XML И БАЗЫ ДАННЫХ

By Ronald Bourret

Данная серия документов подготовлена на основе материалов сайта Школы Консорциума W3C. Этот сайт является экспериментальным сервером, на котором содержание документов хранится в формате XML. Пользователям сайта эти документы доступны в виде HTML (преобразование на стороне клиента с помощью таблицы стилей XSLT) и в виде PDF (преобразование тех же документов в XSL-FO, а затем в формат PDF).

Содержание

2: http://xml.nsu.ru/extra/database_1.xml
3: http://xml.nsu.ru/extra/database_2.xml
4: http://xml.nsu.ru/extra/database_3.xml
5: http://xml.nsu.ru/extra/database_4.xml
6: http://xml.nsu.ru/extra/database_5.xml
7: http://xml.nsu.ru/extra/database_6.xml

1.0 Введение

В этой статье обсуждаются отношения между XML и базами данных и описываются некоторые типы программного обеспечения для обработки XML документов совместно с базами данных. Хотя эта статья не претендует на полный охват, я надеюсь, в ней будут раскрыты многие вопросы использования XML и баз данных.

2.0 Является ли XML базой данных?

Прежде всего нам нужно ответить на часто возникающий вопрос: является ли сам XML базой данных? XML-документ является базой данных лишь в самом общем смысле этого слова, то есть, он является коллекцией данных. Этим он не отличается от многих других файлов - в конечном итоге, все файлы состоят из данных определенного вида. Как формат "базы данных" XML имеет некоторые преимущества, например, он само-описывающий (разметка описывает данные). Легко обеспечить его обрабтку различным программным обеспечением, поскольку данные хранятся в Unicode, он хранит данные в виде древовидной или графоподобной структуры. Но у него есть и некоторые недостатки, например, он слишком многословен и относительно медленно получает доступ к данным из-за необходимости парсирования и конвертирования текста.

8: http://xml.nsu.ru/extra/database_7.xml
9: http://xml.nsu.ru/extra/database_8.xml
10: http://xml.nsu.ru/extra/database_9.xml

Полезнее ответить на другой вопрос: является ли XML и связанные с ним технологии базой данных в более строгом смысле этого слова, то есть системой управления базой данных. Ответ на этот вопрос: "нечто вроде". К положительным сторонам можно отнести то, что XML позволяет реализовать многое из того, что можно обнаружить в обычных базах данных: хранение (XML-документы), схемы (DTD, язык XML-схем), языки запросов (XQuery, XPath, XQL, XML-QL, QUILT, и т.д..), программные интерфейсы (SAX, DOM, JDOM), и т.д. К недостаткам же можно отнести отсутствие многих возможностей, имеющихся в настоящих базах данных: экономичность хранения, индексы, безопасность, транзакции и интегрированность данных, многопользовательский доступ, триггеры, запросы по многим документам и т.д.

Таким образом, хотя и можно использовать XML-документы в качестве базы данных в средах с небольшим количеством данных, небольшим количеством пользователей и невысокими требованиями к производительности, этого нельзя сделать в большинстве бизнес-сред, в которых действует множество пользователей, действуют строгие требования к интегрированности данных и высокие требования к производительности.

Хорошим примером "базы данных", для которой вполне подходит XML-документ, является .ini-файл - то есть, файл, в котором содержится информация о конфигурации приложения. Гораздо легче придумать небольшой язык на основе XML и написать SAX-приложение для его интерпретации, чем создать парсер для файлов, в которых данные разделены запятыми. Кроме того, XML позволяет вкладывать друг в друга элементы данных - это довольно трудно сделать при разделении данных запятыми. Однако, такие файлы трудно назвать базами данных в полном смысле этого слова, поскольку они читаются и записываются линейно и только при открытии или закрытии приложения.

Более интересными примерами наборов данных, в которых удобно использовать XML-документ как базу данных, является личный список контактов (имен, номеров телефонов, адресов и т.д.), закладки в браузерах, описание MP3-файлов, которые вы украли с помощью Napster. Однако, в связи с низкой стоимостью и удобством пользования такими базами данных, как dBASE и Access, даже в этих случаях нет особых причин использовать в качестве базы данных XML-документ. Единственное реальное преимущество XML состоит в легкой переносимости данных от одного приложения к другому, но это преимущество не так важно, поскольку уже сейчас широко распространены инструменты для сериализации баз данных в XML-формат.

3.0 Для чего нужна база данных?

Первый вопрос, на который вы должны ответить, когда начинаете думать о совместном использовании XML и базы данных - зачем вам нужна база данных? Вы хотите обеспечивать доступ к некой функциональной информации? Вы ищете способ хранения своих веб-страниц? Станет ли ваша база данных приложением для электронной коммерции, в котором для транспортировки данных будет использоваться XML? Ответ на этот вопрос будет очень важен для выбора конкретной базы данных и промежуточногно программного обеспечения (если оно понадобится), а также и для выбора методов работы с базой данных.

Предположим, например, что ваше приложение работает в сфере электронной коммерции и для транспортировки данных использует XML. Наверняка в этом случае ваши данные имеют высоко-регулярную структуру и используются не XML-приложениями. Такие составляющие XML-документа, как сущности и кодировки, для вас, скорее всего, не представляют интереса - в конце концов, для вас важны сами данные, а не то, как они хранятся в XML-документе. В этом случае вам, вероятнее, больше подойдет реляционная база данных и специальное обеспечение для переноса данных между XML-документами и базой данных. Если ваше приложе-

ние объектно-ориентировано, вы даже можете создать систему для хранения этих объектов в базе данных и дальнейшей их сериализации в XML.

С другой стороны, пусть вы имеете веб-сайт, состоящий из нескольких контентно-ориентированных XML-документов (то есть, хранящих длинные куски текста и разметки). Вы не только хотели бы обеспечить управление сайтом, но и предоставить пользователям возможность поиска по содержанию сайта. Ваши документы, скорее всего, имеют менее регулярную структуру и использование, например, сущностей, имеет определенное значение, поскольку они имеют прямое отношение к структуре документов. В этом случае вам бы подошла истинная XML-база данных или система управления содержанием. Это бы вам позволило сохранять физическую структуру документов, поддерживать транзакции на уровне документов и выполнять запросы на языке XML-запросов.

4.0 Данные и документы

Вероятно, наиболее важный фактор, влияющий на выбор базы данных - используете ли вы базу данных для хранения **данных** или **документов**. Например, используется ли XML только для транспортировки данных между базами данных и (возможно, не XML-ными) приложениями? Или он применяется интегрально, как в случае XHTML или документов DocBook? Обычно это зависит от ваших задач, но важно с этим определиться, поскольку все документы, ориентированные на хранение данных (мы будем называть их **дата-центричными**) обладают некоторыми общими характеристиками, в свою очередь, **документо-центричные документы** обладают своими особенностями хранения XML в базе дынных. В следующих двух параграфах мы обсудим эти особенности.

(Исторический комментарий: впервые термины "дата-центричный" и "документо-центричный" я обнаружил на форуме xml-dev[11]. Я не знаю, кто его выдумал - было сообщение в форуме за 1997 год, в котором был использован термин "документо-центричный" и сообщение за 1998 год, в котором использовались уже оба термина.)

4.1 Дата-центричные документы

Дата-центричные документы - это документы, которые используют XML для транспортировки данных. Они создаются для машинной обработки и обычно имеет не самое важное значение применение для этого именно XML. То есть, для приложений или базы данных не имеет значения, что в некоторый момент времени данные хранятся в виде XML. Примерами дата-центричных документов являются торговые заказы, расписания рейсов, научные данные и биржевые индексы.

Дата-центричные документы характеризуются строго регулярной структурой, мелкой зернистостью данных (то есть, самый маленький кусочек данных находится на уровне чистых PCDATA-элементов или атрибутов), они отличаются отсутствием или малым количеством смешанного содержимого. Порядок, в котором следуют соседние элементы или секции PCDATA, обычно не имеет значения, кроме тех случаев, когда происходит валидация документа.

Например, следующие торговые заказы являются дата-центричными:

```
<SalesOrder SONumber="12345">
<Customer CustNumber="543">
<CustName>ABC Industries</CustName>
<Street>123 Main St.</Street>
```

11: http://www.xml.org/xml/xmldev.shtml

```
<City>Chicago</City>
<State>IL</State>
<PostCode>60609</PostCode>
</Customer>
<OrderDate>981215</OrderDate>
<Item ItemNumber="1">
<Part PartNumber="123">
<Description>
<p><b>Турецкий ключ:</b><br />
Нержавеющая сталь, цельно-металлическая
конструкция, пожизненная гарантия.</p>
</Description>
<Price>9.95</Price>
</Part>
<Quantity>10</Quantity>
</Item>
<Item ItemNumber="2">
<Part PartNumber="456">
<Description>
<p><b>Набивной сепаратор:<b><br />
Алюминий, гарантия - один год.</p>
</Description>
<Price>13.27</Price>
</Part>
<Quantity>5</Quantity>
</Item>
</SalesOrder>
```

Кроме случаев очевидных дата-центричных документов, вроде приведенного выше, существует много контентно-насыщенных документов, которые тоже можно назвать дата-центричными. Например, страница Amazon.com, на которой отображается информация о книге. Хотя страница образована преимущественно текстом, структура этого текста весьма регулярна, она почти тождественна для всех страниц, несущих информацию о книгах и каждый кусочек текста ограничен в размерах. Таким образом, в этом случае мы можем строить страницу из простого дата-центричного XML-документа, который извлекается из базы данных и содержит информацию о какой-то единственной книге. Далее этот документ можно обработать таблицей стилей XSL, которая добавит общий для всех страниц текст. Говоря в целом, любой веб-сайт, который динамически конструирует HTML-документ на основе шаблона, наполняя его данными из базы, может быть заменен на серию дата-центричных XML-документов и одной или несколькими таблицами стилей XSL.

