第46卷第1期 中国科学技术大学学报 Vol. 46, No. 1
2016年1月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Jan. 2016

文章编号:0253-2778(2016)1-0056-10

面向 Tabular 库的数据模型及其查询问题

黄冬梅[1],孙 乐[1],石少华[2],苏 诚[2],赵丹枫[1]

(1. 上海海洋大学信息学院,上海 201306;2. 国家海洋局东海预报中心,上海 200136)

摘要:信息化的发展使得数据存储及表示形式呈现出分布性、异构性的特点,不仅包括关系数据库、面向对象数据库等传统结构化数据,还包括 Excel、CSV 等不具有明确结构的特殊非结构化数据等,与此同时,其数据呈现了量大、更新快、可用性弱等大数据特点. 然而使用无结构和半结构化文档组织和管理 Excel 等表单数据,存在着数据弱可控、弱可用、及访问效率差的问题. 针对该类问题,本文以 Excel 文本为数据源,提出了一种新的面向 Tabular 库的关系数据模型并讨论了其上的查询及优化问题. 首先,给出了 Tabular 表单数据的形式化定义,其次,设计 PartiPath 划分树实现表格的关系划分及结构转换,在关系模型的基础上,给出其数据模型及数据模式,再者,定义了表单数据上的基本查询问题及融合用户兴趣指数改进查询相似度指标,最后给出实验分析并作出总结.

关键词:Tabular 库;查询;数据模型;PartiPath 划分树;关系模型

中图分类号:TP311 **文献标识码**:A doi:10.3969/j.issn.0253-2778.2016.01.008

引用格式:HUANG Dongmei, SUN Le, SHI Shaohua, et al. Tabular-oriented data model and its query issues [J]. Journal of University of Science and Technology of China, 2016,46(1):56-65.

黄冬梅,孙 乐,石少华,等. 面向 Tabular 库的数据模型及其查询问题[J]. 中国科学技术大学学报,2016,46(1):56-65.

Tabular-oriented data model and its query issues

HUANG Dongmei[1], SUN Le[1], SHI Shaohua[2], SU Cheng[2], ZHAO Danfeng[1]

(1. *School of Information, Shanghai Ocean University, Shanghai* 201306, *China*;
2. *East Sea Forecast Center of Oceanic Administration of China, Shanghai* 200136, *China*)

Abstract: With the rapid development of information technologies, data storage and representation of various sources, including not only the traditional structured data such as relational databases and object-oriented databases, but also those special unstructured data like Excel, CSV documents, manifest distributed and heterogeneous characteristics. Undoubtedly, all above data features high-volume, continuously-updating, low-usability, which falls into Big Data. However, the organization and management of Excel and other forms of data by using unstructured and semi-structured methods leads to a weakly-controllable, weakly-usable data structure with poor access efficiency. To solve this problem, this paper, taking Excel data source into consideration, aims to propose a new tabular-oriented relational data model and discusses Tabular querying and optimizing issues. Firstly, the formal definition of Tabular form data is given; secondly, PartiPath tree is designed to achieve structural transformation by tabular division

收稿日期:2015-08-27;**修回日期**:2015-09-29

基金项目:国家自然科学基金(61272098)资助.

作者简介:黄冬梅,女,1964 年生,教授,研究方向:数据挖掘. E-mail: dmhuang@shou.edu.cn

通讯作者:赵丹枫,女,博士/讲师. dfzhao@shou.edu.cn

and its relation schema as well; then its data model is presented. After that, four basic queries and their optimization by improved DICE with user interest similarity are described. Finally, the experiment was conducted and a conclusion was drawm.

Key words: Tabular repository; query; data model; PartiPath tree; relation model

0　引言

文本数据(尤其是 Excel,CSV 等)是一种特殊的非结构化数据,它兼具结构化数据的结构特征和文本数据的线性顺序性特点[1],该类文本缺乏可用的数据模型,通常需要人为规范描述、抽取表格内容,因此表单文本结构化自动转换是目前数据管理、信息抽取及文件分析系统应用中的关键问题.

近年来,全球数据存储量呈现爆炸式增长,企业及互联网数据以每年 50%的速率在增长,这些数据都属于大数据的范围. 如 Argo 目前在全球共投放浮标 10 231 个,对海洋要素数据进行实时采集,其中一个 Argo 数据中心在过去一年里就积累了逾 50 万条文本以及 21 954 条剖面 CSV 数据[2-3];NASA 发射的海洋观测卫星"水瓶座"仅 2 个月内采集的遥感数据量相当于调查船和浮标 125 年测量的历史记录;北海海洋监测中心自 1978 年至今已经收集了超过 20 G 的 Excel 表单数据. 根据 Google 统计,目前约 90%的数据并没有存储在关系数据库中,若使用无结构和半结构化的结构组织管理文本数据,则会大大降低数据可用性和访问效率,不利于数据的大批量分析和高精度处理[4].

本文提出如下研究问题:如何无需手动调整低层次的视觉及内容细节,以完成表格数据的合理划分及关系转换,从而实现特定领域内其结构化存储高效查询服务. 本文首先提出一种新的面向表格的数据模型,给出其形式化定义;其次,在关系模型的基础上设计其表格模型及关系模式,并定义了该模型上的正匹配查询、负匹配查询、析取匹配查询等的查询问题. 为了便于解决实际问题,本文限定数据源为符合层次结构且具有非递归模式的 Excel 文本数据. 具体任务包括文本数据基于 PartiPath 划分树的结构转换以及表单-关系的规范化输出.

