Method of Understanding Structure and Building Database with Material Experiment Data

Cao Min, Ma Zhiyuan

***Abstract*—With the development of information technology, the online database is getting popular. The experimental data on material has problem to transform because of its diverse structures, redundancy and isolation. This paper aims to analysis the structure of experimental tables in an automatic way by designing a prototype algorithm which can discover the relationships among the headers and to generate a tree structure with data labels to describe these tables, so they can be imported into database finally. Experiment results in this paper show that although the inserting process is slower, the redundancy is reduced and the speed of query is more than 3 times faster. The efficiency of the material database, as one type of data warehouses, is increased overall.**

***Index Terms*—Table understanding, Tree structure, Table structure, Relational database, material database**

I. Introduction

With the development of modern information technology and modern experimental methods, the Material databases, especially online material databases, attract more and more attention in order to access to the latest developments in materials accurately. The online material database gets an earlier start in foreign countries. For instance, MatWeb [9] (http://www.matweb.com) in America is committed to found an online searchable database of material properties and has over 115,000 materials. The National Institute for Material Science in Japan set up a material database called MatNavi, which is consists of more than 11 material databases and provides cross-database search services [10]. China is trying to catch up and find out the research and applications of online material database. The National Science and Technology infrastructure platform, also called Materials Scientific Data Sharing Network project, has been started in 2011 [11]. In order to build the platform, reliable data sources, especially the existing ones, are useful. Generally, there are three types of the existing data sources.

1) The papers on material field, where the data is displayed in the tables. But tables are much like Web-Table [1], instead of formatted tables in computer field.
2) The experimental data sets which come from colleges and enterprise laboratories. Their formats are not unified and the contents are incomplete. The formats could be plain text or form files such as Excel, Txt, Csv, which may be artificial or generated by experimental instruments.
3) Formatted tables, such as databases and data on reference books in material field available. The format and content are uniform and complete.

Among these three types of data, which this paper has come into contact with, the experimental data sets are more popular. The table format is wildly used in material experimental data and its structure has the following characteristics:

1) The structure does not conform to 1NF of database. In order to express the hierarchy of data and reduce duplication, there are merged columns or cells, which destroy the atomicity of the data structure
2) The BoxHeads which stand at the top of the tables contain more than 2 rows and hierarchical relationships are described among the rows, which make the table structure complicated. So it's hard to use them as column names directly.
3) Duplicated column names and uncertain repeat times. For example, an experiment has many conditions but the number of conditions of each experiment is different, so it needs many more columns to describe conditions while many of the cells are empty. The circulation discipline can be easily found among columns. The uncertain and duplicated number of columns would seriously reduce the speed and convenience of query and execution.
4) The same experiment data object may contain more than one row, which is also harmful to database.
5) Hyperlink, external picture and other binary data which are hard to store in relational database may appear among files.

To generate a database from experimental data, table structure analysis and interpretation is necessary [18][19]. Wang et al. have put forward a description of table structure, which is a well-accepted definition [1][2][4][5][6][7]. So this paper will follow the terminology of table structure described by Wang et al. The paper [16][17] etc. discussed table characters such as HTML tags and table layouts, could be used to analyze tables. Rastan R. et al. presented a solution to parsing the stub area of a table [7]. They provided a classification of layout features in table stubs and an end-to-end process system called TEXUS with PDF format. Seth S. et al. focused their system on BoxHead area [6]. With a context-free grammar, a parse tree for the column-header is

Manuscript received December 14, 2016; revised January 9, 2017.This work is supported by the Shanghai Municipal Science and Technology Commission (Grant No. 15DZ2260300)

Cao Min (Email: mcao@staff.shu.edu.cn) and Ma Zhiyuan (Email: mazhiyuam@shu.edu.cn) are with the School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China

ISBN: 978-988-14047-3-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

created to describe the table structure. This paper tries to use the understanding on table structure and cell to reduce the difficulty of transforming the material data table into a relational database.

The main contributions of this paper is as follows. 1) We provide a tree structure using rule-based approach, which structure is used to store material table column-header. Furthermore, we combine the data sets with the tree. We solve the duplicated column problem. 2) Aimed at the tree structure, a solution to building tables in database and transform data into tables is discussed. Though the insert execution is slower, the query speed is about 3 times faster than original storage method.

