一种类Spreadsheet结构的信息汇聚方法

魏永山[1] 韩燕波[1,2] 孙忠林[1] 张 峰[1] 陈 欣[1]
(山东科技大学信息科学与工程学院 青岛 266510)[1] (中国科学院计算技术研究所 北京 100083)[2]

摘 要 SpreadSheet样式的数据操作具有很好的可用性,但在SpreadSheet结构中如何表示并操纵XML数据以及如何使用复制、粘贴、移动等简单操作表示复杂的XQuery查询是两个难点问题。提出一种基于XML模式的操作表示方法,将复杂的XQuery查询语句分解为XML模式上的粘贴节点、移动节点等操作,从而可以表示XQuery语言的核心语句FLOWR。在Spreadsheet结构中将XML模式显示为嵌套表格,用户在嵌套表格上的操作转换为XML模式上的操作。使用该方法可以构造多数据源的XQuery查询,并实现了概念验证的原型系统。与当前流行的XQuery查询构造工具相比较,原型系统更适合于无编程经验的最终用户构造XQuery查询。

关键词 Spread Sheet,XQuery,信息集成,信息资源汇聚
中图法分类号 TP315 **文献标识码** A

Spreadsheet-like Construct for Information Convergence

WEI Yong-shan[1] HAN Yan-bo[1,2] SUN Zhong-lin[1] ZHANG Feng[1] CHEN Xin[1]
(College of Information Science and Engineering, Shandong University of Science and Technology, Qingdao 266510, China)[1]
(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100083, China)[2]

Abstract Spreadsheet is intuitive for many non-technical users. But there are two difficulties in operating xml data on spreadsheet. One is how to represent xml data in spreadsheet, and the other is how to express complex query with simple operations, such as copy, paste and move. A spreadsheet-like construct was proposed to represent XML data and to express complex query using a group of simple operations. XML Schema was represented as nested-table in spreadsheet-like construct and users' operations on spreadsheet were transformed into XQuery statements on xml schema. The spreadsheet-like construct was implemented in an information convergence system to query complex data distributed in heterogeneous data sources. Compared with popular XQuery constructing tools, spreadsheet is suitable for users with little or no experiences in programming to query complex data.

Keywords Spread sheet, XQuery, Information integration, Information resource convergence

1 前言

无编程经验的用户使用XQuery语言查询XML数据源是非常困难的。传统的可视化的XQuery构造工具,比如Stylus Studio[20],Altovo MapForce[19],使用模式映射为基础,真实地表现了XQuery语言的复杂性。虽然开发人员可以使用此类软件构造复杂的XQuery语句,但这种模式并不适合无编程经验的用户。主要原因在于此类XQuery构造工具将查询结果与查询操作分离,在构造过程中无法直接看到操作的效果。无编程经验的用户更喜欢立即看到施加操作后的效果,并且根据当前的数据决定下一步需要的操作。

为解决类似问题,Shneiderman等提出"直接数据操作"(direct data manipulation)[1,2]来表示这种需求。直接数据操作具有3个特点:①持续显示用户感兴趣的内容;②简单的数据操作;③操作可以增量式地施加在数据上,并直接显示操作后的效果。在文本编辑领域以及报表制作领域也存在类似的需求,如所见即所得(WYSIWYG, What You See Is What You Get)。

使用"直接数据操作"模式构造XML数据源上的XQuery查询时,需要将数据源的XML数据显示给用户,用户在这些数据上做数据分析和操纵。最初这些数据是从数据源中提取的,用户对这些数据施加操作后,将数据的操作直接作用在数据集合上,并将经过操作变换的数据立即显示给用户。用户经过一系列的数据操作后得到经过一组数据操作变换后的数据。为满足直接数据操作的第2和第3个要求,这些面向用户的数据操作必须相当简单,并且用户的操作立即作用在数据源上。

Spreadsheet(比如微软的Excel, Open Office的Calc)是

到稿日期:2009-06-26 返修日期:2009-09-10 本文受国家自然科学基金项目(No. 70673098),山东省泰山学者项目,山东科技大学科研计划春蕾项目(No. 2008AZZ056)资助。

