数据库学术会议论文集

郭景峰　陈子阳　主编

中国农业科技出版社

第十八届全国数据库学术会议
组织机构名单

会议主办单位 中国计算机学会数据库专业委员会

会议承办单位 燕山大学

大会主席 施伯乐 王益群 邢广忠

程序委员会

主席 唐常杰 周立柱

委员（按字典序）：

陈晓云 杜小勇 冯玉才 范 明 郭景峰 何守才
何新贵 黄锦辉 黄上腾 贾 焰 乐嘉锦 李建中
李 青 李天柱 李战怀 李昭原 刘国华
LihChyun Shu 刘连芳 陆宏钧 罗晓沛 马应章
马玉书 孟卫一 孟小峰 彭智勇 邵佩英 沈钧毅
施伯乐 孙建伶 孙志辉 唐世渭 童 頫 王 珊
王国仁 王海洋 王洪水 徐洁磐 杨冬青 于 戈
岳丽华 张 霞 张彦春 周傲英 周立柱 周龙骧

组织委员会

主 席 郭景峰

副主席 刘国华 原福永

成 员 张忠平 余 靖 陈 晶 王 璇
周军锋 张大鹏 陈子阳

编辑委员会 余 靖 陈 晶 王 璇

前　言

第十八届全国数据库学术会议（NDBC2001）于 2001 年 8 月 26 日在河北秦皇岛市举行。会议由中国计算机学会数据库专业委员会主办，燕山大学承办。自 2000 年数据库专委会成立以来，在延续多年来形成的传统的同时，积极寻求将这一传统的数据库盛会办成为聚集中国大陆、香港、台湾、澳门和海外华裔数据库专家、学者交流学习的论坛，成为数据库研究人员、实践人员、应用开发人员、用户和企业交流有关数据库管理与应用的研究成果和实验经验，探讨今后数据库管理与应用所面临的关键、挑战问题和研究方向的大本营，并逐步使之成为亚太地区乃至世界性的有影响力的国际学术会议。本次会议共收到论文 324 篇，是历届会议最多的一次，论文来自海内外、高等院校、科研院所、企事业单位和公司，具有广泛的代表性。

2001 年 5 月在南宁召开了第十八届全国数据库学术会议的审稿会议。会议有中国计算机学会数据库专业委员会主任王珊教授主持。经大会程序委员会的认真评审，最后评出 5 篇优秀论文推荐到《软件学报》发表；确定录用长文 94 篇，收入论文集 A 集（研究报告篇），由《计算机科学》设增刊出版；短文及会议交流 104 篇，收入论文集 B 集（技术报告篇），由燕山大学出版。本论文集为 A 集。论文基本上反映了我国目前数据库领域的理论研究、实践技术及数据库应用等方向的研究成果，内容涉及了当今数据库研究的各个方面，如数据库理论、移动数据库、数据挖掘数据仓库、半结构化数据 XML、WEB 数据库应用、海量数据存储、数据安全、工作流管理等。既对当前世界研究热点领域的最新研究进行了跟踪，也对今后数据库研究和应用所面临的关键性挑战性问题作了探讨。反映了我国数据库界追踪国际前沿，为国民经济建设服务的研究水平与成果。

本论文集分为如下几个专题：

A. Web 与数据库

B. B.KDD 与数据挖掘

C. 数据仓库、OLAP、多维数据库

D. 分布式数据库、多数据库系统、移动数据库

E. 特种数据库、并行数据库、多媒体数据库

F. 数据安全、事务处理、工作流管理

G. 面向对象数据库、数据库理论、知识库、查询处理

H. 数据库应用

在论文集出版之际，我们对所有投稿者表示衷心感谢，向审稿的专家、教授所付出的辛勤劳动致以深深的谢意。

我们特别感谢《计算机科学》杂志社在编辑出版本论文集中所付出的努力。

编者

2001 年 8 月

目 次

A.Web 与数据库

B.KDD 与数据挖掘

C.数据仓库、OLAP、多维数据库

D.分布式数据库、多数据库系统、移动数据库

E.特种数据库、并行数据库、多媒体数据库

F.数据安全、事务处理、工作流管理

G.面向对象数据库、数据库理论、知识库、查询处理

H.数据库应用

Variable and Index Processing in XML-QL Join

Liu Qingwen Li Jing

(Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences)