Например, взгляните на следующий документ, описывающий авиарейс:

```
<FlightInfo>
<Airline>ABC Airways</Airline>
provides <Count>three</Count>
non-stop flights daily from <Origin>Dallas</Origin>
to <Destination>Fort Worth</Destination>.
Departure times are <Departure>09:15</Departure>,
<Departure>11:15</Departure>, and
<Departure>13:15</Departure>. Arrival times
are minutes later.</FlightInfo>
```

Он может быть построен из следующего XML-документа с добавлением простой таблицы стилей:

```
<Flights>
```

```
<Airline>ABC Airways</Airline>
<Origin>Dallas</Origin>
<Destination>Fort Worth</Destination>
<Flight>
<Departure>09:15</Departure>
<Arrival>09:16</Arrival>
</Flight>
<Flight>
<Departure>11:15</Departure>
<Arrival>11:16</Arrival>
</Flight>
<Flight>
<Departure>13:15</Departure>
<Arrival>13:16</Arrival>
</Flight>
</Flights>
```

4.2 Документо-центричные документы

Документо-центричными документами (обычно) являются документы, которые созданы для восприятия человеком. Примерами являются книги, сообщения электронной почты, реклама и почти все созданные вручную XHTML-документы. Они характеризуются менее регулярной или иррегулярной структурой, крупно-зернистыми данными (то есть, самые маленькие независимые части данных имеют уровень элементов со смешанным содержимым или целого документа), и содержат много смешанного содержимого. Порядок, в котором следуют соседние элементы или секции типа PCDATA, почти всегда имеет значение.

Например, следующее описание продукта - документо-центрично:

```
<Product>
<Name>Турецкий гаечный ключ</Name>
<Developer>Full Fabrication Labs, Inc.</Developer>
<Summary>Похож на обезьяний ключ,
но не такой большой.</Summary>
<Description>
<Para>Турецкий ключ, который производится <i>
как в варианте для правой руки, так и в варианте
для левой руки (на выбор)</i>, изготовлен
из <b>лучших сортов нержавеющей стали</b>.
Удобная прорезиненная рукоятка очень крепко
держится в руке, даже в ситуации сильного загрязнения.
С помощью дополнительных насадок возможно
расширение его возможностей.</Para>

<Para>Вы можете:</Para>

<List>
<Item><Link URL="Order.html">Заказать турецкий
гаечный ключ</Link></Item>
<Item><Link URL="Wrenches.htm">Узнать больше о
гаечных ключах</Link></Item>
<Item><Link URL="Catalog.zip">Скачать каталог
</Link></Item>
</List>
```

```
<Para>Турецкий ключ стоит <b>всего $19.99</b> и
вместе с ним, если вы его закажете сейчас, вы получите
<b>ручной креветочный молоточек</b> в
качестве подарка.</Para>

</Description>

</Product>
```

4.3 Данные, документы и базы данных

В практике разделение между дата-центричными и документо-центричными документами не всегда однозначно. Например, некоторые дата-центричные документы, скажем, фактуры, могут содержать крупно-зернистые иррегулярно-структурированные данные, например, описание разделов или комментарии. С другой стороны, иногда документо-центричные документы, например, технические руководства, могут содержать отлично сгруппированные, регулярно-структурированные данные (часто это мета-данные), такие, как авторские данные и дата публикации. Другой пример - юридические или медицинские документы, которые создаются с ориентиром на содержание, но содержат дискретные куски информации, такие, как даты, имена, процедуры и часто должны сохраняться как единые документы для общего пользования.

Несмотря на это, полезно характеризовать свои документы как дата-центричные или документо-центричные, поскольку это поможет вам решить, какую базу данных использовать. В качестве общего правила: данные хранятся в традиционной базе данных, например, реляционной, объектно-ориентированной или иерархической, а документы хранятся в **истинной XML-базе данных** (может быть, с помощью приложения, созданного для управления документами и построенного на основе истинной XML-базы данных) или в **системе управления содержанием** (это приложение, сконструированное для управления документами и построенное на основе истинной XML-базы данных).

Эти правила не абсолютны. Данные - особенно полу-структурированные данные - могут храниться в истинной XML-базе данных, а документы можно хранить в традиционной базе данных, когда не нужно особых XML-зависимых операций. Более того, границы между традиционными базами данных и истинными XML-базами начинают стираться по мере того, как в традиционных базах данных появляются свойства, присущие истинным XML-базам, а истинные XML-базы данных начинают поддерживать хранение фрагментов документов во внешних (обычно реляционных) базах данных.

В оставшейся части статьи мы обсудим стратегии и вопросы, связанные с хранением и извлечением данных (глава 5) и документов (глава 6). Если вам нужна свежая информация относительно программных продуктов, связанных с данной темой, смотрите статью XML Database Products[12].

5.0 Хранение и извлечение данных

Данные в дата-центричных документах могут иметь свое происхождение из базы данных (в этом случае их нужно трансформировать в XML) или из внешней базы данных (в этом случае

12: http://www.rpbourret.com/xml/XMLDatabaseProds.htm

вам нужно сохранить их в базе данных). Примером первого типа может являться огромное количество различных данных, хранящихся в реляционных базах данных; примером второго типа можно считать научные данные, собираемые некой измерительной системой и конвертируемые в XML.

Таким образом, в зависимости от ваших нужд, вам может понадобиться программное обеспечение, которое переносит данные из XML-документа в базу данных, из базы данных - в XML-документ, или и то и другое. Это программное обеспечение может либо быть встроено в базу данных, в этом случае говорится, что такая база данных "**оснащена средствами XML**", или этот перенос может осуществляться с помощью специально написанного **промежуточного программного обеспечения**.

5.1 Перенос данных

При переносе данных из XML-документа в базу данных, часто можно опускать информацию о документе, например, его имя или DTD. Кроме того, можно опускать информацию о физической структуре документа, например, определение сущностей и параметр usage, который задает способ хранения бинарных данных (Base64, как непарсируемая сущность или как-то еще), секции CDATA и информацию о кодировке. Иногда даже можно опустить информацию о логической структуре, в частности, процессуальные инструкции и порядок, в котором следуют значения атрибутов или соседние элементы.

Подобным образом, при переносе данных из базы данных в XML-документ, результирующий XML-документ, скорее всего, не будет содержать CDATA или сущностей (кроме изначально заданных сущностей lt, gt, amp, apos, и quot), а порядок, в котором будут следовать соседние элементы или атрибуты будет определяться порядком, в котором эти данные были выданы базой данных.

Хотя сперва это может показаться удивительным, все это вполне разумно. Например, в случае, когда XML используется для переноса торговых заказов из одной базы данных - в другую. В этом случае не имеет значения, хранится в документе номер заказа до или после даты заказа, не имеет значения и как хранится имя заказчика: как секция типа CDATA, как внешняя сущность, как атрибут или прямо как секция PCDATA. Имеет значение только правильный перенос важных данных из одной базы в другую. Таким образом, программное обеспечение для переноса данных должно учитывать только иерархический порядок в котором группируется информация в каждом торговом заказе и ничего больше.

Одним из последствий игнорирования информации относительно документа и его физической структуры является отсутствие обратимости - то есть, сохранение данных из документа в базу данных, а затем обратное восстановление документа из базы часто приводит к другому документу, даже в каноническом смысле слова. Допустимо ли это - зависит от ваших требований, и ответ на этот вопрос может повлиять на выбор базы данных и программного обеспечения для переноса данных.

5.2 Отображение документа в базу данных

Программное обеспечение для переноса данных между XML-документом и базой данных действует на основе некоторого отображения между документом и базой данных. Например, можно отображать элемент <SalesOrder> в таблицу SalesOrders, а элемент <OrderDate> - в колонку SalesOrders.Date. Отображения можно разделить на две категории: на основе шаб-

лона и на основе модели.

5.2.1 Отображения на основе шаблона

При отображении на основе шаблона между документом и базой данных не существует какого-то изначально заданного соответствия. Вместо этого в шаблон каждый раз вставляются специально подобранные команды, которые обрабатываются программами переноса данных. Например, следующий шаблон имеет встроенные выражения SELECT в виде элементов <SelectStmt>:

```
<?xml version="1.0"?>
<FlightInfo>
<Introduction>Есть свободные места на
следующие рейсы:</Introduction>
<SelectStmt>SELECT Airline, FltNumber,
Depart, Arrive FROM Flights</SelectStmt>
<Conclusion>Мы надеемся, вы найдете
что-то подходящее</Conclusion>
</FlightInfo>
```

При обработке программой переноса данных кажде выражение SELECT заменяется его результатом, отформатированным как XML:

```
<?xml version="1.0"?>
<FlightInfo>
<Introduction>Есть свободные места на
следующие рейсы:</Introduction>
<Flights>
<Row>
<Airline>ACME</Airline>
<FltNumber>123</FltNumber>
<Depart>Dec 12, 1998 13:43</Depart>
<Arrive>Dec 13, 1998 01:21</Arrive>
</Row>
...
</Flights>
<Conclusion>Мы надеемся, вы найдете
что-то подходящее</Conclusion>
</FlightInfo>
```

Отображения на основе шаблонов могут быть весьма гибкими. Хотя набор возможностей варьирует от одного программного продукта к другому, обычно имеются следующие возможности:

- Возможность помещения значений результирующего множества (результата запроса в базу данных) в любое место возвращаемого документа, включая их в качестве параметров в последующие выражения SELECT.
- Программирование конструктов, например, циклов и условных выражений.
- Задание переменных и функций.
- Параметризация выражений SELECT через HTTP-параметры.

Отображения на основе шаблонов применяются почти исключительно только для переноса данных из реляционных баз данных в XML-документы. Хотя некоторые продукты, работающие на основе шаблонных отображений могут переносить данные из XML-документов в реляционные базы данных, для этого в них не используется полный язык работы с шаблонами.

