтобы задать границы для даты выбираемых документов. Также можно заметить, что символ «*» внутри самого запроса экранируется символом «\». Это необходимо, чтобы при генерации документа отличать его от начала подстановки.

§ 5. Генерация таблиц и подстановка данных из источников

В отличие от подстановки определений, подстановку данных из источников нельзя выполнять в любом участке шаблона. Она выполняется внутри *шаблонов таблиц*, написанных на специальном языке для генерации таблиц.

Шаблон таблицы определяется в основном шаблоне подобно другим конструкциям. Он определяется словом `table` и имеет один аргумент — свое имя. Подстановка шаблона таблицы выполняется таким же образом, как и подстановка определения, только вместо текста вставляется таблица.

В отличие от других способов описания таблиц, например таких, как в [19], в этом языке содержимое каждой ячейки таблицы явно не указывается. Вместо этого итоговая таблица представляется в виде *склейки* из более простых таблиц.

Таблицы, из которых выполняется склейка, могут быть результатом подстановок из источников данных, результатом вызова встроенных функций, либо являться результатом склейки.

Таблицу, полученную в результате подстановки данных из источника или вызова встроенной функции, а не в результате склейки в дальнейшем будем называть *атомарной таблицей*.

Далее в качестве примера приводится процесс построения шаблона таблицы представленной на рис. 2, а для получения данных будем использовать SQL-запрос `request`, приведенный выше, ответом на который является список всех типов документов, хранящихся в базе и их количество за отчетный период.

Подстановка данных из источника похожа на подстановку определения, только после имени источника данных указывается список аргументов. Этот список представляет собой набор значений аргументов, ограниченный символами «(» и «)». Формально это можно записать как:

```
*inputname(arg1, ... argn);
```

где `inputname` — имя источника данных, а `arg1`, `...`, `argn` — аргументы подстановки.

Благодаря такой форме записи подстановки похожи на вызовы функций в Си. Это позволяет сделать их однородными со встроенными вызовами и тем самым избежать добавления лишних конструкций в язык.

Количество и значение аргументов при подстановке зависит от типа источника данных. В случае SQL-запроса, первый аргумент — это идентификатор сервера, принимающего запрос, второй аргумент — номер столбца из полученного ответа, а третий — опционален и является выражением, в соответствии с которым выбираются строки ответа. В случае CSV-файла первый аргумент отсутствует.

Идентификатор сервера — это уникальное имя сервера в `списке серверов`, специальном источнике данных, являющимся CSV-файлом и содержащим данные серверов, к которым отправляются запросы.

Второй аргумент может указывать номер только одного столбца. Таким образом, результатом подстановки всегда будет таблица с единственным столбцом. Как будет описано дальше, благодаря склейке, необходимости выбора сразу нескольких столбцов нет.

Выражение, являющееся третьим параметром, состоит из логических «и», логических «или», операторов сравнения, чисел и произвольных экранированных строк. Числа обозначают номера столбцов. Операторы сравнения, за исключением проверки равенства, обозначаются таким же образом, как в языке Си. Проверка равенства и логические операторы обозначаются одиночными символами вместо двойных.

Например, на рис. 3 приведен результат подстановки

```
*request("server 1", 1, [1 != "Системный"]);
```

выводящей все типы документов, пришедших на первый сервер за отчетный период и при этом исключающий документы, сгенерированные системой.

Также язык для генерации таблиц имеет встроенные команды, чей вызов выполняется аналогично подстановке данных из источника. Примерами таких команд могут быть `print` и `servers`.

Команда `print` возвращает таблицу с одной ячейкой, содержащей строку, переданной в виде аргумента. На рис. 4 приведен результат вызова

Документ 12
Документ 7
Документ 6
Документ 14

Рис. 3. Результат подстановки `request`

Тип документа

Рис. 4. Результат вызова `print`

```
*print("Тип документа");
```

Эта команда вместе со склейкой, которая будет описана ниже, используется для генерации заголовков таблиц.

Команда `servers` дает доступ к списку серверов. Ее вызов аналогичен подстановке данных из CSV-файла. На рис. 5 в качестве примера приведен результат вызова

```
*sources(4, [1 != "1"]);
```

выводящего все сервера, кроме первого. Эта команда чаще всего используется при многократных вызовах блока, которые будут описаны ниже.