魏永山(1977—),男,博士生,讲师,主要研究方向为信息集成、服务计算等,E-mil:w3wei@sina.com.cn;韩燕波(1962—),男,教授,博士生导师,主要研究方向为互联网应用、服务计算等;孙忠林(1962—),男,教授,主要研究方向为信息集成、服务计算等;张 峰(1982—),男,讲师,主要研究方向为信息集成、服务计算;陈 欣(1977—),女,博士生,讲师,主要研究方向为信息集成、服务计算。

一种流行的采用直接数据操作模式的分析方法，通常用于分析从数据库提取的数据。文献[3]提出一种用于查询关系数据库的 Spreadsheet 方式的操作代数。文献[4,5]提出使用 Spreadsheet 方式构造面向数据操作与聚合的 Mashup。这些方法仍不能直接应用于构造查询并操纵 XML 数据。采用 Spreadsheet 方式构造 XML 查询仍然存在以下两个问题。

如何显示并操纵 XML 复杂数据。Spreadsheet 结构可以灵活地显示数据，但仅限于简单类型的数据，如字符串、整型等。Spreadsheet 的数据模型实际上是一种无结构的数据模型。但 XML 数据模型比 Spreadsheet 结构、关系数据模型要复杂得多，可以看作是一种树型层次结构，可以支持嵌套结构数据。XML 数据模型是一种面向机器处理的数据表示形式，并不是用于显示的数据表示形式，因此需要在 XML 数据模型和 Spreadsheet 模型间做变换，并且需要将用户在 Spreadsheet 数据模型上的操作变换并作用到 XML 数据模型上，然后将 XML 表示的操作再次变换为 Spreadsheet 数据模型表示并展示给用户。

如何分解并表示 XQuery 操作。在 Spreadsheet 中用户常用的操作有复制、粘贴、移动、排序、计算单元格等简单操作，每个操作都不复杂，用户使用这些简单并且容易理解的操作通过叠加方式组合为复杂的数据操作，表达复杂的业务逻辑。而描述型的 XQuery 语句则较为复杂，比如核心语句 FLOWR 由 For，Let，Order，Where 和 Return 子句组成，使用 XQuery 直接表示复杂的业务逻辑不仅容易出错而且可能需要写冗长的描述性的代码。两种不同编程模型间需要建立一种转换方式，要求使用 Shreadsheet 上简单易懂的操作表示 XQuery 语句的操作。

为解决上述两个问题，提出一种类 Spreadsheet 结构的 XQuery 查询构造框架，支持可视化显示、访问并操作多 XML 数据源。框架具有以下特点。

为支持访问并操纵 XML 数据，使用基于 XML 模式的树模型组织 XML 数据源，并将其作为 Spreadsheet 与 XML 数据源间的中间模型。在 Spreadsheet 中以嵌套表格形式显示基于 XML 模式的树模型，用户在嵌套表格上做的数据操作直接作用在树模型上，从树模型转换为 XQuery 语句，执行 XQuery 语句得到数据结果，再将变换后的模式及数据以嵌套表格形式显示给用户。

以 XML 模式为基础，提出一组 Spreadsheet 操作表示方法，用于表示 XQuery 的复杂查询语句。用户可以使用树模型的改变节点层次操作表示 XQuery 中的分组操作并在此基础上可以完成聚合操作，使用树模型的添加数据源节点操作可以表示 XQuery 中的连接操作，使用删除节点操作表示投影，在嵌套表格的标题上设置条件可以表示过滤操作等等。

实现了一个 Spreadsheet 模式的 XML 数据源的显示、查询和操作的界面，并且与当前主流的 XQuery 构造工具做了试验比较，用户使用界面可以较快地表示查询并获取结果。

本文在第 2 节中给出树模型以及在此基础上表示 Spreadsheet 操作；第 3 节介绍树模型的可视化方法；第 4 节给出与流行的 XQuery 构造工具的比较试验以及评价；第 5 节给出相关工作；最后给出结论并讨论进一步工作。

