Виды ограничений целостности в базах XML-данных

М.П. Рекуц

Аннотация. Исследовательская группа MODIS (www.modis.ispras.ru) ИСП РАН на протяжении последних двух лет работает над проектом по созданию полнофункциональной системы управления XML-данными (XML-СУБД) Sedna[1]. Одним из основных требований к любой полнофункциональной СУБД является полноценная поддержка ограничений целостности. В этой статье на основе анализа потребностей современных приложений, работающих с XML-СУБД, выявляются виды ограничений целостности, которые должны поддерживаться XML-СУБД, и предлагаются средства определения этих видов ограничений с учетом специфики XML-модели данных и опыта, накопленного разработчиками реляционных СУБД.

1. Введение

За последние несколько лет XML [1] стал если не повсеместным, то, по крайней мере, весьма популярным форматом данных, что привело к тому, что XML-СУБД все чаще становится ключевым компонентом в сложных решениях. Это, в свою очередь, требует от XML-СУБД широкого набора функциональности, эквивалентного тому, что мы привыкли ожидать от традиционных СУБД. Одним из примеров такой функциональности является **развитая поддержка ограничений целостности** – важнейший элемент СУБД, без которого, в частности, невозможна поддержка полноценного механизма транзакций.

В этой работе мы ставим перед собой цель проанализировать, какие ограничения целостности могут и должны поддерживаться полнофункциональными XML-СУБД, определить виды ограничений целостности, исходя из специфики XML-модели данных, потребностей современных приложений, работающих с XML-данными, и учитывая опыт, накопленный разработчиками реляционных СУБД.

Для начала рассмотрим проблему на примере. Приложение работает с данными некоторого университета.

Пусть имеется следующее описание данных: данные университета представлены в формате XML и включают информацию о студентах университета и курсах лекций, проводящихся в университете. Данные для каждого студента

[1] Проект поддержан грантом РФФИ №05-07-90204

включают адрес, возраст студента, его имя, номер студенческого билета и любое количество курсов, которые он посещает (номера курсов). Данные для каждого курса лекций включают номер курса и его название. Возраст студентов не должен превышать 30 лет. Номера студенческих билетов и номера курсов уникальны. Для каждого номера курса лекции, содержащегося в данных студента, должны существовать данные о курсе лекций с таким номером.

На рис. 1 представлена структура XML-данных университета.

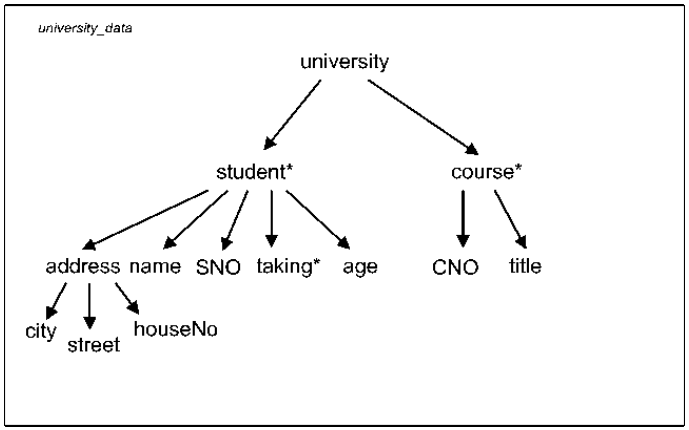

Рис.1. Структура XML-данных университета.

Для корневого элемента *university* может иметься ноль или более элементов student и course. Элемент student содержит информацию о студенте: адрес студента (в элементе address), содержащий подэлементы с названием города, улицы и номером дома (city, street, houseNo), имя студента (в элементе name), уникальный номер студенческого билета (в элементе SNO), возраст студента (в элементе age), курсы лекций, посещаемые студентом (ноль или более элементов taking). Элемент course содержит информацию о курсе лекций: название курса (в элементе title), уникальный номер курса (CNO).