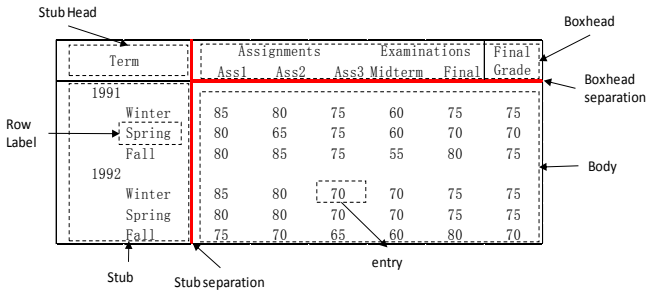
In the next section, we review the table structure definition by Wang et al. and the algorithms from other researchers. Section 3 offers the tree structure as the middleware. Section 4 discusses the algorithm to convert complex material table header to tree structure. Section 5 describes an experiment and discusses the performance of our algorithm. Conclusion is presented in section 6.

II. Related Works

According to Hust et al. the processing of table understanding could be divided into following steps [21]: functional analysis, structural analysis, interpretation relationships. In consideration of the situation in material experiments, we focus on table structural analysis and interpretation relationships.

Wang et al. have put forward a well-known description of table structure in paper [3]. Xin Xin Wang divided tables into 4 parts. The Body contains the data sets. BoxHead is usually at the top of a table which is the index of a column. Stub, the index of a row, lays on the left part of a table. The BoxHead and stub can be called as headers. The overlap among BoxHead and stub is called stub head which lays at the top-left of a table. The division lines are called stub separation and BoxHead separation. They may not be physical, but if the separations are recognizable, functional structure could be analyzed [4]. The structure definition of Wang's table is shown in Table I.

Table I The structure definition of Wang's table



Wang used a grammar-based and text-based tree to describe the table structure, each relationship will be generated into a tree. For example, as shown in Table I, the result can be following:

(Year, {(1991,∅), (1992, ∅)}),

(Term, {(Winter, ∅), (Spring, ∅), (Fall, ∅)}),

(Mark, {(Assignments, {(Ass1, ∅), (Ass2, ∅), (Ass3, ∅)})

The result is composed of three trees, which describe the relationships among the labels in stub and boxhead. However, they didn't pinpoint the relationships among the trees, which introduced some problems while parsing table and building database. The papers [2][5][6] tried to understand the table from the angle of BoxHead. Seth S et al. used a rule-based context-free grammar to get Wang's tree with the edited table headers [6]. But it has only 8 grammar rules, which hardly fits all types of table structures. In addition, it cannot analyze without transforming the column headers manually because the system cannot analyze all types of functional table structures, such as a table who has multiple head roots.

Rastan R et al. combined Wang's table definitions, and designed an end-to-end table processing system (TEXUS) with two phases: table extraction and table understanding [7]. They parsed tables by identifying common patterns of layout features that appear in the stub part. The system can be used in different situations, such as merged cells, blanked columns or rows and relationships found in table format etc.

Alexey O. proposed a table model with two levels: physical and logical [20]. By using regular expression, rules engine and dictionaries, they designed a solution for unstructured tabular data integration. The solution divides the result database into two parts, the one is used to locate or describe the relationship of a table cell and the other contains the data of a cell. While it surely achieves the goal to get a structured information, it suffers the data redundancy problem. The number of columns which do not store data is changeable due to the table structure and the integration algorithm. Besides the information cannot express directly in such database. The cells are discrete and the database has less readability and is suitable for query.

The document media is changing in different time. In earlier time, the media is usually paper, so Optical Character Recognition (OCR) is the main technique. With the development of e-document, PDF becomes the main storage file type. The main application target of paper [7] is PDF files. The age of internet makes the textual information, used on web, popular. The target is changed on unstructured information extraction [15], such as HTML, XML and JSON etc. Liubo Ouyang et al. used DOM tree with geometry model of tables in HTML files to locate the Preparative Core Area and then the Core Area by calculating weighted expectation value [15]. They recognized the similarity of characters between neighbor cells to judge and correct the data content.

This paper uses simple rules to parse the header of a table and create a tree representing the parent-child relationships among labels, then insert data sets after leaves to generate the database instance. Meanwhile readability should be kept mostly.

