АЛГОРИТМ ПРЕОБРАЗОВАНИЙ XML ДОКУМЕНТА С СОХРАНЕНИЕМ ДОПУСТИМЫХ ИЕРАРХИЧЕСКИХ ОТНОШЕНИЙ

Т.М. Сысоев
tim@ccas.ru
Межведомственный суперкомпьютерный центр РАН

Аннотация

В докладе описывается метод преобразования документов в формате XML, который позволяет сохранить допустимые иерархические отношения между тегами при выполнении операций копирования и вставки произвольных фрагментов. Данный метод применяется для программного построения отчетов с использованием офисных пакетов Microsoft Office или OpenOffice на основе поддерживаемого этими пакетами представления документов в виде XML-файлов. В процессе генерирования отчетов на основе шаблонов возникает необходимость выполнения операций копирования и вставки при сохранении структуры XML-документа в целом.

При внедрении электронного документооборота часто возникает задача построения электронных документов на основе информации, содержащейся в каком-либо хранилище. В связи с тем, что основные офисные пакеты в настоящий момент позволяют сохранять свои документы в формате XML [1], [2], итоговый результат может быть представлен в виде XML файла, при этом в процессе построения могут быть использованы стандартные технологии для создания и обработки XML документов. Использование шаблонов, которые сами являются документами с простым языком для подстановки переменных, условных выражений и средствами организации циклов удобно для пользователей, поскольку они самостоятельно могут вносить исправления в шаблон, связанные, например, с оформлением некоторых его частей, используя обычный офисный пакет.

Для генерации документа необходимо взять его шаблон и выполнить описанные в нём преобразования. Основными операциями, которые при этом возникают, являются удаление фрагмента текста между двумя произвольными позициями в документе, и копирование фрагмента текста между двумя произвольными позициями в заданную третью позицию.

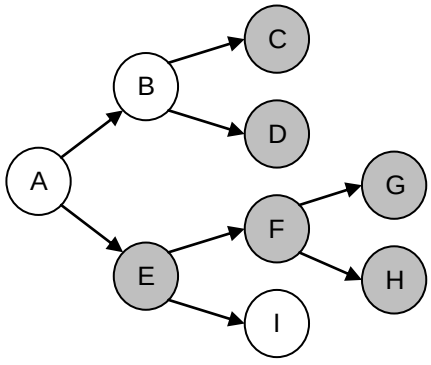

Рис. 1. Множество (B, E]

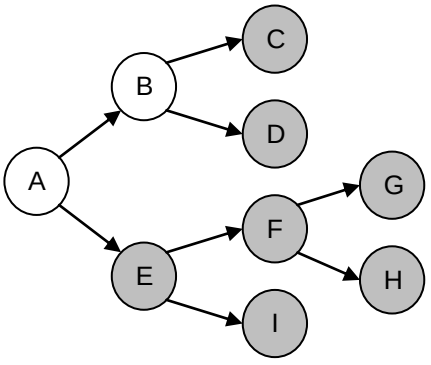

Рис. 2. Фрагмент (A, B)

Обозначим как (X, Y] все элементы, которые находятся в списке вершин, полученных обходом дерева документа алгоритмом поиска в глубину между X и Y, исключая вершину X

(см. Рис. 1). В рассматриваемых офисных пакетах непрерывному выделению текста соответствует подобное множество вершин.

Для реализации операций удаления и копирования оказалось удобным работать со специальными фрагментами документа, которые определяются двумя элементами C и E, такими, что E является потомком (возможно, не непосредственным) у элемента C. Если выписать все дочерние элементы C, полученные обходом дерева алгоритмом поиска в глубину, начиная с элемента C, то мы получим список, в котором присутствует E. В фрагмент документа, заданный элементами C и E, и обозначаемый далее как (C, E), входят все элементы этого списка, следующие после E (см. Рис. 2). Например, в (C, C) входят все дочерние элементы C, но сам элемент C не входит.

Множество (X, Y] может быть представлено в виде (P, X) – (P, Y), где P – общий родительский элемент для X и Y. Например, для ситуации, изображенной на рис 1, (B, E] = (A, B) – (A, E), поскольку множество (A, E) состоит из единственной вершины I. В связи с этим, операция удаления из документа множества (X, Y] сводится к операции удаления фрагмента (P, X) и последующей вставки фрагмента (P, Y) для компенсации удаления "лишних" вершин. Аналогично, операция копирования (X, Y] в произвольную позицию документа сводится к операции вставки фрагмента (P, X) и последующего удаления множества (P', Y'), соответствующего множеству (P, Y) после операции копирования.

