БЫСТРЫЙ АЛГОРИТМ ФИЛЬТРАЦИИ ИЗОМОРФНЫХ XSD-СХЕМ

А.П. Сергеев

Компьютерное издательство ООО «Диалектика», ул. Тверская, д. 6, офис 303, Киев, Украина

a_p_sergeyev@mail.ru

Резюме. *Предлагается быстрый алгоритм фильтрации изоморфизма XSD-схем, который позволит минимизировать объем хранилищ данных, предназначенных для хранения XML-контента. Эта задача решается путем исключения XML-документов, XSD-схемы которых изоморфны канонической (нормализованной) XSD-схеме. Благодаря использованию распараллеливания в процессе фильтрации документов с изоморфными XSD-схемами значительно повышается быстродействие и эффективность алгоритма.*

Ключевые слова

Параллельные вычисления, XSD-схема, Изоморфизм, изоморфные деревья, перенумерация вершин

Вступление

В связи с массовым внедрением электронных хранилищ данных возникают проблемы, связанные с эффективной организацией, хранением и обработкой информации, а также взаимодействия между различными хранилищами данных с помощью как защищенных, так и общедоступных каналов связи. Де факто в качестве универсального формата хранилищ данных используется XML.

Наравне со многими преимуществами, этому формату присущи и определенные недостатки. Один из основных недостатков — *избыточность*, которая приводит к значительному увеличению объема хранилищ данных. Учитывая огромные объемы информации, хранящейся в государственных и корпоративных хранилищах данных (например, объем архива, включающего декларации о доходах граждан Украины за 2009 год, составляет около 180 Тбайт), оптимизация хранилищ данных позволит значительно уменьшить объем накопителей информации, применяемых в хранилищах данных. А это, в свою очередь, позволит уменьшить стоимость хранения единицы данных, а также ускорить поиск и выбору информации.

Описание алгоритма

Как отмечалось в [1], разметка (структура) однозначным образом определяется с помощью языка W3C XSD (XSD-схемы).

Явное построение деревьев на основе формальным образом описанных XSD-схем осуществляется с помощью стандартных парсеров. На практике часто требуется выполнение обратного действия — реконструкция XSD-схемы по ее дереву. Деревья, построенные по XSD-схемам, хранятся в виде матриц смежности. Матрица смежности представляет собой квадратную матрицу n×n (n – количество узлов дерева), элементы которой вычисляются по формулам (1): Mij = 1, i,j=1…n, если узлы i и j соединены ребром; Mij = 0, i,j=1..n, если узлы i и j не соединены ребром (1).

В [2] утверждается, что изоморфные XSD-схемы порождают эквивалентные XML-документы. Следовательно, путем исключения изоморфных XSD-схем можно оставить эталонную (каноническую) XSD-схему, которая будет порождать класс эквивалентных XML-документов.

Поскольку дерево — частный случай графа, алгоритм проверки изоморфизма деревьев является частным случаем алгоритма проверки изоморфизма графов. Напомню основные положения более общего алгоритма [2].

Утверждение 1. Пусть V_1 – множество вершин графа G_1, а E_1 – множество его ребер. Пусть V_2 – множество вершин графа G_2, а E_2 – множество его ребер. Графы [2] G_1 и G_2 называются изоморфными, если существует взаимно однозначное соответствие $f(V_1)\rightarrow V_2$ такое, что (v,w) принадлежит V_1 тогда и только тогда, когда (f(v),f(w)) принадлежит V_2. Иными словами, существует соотношение между вершинами графа G_1 и вершинами графа G_2, которое сохраняет отношение смежности. Это отношение представляет собой количество ребер, входящих (или исходящих) из каждой вершины графа. Т.е. графы, между которыми установлено отношение изоморфизма, отличаются друг от друга лишь пометками вершин.

Утверждение 2. Под *инвариантом изоморфизма* понимаются характеристики графа, которые сохраняются без изменений при изоморфном отображении. Система инвариантов изоморфизма называется *полной*, если из равенства между собой всех инвариантов изоморфизма вытекает изоморфизм графов.

Простейшим инвариантом изоморфизма является количество вершин графа. Очевидно, что между графами с различным количеством вершин не может быть установлено взаимно однозначное отношение изоморфизма. В дальнейшем будут рассмотрены примеры других инвариантов изоморфизма.

Алгоритм определения изоморфизма графов выглядит следующим образом.