III. Proposed Structures

A. The class of the table for processing and input format

Since paper [3][6] has already discussed about the definition of a table, so it won't be covered here. In order to simplify the model, the following assumptions are made to describe the table we discuss:

- Only two-dimensional tables are discussed. The cells in the table could be located by row and column number. Each location can be mapped to no more than one cell.

ISBN: 978-988-14047-3-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

- The neighbor cells can be merged. We assume that merged cells also have rectangle shape. The location should pick the minimum value among the cells.
- For convenient operation, cells can only contain textual data but not pictures or formula etc. unless they are turned into text.
- The structure information is not considered in this paper. We ignore the influence of the cell formats.

Meanwhile the relationships among the cells are restricted by the following rules:

- The cells can at least be divided into two exclusive classes, entry and label.
- For a convenient parsing, we assume that the label area is independent. If not, we can move the body area to meet the condition.
- The headers in the label area should be able to turn into the tree structure we described. We will continue to discuss this in the next section.
- The cells in label area should be compact, which means no blank cells. If there exists a blank cell, it will be merged with a filled cell nearby.
- The cells in the label can be repeated, but can only repeat among the whole level of the label. If not, we can make a new level to meet the rule.

The main object of this paper is the material data in the table, so the notes like footnotes, marginalia and sources are not considered in this paper.

Most of our experimental data sets are Excel and textual data such as CSV while PDF is rare. Excel is not usually used for data analysis and I/O operation, because Excel has rich table styles, formula, uncertain table location and binary data such as pictures. CSV (Comma-Separated Values) files can display tabular data in a textual way. The data records are separated by commas or other separator. It's a simple file format that programs can easily parse. But we should consider the merged cells situation. We define that the merged cells should be displayed into one text cell and blank cells. The hypothesis above show that there should be no blank cells in table. So we can avoid the ambiguity and make sure when blank cells appear, there must be a merged cell. For example, the table in Table I can be converted into the following CSV format:

Title1, ,Title2
Title3,Title4,

We can find the blank cell caused by merged cells. The cells of first row, first column and first row, second column are merged. We can find that the "Title 3" and "Title 4" are the sub-titles of "Title 1." We can take advantage of the characteristic in the stage for header analyzing.

B. Tree model

We use tree model to describe the header and its relationships, as shown in Fig. 2. From the top to bottom of the header, the system parses and builds a tree structure. To union the leaves, a virtual root is created. In our designing phase, the stub and stub head area are not concerned. In other words, the headers in BoxHead will be parsed into tree leaves. This plan is feasible. At worst, the stub cells are perceived as body cells and they will repeat in the data results. One possible way is to judge the stub area and build another table with foreign keys [7]. Another method is to make the same with the stub cells and table columns are used to mark positions [20]. But the result is hard to read and redundant. So the first method can be following in future work.

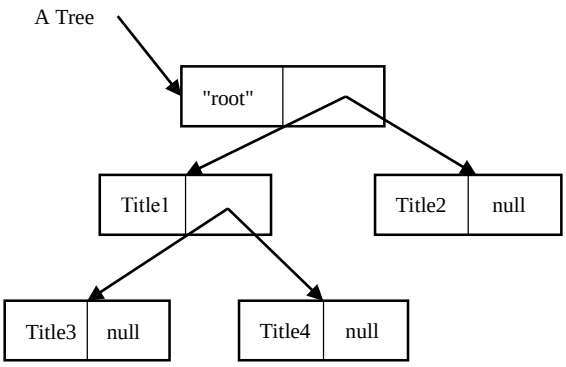


Fig. 1. Map about tree structure

However, not all of tables can be directly transformed, which would change to forests or graphs. The reason is that the relationships of table headers are not consistent. In Table II (a), there is a merged cell under the single cell, which will break the affiliation from the top to bottom. So we make suppose in Section 3.1. It's necessary to change structures into the forms expected by our algorithm, manually or automatically. But the transformation is non-unique. Both Table II (b) and Table II (c) are possible results. Thus how to obtain optimal solutions of transformation is another subject of our further work.