1　相关工作

关于表格文本信息结构化转换的相关研究主要包括以下三方面:①表规范化:将表单转换成适配关系数据库的规范形式;②信息抽取:类似从文本中提取信息的过程,通过选择性提取特征,以产生一个目标数据库;③表单理解:包括从数据,标签(属性)及维度间发现关系,包括以下几个步骤[5]:(a)表单定位(从数据源中获取表格位置);(b)表单识别(发现独立单元 cell);(c)功能分析(区分属性和数据,区分单元角色(role));(d)结构分析(识别表格单元间的关系);(e)表单解释(从表中抽取可用实例).

目前的研究热点主要集中在表单解释. 表格的产生具有多样性,其复杂的数据格式源于企业组织的定制化设计、特定的排版需求及人为创造等,这些都会增加表格理解的复杂性. 现有的相关方法和系统主要基于启发式[6-7],机器学习[8-9],动态规划[10]或概率[11]等方式. 当前研究状态远没有彻底实现表格的充分理解及精准描述,大多数研究仍致力于低级别的表处理问题.

2　问题描述及定义

2.1　问题描述

例 2.1　划分图 1 中的表格并使用关系模型进行结构化入库. 对图 1 中初始表格进行层次划分[12],分析表单内部几何结构,以分隔线和迭代树的方式实现其结构表示. 然而该启发式划分方式仅从物理层交替切分,表格单元角色(role)及彼此关系并没有得到区分. 此外,目前的层次划分尚难以处

滨海旅游度假区监测数据报表

滨海旅游度假区名称:	秦皇岛亚运村滨海旅游度假区		
监测单位:	河北省海洋环境监测中心		
观测日期:	2008 年 9 月 5 日		
水文		08时 (北京时)	14时 (北京时)
水　温 (℃)		25.4	26.8
浪　高 (m)		2.3	1.1
涌　高 (m)		0.5	0.9
气象		08时 (北京时)	14时 (北京时)
天气现象		10	10
风　向		327	193
风　速 (m/s)		0.9	4.4
总 云 量		3	4
气　温 (℃)		23.2	27.9
能 见 度 (km)		6	8
水质			
透明度 (m)	盐度	溶氧mg/l	pH
1.2	31.836	9.25	8.08
观测者	孙大光	校对者	刘树田

图 1　海水浴场监测数据报表

Fig. 1　Bathing beach monitoring data example

理复杂表格或嵌套表格划分，导致其最后的结构很难转换成关系模型.

为了有效地防止上述现象的出现，本文提出新的层次划分方法 PartiPath. PartiPath 首先基于 Tijerino 表格规范化技术[13]，辨识各基础单元数据类型并提取出表格标签及数据；然后对表单区域按一定的模式及关系进行划分，直至满足划分结束条件. PartiPath 能够满足表格内部属性关系，使得在其划分过程中不丢失语义信息，在整体上保留了表单的一致性，是一种近乎无损的层次划分方法. 图 2 为 PartiPath 划分得到的层次结构树.

2.2 Tabular 库形式化定义

对于表格模型概念的界定纷繁复杂，从数据角度出发，可以归纳为：①由基础表格单元（样式、坐标、类型、标签等）构成的视觉层文本内容；②各基础表格单元上的“语义信息”，包括样本元信息关系（维度、行标签与列标签间的映射关系、实体）.

以图 1 为例，其表示为表格模型库中的一个表格片段，其中每张表都存在一个表格文档的根节点，表示为一个完整的表格文档，Tabular（表单）库通常由一个或多个表格文档组成，每张表格都有一个标识符＜table:id＞，＜header＞节点（如无标题则使用虚拟节点）的子节点可用于描述表格文档的元信息，＜text＞节点及其子节点描述文本片段的内容；如图 1 中的文档标题、名称、单位、时间、监测者等可抽取作为该文本的一个＜header＞片段；每张表都包括三个子节点“网格”、“方格”和“形符”（Token）. 对于形符＜t＞，该结点的属性 Level，Parent，Property 分别表示为形符 Token 上的三个不同的语义标注信息.

在表格理解的过程中需要严格遵循下述假设以满足本文各层次之间的关系.

（Ⅰ）cell 中仅包含文本字符. 尽管实际应用中，cell 可以是一个富文本，如 RTF（rich text format），图像或者公式（in Excel）. 为了简化数据结构设计，本文不涉及上述文本格式.

（Ⅱ）cell 可包含若干个连续行和列，即它可以覆盖若干个区域形成矩形.

（Ⅲ）cell 可以作为数据内容（entry）或标签（label）[14]. 一个 entry 代表一个数据值，一个 label 描述（控制）多个 entries.

（Ⅳ）label 可与其他（行或列上的）labels 形成关联的层次关系（嵌套关系）.

（Ⅴ）label 可对应一个属性维度（dimension）.

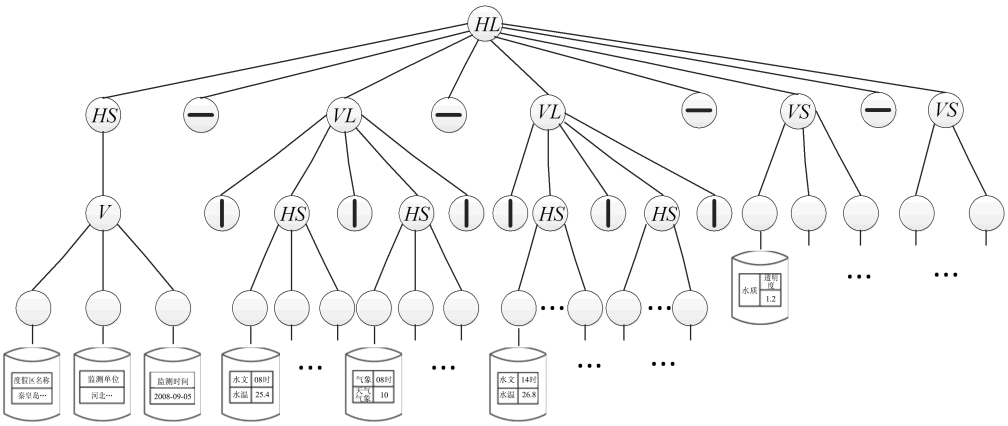

图 2 PartiPath 划分树示意图

Fig. 2 PartiPath partition tree structure

下面，我们给出表格模型及其相关概念的形式化定义：

定义 2.1 词汇表（alphabet）. 词汇表是由类符构成的有限集合 $\Sigma=\{\alpha_1,\alpha_2,\cdots,\alpha_n\}$（$n\geqslant 0$），类符 $a_i\in\Sigma$（$0\leqslant i\leqslant n$）.