2 树模型及其操作

XML 文档具有明显的层次结构，XML 模式可以使用层次的树模型表示。

2.1 树模型

树模型定义为 $T::=Dom|\{T\}|\langle A_1:T_1,\cdots,A_k:T_k\rangle$

式中，T 为树模式，Dom 为值域，$\{T\}$ 表示集合类型，$\langle A_1:T_1,\cdots,A_k:T_k\rangle$ 表示序列类型，其中 A_k 表示唯一的标识名称，在 *XML* 中即为标记名称。

例如 books 模式可以表示为 books:{⟨book:⟨isbn:integer,title:NCName,year:integer⟩⟩}。

也可以直观地使用树型结构表示树模型。使用图形表示的 books 和 authors 的树模型分别如图 1(a)和(b)所示。

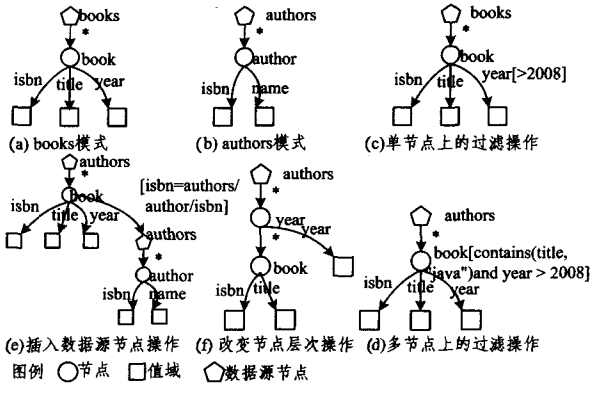

图 1 树模式及其典型操作

模式 T 的实例定义为

$t:=x\in Dom|\{t\}\langle A_1:t_1,\cdots,A_k:t_k\rangle,t_k\in T_k$

例如 t=books:{⟨book:⟨isbn:978-7-04-XXX,title:数据库系统概论,year:2004⟩⟩}。

在树模型上表示查询，把查询操作直接应用在树模型上，这是因为树模型查询使用一组简单的操作表示复杂查询，为用户提供一种近似于示例查询的表示方法。这种表示方法在查询表示上虽然繁琐，但对用户透明。下面介绍在树模型上的几种简单的操作。

2.2 操作及其语义

过滤操作 Filter(T,A_i,conditions) 在树模型的节点 A_i 上施加条件 conditions，显示数据时仅显示符合条件的数据。如果条件仅涉及一个节点，则附加在此节点上。如图 1(c)所示，查询 2008 年以后的书籍，也可以表示为

books:{⟨book:⟨isbn:integer,title:NCName,year[year>2008]:integer⟩⟩}

如果涉及多个节点或多个层次的节点，则在所有涉及到的离根节点最近的节点上施加组合条件。如果在多个节点上均存在条件，则这些条件使用与操作连接。如图 1(d)所示，查询 2008 年以后并且书名中包含 java 的书籍，也可以表示为

books:{⟨book[contains(title,"java") and year⟩ 2008]:⟨isbn:integer,title:NCName,year:integer⟩⟩}

删除节点 Remove(T,A_i) 删除模式树中的 A_i 节点及其子节点。

复制操作 Copy(T) 创建与指定模式树 T 完全相同的模式树 T'。

新建节点 New(T,A_i,A_j) 在指定节点 A_i 下创建节点 A_j。当 A_i 为 null 时创建根节点及数据源节点。为在操作序列过程中表示每个节点的数据来源，根节点称为数据源节点。

插入数据源节点 Insert(T_i,A_i,T_j,A_j,conditions) 将

来自 T_j 的节点 A_j 及其子节点插入 T_i 中作为 A_i 的子节点 A_j'，并将 A_j 中满足条件 conditions 的实例作为 A_j' 的实例。如图 1(e)所示，将 authors 插入到 book 节点下。其语义是根据 isbn 将作者插入至对应的 book 中。如果不指定条件，则在每个 book 节点中都会插入所有的 authors 节点。

改变节点层次 Move(T,A_i,A_j)　在 T 中移动节点 A_j 为 A_i 节点的子节点。比如把一个节点改变为父亲节点的兄弟节点，其语义实际是分组；节点变为子节点的兄弟节点，语义是取消分组或嵌套。如图 1(f)所示，将 year 节点改变为与 book 节点同层，其语义是按照 year 分组，year 的值与 book 变为一对多的联系类型。

