Лекция 17

SQL и XML

XML (Extensible Markup Language — расширяемый язык разметки) представляет собой одну из наиболее важных новых технологий, возникшую в результате эволюции Интернета и Веб. XML является стандартным языком для представления *структурированных данных*. В свою очередь, SQL — стандартный язык для определения, доступа и обновления структурированных данных, хранящихся в реляционных базах данных. Наличие связи между XML и SQL представляется совершенно очевидным. Естественно встает вопрос о том, *какова* эта связь и конфликтуют ли эти две технологии или, напротив, дополняют друг друга? Ответ — понемногу и того, и другого. В этой главе вы найдете обзор азов XML и узнаете об эволюции взаимоотношений XML и SQL, а также об интеграции XML в основные SQL-продукты.[1]

Что такое XML

Как следует из имени XML, он представляет собой *язык разметки*. У него много общего с его знаменитым родственником — языком разметки гипертекста (HyperText Markup Language, HTML), который приобрел популярность как базовая веб-технология. *Разметка* документа — методика столь же древняя, как и книгопечатание. При подготовке к печати сложного документа, такого как книга, газета или журнал, приходится думать о двух логически связанных частях. Это *содержимое* документа, которое обычно состоит из текста и графики и определяет его смысл, и *структура* документа (заголовки, подзаголовки, абзацы, подписи), которая вместе с форматированием (шрифты, отступы, схемы страниц) помогает организовать содержимое и представить его читателю в удобном и понятном виде. С самых первых дней книгопечатания редакторы использовали определенные

---

[1] *При желании более детально ознакомиться с XML обратитесь, например, к книге Д. Хантер, Д. Рафтер и др.* XML. Базовый курс, 4-е издание. — *М.: ООО "И.Д. Вильямс", 2009.* — Примеч. ред.

символы разметки и форматирования, которые определяли структуру и форматирование содержимого документа при печати.

При появлении на сцене компьютеризированных издательских систем команды разметки в содержимом документа превратились в инструкции для издательских программ. У каждого типа издательского программного обеспечения имеется свой набор команд разметки, что затрудняет переносимость документов между системами. Как способ стандартизации языков разметки был разработан и принят в качестве стандарта ISO стандартный обобщенный язык разметки (Standard Generalized Markup Language, SGML). Говоря более строго, SGML представляет собой *метаязык* для определения конкретных языков разметки. Его изобретатели понимали, что ни один язык разметки не в состоянии удовлетворить всем возможным требованиям, но все языки разметки имеют общие элементы. Можно создать семейство тесно связанных языков разметки, стандартизируя эти общие элементы. HTML представляет собой один из таких языков разметки, разработанный специально для использования гипертекста при связывании документов. XML — еще один такой язык, ориентированный на строгую типизацию и четкое структурирование содержимого документа. Общие SGML-корни делают HTML и XML родственными языками и являются причиной их подобия.

И HTML, и XML являются рекомендациями Консорциума W3C (World Wide Web Consortium, W3C), определяемыми спецификациями, которые разработаны, обсуждены и опубликованы W3C. W3C — независимый, некоммерческий консорциум, целью которого является разработка и распространение стандартов для работы в Интернете и Веб. Рекомендации W3C имеют статус "официально одобренных"; это означает, что W3C поддерживает и рекомендует их применение. Благодаря этому HTML и XML являются промышленными стандартами, не зависящими от конкретных производителей.

HTML был первым языком на основе SGML, получившим всемирное распространение. Содержимое огромного количества веб-страниц практически каждого веб-сайта в Интернете представляют собой HTML-документы. В HTML-документе специальные элементы разметки, которые называются дескрипторами, указывают графические элементы (например, такие как кнопки), выводимые веб-браузерами. Дескрипторы также описывают гипертекстовые ссылки на другие документы, которые браузер должен загружать при щелчке на кнопке. Другие дескрипторы идентифицируют графические элементы, которые должны быть вставлены в HTML-текст при его выводе.