1. Определяется система инвариантов изоморфизма.
2. Выполняется сравнение инвариантов изоморфизма.
3. Если установлено неравенство хотя бы в одной из пар инвариантов изоморфизма, графы будут неизоморфны.
4. Если результаты проверки равенства инвариантов изоморфизма положительны и система инвариантов изоморфизма является полной, графы будут изоморфными.
5. Если система инвариантов изоморфизма является неполной, для установления изоморфизма графов используется алгоритм поиска с возвращением (back-tracking).

Предположим, что в нашем распоряжении имеются два графа $G_1 = (V_1,E_1)$ и $G_2 = (V_2,E_2)$, для которых нужно установить факт изоморфизма. Пусть $V_1 = V_2 = \{1,2,...,n\}$ – множество вершин графов G_1 и G_2. Один из графов, например G_1, выбирается в качестве эталона для сравнения в процессе установления факта изоморфизма. Пусть $G_1(k)$ – подграф графа G_1, индуцированный вершинами $\{1,2,\ldots,k\}$, $0\leq k\leq n$. $G_1(0)$ – пустой (нулевой) подграф, а $G_1(1)$ – подграф, который состоит из одной вершины и не имеет ребер. Очевидно, что пустой подграф $G_1(0)$ изоморфен пустому подграфу $G_2(0)$. Предположим, что на определенном этапе найден подграф G_2, который состоит из набора вершин S, являющегося подмножеством множества вершин V_2. Установлено, что этот подграф изоморфен подграфу G_1. А теперь попробуем продолжить отношение изоморфизма на подграф $G_1(k+1)$ путем выбора вершины v из подмножества V_2/S. Если подобная вершина v найдена, зафиксируем соотношение $f_{k+1} \rightarrow v$ и попробуем продолжить отношение изоморфизма на подграф $G_1(k+2)$. Если же требуемая вершина v не найдена, возвращаемся к подграфу $G_1(k)$ и выбираем другую вершину среди k вершин из множества V_1. Описанный процесс продолжается до тех пор, пока k не станет равным n и, соответственно, не будет установлен изоморфизм между подграфом $G_1(n)=G_1$ и G_2. Если же на определенном шаге процесса перебора обнаруживается, что подграфы неизоморфны, это означает, что графы G_1 и G_2 также будут неизоморфными.

В худшем случае (при осуществлении перебора всех подграфов G(k), где k=1...n), вычислительная сложность этого алгоритма равна O(n!) операций. На практике же вычислительная сложность будет существенно меньше. Она будет полиномиальной, если факт изоморфизма устанавливается на основе равенства инвариантов изоморфизма.

На основе общего алгоритма определения изоморфизма графов создается частный алгоритм определения изоморфизма деревьев (XSD-схем). Этот алгоритм основан на следующих утверждениях [2].

Утверждение 3. Если XSD-схемы изоморфны (в смысле изоморфизма соответствующих им деревьев), порождаемые ими XML-документы будут эквивалентны.

Утверждение 4. Проверка изоморфизма XSD-схем сводится к проверке изоморфизма соответствующих этим схемам деревьев.

Из этих утверждений следует, что проверка изоморфизма XSD-схем сводится к проверке изоморфизма деревьев. В этом случае вычислительная сложность алгоритма определения изоморфизма существенно уменьшается. Более точная оценка будет приведена в заключительном разделе статьи.

Поскольку дерево является частным случаем графа (граф без циклов и петель), по отношению к ним справедливо утверждение 1. Перефразируя это утверждение можно сказать, что два дерева будут изоморфны, если они имеют одинаковую структуру. Внешний же вид изоморфных деревьев может отличаться. На рис. 1 показаны два изоморфных дерева, а на рис. 2 — два неизоморфных дерева

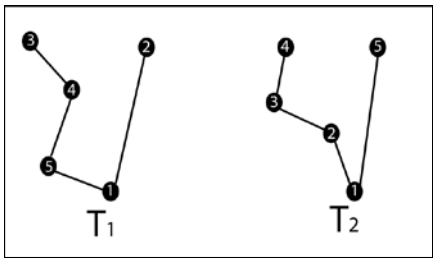

Рис. 1. Изоморфные деревья.

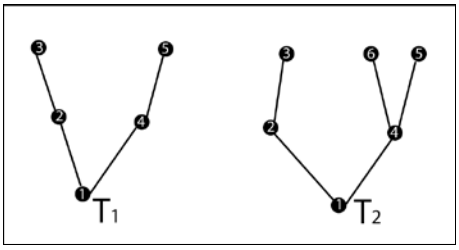

Рис. 2. Неизоморфные деревья.