Ниже приводится пример XML-документа “university_data”, обладающего описанной структурой.

```
<?xml version="1.0"  encoding="UTF-8" ?>
<university>
  <student>
       <address>
            <city>Moscow</city>
            <street>Kommunisticheskaya</street>
            <houseNo>25</houseNo>
       </address>
       <SNO>19283764</SNO>
       <name>Kate</name>
       <taking>120</taking>
       <taking>123</taking>
```

```
        <age>23</age>
    </student>
    <student>
        <address>
            <city>Moscow</city>
            <street>Kommunisticheskaya</street>
            <houseNo>1</houseNo>
        </address>
        <SNO>19283723</SNO>
        <name>Alex</name>
        <taking>123</taking>
        <age>23</age>
    </student>

    <course>
        <CNO>123</CNO>
        <title>Database Systems</title>
    </course>
    <course>
        <CNO>120</CNO>
        <title>Artificial Intelligence</title>
    </course>

</university>
```

Приложению, работающему с этими XML-данными, время от времени необходимо проводить модификации данных. Рассмотрим несколько примеров выражений модификации и обратим внимание на то, какие проблемы, связанные с нарушением изначального описания, могут возникнуть при таких модификациях.

Для наглядности операции модификации мы будем описывать на языке модификации XML-данных XUpdate, поддерживаемом в XML-СУБД Sedna[13]. Язык XUpdate является расширением языка запросов XML-данных XQuery[17] операциями модификации XML-данных. Основные конструкции языка XUpdate, в частности, выражения, адресующие узел XML-документа, базируются на конструкциях языка XQuery.

Следующее выражение модификации XML-данных добавляет информацию об адресе студента с именем Kate:

```
        UPDATE
  (1) insert <address>Moscow, Ostankinskaya st., 8</address>
      into
document("university_data")/university/student[name="Kate"]
```

Согласно описанной выше структуре XML-данных элемент *address* студента состоит из подэлементов *city*, *street* и *houseNo*, содержащих называние города, улицы и номер дома соответственно. Выражение модификации (1) всю информацию об адресе вставляет в элемент address. После проведения рассмотренной операции модификации данные университета не соответствуют структуре,

показанной на рис 1. С точки зрения приложения, работающего с XML-данными заранее определенной структуры, эта операция модификации данных является некорректной, так как после ее проведения XML-данные перестают соответствовать этой структуре. Таким образом, описанную проблему логично рассмотреть как требование приложения к системе управления XML-данными поддерживать XML-данные в соответствии с заранее определенной структурой.

Рассмотрим следующую операцию модификации данных. Студенту с именем Kate добавляется номер курса лекции, посещаемый этим студентом.

```
        UPDATE
(2)   insert <taking>212</taking>
        into
document("university_data")/university/student[name="Kate"]
```

Данная операция модификации добавляет к данным студента номер курса лекций, посещаемого студентом, информация о котором не содержится в данных. То есть, нет данных о курсе лекций с номером 212, что приводит к противоречию с изначальным описанием данных, где сказано, что для каждого номера курса лекции, содержащегося в данных студента, должны существовать данные о курсе лекций с таким номером.

Следующая операция модификации данных добавляет данные о курсе лекций с номером, который уже встречается в данных.

```
       UPDATE
(3) insert <course>
                <CNO>123</CNO>
                <title>Computer Networks</title>
           </course>
    into document("university_data")/university
```

После проведения описанной операции модификации данные содержат информацию о двух курсах лекций с одинаковыми номерами, что противоречит изначальному описанию данных, в которых сказано, что номера курсов лекций уникальны.

Проблемы, возникшие при выполнении операций модификации (2) и (3) имеют непосредственный аналог в реляционных СУБД, в литературе часто называются ограничениями целостности по ссылкам. В разделе 5 мы рассмотрим эти проблемы формально в контексте XML-СУБД.

Рассмотрим следующий пример операции модификации данных.

```
        UPDATE
(4)   REPLACE //student[name="Kate"]/age AS $x
        WITH <age>{$x+10}</age>
```

Данная операция модификации увеличивает возраст студента с именем Kate на 10. Таким образом, после проведения этой операции возраст будет равен 33, что противоречит исходному описанию данных, ограничивающему возраст студентов 30 годами.