§ 6. Склейка таблиц и блоки вызовов

Склейка является одной из основных особенностей предлагаемого языка. Она не имеет специальной синтаксической конструкции, описывающей ее. Для выполнения склейки достаточно, чтобы команды следовали друг за другом и было указано направление и тип склейки для всех команд, за исключением первой.

Направление склейки указывается с помощью слова, обозначающего направление: «left», «right», «up» или «down». Далее при необходимости пишется символ «,» и слово, указывающее тип склейки. Типа склейки всего два: с растяжением и без. Склейка без растяжения используется по умолчанию. Склейка с растяжением обозначается словом «span».

Добавим к таблице на рис. 3 заголовок, как показано ниже:

```
#table doccount
*print("Тип документа");
*request("server1", 1, [1 != "Системный"]) : down;
#end
```

Результат приведен на рис. 6.

Как уже упоминалось, таблицы, составляющие основную таблицу, также могут быть результатом склейки. Для генерации таких «вложенных» таблиц используются *блоки*.

Сервер 2
Сервер 3

Рис. 5. Результат вызова `sources`

Тип документа
Документ 12
Документ 7
Документ 6
Документ 14

Рис. 6. Результат склейки строк

Тип документа	Количество
Документ 12	27
Документ 7	14
Документ 6	31
Документ 14	19

Рис. 7. Результаты склейки столбцов

Начало блока обозначается символом «{», а конец — символом «}». Внутри блока таблица строится «с нуля», независимо от вызовов за его пределами. Сам блок считается отдельным вызовом. Поэтому, если перед ним есть другие вызовы, для него должно быть указано направление склейки.

Усложним таблицу, приведенную на рис. 6, добавив к ней столбец с количеством пришедших документов, как показано ниже:

```
#table doccount
{
    *print("Тип документа");
    *request("server1", 1, [1 != "Системный"]) : down;
}
{
    *print("Количество");
    *request("server1", 2, [1 != "Системный"]) : down;
} : right
#end
```

Результат приведен на рис. 7.

При склейке с растяжением, если количество строк или столбцов в склеиваемых таблицах не совпадает, то вместо добавления новых ячеек растягиваются существующие. Это очень удобно при генерировании заголовков таблиц.

Сыктывкар	
Тип документа	Количество
Документ 12	27
Документ 7	14
Документ 6	31
Документ 14	19

Рис. 8. Результаты склейки с растяжением

Важно отметить, что таблицы, сгенерированные с использованием склейки с растяжением накладывают ограничения на формат итогового документа: этот формат должен иметь возможность хранения таблиц с расширенными на соседние ячейками, так как растяжение выполняется именно за счет расширения ячеек.

Добавим заголовок с названием сервера к таблице на рис. 7, как показано ниже:

```
#table doccount
*print("Server 1");
{
        {
            *print("Тип документа");
            *request("server1", 1, [1 != "Системный"]) : down;
        }
        {
            *print("Количество");
            *request("server1", 2, [1 != "Системный"]) : down;
        } : right
} : down, span
#end
```

Результат приведен на рис. 8.

§ 7. Многократные вызовы блока команд

Иногда необходимо склеить несколько блоков, отличающихся параметром, получаемым при подстановке каких-либо данных. Для этого можно использовать *многократный вызов блока*. В этом случае блок вызывается для каждого результата заданной подстановки. При очередном вызове следующее значение из подстановки присваивается указанной переменной. Эту переменную можно использовать как параметр для любого вызова внутри блока.

Чтобы использовать многократный вызов, необходимо указать перед телом блока переменную, обозначающую очередной результат подстановки, знак «=» и саму подстановку, для которой он будет выполняться. Также для этого блока должно быть указано направление склейки. По нему и будут склеиваться результаты отдельных вызовов.

Расширим таблицу на 8, добавив блоки для каждого сервера, как показано ниже:

```
#table doccount
source = *sources(1) {
    *sources(4, [1 = source]);
    {
        {
            *print("Тип документа");
            *request(source, 1, [1 != "Системный"]) : down;
        }
        {
            *print("Количество");
            *request(source, 2, [1 != "Системный"]) : down;
        } : right
    } : down, span
} : right
#end
```

Васильеву Василию Васильевичу
от Иванова Ивана Ивановича

Отчёт о количестве документов, отправленных сотрудниками

Ниже приведено количество и тип документов, отправленных каждым сотрудником в период с 01.01.2018 по 31.01.2018:

