ИНФОРМАТИКА

2007 **октябрь-декабрь** **№ 4**

ПРИКЛАДНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

УДК 004.82, 004.65

М.С. Павловский

ОРГАНИЗАЦИЯ ХРАНЕНИЯ СТРУКТУРНЫХ ОБЪЕКТОВ, ПРЕДСТАВЛЕННЫХ В ВИДЕ XML-ДОКУМЕНТОВ, В РЕЛЯЦИОННЫХ БАЗАХ ДАННЫХ

Обсуждается проблема хранения фрагментов знаний, представленных в виде XML-документов, в реляционной системе управления базой данных. Представляется схема базы данных, в которой можно хранить разобранные XML-документы. Описываются методики осуществления поиска в базе данных такой структуры, а также механизм создания пользовательских запросов.

Введение

В настоящей работе рассматривается проблема хранения представлений фрагментов знаний сложной структуры (далее – фрагменты знаний) в реляционной системе управления базой данных (СУБД). Проблема состоит в том, что поиск в больших массивах информации, представленных в виде XML-документов, методом прямого перебора является низкоэффективным. В то же время XML-формат де-факто является стандартом для передачи данных. Так, например, XML используется для передачи данных в работе [1], в которой был представлен сервер баз знаний, основанный на использовании информационно-логических таблиц (ИЛТ) и представляющий знания для внешнего потребителя в виде XML-документов. Одним из недостатков такого подхода является то, что знания хранятся в базе данных (БД) во внутреннем двоичном формате, который не позволяет осуществить индексацию и нормализацию данных, а также доступ к базе из других программных комплексов. В настоящей работе предлагаются механизмы, которые бы позволили преодолеть эти недостатки.

1. Механизм хранения фрагментов знаний, представленных в виде XML-документов

В работе [1] предлагается использовать технологию XML для формализации представления знаний. Этот подход позволяет вести «одновременную (параллельную) работу с одним и тем же элементом данных как человеку, так и программе», но кроме представления знаний их требуется и хранить. Основные БД обладают реляционной структурой, а XML – иерархической, и до недавнего времени не было простого и элегантного способа интегрировать их [2].

Традиционной формой хранения XML-данных является либо хранение каждого документа в виде отдельного файла дисковой системы компьютера, либо хранение XML-документов в таблицах БД с использованием средств, которые предоставляют СУБД для обработки XML. В работе [2] отмечается: «Важно помнить, что "перемещение" XML в БД нельзя сделать единым для всех случаев образом. У каждой модели хранения данных есть свои плюсы и минусы. Понимание того, какая модель хранения данных XML наилучшим образом соответствует потребностям вашего приложения, критично для вашего успеха». Следовательно, необходимо доказать преимущества одного способа хранения фрагментов знаний перед другим.

Особенностью фрагментов знаний является то, что имеется очень много классов объектов при небольшом количестве самих объектов. Это означает, что при использовании файлов в качестве хранилища данных их число будет стремительно увеличиваться.

Для осуществления поиска среди множества XML-файлов может быть использован язык запросов XQuery, который является наиболее мощным и современным языком, «разработанным для формулировки запросов к реальным и виртуальным XML-документам и коллекциям этих документов» [3]. К сожалению, XQuery не подходит для осуществления поиска в XML-файлах нашего формата, так как они не содержат достаточно информации для составле-

ния XQuery-запросов напрямую. В работе [4] указывается: «XQuery используется преимущественно для управления контентом и интеграционными сервисами. Такие решения лучше всего подходят для создания и поддержки веб-сайтов и порталов для ограниченной аудитории».

При использовании встроенных инструментов СУБД для работы с XML-данными образуется большое число таблиц, количество которых будет увеличиваться при добавлении новых XML-документов. Кроме неудобств, связанных с разрозненностью хранимых данных, такие подходы не позволяют осуществлять быстрый поиск требуемой информации. Перебор всех документов неприемлем по времени выполнения операции, а реализация механизма индексирования потребует очень больших затрат ресурсов (человеческих и временных) и результатом будет низкая релевантность получаемых результатов.