Для этого применяется описываемое ниже отображение на основе таблиц.

5.2.2 Отображения на основе моделей

В отображениях на основе моделей данные в XML-документе моделируются в соответствии с некоторой изначально заданной моделью, которая явно или скрыто отображается в базу данных. Несмотря на потерю гибкости, такая схема выигрывает в простоте, поскольку система, которая основывается на конкретной модели обычно может сделать для пользователя больше, чем отображение на основе шаблонов.

Поскольку перенос данных осуществляется на основе предварительно заданной модели, а многие документы могут не соответствовать этой модели, в продукты, работающие на основе модельного отображения обычно интегрируется XSLT. Это позволяет приложениям трансформировать документы к нужной модели перед переносом данных в базу данных и наоборот, обеспечивая гибкость, присущую шаблонным отображениям.

Существует две распространенных модели данных в XML-документе: **модель таблицы**, которая является основой отображений на основе таблиц, и **объектная модель**, которая является основой объектно-реляционных отображений.

5.2.2.1 Отображения на основе таблиц

Отображения на основе таблиц применяются во многих программах переноса данных между XML-документами и реляционными базами данных. В них XML-документ представляется как таблица или множество таблиц. То есть, структура документа должна быть примерно такой, как в этом примере, при этом элемент <database> и дополнительный элемент <table> отсутствуют в одно-табличном случае:

```
<database>
<table>
<row>
<column1>...</column1>
<column2>...</column2>
...
</row>
<row>
...
</row>
...
</table>
<table>
...
</table>
...
</database>
```

В некоторых программных продуктах можно конкретизировать, сохраняются ли данные в колонке как дочерние элементы или как атрибуты, также, как и имена, используемые для каждого элемента и атрибута. Кроме того, в продуктах на основе табличных отображений часто реализована возможность включения табличных и колоночных метаданных либо в начале документа либо в виде атрибутов каждой табличного или колоночного элемента. Заметьте, что термин "таблица" обычно понимается весьма вольно. То есть, при переносе данных из

базы данных в XML, "таблицей" может быть результирующее множество, а при переносе данных из XML в базу данных, "таблицей" может являться реальная таблица или другое модифицируемое отображение.

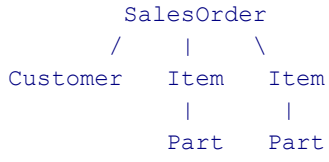
Отображения на основе таблиц полезны для сериализации реляционных данных, например, при переносе данных между двумя реляционными базами данных. К очевидному недостатку таких систем можно отнести то, что они не могут работать с XML-документами, которые не удовлетворяют приведенному выше формату.

5.2.2.2 Объектно-реляционное отображение

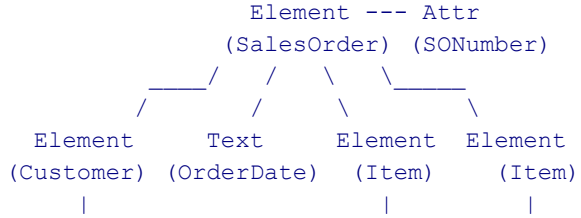
Объектно-реляционное отображение применяется всеми реляционными базами, оснащенными средствами XML, и некторыми программами переноса данных между XML-документами и реляционными базами данных. В этом отображении данные в XML-документе моделируются деревом объектов, специфичных для данного типа документов. В этой модели типы элементов с атрибутами, содержимое элементов или смешанное содержимое (**сложные типы элементов**) обычно моделируются классами. Типы элементов, чье содержимое является чистыми секциями PCDATA (**простые типы элементов**), атрибуты и секции PCDATA моделируются скалярными свойствами. Затем модель отображается в реляционную базу данных с помощью традиционных техник объектно-реляционного отображения или с помощью объектных представлений SQL 3. То есть, классы отображаются в таблицы, скалярные свойства - в колонки, а объектно-означенные свойства отображаются в пару первичный ключ / внешний ключ.

(Термин "объектно-реляционное отображение" не совсем верен, поскольку дерево объектов может быть прямо отображено в объектно-ориентированную или иерархическую базу. Однако, мы его все равно будем использовать, поскольку огромное количество продуктов, которые используют этот вид отображений, действуют на основе реляционных баз данных, и термин "объектно-реляционное отображение" широко используется.)

Важно понимать, что объектная модель, используемая в этом типе отображения отличается от объектной модели документа DOM. DOM моделирует сам документ и является одинаковой для всех XML-документов, а выше описанная модель моделирует данные в документе и различается для каждого набора XML-документов, которые соответствуют определенным DTD. Например, описанный выше документ, содержащий информацию о торговых заказах, можно смоделировать как дерево объектов, принадлежащих четырем классам - SalesOrder, Customer, Item, и Part - как показано на этой диаграмме:



В то время как дерево DOM, построенное по тому же документу, должно быть составлено из таких объектов, как Element, Attr, и Text:



```
etc.                etc.      etc.
```

Инициализируются ли на самом деле объекты этой модели - зависит от продукта. Некоторые программные продукты позволяют вам порождать классы в модели, а затем использовать объекты этих классов в своих приложениях. С помощью таких продуктов данные переносятся между XML-документом, этими объектами и базой данных. В других продуктах объекты используются лишь как инструмент, который помогает вам визуализировать отображение, а перенос производится прямо между XML-документом и базой данных. И полностью от вашего приложения зависит, есть ли смысл в инициализации этих промежуточных объектов.

(Привязка XML-документов к объектам обычно называется **привязкой XML-данных** (XML data binding) - после реализации компанией Sun технологии Java Architecture for XML Binding[13]. В некоторых программных продуктах реализуется привязка XML-данных; большинство из них могут переносить также и данные между объектами и базой данных.)

Способы осуществления объктно-реляционного отображения вариьируют, например:

- Во всех программных продуктах поддерживается отображение сложных типов элементов в классы, а простых типов элементов и атрибутов - в свойства.
- Все (?) продукты позволяют вам указать корневой элемент, который не отображается в объектную модель или базу данных. Этот несущий элемент полезен в тех случаях, когда вам нужно в одном XML-документе хранить несколько объектов верхнего уровня. Например, если вы хотите хранить в одном XML-документе несколько торговых заказов, вам потребуется охватить их единым корневым элементом.
- Корневой элемент эквивалентен элементу <database>, он используется, когда XML-документ, использующий табличное отображение содержит множество таблиц и присутствует только для того, чтобы выполнить требование XML (документ должен иметь единственный корневой элемент). В некоторых программных продуктах также можно обозначить элементы более низкого уровня как несущие.
- Большая часть продуктов позволяет определить, отображаются ли свойства в атрибуты или дочерние элементы XML-документа. Некоторые продукты позволяют смешивать атрибуты и дочерние элементы, другие разрешают либо одно, либо другое.
- Большая часть продуктов позволяет использовать различающиеся имена в XML-документе и базе данных, но в некоторых требуется использовать одинаковые.
- Большая часть продуктов позволяет определить порядок, в котором следуют дочерние элементы в родительском элементе, хотя это реализуется так, что невозможно построить много моделей контента. К счастью, поддерживаемые модели контента годятся для большей части датацентричных документов. (Порядок следования дочерних элементов важен при валидации документа.)
- Некоторые продукты позволяют отображать комплексные типы элементов в скалярные свойства. Это полезно, когда тип элемента содержит смешанное содержимое, такое, как элемент <Description> в приведенном выше примере торгового заказа. Хотя элемент <Description> имеет дочерние элементы и техст в форме XHTML, гораздо полезнее рассматривать этот элемент как единое свойство, нежели разбивать его на части.
- Некоторые продукты поддерживают отображение PCDATA в смешанное содержимое.

Более полное описание объектно-реляционного отображения смотрите в разделе 3 материала "Отображение DTD в базы данных"[14].

5.3 Хранение данных в истинной XML-базе данных

Также можно хранить данные XML-документов в истинной XML-базе данных (native XML

13: http://java.sun.com/xml/jaxb
14: http://www.xml.com/pub/a/2001/05/09/dtdtodbs.html#object

database). Есть несколько причин для этого. Первая - полу-структурированность данных. То есть, они обладают регулярной структурой, но эта структура достаточно сильно варьирует и отображение ее в реляционную базу данных приводит к большому числу колонок с пустым значением (пустая трата места) или к большому количеству таблиц (что неэффективно). Хотя полу-структурированные данные можно хранить в объектно-ориентированной или иерархической базе данных, можно хранить их в истинной XML-базе данных в виде XML-документов.

Вторая причина хранить данные в истинной XML-базе данных - это скорость доступа. В зависимости от того, как физически истинная XML-база данных хранит данные, она может получать доступ к данным гораздо быстрее, чем реляционная база данных. Причина в том,что некоторые стратегии хранения, применяемые в истинных XML-базах хранят целые документы физически вместе или используют физические (а не логические) указатели между частями документа. Это позволяет извлекать документы либо без объединений, либо с помощью физических объединений, и то и другое быстрее, чем логические объединения, которые применяются в реляционных базах данных.

Рассмотрим, например, приведенный выше документ, содержащий торговые заказы. В реляционной базе данных его можно было бы хранить в четырех таблицах: SalesOrders, Items, Customers, и Parts, и извлечение документа могло бы потребовать объединения по всем этим таблицам. В истинной же XML-базе данных весь документ мог бы храниться в одном месте на диске, так что получение доступа к документу или к его фрагменту потребовало бы единственного обращения к диску и единственного чтения данных. Реляционная база потребовала бы четыре поиска индексов и по меньшей мере четырех чтений с диска.