定义 2.2 方格（tessellations）. 一个方格 $s=t_1,t_2,\cdots,t_n(n\geqslant 0)$ 是由形符及其属性表示的串. 其中，$t_i=t_{i1}=t_{i2}=\cdots=t_{ik}(0\leqslant k\leqslant n)$ 是类符在串中的一次出现，称为形符.

定义 2.3 网格（grid）. 一个网格 $g=s_1,s_2,\cdots,s_n(n\geqslant 0)$ 是由方格组成的序列.

定义 2.4 表格（table）. 一个表格 $t=g_1,g_2,\cdots,g_n(n\geqslant 0)$ 是由网格组成的序列.

定义 2.5 映射（Mapping）. 假设存在由全体

形符构成的有限集合 S，M_R，M_C分别是标签属性为“row_label”，“column_label”的形符构成的有限子集，一次映射是一个全函数 $f_M:\delta(M_R\times M_C)_\varepsilon\to\Delta$，也表示为一条基元路径. 以图 1 为例，形符“25. 4”需由集合 M_R中{08 时(北京时)}及集合 M_C中{水文. 水温}所决定.

定义 2. 6　标注(annotation). 假设存在由全体形符、方格、网格、表格分别构成的集合 W，S，P，T 以及两个互不相交的有限集 L_L和 L_V，L_L是标注类名的集合，L_V是值的集合，一次标注是一个全函数 $f_A:W\cup S\cup P\cup T\to(L_L\times L_V)_\varepsilon$. 表格模型标注提供了每一个形符的描述信息，Level 表示为形符 t 在结构树(PartiPath 划分树)中所在的层位置，Parent 描述了形符 t 间的父子依赖关系，Property 为其单元角色(role).

在操作中，给表格模型中的各 cell 添加信息的步骤及过程(包括映射和标注)称为对表格模型库的“加工”.

定义 2. 7　表格模型库 (Tabular repository, TR). 模型库 TR＝(D,A,F) 是一个三元组. 其中，D 是表格文档，D＝(Σ,T,P,S,W)，Σ 是词汇表；T 是全体方格构成的有限集合；P 是全体网格构成的有限集序列；S 是全体表格构成的有限序列；W 是全体形符构成的有限序列.

A 是语义信息，A＝(M_R，M_C，L_C，L_V)，M_R 是表格行标签的有限集合；M_C 是表格列标签的有限集合；L_L是标注类名的有限集合；L_V是标注内容的有限集合.

F 是表格文档的加工及结构表示，F＝(Σ，T，Syn，f_M，f_A，$T_i(i=1,\cdots,m)$，M_j)，Σ 是词汇表；$T\subseteq\Sigma$ 为其开始与结束形符元组；Syn ＝ {(i, j) | $i,j\in\{1,\cdots,m\}$} 表示为彼此可连接的元路径对. f_M 是 W 上的映射，f_M＝(M_R,M_C)，一次映射是一个全函数 $f_M:\delta(M_R\times M_C)_\varepsilon\to\Delta$，表示为一个元路径；$f_A$ 是对 R 中的形符的标注，标注是一个三元组 f_A＝(level,Parent,Property)，一次标注是一个全函数 $f_A:W\cup S\cup P\cup T\to(L_L\times L_V)_\varepsilon$；$T_i$，$i\in\{1,\cdots,m\}$，表示为由基元路径构成的集合，包括划分操作规则集 {$R(i,j)$| $(i,j)\in$ Syn,$i,j\in(1,\cdots,m)$}. 若一个操作规则对 (α,β)，$\alpha,\beta\in f_M$ 可允许相邻方格逐步合并，那么则说明 α,β 上的网格(或表格)可再分；M_j，为最终输出的 PartiPath 划分树.

以图 1 为例最终形成的层次树如图 2 所示，其中，从 VL 到叶子结点可表示一条元路径(元路径划分及表示详见 3. 2 节)，其边上带有划分属性，如 VL-HS 表示网格的一次纵向 PartiPath 划分.

3　PartiPath 划分树

层次划分树是一种表示文本数据的数据结构[12]，是完成表格数据存储及检索的一个基本任务，其设计通常分为由底向上或自顶向下两种方式. 层次划分可以从文本中提取单元信息，随后聚集更高的逻辑结构，并创建最终的数据模式或结构.

3. 1　基本定义

给定已知条件：根据表格规范化技术，完成表格数据区域的定位和单元角色(role)的识别，role∈{label,value}.

PartiPath 划分树通过提取适当的路径来描述内部结点的邻近关系，为数据模型和数据查询建立了应用基础. 其根结点对应文档中的表单区域，到叶子结点的路径表示为表单中的一条元路径，且每一层结果由横向或纵向交替划分得到.

定义 3. 1　PartiPath 划分树，P2Tree ＝ {N, A,A_N,A_A}，其中 N 为结点集合，A 为结点间弧边集合，A_N 为结点属性集，A_A 为边的属性.