Фрагменты знаний можно отнести к типу документов, ориентированных на данные, а не на документы [5], что говорит о выборе в пользу реляционной СУБД.

При выборе хранилища для XML-документов следует все-таки остановиться на реляционных СУБД. Основными их достоинствами являются высокое быстродействие, надежность, большой опыт использования, отличная поддержка со стороны других производителей средств разработки [6, 7] и пр.

Методом преодоления проблемы хранения структурных объектов может служить слитное хранение фрагментов знаний, представленных в виде XML-документов, в реляционной СУБД. Слитность хранения заключается в том, что фрагмент знаний разбирается на составные части и они последовательно сохраняются в БД. Разработанная модель БД (рис. 1), состоящая из фиксированного числа таблиц, позволяет хранить неограниченное число фрагментов знаний (ограничениями являются только максимальный размер БД и доступное дисковое пространство). Модель соответствует нотации IDEF1X [8].

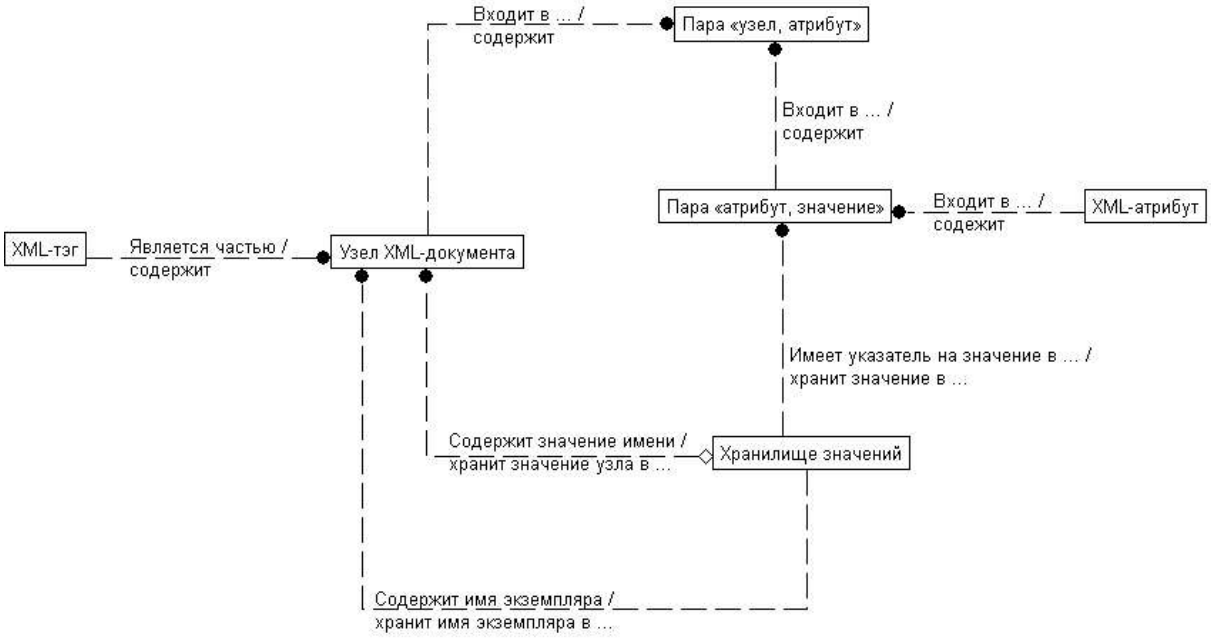


Рис. 1. Описание БД в терминах «сущность» – «связь»

Конкретная реализация схемы БД для слитного хранения фрагментов знаний может выглядеть так, как показано на рис. 2. Для увеличения скорости поиска следует создать индексы для тех полей, которые используются наиболее часто.