Table II different expressions of the header

year	midterm	final	midterm	final
	male		female	
2014	89	87	84	91

(a) Original header

year	male		female	
	midterm	final	midterm	final
2014	89	87	84	91

(b) One of transformation header

year	sex	midterm	final
2014	male	89	87
	female	84	91

(c) Another transformation header

IV. THE ALGORITHM FOR CREATING TREE AND BUILDING DATABASE

The algorithm of building tree will be proposed in this section. The main ideas of the algorithm are as followings:

1) Check whether the table meets the conditions. If not, transform it manually.
2) Analyze and pick up the BoxHead area, at least one row.
3) Analyze the cells and create the tree structure.
4) Get another part of table.
5) Combine the tree structure with the data sets. Prepare the preliminary data to insert into the database.
6) Build table in database with foreign key constraints and insert data.

This Section will introduce the algorithm for creating tree and building tables in database. In this paper, the program language is Java with map class used to simulation the tree structure. In order to get a union root, we name the root of

ISBN: 978-988-14047-3-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

tree as 'root'.

A. The algorithm of the headers

The purpose of this algorithm is to create a tree describing the relationships among the hear cells. The process is recursively implemented, which is finding the child tree and building it then inserting it into the parent tree. Finally insert the result into the virtual root.

Table III Algorithm: AddTree

Algorithm 1. AddTree (row, col, length)

```
if (getCSVList(row, col ) = NULL) then
   Return NULL;// ERROR or end of header
end if
TempRoot = CreateNode();
rownum = getRowSize(); colnum = getColSize();
for (i=1; i≤ col+ length; i=i+1) do
   title = getCSVList(row, col );
   range=i; i=i+1;
   if ( i != ( col + length )) then
      while (i != ( col + length ) and getCSVList(row, i )
= null ) do
         i=i+1;
      end while
   end if
   NewLength = i – range;
   i=i-1;
   TempRoot.put ( title, AddTree( row+1, range,
NextLength ) );
end for
Return TempRoot;
```

Table IV The example of headers of material data (part)

Band name	Material name	Chemical composition NO.	Material information NO.	Performance information						Experimental condition							
				Class of performance	Performance name	Value	Unit	Testing equipment	Testing institution	Condition name	Parameter name	Value	Unit	Condition name	Parameter name	Value	Unit

According to the experiment data we have, the columns could be repeated and we can find multiple same leaves, as shown in Table IV. This paper use map class to store the tree structure, which overcomes the problem of duplication leaves. To keep order of data, we assign a number in the data leaves. We also enforce rules as following:

1) $\forall l \in VirtualRoot, l \in \{leaves\} \nRightarrow l$ repeats
2) If a leave l repeats, its loop set is ls ,then $\forall\ child \in l \Rightarrow child \in ls$
3) If 2) doesn't satisfy. Create a leave $newleave$, who loop set is ls, then
   $newleave \in l,\ \forall\ child \in newleave \Rightarrow child \in ls$

With the algorithm Addtree, we can create the following Map with Java Language, whose logical graph shown in Fig. 2. The map class can be easily transformed into XML and Json files.

{ "Virtual root" = {"Band name" = null, "Material name" = null, "Chemical composition NO." = null, "Material information NO." = null, "Performance information" = {"Class of performance" = null, "Performance name" = null, "value" = null, "unit" = null, "Testing equipment" = null, "Testing institution" = null}, "Experimental condition" = {"Condition name" = null, "Parameter name" = null, "Value" = null, "Unit" = null}}}

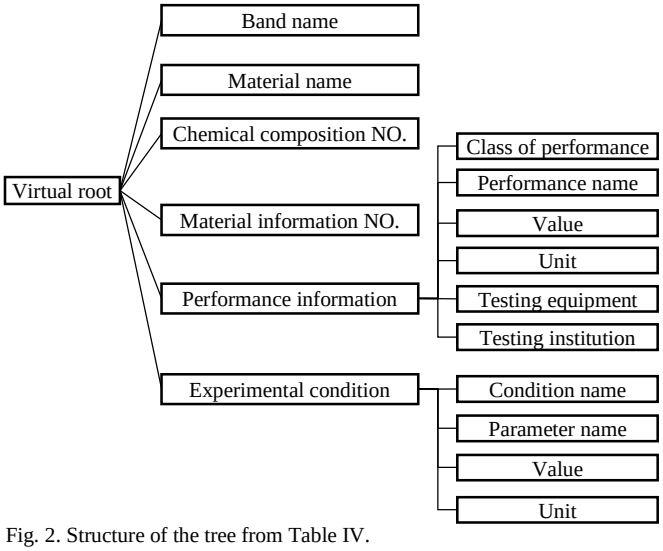


Fig. 2. Structure of the tree from Table IV.