图 3 是一个 PartiPath 划分树例子，给定已知条件. 图 3(a)的横线单元，其属性为标签，(d)为最终层次划分树，其结点和弧边属性定义如下：

定义 3. 2　结点属性集(node attributes)，是一个三元组：A_{N_i} ＝ {T_i,Pos_i,Child_i}，其中 T_i 为结点类型. Pos_i 为结点 N_i 所在区域(xu_i,yu_i,xb_i,yb_i)，分别对应其上下左右四个位置. Child_i 为 N_i 的孩子结点.

如果 N_i 为内部结点，划分树将执行层次切分，标识符 HL、VL 分别代表横向或纵向切分方式，HS、VS 表示其孩子结点为横向或纵向排列方式，V 表示没有切分. 若 N_i 为叶子结点，T_i＝L(-,|)，则结点记录一次划分操作，否则对应一条元路径.

定义 3. 3　弧边属性集是一个二元组：$A_{A_{i,j}}$ ＝ {$P_{i,j}$,$d_{i,j}$}，其中 $P_{i,j}$ 为结点 N_i 和 N_j 间的方位标识，$P_{i,j}$ ＝ {rightof,below}；rightof 表示 N_i 在 N_j 右方串连，below 表示 N_i 和 N_j 在不同的网格(grid,定义 2. 3)内，$d_{i,j}$ 为结点 N_i 和 N_j 间的距离差，表示层次差.

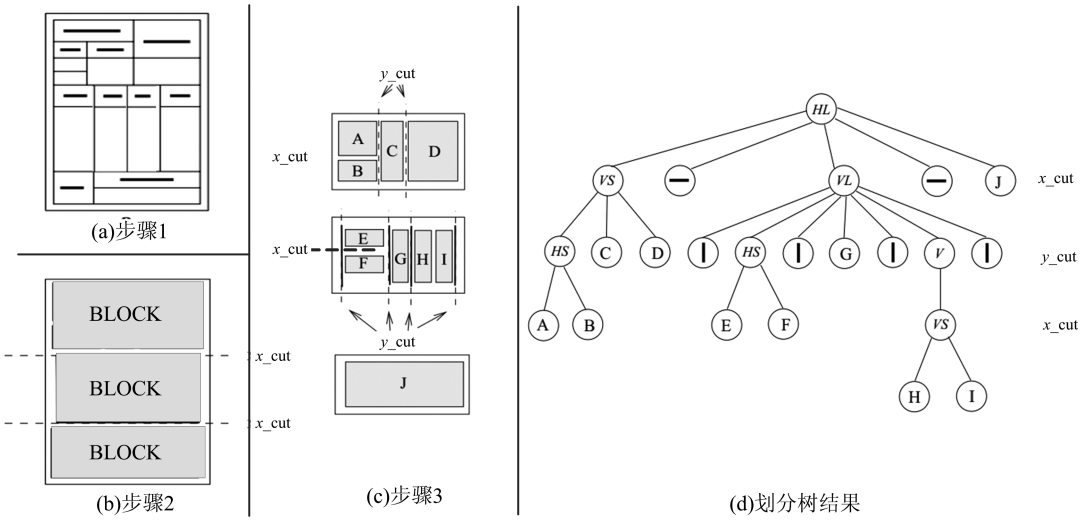

图 3 PartiPath 划分树

Fig. 3 PartiPath partition tree

例 3.1 如图 4 所示,(a)和(b)分别表示 PartiPath 划分树根结点上的一层和二层关联. 二层关联 BiCorrelation 结果表示为一个表格,一层关联 Correlation=<*ABCD*>表示为一个网格,其中 *A*,*B*,*C*,*D* 表示一个方格,其内部映射关系为一条元路径. 换言之,一个网格内部有多个方格,方格间由 *L*(-,|) 分隔(网格内部分隔符可以省略,但为了表述一致性,网格外部分隔符不可省略),因此一张表格 *T*=<(|)*A*(−)*B*(|)*C*(|)*D*(|)−|*E*(−)*F*|*G*|*H*(−)*I*|−*J*>为一个完整的序列,其中()为省略符.

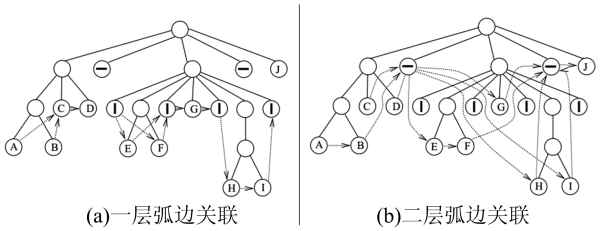

图 4 PartiPath 划分树

Fig. 4 PartiPath partition tree

3.2 元路径操作

PartiPath 划分树使用关系路径辅助空间位置划分实现数据结构的输出,保留了表格内部的语义关系,并最终以关系模式存储. PartiPath 重组操作符能够完成表结构的转换及划分.

在层次划分过程中,表格可以从顶向下,从左至右递归划分,且表格含有四部分基本信息{Row_Label,Col_Label,Stub,Entry}[15]. 其中 EntryΔ 为数值区域,其余为 label 区域.

例 3.2 $\delta(\{A,\Phi\})=\Delta$, $\delta(\{A.A_1,\Phi\})=\Delta$ 作为一元路径,表示为(key,value)存储关系,其划分如图 5 所示. 此外,通常在嵌套表格中,常存在关联子标签的情况.