Дадим краткое описание таблиц:

XML_Attribute – имена атрибутов, которые встречаются в XML-документе, например Colspan, Rowspan, Predicate;

Domain_Values – значения любого элемента XML-файла (атрибута, тэга и пр.), например «Номинальный диаметр "D"», «BTW», «3», «120-260»;

XML_Attribute_Values – пары «атрибут, значение», которые встречаются в XML-документах, например «colspan="6"»;

XML_Node_Attributes – перечисление всех пар «атрибут, значение» в рамках одного узла, которые встречаются в XML-документах, например «colspan="6", predicate="BTW"»;

XML_Tag – имена используемых XML-тэгов, например NV, NT, VV;

XML_Node – таблица фактов. Содержит элементы, сочетание которых представляет отдельный узел XML-документа (кроме списка атрибутов этого узла). Parent_Id – указатель на родительский узел (принимает значение Null, если это корневой узел), Node_Number – порядковый номер узла в документе, Row_Number – номер строки, в которой находится этот элемент в документе, Column_Number – номер столбца, в котором находится этот элемент в документе. Два последних параметра являются избыточными, но они нужны для увеличения скорости обработки данных.

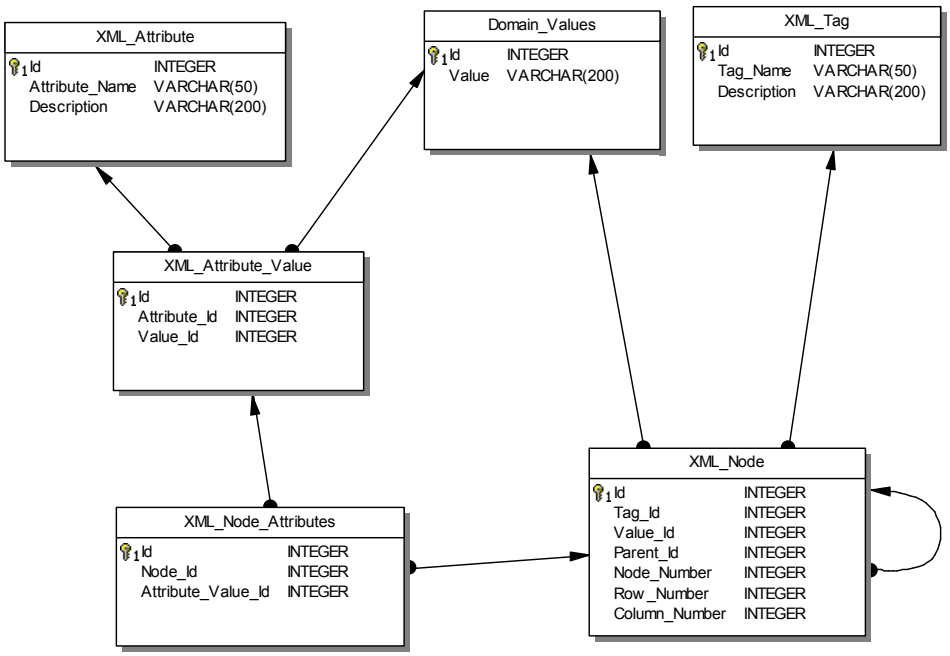


Рис. 2. Схема БД для хранения фрагментов знаний

Слитное хранение XML-документов в реляционной СУБД позволит решить задачу поиска факта, который может встречаться в различных документах в различных контекстах. При традиционном подходе пришлось бы последовательно открывать XML-документы и осуществлять поиск средствами XML-парсеров или же последовательно осуществлять поиск по всем таблицам и всем полям БД, в которой хранятся XML-документы. При использовании нового подхода поиск в БД будет осуществляться с учетом наличия индексов, что значительно повысит скорость поиска среди преобразованных документов.