Стремительное развитие Веб в 1990-х годах привело к тому, что HTML был быстро адаптирован для вывода гораздо более богатого содержимого форматированных веб-страниц. Вскоре были созданы дескрипторы HTML для управления форматированием веб-страниц, использования курсивного или полужирного шрифта, центрования и отступов и т.п. В некоторых случаях эти дескрипторы были уникальны для конкретного веб-браузера, такого как Netscape или Microsoft Internet Explorer. Со временем основным назначением разметки на HTML-страницах стало форматирование и представление информации. Преимуществом этого подхода стало то, что страницы стали выводиться одинаково на разных уст-

Важной исходной целью SGML было то, чтобы заданный логический элемент, такой как заголовок страницы или раздела, мог быть единообразно идентифицирован в сотнях документов (например, в сотнях страниц веб-сайта). После этого обеспечить единообразность вывода можно просто директивой наподобие "выводи все заголовки разделов синим полужирным шрифтом Times New Roman 16-го размера". Однако, к сожалению, авторы веб-страниц проявляют тенденцию к явной разметке каждого элемента, такого как заголовок подраздела, с применением детальных инструкций форматирования. Это быстро приводит к несогласованности, и, что еще хуже, изменение форматирования требует редактирования сотен отдельных страниц — вместо одного изменения, которое будет воспринято всеми страницами.

Одной из основных движущих сил развития XML было восстановление подхода к разметке на уровне логики, а не на уровне внешнего представления. XML реализует гораздо более строгие правила, связанные со структурой документа, чем HTML. Большинство его компонентов и возможностей непосредственно связано с представлением логической структуры документа. Имеются сопутствующие стандарты для определения типов документов, такие как XML Schema, которые позволяют еще более усилить эту направленность XML.

В дополнение к иерархии элементов, на рис. 25.1 приведены примеры *атрибутов*, еще одной фундаментальной структуры XML. Атрибут связан с конкретным элементом XML и описывает некоторые его характеристики. Каждый атрибут имеет имя и значение. На рис. 25.1 элемент `chapter` имеет атрибут `chapNum`, значением которого является номер главы, связанный с определенным содержимым. Элемент `chapter` имеет другой атрибут — `revStatus`, значение которого указывает, является ли текст черновым вариантом, исправленным или окончательным. Отдельные элементы `<header>` на рис. 25.1 также имеют атрибут `hdrLevel`, указывающий, принадлежит ли данный заголовок к верхнему уровню (уровень 1) или нижнему (уровни 2 или 3).

Первая строка XML-документа на рис. 25.1 указывает, что перед вами документ XML 1.0. Каждая прочая часть документа описывает структуру элемента, его содержимое или атрибуты элементов. XML-документы могут быть и значительно более сложными, но эти фундаментальные компоненты являются важной частью взаимодействия XML с базами данных. Обратите внимание на чувствительность имен элементов и атрибутов к регистру. Элемент с именем `bookPart` и именем `bookpart` — это разные элементы. В этом существенное отличие от соглашения SQL для имен таблиц и столбцов, которые обычно не чувствительны к регистру.

На рис. 25.1 не показана одна дополнительная возможность XML, весьма полезная на практике. Если у элементов нет содержимого, а есть только атрибуты, то конец элемента может быть указан в той же паре угловых скобок, что и его начало, — при помощи косой черты непосредственно перед закрывающей угловой скобкой. При использовании этого соглашения элемент

```
<figure figNum="5-1"></figure>
```

с рис. 25.1 может быть записан как

```
<figure figNum="5-1"/>
```

Спецификация XML определяет ряд правил, которым должны следовать все XML-документы. Она указывает, что элементы в XML-документе должны быть строго вложенными один в другой. Закрывающий дескриптор более низкоуровневого элемента должен располагаться до закрывающего дескриптора элемента более высокого уровня, содержащего указанный низкоуровневый элемент. Стандарт также указывает, что атрибут должен иметь уникальное для данного элемента имя; не допускается наличие у одного элемента двух атрибутов с одним и тем же именем. XML-документы, соответствующие всем описанным правилам, называются *корректными* (well-formed) XML-документами.