Сотрудник	Всего	Документ 2				Документ 17				Документ 15			
		Подготовлено	Отправлено	Ошибка	Всего	Подготовлено	Отправлено	Ошибка	Всего	Подготовлено	Отправлено	Ошибка	Всего
Иванов Иван Иванович	787	123	167	1	291	87	186	0	273	106	115	2	223
Петров Пётр Петрович	894	111	113	1	223	140	184	3	327	133	209	0	342
Васильев Василий Васильевич	815	137	143	2	282	136	150	1	287	126	119	1	246
Сергеев Сергей Сергеевич	789	107	122	0	229	131	137	1	269	139	150	2	291
Александров Александр Александрович	886	115	177	2	294	113	181	0	294	112	185	1	298

Рис. 9. Пример сгенерированного документа: отчет об ошибках при отправке документов

В результате будет получена таблица, приведенная на рис. 2.

§ 8. Заключение

Представлен подход к проектированию языка макроподстановок для генерации документов со встроенным языком для генерации таблиц, имеющих сложную структуру, из вложенных друг в друг подтаблиц.

Данный язык не имеет привязки к конкретным форматам входных и выходных данных. Для формата входных данных достаточно, чтобы они имели структуру электронной таблицы или приводились к ней, а для выходных данных, чтобы он поддерживал все элементы, генерируемые шаблоном. Новые форматы можно добавлять посредством написания модулей к интерпретатору, выполняющих конвертацию между ними и внутренними форматами.

Для языка был написан прототип интерпретатора и несколько модулей к нему: модуль для получения данных с помощью SQL-запросов, модуль для чтения CSV-файлов, а также модуль для вывода сгенерированных таблиц в формате HTML.

Данный интерпретатор был протестирован на задаче генерации документов, содержащих таблицы со статистикой, подсчитываемой в нескольких однотипных базах данных. Шаблоны оказались достаточно простыми, несмотря на разнообразную структуру генерируемых документов. Два документа приведены на рис. 9 и 10.

Благодаря простой структуре шаблонов и отсутствию в них сложных и затратных по времени операций, даже при большом объеме данных, время работы интерпретатора оказалось незначительно малым по сравнению с временем выполнения SQL-запросов: как правило, интерпретатор завершает свою работу меньше, чем за секунду, в то время, когда сложные запросы к базе данных могут выполняться от нескольких минут до получаса.

Однако, несмотря на описанные преимущества, на данный момент нет возможности задавать внешний вид отдельных групп ячеек таблиц, что часто требуется для наглядного представления данных.

Еще одна проблема заключается в отсутствии альтернативы блокам при описании вложенных подтаблиц в шаблоне. Несмотря на то, что это не влияет на функциональность, при большом уровне вложенности могут возникать неудобства.

Для этого требуется расширить язык для генерации таблиц, добавив в него переменные и пользовательские функции, а также заменив многократные вызовы полноценными циклами, что позволит решить проблему с большим уровнем вложенности блоков. Также в него можно добавить набор команд для задания графических свойств ячеек и их содержимого, таких как шрифт, цвет и размер, что, в целом, не составляет особой сложности.

Васильеву Василию Васильевичу
от Иванова Ивана Ивановича

Список документов с ошибками отправки

Ниже приведены документы с ошибками отправки в период с 01.01.2018 по 31.01.2018:

Сервер 1		Сервер 2		Сервер 3	
Тип документа	Номер документа	Тип документа	Номер документа	Тип документа	Номер документа
Документ 2	49634-PH	Документ 2	93334-AV	Документ 2	89374-CE
Документ 2	85093-YO	Документ 17	58098-BH	Документ 2	51065-BT
Документ 17	28562-EM	Документ 17	23586-AH	Документ 2	56764-AF
Документ 17	95222-GH	Документ 17	76863-PQ	Документ 15	70224-AO
Документ 15	25603-VF	Документ 15	35014-GJ		
		Документ 15	86781-VO		
		Документ 15	91111-PZ		
		Документ 15	60098-XM		

Рис. 10. Пример сгенерированного документа: отчет о количестве документов, отправленных сотрудниками

Для работы с графиками в дальнейшем планируется добавить отдельный язык. Это, как и в случае с таблицами, позволит спроектировать синтаксис, максимально удобный для этой задачи, не добавляя в основной язык лишних конструкций.