2. Примеры SQL-запросов для поиска информации

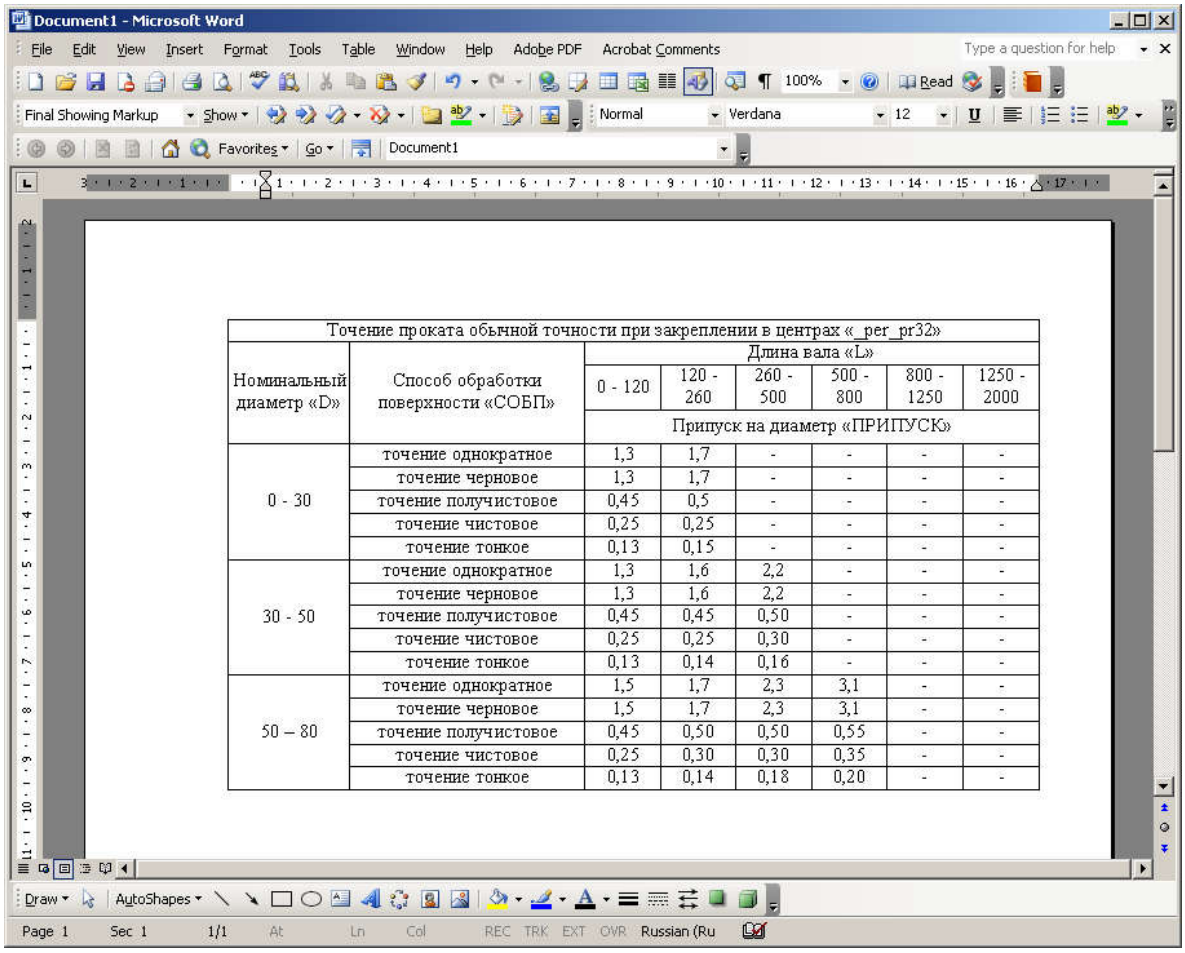
Если в БД поместить ИЛТ, представленную на рис. 3, можно составить несколько поисковых запросов (синтаксис соответствует языку СУБД Firebird [9]).

Листинг 1. Составить SQL-запрос для решения задачи «Найти все фрагменты знаний, у которых есть значения "точение однократное"».

```
Select distinct N.Parent_Id from XML_Node N, Domain_Values D where N.Value_Id = D.Id and D.Value =
"точение однократное";
```

Листинг 2. Составить SQL-запрос для решения задачи «Найти все фрагменты знаний, у которых есть совпадение поля "Способ обработки поверхности" и значения "точение однократное"».

```
Select distinct N.Parent_Id from XML_Node N, XML_Node N1 where
(N.Tag_Id = GetTagIdByName("NL") and N.Value_Id = GetDomainIdByValue("Способ обработки поверхности") and N1.Tag_Id = GetTagIdByName("VL") and N1.Value_Id = GetDomainIdByValue("точение однократное") and N1.Column_Number = N.Column_Number and N1.Parent_Id = N. Parent_Id)
or
(N.Tag_Id = GetTagIdByName("NV") and N.Value_Id = GetDomainIdByValue("Способ обработки поверхности") and N1.Tag_Id = GetTagIdByName("VV") and N1.Value_Id = GetDomainIdByValue("точение однократное") and N1.Row_Number = N.Row_Number + 1 and N1.Parent_Id = N. Parent_Id);
```