В этом примере мы столкнулись с проблемой, для которой также имеется аналог в реляционных СУБД. Согласно [16], аналогом ограничения на возраст студентов из нашего примера в реляционных СУБД является ограничение на атрибут, а аналогичная операция модификации, в результате которой значение становится больше 30, нарушает это ограничение. Несмотря на аналогии, в контексте XML-СУБД эта проблема требует пересмотра. В разделе 5 мы рассмотрим ее формально в контексте XML-СУБД.

Итак, мы рассмотрели пример приложения, работающего с XML-данными, и несколько выражений модификации этих данных, выполняемых приложением. После выполнения этих операций модификации данные перестают удовлетворять своему исходному описанию. Естественно предположить, что решение таких проблем должна предоставлять система управления XML-данными, используемая приложением. Как правило, такие требования выдвигаются к СУБД вне зависимости от того, на какой модели данных основана эта СУБД. Таким образом, следующей нашей целью в этой работе является формальное определение видов ограничений целостности, которые должны поддерживаться современной XML-СУБД.

2. Структурные ограничения целостности

В этом разделе мы рассмотрим первый вид ограничений целостности, который назовем ***структурными ограничениями целостности.*** Вернемся к первой операции модификации из примера. Эта операция добавляет элемент address в качестве потомка элемента student, что делает XML-данные не соответствующими структуре, изображенной на рис 1. Таким образом, эта структура является ограничением, накладываемым на данные.

В контексте XML структура, которой должны обладать XML-данные, называется *схемой XML-данных*. В отличие от HTML, являющегося конкретным языком, синтаксис которого включает фиксированный предопределенный стандартом языка набор тегов разметки, XML представляет собой расширяемый язык, с использованием которого можно создавать различные языки, т.е. *метаязык*. При этом XML-данные являются самоописываемыми благодаря использованию тегов. С другой стороны, важнейшее свойство XML – *расширяемость*, обозначенное в названии языка, подразумевает наличие механизмов определения схемы XML-данных. То есть XML-данные могут поступать как самостоятельно, так и с заранее определенной схемой, которая может использоваться как для адекватной интерпретации самих данных, так и для проверки правильности структуры данных.

Таким образом, в XML-СУБД определение схемы, которой должны удовлетворять XML-данные (говоря о XML-данных, будем иметь в виду XML-документы), является определением структурного ограничения целостности для этих данных.

На этом этапе наших рассуждений возникают следующие вопросы: каким образом можно определить схему XML-данных? И что значит то утверждение,

что XML-данные не нарушают структурное ограничение целостности, определенное в СУБД для этих данных?

Перейдем к обсуждению первого вопроса. В настоящее время существует около десятка языков описания схем XML-данных [8, 9], поддерживаемых различными организациями. Наиболее популярными из них являются языки DTD [1], XML Schema [3] (предоставлены консорциумом W3C), RELAX (разработан Макото Мурата, в настоящее время проводится стандартизация в JSA (Japanese Standard Association)) [19], XDR (Microsoft)[9], SOX(Schema for Object-Oriented XML) [9]. Схемы XML-данных (как, впрочем, и схемы любой модели данных) предназначены для описания структурных и семантических ограничений, которые должны выполняться на любом наборе данных, удовлетворяющем данной схеме. Характерными примерами структурного ограничения являются спецификации содержания элементов (например, элемент с именем A может содержать только элементы с именем B). Характерными примерами семантических ограничений являются спецификации ключей (уникальных атрибутов). Несмотря на то, что языки описания схем достаточно сильно различаются по выразительной мощности, многие принципы построения ограничений схожи и основаны на использовании регулярных выражений. В большей степени это касается структурных ограничений. Поэтому многие свойства структурных ограничений в рамках одного языка имеют свои аналоги в других языках.

