Компьютерные инструменты в образовании, 2018
№ 1: 51–64
УДК: 004.89 : 004.62
http://ipo.spb.ru/journal

информационные системы

XML-РЕЛЯЦИОННОЕ ПРЕОБРАЗОВАНИЕ С ИСПОЛЬЗОВАНИЕМ СИСТЕМЫ ПРОДУКЦИОННЫХ ПРАВИЛ

Лямин А. В.[1]

[1]Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, Санкт-Петербург, Россия

Аннотация

XML-реляционное преобразование является нетривиальной задачей, для решения которой могут быть применены продукционные правила. Цель исследования заключается в разработке системы продукционных правил для выполнения однозначного преобразования XML-документов и реляционных структур данных. В рамках исследования был проведен анализ различных способов XML-реляционного преобразования, используемых в современных информационных системах, был разработан алгоритм однозначного преобразования иерархических структур данных к реляционным и обратно с использованием системы продукционных правил. В статье рассмотрен пример использования разработанной системы продукционных правил для передачи данных о виртуальных лабораториях.

Ключевые слова: *информационные системы, XML-реляционное преобразование, система продукционных правил.*

Цитирование: Лямин А. В. XML-реляционное преобразование с использованием системы продукционных правил // Компьютерные инструменты в образовании. 2018. № 1. С. 51–64.

1. ВВЕДЕНИЕ

В быстро меняющейся и расширяющейся информационной среде существует множество информационных систем, работающих с большими массивами данных. К числу таких систем относятся банковские и бухгалтерские системы, онлайн-энциклопедии, системы открытого онлайн-обучения и др. Для хранения перманентных данных в таких системах чаще всего используются реляционные базы данных, обладающие высокой производительностью и надежностью. Применение реляционной модели позволяет минимизировать избыточность данных и снизить риск логических ошибок при записи, обновлении, извлечении и удалении данных.

В каждой информационной системе необходимо определить способ обмена информацией. Множество систем использует для обмена информацией язык разметки XML (eXtensible Markup Language) [1, 2] или формат JSON (JavaScript Object Notation), основанные на иерархической модели данных, преимуществами которой являются:

— доступность, гибкость и тематическая ориентированность формата не требует глубоких знаний в области информационных технологий;
— стандартизация структуры данных на основе DTD-определения (Document Type Definition) или XSD-схем (XML Schema Definition) позволяет построить схемы для обмена информацией между различными системами;
— существование единого стандарта данных позволяет верифицировать структуру получаемых, передаваемых и хранимых данных.

Однако форматы данных, основанные на иерархической модели, приводят к избыточности и неэффективны для хранения больших объемов данных.

Применение реляционной модели для хранения данных и иерархической для обмена информации предоставляет возможности удобного хранения и передачи данных, но в то же время поднимает вопрос об определении условий соответствия данных, представленных с использованием двух разных моделей.

Структура XML-документа представляет собой дерево, главным структурным узлом которого является элемент. Каждый элемент в XML-документе может быть описан набором атрибутов и содержать другие элементы. Элемент, включающий все остальные элементы XML-документа, называется корневым элементом. При этом корректный XML-документ должен удовлетворять нескольким правилам:

— корневой элемент в документе должен быть представлен только один раз;
— элементы, не включающие другие элементы, называются пустыми и должны иметь открывающие и закрывающие теги;
— закрывающие теги должны в документе появляться в обратном порядке по отношению к открывающим тегам.

Описание структуры XML-документа может представляться в виде DTD-определения или XML-схемы, являющейся корректным XML-документом, предоставляющим возможность использования простых и сложных типов данных, установленного списка значений и установки типов отношений. Поскольку XML-документ представляет собой дерево, то представление XML-документа в виде табличных данных с определенными столбцами и строками является нетривиальной задачей.

2. ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ

На данный момент не существует единого метода хранения данных XML-документа в реляционной базе данных [2]. Все существующие на текущий момент методы могут быть разделены на три категории, каждая из которых имеет свои достоинства и недостатки.

Хранение информации в виде текстовых блоков данных. Одно из основных преимуществ данного метода — это отсутствие необходимости проверки корректности XML-документа, поскольку документ хранится в оригинальном формате. Однако пользователи не могут использовать возможности системы управления базами данных (DBMS) для осуществления эффективного поиска, выбора и обновления информации. Для осуществления этих действий документ необходимо скачать, изменить и загрузить в базу данных повторно. Такой метод хранения информации не эффективен и ведет к избыточности данных.

Использование типа данных XML реляционных баз данных. В некоторых системах управления базами данных существует встроенный тип данных XML. Такие методы не требуют предварительных преобразований XML-документов и поддерживают методы

эффективного манипулирования данными XML. В то же время они не исключают избыточности иерархических структур, и, следовательно, операции поиска и изменения информации становятся более сложными и требуют большего количества ресурсов.

Преобразование иерархических данных к реляционной модели. Методы, относящиеся к третьей категории, подразумевают преобразование XML данных к столбцам и строкам таблицы базы данных. Это позволяет использовать встроенные методы поиска, выбора, изменения и удаления данных. Запросы, представленные в структуре XML-документа, должны быть преобразованы в запросы языка SQL, и, следовательно, результаты выполнения запросов должны быть преобразованы в формат XML. Эффективность таких методов зависит от количества и сложности выполняемых преобразований. Известные разработчики DBMS реализуют инструменты, поддерживающие выполнение таких преобразований. Универсального метода на текущий момент не существует, однако известны следующие методы:

— создание неизбыточного нормализованного представления содержимого XML-документа с итоговым минимальным набором отношений путем анализа первичных и внешних ключей и функциональных зависимостей [4];
— определение набора таблиц и специального алгоритма, описывающего преобразование запросов типа XPath в язык запросов SQL (Structured Query Language) [5, 6];
— использование самообучаемого алгоритма преобразования XML-документа к реляционной модели [7].