排序 Sort(T,A_i,{asc | desc})　在节点 A_i 上指定排序方式为升序 asc 或降序 desc，仅用于数据显示，不改变数据源中节点顺序。所有节点上的排序方式按照前序遍历得到的顺序排列，即先按父亲节点排，再按照子节点排，子节点按照从左到右的顺序排列。

改名 Rename(T,A_i,A_i')　将 T 中 A_i 节点名称改为 A_i'。

根据是否改变模式，上述操作可以分为两类：一类改变模式操作，如删除、复制、新建、插入等操作，同时改变数据以保持数据与模式的一致；另一类操作不改变模式，如排序和过滤，只改变数据的显示。

可以改变模式的树模式操作，其输入是树模式及其实例，输出也是树模式及其实例，输出的实例是满足树模式的。由于操作的输出是模式及其实例，因此这些操作是可以叠加的，并且叠加的操作序列的输出也是树模式及其实例。

通过这些简单操作在树模式上的叠加产生一个在树模式上的操作序列，可以表示复杂的 XML 查询。下面考察树模式上的操作序列的表达能力。

2.3　树模式操作的表达能力

可以从两个方面考察查询树模式的表达能力：一是查询树模式向 XQuery 的转换，二是将给定的 XQuery 语句转换为查询树模式。下面通过两个算法说明这两个问题。

算法 1　将查询树模式转换为 XQuery 语句

输入：查询树模式

输出：XQuery 语句

L1：从根节点开始前序遍历查询模式树；

L2：输出当前节点标记；

L3：当前节点如果有子节点则在 Xquery 语句中使用 for 语句遍历此节点；

L4：如果节点上存在附加的条件，则将此条件附加在 where 子句中；

L5：如果没有子节点则在 XQuery 语句中输出节点的值；

L6：如果子节点上存在附加条件，则将此条件以与方式附加在 where 子句后；

L7：如果有数据源节点，则在 XQuery 语句中使用 for 语句遍历此数据源中的所有节点；

L8：如果数据源节点存在条件，则将条件附加在 for 语句的 where 字句中；

L9：如果是根节点，则退出，否则返回父节点。

例 1　将图 1(e) 所示的查询树模式转换为 XQuery 语句。这是一个内外相关的嵌套查询语句，如图 2 所示。

```
L1:  〈books〉{
L2:    for $book in
L3:        doc('books.xml')/books/book
L4:    return
L5:     〈book〉
L6:        〈isbn〉{ $book/isbn/text() }
L7:          〈/isbn〉
L8:        〈title〉{ $book/title/text() }
L9:          〈/title〉
L10:       〈year〉{ $book/year/text() }
L11:         〈/year〉
L12:       {
L13:         for $author in
L14:       doc('authors.xml')/authors/author
L15:         where $author/isbn = $book/isbn
L16:         return
L17:          〈authors〉
L18:            〈author〉
L19:              〈isbn〉{$author/isbn/text()}
L20:                〈/isbn〉
L21:              〈name〉{$author/name/text()}
L22:                〈/name〉
L23:            〈/author〉
L24:            〈/authors〉
L25:          }
L26:     〈/book〉
L27:  }〈/books〉
```

图 2　表示 books 模式和 authors 模式连接的 XQuery 语句

下面来考察如何使用模式树上的操作表示一个给定的 FLWOR 语句。

算法 2　使用模式树操作表达 FLWOR 语句

输入：FLWOR 语句

输出：查询树模式

L1：使用创建新节点操作建立与 for 语句中指定数据源相应的节点；

L2：Let 语句的作用是建立中间变量，其实每个节点都对应一个中间变量；

L3：将 where 子句的条件表达式分解为节点上附加的过滤条件；

L4：根据 order 子句中指定的排序节点标记分别在树模式上附加相应的排序方式；

L5：上述操作完成后即可以完成 return 子句。

分组操作可以由改变节点层次实现。分组操作的语义是在分组属性和其他属性间建立一对多的联系，因此只将需要分组的节点移动到其他属性的上一层即可获得一对多的语义联系。相反的操作是取消分组，实际的语义是取消嵌套层次。涉及多个数据源的多 for 语句的嵌套其实际的语义是在多个数据源间实施连接操作，可以使用插入数据源节点操作将数据源附加为其他数据源的子节点，在此基础上的后续操作可以参考算法 2。数据源的合并操作可以通过插入实例操作实现。