Очевидное соображение в этом случае: увеличение скорости будет ощущаться только в тех случаях, когда данные на выходе должны быть организованы именно так, как они хранятся на диске. Если вы хотите получать данные в различных представлениях, например, в виде списка заказчиков и относящихся к ним заказов, производительность была бы, по всей видимости, меньше, чем в реляционной базе данных. Таким образом, хранение данных в истинной XML-базе из-за стремления к большей производительности оправдано лишь в тех случаях, когда в вашем приложении доминирует какое-то одно представление данных.

Третья причина хранить данные в истинной XML-базе данных - если вам нужны некие XML-специфичные возможности, например, выполнение XML-запросов. С учетом того, что немногие дата-центричные приложения нуждаются сегодня в этом и многие реляционные базы данных могут выполнять XML-запросы, эта причина становится менее существенной.

Одна из проблем хранения данных в истинной XML-базе заключается в том, что большинство истинных XML-баз данных могут возвращать данные только в виде XML. (слабо поддерживается привязка элементов или атрибутов к переменным приложения.) Если вашему приложению понадобятся данные в другом формате (часто так и происходит), приложению придется сначала отпарсить XML и только после этого оно сможет использовать эти данные. Это очевидный недостаток для локальных приложений, которые вместо реляционой базы данных используют истинную XML-базу, поскольку требуется дополнительные ресурсы, в которых нет нужды, например, в ODBC-приложениях. Но это не проблема в распределенных приложениях, которые используют XML для транспортировки данных, поскольку дополнительные ресурсы понадобятся вне зависимости от того, какая используется база данных.

5.4 Типы данных, пустые значения, наборы символов и прочее

В этом параграфе мы обсудим несколько вопросов, имеющих отношение к хранению данных из XML-документов в традиционной базе данных. (вопросы,связанные с типами данных и бинарными данными также относятся и к хранению даных в истинной XML-базе данных.)

Обычно нет выбора в вариантах решения этих проблем программами переноса данных. Однако, вы должны знать о существовании этих проблем, поскольку это может вам помочь при выборе программного обеспечения.

5.4.1 Типы данных

XML не поддерживает типизацию данных в любом значимом смысле этого слова. Кроме непарсируемых сущностей, все данные в XML-документе являются текстом, даже если он представляет другие типы данных, например, даты или целые числа. В общем случае программы переноса конвертируют данные из текста (в XML-документе) в другие типы (в базе данных) и наоборот. Однако, число текстовых форматов, в которых распознаются данные того или иного типа, ограничено, например, теми, котрые заданы в JDBC-драйвере. Представление дат - наиболее проблемное место, поскольку диапазон возможных форматов очень велик. Числа, которые также имеют несколько национальных форматов, тоже могут вызвать проблемы.

5.4.2 Бинарные данные

Есть два распространенных способа хранения бинарных данных в XML-документе: как непарсируемые сущности и в кодировке Base64 (MIME-кодировка, которая отображает бинарные данные в подмножество US-ASCII). Для реляционных баз данных оба способа проблематичны, поскольку правила, по которым данные отправляются или извлекаются из базы данных, могут быть очень строгими и вызывать проблемы в программном обеспечении. Кроме того, не существует стандартной XML-записи, указывающей, что элемент содержит данные, закодированные в кодировке Base64, так что программное обеспечение может даже не узнать, что данные являются закодированными бинарными данными. И наконец, записи, связанные с непарсируемыми сущностями или элементами в кодировке Base64 - одна из тех вещей, которые скорее всего будут опускаться при переносе данных в базу данных. Таким образом, если для вас важны бинарные данные, проверьте, поддерживает ли работу с ними ваше программное обеспечение.

5.4.3 Пустые данные

В мире баз данных, "**null**" означает, что данных просто нет. Это нечто совсем другое, чем 0 (для чисел) или строка нулевой длины. Например, у вас имеются данные, собранные на метеостанции. Если термометр не работает, в базе должно сохраняться пустое значение, а не нуль, потому что нуль в данном случае обозначает нечто другое.

XML тоже поддерживает концепцию пустых данных - введением необязательных типов элементов или атрибутов. Если значение необязательного типа элемента или атрибута - пустое, он просто не включается в документ. Как и в базах данных, атрибут, содержащий строку нулевой длины или пустой элемент не является пустым значением: их значение равно строке нулевой длины.

При отображении структуры XML-документа в базу данных и обратно, следует позаботиться о том, чтобы необязательные типы элементов и атрибуты отображались в колонки, для которых возможны пустые значения, и наоборот. В противном случае вероятны ошибки вставки данных (при переносе данных в базу) или невалидные документы (при переносе данных из

базы).

Поскольку XML-сообщество, в отличие от специалистов по базам данных, имеет, похоже, более широкий взгляд на то, что такое пустое значение - в частности, многие пользователи XML считают атрибуты нулевой длины и пустые элементы пустыми значениями - вам следует проверить, как себя ведет ваше программное обеспечение в этих ситуациях. Некоторые программы предоставляют пользователям возможность выбора в определении пустого значения в XML-документе, включая поддержку атрибута xsi:null из XML-схем.

5.4.4 Наборы символов

По определению, XML-документ может содержать любые символы Unicode, кроме некотрых специальных символов. К сожалению, многие базы данных плохо поддерживают или совсем не поддерживают Unicode и требуют специальной конфигурации для работы с не-ASCII символами. Проверьте, как работает с ними ваша база данных и программы переноса.

5.4.5 Процессуальные инструкции и комментарии

Процессуальные инструкции и комментарии не являются частью данных XML-документа и все известные программы переноса игнорируют их. Проблема состоит в том, что они могут виртуально возникать в любом месте XML-документа и таким образом, с трудом встраиваются в табличное или объектно-реляционное отображение. (Одно из плохо работающих решений для реляционных баз данных - добавление специальных таблиц для процессуальных инструкций и комментариев, добавление в эти таблицы внешних ключей для применения во всех других таблицах и проверка этих таблиц при каждой обработке какой-то другой таблицы.) Таким образом, большая часть программ переноса просто их игнорирует. Если процессуальные инструкции и комментарии для вас важны, проверьте, что с ними делает ваше программное обеспечение. Тут можно также применить истинную XML-базу данных.

5.4.6 Сохранение разметки

Как уже говорилось в параграфе 5.2.2, иногда лучше хранить элементы со смешанным содержимым в базе данных без дальнейшего парсирования. Когда мы так делаем, разметка сохраняется с символами разметки. Однако возникает проблема с представлением символов разметки, которые не используются для разметки. В XML-документах они сохраняются как сущности lt, gt, amp, quot, и apos. То же самое может быть сделано и в базе данных. Например, следующее описание (description):

```
<description>
<b>Пример путаницы:</b> &lt;foo/&gt;
</description>
```

может быть сохранен в базе данных как:

```
<b>Пример путаницы:</b> &lt;foo/&gt;
```

Проблема тут в том, что не XML-ные языки запросов, такие, как SQL, не сканируют значения колонок на наличие разметки и на предмет использования сущностей и интепретируют их соответствующе. Таким образом, если вы хотите найти строку "<foo/>" с помощью SQL, вам нужно знать, что на самом деле вам следует искать строку "<foo/>".

С другой стороны, если ссылки на сущность расшифрованы, программы переноса не могут различить разметку от использования сущностей. Например, если приведенное выше описание сохраняется как показано ниже, программное обеспечение не может определить, являются ли , , и <foo/> разметкой или текстом.

```
<b>Пример путаницы:</b> <foo/>
```

Решение этой проблемы в перспективе - распознающие XML базы данных, в которых реальная разметка обрабатывается иначе, чем то, что только выглядит как разметка. А пока при работе не XML-ных приложений с XML могут возникать проблемы.

5.5 Генерация определений DTD из реляционной схемы и обратно

При переносе данных между XML-документами и базой данных частно возникает вопрос - как сгенерировать XML DTD из схемы базы данных и обратно. Прежде, чем мы поговорим о том, как это делается, сразу заметим, что это должно делаться на стадии разработки. Причина в том, что большая часть дата-центричных приложений и виртуально все вертикальные приложения работают с известным набором определений DTD и схем базы данных. Так что им нет необходимости генерировать схемы в процессе работы. Более того, как мы увидим ниже, процедуры генерации схем не совершенны. Приложениям, которым приходится сохранять случайные XML-документы в базе данных, лучше работать с истинной XML-базой, нежели генерировать схемы в процессе работы.

Самый простой способ генерации реляционной схемы из определений DTD и обратно - просто заложить в код путь через объектно-реляционное отображение, у которого есть несколько необязательных свойств. Похожие процедуры имеются в объектно-ориентированных базах данных. (В 4 разделе материала "Отображение определений DTD в базы данных"[15] имеются пошаговые примеры каждой процедуры.)

Чтобы сгенерировать реляционную схему из DTD нужно:

- Для каждого комплексного типа элементов создать таблицу, а в ней - колонку с первичным ключом.
- Для каждого элемента со смешанным содержимым создать отдельную таблицу, в которой будут храниться секции PCDATA, связанные с родительской таблицей через первичный ключ таблицы.
- Для каждого атрибута с единственным значением, принадлежащего элементу этого типа, и для каждого единично появляющегося простого дочернего элемента, создайте колонку в этой таблице. Если дочерний тип элемента или атрибут - необязателен, сделайте колонку допускающей пустое значение.
- Для каждого многозначного атрибута или для каждого многократно-возникающего простого дочернего элемента создайте отдельную таблицу для хранения значений, привязанных к родительской таблице через первичный ключ таблицы.
- Для каждого сложного дочернего элемента свяжите таблицу родительского типа элемента с таблицей дочернего типа элемента с помощью первичного ключа родительской таблицы.

