

异构数据集成查询处理研究

毛佳飞, 叶霞, 李俊山

(火箭军工程大学 信息工程系, 陕西 西安 710025)

摘 要: 对集成系统的查询分解和查询优化进行了深入的研究. 在查询分解方面设计了依据全局元数据字典的分解算法, 并以实例证明了算法的准确性. 在查询优化方面提出了基于 XML Schema 的查询优化算法, 该方法可以有效处理路径表达式中“//”和“*”情况, 充分利用局部数据源 XML Schema 信息减小了查询节点范围, 减少了连接操作次数, 提高了集成系统的查询效率.

关键词: 异构数据; 数据集成; 查询分解; 查询优化

中图分类号: TP311

文献标识码: A

文章编号: 1000—7180(2018)05—0045—06

Research on Heterogeneous Data Integration Query Processing

MAO Jia-fei, YE Xia, LI Jun-shan

(Department of Information Engineering, Rocket Force University of Engineering, Xi'an 710025, China)

Abstract: In the aspect of query decomposition, a decomposition algorithm based on global metadata dictionary was designed, and the accuracy of the algorithm was proved by an example. In the query optimization, an query optimization algorithm based on XML Schema was proposed. This method could effectively deal with the // and * cases in the path expression, and maked full use of the local data source XML Schema information to reduce the query node range, decreased the times of connection operation and improved the query efficiency of integration system.

Key words: heterogeneous data; data integration; query decomposition; query optimization

1 引言

互联网技术的发展,使得人类所积累的数据急剧增加,而不同企业和机构组织使用的数据源和管理系统各不相同,所以异构数据集成^[1]的问题迅速成为研究热点.

查询处理是异构数据集成的研究焦点,主要包括查询分解和查询优化两方面. 在查询分解方面,通行做法是依据元数据字典存储的模式映射信息将全局查询分解为子查询. 而在查询优化方面,这种方法未充分利用各局部数据源模式 XML Schema,查询优化效果不明显. 文献[2]仅仅为系统建立了缓存,没有深入讨论 XQuery 子查询的查询机制. 文献[3]小枝 Twig 查询需要将查询树模式与文档树进行映射,耗时较长.

本文设计了以 XQuery 为全局查询语言和 XML 为公共数据格式的异构数据集成系统框架,研究了集成系统查询处理全过程,提出了基于 XQuery 查询语言的查询分解算法和查询优化算法,充分利用了模式映射信息,有效提高了集成系统的查询效率.

2 异构数据集成系统框架

本文设计的系统框架采用中间件^[4]方式集成异构数据源,系统总体设计如图 1 所示,中间件主要包含虚拟的全局视图和全局查询处理机制,可以将用户的 XQuery 全局查询分解转换到各个局部数据源执行,并将查询结果整合为 XML 文档提交用户. 系统采用 XML Schema 建立全局模式到局部模式的映射关系,并将其保存在元数据字典作为查询分解

和查询优化的依据,并设置查询结果缓存,进一步提高系统查询效率.

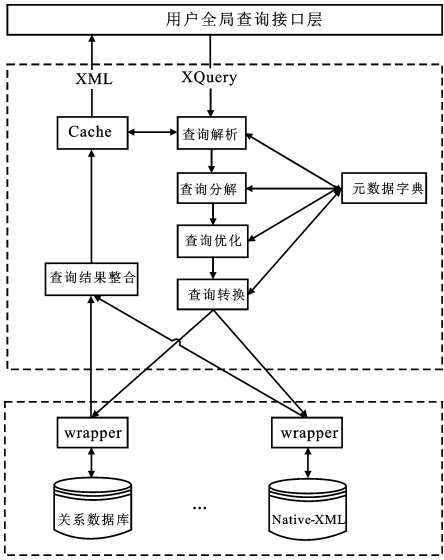


图 1 异构数据集成系统总体设计

异构数据集成系统总体设计分为用户接口层、中间件层和数据源层三层. 在该体系机构中,每个模块的功能如下.

(1)用户接口层:用户提交全局模式上的查询,系统对查询进行相应处理,将查询结果整合为 XML 文档格式,通过此接口层反馈给用户.