Современные системы управления базами данных обладают встроенными методами преобразования иерархических данных к реляционной модели. В основном, встроенные методы преобразования различных моделей данных предполагают наличие следующих связей: одно отношение для одного элемента XML, отдельный атрибут отношения для отдельного дочернего элемента XML в элементе, соответствующем отношению. В качестве примера можно рассмотреть Microsoft Access DBMS. Она поддерживает импорт и экспорт иерархических структур данных, экспорт XSD-определения, XSL и HTML представлений. В данной системе существует ограничение на экспорт более одного отношения, что не позволяет экспортировать связи между отношениями, основанными на внешних ключах. Импорт в базу данных доступен из любого корректного XML-документа вне зависимости от того, основан он на XML-схеме или нет. При наличии XML-схемы создается новое отношение с определенными индексами и типами данных. Если существует отношение с таким же именем, к имени добавляется автоинкрементируемое число. При отсутствии XML-схемы все созданные связи будут представлены строковым типом.

Другой системой управления базами данных, имеющей встроенные методы преобразования данных, является клиент-серверная DBMS Oracle. Она позволяет хранить XML-документы в различных форматах, в частности, в виде текстовых блоков CLOB. Помимо этого, поддерживаются методы манипулирования данными XML, например запросы выборки, изменения и добавления данных, генерирующие XML-документы. Oracle 9 стал поддерживать тип данных XML (XMLType), что позволяет создавать связи и отношения, имеющие тип XMLType и отдельные атрибуты, хранимые в структурированном или неструктурированном виде. В случае использования структурированного способа хранения данных с использованием XML-схемы применяются те же типы индексирования данных, что и в реляционной модели, что позволяет расширить возможности получения доступа к данным с использованием таких команд SQL, как existNode, extract и т. д. Однако такой способ доступа к данным имеет ограничения преобразований сложных и простых типов данных. Oracle предоставляет различные инструменты

работы с XML-данными, например XML-схема, язык шаблонов XSLT (eXtensible Stylesheet Language Transformation), анализатор XML данных, утилита XSU (XML SQL Utility) для генерации XML вместе с DTD-определением и XML-схемой на основе запросов. С другой стороны, неструктурированное хранение XML-документов существенно увеличит объем базы данных, а использование технологий сжатия XML замедлит общую производительность системы.

Наиболее эффективными методами считаются методы, основанные на принципах искусственного интеллекта, объединяющего множество областей знаний, в том числе машинное обучение, автоматическое планирование, анализ естественного языка, представления знаний, аффективные вычисления и т. д. Такие методы требуют подробный анализ входных аргументов и построение выходных значений на основе определенных правил, например при построении онтологий или задачах распознавания образов.

Система продукционных правил — один из наиболее распределенных подходов искусственного интеллекта, основанный на анализе содержания входных данных. Продукционные правила анализируют пререквизиты выполняемого действия ("IF"), а результаты применения правил располагаются в блоке действий ("THEN"). Таким образом, должны быть определены следующие параметры для корректного формирования правил:

S — класс применимости правила;

L — условие запуска правила;

$A \to B$ — основная часть правила, содержащая описание выполняемого действия и условия его выполнения;

Q — постусловия выполнения правила.

Полная формула для продукционного правила выглядит следующим образом:

$$\langle S, L, A \to B, Q \rangle$$

В статье представлен метод XML-реляционного преобразования, основанный на системе продукционных правил. Метод определяет правила, позволяющие осуществлять эффективное двустороннее преобразование между иерархическими и реляционными структурами данных. Разработанный метод успешно реализуется в электронной информационно-образовательной среде AcademicNT Университета ИТМО [8, 9], содержащей множество информации об используемых в системе обучающих материалах.

3. МЕТОД XML-РЕЛЯЦИОННОГО ПРЕОБРАЗОВАНИЯ

Разработанный метод XML-реляционного преобразования основан на использовании продукционных правил, сформулированных для реализации алгоритма преобразования, применяемого в электронной информационно-образовательной среде Университета ИТМО AcademicNT. В ходе разработки данной информационной системы в качестве основного хранилища данных было решено использовать реляционную базу данных и формат XML для обмена информацией об электронных учебно-методических материалах с другими системами. Стандарт описания обучающих материалов импортируется в реляционные базы данных в виде нескольких связанных между собой XML-документов для разных компонентов, в том числе учебного плана, рабочей программы дисциплины, онлайн-курса, электронного конспекта, электронного теста, виртуальной лаборатории, электронного практикума и информационного ресурса. Для каждого отдельного компонента определяются DTD-определения или XML-схемы, а

каждый документ является логически завершенным и может быть заполнен данными, характеризующими компонент. В тот же момент, реляционная база данных, содержащая необходимые сведения, имеет структуру, удовлетворяющую всем необходимым требованиям, то есть нормализации, оптимизации выборок данных и т. д., и ориентированную на динамическую обработку и выдачу хранимых данных в соответствии с отправляемыми запросами. Импорт и экспорт XML-документов реализуется с помощью разработанной библиотеки, осуществляющей анализ данных и проверку всех необходимых условий преобразования данных. Библиотека разработана на языке программирования Java и ориентирована на работу с системой управления реляционными базами данных Oracle.