XML для данных

Хотя своими корнями XML уходит в документы и их обработку, он вполне применим для представления структурированных данных, с которыми обычно имеют дело приложения для обработки данных. На рис. 25.3 показан типичный XML-документ — очень упрощенный заказ. Он существенно отличается от документа на рис. 25.1, но его ключевые компоненты — те же. Вместо `chapter` элемен-

том верхнего уровня является purchaseOrder. Его содержимым, как и содержимым элемента chapter, являются подэлементы — customerNumber, orderNumber, orderDate и orderItem. Элемент orderItem, в свою очередь, состоит из других подэлементов.

XML и SQL

Происхождение из SGML дает XML несколько уникальных и полезных характеристик, имеющих аналоги в языке SQL.

- **Описательный подход**. XML говорит о том, чем является каждый элемент документа, а не о том, как его обрабатывать. Вы можете вспомнить, что это же является характеристикой SQL, который сосредоточивает свое внимание на том, какие данные запрошены, а не на том, как их получить.
- **Составные блоки**. Документы XML построены из небольшого количества базовых блоков, включающих две фундаментальные концепции, *элементы* и *атрибуты*. Имеются определенные (хотя и не совершенные) параллели между элементами XML и таблицами SQL и между атрибутами XML и столбцами SQL.
- **Типы документов**. XML определяет и проверяет документы на согласованность с определенными типами, которые соответствуют документам реального мира — таким, например, как заказ или отпускная записка. И здесь вновь имеется параллель с SQL, где таблицы представляют различные сущности реального мира.

Хотя между XML и SQL имеется много сходства, они достаточно сильно отличаются друг от друга.

- **Ориентация на документы или данные**. Базовые концепции XML связаны с типичными структурами документов. XML "текстоцентричен" и разграничивает содержимое (элементы документа) и его характеристики (атрибуты). Базовые концепции SQL происходят из типичных структур записей для обработки данных. SQL ориентирован на данные, обладает широким диапазоном типов данных (в их бинарных представлениях), а его структуры (таблиц и столбцов) ориентированы на содержимое (данные). Несоответствие фундаментальных моделей XML и SQL может привести к определенным конфликтам или сложному выбору при их совместном использовании.
- **Иерархическая и табличная структуры**. Естественной структурой XML является иерархическая, отражающая иерархию элементов в большинстве типов документов. (Например, книга содержит главы, главы состоят из разделов, разделы включают заголовок, абзацы и рисунки.) Эти структуры гибки и изменчивы. Один раздел может содержать пять абзацев и один рисунок, другой — три абзаца и два рисунка, третий — шесть абзацев и ни одного рисунка. Структуры же SQL табличные, а не иерархичные, и отражают типичные для приложений по обработке данных записи. Структуры SQL достаточно жесткие. Каждая строка таблицы содержит одни и те же столбцы, в одном и том же порядке. Каждый столбец в каждой строке содержит данные одного и того же типа. Здесь нет необязательных столбцов — каждый столбец должен присутствовать в каждой строке. Эти отличия могут привести к конфликтам при совместном использовании XML и SQL.
- **Объекты и операции**. Основное предназначение XML — *представление* объектов. Если взять значимую часть XML-кода и спросить "что она представляет", то ответом будет объект — например, абзац, заказ, адрес

Элементы и атрибуты

Реляционная модель предлагает только один способ представления значений данных в базе данных — как значений отдельных столбцов в отдельных строках таблицы. Модель XML-документа предлагает два способа представления данных.