СПИСОК ЛИТЕРАТУРЫ

1. Cocoon Main Site — Welcome. https://cocoon.apache.org/ (дата обращения: 07.07.2019)
2. Apache Lenya — Open Source Content Management (Java/XML). https://lenya.apache.org/ (дата обращения: 09.07.2019)
3. González E.J., Hamilton A., Moreno L., Méndez J.A., Sigut J., Sigut M. A system generating CV through intelligent agents and Apache Cocoon // Informatica. 2006. Vol. 30. No. 4. P. 453–460. http://www.informatica.si/index.php/informatica/article/view/116
4. González E.J., Hamilton A., Moreno L., Méndez J.A., Marichal G.N., Sigut J., Sigut M., Felipe J. Intelligent agents and Apache Cocoon for a CV generation system // 2007 IEEE/ACS International Conference on Computer Systems and Applications. Amman, Jordan, 2007. P. 9–15. https://doi.org/10.1109/AICCSA.2007.370858
5. González E.J., Hamilton A., Moreno L., Felipe J., Muñoz V. A multiagent architecture applied to dynamic generation of CV documents // International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008). Springer, 2009. P. 47–51. https://doi.org/10.1007/978-3-540-85863-8_7
6. Casal D. Apache Cocoon: a web applications framework for the JISC IE // VINE. 2005. Vol. 35. No. 1/2. P. 70–77. https://doi.org/10.1108/03055720510588515
7. Marchetti A., Tesconi M. PowerXML: How to user SVG to create an open tool alternative to PowerPoint // 4th Annual Conference on Scalable Vector Graphics. Enschede, Netherlands, 2005. http://www.svgopen.org/2005/papers/PowerXML/index.html

8. Фам Ван Тап, Пономарев А.А. Организация медицинской информационной системы с использованием электронных клинических документов в стандарте HL7 CDA при поддержке форматов Office Open XML // Известия ТПУ. 2010. Т. 316. № 5. С. 177–182. http://www.lib.tpu.ru/fulltext/v/Bulletin_TPU/2010/v316/i5/34.pdf
9. Kraus S., Toddenroth D., Unberath P., Prokosch H.-U., Hueske-Kraus D. An extension of the Arden syntax to facilitate clinical document generation // Studies in Health Technology and Informatics. Vol. 259. Amsterdam: IOS Press, 2019. P. 65–70. https://doi.org/10.3233/978-1-61499-961-4-65
10. Carnaz G., Beires Nogueira V., Antunes M., Ferreira N. An automated system for criminal police reports analysis // Proceedings of the Tenth International Conference on Soft Computing and Pattern Recognition (SoCPaR 2018). Cham: Springer, 2019. P. 360–369. https://doi.org/10.1007/978-3-030-17065-3_36
11. Hotomski S., Glinz M. GuideGen: An approach for keeping requirements and acceptance tests aligned via automatically generated guidance // Information and Software Technology. 2019. Vol. 110. P. 17–38. https://doi.org/10.1016/j.infsof.2019.01.011
12. Qu M., Wu X., Tao Y., Liu Y. Research on generating method of embedded software test document based on dynamic model // IOP Conference Series: Materials Science and Engineering. 2018. Vol. 322. 062018. https://doi.org/10.1088/1757-899X/322/6/062018
13. Яфаев В.Э. XML-технология создания на WEB-сервере персонализированных word-документов на основе XSL-трансформации: автореф. дис. ... канд. тех. наук. Уфа, 2009. 19 с.
14. Chen G., An B., Zeng S. A rule-based information extraction system for human-readable semi-structured scientific documents // 2015 4th International Conference on Computer Science and Network Technology (ICCSNT). Harbin, China, 2015. P. 75–84. https://doi.org/10.1109/ICCSNT.2015.7490711
15. Белых Е.А. Язык подстановок для генерации электронных документов // Математическое моделирование и информационные технологии: Национальная (Всероссийская) научная конференция (6–8 декабря 2018 г., г. Сыктывкар): сборник материалов. Сыктывкар: Изд-во СГУ им. Питирима Сорокина, 2018. С. 97–98. http://mmit2018.syktsu.ru/files/mmit2018_97.pdf
16. Фам Ван Тап. Алгоритмические и программные средства интеграции данных при создании электронных медицинских карт: автореф. дис. ... канд. тех. наук. Томск, 2011. 22 с.
17. Martens W., Neven F., Niewerth M., Schwentick T. BonXai: Combining the simplicity of DTD with the expressiveness of XML schema // ACM Transactions on Database Systems. 2017. Vol. 42. Issue 3. P. 1–42. https://doi.org/10.1145/3105960
18. Mi L., Li C., Du P., Zhu J., Yuan X., Li Z. Construction and application of an automatic document generation model // 2018 26th International Conference on Geoinformatics. Kunming, China, 2018. https://doi.org/10.1109/GEOINFORMATICS.2018.8557127
19. html 5.2: 4.9. Tabular data. https://www.w3.org/TR/html52/tabular-data.html#elementdef-table (дата обращения: 05.07.2019)