(2)查询解析:校验用户提交的 XQuery 查询合法性,按照标准化规则对 XQuery 进行标准化,将标准化的结果传递给查询分解模块.

(3)查询分解:根据元数据字典存储的模式信息将标准化的 XQuery 查询分解为各数据源模式的 XQuery 子查询,记录 XQuery 子查询之间的关系运算和执行先后顺序.

(4)查询优化:根据局部数据源 XML Schema 信息对 XQuery 子查询进行优化.

(5)查询转换:将 XQuery 子查询转化为本地数据源能够执行的查询语言,将查询命令下发给各数据源执行.

(6)元数据字典:元数据字典存储所有异构数据源的信息,包括各数据源的位置、数据模式等,负责管理更新全局视图、局部模式和全局模式的映射关系,提供查询分解和优化的依据.

(7)查询结果整合:此模块负责将各数据源的查询结果整合统一转化为 XML 文档并进行排序、去重、分组连接、关系运算等操作.

(8)Cache 模块:建立缓存淘汰算法,实现对查

询结果的重用,提高系统查询效率.

3 异构数据集成系统的查询处理

对异构数据查询处理的研究集中在查询分解和查询优化两方面,下面就这两方面进行深入的分析.

3.1 全局查询语言 XQuery

因为系统采用 XML Schema 作为公共的数据模式来消除各数据源的异构性,所以将 XQuery 作为全局查询语言,其具有以下特征^[5]:XQuery 被设计用于操作无类型(与数据无关的 XML Schema)、有类型的 XML 文档或者两种类型混合的文档;非常适合从大量 XML 文档抽取特定数据并提供过滤、排序以及相互转化所需数据的功能;查询的输入和输出均为 XML 格式;能实现多种数据操作,支持多种类型. XQuery 的 FLWOR 语句与 T-SQL 语句类似. 例如:

```
for $ b in doc("bib. xml")/bib/book
let $ a := $ b/author
where $ b/publisher/text() = "Addison - Wesley" and
$b/@year > 1991
order by $ b/@year
return
< book year = " { $ b/@ year } " > { $ b/author } </
book >
```

上述表达式中,for 子句创建一个表示节点集多次求值的变量,let 子句绑定中间变量,where 子句进行谓词过滤,order by 子句指定返回结果的顺序,return 子句构造查询结果. FLWOR 语句兼具结构化查询语言和过程化查询语言的特点,支持表达式的任意嵌套.

3.2 查询分解

查询分解^[6]是异构数据集成系统查询处理的关键技术之一,它将用户的全局查询分解为针对不同数据源的子查询. 查询分解算法在保证全局查询分解前后查询结果一致的情况下,使针对特定数据源的查询尽可能准确,并且尽量减少从数据源返回的结果集,提高网络传输效率.

系统的查询分解算法建立在全局查询 XQuery 语句上,系统的中间件需要处理事件包括以下几种:

- (1)数据源的位置及相对于全局路径的相对路径表达式;
- (2)查询的谓词筛选条件及执行顺序. 元数据字典保存的是全局 XML Schema 和局部数据源 XML Schema 的映射关系,作为查询分解的依据. 系统查询分解模块将全局 XQuery 查询分解为 XQuery 子查询并传递给查询优化模块,系统的查

询分解算法如下.

算法1 全局查询 XQuery 分解

输入:全局查询 Q 和元数据字典中数据源信息

输出:子查询 $Q_1, Q_2, \dots, Q_n$

(1)解析并标准化全局查询 XQuery;

(2)读取元数据字典中数据源信息;

(3)分解全局查询 XQuery,将其分解为 for 子句、where 子句、order by 子句和 return 子句;

(4)确定全局查询所涉及的数据源;

(5)根据分解后的变量绑定、谓词操作、排序列表以及 XML 结构重构子查询 $Q_i (1 \leq i \leq n)$;

(6)构造 for 子句:

for(每个数据源 S_i)

解析 for 子句 $\$var$ in E ,确定绑定变量 $\$var$ 及路径表达式 E ;

依据模式映射信息计算 E 的映射表达式 E_i ;