- **Элементы**. Элемент XML-документа имеет содержимое, и это содержимое может включать значения данных в форме текста этого элемента. При таком представлении значения данных являются фундаментальной частью иерархии XML-документа; иерархия строится из элементов. Кстати, иерархическая древовидная структура приводит к тому, что практики, говоря о коллекции связанных элементов XML-документов, называют ее *лесом*. Часто элемент, содержащий значение данных, является листом в дереве XML-документа; такой элемент является дочерним по отношению к элементу более высокого уровня, но сам дочерних элементов не имеет. Это почти всегда так для элементов, представляющих данные из реляционной базы данных. Однако XML поддерживает *смешанные* элементы, которые содержат объединение текста (содержимого) и других подэлементов.
- **Атрибуты**. Элемент в XML-документе может иметь один или несколько именованных атрибутов, а каждый атрибут имеет текстовое значение. Атрибуты присоединяются к элементу в XML-иерархии, но не к содержимому элемента. Имена различных атрибутов элемента должны быть различными, так что двух атрибутов с одним именем быть не может. Кроме того, XML рассматривает порядок атрибутов элемента как не имеющий значения; они могут появляться в любом порядке. Это отличается от рассмотрения элементов в XML, которые имеют определенную позицию в документе и где различия между первым, вторым и третьим дочерними элементами элемента более высокого уровня существенны.

Наличие двух различных способов представления данных в XML означает, что у вас есть два разных корректных способа для выражения содержимого реляционной базы данных в виде XML. Вот две строки данных

```
ORDER_NUM MFR PRODUCT QTY AMOUNT
--------- --- ------- --- ----------
   112963 ACI   41004   28  $3,276.00
   112983 ACI   41004    3    $702.00
```

которые могут быть представлены в виде следующего XML-документа, в котором для представления значений столбцов использованы элементы.

Сторонники подхода с применением атрибутов приводят следующие аргументы.

- Атрибуты представляют собой фундаментальное соответствие столбцам реляционной модели. Отдельные строки представляют сущности, так что они должны отображаться на элементы. Значения столбцов описывают атрибуты сущности (строки), в которой они находятся; соответственно, в XML они должны быть представлены значениями атрибутов.
- Ограничение на уникальность имен атрибутов в пределах элемента соответствует требованию уникальности имен столбцов в пределах таблицы. Неупорядоченная природа атрибутов соответствует неупорядоченной природе столбцов фундаментальной реляционной модели. (Позиции столбцов используются только как сокращения, для удобства, но не являются важными для отношений между столбцами.)
- Представление с применением атрибутов более компактное, поскольку имена столбцов появляются в XML только один раз, а не два раза — в открывающем и закрывающем дескрипторах. Это практическое преимущество при хранении и передаче XML.

В сегодняшних продуктах XML и SQL можно обнаружить оба подхода. Конкретный выбор зависит от предпочтений автора документа и удобства организации использования XML с SQL. Кроме того, диктовать применение того или иного стиля могут промышленные стандарты обмена информацией с использованием XML.

Использование XML с базами данных

Быстро растущая популярность XML заставляет производителей баз данных внедрять поддержку XML в своих продуктах. Виды этой поддержки различны, но обычно попадают в одну из приведенных ниже категорий.

- **Вывод XML.** XML-документ может легко представить данные в одной или нескольких строках результатов запроса. При такой поддержке СУБД в ответ на SQL-запрос генерирует XML-документ вместо обычных результатов запроса в виде обычных строк/столбцов. Стандарт SQL определяет ряд SQL-функций, которые могут использоваться для преобразования данных, полученных из реляционных таблиц, в XML.
- **Ввод XML.** XML-документ может легко представить данные, вставляемые в одну или несколько новых строк таблицы. Он может также представлять данные для обновления строки таблицы или идентификации удаляемой строки. При такой поддержке СУБД принимает XML-документ в качестве входных данных вместо запроса SQL.
- **Обмен данными XML.** XML представляет собой естественный способ выражения данных, которыми обмениваются СУБД или серверы баз данных. Данные из исходной базы данных трансформируются в XML-документ и передаются целевой базе данных, которая преобразует их в свой формат. Тот же стиль обмена данными полезен для перемещения

данных между реляционными базами данных и приложениями, не являющимися СУБД, такими как программы учета ресурсов предприятия, интеграции приложений предприятия и т.п.