Точение проката обычной точности при закреплении в центрах «_per_pr32»

Номинальный диаметр «D»	Способ обработки поверхности «СОБП»	Длина вала «L»					
		0 - 120	120 - 260	260 - 500	500 - 800	800 - 1250	1250 - 2000
		Припуск на диаметр «ПРИПУСК»					
0 - 30	точение однократное	1,3	1,7	-	-	-	-
	точение черновое	1,3	1,7	-	-	-	-
	точение получистовое	0,45	0,5	-	-	-	-
	точение чистовое	0,25	0,25	-	-	-	-
	точение тонкое	0,13	0,15	-	-	-	-
30 - 50	точение однократное	1,3	1,6	2,2	-	-	-
	точение черновое	1,3	1,6	2,2	-	-	-
	точение получистовое	0,45	0,45	0,50	-	-	-
	точение чистовое	0,25	0,25	0,30	-	-	-
	точение тонкое	0,13	0,14	0,16	-	-	-
50 – 80	точение однократное	1,5	1,7	2,3	3,1	-	-
	точение черновое	1,5	1,7	2,3	3,1	-	-
	точение получистовое	0,45	0,50	0,50	0,55	-	-
	точение чистовое	0,25	0,30	0,30	0,35	-	-
	точение тонкое	0,13	0,14	0,18	0,20	-	-

Рис. 3. Пример ИЛТ

Листинг 3. Составить SQL-запрос для решения задачи «Найти решение "Припуск на диаметр", где номинальный диаметр 0-30, и способ обработки поверхности "точение получистовое", и длина вала 0-120».

```
Select N.Value_Id from Nodes N, LeftValuesRows(:LeftConditions) LVR, VerticalValueColumns(:VerticalConditions) VVC where
LVR.Parent_Id = VVC.Parent_Id and
LVR.Row_Number = N.Row_Number and
VVC.Column_Number = N.Column_Number and
N.Parent_Id = LVR.Parent_Id;
```

В данном листинге использовались следующие хранимые процедуры (ХП):

1. LeftValuesRows(), вычисляющая список номеров строк фрагментов знаний, которые соответствуют левым наборам условий. В рассматриваемом случае этот набор условий включает: номинальный диаметр 0-30, способ обработки поверхности «точение получистовое».

2. VerticalValueColumns(), вычисляющая список номеров столбцов фрагментов знаний, которые соответствуют вертикальным наборам условий. В рассматриваемом случае это условие «длина вала 0-120».

ХП LeftValuesRows() и VerticalValueColumns() по внутреннему устройству практически идентичны, поэтому приведем текст только процедуры LeftValuesRows():

```
Select LV1.* from LeftValues(:Parent_Id, :ConditionSetName1, :ConditionSetValue1) LV1,
LeftValues(:Parent_Id, :ConditionSetName2, :ConditionSetValue2) LV2 … where
LV1.Parent_Id = LV2.Parent_Id and
LV1.Row_Number = LV2.Row_Number and
……
LVn-1.Parent_Id = LVn.Parent_Id and
LVn-1.Row_Number = LVn.Row_Number;
```

3. LeftValues(), вычисляющая номер строки, в которой находится соответствующее условие. Она может быть записана как

```
Select N.Parent_Id, N.Row_Number from XML_Node N, XML_Node N1 where
(N.Tag_Id = GetTagIdByName(:TagName) and N.Value_Id = GetDomainIdByValue(:DomainValue) and
N1.Tag_Id = GetTagIdByName(:TagName) and N1.Value_Id = GetDomainIdByValue(:DomainValue) and
N1.Column_Number = N.Column_Number and N1.Parent_Id = N. Parent_Id);
```

4. GetTagIdByName() и GetDomainIdByValue() – простые ХП, возвращающие значения Id по имени объекта из соответствующей таблицы. Описанный набор ХП можно использовать также для поиска документов, обладающих общим контекстом.