3　树模型可视化

在 Spreadsheet 中使用嵌套表格表示树模型。嵌套表格分为标题和数据两部分。用户的操作以列或行为单位，在列上的操作映射为树模式操作，在行上的操作映射为数据操作。

在列标题上可以直接输入过滤条件，通过单击列标题排序。插入数据源节点操作可以通过鼠标拖拽数据源至目标数据源的某个节点下，改变节点层次可以直接使用鼠标节点至上层节点或下层节点。因此树模式上的操作可以较为直观地在Spreadsheet上实现。在可视化方面部分借鉴了文献[18]关于嵌套表格的思想。

图3是books和authors两个XML数据源在Spreadsheet中的显示方式。图4中(a)所示为在year标题上附加条件[year>2008]，(b)所示为在book标题上附加条件[constinas(title,"java") and year>2008]，(c)所示为将year标题拖拽至books下。为维护一对多联系的完整性约束，会在books下自动产生一个YEAR节点，year节点与book同层，从而实现对book的分组操作。(d)所示为将authors数据源拖拽至books的book节点下，使得authors变为book的子结点。如果不附加条件，则在每个book实例下面均会添加所有的authors节点。为获得正常需要的结果，需要在authors标题上附加条件[isbn=author /isbn]，则只将符合条件的authors的author实例添加至book的实例作为子结点。

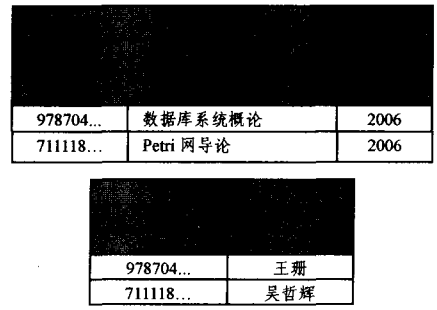

图3 Books和authors的嵌套表格表示

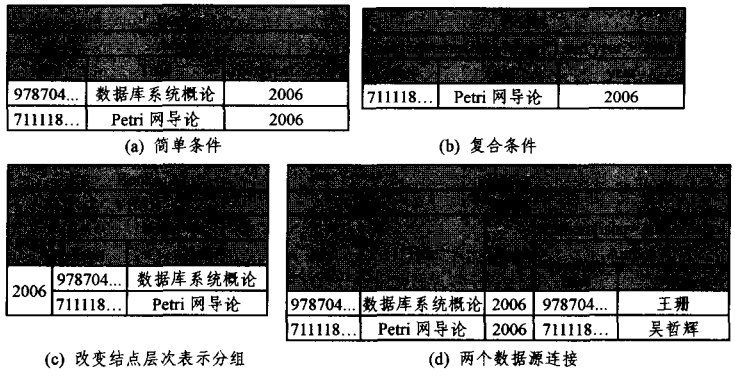

图4 使用嵌套表格表示模式树操作

4 试验与评价

用户评测的主要目标是评价原型系统的可用性，并且与当前流行的XML查询工具作对比。

主要比较了使用原型系统Spreadsheet方法和使用MapForce[19]工具构造XQuery语句。10个本科或研究生阶段的学生作为用户参与了试验。5个学生熟悉XML，其余5个是随机挑选的学生。

给这些用户提供了4个测试任务完成评价。这些任务均是在数据变换中常用的任务。下面按照从简单到复杂的顺序列出测试任务。

BM1：节点的直接映射。将customer表模式映射至全局ENT模式。这个版本的ENT全局模式已经包含了customer模式的结构，因此直接映射即可，不必在全局ENT模式中创建customer模式。

BM2：修改节点名称。修改名为CustomerFax的节点为Fax。

BM3：添加本地模式orders为全局模式的子节点。

BM4：构建多步骤的变换。从数据库erp_site1的关系表customers中提取数据，从数据库erp2数据库的关系表Orders中提取数据，连接customers和Orders表；按照CustomerID分组；选取订单时间在2008-12-1至2008-12-31之间的数据；按照订单总金额由大到小排序；选择前10条结果。