С целью загрузки обучающих материалов, переданных в формате XML, в реляционную базу данных была разработана система продукционных правил, представленная в таблице 1. Использование разработанных правил позволяет сформировать полное отражение элементов и их атрибутов в отношения и их атрибуты в реляционной структуре и дочерние элементы и их атрибуты в связи между отношениями. В ходе преобразования необходимо хранить последовательность обрабатываемых элементов для осуществления корректного отображения данных. Таким образом, выполняются главные принципы системы продукционных правил, основанные на разделении различных классов ситуаций, возникающих в ходе выполнения преобразования. В рамках разработанной системы правил используются следующие обозначения:

- *Attributes*(*source*) — функция возвращает список атрибутов для указанного объекта (*source*);
- *checksumCRC32*(*binary object*) — функция рассчитывает и возвращает контрольную сумму по алгоритму *CRC*32 для бинарного объекта (*binary object*);
- *Descendants*(*element*) — функция возвращает список потомков текущего элемента (*element*);
- *download*(*binary object*) — функция возвращает файл, сформированный на основе бинарного объекта (*binary object*) базы данных;
- *exists*(source) — функция проверяет наличие объекта (*source*) в документе или базе данных и возвращает true, если объект существует, иначе — *false*;
- *getAttribute*(*name*) — метод возвращает атрибут с указанным именем (*name*);
- *getRelation*() — метод возвращает отношение базы данных на основе анализа имени объекта, для которого вызывается;
- *getValue*() — метод возвращает значение объекта;
- *isEmpty*() — метод проверяет значение объекта, для которого вызывается, возвращает *true*, если объект пустой, иначе — *false*;
- *not*() — логическая функция отрицания;
- *Relations* — множество отношений реляционной базы данных;
- *setValue*(*value*) — метод присваивает значение (*value*) объекту, для которого вызывается;
- *ShowError*() — функция возвращает сообщение об ошибке;
- *store*(*source*) — функция сохраняет информацию (*source*) в базе данных;
- *TextElements* — множество элементов с текстовым типом данных (PCDATA);
- *Tuple*(*relation*) — функция возвращает кортежи отношения (*relation*);
- *updateConnections*() — метод обновляет информацию о связях между кортежами отношений;
- *upload*(*file name*) — функция загружает файл (*file name*) в базу данных.

Перед началом преобразования XML-документа для каждого элемента и атрибута, содержащих данные в текстовом формате, значения сначала преобразуются к типу java.util.Date, в случае возникновения ошибки — к типу java.lang.Double, иначе — к типу java.lang.String.

Таблица 1. Правила преобразования XML-формата к реляционному

№	Описание правила
1	Обход дерева XML-документа происходит сверху вниз, в качестве текущего отношения определяется первый элемент, имя которого совпадает с именем отношения. S : Обход дерева XML-документа L : *currentElement.getRelation*() ∈ *Relations* $A \rightarrow B$: *currentRelation* = *currentElement.getRelation*() Q : *currentRelation* ∈ *Relations*
2	Элемент с данными в PCDATA-формате, имя которого совпадает с именем отношения, интерпретируется как DATA-атрибут отношения. S : анализ элемента (*currentElement*) L : *currentElement* ∈ *TextElement* $A \rightarrow B$: *currentRelation.getAttribute*("*DATA*").*setValue*(*currentElement.getValue*()) Q : *not*(*currentRelation.getAttribute*("*DATA*").*isEmpty*())
3	Элемент, являющийся потомком текущего элемента, с данными в PCDATA-формате, имя которого не совпадает с именем отношения, интерпретируется как атрибут текущего отношения (имя атрибута соответствует имени элемента). S : анализ потомков элемента (*currentElement*) L : *descendant* ∈ (*Descendants*(*currentElement*) ∩ *TextElements*) ∧ *not*(*exists*(*descendant.getRelation*())) $A \rightarrow B$: *currentRelation.getAttribute*(*descendant.getName*()) .*setValue*(*descendant.getValue*()) Q : *not*(*currentRelation.getAttribute*(*descendant.getName*()).*isEmpty*())
4	Элемент, имя которого совпадает с именем отношения и имеет атрибут *IsKey* со значением "*Yes*", вносится в отношение, если в отношении не существует аналогичного кортежа. Если такой кортеж существует, то будет сгенерировано сообщение об ошибке. Если атрибут *IsKey* имеет значение "*No*" и кортеж, соответствующий элементу, в отношении не присутствует, также будет сгенерировано сообщение об ошибке.
4.1	S : анализ элемента (*currentElement*) с атрибутом *IsKey* L : *IsKey* ∈ *Attributes*(*currentElement*) ∧ ((*currentElement.getAttribute*("*IsKey*").*getValue*()=="*Yes*") ∧ (*currentElement* ∉ *Tuple*(*currentRelation*))) $A \rightarrow B$: *currentRelation.store*(*currentElement*) Q : *currentElement* ∈ *Tuple*(*currentRelation*)
4.2	S : анализ элемента (*currentElement*) с атрибутом *IsKey* L : *IsKey* ∈ *Attributes*(*currentElement*) ∧ (((*currentElement.getAttribute*("*IsKey*").*getValue*()=="*Yes*") ∧ (*currentElement* ∈ *Tuple*(*currentRelation*))) ∨ ((*currentElement.getAttribute*("*IsKey*").*getValue*()=="*No*") ∧ (*currentElement* ∉ *Tuple*(*currentRelation*)))) $A \rightarrow B$: *ShowError*() Q : *currentElement* ∉ *Tuple*(*currentRelation*)
5	Элемент, имя которого совпадает с именем отношения и для которого не задан атрибут *IsKey*, вносится в таблицу. В случае существования кортежа происходит обновление связей отношения, иначе — в отношении создается новый кортеж.