Поступила в редакцию 24.04.2019

Белых Евгений Анатольевич, студент, Институт точных наук и информационных технологий, Сыктывкарский государственный университет имени Питирима Сорокина, 167001, Россия, г. Сыктывкар, Октябрьский пр., 55.
E-mail: eugen.belyx@gmail.com

Гольчевский Юрий Валентинович, к. ф.-м. н., доцент, заведующий кафедрой информационных систем, Институт точных наук и информационных технологий, Сыктывкарский государственный университет имени Питирима Сорокина, 167001, Россия, г. Сыктывкар, Октябрьский пр., 55.
E-mail: yurygol@mail.ru

Цитирование: Е. А. Белых, Ю. В. Гольчевский. Подход к проектированию языка подстановок для генерации электронных документов, содержащих сложные таблицы // Вестник Удмуртского университета. Математика. Механика. Компьютерные науки. 2019. Т. 29. Вып. 3. С. 422–437.

COMPUTER SCIENCE 2019. Vol. 29. Issue 3. Pp. 422–437.

E. A. Belykh, Yu. V. Golchevskiy

An approach to designing a substitution language for generating electronic documents containing complex tables

Keywords: substitution language, programming languages, document generation, templates, electronic tables.

MSC2010: 68N15, 68N20

DOI: 10.20537/vm190311

This paper describes an approach to designing a substitution language for generating electronic documents based on the contents of databases and files. The proposed language involves the ability to work both with a single database and with a large number of databases with a similar structure. It has a modular structure, where additional auxiliary languages are used for generating complex document elements. One such auxiliary language is the language for generation of tables having a complex structure with subtables and extended cells. This auxiliary language will make it possible to group in a readable form a large amount of various data. It is also assumed that the substitution language and its auxiliary languages will not be bound to any input or output data formats, which will allow using any suitable formats by writing an appropriate module for the interpreter.

REFERENCES

1. Cocoon Main Site — Welcome. https://cocoon.apache.org/ (accessed 7 July 2019)
2. Apache Lenya — Open Source Content Management (Java/XML). https://lenya.apache.org/ (accessed 9 July 2019)
3. González E.J., Hamilton A., Moreno L., Méndez J.A., Sigut J., Sigut M. A system generating CV through intelligent agents and Apache Cocoon, *Informatica*, 2006, vol. 30, no. 4, pp. 453–460. http://www.informatica.si/index.php/informatica/article/view/116
4. Gonzalez E.J., Hamilton A., Moreno L., Méndez J.A., Marichal G.N., Sigut J., Sigut M., Felipe J. Intelligent agents and Apache Cocoon for a CV generation system, *2007 IEEE/ACS International Conference on Computer Systems and Applications*, Amman, Jordan, 2007, pp. 9–15. https://doi.org/10.1109/AICCSA.2007.370858
5. González E.J., Hamilton A., Moreno L., Felipe J., Muñoz V. A multiagent architecture applied to dynamic generation of CV documents, *International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008)*, Springer, 2009, pp. 47–51. https://doi.org/10.1007/978-3-540-85863-8_7
6. Casal D. Apache Cocoon: a web applications framework for the JISC IE, *VINE*, 2005, vol. 35, no. 1/2, pp. 70–77. https://doi.org/10.1108/03055720510588515
7. Marchetti A., Tesconi M. PowerXML: How to user SVG to create an open tool alternative to PowerPoint, *4th Annual Conference on Scalable Vector Graphics*, Enschede, Netherlands, 2005. http://www.svgopen.org/2005/papers/PowerXML/index.html
8. Fam Van Tap, Ponomarev A.A. Organization of a medical information system using electonic HL7 CDA clinical document supperted by Office Open XML formats, *Izvestiya Tomskogo Politekhnicheskogo Universiteta*, 2010, vol. 316, no. 5, pp. 177–182 (in Russian). http://www.lib.tpu.ru/fulltext/v/Bulletin_TPU/2010/v316/i5/34.pdf
9. Kraus S., Toddenroth D., Unberath P., Prokosch H.-U., Hueske-Kraus D. An extension of the Arden syntax to facilitate clinical document generation, *Studies in Health Technology and Informatics*, vol. 259, Amsterdam: IOS Press, 2019, pp. 65–70. https://doi.org/10.3233/978-1-61499-961-4-65
10. Carnaz G., Beires Nogueira V., Antunes M., Ferreira N. An automated system for criminal police reports analysis, *Proceedings of the Tenth International Conference on Soft Computing and Pattern*