- **Хранение XML.** Реляционная база данных может легко принимать XML-документы (которые представляют собой строки текстовых символов) как части строк переменной длины (VARCHAR) или символьных больших объектов (CLOB). На базовом уровне поддержки XML весь документ становится содержимым одного столбца в одной строке базы данных. Немного более мощная поддержка XML возможна, если СУБД позволяет объявлять столбец с использованием явного типа данных XML. Хотя стандарт ANSI/ISO SQL содержит спецификацию типа данных XML (XML), пока что его не реализует ни один производитель. Однако Oracle, DB2 UDB и SQL Server поддерживают собственные XML-типы в своих частных реализациях.
- **Интеграция данных XML.** Более серьезный уровень интеграции XML возможен в том случае, когда СУБД в состоянии анализировать XML-документ, разбирать его на составные элементы и сохранять отдельные элементы в отдельных столбцах. После этого можно использовать обычный SQL для поиска информации в столбцах, тем самым выполняя поиск в XML-документе. В ответ на запрос СУБД может генерировать XML-документ из сохраненных составных элементов.

Вывод XML

Одна из наиболее простых комбинаций XML и баз данных заключается в применении XML в качестве формата для вывода результатов SQL-запроса. Результаты запроса имеют структурированный табличный формат, который легко транслировать в XML-представление.

Это типичный результат для некоторых популярных СУБД с поддержкой вывода в формате XML. Результаты запроса представляют собой корректный самодостаточный XML-документ. Если передать полученные результаты анализатору XML (они описываются позже в данной главе), то он интерпретирует их как содержащие

- один корневой элемент — `queryResults`;
- четыре подэлемента `row`, являющихся дочерними по отношению к корневому;
- по пять подэлементов у каждого элемента `row`, и в этом случае у каждого элемента `row` имеются все пять подэлементов, в одном и том же порядке.

Предыдущий пример XML-документа, генерируемого непосредственно из базы данных, идеален. На самом же деле поддержка XML у разных реализаций существенно отличается. Например, Microsoft участвовала в разработке спецификации SQL/XML в стандарте ANSI/ISO SQL; однако позже ею было принято решение не реализовывать стандарт, а разработать собственное решение. В SQL Server для вывода результатов в формате XML используется предложение `FOR XML`. Хотя генерируемые результаты не включают корректный XML-документ, они представляют собой корректные XML-элементы, которые легко инкорпорируются в XML-документы.

Возможность получения результата в формате XML может быть значительным преимуществом. Для выполнения дальнейшей обработки полученный вывод можно непосредственно передать в программу, принимающую в качестве входной информации XML-документы. Вывод может быть передан по сети другой системе, а в силу XML-формата его элементы самоописываемы, т.е. *каждая* система или приложение будут интерпретировать результаты запроса одинаково — как четыре строки с пятью элементами каждая. Поскольку вывод представляет собой чисто текстовый формат, он не может быть некорректно интерпретирован из-за отличий в представлении бинарных данных в системе-отправителе и системе-получателе. Наконец, если XML передается при помощи протокола HTTP с использованием стандарта SOAP (Simple Object Access Protocol — простой протокол обращения к объекту), то XML-сообщение может легко проходить через корпоративные брандмауэры и связывать отправляющее приложение в одной компании с приложением-получателем в другой компании.

Вывод в формате XML не лишен и недостатков. Одним из них является большой объем выводимых данных, который превышает объем данных в табличном виде примерно раза в четыре. Если этот вывод должен быть сохранен на диск, потребуется в четыре раза больше дискового пространства. При пересылке в другую систему потребуется в четыре раза больше времени либо, чтобы время осталось неизменным, повышение пропускной способности сети в четыре раза. Это не очень серьезные проблемы для небольших данных, как в рассмотренных примерах, но в случае результатов из тысяч и десятков тысяч строк они могут стать очень существенными, особенно в случае крупных предприятий, когда эти результаты надо будет умножить на сотни работающих приложений.

Простой XML-вывод также теряет некоторую информацию о данных. Символ доллара в случае XML-вывода исчезает, так что из самого содержимого невозможно определить, представляют ли данные денежные суммы и, если да, в какой именно валюте. Возможности XML Schema предоставляют способ получения этой информации, но только ценой еще большего увеличения размера результатов запроса.