Чтобы сгенерировать DTD из реляционной схемы нужно:

- Для каждой таблицы создать тип элемента.
- Для каждой не-ключевой колонки в этой таблице, также, как и для колонки (колонок) с первичным ключом, добавьте к контентной модели элемента атрибут или дочерний элемент, который может содержать только секцию типа PCDATA.
- Для каждой таблицы, в которую передается первичный ключ, добавьте дочерний элемент к

15: http://www.xml.com/pub/a/2001/05/09/dtdtodbs.html?page=4#generate

контентной модели и обрабатывайте таблицу рекурсивно.

- Для каждого внешнего ключа добавьте к контентной модели дочерний элемент и обрабатывайте внешний ключ рекурсивно.

В этих процедурах есть несколько возможных проблем. Многие из них можно исправить вручную, например, конфликт имен или задание типа данных колонок и их длины. (В DTD не содержится информация о типах данных, так что предсказать, какой тип данных следует использовать в базе, невозможно. Обратите внимание, что тип данных и длина данных может быть предсказана из документа XML-схемы.)

Более серьезная проблема состоит в том, что "модель" данных, используемая в XML-документе часто отличается (и обычно более сложна), чем большинство эффективных моделей хранения данных в базе данных. Например, рассмотрим следующий фрагмент XML-кода:

```
<Customer>
   <Name>ABC Industries</Name>
   <Address>
      <Street>123 Main St.</Street>
      <City>Fooville</City>
      <State>CA</State>
      <Country>USA</Country>
      <PostCode>95041</PostCode>
   </Address>
</Customer>
```

Процедура генерации реляционной схемы из определения DTD могла бы в этом случае создать две таблицы: одну для заказчиков и одну для адресов. Однако, в большинстве случаев было бы разумнее хранить адрес в таблице заказчиков, а не в отдельной таблице.

Элемент <Address> - это хороший пример корневого несущего элемента. **Несущий элемент** обычно нужен по двум причинам. Во-первых, он предоставляет дополнительную структуру, которая облегчает понимание документа. Во-вторых, он часто используется в качестве некой формы типизации данных, например, элемент <Address> можно передавать процедуре, которая конвертирует все адреса в объект Address, вне зависимости от того, где находится этот элемент.

Хотя корневые элементы полезны в XML, они обычно становятся причиной проблем в форме излишней структуры при отображении в базу данных. По этой причине перед генерацией реляционной схемы их обычно удаляют из определения DTD. Поскольку вряд ли хорошим решением является радикальное изменение DTD, это приводит к расхождению между реальными документами и документами, подходящими для обработки программами переноса, поскольку корневой элемент не вовлекается в отображение. Эту проблему можно решить, преобразуя документы на лету, например, с помощью XSLT: корневые элементы удаляются перед переносом данных в базу и снова вставляются после переноса данных из базы.

Не смотря на эти трудности, приведенные выше процедуры являются хорошей стартовой точкой для решения проблемы генерации определений DTD из реляционных схем и обратно, особенно в случае больших систем.

6.0 Хранение и извлечение документов

Документо-центричные документы обычно пишутся вручную в виде XML или другого формата, например, RTF, PDF, или SGML, которые затем конвертируются в XML. В отличие от дата-центричных документов, они обычно не несут свое происхождение из базы данных. (документы, построенные на основе шаблона, в который вставляются данные - дата-

центричны. За дополнительной информацией обратитесь к разделу 4.1.)

Таким образом, если вы работаете с документо-центричными документами, вам вероятно понадобится способ хранения и извлечения документов, и методы конвертации их в тот или иной формат. В этом разделе мы обсудим истинные XML-базы данных, которые можно использовать для хранения XML-документов. Если вам нужна информация о конвертирующем программном обеспечении, смотрите список сайтов, посвященных XML-программному обеспечению в разделе 3.0 статьи XML Database Products[16].

6.1 Простые системы хранения документов

Если вы имеете простой набор документов, например, набор документации, у вас есть несколько вариантов того, как их хранить. Простейшее решение - хранить их в виде файловой системы и использовать инструменты типа grep для запроса к ним. (Полнотекстовые поиски по XML-документам, очевидно, не очень эффективны, поскольку им трудно различить разметку и текст и они не могут распознать использование сущностей. Однако, в маленьких системах даже это может быть вполне приемлемо.) Если вы хотите иметь простой контроль транзакций, вы можете поместить свои документы в систему контроля, например CVS или RCS.

Немного более хитрое решение - это хранение документов как BLOB-ы в реляционной базе данных. Это предоставляет несколько возможностей: контроль транзакций, безопасность, многопользовательский доступ и т.д. Кроме того, многие реляционные базы данных обладают инструментами поиска текста и могут обеспечивать полнотекстовый поиск, примерный поиск, поиск по синонимам и нечеткий поиск. Некоторые из этих инструментов распознают XML, что решает проблемы, связанные с поискам по XML-документам как по простому тексту.

Когда вы сохраняете XML-документы как BLOB-ы, вы легко можете создать простое распознающее XML-индексирование, даже если база данных не может индексировать XML. Один из способов - создать пару таблиц: индексную таблицу (известную как **боковая таблица** в DB2, именно оттуда происходит эта идея) и документную таблицу. Документная таблица содержит первичный ключ и колонку с типом данных BLOB, в которой хранится документ. Индексная таблица содержит колонку, в которой размещаются индексируемые значения и внешний ключ, указывающий на первичный ключ документной таблицы.

Когда документ хранится в базе данных, ведется поиск по всем экземплярам инедексированного элемента или атрибута. Каждый экземпляр сохраняется в индексной таблице, вместе с первичным ключом документа. Затем значение колонки индексируется что позволяет приложению быстро находить конкретные значения элементов или атрибутов и извлекать соответствующий документ.

Пусть, например, вы имеете набор документов, которые удовлетворяют приведенному ниже DTD и вы хотите создать индекс авторов:

```
<!ELEMENT Brochure (Title, Author, Content)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Content (%Inline;)>
<!-- Inline entity from XHTML -->
```

Вы можете сохранить документы в следующих таблицах:

```
Authors                    Brochures
----------------------     ---------
Author     VARCHAR(50)     BrochureID INTEGER
```

16: http://www.rpbourret.com/xml/XMLDatabaseProds.htm#links

```
BrochureID INTEGER          Brochure   LONGVARCHAR
```

Когда вы вставляете брошюру в таблицу Brochures, вы сканируете элементы <Author> и сохраняете их значения и ID брошюры в таблице Authors. Тогда приложения смогут извлечь бошюры по автору с помощью простого выражения SELECT. Например, чтобы извлечь все брошюры, написанные автором по имени Chen, приложение должно выполнить следующее выражение:

```
SELECT Brochure
FROM Brochures
WHERE BrochureID IN
(SELECT BrochureID FROM Authors WHERE Author='Chen')
```

Более сложная индексная таблица может состоять из четырех колонок: элементный тип или атрибут name, тип (элемент или атрибут), значение и ID документа. Это помогло бы сохранять значения множественных разметочных конструктов в одной таблице и потребовало бы индексации по имени, типу и значению. Написание специального SAX-приложения для работы с этой таблицей было бы относительно несложным.

6.2 Истинные XML-базы данных

Если вам нужно больше возможностей, чем предоставляется простыми системами, описанными выше, вам следует подумать об истинной XML-базе данных. **Истинные XML-базы данных** созданы специально для хранения XML-документов. Как и другие базы данных, они поддерживают возможности транзакций, безопасности, многопользовательского доступа, программных API, языки запросов и т.д. Единственное отличие от других типов баз данных состоит в том, что их внутренняя модель основана на XML и не на чем другом, например, реляционнной модели.

Истинные XML-базы данных, очевидно, наиболее полезны для хранения дата-центричных документов. Причина в том, что этот тип баз данных сохраняют порядок документа, процессуальные инструкции, комментарии, секции CDATA и использование сущностей, а традиционные базы данных даже оснащенные средствами работы с XML - нет. Кроме того, истинные XML-базы данных поддерживают языки запросов XML, позволяя давать базе данных запросы вроде "Дай мне все документы, в которых третий параграф содержит слово, выделенное жирным шрифтом". В языках, подобных SQL такие запросы реализовать довольно затруднительно.

Истинные XML-базы также полезны для хранения документов, "родным форматом" для которых является XML, вне зависимости от того, что именно содержится в этих документах. Предположим, например, что XML-документы используются для обмена сообщениями в системе электронной коммерции. Хотя эти документы, вероятно, дата-центричны, их натуральным форматом является XML. Таким образом, при хранении сообщений в архиве, его разумнее строить на основе истинной XML-базе данных, чем на не-XML-ной базе. Истинная XML-база данных предоставляет XML-специфичные возможности, например, языки запросов XML, и будет быстрее получать доступ к отдельным сообщениям в архиве. Другим примером подобного сорта является система расчетов через Интернет (Web cache).

Еще одним применением истинной XML-базы является хранение полу-структурированных данных, что в некоторых ситуациях увеличит скорость доступа к данным, а также для хранения документов, которые не обладают определениями DTD (но являются правильными XML-документами). Пример первого типа был описан в разделе 5.3 (Хранение данных в истинной XML-базе данных). А вот вещи второго рода плохо хранятся в не-XML-ных базах данных. То есть, истинная XML-база данных может принять, сохранить и понять любой XML-документ,

даже не обладающей полной конфигурацией (например, без DTD). Перенос данных в XML-документе в реляционную или объектно-ориентированную базу данных требует, чтобы вы сначала создали отображение и схему базы данных. Отсутствие первичной конфигурации является преимуществом в приложениях типа поисковых машин, в которых ко всем документам не применимо одно или несколько каких-то определений DTD.

6.2.1 Что такое "истинная XML-база данных"?

Термин "истинная XML-база данных" впервые приобрел известность в время маркетинговой компании Tamino, истинной XML-базы данных от Software AG. Может быть из-за успеха этой компании, этот термин вошел в общее употребление среди компаний, производящих подобные продкуты. Однако, будучи маркетинговым термином, он никогда не получал формального технического определения.