(Object Technology Center, Institute of Software, Chinese Academy of Sciences)

Abstract The growth of the internet has made it possible to query data in all corners of the globe. This trend is being made by the emergence of standards for data representation, such as XML. When used in conjunction with a DTD (Document Type Definition), XML permits the execution of a rich collection of queries using a query language such as XML-QL, XQL, Xquery. Our project is to develop an XML database : XML Content Server with native storage of XML documents and using XML-QL to be manipulating language for it. The paper depicts two problems when we executing interpretation of XML-QL. One is the variable table processing when joining XML documents of different DTDs. The other is index processing of query.

Keywords XML-QL, XML database, content management

1 Introduction

The Extensible Markup Language(XML) is an emerging standard for data representation and exchange on the Internet. In the near future it is expected that XML will replace HTML as the document flat file format for web resident data. When compared with other mark-up languages such as HTML the main advantage of XML is that each XML file can have a Document Type Definition(DTD) associated with it. A DTD serves an implicit semantic schema for the XML document and makes it possible to define much more powerful queries than simple, text-based retrievals. In many ways XML document and DTDs closely resemble the semi-structured datamodel that has been actively studied in recent years by the database community. Overall XML can serve at least two roles. First, as a new mark-up language, a web browser can browse XML file in the same way as an HTML file. Second, and more interesting to the database community, it can serve as a standard way of storing semi-structured data sets. XML offers significant opportunities for users to ask very powerful queries against the web. For example, doing a keyword search of "book", 'price" and "publisher" on the web will probably return millions of documents. Posing the query "find the top 10 well sold books that are priced below $10" using an XML-based query language such as XML-QL makes it possible to return the answer the user really wants.

One important question is what is the best way of storing XML documents as the performance of underlying storage representation has a significant impact on query processing efficiency. There have been numerous studies of alternative storage models and systems and in recent years several projects have proposed alternative strategies for storing XML data sets. XML storage strategies can be classified into three categories according to the underlying system used: file system, Relational database system and Object-Oriented

database. And it can also be classified in another way: file storage system and native storage system. For large queries file system is not suitable because the cost of parsing XML document is large. For native storage system, if XML documents are stored in tables of relational database, the overhead of mapping from relational tables or to relational databases is large and not efficient for implementation of query language XML-QL.

We implement XML Content Server on a page server with native XML storage. Our XML document native storage is to parse XML document when inserting into XML database. So when we process XML-QL query, we need not parse them again. Parsed XML documents are stored in compound documents efficient enough for XML-QL path navigation. A Compound document is a file system inside a file. So the native XML storage schema is very clear: non-leaf nodes are directories in compound document, leaf nodes are files in compound document. Currently we use Microsoft's compound document interface and implement the interfaces in page server to provide crash recovery, transaction's ACID properties, memory management and disk management. In XML content server, a DTD represents a collection of XML documents, one XML document can be viewed as an record of traditional database. The query language for retrieving qualified XML document is XML-QL. The interpretation of XML-QL is to navigate paths specified in XML-QL in compound filed stored in XML Content Server. There are two important problems in the process of XML-QL interpretation. One is the variable table processing when joining documents of different DTDs. The other is index processing of query.