Реализовав обертки для ХП на высокоуровневом языке программирования, можно создать механизм пользовательских запросов. Пользовательские запросы – это SQL-подобные конструкции, которые позволяют записать поисковую конструкцию в терминах фрагмента знаний. Таким образом, переписав задачу «Найти решение "Припуск на диаметр", где номинальный диаметр 0-30, и способ обработки поверхности "точение получистовое", и длина вала 0-120» в виде пользовательского запроса, получим

```
Select ПРИПУСК from _per_pr32 where
D = "0-30" and СОБП = "точение получистовое" and L = "0-120"
```

При вызове пользовательского запроса он будет транслирован во внутренние SQL-команды и выполнен на сервере БД.

Пользовательские запросы значительно проще обычных, поэтому могут быть составлены любым инженером без проведения серьезного переобучения и быть использованы для осуществления поиска требуемой информации в массиве фрагментов знаний, представленных в виде XML-документов.

3. Сравнение быстродействия

В разд. 1 настоящей статьи упоминалось, что одним из главных достоинств реляционных СУБД является их высокое быстродействие при работе с данными, представленными в табличном виде. Для сравнения быстродействия систем, основанных на хранении данных в XML-файлах и реляционных СУБД, был проведен следующий тест.

Поисковым запросом служило следующее условие: «Найти решения, для которых длина вала 260-500, и номинальный диаметр 50-80, и способ обработки поверхности “точение черновое”». Поиск решения осуществлялся в хранилищах трех различных объемов: 950, 6650 и 19 000 файлов, что соответствует малому, среднему и большому объему хранимой информации. Для расчета времени поиска решения использовался следующий подход. В каждом случае поисковый запрос запускался пять раз: две попытки – самая лучшая и самая худшая – отбрасывались, а на основании оставшихся вычислялось среднее арифметическое, которое округлялось до сотых долей секунды и заносилось в таблицу.

Тестирование осуществлялось на компьютере Celeron 2.8 ГГц, 512 МБ ОЗУ, 160 ГБ жесткий диск, Win XP SP2. Для тестирования XML-хранилища была разработана программа на C# (.Net Framework 2.0), исходными данными для которой служит имя каталога, где хранятся XML-файлы, и которая зачитывала, разбирала и выполняла поиск в каждом файле. После окон-

чания разбора и поиска во всех файлах выводилось время, потраченное на работу. Для разбора XML-документов использовался MSXML-парсер.

Для тестирования БД была выбрана СУБД Firebird 1.5.0, в которой были созданы таблицы и ХП. Для работы с СУБД Firebird использовалась система IB Expert v2007.01.20. Данная система позволяет задать входные параметры для ХП, запустить ее на выполнение и получить результаты вызова этой ХП. Система предоставляет полную информацию о выполненном запросе (используемый план SQL-запросов, количество индексированных и неиндексированных чтений и время выполнения запроса).

Таблица

Сравнение скорости поиска

Количество файлов в хранилище	Тип хранилища	Время поиска решения, с	Разница, %
950	XML	2,80	2800
	БД	**0,10**	
6650	XML	19	2814,81
	БД	**0,68**	
19 000	XML	56,55	2469,43
	БД	**2,29**	

Примечание: жирным шрифтом выделены результаты с наименьшим временем поиска.

Приведенные данные свидетельствуют о том, что скорость поиска информации в реляционной СУБД в 24-28 раз больше, чем скорость поиска в XML-файлах. Поэтому можно считать, что выбор в пользу реляционной СУБД полностью оправдан.

4. Практика применения

Для приведенной модели разработана БД, в которую была импортирована информация из произвольных ИЛТ. Для БД были написаны ХП, которые обрабатывают код, получаемый от сервера приложений (как показано в примерах). Такая реализация используется для того, чтобы сделать приложение гибким и более безопасным. Сервер приложений реализует веб-сервис, который позволяет осуществлять запросы как с локального компьютера, так и с удаленного (по локальной сети или через Интернет).