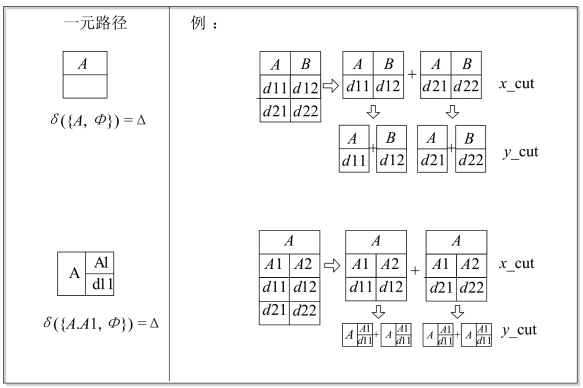

图 5 一元路径划分

Fig. 5 One-tuple path partition example

例 3.3 $\delta(\{A,B\})=\Delta$, $\delta(\{A.A_1,B\})=\Delta$, $\delta(\{A,B,B_1\})=\Delta$ 作为二元路径,其划分如图 6 所示. 通常由同时具有行标题和列标题的表单划分得到.

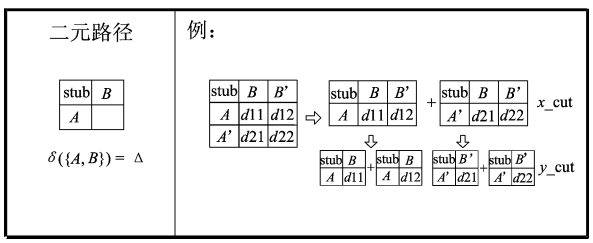

图 6 二元路径划分

Fig. 6 Two-tuple path partition example

由上可知，表格中最小关系单元的呈现形式只有上述三种，我们把最小单元上的映射称为元路径(定义 2.5). 在此基础上执行逆操作能够得到任意结构的数据表.

定理 3.1　无论表单层次结构或者内部结构如何变化，最终都能由元路径逐层构成.

证明　记图 7 表示为一个具有嵌套格式的复杂表格(table)，$T=(C,d)$，其中：

Wang Category[16]标记 =

{(A{(A1,Φ),(A2,Φ)}),(B,{(B1,Φ),(B2,Φ),(B3,Φ)})}

Delta Mappings：

δ({A. A1,B. B1})=d11

δ({A. A1,B. B2})= d12

δ({A. A1,B. B3})= d13…

		B		
		*B*1	*B*2	*B*3
	*A*1	*d*11	*d*12	*d*13
	*A*2	*d*21	*d*22	*d*23

图 7　具有嵌套结构的复杂表格

Fig. 7　Tabular data example with nested structure

Wang category 是一种有效表示表格的数据结构，其涉及的操作主要包括：定位表格区域的四个基本区域(Stub，行标题，列标题及数据单元[15])；并生成标签路径；当所有的 category 标记生成时，将这些路径笛卡尔积连接起来，并且每一个键值都对应一个 delta 数据区域.

层次树的构建方式如图 8 所示，深度优先遍历 index 序号就能得到最终的 PartiPath 结构. 该树共有 8 个内部结点(包括根节点)和 14 个叶子结点对应表格单元，每一个结点对应一条路径映射，表示一条元路径. 孩子结点自上至下或从左至右的方式遍历.

Index	Node	Parent	Children	H × W
1	Outer Frame	None	2,9	4×5
2	Left Side	1	3,4	2×5
3	Stub	2	None	2×2
4	*A*+(*A*1+*A*2)	2	5,6	2×2
5	*A*	4	None	2×1
6	(*A*1+*A*2)	4	7,8	2×1
7	*A*1	6	None	1×1
8	*A*2	6	None	1×1
9	Right Side	1	10,11,15,19	4×3
10	*B*	9	None	1×3
11	*B*1+*B*2+*B*3	9	12,13,14	1×3
12	*B*1	11	None	1×1
13	*B*2	11	None	1×1
14	*B*3	11	None	1×1
15	*d*11+*d*12+*d*13	9	16,17,18	1×3
16	*d*11	15	None	1×1
17	*d*12	15	None	1×1
18	*d*13	15	None	1×1
19	*d*21+*d*22+*d*23	9	20,21,22	1×3
20	*d*21	19	None	1×1
21	*d*22	19	None	1×1
22	*d*23	19	None	1×1

图 8　层次树转换示意图

Fig. 8　Hierarchy tree scheme

3.3　PartiPath 算法

PartiPath 算法主要包含三个阶段：初始化、交替划分(grid partitioning)以及构建树(tree construction).

注：Grid _curr：当前活动区域；

$S=s[i], i=1,\cdots,n$：当前活动区域横向划分线集合；

n：当前区域中横向划分线数量；

s_curr：S 中正要被执行的划分线；

s_end：结束横向划分线；

GridQ：包含 Grids 的队列.

输入：TR = (*D*,*A*,*F*) 中各类表格文档

输出：PartiPath Tree

```
PartiPath ALGORITHM (Table){
1 Grid_curr = Table; //初始化将表格赋给当前活动 Grid
2 Grid_curr. direction = horizontal; //横向切分
3 GridQ = CreateQueue; //初始化队列，存放 Grids
4 GridPartitioning (GridQ, Grid_curr); //划分当前活动 Grid
5 TreeConstruction(GridQ, Grid_curr); //将当前活动区域划分后的 Grids 存放到队列中
}
GridPartitioning (GridQ, Grid _curr){
1 i=1;
2 While (i<n) Do //遍历所有横向划分线
3 {
4 s_curr = s[i]; //划分当前活动 Grid
5 s_end = FindEndingHSL(s_curr); //结束横向划分
6 Grid = DefineGrid (s_curr,s_end); //定义 Grid 实例
7 AddQueue(GridQ, Grid); // Grid 入队
8 i = i+1;
9 }
```

```
}
TreeConstruction (GridQ, Grid _curr){
1 While NotEmptyQueue(GridQ)
2 {
3 Grid = DeQueue(GridQ); //获取 GridQ 中的子 Grid
4 InsertTree(Grid_curr, Grid); //插入树中
5 If Not Cell(Grid) Then //判断最小单元
6 {
7 Grid_curr. direction=NOT(Grid_curr. direction); //改变划分方向
8 GridPartitioning(GridQ, Grid); //重新划分活动区域
9 TreeConstruction (GridQ, Grid); //递归入队
9 }
10 }
}
```

4 Tabular 关系模型

对关系数据库的定义[16],应该包括数据库模式和数据库实例两个方面. 我们以图 1 为例,使用关系模型来逐层存储该表格数据结构(层次划分树).