构造迭代项 for 子句 $\$var$ in E_i ;

(7)构造 where 子句:

依据模式映射信息计算 operator 左边映射表达式 E_{L_i} ;

if(operator 右边是常量)

{依据模式映射信息找到数据源 S_i 对应 E_{L_i} 的路径表达式并将它添加到 where 子句中;}

else{依据模式映射信息计算的 operator 右边

映射表达式 E_{R_i} ;

if(E_{L_i} 和 E_{R_i} 的数据源相同)

{将此 operator 添加到 where 子句中作为连接条件;}

else{将两数据源子查询间的连接条件加入到连接列表;}}

(8)构造 order by 子句:

从全局查询分离出排序说明 sorting;

构造各数据源的排序项 E_i sorting;

(9)构造 return 子句:

获取绑定变量 $\$var$ 及路径表达式 E ;

根据模式信息利用 E 计算出的 E_i ,构造投影项 $\$var \rightarrow E_i$;

产生对应数据源 S_i 的 return 子句;

由算法可知,对合法且标准化的全局查询 Q,首先要根据元数据字典的局部数据源的注册信息确定全局查询 Q 所涉及的数据源个数和模式映射信息,然后循环调用涉及的数据源,依据全局模式与局部模式的映射信息依次重新构造 XQuery 子查询.

例1 现有一个图书网站运行着两个数据库,分别存储着书籍信息和书籍评论,现在要求查找2000年之后的书籍名、作者、评论信息,全局查询分解如表1所示.

表1 全局查询分解

全局查询 Q	子查询 Q_1	子查询 Q_2
for \$b in lib/book where \$b/book/year>2000 order by \$b/book/year return <tile>{\$/tile}</tile> <author>{\$/author}</author> <review>{\$/review}</review>	for \$b in lib1/book where \$b/book/year>2000 order by \$b/book/year return <tile>{\$/tile}</tile> <author>{\$/author}</author>	for \$b in lib2/book where \$b/book/year>2000 order by \$b/book/year return <tile>{\$/tile}</tile> <review>{\$/review}</review>

3.3 查询优化

异构数据集成的目的是为了更为有效地查询分布在各个异构数据源中的数据,所以查询优化^[7]成为了影响集成系统性能的关键. 与传统的数据管理系统的查询优化相比,异构数据集成系统因其具有局部数据源的分布性、自治性和异构性,所以使查询优化显得更为复杂.

查询优化处理技术主要分为两大类:物理优化和逻辑优化. 异构数据集成系统的查询优化不仅要考虑物理优化,如减少 I/O 次数、降低 CPU 计算时间和计算数据传输代价,更为主要的是考虑提高 XQuery 查询语言效率和制定高效查询计划的逻辑优化.

异构数据集成系统为各局部数据源建立了输出模式 XML Schema,本文提出的查询优化算法充分利用 XML Schema 对各局部数据源的结构和内容描述来优化 XQuery 查询,主要思路是将 XQuery 中的路径表达式和 XML Schema 分别映射为查询树和模式树,建立模式树节点和 XML 文档节点对应关系,然后利用模式树对查询树进行逻辑优化,经过谓词判断得到符合条件的模式节点编码,而后在实例 XML 文档中获得查询结果并进行结果整合,最后将整合结果提交用户.

3.3.1 模式树和查询树

模式树^[8]是由局部数据源的 XML Schema 生成,其中的元素、属性和文本映射为树中节点,如图

2 所示。子查询 XQuery 语句的路径表达式可以生成一个查询树,例如:

```
for $b in ("db.xml")/bookmanagement
where $b/reader/readername="Jack" and
$b/book//author/name="O. Henry"
return $b/retdate
```