Рассмотрим также вопрос стандартизации определения данных. Сам по себе XML является стандартом, но две компании не смогут работать с заказами друг друга в XML-формате до тех пор, пока дескрипторы в соответствующих XML-документах не будут иметь одни и те же имена и определения. Например, если первая компания использует для идентификатора заказываемого товара дескриптор `<productcode>`, а вторая — `<SKU>` (от stock keeping unit), то эти заказы не мо-

гут быть обработаны одним и тем же способом. Одним из решений может служить появление в области XML-документации специализированных стандартов кодирования, таких как HR-XML, разработанный HR-XML Consortium для кадровых служб. Такие стандарты представляют собой аналог стандартов корпоративного обмена данными (Enterprise Data Interchange, EDI), разработанных в последние пару десятилетий.

Функции SQL/XML

Стандарт SQL определяет ряд функций, которые могут использоваться для преобразования данных столбца в XML-элементы. Функция SQL/XML представляет собой просто функцию, которая возвращает значение в XML-виде. Например, запрос может выбирать не-XML-данные (т.е. данные отличных от XML типов), но затем можно форматировать результаты запроса в виде XML-документа для вывода на веб-странице или для передачи другому приложению. Основные функции SQL/XML показаны в табл. 25.1.

Таблица 25.1. Функции SQL/XML

Функция	Возвращаемое значение
`XMLAGG`	Единое XML-значение, содержащее XML-лес, образованный коллекцией строк, каждая из которых содержит единственное XML-значение
`XMLATTRIBUTE`	Атрибут в виде `name=value` в `XMLELEMENT`
`XMLCOMMENT`	Комментарий XML
`XMLCONCAT`	Связанный список XML-значений, создающий единое значение, содержащее XML-лес
`XMLDOCUMENT`	XML-значение, содержащее единый узел документа
`XMLELEMENT`	XML-элемент, который может быть дочерним по отношению к узлу документа, с именем, переданным в качестве параметра
`XMLFOREST`	XML-элемент, содержащий последовательность XML-элементов, полученную из столбцов таблицы, с использованием в качестве имен элементов имена соответствующих столбцов
`XMLPARSE`	XML-значение, образованное путем анализа переданной строки без проверки ее корректности
`XMLPI`	XML-значение, содержащее команду обработки XML
`XMLQUERY`	Результат выражения XQuery (XQuery — подъязык, используемый для выполнения поиска в тексте XML, хранящемся в базе данных; будет рассмотрен позже в данной главе)
`XMLTEXT`	XML-значение, содержащее единственный текстовый узел XML, который может быть дочерним по отношению к узлу документа
`XMLVALIDATE`	XML-последовательность, являющаяся результатом проверки корректности XML-значения

Имеются и другие функции помимо перечисленных в табл. 25.1. Функции SQL/XML могут комбинироваться для создания очень сложных запросов. Доступность конкретных функций варьируется в зависимости от конкретной реализации SQL. Вот простые примеры, которые должны пояснить применение функций `XMLELEMENT` и `XMLFOREST` на практике.

```
SELECT XMLELEMENT("OrderNumber", ORDER_NUM)
  FROM ORDERS
 WHERE ORDER_NUM=112963;

<OrderNumber>112963</OrderNumber>


SELECT XMLFOREST(ORDER_NUM AS "OrderNumber",
                 MFR, PRODUCT, QTY, AMOUNT)
  FROM ORDERS
 WHERE ORDER_NUM=112963;

<OrderNumber>112963</OrderNumber>
<MFR>ACI</MFR>
<PRODUCT>41004</PRODUCT>
<QTY>28</QTY>
<AMOUNT>3276</AMOUNT>
```

Обратите внимание на то, что имена XML-элементов берутся из имен столбцов и переводятся в верхний регистр, как это принято в SQL. Однако, используя псевдонимы столбцов, как это было сделано для столбца ORDER_NUM, можно изменить имена столбцов на другие, которые вам нужны.