D_{tabular}=(relname$_D$, R_D, dom$_D$, DOM$_D$) 是一个 Tabular 库.

relname$_D$ = { Tokens, TessionN, Grid, Table }

R_D={Tokens(TOK#, NEXTOK#, Token, BLK#, Level, Parent, Prpperty),

Tession (TSN #, NEXTSN #, GRD #, Mode,Type),

Grid (GRD#, NEXGRD #, TAB#)

Table (TAB #, TITLE, UNIT, Observor, Date, Moniplace, Publisher)}

dom$_D$= {{t_1,t_2,…},{s_1,s_2,…}…}

DOM$_D$(TOK#)={t_1,t_2,…},

DOM$_D$(GRD#)={g_1,g_2,…},

…

语义表示如表 1~4 所示.

表 1 形符表

Tab. 1 Token table

TOK#	NEXTOK#	TOKEN	TSN#	LEVEL	PARENT	PROPERTY
t_{26}	t_{27}	水文	s_4	0	*null*	*label*
t_{27}	t_{28}	水温	s_4	1	t_{26}	*label*
t_{28}	t_{29}	浪高	s_4	2	t_{26}	*label*
t_{29}	t_{30}	涌高	s_4	3	t_{26}	*label*
t_{30}	t_{31}	08 时	s_4	0	*null*	*label*
t_{31}	t_{32}	14 时	s_4	0	*null*	*label*
t_{32}	t_{33}	25.4	s_4	1	t_{30}	*value*
t_{33}	t_{34}	26.8	s_4	1	t_{31}	*value*
t_{34}	t_{35}	2.3	s_4	2	t_{30}	*value*
t_{35}	t_{36}	1.1	s_4	2	t_{31}	*value*
t_{36}	t_{37}	0.5	s_4	3	t_{30}	*value*
t_{37}	t_{38}	0.9	s_4	3	t_{31}	*value*
t_{38}	t_{39}	气象	s_5	0	*null*	*stub*
…	…	…	…	…	…	…

表 2 方格表

Tab. 2 Tessellation table

TSN#	NEXTSN#	GRD#	Mode	Type
s_4	s_5	g_1	Meta-path	*below*
s_5	s_6	g_2	Meta-path	*rightof*
s_6	s_7	g_2	Meta-path	*rightof*

表 3 网格表

Tab. 3 Grid table

GRD#	NEXGRD #	TAB#
g_1	g_2	H9C
g_2	g_3	H8C
g_3	g_4	H8C

表 4 表格文档表

Tab. 4 Tabular table

TAB#	Title	Unit	Observor	Date	Moniplace	Publisher
H9C	滨海旅游度假区监测数据表报表	河北省海洋环境监测中心	苗建波	2009-04-26	秦皇岛亚运村滨海旅游度假区	杨义俊

当建立关系模式时，叶子结点的二层关联 BiCorrelation 可以产生一个表格；其一层关联 Correlation 可以得到一个网格；关联 Tokens 表里的 Level 和 Parent 标签可以得到一条元路径. 其关系模式完整地记录了 PartiPath 的划分步骤，保持其完整性及一致性.

5　基于关系模型的查询问题

传统的文件系统大多使用文本文档来存储和管理数据，主要采取全文检索字符串匹配的方法对文本和结构化信息进行搜索. 由于本文设计的表格模型库提取了不同结构和语义，其上的查询需要定义不同的查询语言. 本文基于用户查询兴趣相似性优化了查询相似性，提高查询可用性及可靠性.

5.1　查询问题描述

D_{tabular}描述的数据库模式记作 D，其上的实例记作 C，对于每个 C，查询 q 的结果是 $q(C)$. 我们想通过查询知道：①q 指定的多个实例是不是出现在 C 中（如列出表格中包括关键词“08 时，水温/浪高”的“方格”，其中水温/浪高表示“水温或者浪高”，“方格”表示为一次水文表查询）；②q 指定的满足部分否定条件的实例（如列出表格中包含形符“1.1”，但其属性不为 value 的“方格”）；

上述部分查询问题分别形式化定义如下：

（Ⅰ）析取匹配查询问题：存在正存在查询满足析取匹配查询 q；

（Ⅱ）负匹配查询问题：存在关系演算查询满足负匹配查询 q；

① $q=\{x_s \mid \exists\ x_{t1},x_{n1},x_{n2},x_{l1},x_{l2},x_{p1},x_{p2},x_{r1},x_{r2}$

$(\text{Tokens}(x_{t1},x_{n1},''08\text{时}'',x_s,x_{l1},x_{p1},x_{r1})$

$\wedge$

$(\text{Tokens}(x_{n1},x_{n2},''\text{水温}'',x_s,x_{l2},x_{p2},x_{r2}) \vee TOKENS(x_{n1},x_{n2},''\text{浪高}'',x_s,x_{l2},x_{p2},x_{r2})))\}$

② $q=\{x_s \mid \exists\ x_t\ x_n,\ x_p,\ x_l, x_c(\text{Tokens}(x_t,x_n,''1.1'',x_s,x_p,x_l,x_c)$

$\rightarrow\text{Tokens}(x_t,x_n,''1.1'',x_s,x_p,x_l,''\text{value}''))\}$

5.2　查询相似性度量及优化

首先通过分析历史查询 queries，将其划分成若干个相似的主题查询区域；然后从对应的主题中选择经常查询频繁访问数据的语句（频繁查询）. 这些被选出的频繁查询具有明确的主题，只有少部分查询缺乏方向界定，如在海洋灾害中，一次典型的水文信息查询包括{水温、水深、盐度}，通常不会与{能见度}关联.