В процессе проверки «подозрительных на изоморфизм» деревьев вычисляются инварианты изоморфизма, Сначала для всех деревьев вычисляется главный инвариант изоморфизма — количество вершин для каждого дерева. Затем все деревья разбиваются на n классов, где n=1…n — количество вершин. Затем в игру включается распараллеливание, когда каждый отдельный процессор выполняет поиск и фильтрацию изоморфных деревьев, входящих в один из n классов. Количество процессоров, обеспечивающих наибольшую скорость вычислений, p=n, где n – количество классов потенциально изоморфных деревьев (XSD-схем).

Затем в каждом классе рассматривается ограниченный класс деревьев, которые имеют общий корень. Подобные деревья порождаются XSD-схемами, которые имеют общий корневой элемент.

Чтобы определить изоморфизм деревьев, имеющих общий корень, используется *алгоритм сравнения*. В процессе выполнения этого алгоритма идентифицируется структура самого дерева, причем не учитываются пометки вершин (деревья одинаковой структуры, у которых перенумерованы вершины, считаются изоморфными).

На рис. 1 показаны два изоморфных дерева T_1 и T_2. Эти деревья имеют одинаковую структуру (дерево в правой части рисунка является практически идентичным дереву, изображенному в левой части рисунка). Изоморфизм означает взаимно однозначное соответствие, установленное между вершинами деревьев T_1 и T_2 (2):

1↔1
2↔5
3↔4
4↔3
5↔2

Каждой вершине дерева ставится в соответствие числовая последовательность вида $a,b,(x_1,x_2,...x_n)$, где a - уровень вершины дерева, отсчитывая от его корня; b - количество потомков для данной вершины (длина максимальной линии, образованной потомками); $(x_1, x_2,\ldots,x_n)$ - ряд, содержащий количество потомков для дочерних вершин данной вершины. Назовем эту последовательность *характеристическим вектором*. Как будет показано в дальнейшем, именно характеристический вектор является инвариантом изоморфизма для деревьев.

Построим характеристический вектор для рассматриваемого примера. Для вершин дерева T_1 этот вектор будет иметь следующий вид (3):

- вершина 1 – 0, 3, (2,0);
- вершина 2 – 1, 4, (3);
- вершина 3 – 3, 4, (3);
- вершина 4 – 2, 3, (0,2);
- вершина 5 – 1, 2, (1, 1).

А теперь построим характеристический вектор для дерева T_2 (4):

- вершина 1 – 0, 3, (2,0);
- вершина 5 – 1, 4, (3);
- вершина 4 – 3, 4, (3);
- вершина 3 – 2, 3, (0,2);

- вершина 2 – 1, 2, (1, 1).

Результатом построения характеристических векторов для деревьев T_1 и T_2 явились два массива записей. Теперь в соответствии с утверждением 1 можно заключить, что если между элементами этих двух массивов возможно установление взаимно однозначного соответствия, то заданные этими массивами деревья (частный случай графа) будут изоморфными. И действительно, на основе анализа характеристических векторов (3) и (4) нетрудно заметить, что имеет место однозначное соответствие (изоморфизм) между вершинами деревьев T_1 и T_2 (1). С точностью до перестановки элементов характеристические векторы (3) и (4) равны между собой, и поскольку они соответствуют деревьям, имеющим общий корень, из равенства векторов вытекает факт изоморфизма деревьев (и соответствующих им XSD-схем). В данном случае характеристические векторы представляют собой полную систему инвариантов изоморфизма – т.е. на основе их равенства между собой можно сделать вывод об изоморфизме деревьев, а затем отфильтровать лишние деревья.

Алгоритм сравнения безупречно работает в том случае, когда сравниваемые деревья имеют один и тот же корень. В связи с этим возникает вопрос, как быть в более общем случае, когда деревья имеют разные корневые вершины, как показано на рис. 3.

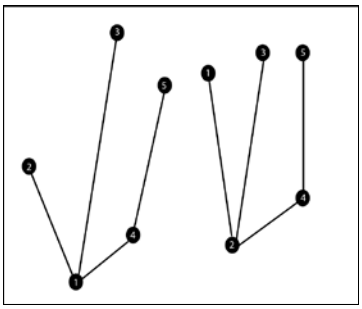

Рис. 3. Изоморфные деревья, имеющие различные корневые вершины

Если требуется установить изоморфизм для деревьев с различными корневыми вершинами, то в дополнение к описанному выше алгоритму сравнения применяется *алгоритм перенумерации вершин*. Ниже приводится краткий обзор этого алгоритма.