Ввод XML

Так же как XML может применяться для представления строки результатов запроса, являющейся выводом базы данных, его можно применять и для представления строки данных, вносимой в базу данных. Для обработки XML-данных СУБД должна проанализировать XML-документ, содержащий вносимые данные, и идентифицировать отдельные элементы данных (представленные элементами или атрибутами). СУБД должна сопоставить имена элементов или атрибутов (обычно с применением имен столбцов) или преобразовать их (с применением схемы конкретной СУБД) в имена столбцов целевой таблицы, которая будет получать новые данные. Концептуально простая инструкция INSERT

```
INSERT INTO OFFICES (OFFICE, CITY, REGION, SALES)
     VALUES (23,'San Francisco','Western',0.00);
```

может быть легко транслирована в эквивалентную гибридную SQL/XML-инструкцию наподобие следующей.

```
INSERT WITH <?xml version="1.0"?>
       INTO OFFICES (OFFICE, CITY, REGION, SALES)
     VALUES <row>
              <office>23</office>
              <city>San Francisco</city>
              <region>Western</region>
              <sales>0.00</sales>
            </row>
```

Обновление базы данных обрабатывается аналогично. Простая инструкция UPDATE

```
UPDATE OFFICES
   SET TARGET = 200000.00,
```

```
       MGR    = 108
 WHERE OFFICE = 23;
```

может быть легко транслирована в эквивалентную гибридную SQL/XML-инструкцию наподобие следующей.

```
UPDATE WITH <?xml version="1.0"?> OFFICES
 WHERE OFFICE = 23
                <update_info>
                  <values>
                    <target>200000.00</target>
                    <mgr>108</mgr>
                  </values>
                  <where>office = 23</where>
                </update_info>
```

Инструкция DELETE требует только наличия предложения WHERE, с применением указанных выше соглашений.

Хотя некоторые производители SQL СУБД и добавили возможность обработки XML-операций INSERT, UPDATE и DELETE с использованием указанного подхода, методы представления имен таблиц и столбцов, а также значений данных в XML-тексте, как и отображение их на соответствующие структуры базы данных, остаются зависящими от конкретной СУБД. Хотя стандарт ANSI/ISO SQL и включает спецификацию применения инструкций INSERT, UPDATE и DELETE и столбцов с типом данных XML, а также предложения WITH, которое поддерживает использование XML в инструкциях SQL, стандарт для гибридного SQL/XML (пока что) отсутствует.

Хотя представление входных и обновляемых значений в виде небольших XML-документов концептуально простое и понятное, при этом встают некоторые важные вопросы функционирования СУБД. Например, список столбцов в инструкции SQL INSERT является избыточным, если XML-документ, содержащий вставляемые значения данных, включает имена столбцов как имена элементов или атрибутов. Почему бы просто не опустить список столбцов и не позволить XML-документам определять, какие столбцы должны быть вставлены? В случае интерактивного SQL это не составляет проблемы, но вряд ли формат XML будет использоваться в интерактивном режиме. В случае программного SQL проблема заключается в том, что XML-документ и содержащиеся в нем значения данных передаются СУБД во время работы программы. Если имена столбцов (или даже имя таблицы) также передаются только в XML-документе, то СУБД до момента выполнения не знает, какие таблицы и столбцы будут задействованы. В этой ситуации СУБД вынуждена использовать динамический SQL, описанный в главе 18, "Динамический SQL", со всеми его накладными расходами и сниженной производительностью.

Подобные проблемы возникают и у предложения WHERE в инструкциях UPDATE или DELETE, а также предложения SET инструкции UPDATE. Чтобы получить производительность и эффективность статического SQL, СУБД должна знать заранее (во время компиляции), какие будут задействованы условия отбора и какие столбцы будут обновляться. Один подход к этой проблеме заключается в применении параметризованного вида этих инструкций. Вот все тот же пример UPDATE, использующий упомянутый подход.