查询树表示了查询感兴趣的一组有祖先后代关系的节点,其中单线边表示父子关系,双线边表示祖先后代关系,如图 3 所示。

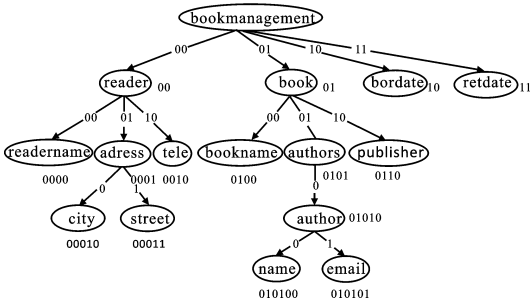


图 2 模式树编码示例

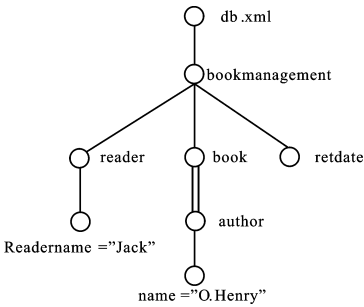


图 3 查询树示例

子查询 XQuery 语句的路径表达式在模式树中可能由一个或者多个简单路径构成,而最后的查询结果也是这几个简单路径查询结果做关系运算的的整合。

3.3.2 XML Schema 编码和 XML 文档编码

XML Schema 规定了 XML 文档中的元素、属性和数据类型。本文在模式树编码中引入了前缀码,如图 3 所示。前缀码^[9]很好地保留了节点之间的路径信息,可以保证与查询树的快速匹配。对 XML 文档采用区域编码的方法,同时保留模式树中的节点编码,XML 文档树的节点 v 编码为:

$$\text{code} = (\text{start}(v), \text{end}(v), \text{level}(v), \text{pcode}(v)) \quad (1)$$

式中, $\text{start}(v)$, $\text{end}(v)$ 表示深度优先遍历时第一次访问节点 v 和第二次访问的顺序值, $\text{level}(v)$ 表示节点 v 在文档树中的层数, $\text{pcode}(v)$ 表示节点 v 在

模式树中编码。查询时先检验查询树是否能与模式树相匹配,如果符合查询条件,则利用模式树中查询结果在模式树对应的 XML 文档进行实例节点匹配。

3.3.3 基于 XML Schema 的查询优化算法

考虑到路径表达式中的非确定因素“//”和“*”，本文这里采用文献[10]中的方法将子查询 XQuery 语句中的路径表达式分解为主路径多个简单路径,来解决路径表达式含有“//”和“*”的情况,然后用分解后的路径片段在模式树中查找对应模式节点,分解规则如下:

- (1) 每个路径片段均为最长的简单路径;
- (2) 遇“//”将路径分解,支持通配符,如 $/a/b//c/* /d$ 分解为 $/a/b$ 和 $//c/* /d$ 。

在路径片段的查询中,如果在模式树中找不到满足路径片段的模式节点,则无查询结果,反之如果局部数据源为 XML 文档,则利用满足这些简单路径的模式节点编码在实例 XML 文档树中查找相应节点元素并做谓词判断,得到最终的查询结果,如果局部数据源为关系数据库,则将优化后的子查询 XQuery 语句转换为 SQL 语句执行本地查询,并将查询结果转换为 XML 文档提交用户,至此便完成了异构数据集成系统的查询优化,具体算法如下。

算法 2 基于 XML Schema 的查询优化

输入:子查询 XQuery 语句

输出:查询结果 XML 文档

Step1(查询初始化):

将子查询 XQuery 转换为查询树 QT;

先序遍历查询树 QT 按规则分解查询树得到主路径 P_M 和各分支路径 $P_i (i=1, 2, 3 \dots, m)$;

将路径信息存入数组 L;

Step2(模式节点匹配):

主路径 P_M 与模式树根节点进行匹配;

if(不存在满足查询条件的节点)

return 无查询结果

else

for($i=1; i \leq m; i++$)

利用索引在模式树中查找分支路径 P_i 终节点;

记录查询所得节点 T_R 和对应模式编码;

Step3(本地数据源查询执行):

if(局部数据源为关系数据源)

将查询结果 T_R 转换为 SQL 语句执行;

按照映射关系将查询得到的关系数据转换为 XML 文档;

else

根据查询所得节点 T_R 和对应模式编码查找 XML 文档索引,得到目标点集合;

作谓词判断得到各分支路径满足条件的目标点集合;

节点集合进行关系连接运算得到最终的目标点集合;

return 最终结果集 XML 文档;

算法通过将子查询 XQuery 的分解为主路径和分支路径来解决非确定因素“//”和“*”的情况,然后利用路径片段在模式树中查找符合条件的模式节点编码,最后针对异构数据集成系统的关系数据源进行转换 SQL 语句执行,具体转换方式在文献[11]中涉及,本文不作详细介绍,针对 XML 格式的半结构数据源根据模式节点编码在 XML 文档中进行节点筛选和谓词操作. 该算法充分利用了模式映射为各局部数据源建立的 XML Schema,减少了连接操作的次数,提高了系统查询效率.