Исходя из этих рассуждений, структурные ограничения целостности в XML-СУБД разумно описывать при помощи одного из языков описания схем XML-данных. На сегодняшний день существует множество работ по сравнению языков описания XML-схем. В этих работах представляется глубокий анализ языков и предлагаются различные критерии для их сравнения [7, 8]. В частности, сравнительный анализ проводится путем сопоставления языков описания схем регулярным грамматикам [7].

В нашей работе, опираясь на результаты [7, 8, 9], мы предлагаем в качестве языка описания структурных ограничений целостности для XML-СУБД использовать XML Schema [3] по следующим причинам:

1) XML Schema является гораздо более мощным языком по сравнению с очень популярным на первых этапах развития XML языком XML DTD;
2) Язык обладает механизмами, облегчающими спецификацию схем: определение сложных типов, определение анонимных типов, определение групп, определение подтипов путем расширения и ограничения, группы подстановок, определение абстрактных типов;
3) XML Schema в настоящее время успешно заменяет DTD;
4) XML Schema является стандартом W3C [2];
5) В XML Schema для описания схем используется синтаксис XML, что облегчает обработку ограничений целостности, позволяя пользоваться развитым инструментарием для обработки XML.

Если вернуться к нашему примеру, структурное ограничение целостности, то есть определение структуры данных из рис.1, выраженное на языке описания XML-схем, выглядит следующим образом:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns="http://www.w3.org/2001/XMLSchema">
 <xs:element name="university">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="student" minOccurs="0" maxOccurs="unbounded">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="address">
        <xs:complexType>
         <xs:sequence>
          <xs:element name="city" type="string"/>
           <xs:element name="street" type="string"/>
            <xs:element name="houseNo" type="integer"/>
             </xs:sequence>
              </xs:complexType>
              </xs:element>
              <xs:element name="name" type="string"/>
              <xs:element name="taking" minOccurs="0"
                    maxOccurs="unbounded" type="integer"/>
              <xs:element name="age" type="integer"/>
             </xs:sequence>
           </xs:complexType>
          </xs:element>
        <xs:element name="course" minOccurs="0"
maxOccurs="unbounded">
           <xs:complexType>
             <xs:sequence>
                <xs:element name="CNO" type="integer"/>
                <xs:element name="title" type="string"/>
             </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
</xs:schema>
```

Продолжая наши рассуждения, ответим на второй вопрос. XML-данные не нарушают структурного ограничения целостности, которое определено в СУБД для этих данных, если они являются валидными по отношению к XML-схеме, представляющей собой данное структурное ограничение целостности. XML-схема специфицирована на некотором языке описания XML-схем (в нашем случае – XML Schema). Иначе говоря, проверка структурного ограничения целостности является *задачей валидации* (или *верификации*) данных [7].

Валидация XML-документа относительно заданной схемы во многих случаях подразумевает достаточно нетривиальный процесс; понятно, что проверка структурных ограничений целостности при операциях модификации данных может обернуться неприемлемо дорогим процессом для XML-СУБД. Необходима оптимизация процесса валидации, состоящая, возможно, в валидации только фрагмента XML-документа, который подвергается модификации.

Некоторые проблемы, касающиеся реализации предлагаемых в данной работе видов ограничений целостности, требуют тщательного исследования и проработки. Оптимизация процесса валидации при проверке структурных ограничений целостности в XML-СУБД входит в их число. Об этих проблемах мы поговорим в разделе 5.

3. Ограничения целостности по ссылкам

Перейдем к следующему виду ограничений целостности; для этого вернемся к выражению модификации данных (2) из раздела 1. Данное выражение модификации нарушает следующее начальное условие: для каждого номера курса лекций, содержащегося в данных студента, должны существовать данные о курсе лекций с таким номером. Такое условие является типичным примером ***ограничения целостности по ссылкам***.

Ограничения целостности по ссылке широко используются в реляционных СУБД и впервые были определены Э. Коддом. Чтобы сформулировать ограничение целостности по ссылке, используем определение ключа, данное в [16]:

Ключ – совокупность элементов данных, являющихся компонентами экземпляров данных иного вида, которая идентифицирует экземпляры данных или определяет их свойство, используемое в некоторых операциях управления данными и/или их обработки. *Первичный ключ* – ключ, каждому значению которого может соответствовать не более одного экземпляра данных.

Сформулируем ***ограничение целостности по ссылкам*** для XML-данных следующим образом: элемент с некоторыми значениями атрибутов и подэлементов (дочерних элементов) может быть включен в XML-документ тогда и только тогда, когда эти значения являются актуальными значениями первичного ключа, указанного для другого элемента.

В добавление к введенным определениям, мы выделим еще один вид ограничений целостности – ***уникальность данных***.

Уникальность данных – ограничение целостности, определяемое на некотором наборе элементов, требующее уникальности каждого элемента из этого набора по значению некоторого подэлемента или атрибута.

В нашем примере операция модификации (3) демонстрирует нарушение уникальности данных.

XML предоставляет механизм для поддержки уникальности атрибутов и ссылок на атрибуты через использование типов атрибутов *ID*, *IDREF*, *IDREFS*. Этот же механизм предлагает и язык XML DTD для описания ссылок в XML-

документах. Однако в нем имеется ряд известных недостатков: уникальностью (типом ID) может обладать только атрибут, объявление уникальности атрибута глобально в пределах документа. Язык XML Schema, выбранный нами в качестве средства для описания структурных ограничений целостности в предыдущем разделе, предоставляет механизм описания ссылок в XML-документе, избегающий недостатков механизма XML DTD. Это является еще одним плюсом в пользу выбора языка XML Schema и дает нам повод использовать его и в качестве средства описания ограничений целостности по ссылке в XML-СУБД.

Итак, рассмотрим механизмы XML Schema для определения уникальности данных и ссылок в XML-документах. Уникальность данных определяется при помощи элемента *xs:unique*. Местоположение элемента *xs:unique* в схеме определяет контекстный узел дерева, относительно которого должна поддерживаться уникальность. Таким образом, мы имеем возможность определять это ограничение локально. Элемент *xs:selector* при помощи XPath определяет элемент, который имеет уникальное значение. Выражение XPath в элементе *xs:selector* вычисляется относительно контекстного узла. *xs:field* определяет элемент или атрибут, значение которого будет проверяться на уникальность. Выражение XPath, содержащееся в элементе *xs:field,* определяет путь к этому элементу относительно элемента, определенного в *xs:selector*.

Конструкция *xs:key* для определения ключей схожа с *xs:unique*, но имеет дополнительное требование: значение, которое определяется как уникальное, должно обязательно существовать.

Конструкция *xs:keyref* позволяет определять ссылку на *xs:key* и *xs:unique*. Для иллюстрации перечисленных конструкций специфицируем ограничение целостности по ссылке и уникальность данных для нашего примера.