Одно из возможных определений истинной XML-базы данных таково (оно было выработано участниками почтовой рассылки XML:DB[17]):

- Она задает (логическую) модель XML-документа - а не данных в документе - и сохраняет и извлекает документ в соответствии с этой моделью. Как минимум, модель должна включать в себя элементы, атрибуты, секции PCDATA, и порядок документа. Примерами таких моделей является модель данных XPath, XML Infoset, модели, основанные на DOM и событиях в SAX 1.0.
- Фундаментальной единицей (логического) хранения в них является XML-документ, также, как в реляционных базах данных фундаментальной единицей (логического) хранения является один ряд таблицы.
- Не обязательно придерживаться какой-то определенной физической структуры хранения. Например, такая база может строиться на основе реляционной, иерархической или объектно-ориентированной базе данных или использовать собственный формат хранения, например, в виде индексированных и заархивированных файлов.

Первая часть этого определения похожа на определения других типов баз данных, в которых описывается модель, используемая базой данных. Естественно, что истинная XML-база данных может сохранять больше информации, чем содержится в используемой ею модели. Например, она может поддерживать язык запросов на основе модели данных XPath, но хранить документы как текст. В этом случае такие вещи, как секции CDATA и использования сущностей хранятся в базе данных, но не присутствуют в модели.

Вторая часть данного определения утверждает фундаментальной единицей хранения в истинной XML-базе данных XML-документ. Хотя кажется возможным для истинной XML-базы данных эту роль придать фрагментам документа, сегодня это условие выполняется во всех истинных XML-базах данных.

(Фундаментальная единица хранения - это контекст самого низкого уровня, в который встраивается данный кусочек данных и она эквивалентна ряду в реляционной базе. Существование этой единицы не препятствует извлечению еще более мелких единиц данных, таких, как фрагментов документов или отдельных элементов, также и не препятствует комбинированию фрагментов одного документа с фрагментами другого. В реляционных терминах это эквивалентно тому, что существование рядов не препятствует извлечению значения отдельной колонки или создания новых рядов из существующих.)

Третья часть данного определения утверждает, что лежащий в основе работы базы формат хранения не важен. Это действительно так - аналогично утверждению, что физический формат хранения, применяемый в реляционной базе данных не имеет значения для "реляционности" базы.

17: http://www.xmldb.org/projects.html

6.2.2 Архитектура истинной XML-базы данных

Возможная архитектура истинных XML-баз данных делится на две широкие категории: текстовые и модельные.

6.2.2.1 Текстовые истинные XML-базы данных

В **текстовых истинных XML-базах данных** XML хранится как текст. Это может быть файл в файловой системе, BLOB в реляционной базе данных, или любой другой текстовый формат. (В этом отношении реляционная база данных, в которую встроено распознающая XML обработка колонок типа CLOB (Character Large OBject) является, фактически, истинной XML-базой данных.)

Общим для всех текстовых истинных XML-баз данных является индексирование, которое позволяет машине запросов легко переходить в любую точку любого XML-документа. Это дает таким базам огромное увеличение скорости доступа при извлечении целого документа или фрагментов документов. Причина в том, что база данных производит один поиск индекса, один раз позиционирует читающую головку на диске и извлекает целый документ или его фрагмент за один цикл чтения, поскольку необходимый фрагмент хранится в виде последовательных байтов на диске. И напротив, воссоздание документа из кусочков, как это делается в реляционной базе и некоторых модельных истинных XML-базах, требует многократных поисков индекса и многократных чтений с диска.

В этом смысле текстовые истинные XML-базы схожи с иерархическими базами данных в том плане, что они имеют большую производительность, чем реляционные базы при доступе и извлечении данных в соответствии с некоторой заранее известной иерархией. Также, как и иерархические базы данных, текстовые истинные XML-базы будут иметь проблемы с производительностью при извлечении данных в любом другом формате, например, с перевернутой иерархией или только при запросе ее части. Хотя не известно строгого доказательства, доминирование реляционных баз данных, чьи логические указатели позволяют всем запросам одинаковой сложности выполняться с одинаковой скоростью, гововорит о том, что это действительно так.

6.2.2.2 Модельные истинные XML-базы данных

Вторая категория - это **модельные истинные XML-базы данных**. Вместо хранения XML-документа как текста, в них строится внутренняя модель документа и он сохраняется в этой модели. Как именно она сохраняется - зависит от базы данных. Некоторые базы сохраняют ее в реляционной или объектно-ориентированной базе данных. Например, сохранение DOM в реляционной базе данных может потребовать таблиц Elements, Attributes, PCDATA, Entities, и EntityReferences. В других базах используется собственный формат хранения, оптимизированный для их моделей.

(В качестве примера простой модельной истинной XML-базы данных, построенной на основе реляционной базы, смотрите систему, описанную Mark Birbeck в почтовой рассылке XML-L[18] в декабре 1998 года. В системе используется пять таблиц - определение атрибутов, связи между элементами и атрибутами, определение контентной модели, значения атрибутов и

18: http://listserv.heanet.ie/lists/xml-l.html

значения элементов (PCDATA или указателей на другие элементы) и модель, которая включает в себя только элементы, атрибуты, текст и порядок документа. Посмотрите темы "Окончания записей, смешанное содержимое и хранение XML-документов в реляционной базе данных"[19] и "Хранение XML-документов в реляционной базе данных"[20].)

Модельные истинные XML-базы данных строятся на других базах данных и обычно имеют ту же производительность, что и эти базы данных при извлечении документов по очевидной причине: они основываются на этой же системе доступа к данным. Однако, конструкция базы данных, особенно это относится к истинным XML-базам данных, основанных на реляционных базах, имеет значительное пространство для вариаций. Например, база данных, использующая прямое объектно-реляционное отображение DOM может являться системой, которая выполняет отдельные выражения SELECT для извлечения дочерних элементов каждого узла. С другой стороны, в большей части таких баз данных модели хранения и программное обеспечение для получения доступа оптимизировано. Например, Richard Edwards[21] описывает систему для хранения DOM в реляционной базе данных[22], которая может извлекать любой фрагмент документа (и весь документ) с помощью единственного выражения SELECT.

Модельные истинные XML-базы данных, в которых используется собственный формат хранения, обычно имеют производительность, аналогичную текстовым истинным XML-базам данных при извлечении данных в том порядке, в котором они хранятся. Причина в том, что большая часть таких баз данных использует физические указатели между узлами, которые дают при доступе примерно такую же производительность, что и при извлечении текста. (Что быстрее - зависит от выходного формата. Текстовые системы очевидно быстрее при извлечении документов как текста, а модельные системы быстрее при возвращении документов в виде деревьев DOM, поскольку их модель легко отображается в DOM.)

Как и текстовые истинные XML-базы, модельные базы, скорее всего, будут испытывать проблемы с производительностью при извлечении и возвращении данных в любом другом формате, отличном от того, в каком они хранятся, например, с перевернутой иерархией или только ее частью. Будут ли они здесь быстрее или медленнее, чем текстовые XML-базы - не ясно.

6.2.3 Возможности истинных XML-баз данных

В этом разделе мы кратко обсудим несколько возможностей, имеющихся в истинных XML-базах данных. Вы узнаете не только о возможностях, имеющихся сегодня, но и о тех, которые можно ожидать в ближайшем будущем.

6.2.3.1 Коллекции документов

Многие истинные XML-базы данных поддерживают понятие коллекции. Оно имеет схожую роль с понятием таблицы в реляционной базе или директории в файловой системе. Предположим, например, что вы используете истинную XML-базу данных для хранения торговых заказов. В этом случае вы могли бы определить коллекцию торговых заказов, так что запрос по торговым заказам мог бы быть ограничен по документам этой коллекции.

Другой пример: пусть вы храните описание всех продуктов компании в истинной XML-базе данных. В этом случае, вы могли бы задать иерархию коллекций. Например, вы могли бы

19: http://listserv.heanet.ie/cgi-bin/wa?A1=ind9812&L=xml-l#26
20: http://listserv.heanet.ie/cgi-bin/wa?A1=ind9812&L=xml-l#55
21: http://www.sees.bangor.ac.uk/~rich/index.html
22: http://www.sees.bangor.ac.uk/~rich/research/papers/uwb_rge_IDEAL2000.pdf

иметь коллекцию для каждого продукта, а внутри этой коллекции иметь коллекции глав каждого описания.

Могут ли быть вложены друг в друга коллекции - зависит от конкретной базы данных.

6.2.3.2 Языки запросов

Почти все истинные XML-базы данных поддерживают один или несколько языков запросов. Наиболее популярным является XPath (расширенный для выполнения запросов по многим документам) и XQL, хотя есть также несколько примеров и собственный языков. Выбирая истинную XML-базу данных, следует проверить, удовлетворяет ли поддерживаемый ею язык запросов вашим нуждам, поскольку они могут варьировать от полнотекстовых поисков до необходимости рекомбинации фрагментов из различных документов.

Вероятно, в будущем большая часть истинных XML-баз данных будет поддерживать XQuery, предложенный W3C.

6.2.3.3 Обновления и удаления

Истинные XML-базы данных имеют несколько стратегий обновления и удаления документов - от простой замены и удаления существующих документов до модификации через дерево DOM и языков, которые определяют, как производится модификация фрагментов документа. В ближайшем будущем возможности обновления будут, скорее всего, оставаться частичными, поскольку эта проблема лишь недавно (в июне 2001 года) была поставлена перед индустрией и академическими кругами.

6.2.3.4 Транзакции, блокировка (locking) и конкуренция

Виртуально все истинные XML-базы данных поддерживают транзакции (и вероятно поддерживают откаты). Однако, блокировка часто делается на уровне целого документа, а не на уровне его фрагментов, так что проблема много-пользовательской конкуренции решается плохо. Проявится ли серьезно эта проблема - зависит от приложения и того, что считается "документом".