Таблица 1. Правила преобразования XML-формата к реляционному

№	**Описание правила**
5.1	*S* : анализ элемента (*currentElement*) без атрибута *IsKey* *L* : *IsKey* ∉ *Attributes(currentElemen*t) ∧ (*currentElement* ∈ *Tuple*(*currentRelation*)) *A* → *B* : *currentRelation.updateConnections*() *Q* : *currentElement* ∈ *Tuple*(*currentRelation*)
5.2	*S* : анализ элемента (*currentElement*) без атрибута *IsKey* *L* : *IsKey* ∉ *Attributes(currentElement*) ∧ (*currentElement* ∉ *Tuple*(*currentRelation*)) *A* → *B* : *currentRelation.store*(*currentElement*) *Q* : *currentElement* ∈ *Tuple*(*currentRelation*)
6	Значение атрибута *SRC* элемента заносится в атрибут *SRC* отношения, а сам файл, имя которого содержится в атрибуте, загружается в атрибут *CNT* отношения в бинарном виде. *S* : анализ элемента (*currentElement*) с атрибутом *SRC* *L* : *SRC* ∈ *Attributes*(*currentElement*) *A* → *B* : (*currentRelation.getAttribute*("*CNT*") .*setValue*(*upload*(*currentElement.getAttribute*("*SRC*").*getValue*())), (*currentRelation.getAttribute*("*SRC*") .*setValue*(*currentElement.getAttribute*("*SRC*").*getValue*()) *Q* : *not*(*currentRelation.getAttribute*("*CNT*").*isEmpty*()) ∧ *not*(*currentRelation.getAttribute*("*SRC*").*isEmpty*())
7	Для элемента с именем *ATTACH* автоматически создается атрибут *CRC*32, хранящий значение контрольной суммы, вычисленной по алгоритму *CRC*32. Этот атрибут вносится в отношение *ATTACH*. Расчет контрольной суммы осуществляется с целью соблюдения уникальности данных и предотвращения возможности дублирования данных. *S* : анализ элемента (*currentElement*) с именем *ATTACH* *L* : *currentElement* == *ATTACH* ∧ *not*(*currentElement.getAttribute*("*CRC*32")*isEmpty*()) *A* → *B* : *currentRelation.getAttribute*("*CRC*32") .*set Value*(*checksumCRC*32(*currentElement.getAttribute*("*SRC*").getValue()) *Q* : *not*(*currentRelation.getAttribute*("*CRC*32")*isEmpty*())
8	Если имя атрибута некоторого элемента совпадает с именем отношения, то из отношения *TABLE_INDEX* с атрибутами *TABLE_NAME*, *INDEX_COLUMN*, *RETURN_COLUMN* выбирается кортеж, для которого его атрибут *TABLE_NAME* совпадает с именем атрибута элемента. После чего из отношения с именем, совпадающим с именем атрибута элемента, выбирается атрибут, имя которого содержится в атрибуте *RETURN_COLUMN* кортежа, для которого выполняется условие равенства *INDEX_COLUMN* значению атрибута элемента. *S* : анализ атрибутов элемента (*currentElement*) *L* : *attribute* ∈ *Attributes*(*currentElement*) ∧ *attribute.getRelation*() ∈ *Relation* ∧ *attribute.getValue*() == *TABLE_NAME.INDEX_COLUMN* *A* → *B* : *TABLE_NAME.INDEX_COLUMN* == *currentRelation.getAttribute*(*attribute*).*getValue*() *Q* : *currentRelation.getAttribute*(*attribute*).*getRelation*() ∈ *Relations*
9	Все атрибуты текущего отношения должны соотноситься с атрибутами элемента. Если атрибут элемента отсутствует в текущем отношении, то необходимо подняться вверх по иерархии XML-документа и найти отношение с нужным атрибутом. Поиск начнется с отношения, имя которого сформировано именами дочернего и родительского элементов с разделением через символ "_". *S* : анализ атрибутов элемента (*currentElement*)