表1总结了使用两个工具完成任务所需时间的试验结果。在完成任务BM1时使用Spreadsheet界面的时间比使用MapForce的时间要长，说明Spreadsheet并不擅长模式映射的工作，因为在Spreadsheet中映射是通过修改字段的名称而不是通过鼠标的拖拽实现的，在后续工作中需要在Spreadsheet中引入模式映射的操作，以改进这方面的功能。用户使用了相近的时间完成简单的测试任务BM2。

表1 试验结果

任务编号	MapForce	Spreadsheet
BM1	4s	10s
BM2	5s	6s
BM3	30s	12s
BM4	400s	120s

对应相对复杂的测试任务BM3和BM4，用户使用MapForce完成任务的时间明显多于使用Spreadsheet完成任务的时间。对于测试任务BM3，可能的原因是MapForce不支持直接修改目标模式，必须使用另外一个工具，比如XML Spy编辑好目标模式，再使用MapForce打开，然后再完成测试任务。但原型系统提供了直接编辑Spreadsheet模式的功能。对于测试任务BM4，使用Spreadsheet方式的时间比Mapforce的时间少很多，可能的原因是任务具有明确的步骤，在MapForce中实现时需要先分析任务如何使用Mapforce的可视化元素表示，然后再组合在一起，前后步骤混合在一起表示为一个复杂的XQuery。但使用Spreadsheet的简单操作即可实现复杂的XQuery，并且每一步操作的结果用户都能立即看到，可以根据当前的数据显示观察与目标的距离，然后决定下一步采取什么操作。

试验任务结束后，每个用户回答一组测试问题。所有的用户认为 Spreadsheet 具有更强的可用性，其数据即时操作的特点比较适合于无编程经验的用户。所有的用户均认为 BM4 非常复杂，以至于在 MapForce 中难以分清模式间的连线，而在原型系统中以 Spreadsheet 非常直观。用户提供的负面反馈是在完成测试任务过程中原型系统有时会产生映射验证错误。原型系统不是很稳定，在映射验证上需要进一步提高。在下一步工作中从更多的方面使用更多的测试任务来比较、评价原型系统。

5 相关研究工作

Spreadsheet 已经被证明是用户友好的数据处理界面之一，微软的 Excel 被广泛采用也可以作为例证。Tableau[6] 建立在 VizQL[7] 之上，特点是交互的数据可视化，但查询能力有限。Spreadsheet 也用于数据清洗[8]、可视化探索[9]、图像管理[10]等。Witkowski 等人提出扩展 SQL，以使关系数据库管理系统支持 Spreadsheet 方式计算。

有很多在线数据库查询和管理工具也使用 Spreadsheet 方式的界面。Zoho[12] 数据库允许用户导入、创建、查询以及可视化在线数据库。但其查询能力相对有限，只支持简单查询和排序。Dabble[13] 数据库与 Zoho 数据库相似，增加了分组功能。

在关系数据库的可视化查询界面领域也有许多工作。文献[14]采用本体帮助用户构造查询语句。VisTrails[15] 为用户提供可视化界面，增量地修改一种工作流，以操纵数据库。文献[16]提出一种可视化查询语言以支持用户通过操纵模式树查询含有复杂值的关系数据库，其模式树与树模型很相似，但模式树上的操作没有考虑层次模型上的语义。

文献[3]提出一种用于查询关系数据库的 Spreadsheet 方式的操作代数。文献[4,5]提出使用 Spreadsheet 方式构造面向数据操作与聚合的 Mashup。