Recognition (SoCPaR 2018), Cham: Springer, 2019, pp. 360–369. https://doi.org/10.1007/978-3-030-17065-3_36

11. Hotomski S., Glinz M. GuideGen: An approach for keeping requirements and acceptance tests aligned via automatically generated guidance, *Information and Software Technology*, 2019, vol. 110, pp. 17–38. https://doi.org/10.1016/j.infsof.2019.01.011
12. Qu M., Wu X., Tao Y., Liu Y. Research on generating method of embedded software test document based on dynamic model, *IOP Conference Series: Materials Science and Engineering*, 2018, vol. 322, 062018. https://doi.org/10.1088/1757-899X/322/6/062018
13. Yafaev V.E. XML techology of creating personilized XSL-transformation based word-documents on WEB-server, *Abstract of Cand. Sci. (Eng.) Dissertation*, Ufa, 2009, 19 p. (In Russian).
14. Chen G., An B., Zeng S. A rule-based information extraction system for human-readable semi-structured scientific documents, *2015 4th International Conference on Computer Science and Network Technology (ICCSNT)*, Harbin, China, 2015, pp. 75–84. https://doi.org/10.1109/ICCSNT.2015.7490711
15. Belykh E.A. Substitution language for electronic documents generation, *Matematicheskoe modelirovanie i informatsionnye tekhnologii: Natsional'naya (Vserossiiskaya) nauchnaya konferentsiya (6–8 dekabrya 2018 g., g. Syktyvkar): sbornik materialov* (Mathematical Modeling and Information Technologies: National (All-Russian) Scientific Conference (December 6–8, 2018, Syktyvkar): collection of materials), Syktyvkar: Pitirim Sorokin Syktyvkar State University, 2018, pp. 97–98 (in Russian). http://mmit2018.syktsu.ru/files/mmit2018_97.pdf
16. Fam Van Tap. Algorithmic and software data intergration tools when creating electonic medical records, *Abstract of Cand. Sci. (Eng.) Dissertation*, Tomsk, 2011, 22 p. (In Russian).
17. Martens W., Neven F., Niewerth M., Schwentick T. BonXai: Combining the simplicity of DTD with the expressiveness of XML schema, *ACM Transactions on Database Systems*, 2017, vol. 42, issue 3, pp. 1–42. https://doi.org/10.1145/3105960
18. Mi L., Li C., Du P., Zhu J., Yuan X., Li Z. Construction and application of an automatic document generation model, *2018 26th International Conference on Geoinformatics*, Kunming, China, 2018. https://doi.org/10.1109/GEOINFORMATICS.2018.8557127
19. html 5.2: 4.9. Tabular data. https://www.w3.org/TR/html52/tabular-data.html#elementdef-table (accessed 5 July 2019)

Received 24.04.2019

Belykh Evgenii Anatol'evich, Institute of Exact Sciences and Information Technologies, Pitirim Sorokin Syktyvkar State University, Oktyabrskii pr., 55, Syktyvkar, 167001, Russia.
E-mail: eugen.belyx@gmail.com

Golchevskiy Yuriy Valentinovich, Candidate of Physics and Mathematics, Associate Professor, Head of Department of Information Systems, Institute of Exact Sciences and Information Technologies, Pitirim Sorokin Syktyvkar State University, Oktyabrskii pr., 55, Syktyvkar, 167001, Russia.
E-mail: yurygol@mail.ru

Citation: E. A. Belykh, Yu. V. Golchevskiy. An approach to designing a substitution language for generating electronic documents containing complex tables, *Vestnik Udmurtskogo Universiteta. Matematika. Mekhanika. Komp'yuternye Nauki*, 2019, vol. 29, issue 3, pp. 422–437.