Таблица 1. Правила преобразования XML-формата к реляционному

№	**Описание правила**
	L : $attribute \in Attributes(currentElement) \wedge$ $attribute \notin Attributes(currentRelation) \wedge$ $(\exists ancestor)\ (\exists relation) ancestor \in Ancestors\ (currentElement) \wedge$ $attribute \in Attributes(ancestor)$ $A \rightarrow B$: $relation = ancestor.getRelation()$ Q : $relation \in Relations$
10	Если существует отношение с именем, которое сформировано за счет имен родительского и дочернего элементов, разделенных символом подчеркивания ("_"), то к отношениям применяется связь "Многие ко многим". S : анализ дочернего и родительского элементов L : $(parentElement\_childElement).getRelation() \in Relations$ $A \rightarrow B$: $currentRelation = (parentElement\_childElement).getRelation()$ Q : $currentRelation \in Relations$
11	Если отношение для связи не было найдено, реализуется попытка применения связи "Один ко многим" с внесением значения атрибута ID из отношения с именем родительского элемента в отношение с именем дочернего элемента. В отношении с именем дочернего элемента должен находиться атрибут с именем родительского элемента. S : анализ дочернего и родительского элементов L : $parentElement \in Attributes(childElement) \wedge$ $childRelation = childElement.getRelation() \wedge$ $parentRelation = parentElement.getRelation()$ $A \rightarrow B$: $childRelation.getAttribute(parentElement)$ $.setValue(parentRelation.getAttribute("ID").getValue())$ Q : $childRelation.getAttribute(parentElement) \in Attributes(childRelation)$
12	Если не получается применить связь "Один ко многим", значение атрибута ID из отношения с именем дочернего элемента вносится в отношение с именем родительского элемента. В таблице с именем родительского элемента должен находиться атрибут с именем дочернего элемента. S : анализ дочернего и родительского элементов L : $childElement \in Attributes(parentElement) \wedge$ $childRelation = childElement.getRelation() \wedge$ $parentRelation = parentElement.getRelation()$ $A \rightarrow B$: $parentRelation.getAttribute(childElement)$ $.setValue(childElement.getAttribute("ID").getValue())$ Q : $parentRelation.getAttribute(childElement) \in Attributes(parentdRelation)$

В момент осуществления импорта информации в базу данных XML-элемент документа считается ключевым элементом, начиная с которого осуществляется анализ документа. Текущее отношение, отношение, стоящее выше в иерархии документа, атрибуты отношения и внешние ключи определяются на уровне отдельного элемента.

Экспорт хранимой в базе данных информации осуществляется на основе DTD-определения или XML-схемы, которые вместе с идентификаторами кортежей из отношения, соответствующего элементу XML-документа, устанавливаются в начале преобразования. Необходимые элементы экспортируются на основе текущего состояния базы данных. В результате формируются XML документы, соответствующие DTD-определению или XML-схеме. Правила, используемые при экспорте данных, представлены в таблице 2.

Таблица 2. Правила преобразования реляционных данных к XML-формату

№	**Описание правила**
1	Для заданного отношения и множества кортежей отношения определяется первый элемент, имя которого соответствует имени отношения. S : построение дерева XML-документа L : $relation \in Relations \wedge$ $Tuple(relation) \neq \varnothing \wedge$ $currentElement == relation$ $A \rightarrow B$: $currentElement.setRelation$(relation) Q : $currentElement.getRelation() \in Relations$
2	В рамках XML-документа осуществляется поиск дочернего элемента, имя которого совпадает с именем отношения. Для найденного отношения определяется множество кортежей, связанных с помощью атрибута или промежуточного отношения с кортежами отношения, соответствующего родительскому элементу. Промежуточное отношение должно иметь имя, образованное из имен родительского и дочернего элементов с разделением с помощью символа подчеркивания ("_"). S : анализ дочернего и родительского элементов L : $childElement \in Descendants(parentElement) \wedge$ $childElement \in Relations \wedge$ $parentElement == currentElement \wedge$ $exists(parentElement\_childElement)$ $A \rightarrow B$: $childElement = (parentElement\_childElement).getAttributue(parentElement) \wedge$ $parentElement = (parentElement\_childElement).getAttributue(childElement)$ Q : $(parentElement\_childElement).getRelation() \in Relations$
3	Атрибуты элемента, соответствующего отношению, данные в формате PCDATA, вложенные элементы, не совпадающие с именами отношений, их атрибуты и данные в формате PCDATA соотносятся с атрибутами отношения, которое соответствует текущему элементу, и отношения базы данных, имя которого состоит из имен текущего элемента и ближайшего родительского элемента с разделением символом подчеркивания ("_"). Все элементы и атрибуты включают данные. S : анализ атрибутов отношения и элемента, дочерних элементов и атрибутов L : $relation \in Relations \wedge$ $childElement.getRelation() \notin Relations \wedge$ $(Atributes(currentElement) \cap Atributes(childElement)) \subset Atributes(relation)$ $A \rightarrow B$: $(Atributes(currentElement) \cap Atributes(childElement)) = Atributes(relation)$ Q : $(Atributes(currentElement) \cap Atributes(childElement)) \neq \varnothing$
4	Если атрибут некоторого элемента совпадает с именем отношения, то из отношения *TABLE_INDEX*, имеющего атрибуты *TABLE_NAME*, *RETURN_COLUMN*, *INDEX_COLUMN* выбирается кортеж, для которого атрибут *TABLE_NAME* соответствует имени атрибута элемента. После чего из отношения с именем атрибута выбирается значение атрибута отношения, имя которого содержится в атрибуте *INDEX_COLUMN* кортежа, для которого выполняется условие равенства *RETURN_COLUMN* значению атрибута отношения. S : анализ атрибутов отношения ($relation$) L : $attribute \in Attributes(relation)$, $attribute.getRelation() \in Relations \wedge$ $attribute.getValue() == TABLE\_NAME.RETURN\_COLUMN$ $A \rightarrow B$: $currentElement.getAttribute(attribute).setValue(TABLE\_NAME.RETURN\_COLUMN)$ Q : $(Atributes(currentElement) \in Relations) \neq \varnothing$