```
<xs:key name="key_constraint">
  <xs:selector xpath="student"/>
  <xs:field xpath="SNO"/>
</xs:key>
<xs:unique name="unique_constraint">
  <xs:selector xpath="course"/>
  <xs:field xpath="CNO"/>
</xs:unique>
<xs:keyref name="keyref_constraint" refer="key_constraint">
  <xs:selector xpath="student"/>
  <xs:field xpath="taking"/>
</xs:keyref>
```

4. Ограничения целостности, описываемые произвольным выражением

Перейдем к последнему и наиболее общему виду ограничений целостности – ***ограничения целостности, описываемые произвольным выражением***.

Рассмотрим операцию модификации (4) из нашего примера. Данная операция модификации нарушает ограничение: «Возраст студентов не должен превышать 30 лет», иначе говоря, представляет собой выражение, значение которого «истина», если возраст студента меньше 30, и «ложь», если возраст студента больше 30.

Таким образом, ограничение целостности, описываемое выражением, устанавливается путем определения некоторого выражения (правила – *integrity rule*), значением которого является «истина» или «ложь».

Данная формулировка принадлежит К. Дейту, предлагается им в “An Introduction to Database Systems” [17] и, таким образом, не является специфичной для XML-СУБД.

Для того, чтобы определить данный вид ограничений целостности для XML-СУБД, необходимо выбрать средство для задания выражения. В качестве этого средства мы предлагаем использовать язык запросов к XML-данным XQuery [5]. Это развитый функциональный язык, позволяющий формировать несколько видов выражений, которые могут использоваться в разных сочетаниях. Язык базируется на системе типов XML Schema и совместим с другими стандартами, связанными с XML. В последнее время он часто рассматривается исследователями как язык общего назначения.

Итак, на языке XQuery формируется выражение, используемое для ограничения целостности. Для нашего примера оно выглядит так:

```
document(“university_data”)/university/student/age < 30
```

Следует заметить, что данное ограничение можно выразить также и средствами XML Schema. Однако в большинстве случаев XML Schema недостаточно, так как средства XML Schema в основном предназначены для определения структурных ограничений, а не семантических. Например, ограничение целостности, имеющее агрегатный характер, более сильно демонстрирует необходимость использования XQuery.

С точки зрения реализации имеет смысл разделять такие ограничения целостности на *динамически проверяемые* и *статически проверяемые*. При операции модификации данных всегда предпочтительно до ее выполнения знать, нарушает ли операция ограничение целостности, чем провести модификацию данных и потом проверить, не нарушила ли она ограничение. Однако это не всегда возможно, и в таких случаях в выражении XQuery необходимо выделять подвыражение, которое можно проверить статически и, если это не дает нужной информации, проводить динамическую проверку при выполнении операции модификации.

В заключение этого раздела рассмотрим возможную форму задания ограничений целостности, описываемых выражением (в соответствии с предложениями К. Дейта).

CREATE INTEGRITY RULE <название>

<XQuery выражение, значение которого «истина» или «ложь» >
ON ATTEMPTED VIOLATION <процедура при нарушении>

DESTROY INTEGRITY RULE <название>.

Оператор CREATE INTEGRITY RULE содержит название правила, под которым оно будет зарегистрировано в системном каталоге; XQuery-выражение; процедуру, выполняемую СУБД при нарушении ограничения. Например, можно отказать пользователю в выполнении операции модификации, спровоцировавшей нарушение ограничения целостности (на практике такая реакция на нарушение, вероятно, будет наиболее частая, и имеет смысл использовать ее по умолчанию).

5. Обзор близких работ и направления дальнейшего исследования

На момент написания данной работы ни одна из известных автору существующих XML-СУБД (мы имеем в виду *прирожденные XML-СУБД* – системы, специально спроектированные и разработанные для управления XML - данными) не предоставляет полноценной поддержки ограничений целостности.

Существует ряд работ по исследованию проблем, в той или иной степени касающихся ограничений для XML; на наш взгляд, в основном они фокусируются на частных проблемах. Автору неизвестны работы, представляющие полною постановку задачи поддержки ограничений целостности в XML-СУБД.

Стоит обратить внимание на работы В.Фана (Wenfei Fan) с разными соавторами, посвященные проблемам правильности (consistency) схем XML-данных, рассматриваемых в качестве ограничений целостности [10, 11, 12, 13].

Задача правильности схемы XML-данных формулируется следующим образом: существуют ли для заданной схемы XML-данные, удовлетворяющие ограничениям, поставленным в схеме? В контексте XML-СУБД задачу можно сформулировать: существуют ли XML-данные, удовлетворяющие структурному ограничению целостности или ограничению целостности по ссылке, определенному данной XML-схемой?

В работах В. Фана проблемы правильности рассматриваются в контексте современных языков описания схем, в частности, рассматривается XML Schema. Показано, что существуют XML-схемы, которым не удовлетворяет ни один XML-документ (не являющиеся правильными). Авторы, используя формальные методы, пытаются решить в общем случае проблему проверки правильности XML-схемы и приходят к выводу, что в общем случае такая задача не имеет решения, а для предложенного в работе класса упрощенных XML-схем задача является NP-сложной. Также предлагаются языковые конструкции для описания ограничений целостности по ссылкам в XML-данных [13], не подкрепленные никакими рекомендациями по реализации. По