2 Variable Table Processing of XML-QL Join

Multi-variable table can facilitate retrieval flexibly of any kind of XML-QL variation. Our scheme of processing XML-QL interpretation is as follows. Get retrieval condition by lexical analysis and syntax analysis. Then open sub document trees for compare and fetch. When encounter variable, whose element or attribute has $ prefix in WHERE sub clause , check DTD definition meta information stored in database to see whether it is a leaf node or non-leaf node. When the variable represents a sub-tree variable, copy whole sub–tree to variable table. When encounter #PCDATA, CDATA type element or attribute or other inner implemented strong type, open according stream and compare condition with the value in document stream. If it is not satisfied, go to next iteration. When all conditions are satisfied execute CONSTRUCT sub-clause and construct result sub tree according variable value got in variable table. For example:

```
WHERE <book>$p</> IN "bib.xml"
        <title>$t</> in $p,
        <publisher><name>hope</></> IN $p
CONSTRUCT <result>
          <title>$t</>
          {
              WHERE <author>$a</> IN $p
              CONSTRUCT <author>$a</>
          }
          </>.
```

For every book document, if it satisfies that publisher name is "hope" then get title sub-tree and generate result. When processing nested sub retrieval in construction sub clause, gets author sub-tree in sub tree $p. In the execution of nested queries. We need to preserve multi-level

variable table. When exit current iteration, we delete it. Because lower sub clause references of variables of higher sub-clause, some times it is necessary to trace back to visit higher level variable table. For the case of element join. an example query is :

```
WHERE <article>
<author>
<firstname>$f</>
<lastname>$l</>
</>
</>CONTENT_AS $a IN "bib.xml"
<book year=$y>
<author>
<firstname>$f</>
<lastname>$l</>
</>
</> IN "bib2.xml"
CONSTRUCT <article>$a</>
```

For the second condition's variables, Whether it is a fetch operation or compare operation depends on whether variable table exist their value. We do not fetch sub-tree for $f and $l in second condition. But first check variable table to see if $f and $l variable value already existed, If there exist their value then compare them with element values in XML document. Otherwise generates new variable table entries. It equals following query:

```
wWHERE <article>$t1</>
        CONTENT_AS $a IN "bib1.xml"
CONSTRUCT <author>
                <firstname>$f</>
                <lastname>$l</>
                </> IN $t1
                {
                  WHERE <book year=$y>
                          <author>
                                <firstname>$f</>
                                <lastname>$l</>
                                </> IN "bib2.xml"
                                y>1995
                    CONSTRUCT <article>$a</>
                }
```

In the above query, if we had only one variable table there would be errors. Variable $a becomes join when we process nested where sub-clause. But in fact it is not. So we must have every sub clause a variable table. And nested query variables are in different variable table so that above query will not be treated as join operation.

3 Index Processing of XML-QL

As the purpose of project of XML Content Server is to implement a native storage XML database. So to some extent we extend XML DTD. One extension is to allow user to specify indexed leaf node so that query documents in the repository can be accelerated. Its format is as follows:

<!ELEMENT ...> indexed

specify when parsing this element, the content of the element is indexed.

<!ATTLIST ...> indexed

specify when parsing attribute of the element, attribute is indexed. And when elements or attributes are specified with indexes on their DTD declaration, element content or attribute will be indexed when the whole document be parsed and stored into XML content server. So when documents has indexed element, we can make use the indexes. For example, in the following XML-QL retrieval:

```
WHERE <Quotes><Quote>
        <Symbol>MSFT</>
    </></> ELEMENT_AS $g
IN "quotes.xml"
ONSTRUCT $g
```

When an index defined on Symbol element , we need not scan all documents in the collection. At the run time, we open indexes and just scan a small range in the collection. If there are multiple indexes, the intersection set must be calculated. And for large data set it is efficient. But with the flexibility of XML-QL , something must be considered when open indexes. For example, for the following query:

```
WHERE <book year=$y>
        <author>$a</>
        </> IN "bib2.xml",
        <firstname>$f</> IN $a
        <lastname>li</> IN $a
CONSTRUCT <author>$a</>
```

If there has already defined index on lastname, we must open it when we compute the first sub condition. So in order to open all indexes with XML document collection file, we scan all sub conditions to find indexed element. We use following table for search

Variable	Element or Attribute	Data source
$a	author	"bib2.xml"
$y	year	"bib2.xml"
$f	firstname	$a
li	lastname	$a

Scan data source column for certain collection file in the table. If element or attribute is not variable but has value then add to list for index open. If element or attribute's data source is not XML document collection file but XML-QL variable then check precede rows to find this variable and check the data source whether or not the same data source. For the same data source add the element or attribute to list for index open. After all items in the table checked, open all indexes according to the list and compute their range intersection for final retrieval.