Восстановление исходного документа из БД осуществляется очень просто. Таблица «XML_NODE» содержит для каждого узла XML-документа следующую информацию: имя тэга; значение, содержащееся в узле; порядковый номер тэга в XML-документе; номер колонки и строки, в которой находится этот узел. Атрибуты, принадлежащие восстанавливаемому узлу, извлекаются из соответствующих таблиц и вместе с информацией об узле возвращаются в вызвавшую программу, где и преобразуются в исходный XML-документ.

Предоставляемый сервис позволяет осуществлять различные виды поиска в ИЛТ. Это может быть поиск конкретных решений или поиск справочной информации. Сам сервис может быть использован как одна из составляющих систем поддержки принятия решений на этапе технологической подготовки производства.

Результаты данной работы были применены в качестве основы для расширения функциональности сервера ИЛТ – добавлены возможности поиска по массиву ИЛТ, обработки пользовательских запросов и пр.

Заключение

Различные фрагменты знаний, представленные в виде XML-файлов определенного формата, можно хранить в реляционной БД в разобранном виде. Такой подход позволяет значительно ускорить поиск информации в документах за счет использования всех возможностей современных реляционных СУБД и одновременно решает актуальную задачу поиска общего факта, который может содержаться в документах с разным контекстом.

Список литературы

1. Кочуров, В.А. О проблеме принятия проектных решений в САПР / В.А. Кочуров // Моделирование интеллектуальных процессов проектирования, производства и управления: сб. науч. тр. – Минск: ОИПИ НАН Беларуси, 2003. – Вып. 1. – С. 199–206.

2. Скардина, М. Модели хранения XML-данных: единого варианта на все случаи нет / М. Скардина // Citforum.ru [Электронный ресурс]. – 2003. – Режим доступа: http://www1.citforum.ru/internet/xml/storage_models. – Дата доступа: 14.11.2007.

3. Eisenberg, A. XQuery 1.0 is Nearing Completion / A. Eisenberg, J. Melton // SIGMOD Record [Electronic resource]. – Vol. 34, № 4. – 2005. – Mode of access: http://www.sigmod.org/sigmod/record/issues/0512/p78-column-eisenberg-melton.pdf. – Date of access: 12.06.2007.

4. Cohen, F. Debunking XQuery myths and misunderstandings / F. Cohen // [Electronic resource]. – Mode of access: http://www.ibm.com/developerworks/xml/library/x-xqmyth.html. – Date of access: 08.02.2006.

5. Буре, Р. XML и базы данных / Р. Буре // Открытые системы [Электронный ресурс]. – № 10. – 2000. – Режим доступа: http://www.osp.ru/os/2000/10/178269/. – Дата доступа: 01.10.2006.

6. Павловский, М.С. Исследование и разработка методов создания интеллектуальных САПР на основе распределенных систем: дис. … магистра техн. наук: 05.13.12 / М.С. Павловский. – Минск, 2003. – 57 с.

7. Крил, П. Дискуссия о будущем баз данных / П. Крил // Computerworld [Электронный ресурс]. – № 20. – 2002. – Режим доступа: http://www.osp.ru/cw/2002/20/52676/. – Дата доступа: 12.03.2007.

8. Зеленков, Ю.А. Введение в базы данных / Ю.А. Зеленков // Мурманский государственный технический университет [Электронный ресурс]. – 1997. – Режим доступа: http://www.mstu.edu.ru/education/materials/zelenkov/ch_2_4.html. – Дата доступа: 18.12.2006.

9. Ковязин, А.Н. Мир InterBase / А.Н. Ковязин, С.М. Востриков. – М.: Кудиц-Образ, 2006. – 496 с.

Поступила 12.04.07

Белорусский национальный
технический университет,
Минск, пр. Независимости, 65
e-mail: Max.Paulousky@gmail.com

M.S. Paulousky

STORING THE STUCTURAL OBJECTS PRESENTED AS XML DOCUMENTS IN RELATIONAL DATABASE

The problem of knowledge fragments storing in DBMS presented as XML documents is discussed. The database scheme that contains «disassembled» XML documents is created. For database scheme the search methods and procedure that helps creating custom queries are described.