Таблица 2. Правила преобразования реляционных данных к XML-формату

№	Описание правила
5	Атрибут *SRC* отношения вносится как атрибут *SRC* элемента. Бинарные данные из атрибута *CNT* отношения записываются в файл с именем из атрибута *SRC*. *S* : анализ отношения (*relation*) с атрибутом *SRC* *L* : *SRC* ∈ *Attributes*(*relation*) ∧ *not*(*relation.getAttribute*("*CNT*").*isEmpty*()) *A* → *B* : *currentElement.getAttribute*("*SRC*").*setValue*(*relation.getAttribute*("*SRC*").*getValue*()), (*download*(*relation.getAttribute*("*CNT*").*getValue*())) *Q* : *currentRelation.getAttribute*("*CNT*") ∈ *Attributes*(*currentRelation*)

Главная цель использования разработанных правил заключается не только в выполнении XML-реляционного преобразования данных, но и для поддержания логических связей и контроля соответствия иерархической и реляционной структур. Таким образом, благодаря правилам, появляется возможность отслеживать изменения, вносимые в одну из структур, с целью их репликации в автоматическом режиме в другую структуру.

4. ОПИСАНИЕ ПРИМЕРА ИСПОЛЬЗОВАНИЯ СИСТЕМЫ ПРОДУКЦИОННЫХ ПРАВИЛ

Рассмотрим пример использования разработанной системы продукционных правил для передачи данных о виртуальных лабораториях [9–12]. DTD-определение XML-документа, содержащего описание виртуальной лаборатории, приведено ниже:

```
<!ELEMENT LaboratoryTools (ToolIndex*)>
<!ELEMENT ToolIndex (LaboratoryTool)>
<!ATTLIST ToolIndex
  Scheme CDATA \#REQUIRED
  ToolType CDATA \#FIXED "Laboratory"
  ToolID CDATA \#REQUIRED
    IsKey CDATA \#FIXED "Yes"
>
<!ELEMENT LaboratoryTool (Attach*)>
<!ATTLIST LaboratoryTool
  Name CDATA \#REQUIRED
  Code CDATA \#REQUIRED
  Width CDATA \#REQUIRED
  Height CDATA \#REQUIRED
  SRC CDATA \#IMPLIED
  Language CDATA "ru"
>
<!ELEMENT Attach EMPTY>
<!ATTLIST
  SRC CDATA \#REQUIRED
>
```

Корневой элемент XML-документа — LaboratoryTools — не содержит атрибутов и может содержать любое количество элементов ToolIndex, идентифицирующих виртуальную лабораторию в системе. При этом виртуальная лаборатория может быть разделе-

на на несколько административных схем. Атрибуты ToolIndex содержат следующие данные:

- Scheme — идентификатор административной схемы;
- ToolType — тип установки виртуальной лаборатории;
- ToolID — уникальный номер установки в рамках схемы.

Таким образом, триплет значений (Scheme, ToolType, ToolID) представляет собой составной ключ для установки, который должен быть уникальным. Поскольку атрибут IsKey содержит значение по умолчанию “Yes”, указанное условие всегда должно выполняться. В соответствии с DTD-определением каждый элемент ToolIndex может включать только один элемент LaboratoryTool, дочерние элементы которого описывают установку виртуальной лаборатории:

- Name — название установки;
- (Width, Height, Code) — параметры установки для отображения ее на странице обозревателя;
- SRC — ссылка на архив с программным кодом установки для загрузки.

В случае, если установка виртуальной лаборатории содержит ссылки на внешние файлы, каждый из этих файлов должен быть указан в атрибуте SRC элементов Attach, включенных в элемент LaboratoryTool. Триплет значений (ToolIndex, ToolID, Scheme) корневого элемента должен быть уникальным для однозначного определения виртуальной лаборатории в системе. Объектная модель XML-документа, построенная на основе указанного DTD-определения, представлена на рис. 1. Реляционная структура данных, которая так же соотносится с DTD-определением и построена на основе правил импорта иерархических структур в реляционную базу данных, представлена на рис. 2.

В ходе импорта документа в базу данных происходит анализ дерева XML-документа. Определяется первый элемент ToolIndex, имя которого совпадает с именем отношения. Атрибут IsKey установлен как “Yes”, следовательно, триплет значений (Scheme, ToolType, ToolID) включается в отношение ToolIndex, если такой триплет не был представлен в отношении ранее (Правило № 4). Следующим элементом документа является LaboratoryTool. Триплет значений (Scheme, ToolType, ToolID) отношения ToolIndex включается в атрибут ToolIndex отношения LaboratoryTool (Правило №11). Атрибут Language представляет собой идентификатор языка из соответствующего отношения (Правило № 8). Другие атрибуты элемента содержат значения остальных атрибутов отношения (Правило № 5). Атрибут CNT заполняется данными о файле, ссылка на который указана в атрибуте SRC (Правило № 6). Данные из элементов Attach включаются в соответствующее отношение (Правило № 7). Отношение LaboratoryTool_Attach отвечает за установку отношения «Многие-ко-Многим» между кортежами отношений LaboratoryTool и Attach (Правило № 10).