B. Data sets combining with structure tree

After building the tree of the BoxHead part, we insert the data sets as leaves into the structure tree to prepare the database execution. This paper is trying to get a mix tree with the data and header one by one. We should pay attention to the duplicated columns. During the process of copying instance of map class, the system is doing shallow copy by default. So an overloaded algorithm of deep copy is necessary. The algorithm is as following:

Table V Algorithm: Clone

Algorithm 2. Clone (Map root)

```
res= CreateNode();
for (i=0; i<root.Size(); i=i+1) do
   res_key = root.getKeyByIndex(i);
   root_value = root.getValueByIndex(i);
   res_value= clone( root_value );
   res.put ( res_key, res_value );
end for
Return res;
```

The algorithm that locates the position of the header in the tree and inserts the data sets, is the same as algorithm AddTree. So we don't explore it. Fig. 3 shows a simple example after inserting data sets. The Data1 and Data2 belong to the header Title2 and the Title2 in the structure tree is a leaf node who has no children. A number should be recorded to mark their order.

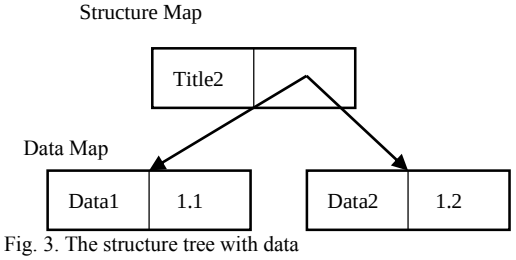


Fig. 3. The structure tree with data

ISBN: 978-988-14047-3-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Table VI Algorithm: BuildDataTree

Algorithm 3. BuildDataTree (Set data[])

```
res= clone (VirtualRoot);
for( i =0 ; i< data.Size(); i=i+1) do
   for(j=0; j<data[i].Size(); j=j+1) do
      node = res.LocateLeaf(data[i][j]);
      node.insert (number,data[i][j]);
   end for
end for
Return res
```

C. Building database

Before inserting data, we should build tables in databases while we are preserving relationship information as much as possible. So we think the every child nodes should be a new table while an ID column in table created by its parent nodes should be the foreign keys of the table. The foreign keys can reduce the redundancy and maintain the integrity.

V. Experiment Results

The experiment configuration is as followings:
Oracle Java(TM) 8
MariaDB 5.5.36, for x86

Before the experiment, we convert all the Excel to the CSV format files, and change the structure which may cause the problem in Table II (a). The BoxHead area is ensured to be compact which has no blank cells.

A. Structure of results

Table VII result of Example in Table IV -- Table Virtual Root

id	Band name	Material name	Chemical composition NO.	Material information NO.
1	NO80310009	La0.4Sr0.4TiO3	1	1
2	NO80310010	TiO2	2	2
3	16MnCr5	pinion steel	3	3
4	20CrMo	alloy steel	4	4

The result of the Table IV is shown in Table VII, Table VIII and Table X. The headers are divided into three parts of nodes and their relationships. Tree relational tables are generated based on the nodes. The main table is named "Virtual Root". The foreign keys named "fk" from other two tables point to the "id" in table "Virtual Root ".

Table VIII result of Example in Table IV -- Table Performance Information

id	Class of performance	Performance name	Value	Unit	Testing equipment	Testing institution	fk
1	mechanical property	granularity	20	μm	temperature testing instrument DATAPAQ	Experimental Center in University of Science and Technology Beijing	1
4	chemical property	Coefficient of thermal expansion	3×106				2
5	performance	HV(49N) hardness	286		Vickers hardness tester HV-10	Northeastern University	3
8	performance	Microstructure			Microscope	Northeastern University	3
9	performance	high-temperature mechanical properties			Gleeble-1500D Thermal simulator	Chongqing University	4

The result shows that the relationships between headers and cells are stored among the tables. The tables are more readable than the one by Shigarov et al [20]. The result also shows blanked fields in a few records because of missing experiment data.