结束语 采用 Spreadsheet 结构提供直接数据操作能力，界面友好，适合无编程经验的用户构造查询。为解决 Spreadsheet 中处理复杂 XML 数据及查询表示问题，提出一种类 Spreadsheet 结构的基于 XML 模式的信息汇聚框架，以支持可视化显示，访问并操作多 XML 数据源。结构具有以下特点：①使用基于 XML 模式的树模型作为 Spreadsheet 结构与 XML 数据源间的中间模型。在 Spreadsheet 中以嵌套表格形式显示树模型，用户在嵌套表格上的数据操作转换为树模型上的 XQuery 查询，执行 XQuery 查询得到结果，再以嵌套表格形式显示给用户。②以 XML 模式为基础，提出一组简单的 Spreadsheet 操作表示方法，用于表示复杂的 XQuery 查询。③实现了一个 Spreadsheet 模式的 XML 数据源的显示、查询和操作的界面。无编程经验的用户通过界面可以使用简单的复制、粘贴、移动等操作构造复杂的 XML 数据源查询。

Spreadsheet 界面中用户可以直接修改数据源的数据并更新至数据源，但经过多个操作叠加产生的数据并不一定能直接映射到原始数据源，因此将进一步研究动态更新机制。操作的可交互性也是下一步研究的内容。

参 考 文 献

[1] Shneiderman B. The future of interactive systems and the emergence of direct manipulation[J]. Behaviour & Information Technology, 1982, 1(3): 237-256

[2] Shneiderman B. Direct manipulation: a step beyond programming languages[J]. IEEE Computer, 1983, 16(8): 57-69

[3] Liu B, Jagadish H V. A Spreadsheet Algebra for a Direct Data Manipulation Query Interface[C]// ICDE09. 2009

[4] Kongdenfha W, Eenatallah B, Vayssiere J, et al. Rapid Development of Spreadsheet-based Web Mashups[C]// WWW09. 2009

[5] Wang G, Yang S, Han Y. A Spreadsheet-like Construct for Streamlining and Reusing Mashups[C]// ICYCS08. 2008

[6] Tableau software[EB/OL]. http://www.tableausoftware.com/

[7] Hanrahan P. Vizql: a language for query, analysis and visualization[C]// SIGMOD. 2006: 721

[8] Raman V, Hellerstein J M. Potter's wheel: An interactive data cleaning system[C]// VLDB. 2001: 381-390

[9] Jankun-Kelly T J, Ma K-L. A spreadsheet interface for visualization exploration[C]// IEEE Visualization. 2000: 69-76

[10] Kandel S, Paepcke A, Theobald M, et al. The photospread query language[R]. Stanford Univ., 2007

[11] Witkowski A, Bellamkonda S, Bozkaya T, et al. Spreadsheets in rdbms for olap[C]// SIGMOD. 2003

[12] Zoho db & reports[EB/OL]. http://db.zoho.com/

[13] Dabble db-online database[R]. http://dabbledb.com/

[14] Catarci T, Dongilli P, Mascio T D, et al. An ontology based visual tool for query formulation support[C]// ECAI. 2004: 308-312

[15] Scheidegger C E, Vo H T, Koop D, et al. Querying and re-using workflows with vistrails[C]// SIGMOD. 2008

[16] Koch C. A visual query language for complex-value databases [J]. ArXiv Computer Science e-prints, 2006

[17] Dang-Ngoc T, Gardarin G. Federating heterogeneous data sources with XML[C]// Proceeding of IASTED IKS Conf. 2003

[18] 王桂玲. 用户主导的互联网虚拟应用构造原理与方法研究[R]. 中国科学院计算技术研究所, 2009

[19] Altova MapForce[EB/OL]. http://www.Altova.com/MapForce

[20] Stylus Studio[EB/OL]. http://www.stylusstudio.com

(上接第 133 页)

[8] Mahlke S A, Lin D C, Chen W Y, et al. Effective compiler support for predicated execution using the hyperblock[J]. SIGMICRON ew sl., 1992, 23 (122): 45-54

[9] Quinones E, Parcerisa J M, González A, et al. Improving branch prediction and predicated execution in out-of-order processors [C]// Proceedings of International Symposium on High-Performance Computer Architecture, 2007 IEEE 13th Annual International Symposium on High Performance Computer Architecture, HPCA-13. 2007

[10] Faraboschi P, Fisher J A, Young C, et al. Instruction scheduling for instruction level parallel processors [J]. Proceedings of the IEEE, 2001, 89 (11): 1638-1659

[11] 朱朝霞，周云才. 语法分析方法实践教学模式[J]. 重庆工学院学报：自然科学版，2008，22(6)：176-180