With index on elements or attributes we can use indexed join. For case of join variables in the variable table must also be considered. For following example:

```
WHERE <article>
        <author><firstname>$f</>
                <lastname>$l</>
        </>
        </> CONTENT_AS $r IN "bib1.xml",
        <book year=$y>
        <author>$a</>
        </> IN "bib2.xml",
        <firstname>$f</> IN $a,
        <lastname>$l</> IN $a,
        $y>1995
CONSTRUCT <result>$r</>
```

When we scan XML document collection file bib2 , we not only open index on year attribute but also open indexes on firstname and lastname according to the value firstname and last name has in variable table for the first condition fetch operation.

4 Conclusion

One of the key part of XML Content Server is the implementation of XML retrieval language XML-QL. Multi-variable table can join element or attribute in a very flexible way. Enhanced by index, retrieval can be accelerated. An algorithm to open all present indexes is provided.

References

1 Alin Deutsch , Mary Fernandez , Daniela Florescu, Alon Levy , Dan Suciu , XML-QL: A Query Language for XML. http: // www. w3 . org / TR/1998/NOTE-xml-ql-19980819

2 Jianjun Chen, David J. DeWitt, Feng Tian, Yuan Wang. NiagaraCQ: A Scalable Continuous Query System for Internet Databases. SIGMOD 2000, p379-390.

3 Feng Tian, David J. DeWitt, Jianjun Chen, Chun Zhang.The Design and Performance Evaluation of Various XML Storage Strategies

4 Jayavel Shanmugasundaram, H. Gang, Kristin Tufte,

Chun Zhang, David DeWitt, and Jeffrey F. Naughton Relational Databases for Querying XML Documents: Limitations and Opportunities., Proceedings of the 1999 VLDB Conference, September 1999.

5 Jayavel Shanmugasundaram, Kristin Tufte, David DeWitt, Jeffrey F. Naughton, and David Maier. Architecting a Network Query Engine for Producing Partial Results. WebDB'2000, May 2000.

6 Active Query Caching for Database Web Servers . Qiong Luo, Jeffrey F. Naughton, Rajesekar Krishnamurthy, Pei Cao, and Yunrui Li. WebDB'2000, May 2000.

7 C. Zhang, J. Naughton, D. DeWitt, Q. Luo, G. Lohman. On Supporting Containment Queries in Relational Database Management Systems. SIGMOD'01.

8 http://www.w3c.org

Web 查询语言 WD-SQL 系统

魏国志　刘海涛　陈世福

（南京大学计算机软件新技术国家重点实验室　南京 210093）

摘　要　在分析现有 Web 查询语言的基础上，提出了一种新的 Web 查询语言 WD-SQL，给出了该查询语言的语法、查询过程、查询实例，同时还给出了该查询语言进行数据抽取的算法，该查询语言具有易于理解，使用简单，功能强，能进行模式抽取和数据抽取等多种优点。

1 前言

随着 WWW 技术在全球范围内的迅速发展与普及，网络信息资源日趋丰富，Web 页面作为一个全球化信息空间，蕴含着具有极大潜在价值的信息和知识，然而对于用户来说有用的内容可能是其中极小的一部分，却难于获得。最初用户只能通过一些网站或基于关键字的搜索引擎来获得所需要的信息，但是由于搜索结果的庞大使用户难以得到有用的信息，已经远远不能满足用户对于信息的要求，对于用户来说已经到了无法使用的地步，导致 WWW 上信息的闲置与浪费的情况非常严重，需要研究新的搜索技术以便能够迅速、准确的查找到用户所需的资源，而 Web 查询语言就是目前研究的热点[1][2]。