This paper processed 10000 original data sets. 2620 of them repeat partially and 120024 cells are blanked or meaningless, accounting for 27.9%. After processing, 27381 of them remain, which reduce by 77.2%. The algorithm saves the database storage cost and advance the semantic relations among cells.

Table IX Algorithm: Data_To_SQL

Algorithm 4. Data_To_SQL (Map data_tree, foreign_id, table_name, forerign_table)

```
for ( i = 0; i< data_tree.Size(); i=i+1 ) do
   data_record= data_tree.getIndex( i );
   if (IsExist (data_record, table_name) = False ) then
      InsertData (table_name, data_record, foreign_id);
   end if
   child_foreign_id = QueryID
(table_name,data_record);
   if (data_tree.HasChild() = True ) then
      while (data_tree.HasNextChild() != False) do
         child_data=data_tree.getNextChild();
         Data_To_SQL (child_data, child_foreign_id,
child_table_name, table_name);
      end while
   end if
end for
```

Table X result of Example in Table IV -- Table Experimental Condition

id	Condition name	Parameter name	Value	Unit	fk
1	Heat medium oil	Calcination temperature	600	℃	1
2	Acetone toluene	Calcination temperature	650	℃	1
5	Solid phase method	Test temperature	1100	℃	2
6	Solid phase method	pressure	1100	Mpa	2
7	Austenitization	temperature	1200	℃	3
8	Compression deformation	deformation	0	%	3
9	Test temperature insulation	Time	5	min	4
10	Actual billet continuous casting strain rate	Strain rate	2×10-2	/	4

B. Expenses on execution and query

This paper compares the time consumption between our algorithm and the original algorithm to generate one table with whole data cells directly on executions and queries. The material database is a kind of data warehouses. Most operations on data are queries [22]. So we don't test the deleting and modifying operations. The result is shown in Table XI.

Table XI the consuming comparison between our and original algorithm

Unit: second	Data size	Our algorithm takes	Original algorithm takes	Ratio
Insert executions	1000 executions	71.611	6.732	0.094
Query by keyword full fields output	1000 queries	6.921	27.899	4.031
Query by keyword part fields output	1000 nested queries	3.778	11.798	3.122

ISBN: 978-988-14047-3-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

We find that the speed of inserting executions of our algorithm is 10 times slower than the original one. The main reason is that when inserting data into child tables, we need to know the foreign key "fk" which is the "id" field in their parent table. So for one inserting execution in our algorithm, we need two queries and one execution if we don't find the same record, and one query if we find the same. So the insert speed is much slower.

Our algorithm is more than 3 times faster than the original algorithm. It shows that redundancy influences the query speed. When the data sets are divided into tables with smaller scale, higher paradigm and lower redundancy, the query efficiency improves greatly. As one kind of data warehouse, the material databases are more used in query than in inserting executions [22]. Therefore this paper argues that our algorithm can reduce the overall expense.

VI. Conclusion and Future Work

This paper proposes an end-to-end table processing algorithm and its prototype system. Our goal is to standardize the complex table in the material data resources and finally build a relational database. The experiment shows that it has a faster query speed, lower redundancy and keep the relationships among the cells while the table is a little more readable. Therefor experts can analyze data by using the material databases.

In addition, the drawbacks of the algorithm are as following:

- We make too many limits and manual corrections to make sure the output structure is a tree instead of a forest.
- The multi-to-multi relationship in E-R model can't be achieved.
- We only consider the Boxhead area but not Stub area.
- The repeat judgment is too easy.

A further research is required as following:

- To support every kinds of tables. Try to transform tables into the discernible ones.
- Improve the semantic analysis in order to discover the hidden relationship between the tables.
- Make the tree structure on the Stub area to lower redundancy.
- Try to discover and show the multi-to-multi relationships.
- Support more file formats such as PDF, Excel and HTML etc.

References

[1] e Silva, Ana Costa, Alípio M. Jorge, and Luís Torgo. "Design of an end-to-end method to extract information from tables." International Journal of Document Analysis and Recognition (IJDAR) 8.2-3 (2006): 144-171.