1. Для первого дерева (рис. 3, слева) следует найти *концевую вершину* (вершина, у которой отсутствуют потомки), затем сделать ее корневой. Другими словами, происходит «захват» дерева за одну из вершин и его «вытягивание» вниз. Полученное в результате выполнения этой операции дерево принимается за эталонное (рис. 4, справа).
2. Выбираются все концевые вершины для второго дерева, которые поочередно «превращаются» в корневые. Затем производится сравнение преобразованного дерева с эталонным. При этом применяется описанный ранее алгоритм сравнения.
3. Если в результате применения алгоритма сравнения было хоть раз достигнуто равенство, это означает наличие изоморфизма деревьев.

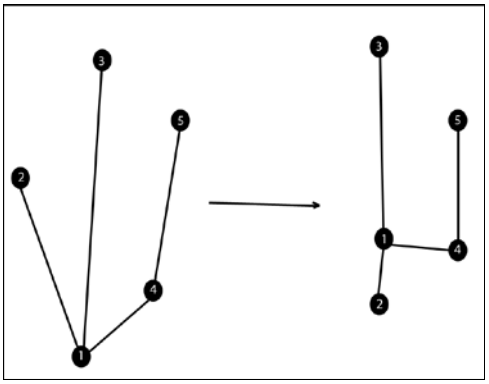

Рис. 4. Процесс «вытягивания» дерева

Вычислительная сложность описанного алгоритма проверки изоморфизма деревьев является полиномиальной и в большинстве случаев оценивается величиной $O(n^2)$, где n – количество вершин в любом из сравниваемых деревьев. Если проверяется изоморфизм деревьев, имеющих общий корень, вычислительная сложность оценивается как O(n).

На рис. 5 приведено схематическое изображение алгоритма определения изоморфизма деревьев.

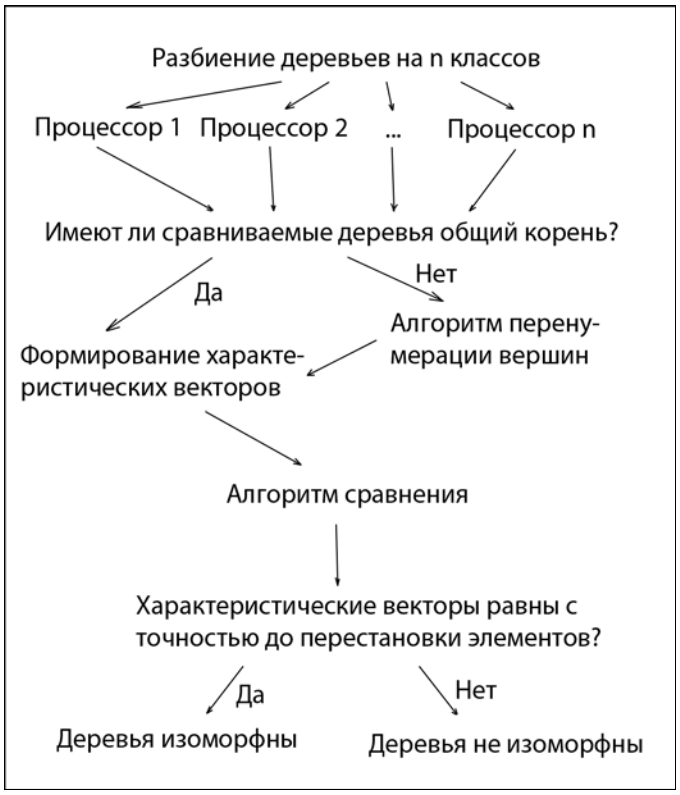

Рис. 5. Блок-схема быстрого алгоритма фильтрации изоморфных XSD-схем

Заключение

В работе рассмотрен быстрый алгоритм определения изоморфизма XSD-схем, сводящийся к определению изоморфизма соответствующих им деревьев. Благодаря применению этого алгоритма из хранилища данных исключаются XML-документы, XSD-схемы которых изоморфны каноническим XSD-схемам. В результате существенно уменьшается объем памяти хранилища данных, выделяемой для хранения XML-документов. Благодаря применению алгоритма распараллеливания по отношению к разбитым на классы потенциально изоморфных XSD-схемам существенно возросло быстродействие алгоритма. Максимальный прирост скорости при использовании n процессоров (по сравнению с одним процессором) составит n/2.

Ссылки

[1] W3C/XML Technology/Schema [http://www.w3.org/standards/xml/schema].

[2] Алгоритм определения изоморфизма XML-схем. *Проблемы программирования №2-3, 2010. Материалы седьмой международной научно-практической конференции по программированию УкрПРОГ 2010, 2527 мая 2010 г. Украина, Киев* 2(3):530-536, 2010.