该算法利用了 XML Schema 所提供的语义信息,来避免对 XML 文档的完全遍历,在很大程度上提高了查询的效率.

3.3.4 实验与分析

为了验证查询优化算法的正确性和有效性,本文进行了查询优化算法在数据集上的测试实验,测试主要针对基于 XML Schema 的查询优化算法(XSEO),实验环境为一台 intel Core i3, 2 GB RAM,500 GB 硬盘的 PC 机,操作系统为 windows 7,开发工具用 VC++6.0,由 XMark 生成 5 个不同大小的测试数据集,实验涉及的查询如表 2 所示.

表 2 实验用到的 XQuery 查询

查询	XQuery
Q ₁	Q ₁ = for \$b in /bookmanagement/book[name = Jane Eyre]/publisher return \$b
Q ₂	for \$b in //authors/author[name=Catherine]/email return \$b
Q ₃	for \$b in //reader/* /city return \$b
Q ₄	for \$b in/bookmanagement/[reader/readername]book return \$b

实验所涉及的查询包含“//”和“*”等特殊情
况,具有代表性和覆盖性. 本文将 XSEO 算法与
Twiglist 算法^[12]分别进行了实验. 图 4 是在同一个
4.9 MB 的测试数据集上,对每个查询反复执行 5
次所求的平均执行时间. 图 5 是在 2.1 MB,3.5
MB,4.9 MB,6.3 MB,8.3 MB 的测试数据集上,对

查询 Q₄ 反复执行 5 次所求的平均时间.

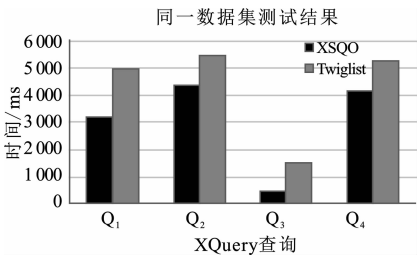


图 4 同一数据集测试结果

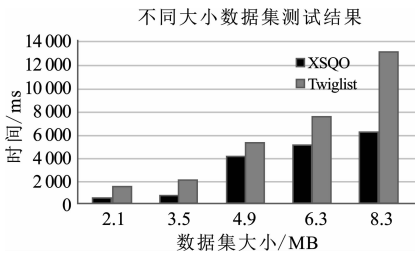


图 5 不同大小数据集测试结果

根据两次实验结果之间的对比情况,可以得出
以下结论,基于 XML Schema 的查询优化算法在查
询处理的时间上优于 Twig 算法. 主要原因在于基
于 XML Schema 的查询优化算法给模式树编码提
升了模式树与实例树匹配效率,充分利用局部数
据源的 XML Schema 信息,缩小了实例文档的匹
配范围,有效提高了查询效率.

4 结束语

本文针对异构数据集成系统的查询处理过程,
尤其是从查询分解和查询优化两个方面进行了
研究. 查询分解方面设计了依据元数据字典的查
询分解算法,能够实现全局分解的高效和准确性,
子查询优化方面提出了基于 XML Schema 的查询
优化算法,利用路径表达式分解、模式树与实例
树编码匹配和局部数据源 XML Schema 信息,并
有效了处理“//”和“*”情况,缩小了 XML 文
档节点的匹配范围,提高查询效率. 实验结果证
明了查询优化算法的正确性和有效性. 下一步的
研究方向是如何处理 XML Schema 中元素的引
用关系以及设计一种为非结构数据在集成系统
中建立正确 XML Schema 的方法.