[2] Nagy, George. "Learning the characteristics of critical cells from web tables." Pattern Recognition (ICPR), 2012 21st International Conference on. IEEE, 2012.

[3] Wang, Xinxin, and Derick Wood. Tabular abstraction, editing, and formatting. Canada: University of Waterloo, 1996.

[4] Jha, P., and G. Nagy. "Wang Notation Tool: Layout independent representation of tables." Pattern Recognition, 2008. ICPR 2008. 19th International Conference on IEEE, 2008:1 - 4.

[5] Alrayes, Norah, and Wo-Shun Luk. "Automatic transformation of multi-dimensional web tables into data cubes." International Conference on Data Warehousing and Knowledge Discovery. Springer Berlin Heidelberg, 2012.

[6] Sharad Seth, Ramana Jandhyala, Mukkai Krishnamoorthy, and George Nagy. "Analysis and taxonomy of column header categories for web tables." Proceedings of the 9th IAPR International Workshop on Document Analysis Systems. ACM, 2010.

[7] Roya Rastan, Hye-young Paik, John Shepherd, Armin Haller. "Automated Table Understanding Using Stub Patterns." International Conference on Database Systems for Advanced Applications. Springer International Publishing, 2016.

[8] Gao, Zhiyu, and G. Liu. "Recent Progress of Web-enable Material Database and a Case Study of NIMS and MatWeb." Cailiao Gongcheng/journal of Materials Engineering volume 3.11(2013):89-96.

[9] Ramalhete, P. S., A. M. R. Senos, and C. Aguiar. "Digital tools for material selection in product design." Materials & Design 31.5 (2010): 2275-2287.

[10] Yamazaki, Masayoshi. "Development and challenge of materials database-a case study of NIMS materials database." Proceedings of the 2nd Asian Materials Database Symposium. Shanya: University of Science and Technology Beijing (USTB), 2010.

[11] Ministry of Science and Technology of the People's Republic of China. "2011 China science and technology development report." Scientific and Technical Documentation Press, 2011.

[12] Jianhua Zhao, Rudolf Plagge, Nuno MM Ramos, M Lurdes Simões, John Grunewald. "Application of clustering technique for definition of generic objects in a material database." Journal of Building Physics 39(2015).

[13] Jianhua Zhao , Rudolf Plagge, Nuno M.M. Ramos, M. Lurdes Simões and John Grunewald. "Concept for development of stochastic databases for building performance simulation-A material database pilot project." Building and Environment 84 (2015): 189-203.

[14] Antoine Doucet, Gabriella Kazai, Bodin Dresevic, Aleksandar Uzelac, Bogdan Radakovic, Nikola Todic. "Setting up a competition framework for the evaluation of structure extraction from ocr-ed books." International Journal on Document Analysis and Recognition (IJDAR) 14.1 (2011): 45-52.

[15] Ouyang, Liubo, Rui Dong, and Beiji Zou. "Information Extraction Based on Table Area Locating for E-Commerce Websites." 2009 WRI Global Congress on Intelligent Systems. Vol. 4. IEEE, 2009.

[16] Nagy, George, Sharad Seth, and David W. Embley. "End-to-end conversion of HTML tables for populating a relational database." Document Analysis Systems (DAS), 2014 11th IAPR International Workshop on. IEEE, 2014.

[17] Nagy, George, and Mangesh Tamhankar. "Vericlick: an efficient tool for table format verification." IS&T/SPIE Electronic Imaging. International Society for Optics and Photonics, 2012.

[18] Shigarov, Alexey. "Rule-Based Table Analysis and Interpretation." International Conference on Information and Software Technologies. Springer International Publishing, 2015.

[19] Lee, Dongchan, Jaemin Choi, and Sangjin Lee. "Database forensic investigation based on table relationship analysis techniques." 2009 2nd International Conference on Computer Science and Its Applications, CSA 2009. 2009.

[20] Shigarov, Alexey O. "Table understanding using a rule engine." Expert Systems with Applications 42.2 (2015): 929-937.

[21] Hurst, Matthew. "Layout and language: Challenges for table understanding on the web." Proceedings of the International Workshop on Web Document Analysis. 2001.

[22] Inmon, William H. Building the Data Warehouse. John Wiley & Sons, Inc. 1996.

ISBN: 978-988-14047-3-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)