由于大多数搜索引擎仅仅支持简单的信息检索功能，无法支持更复杂的查询要求，如页面之间以及页面内部信息的聚集（aggregation）、排序（sorting）、选择（select）以及投影（Project）等等功能，而传统的数据库应用系统的开发中所使用的 SQL 可以满足上面的所有功能要求，研究人员认为使用类似 SQL 的语言来访问 Web 上的信息将可以达到同样的效果[3]。

一般情况下 Web 查询语言应当至少支持如下功能：

- 从大量的 Web 页面中抽取数据；
- 支持 Web 页面与关系数据库或面向对象数据库之间转换数据；
- 能够整合多个数据源的数据；
- 能够支持类似关系数据库的操作，如聚集（aggregation）、排序（sorting）、选择（select）以及投影（Project）等等功能；
- 支持查询优化；
- 支持视图；

但目前的查询语言并不完全支持上述的功能，我们研制的 Web 查询语言 WD-SQL，不仅具有上述的功能，并且可以对 Web 数据源进行查询，又具有对关系数据库查询的能力。

2 Web 查询语言现状

最早的 Web 查询语言 WebSQL 是由加拿大多伦多大学开发的，它类似传统数据库应用中的 SQL 语言。此后不断出现新的 Web 查询语言。下面简单介绍几种重要的 Web 查询语言。

LOREL[4]：LORE 系统中应用的查询语言，是一种类 SQL 语言，之所以这样说是因为 LOREL 语法与 SQL 语法非常类似，设计目的是用来查询内部（Intranet）环境中异构的半结构化信息。LOREL 将域名的概念进一步扩展为信息/对象的路径描述并且提供函数调用，如 Grep，在信息检索中可以提供更加灵活的字符串匹配功能。

WebSQL[5]：WebSQL 把 Web 看做由

页面构成的一张数据库表，其中页面的URL，类型，最后修改日期，作为这张表的域，WebSQL 扩展了标准的 SQL，增加了一些与页面有关的信息，如 URL 和页面标题可以作为域名进行查询，以及预定义的字符串搜索函数，如“Mentions”。

WebLog[6]：是一种基于 Schemalog 的 Web 查询语言，这种语言的目标是一个功能更加完善的语言，不仅支持查询而且支持安排查询结果的显示以及排列格式。这种语言非常复杂，而且缺乏查询接口，使其非常难于使用。

TSIMMIS[7]：是斯坦福大学为了解决查询异构信息源而进行的一个项目，TSIMMIS 与 WebLog 非常相像，同样提供许多预定义的查询而不需要用户直接提供复杂的查询，但是预先提供的查询是很有限的，因此查询受限于这些预先提供的查询。

W3QS(WWW Query System) [8]：是以色列工业大学开发的高级类 SQL 的 Web 查询语言的一个项目，W3QL 语言将整个 Web 看作一个巨大的数据库，W3QL 可以查询其结构以及内容。W3QS 允许用户设置起始查询地址，同样 W3QS 允许用户设置查询的地址深度，另外 W3QS 允许用户通过任意的 Web 结构进行查询。

The Araneus Project[9]：该项目引入一系列语言用来管理和构建 Web 中的数据，并且以特定的数据格式提供给用户，有 Araneus 的数据格式，描述 Web 超文本的格式。在这个数据格式之上，The Araneus Project 开发了两种语言 Ulixes 以及 Penelope，用来支持定义用户视图。Ulixes 用来建立 Web 的数据库视图，这些视图可以通过数据库技术分析以及整合得到，这些视图用来产生超文本视图，Penelope 可以通过这些视图构建出 Web 超文本。