5. ЗАКЛЮЧЕНИЕ

Современные информационные системы часто используют реляционную модель данных для эффективного хранения и манипулирования данными. В то же время, для передачи данных чаще используется формат XML, являющийся более удобным и доступным для понимания. Нетривиальность задачи преобразования иерархической модели, лежащей в основе формата XML, к реляционной обуславливает необходимость

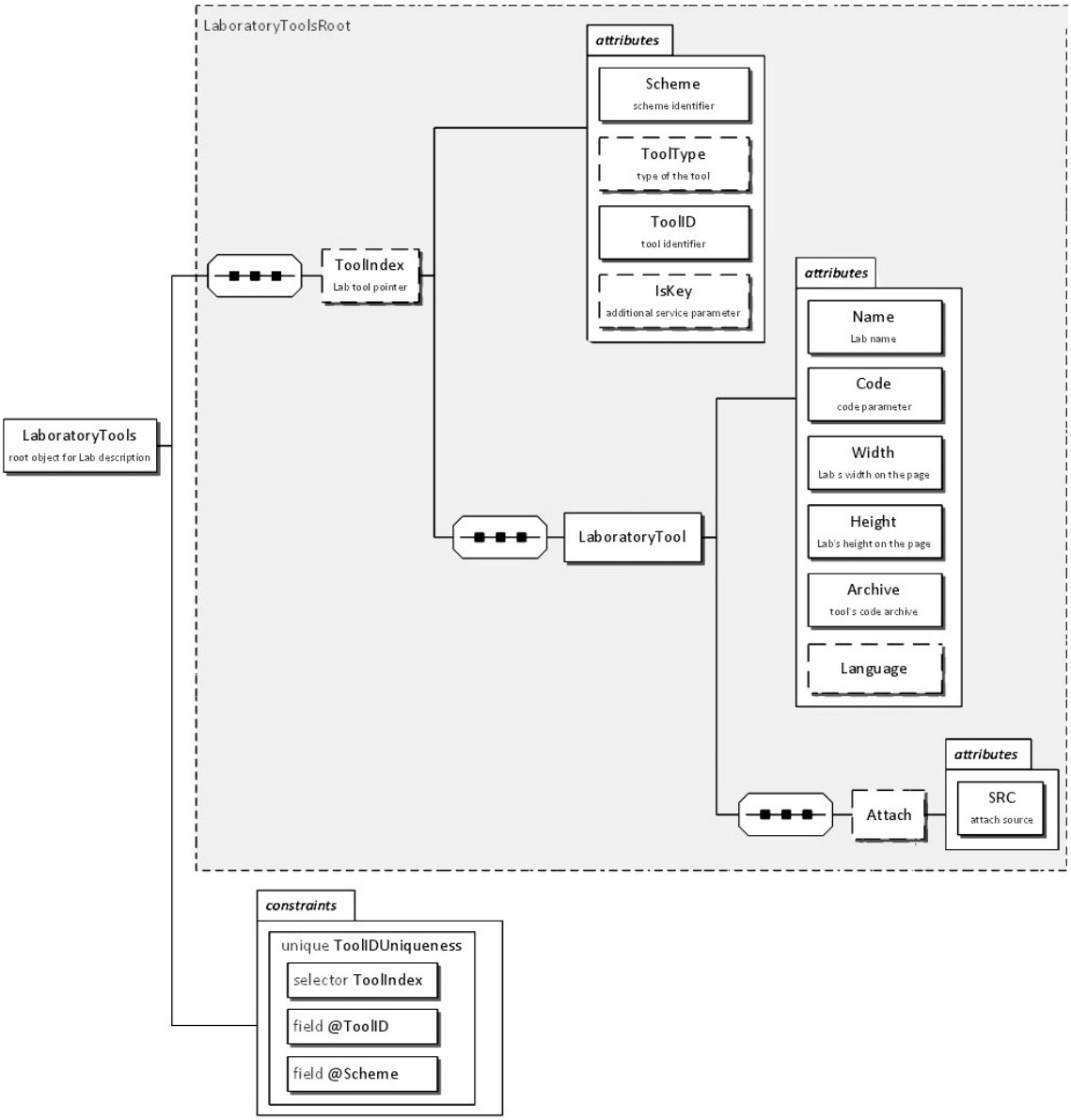

Рис. 1. Объектная модель документа

разработки новых методов, осуществляющих прямое и обратное преобразования двух моделей данных.

В статье описан разработанный метод преобразования иерархической и реляционной моделей данных, основанный на системе продукционных правил. Предложенный метод позволяет избежать избыточности данных и обеспечивает полный анализ структуры данных системы. Разработанные правила позволяют поддерживать в актуальном состоянии логические связи и осуществлять контроль соответствия иерархической и реляционной структур. Разработанная библиотека, содержащая программное представление всех разработанных правил, успешно внедрена и используется в электронной информационно-образовательной среде Университета ИТМО для импорта и экспорта данных об электронных учебно-методических материалах.