Например, если руководство для пользователя было разбито на отдельные главы, каждая из которых является отдельным документом, тогда проблема конкуренции будет, вероятно, несущественна, поскольку вряд ли вероятна ситуация, когда два автора одновременно пытаются обновлять одну и ту же главу. С другой стороны, если все данные о заказчиках целой компании хранятся в одном документе (плохой дизайн), блокировка на уровне документа вызовет серьезные проблемы.

В будущем, вероятно, большая часть истинных XML-баз данных будет обеспечивать блокировку на уровне фрагментов.

6.2.3.5 Программные интерфейсы (API)

Почти все истинные XML-базы данных предоставляют программные API. Они обычно выпол-

нены в виде ODBC-подобных интерфейсов, имеющих методы присоединения к базе данных, работы с метаданными, выполнения запросов и получения их результатов. Результаты обычно возвращаются в виде строки XML, в виде дерева DOM, SAX-парсера или в формате XMLReader над возвращаемым документом. Запросы могут выполняться по множественным документам, кроме того, доступны методы итераций через результирующий набор запроса.

Одна из возможностей, которая кажется интересной в дата-центричных приложениях (и которая имеется по крайней мере в одной истинной XML-базе) - это способность привязывать переменные приложения к заданным элементам или атрибутам возвращаемых переменных. Это освобождает приложение от необходимости парсить документ,чтобы построить внутренние объекты данных, и похоже, эта способность будет все более широко реализовываться по мере развития технологий привязки XML-данных.

Хотя большинство истинных XML-баз данных предлагают собственные API, XML:DB.org[23] разработал независимый от производителей API базы данных XML[24]. Он был реализован в нескольких истинных XML-базах данных и может быть также реализован и в не XML-ных базах данных. Независимо от того, этот ли API или какой-то другой станет стандартом, последующее широкое принятие такого API кажется неизбежным.

Большинство истинных XML-баз данных также предлагают возможность выполнения запросов и возвращения результатов через HTTP.

6.2.3.6 Обратимость

Одной из важных возможностей истинных XML-баз данных является **обратимость** XML-документов. То есть, если вы сохраните XML-документ в истинной XML-базе, а затем извлекете его из базы, это будет "тот же самый" документ. Это очень важно для документо-центричных приложений, для которых такие вещи, как секции CDATA, использование сущностей, комментарии и процессуальные инструкции являются интегральной частью документа. Также это важно и во многих юридических или медицинских приложениях, которые по закону должны хранить точные копии документов.

(Обратимость менее важна для дата-центричных приложений, поскольку в них имеет значение обычно только элементы, атрибуты, текст и иерархический порядок. Все программное обеспечение, созданное для переноса данных между XML-документами и базами данных сохраняют обратимость этих элементов. Оно также в ограниченном числе случаев может обращать и порядок следования соседних элементов (последовательность, в которой следуют элементы и секции PCDATA у одного родительского элемента) - когда это важно для дата-центричных приложений. Однако из-за того, что эти программы не обращают порядок соседних элементов в целом, они не могут обращать процессуальные инструкции, комментарии и физическую структуру (ссылки на сущности, секции CDATA и т.д.), и плохо подходят для документо-центричных приложений.)

Все истинные XML-базы данных могут обращать документы на уровне элементов, атрибутов, секций PCDATA и внутреннего порядка документа. Насколько хорошо они выполняют обращение, зависит от базы данных. Как правило, текстовые истинные XML-базы совершенно точно обращают документы, а а модельные базы обращают XML-документы на уровне своей модели документа. Если модель документа минимальна, обращение выполняется на меньшем уровне, чем канонический XML.

Поскольку нужный вам уровень обращения целиком зависит от вашего приложения, ваш выбор из многих баз данных может свестись лишь к некоторым.

23: http://www.xmldb.org/
24: http://www.xmldb.org/xapi/index.html

6.2.3.7 Внешние данные

Некоторые истинные XML-базы данных могут вставлять в документы данные, которые хранятся во внешней базе данных. Обычно эти данные извлекаются из реляционной базы данных с помощью ODBC, OLE DB, или JDBC и моделируются с помощью табличного отображения или объектно-реляционного отображения. Являются ли эти данные живыми - то есть, отражаются ли модификации документа в истинной XML-базе во внешней базе данных - зависит от истинной XML-базы данных. В конечном итоге, большая часть истинных XML-баз данных будет, вероятно поддерживать живые внешние данные.

6.2.3.8 Индексы

Большинство XML-баз данных поддерживает индексирование значений элементов и атрибутов. Как и в не-XML-ных базах, индексирование используется для увеличения скорости выполнения запросов.

6.2.3.9 Внешнее хранение сущностей

Трудный вопрос при хранении XML-документов - что делать с внешними сущностями. То есть, следует ли их расшифровывать и сохранять их значение в основном документе или на их месте оставлять ссылки на сущности? На этот вопрос нет единого ответа.

Пусть, например, документ содержит внешнюю общую сущность, которая вызывает CGI-программу для получения сообщения о текущей погоде. Если документ используется как веб-страница, предоставляющая информацию о погоде, было бы ошибкой расшифровывать ссылку на сущность, поскольку веб-страница не смогла бы возвращать живые данные. С другой стороны, если документ является частью коллекции документов, хранящих исторические сводки погоды, было бы напротив, ошибкой **не** расшифровывать ссылку на сущность, поскольку в противном случае документ будет содержать не исторические данные, а всегда текущую сводку.

Другой пример: возьмите описание продукта, состоящее из ничего иного, а только ссылок на внешние сущности, которые указывают на главы описания. Если некоторые из этих глав использованы в других документах, таких, как описание других моделей того же самого продукта, было бы ошибкой расшифровывать ссылку на сущность, поскольку это вызвало рассогласованность копий одной и той же главы.

В истинных XML-базах данных нет общего решения этой проблемы. В идеале, должна быть предусмотрена возможность указания расшифровывать или нерасшифровывать ссылку на сущность в каждом случае.

6.2.4 Нормализация, референциальная интегрированность и масштабируемость

У многих людей, привыкших работать с реляционными базами данных, истинные XML-базы данных поднимают несколько сомнительных проблем, особенно относящихся к вопросам хра-

нения данных (а не документов). Они будут обсуждаться в следующих разделах.

6.2.4.1 Нормализация

Нормализацией называется процесс создания схемы базы данных, в которой каждый кусочек данных представлен лишь один раз. Нормализация дает несколько очевидных преимуществ, таких, как уменьшение требуемого дискового пространства и устранение возможности разсогласования данных, что может произойти, когда один и тот же кусочек данных хранится в нескольких местах. Это - один из краеугольных камней реляционной технологии и горячая точка при обсуждениях хранения данных в истинных XML-базах.

Прежде, чем мы продолжим разговор о нормализации, полезно заметить, что это не является проблемой для многих документо-центричных документов. Например, рассмотрим коллекцию документов, описывающих продукты компании. Во многих подобных коллекциях лишь небольшое количество данных одинаково для всех документов - записи об авторских правах, контактные данные и телефонные номера, логотип и т.д. - и эти данные настолько малы по отношению к основным, что мало кому в голову приходит нормализовывать их. (С другой стороны, другие наборы документов имеют значительное пересечение между собой и требуют нормализации.)

Как и в реляционных базах данных, никто не станет вам навязывать нормализацию данных в истинных XML-базах. То сеть, вы можете создать плохую систему хранения данных в XML-базах также, как и в реляционных. Таким образом, важно продумать структуру своих документов, прежде, чем сохранять их в истинной XML-базе. (Здесь XML-базы имеют только одно преимущество пред реляционными - поскольку истинные XML-базы данных не имеют схем базы данных, вы можете хранить в базе одновременно несколько схожих документов с различными схемами. И хотя вам все равно потребуется менять запросы и конвертировать существующие элементы - а это нетривиальное занятие - это может облегчить процесс переноса.)

Нормализация данных в XML-базах обеспечивается также, как и в реляционных базах данных: документы нужно сконструировать так, чтобы данные не дублировались. Единственная разница в том, что XML поддерживает многозначные свойства, в то время как большинство реляционных баз данных - нет. Это позволяет "нормализовать" данные в истинной XML-базе так, как этого нельзя сделать в реляционной базе.

Например, рассмотрим торговый заказ. Он состоит из заголовка, например, номера заказа, даты и номера заказчика, а также одного или нескольких внутренних пунктов, которые содержат номер части заказа, количество и общую стоимость. В реляционной базе данных информацию заголовка нужно хранить в отдельной таблице, поскольку одному заголовку может соответствовать несколько внутренних пунктов. В истинной XML-базе данных эта информация может содержаться в единственном документе безо всякой избыточности, поскольку иерархическая природа XML позволяет родительскому элементу иметь много дочерних элементов.

К сожалению, в реальных условиях, нормализацию выполнить не так просто. Например, что произойдет, если вы захотите, чтобы торговые заказы содержали информацию о заказчике, например, контактные имена, адреса доставки и отправки счетов? Тут есть два варианта. Во-первых, вы можете дуплицировать информацию о заказчике в каждом торговом заказе, что приводит к избыточности данных и всем связанным с этим проблемам. Во-вторых, вы можете хранить информацию о заказчиках отдельно и либо предоставить XLink-ссылку на документ с торговым заказом, либо объединить два документа, когда данные будут запрашиваться. Тогда требуется, чтобы поддерживался язык XLink (в большинстве случаев его нет, хотя под-

держка планируется) либо чтобы язык запросов поддерживал объединения (тоже не всегда так).