Таким образом, реализовав операции удаления и копирования фрагментов (C, E), мы сможем выполнять эти же операции для любого множества (X, Y], сводя их к операциям над фрагментами. В свою очередь, к операциям копирования и удаления фрагментов предъявляется требование сохранения структуры: отношения родитель-потомок между элементами в документе, полученном после выполнения операции, должны соответствовать отношениям в документе до выполнения операции. Например, если в исходном документе элемент LI является потомком элемента UL, то в преобразованном документе не должно быть ситуации, когда он является потомком элемента P. Для выполнения этого требования при операции копирования может потребоваться удаление существующих, либо создание новых элементов.

Операция удаления всех элементов (C, E) из документа удовлетворяет требованию сохранения структуры, поскольку новых связей (присоединения вершин к новым родителям) при этом не образуется. В то же время, для операции удаления произвольного множества (X, Y] это утверждение неверно: например, на рис. 1 изображена ситуация, когда после удаления у нас остаётся элемент I, который необходимо присоединить к какому-либо элементу в документе таким образом, чтобы требование сохранения структуры выполнялось.

Операция вставки выполняется следующим образом:

1. Для фрагмента (C, E) строится множество вершин [E, E_1, E_2, …, C] таких, что каждая последующая является родителем предыдущей, первая совпадает с E, последняя с C (на рис. 3 это множество [C, B, A]).
2. Для каждого элемента этого множества, начиная от позиции для копирования, ищется элемент, который соответствует текущему элементу в том смысле, что может содержать тех же потомков. Данная операция может приводить к "смещению" позиции для копирования вверх или вниз по иерархии, а также к созданию новых элементов для сохранения допустимой иерархии или для обеспечения того, чтобы добавляемые элементы шли последовательно.

Например, если нам необходимо добавить элемент LI после элемента P, в то время как LI может быть дочерним элементом только для UL, понадобиться создать элемент UL как потомок P, после чего он становится текущей позицией для копирования.

3. К найденному элементу добавляются потомки текущего элемента.

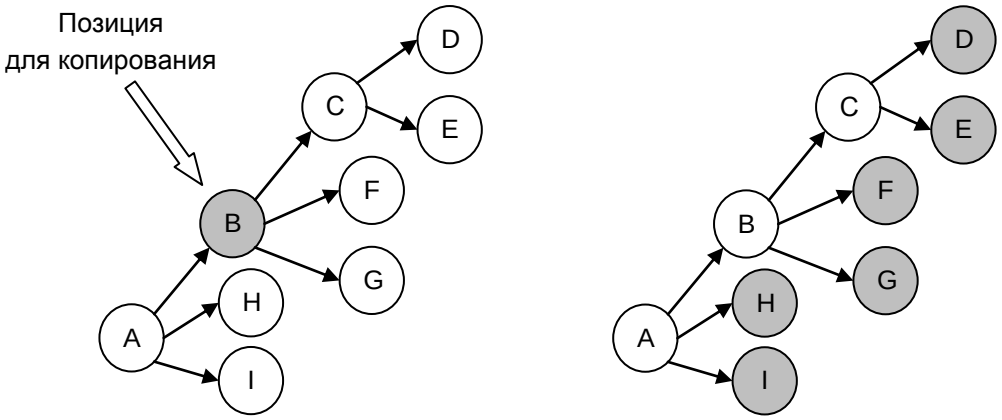

Рис. 3. Выполнение операции копирования

Заключение

В статье рассмотрен наиболее сложный вопрос, который возник при построении системы генерации отчетов с использованием офисных приложений на основе шаблонов, редактируемых пользователем. Следует отметить, что оба поддерживаемых офисных приложения позволяют выполнять операции над текстом через программный интерфейс. В то же время, при использовании этого интерфейса возникают серьезные проблемы, связанные с производительностью и масштабируемостью решения, которые привели к необходимости прямой работы с файлами документов.

Литература

[1] Open Document Format for Office Applications (OpenDocument) v1.1, http://docs.oasis-open.org/office/v1.1/OS/OpenDocument-v1.1.pdf – OASIS, 2007

[2] Office Open XML File Formats, http://www.ecma-international.org/publications/standards/Ecma-376.htm – Ecma International, 2008