WQL(WebDB Query Language) [10]：大多数的搜索引擎仅仅支持简单的信息检索功能，而 WebDB 的目标是支持更加高级的查询能力，包括在页面级别上支持 aggregation、sorting、select 以及 Projection 操作。WebDB 是一个提供全部的类似数据库查询功能的查询系统。

3 WD-SQL 语言

3.1 WD-SQL 的功能

我们研制的 WD-SQL 语言作为 Web 查询语言的一种，不仅仅支持上面提出的 Web 查询语言应当支持的功能，而且还有一些独有的功能。WD-SQL 语言支持的功能有：

1)从 Web 数据源中抽取模式：WD-SQL 语言具有模式抽取的功能，可以从 Web 数据源中抽取模式，并作为查询及数据抽取的基础。

2)从大量的 Web 页面中抽取数据：WD-SQL 建立在模式抽取的基础之上，可以利用模式抽取的结果（数据模式）进行查询并抽取数据。由于模式抽取的数据库模式中已经包含了 Web 数据源的结构，因此 WD-SQL 具有对 Web 数据源进行查询并抽取的能力。

3)支持 Web 页面与关系数据库之间转换数据：WD-SQL 建立在模式抽取的基础之上，而模式抽取即实现了 Web 数据源结构与关系数据库模式之间的对应，使支持 Web 页面与关系数据库之间的数据转换非常简单。

4)能够整合多个 Web 数据源的数据：WD-SQL 的模式抽取实现了单个 Web 数据源与关系数据库模式之间的对应，而如果对多个 Web 数据源进行模式抽取，并将获得的数据库模式进行模式整合，即获得整合过的数据库模式，则可以实现多个 Web 数据源与关系数据库模式之间的对应，另外由于支持 Web 页面与关系数据库之间数据的转换，使整合多个 Web 数据源的数据成为可能。

5)能够支持类似关系数据库的操作，如聚

集（aggregation），排序（sorting），选择（select）以及投影（Project）等等功能。

6) 支持查询优化：WD-SQL 支持查询的优化，因为 WD-SQL 采用类似关系数据库中所定义的标准 SQL，所有可以对标准 SQL 进行的优化，全部都可以应用到 WD-SQL。

7) 支持 Web 视图。[11]

8) 具有对关系数据库查询的能力。

3.2 WD-SQL 的语法及语义

WD-SQL 是一种类似 DBMS 的标准 SQL 的语言，语法结构仍然采用 select-from-where 结构，实际上 WD-SQL 可以看作是标准 SQL 的一个子集的扩充，其语法描述如下：

Select [{<表名>.]<属性名>[,[{<表名>.]<属性名>]} | [{<页面>.]<基本项名>[,[{<页面>.]<基本项名>]} | All

From [Table <表名>{Of <数据源名>}][,[Table <表名> {Of <数据源名>}]} |[Page <页面>][,[Page <页面>]} | [Page All {Of <数据源名>}][,[Page All {Of <数据源名>}]]

Where <条件表达式>

其中 Of、Table 等等是 WD-SQL 引进的新保留字。在对某个 Web 数据源进行查询之后，关系数据库中就已经存储了该 Web 数据源的模式以及从该 Web 数据源中抽取的数据，必须区分查询是对关系数据库中所存储的该数据源的数据还是对 Web 数据源的数据进行查询，为此 WD-SQL 引进保留字 Of；此外还必须区分查询是对关系数据库还是对 Web 数据源进行查询，因此 WD-SQL 引进了 Table 与 Page 这两个保留字。

由于 WD-SQL 建立在对 Web 数据源抽取模式的基础之上，在进行 Web 数据源模式抽取的过程中已经实现页面与实体，基本项与属性的对应关系，因此 WD-SQL 查询语句均可以转换为标准 SQL 语句，而且无论查询语句如何变化，其最终均体现为对关系数据库的查询，即 WD-SQL 查询语句最终将转换为标准 SQL 语句。因此可以说 WD-SQL 的语义与标准 SQL 是相同的，另外由于同样的原因，WD-SQL 查询的优化完全相同与标准 SQL。本文不再赘述。