本文采用最近邻聚集技术[17]实现查询主题分类，使用 DICE 因素[18]作为查询任务间的相似性的度量标准；此外，本文提出以用户查询兴趣相似性（user interest similarity）作为辅助指标，增强可信度.

定义 5.1　DICE 相似度[18]. 给定 Q_i、Q_j 为两个查询任务，其 DICE 相似度为 $\text{Sim}(Q_i,Q_j)=\dfrac{2\mid R(Q_i)\cap R(Q_j,))\mid}{\mid R(Q_i)\mid+\mid R(Q_j)\mid}$. 其中，$R(Q_i)$、$R(Q_j)$ 为 Q_i、Q_j 与之相关的关系，Q_i、Q_j 由多个关系路径组成.

定义 5.2　用户查询兴趣相似性（user interest similarity）. 主要考虑两个用户同时频繁使用关系（元）路径 p_x 的兴趣指数，其权重值 $\text{score}_{p_x}(p_x \mid (uc_i,p_x)\in \varepsilon c\}\cap p_x \mid (us_j,p_x)\in \varepsilon s\}\mid=\mid \Gamma p(uc_i)\cap \Gamma p(us_j)\mid$. 其中 $\mid P\mid$ 为 P 的基数，p_x 为Q_i、Q_j 中的一个关系（元路径），uc_i 为与 Q_i 相关的用户，εc 为 uc_i 与 P 的关联边集，如图 9 所示.

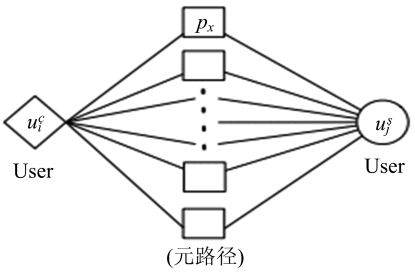

图 9　对象兴趣关系图

Fig. 9　Common interest

定义 5.3　多级相似度指标，$\text{Mutli\_Sim}(Q_i,Q_j)=\text{avg}(\text{Sim}(Q_i,Q_j),\sum_{x=1}^{P}\text{score}_{p_x}(u_i,u_j))$. 其中 P 为 Q_i，Q_j 中所有不同关系组成的集合.

例如，给定若干基本组合查询 queries，如表 5 所示，其中每一个查询都包含多个关系（元路径）.

在上述历史查询上分别使用 Subject Area Identification 算法和 Frequent Query Selection 操作[17]，可获得频繁关系集为 {WaterTemp, WaterDepth}，{Location, RegionName}. 通过不断的样本训练，最终可以构建对应的物化视图以满足后续查询需求，最大程度上满足用户查询习惯.

表5 历史queries对应关系表

Tab. 5 Corresponding relation between historical queries and its number

Q_1	WateTemp, WaterDepth, Salinity	Q_9	WateTemp, WindDirection, WindSpeed
Q_2	Location, RegionName, Observer	Q_{10}	Location, RegionName, Station
Q_3	Location, RegionName, Station	Q_{11}	WaterDepth, Module, WindSpeed
Q_4	Location, RegionName, Proofreader	Q_{12}	Station, Observer, Proofreader
Q_5	WateTemp, WateDirection, Salinity	Q_{13}	Station, Observer, Proofreader
Q_6	Location, Observer, Proofreader	Q_{14}	WateTemp, Location, DO
Q_7	Location, RegionName, Observer	Q_{15}	Station, Observer, Proofreader
Q_8	WaterDepth, WindDirection, AirTemp	Q_{16}	WateTemp, WindDirection, WaterDepth

6 实验分析

本文通过调研得知,某海洋信息中心自2000年到2012年各生态系统生成及记录的报表数据量以及数据存储情况如下表6所示.

表6 信息中心数据应用明细表

Tab. 6 Data application schedule of information center

编号	应用类型	数据类型	数据量(MB)
1	风暴潮预测	海洋水文、气象文本	250
2	决策支持	海上应急方案	100
3	灾害反演	海洋地形、水文	80
4	数值预报	各类海洋数据	320
5	海浪预测	海洋水文	600
6	海表监测	海洋环境	780

文中采用水温、气象文本数据作为数据源来验证算法的可行性及有效性.经处理的Excel表格共计63类,总计3 562张数据表,其中海水浴场监测表共计657张.

对上述文本表单进行关系实例化操作,根据上述策略获取关系模式并存入的元组数量总计21万余条.与传统以文档类型定义方法为标准,其转换渗透率增加了58.31%.相同条件下,若应用CML DTD抓取数据模式,所需要连接SQL次数实验如表7所示.

从表7可以发现,CML在关系实例化后,其查询性能成稳定线性变化,如图10所示,即以该方式描述文本后并没有导致数据模型上处理性能降低.虽然关系数据库包含的高效算法及数据结构提高了查询效率,但仍不能满足访问需求;由于本文方法在层次划分时设计了层之间的一级与二级关联,很大程度上降低了SQL连接访问数,因此在查询效率上具有相对的优势.值得注意的是,由于本文基于Phython语言,定制化描述了表格数据结构,往往在性能条件与因素设置上都存在一定的偏差.