参考文献:

[1] Chen H, Ouyang Y, Jiang W. An optimized data integration model based on reverse cleaning for heterogeneous multi-media data[J]. Multimedia Tools & Ap-

plications, 2015(2):1-16.

[2] 尚蕾,孙志挥. 基于 XML 的异构数据集成系统的查询处理[J]. 计算机工程, 2005, 31(5):79-81.

[3] 何志学,廖湖声. 一种基于语义信息的 XML Twig 查询处理方法[J]. 微电子学与计算机, 2015, 32(5): 95-98.

[4] 李军怀,周明全,耿国华,等. XML 在异构数据集成中的应用研究[J]. 计算机应用, 2002, 22(9):10-12.

[5] 夏凉. 面向 XQuery 语言的树模式提取方法[D]. 北京:北京工业大学, 2015.

[6] 余乐清. 基于 XML 的多源异构数据集成系统的研究与实现[D]. 西安:西安电子科技大学, 2011.

[7] 刘旭红,王春莲,施运梅. 异构医疗数据集成中的查询解析解析技术[J]. 计算机应用, 2010, 30(增刊 2): 156-158.

[8] 田雪,朱晓杰,申培松,等. 基于相似查询树的快速密文检索方法[J]. 软件学报, 2016, 27(6):1566-1576.

[9] 文思,文贵华. 基于扩展前缀编码的左孩子右兄弟结

构连接算法[J]. 计算机工程与设计, 2010, 31(10): 2312-2315.

[10] 张剑妹,陶世群. 基于模式的 XML 路径表达式查询处理技术[J]. 计算机应用, 2009, 29(11):3080-3083.

[11] 王振辉. 基于 XQuery 的 XML 与 SQL 查询转换算法研究[J]. 科学技术与工程, 2010, 10(27):6769-6771.

[12] 高万辰,廖湖声,苏航,等. 基于部分求值的 Twig 查询优化[J]. 计算机工程, 2016, 42(3):53-60.

作者简介:

毛佳飞 男, (1992-), 硕士研究生. 研究方向为数据库技术、数据集成. E-mail: 1660702907@qq. com.

叶 霞 女, (1977-), 博士, 副教授. 研究方向为数据库技术、计算机网络.

李俊山 男, (1956-), 博士, 教授. 研究方向为数据库技术、网络信息安全.

(上接第 44 页)

方法,图形用户界面降低了平台的搭建难度,使用脚本语言编写的程序使得评测平台的搭建过程自动完成. 评测人员不需要掌握平台的具体结构,根据软件提示就能完成 IP 核评测工作,减少评测所需时间,提高评测效率.

参考文献:

[1] 肖羽祥,周继承,罗宏伟. 基于工具软件的数字 IP 软核评测平台设计[J]. 微计算机信息, 2008, 24(33): 225-227.

[2] 李杰,肖立伊,赤诚,等. 基于 SoC 系统的 IP 核评测平台开发[J]. 微电子学与计算机, 2017, 34(6): 45-48.

[3] Stohr B, Simmons M, Geishauser J. FlexBench: reuse of verification IP to increase productivity[C]//Design, Automation and Test in Europe Conference and Exhibition, 2002. Proceedings. Nice, France. IEEE, 2002: 1131.

[4] 徐栋磊. VIP 及其在验证平台中应用的研究[D]. 成

都: 西南交通大学, 2007.

[5] Owczarek M S, Langer M, Wozny J, et al. Design of graphical user interface (GUI) for analog EDA tool [C]//Modern Problems of Radio Engineering, Telecommunications and Computer Science, 2004. Proceedings of the International Conference. Australian, IEEE, 2004: 560-562.

[6] Biham E, Shamir A. Differential cryptanalysis of the data encryption standard[M]. New York: Springer Science & Business Media, 2012.

作者简介:

赤 诚 男, (1993-), 硕士研究生. 研究方向为先进集成系统与 SoC.

肖立伊(通讯作者) 女, (1961-), 博士研究生, 教授, 博士生导师. 研究方向高可靠集成电路设计、SoC/IP 核设计与标准化. E-mail: xiaoly@hit. edu. cn.

李 杰 男, (1990-), 博士. 研究方向为集成电路抗辐射可靠性设计.