нашему мнению, эти конструкции неприемлемы на практике ввиду своей чрезмерной сложности. Как заключают сами авторы, для полного понимания эффективности предлагаемого ими механизма требуется некий прототип системы, реализующий этот механизм.

Проблемы, рассматриваемые в работах В.Фана, обоснованы, и в отдельных приложениях могут играть значительную роль. Однако на данный момент мы затрудняемся оценить, насколько они актуальны в контексте XML-СУБД. Для этого требуются дополнительные исследования.

Требуется также исследование проблем, касающихся эффективной реализации предложенных видов ограничений целостности: быстрая валидация, выделение подвыражений из XQuery-выражений для статической проверки.

6. Заключение

В данной работе на основе анализа требований современных приложений к XML-СУБД выявляются виды ограничений целостности для XML-СУБД: структурные ограничения целостности, ограничения целостности по ссылке и ограничения целостности, описываемые произвольным выражением. Предлагаются модельно-языковые средства – XML Schema[3] и XQuery[5] для определения этих видов ограничений целостности.

Литература.

1. Extensible Markup Language 1.0 (Second Edition), W3C Recommendation (4 February 2004), *http://www.w3.org/TR/REC-xml* .
2. World Wide Web Consortium, *http://www.w3.org*.
3. Schema, Parts 0, 1, and 2, W3C Recommendation (28 October 2004), *http://www.w3.org/TR/xmlschema-0*, *http://www.w3.org/TR/xmlschema-1* and *http://www.w3.org/TR/xmlschema-2*.
4. Path Language Version 1.0, W3C Recommendation (1999 16 November), *http://www.w3.org/TR/xpath*.
5. XQuery 1.0 and XPath 2.0 Data Model, W3C Working Draft (29 October 2004), *http://www.w3.org/TR/query-datamodel*.
6. XQuery 1.0 Formal Semantics, W3C Working Draft (20 February 2004), *http://www.w3.org/TR/query-semantics*.
7. Рекуц Мария, дипломная работа «Сравнительный анализ средств описания схем документов. Разработка метода валидации XML-данных», МГУ им. Ломоносова. 2003г.
8. Dongwon Lee, Murali Mani, Makoto Murata “Reasoning about XML Schema languages using Formal Language Theory”, Technical Report, IBM Almaden Research Center, 2000.
9. Angela Bonifati, Dongwon Lee “Technical Survey of XML Schema and Query Languages”, January 2001.
10. Marcelo Arenas, Wenfei Fan, Leonid Libkin “Consistency Issues in XML Databases”, Private paper.
11. Marcelo Arenas, Wenfei Fan, Leonid Libkin “What’s Hard about XML Schema Constraints?”, DEXA 2002.
12. Wenfei Fan, Jerome Simeon, Scott Weinstein “Constraints for Semistructured Data and XML”, 2003.

13. Peter Buneman, Susan Davidson, Wenfei Fan, Carmem Hara, Wang-Cheiw Tan “Keys for XML”, March 2002.
14. Максим Гринев, Сергей Кузнецов, Андрей Фомичев «XML-СУБД Sedna: технические особенности и варианты использования», журнал «Открытые системы», август 2004.
15. Maxim Grinev, Andrey Fomichev, Sergey Kuznetsov, Kostantin Antipin, Alexander Boldakov, Dmitry Lizorkin, Leonid Novak, Maria Rekouts, Peter Pleshachkov “Sedna: A Native XML DBMS”.
16. М.Р. Когаловский «Энциклопедия технологий баз данных».
17. C. J. Date “An Introduction to Database Systems” Sixth Edition, the Systems Programming Series.
18. ISPRAS MODIS “Sedna Programmer’s Guide”, http://www.modis.ispras.ru/Development/sedna.htm
19. Makoto Murata “RELAX: Regular Language description for XML”, 2001