表7 长度*N*的查询路径的SQL平均连接数

Tab. 7 Average SQL link for *N*-lengths query path

查询路径长度	SQL平均连接数(CML)	SQL平均连接数(本文方法)
1	1	1
2	1	1
3	2	1
4	2.25	1.6
5	3.25	1.76
6	4	1.92
7	4.67	2.1

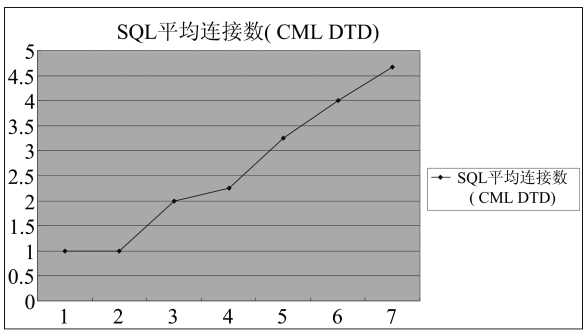

图10 CMLN路径平均SQL连接数示意图

Fig. 10 Average SQL link illustration with CMLN path method

为了验证表单文本关系实例化转换的有效性,本文用抽取比、查全率以及准确度三个指标衡量算法的正确性.分别抽取10%的数据作为数据本体,对比分析基于自然语言处理(NLP),ontology,Wrapper Induction(WI)以及本文基于表格规范规则和语法树的四种信息抽取方法;构建如下实验环境:Athlon 2。8GHz CPU, 16G memory, Windows7和C# 编程语言.

准确度是指在所有的被抽取出的数据对象中描述正确的数据对象所占频次;平均抽取比是实际抽

取数据信息数量与所有数据信息抽取次数的比值；查全率是指在所有抽出并描述正确的数据对象中实际被抽出的正确数据信息所占比率，其指标的取值范围皆为{0,1}.

实验结果如表 8 所示. 通过分析实验数据，其实际存在的关系 3 000 个，横向对比各方案，本文提出的表格信息抽取算法实际抽取数据对象 2 876 个，抽取比高达 0.96，实现了相对较优的数据抽取效果，且其准确度和查全率比基于 NCP 和 Ontology 的方法转换率更高、效果更优. 主要原因是本文在 Hurst 等[8]的研究基础上，利用层次划分树提取出表格内部的附加信息，使其具有准确的数据结构. 此外，本文无损建立起数据信息之间的关系，为半结构化数据以及非结构化数据的关系化奠定了基础.

表 8　实验结果对比

Tab. 8　Experiment results

方法	关系（3000）		准确度	查全率
	抽取数	正确数		
NCP	2680	2144	0.8	0.7
Ontology	2750	2255	0.82	0.73
WI	2769	2381	0.86	0.82
本文算法	2876	2675	0.93	0.91

7　结论

研究表格模型和其上的查询问题是目前非结构化及半结构化数据应用的基础，本文提出了一种适合用于表单文本数据的数据模型 D_{Tabular}，该模型结构化以及关系化表格数据. 此外，本文定义了表格模型上的基础查询问题，并基于用户兴趣改进了传统的查询相似度计算，为进一步研究其查询方法奠定了基础.

参考文献(References)

[1] Mayer-Schönberger V, Cukier K. Big Data: A Revolution That Will Transform How We Live, Work, and Think [M]. Boston: Houghton Mifflin Harcourt, 2013.

[2] China Argo News Letter, 2014, NO. 2.

[3] Argo data center in China, http://www.argo.org.cn/

[4] Chui M, Brown B, Bughin J, et al. Big data: The next frontier for innovation, competition, and productivity [R]. McKinsey Global Institute, 2011.

[5] Hurst M. Layout and language: Challenges for table understanding on the web[EB/OL]. http://cgi.csc.liv.ac.uk/~wda2001/Papers/12_hurst_wda2001.pdf.

[6] Embley D W, Tao C, Liddle S W. Automating the extraction of data from HTML tables with unknown structure[J]. Data & Knowledge Engineering, 2005, 54(1): 3-28.

[7] Douglas S, Hurst M, Quinn D, et al. Using natural language processing for identifying and interpreting tables in plain text[J]. Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval, 1997, 21(2-4): 231-243.

[8] Gatterbauer W, Bohunsky P, Herzog M, et al. Towards domain-independent information extraction from web tables [C]// Proceedings of the16th International Conference on World Wide Web. Banff, Canada: ACM Press, 2007: 71-80.

[9] Ferrucci D, Lally A. UIMA: An architectural approach to unstructured information processing in the corporate research environment [J]. Natural Language Engineering, 2004, 10(3-4): 327-348.

[10] Pivk A, Sure Y, Cimiano P, et al. Transforming arbitrary tables into logical form with tartar[J]. Data & Knowledge Engineering, 2007, 60(3): 567-595.

[11] Pinto D, McCallum A, Wei X, et al. Table extraction using conditional random fields[C]// Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval. Tprpnto, Canada: ACM Press, 2003: 235-242.

[12] Duygulu P, Atalay V. A hierarchical representation of form documents for identification and retrieval[J]. International Journal on Document Analysis and Recognition, 1995, 5(1): 17-27.

[13] Tijerino Y A, Embley D W, Lonsdale D W, et al. Towards ontology generation from tables[J]. World Wide Web: Internet and Web Information Systems, 2005, 8(3): 261-285.

[14] Shigarov A O. Table understanding using a rule engine [J]. Expert Systems with Applications, 2015, 42(2): 929-937.

[15] Lopresti D, Nagy G. A tabular survey of automated table processing[C]// Graphics Recognition Recent Advances. Springer, 2000: 93-120.

[16] Wang X. Tabular abstraction, editing, and formatting [D]. University of Waterloo, Canada, 1996.

[17] Kumar T V V, Goel A, Jain N. Mining information for constructing materialised views [J]. International Journal of Information and Communication Technology, 2010, 2(4): 386-405.

[18] Frakes W B, Baeza-Yates R. Information Retrieval: Data Structure and Algorithms[M]. Upper Saddle River, USA: Prentice-Hall, 1992.