В практике не существует ясных ответов. Реальные реляционные данные часто не нормализуются ради более высокой производительности. так что наличие не-нормализованных XML-данных может быть не таким плохим, как это звучит - но тут нужно ваше решение. Если вы храните документо-центричные документы и можете нормализовать их до некой разумной степени - например, хранение глав или процедур в виде отдельных документов и объединять их для того, чтобы выдавать пользователю конечный документ - тогда истинная XML-база данных может стать правильным решением, особенно потому, что она предоставит возможности, которых нет в других базах данных, например, выполнение запросов на XML-языке. Если вы храните дата-центричные документы и истинная XML-база увеличивает производительность вашего приложения или позволяет хранить полу-структурированные данные, которые трудно хранить в другой базе данных, правильно ее использовать. Но если вы обнаружите, что по сути, внутрь вашей XML-базы данных встроена реляционная база, тогда следует спросить себя, почему не использовать прямо реляционную базу данных.

6.2.4.2 Референциальная интегрированность

Референциальная интегрированность близко соотносится с нормализацией. **Референциальной интегрированностью** называют действительность указателей на соответствующие данные и она является необходимой частью поддержания согласованного состояния базы данных: не будет ничего хорошего, если торговые заказы будут содержать номер заказчика, а соответствующей записи о заказчике не окажется. Отдел доставки не узнает, куда отправлять заказ, а отдел расчетов не узнает, кому отправлять счет.

В реляционной базе данных референциальная интегрированность означает, что внешние ключи указывают на верные первичные ключи - то есть, проверку, что ряд первичного ключа соответствует какому-то внешнему ключу. В истинной XML-базе референциальная интегрированность означает, что XLinks или другой собственный механизм ссылок указывает на верный документ или фрагмент документа.

В настоящее время лишь некоторые истинные XML-базы данных обеспечивают референциальную интегрированность. Причина в том, что большая часть XML-баз данных сегодня не поддерживают механизм ссылок-связей, так что просто не существует референций, чью интегрированность можно было бы проверить. В результате этого приложения, которые используют XLinks или другие механизмы ссылок, сами должны обеспечивать эту интегрированность.

Вероятно, в будущем многие истинные XML-базы данных будут поддерживать механизмы ссылок и референциальную интегрированность.

6.2.4.3 Масштабируемость

Масштабируемость лежит довольно далеко от сферы моего опыта, так что многое из последующего - спекуляция. Говоря в общем, я думаю, что истинные XML-базы в одних средах масштабируются очень хорошо, а в других - очень плохо.

Как иерархические и реляционные базы данных, истинные XML-базы используют для начального обнаружения данных индексы. Это означает, что локализация документов и их фрагментов зависит только от размера индекса, а не от размера документов или их коли-

чества, и что истинные XML-базы данных могут локализовывать начало документа или фрагмента настолько же быстро, как и другие базы данных, использующие ту же технологию индексации. Таким образом, в этом отношении XML-базы могут масштабироваться настолько же легко, как и другие базы данных.

Как и другие иерархические базы данных (но в отличие от реляционных), многие истинные XML-базы данных физически связывают соотносящиеся данные. В частности, текстовые XML-базы данных физически группируют соотносящиеся данные, а модельные XML-базы данных используют собственные системы хранения, в которых часто применяются физические указатели для группировки соотносящихся данных. (модельные XML-базы построены на реляционных базах данных и как сами реляционные базы, используют логические связи.)

Поскольку физические связи при навигации действуют быстрее, истинные XML-базы данных, например, иерархические базы, могут извлекать данные быстрее, чем реляционные базы данных. Таким образом, они должны хорошо масштабироваться в отношении извлечения данных. На самом деле в этом отношении они должны масштабироваться даже лучше реляционных баз, поскольку масштабируемость зависит от одного начального поиска индекса, а не от многих поисков индексов, требующихся в реляционной базе. (Следует заметить, что реляционные базы данных также предлагают физические связи данных в форме кластеризованных индексов. Однако такие связи применяются к отдельным таблицам, а не ко всей иерархии.)

К сожалению, масштабируемость ограничена. Как и в иерарических базах данных, в XML-базах физические связки действуют только в перделах отдельных иерархий. То есть, извлечение данных из той иерархии, в которых они находятся происходит очень быстро, но извлечение тех же данных из другой иерархии происходит медленно. Например, пусть информация о заказчике хранится внутри каждого документа с торговым заказом. Извлечение документов с торговыми заказами будет происходить очень быстро, поскольку это именно тот порядок, в котором хранятся данные. Извлечение же других представлений данных, например, списка торговых заказов для конкретного заказчика, будет выполняться гораздо медленнее, поскольку физические связи тут не действуют.

Чтобы смягчить эту проблему, истинные XML-базы очень интенсивно используют индексы, часто индексируются все элементы и атрибуты. Хотя это помогает уменьшить время доступа, поддержка таких индексов может оказаться накладной. Это не проблем для сред, в которых документы только читаются, но может быть проблемой в средах с активными транзакциями.

Наконец, истинные XML-базы гораздо хуже, чем реляционные масштабируются в поиске неиндексированных данных. В этом случае и истинные XML-базы и реляционные должны искать данные линейно, но для XML-баз данных ситуация гораздо хуже, поскольку в них нормализация менее полна. Например, пусть вы хотите найти все торговые заказы, с определенной датой и эти даты не индексированы. В реляционной базе данных это потребует чтения всех значений колонки OrderDate. В XML-базе данных это потребует чтения всех торговых заказов целиком и поиск по элементам <OrderDate>. Читается не только содержание каждого элемента <OrderDate>, но читается содержимое всех элементов. Что еще хуже, в текстовых XML-базах текст сначала надо будет отпарсить и конвертировать в формат даты, прежде, чем его может будет сравнить с датой, которую мы ищем.

Так является ли масштабируемость для вас проблемой? Это зависит только от вашего приложения. Если вашему приложению нужны данные в той форме, в какой они хранятся, тогда истинная XML-база будет хорошо масштабироваться. Это годится для многих документоцентричных документов. Например, документы, содержащие описания продуктов, почти всегда извлекаются целиком. С другой стороны, если в вашем приложении нет какого-то одного предпочтительного представления данных, масштабируемость может стать проблемой.

Этим мы заканчиваем описание истинных XML-баз данных. В следующих двух разделах мы

обсудим два специализированных типа истинных XML-баз данных: строгие DOM-базы и системы управления содержанием.

6.3 Строгие DOM-базы (PDOM)

Строгие DOM-базы или **PDOM** - это специализированный тип истинных XML-баз данных, в которых реализована DOM на основе некоего вида строгого хранения. В отличие от многих XML-баз, которые возвращают данные в виде дерева DOM, дерево DOM, возвращаемое базой PDOM - живое. То есть, изменения, сделанные в дереве DOM отражаются прямо в базе данных. (Происходят ли эти изменения немедленно или в результате вызова специального метода - зависит от базы.) В большинстве же XML-баз данных дерево DOM возвращается приложению как копия и изменения в базе данных делаются либо через XML-язык модифицирования либо заменой всего документа.

Поскольку деревья PDOM - живые, эти базы данных обычно локальны. То есть, они расположены в том же процессуальном пространстве, что и приложение, или по крайней мере, на той же самой машине, хотя это не является необходимым требованием. Это нужно для эффективности - работа PDOM с удаленной базой данных требует частых обращений к удаленному серверу.

PDOM играют ту же самую роль для DOM-приложений, что и объектно-ориентированные базы данных для объектно-ориентированных приложений: они предоставляют строгое хранение данных приложения, а также служат виртуальной памятью для такого приложения. Последнее особенно важно для DOM-приложений, которые работают с большими документами, поскольку деревья DOM могут легко превысить размер XML-документов в десять раз. (Реальное увеличение размера зависит от среднего размера текста в документе - у документов, содержащих в среднем небольшое количество текста оно еще выше.)

6.4 Системы управления содержанием

Системы управления содержанием - другой специализированный тип истинных XML-баз данных. Они предназначены для управления созданными человеком документами, например, руководствами и статьями, создаются они на основе истинной XML-базы данных. База данных обычно скрыта от пользователя за интерфейсом, который предоставляет такие возможности, как:

- Контроль версий и доступа
- Поисковые машины
- Редакторы XML/SGML
- Машины публикации, например, в газету, на CD, или в Интернете
- Отделение стиля от содержания
- Расширение через скриптовые языки или программирование
- Интеграция данных в базе данных

Термин "система управления содержанием" в отличие от системы управления документами означает, что такие системы обычно позволяют разбивать документы на дискретные куски содержания, например, примеры, процедуры, главы, врезки, или метаданные: имя авторов, дата модификации и номера документа, а не управляют документом как целым. Это не только облегчает координацию работы многих авторов над одним и тем же документом, но и позволяет составлять совершенно новый документ из существующих компонентов.

7.0 XML Database Products

Обновляемый список баз данных, которые можно использовать для работы с XML, вы найдете на странице XML Database Products[25].

8.0 Дополнительные ссылки

На странице XML / Database Links[26] вы найдете ссылки на различные ресурсы по XML и базам данных, включая ресурсы по программному обеспечению и другие статьи.

9.0 Комментарии и обратная связь

Пожалуйста, направляйте свои комментарии и обратную связь Ronald Bourret по адресу rpbourret@rpbourret.com[27]. Прошу иметь в виду, что я часто бываю в разъездах и могу задержаться с ответом на пару-тройку недель.

Благодарю Michael Champion, John Cowan, Marc Cyrenne, Marc de Graauw, Kelvin Ginn, Ingo Macherius, Lars Martin, Nick Leaton, Evan Lenz, Michael Rys, Walter Perry, Kimbro Staken, Jim Tivy, Phillipe Vauclair, Dylan Walsh, Irsan Widarto, и других за их комментарии и терпение.

Developed by Metaphor (c) 2002

25: http://www.rpbourret.com/xml/XMLDatabaseProds.htm
26: http://www.rpbourret.com/xml/XMLDBLinks.htm
27: mailto:rpbourret@rpbourret.com