4 WD-SQL 查询技术

4.1 WD-SQL 查询过程

WD-SQL 的查询执行过程如下，首先从关系数据库中获得查询结果，如果查询结果不存在，或者关系数据库中的数据已经失效（因为 Web 数据源中的数据经常发生变化，即存在一个时效），则向元搜索引擎发出搜索命令（关于元搜索引擎，参看文献[11]），元搜索引擎调用数据源本身的搜索引擎，获取搜索结果页面。判断搜索结果页面是否已经抽取模式，如果已经抽取，则应用数据库模式进行数据抽取，如果发现抽取错误，则重新对搜索结果页面进行模式抽取，并修改全局数据库模式，此后进行数据抽取，如果抽取过程正常进行，则将结果保存到关系数据库中，再次从数据库中获得查询结果。

在查询过程中，涉及到元搜索引擎，这是因为 WD-SQL 对 Web 数据源的查询是通过元搜索引擎进行的。另外涉及到了关系数据库，关系数据库扮演的角色很多，一是抽取数据的存放地，二是 Web 数据源模式的存放地，另外还扮演着缓存的角色。

4.2 WD-SQL 查询算法

```
Query（）//查询执行过程
//用户发出查询请求。
UserQuery=GetUserQuery（）
//处理查询请求，解释并转换查询请求
UserQuery=TransToDBSQL（UserQuery）
//检查查询请求语法语义是否正确
If Check(UserQuery) then
```

```
    Return "错误"
End if
//检索关系数据库，查找符合查询要  求的数据
If Database.Query（UserQuery） Then
//如果关系数据库中，已经存在符合查询要
//求的数据，则直接返回查询结果
Return UserQueryResult
Else
//否则，根据查询要求向元搜索引擎发出搜索命令
SearchResult =
 MetaSearchEngine.Query(UserQuery）
      //如果查询结果集中有尚未抽取页面
      //模式的新页面
If Exist NewPages In SearchResult Then
//用已经抽取的数据库模式抽取数据
If Not ExtractData(NewPages) Then
//抽取新页面集合的数据
//库模式并进行模式整合
ModelIntegrate（ExtractModel(NewPages))
//按照数据库模式抽取数据并保
//存在数据库中
                ExtractData(NewPages)

          End if
      Else
//用已经抽取的数据库模式抽取数据
If Not ExtractData(SearchResult) Then
//抽取并修改原有数据库模式
ModelIntegrate（ExtractModel(SearchResult))
          // 按照数据库模式抽取数据并保存在
          //数据库中
        ExtractData(SearchResult)
      End if
    End if
End if
//检索关系数据库，查找符合查询要求的数据
Return Database.Query（UserQuery）
```

4.3 查询处理

查询处理部分主要处理查询语句的转换、语法的检查和执行查询等。

1)查询语句转换

查询语句转换包括两种情况，一是WD-SQL格式转换为标准SQL格式，这是因为WD-SQL语言查询过程中需要多次查询关系数据库；二是标准SQL格式转换为WD-SQL格式，这种情况出现在关系数据库中已经抽取并存储的数据不能满足查询要求而向元搜索引擎发出搜索命令时。由于在进行Web数据源模式抽取的过程中已经实现页面与实体，基本项与属性的对应关系，而且这些对应关系存储在关系数据库中，因此只需通过对页面-实体，基本项-属性对照表即可实现查询语句的双向转换。

2)查询语句语法检查

在WD-SQL格式转换为标准SQL格式之后，将进行查询语句的语法检查。

3)执行查询

在WD-SQL格式转换为标准SQL格式之后，执行对关系数据库的查询，如果无法获得查询结果，则将标准SQL格式转换为WD-SQL格式，向元搜索引擎发出搜索命令。

5 查询举例

下面将举例说明用WD-SQL查询语言的查询示例。

例：Select ware.price,ware.type from table ware of 263shop where ware.name= "手机"