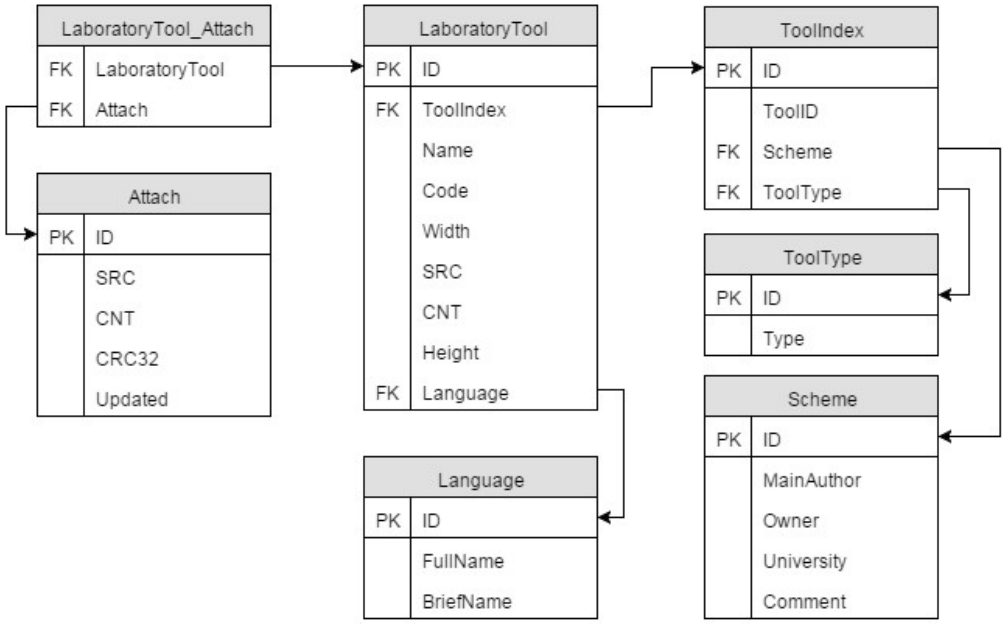

Рис. 2. Реляционная структура данных виртуальной лаборатории

Список литературы

1. *Elmadany H.A., Alfonse M., Aref M.* XML summarization: A survey // Proceedings of the 2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS). 2015. P. 537–541.
2. *Sayih M., Conrads M., Brüggemann-Klein A.* Multi-client XML web applications // 2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS). 2017. P. 8–12.
3. *Vighio M. S., Khanzada T. J., Kumar M.* Analysis of the effects of redundancy on the performance of relational database systems // 2017 IEEE 3rd International Conference on Engineering Technologies and Social Sciences (ICETSS). 2017. P. 1–5.
4. *Ahmad K., Samad R.* Semantic Based Mapping from XML to Relations // Proceedings of the 2012 8th International Conference on Information Science and Digital Content Technology (ICIDT). 2012. Vol. 2. P. 255–260.
5. *Bhargavi K., Chaithra H. S.* Join queries translation from SQL to XPath // 2013 IEEE International Conference ON Emerging Trends in Computing, Communication and Nanotechnology (ICECCN). 2013. P. 349–364.
6. *Qtaish A., Ahmad K.* Query mapping techniques for XML documents: A comparative study // 2015 International Conference on Electrical Engineering and Informatics (ICEEI). 2015. P. 529–534.
7. *Kahloula B., Bouamrane K.* Using a Mapping Knowledge Base in a System for (Semi-) Automatic Loading of XML Data into Relational Databases // Proceeding of the 2013 World Congress on Computer and Information Technology (WCCIT). 2013. P. 1–7.
8. *Лямин А. В., Чежин М. С.* Развитие электронного обучения, дистанционных образовательных технологий в НИУ ИТМО // Информационная среда вуза XXI века: материалы VII Всероссийской научно-практической конференции. 2013. С. 145–148.
9. *Ефимчик Е. А., Лямин А. В.* Определение сложности вариантов алгоритмических заданий для виртуальных лабораторий на основе автоматной модели // Компьютерные инструменты в образовании. 2014. №. 3. С. 34–43.
10. *Ефимчик Е. А.* Автоматическое построение заданий для проверки навыков выполнения операций над множествами // Новые информационные технологии в образовании: материалы междунар. науч.-практ. конф. 2013. С. 278–281.
11. *Djeghloud H., Larakeb M., Bentounsi A.* Virtual Labs of Conventional Electric Machines // Intern. Conf. on Interactive Mobile and Computer Aided Learning (IMCL). 2012. P. 52–57.

12. *Hristov G., Zahariev P., Bencheva N., Ivanov I.* Designing the Next Generation of Virtual Learning Environments – Virtual Laboratory with Remote Access to Real Telecommunication Devices // Proc. of the 24th EAEEIE Annual Conf. (EAEEIE). 2013. P. 139–144.

Поступила в редакцию 09.01.2018, окончательный вариант — 22.02.2018.

Computer tools in education, 2018
№ 1: 51–64
http://ipo.spb.ru/journal

PRODUCTION RULE SYSTEM FOR XML-RELATIONAL MAPPING

Lyamin A. V.[1]

[1]Saint-Petersburg National Research University of Information Technologies, Mechanics and Optics, Saint Petersburg, Russia

Abstract

Production rule system is an artificial intelligence method that can be used for XML-relational mapping, which is a nontrivial task. The purpose of the presented study is to develop a production rule system that could be successfully applied for unique mapping XML documents to relational model and back. Analysis of different XML-relational mapping methods used in modern informational systems had been conducted. A unique method of mapping relational data structures to hierarchical ones had been developed based on the production rule system. A mapping example of XML document with information about virtual labs had been described.

Keywords: *Information Systems, XML-Relational mapping, Production Rule System.*

Citation: A. V. Lyamin. "Production Rule System for XML-Relational Mapping," *Computer tools in education*, no 1, pp. 51–64, 2018 (in Russian).

Received 09.01.2018, the final version — 22.02.2018.

Andrey V. Lyamin, PhD, Tech., Associate Professor, Saint-Petersburg National Research University of Information Technologies, Mechanics and Optics; 197101, Russian Federation, Saint-Petersburg, Kronverkskiy pr., 49 A, 457, lyamin@mail.ifmo.ru

Лямин Андрей Владимирович, кандидат технических наук, доцент Санкт-Петербургского национального исследовательского университета информационных технологий, механики и оптики; 197101, Санкт-Петербург, пр. Кронверкский, д. 49, лит. А, 457, lyamin@mail.ifmo.ru