Обмен XML-данными

В простейшем виде СУБД может поддерживать обмен XML-данными, просто выводя результаты запроса в виде XML-документа и передавая его в качестве данных операции `INSERT`. Однако при этом от пользователя или программиста требуется аккуратная разработка формата генерируемых результатов запроса исходной базой данных, чтобы он соответствовал ожидаемому инструкцией `INSERT` в целевой базе данных. Обмен XML-данными более полезен, если СУБД обладает явной его поддержкой.

Некоторые коммерческие СУБД предлагают возможность выполнить быстрый экспорт таблицы (или, в более сложном случае, результатов запроса) во внешний файл, форматированный как XML-документ. Ей сопутствует возможность быстрого импорта из такого файла в таблицу СУБД. В такой схеме файл XML-документа становится стандартным способом представления содержимого таблицы для обмена информацией.

Заметим, что при наличии такой возможности импорта/экспорта таблицы с использованием XML ее применение не ограничивается обменом информацией между базами данных. Источником XML-документа может быть и некоторое приложение уровня предприятия, наподобие системы управления поставками (Supply Chain Management, SCM). Точно так же приложение может быть и приемником такого файла. Кроме того, многие приложения уровня предприятия в настоящее время поддерживают работу с XML-документами. Они могут выполнять дальнейшую обработку и интеграцию данных, например устранение дублей или слияние нескольких входных файлов.

Хранение и интеграция XML-данных

Ввод, вывод и обмен XML-данными предлагают очень эффективный способ интеграции существующих реляционных баз данных с растущим миром XML. При таком подходе XML используется во внешнем мире для представления структуриро-

многое для "понимания" содержимого таких документов. Вероятно, каждый документ можно идентифицировать одним или несколькими ключевыми словами или атрибутами, которые легко выделить и сохранить как обычные столбцы для обеспечения поиска.

Если обрабатываемые XML-документы в действительности представляют собой обрабатываемые записи с данными, то простая интеграция с применением больших объектов может быть слишком примитивной. По всей вероятности, в этом случае потребуется доступ и обработка отдельных элементов, а также поиск на основе их содержимого и атрибутов. СУБД предоставляет такие возможности для своих "родных" данных в строках и столбцах. Почему бы в таком случае СУБД не выполнять разложение входного XML-документа, преобразование содержимого его элементов и значений атрибутов в соответствующее множество строк и столбцов внутренних данных для последующей обработки? Что касается работы с внешними программами, то мы уже видели, как такой подход может работать при преобразовании табличных результатов запроса в XML-документ. Тот же метод можно использовать и для восстановления XML-документа, если он вновь потребуется в текстовом виде.

Преобразование XML-документов, которые представляют собой отличное представление данных для внешних приложений, во внутреннее представление данных и из него, полезно не только при работе баз данных. Та же проблема возникает, например, при обработке XML в Java, когда крайне желательно преобразовать XML-документ в набор экземпляров Java-классов для внутренней обработки и обратно в XML-документ. Процесс декомпозиции XML-документа на элементы и атрибуты в некотором внутреннем, бинарном представлении называется в литературе, посвященной XML, *демаршалингом* (unmarshaling). И наоборот, процесс сборки представления в виде набора отдельных элементов и атрибутов в полный текстовый XML-документ называется *маршалингом* (marshaling).

В случае простых XML-документов маршалинг и демаршалинг также просты, и коммерческие СУБД начинают их поддерживать. Рассмотрим еще раз простой заказ, представленный на рис. 25.3. Его элементы взаимно однозначно отображаются на столбцы таблицы `ORDERS`. В простейшем случае имена элементов (или атрибутов) идентичны именам соответствующих столбцов. СУБД может получить входной XML-документ наподобие показанного и автоматически превратить его элементы (или атрибуты, в зависимости от используемого стиля документа) в значения столбцов, используя имена элементов (или атрибутов) для управления процессом преобразования. Восстановление XML-документа из строк таблицы также не представляет никакой проблемы.

СУБД потребуется выполнить немного больше работы, если имена элементов в XML-документе не совпадают с именами столбцов. В этом случае требуется указать некоторое отображение между именами элементов (или атрибутов) и именами столбцов. Такое отображение несложно разместить в системном каталоге СУБД.