该查询将从已经由 263shop 数据源中抽取的并存储在关系数据库中的数据中获得所有手机的价格以及类型。

例：Select all from table ware of 263shop where ware.name=“手机”

该查询将从已经由 263shop 数据源中抽取的并存储在关系数据库中的数据中获得所有手机的所有属性值。

例：Select ware.price,ware.type from page all of 263shop where ware.name=“手机” and ware.type=“Nokia”

该查询将从 263shop 数据源中查询诺基亚手机，并进行数据抽取，以便获得所有手机的价格以及类型。

例：Select all from page “http://shopping.263.net/Article/articlelist.asp?Class1=09&Class2= 01&Class3=02 ” where ware.name=“笔记本电脑” and ware.type=“联想”

该查询将从指定的页面中查询联想笔记本电脑，并进行数据抽取，以便获得所有笔记本电脑的属性值。

6 小结

本文中研制的 WD-SQL 将 Web 数据源视为一种数据库，而整个 Web 则视为一个分布式异构数据库系统，另外结合采用映射的方法，首先将 Web 数据源映射到关系数据库中，然后将通过数据抽取将 Web 数据源中的页面映射为关系数据库中的实例。

相对于其他 Web 查询语言，如 WebSQL、WebLog、TSIMMIS、W3QS，WD-SQL 具有一定的特色：

WD-SQL 充分利用当前已经成熟的关系数据库技术，其语法结构仍然采用 select-from-where 结构，与标准 SQL 非常相似，易于理解，容易使用。

WD-SQL 同时具有对关系数据库以及对 Web 数据源进行查询的能力，并具有模式抽取及数据抽取的功能，可以从 Web 数据源中抽取模式或数据，作为查询的基础。

本文对查询语言在 Web 数据源查询中的作用进行了研究，研制出一种 Web 查询语言 WD-SQL，以后的工作进一步的完善和提高 WD-SQL 的功能。

参考文献

1 Robert Filman, Feniosky Pena-Mora, Seek, and ye shall find, IEEE Internet Computing, July.August 1998

2 Mecca P.G, Merialdo P. To Weave the Web, Proceedings of International Conference on Very Large Data Bases, Athens, 1997

3 Gustavo O.Arocena, Applications of a Web Query Language, Computer Networks and ISDN Systemsn 29(1997), 1305-1316

4 Abiteboul S, Quass D, McHugh J, Widom J, Wiener J, The Lorel Query Language for Semi-structured Data, International Journal on Digital Libraries, 1997, 1(1): 68-88

5 Li Wen-Syan, Candan K.Selcuk, Kyoji Hirata, Yoshinori Hara, Facilitating Multimedia Database Exploration through Visual Interfaces and Perpetual Query Reformanlations, Proceedings of the 23th International Conference on VLDB, Athens, Greece, August, 1997

6 Lakshmanan L, Sadri F, Subramanian I, A declarative language for querying and restructuring the web, Proceedings of RIDE_NDS, IEEE Computer, 1996

7 Chawathem S, Molina H.Garcia, Hammer J, Ireland K, Papakonstantinou Y, Uliman J, Widom J, The TSIMMIS project: Integration of heterogeneous information sources, IPSJ Conference, 1994

8 Konopnicki D, Shmueli O, W3QS:A Query System for the World Wide Web, Proceedings of the 21st International Conference on Very Large Data Basesm,1995,54-64

9 Mecca P.G, Merialdo P, Semistructured and structured Data in the Web: Going Back and Forth, Proceedings of the Workshop on Semi-structured Data, Tucson, Arizona, May 1997

10 Li Wen-Syan, Junho Shim, Candan K.Selcuk, Yoshinori Hara, WebDB: A Web Query system and its Modeling, Language and Implimentation, Proceedings of International Forum on Research and Technology Advances in Digital Libraries, 1998, 216-227

11 刘海涛，Web 信息抽取及搜索引擎的研究，博士论文，南京大学，2001 年 3 月