# Conjoint Data Mining of Structured and Semi-structured Data

Qi H. Pan, Fedja Hadzic, Tharam S. Dillon

*Digital Ecosystems and Business Intelligence Institute, Curtin University of Technology, Perth, Australia*

{helen.pan, f.hadzic, t.dillon}@cbs.curtin.edu.au

***Abstract*— With the knowledge management requirement growing, enterprises are becoming increasingly aware of the significance of interlinking business information across structured and semi-structured data sources. This problem has become more important with the growing amount of semi-structured data often found in XML repositories, web logs, biological databases, etc. Effectively creating links between semi-structured and structured data is a challenging and unresolved problem. Once an optimized method has been formulated, the process of data mining can be implemented in a conjoint manner. This paper investigates a way in which this challenging problem can be tackled. The proposed method is experimentally evaluated using a real world database and the effectiveness and the potential in discovering collective information is demonstrated.**

## I. INTRODUCTION

Currently, information is gathered and stored by enterprises in three forms: structured, unstructured and semi-structured. In structured or relational data, the information is represented in a two-dimensional table called a relation. The information is well structured and the schema or structure of the data is fixed and known beforehand. Unstructured data has no schema that describes the underlying structure of the data, or the form of structure is not helpful for the desired processing task. Examples of unstructured data may include audio, video and text. Processing such unstructured data is very challenging using the currently available data mining methods. Fortunately, semi-structured data refers to an intermediate between the two forms above wherein "tags" or "structure" are associated or embedded within unstructured data. Semi-structured data may not have a fixed structure or schema for a precise description of concept attributes and their relationships. A semi-structured document can be composed of data from several heterogeneous sources each structured in a different way. Semi-structured data if often found in XML databases, RDF databases, molecular databases, graph databases etc. According to [1], in 2003, only 15% of information in enterprises is in form of structured data. More recent studies presented in [2], confirm that 10%-15% of information is in structured form in 1990's, while in the period 2006–2010, this number is expected to reduce well below 5%.

Frequent pattern mining is the most important and difficult task when the aim is to discover useful associations between data objects in a database (i.e. association rule mining) [3]. It consists of finding all the frequent sub-patterns that occur at least as many times as the user supplied minimum occurrence threshold. Semi-structured documents such as XML possess a hierarchical document structure, where an XML element may contain further embedded elements, and each element can have a number of attributes attached to it. It is therefore frequently modeled using a labeled ordered tree. In this scenario, the frequent pattern mining problem becomes that of frequent subtree mining, and depending on the application different types of subtrees are mined and/or different support definitions are used [4, 8]. An induced subtree preserves the parent-child relationships from the original tree while in an embedded subtree the parent-child relationship are allowed to be ancestor-descendant relationships in the original tree. The subtrees can be further distinguished based upon the ordering among the sibling nodes. In an ordered subtree the left-to right order among the sibling nodes needs to be preserved while in an unordered subtree the order of the sibling nodes (and the subtrees rooted at those nodes) can be exchanged and the resulting subtree is still considered the same. For an extensive overview of the frequent subtree mining we refer the interested reader to [4], where different approaches and various implementation issues are discussed in detail.

Currently, there are many well developed techniques for data mining on structured data [3, 5, 6] or semi-structured data [4, 7, 8] on their own, but not for data mining on the combination of both types conjointly. While some work has gone towards schema matching and data integration of structured and semi-structured data, the focus was placed on querying and other knowledge management related tasks, rather than data mining of the merged data [9, 10, 11]. This work is aiming to develop an effective framework and method to carry out the data mining technology on the structured and semi-structured data sources conjointly. A real world example is used to demonstrate the effectiveness of the proposed approach, and to show how additional information can be obtained by mining of the merged data source.

## II. MOTIVATING EXAMPLE

Consider the following problem: A credit card provider wants to find out the common characters in the profiles of

978-0-7695-3401-5/08 $25.00 © 2008 IEEE
DOI 10.1109/SKG.2008.57

those clients who are likely to discontinue the use of the credit card service after three complaints have been received about transactions involving amounts greater than $100?

Fequently, customers'profiles can be easily obtained from Relational Database, such as 'salary', 'age', 'address' and so on. But the reasons why they discontinue the use of a particular bank service are more likely to be found in the XML docuements format, because there are more opportunities for customers lodging complaints through customer contact centres, sending emails, filling survey forms or putting in their opinions through website forums for customers' feedback. These contents can be easily processed into XML format rather than Relational Database format. Currently, many companies are finding it difficult to extract meaningful information from plain text. Customers may make complaints for different reasons, such as 'Fee charged', 'Bad service', 'Faulty Transactions' and so on. This type of information is not easily categorized into relational database without any information being lost. For example, when transferring the contents of a complaint into a well-structured relational database, one will usually extract some keywords or related information from the original contents that will fit within the relational schema. However, the rest of the textual information cannot be represented within the schema and hence some infromation is being lost. These reasons increase the difficulties of storing information from textual content into a Relational Database. A detailed example will be presented as follows:

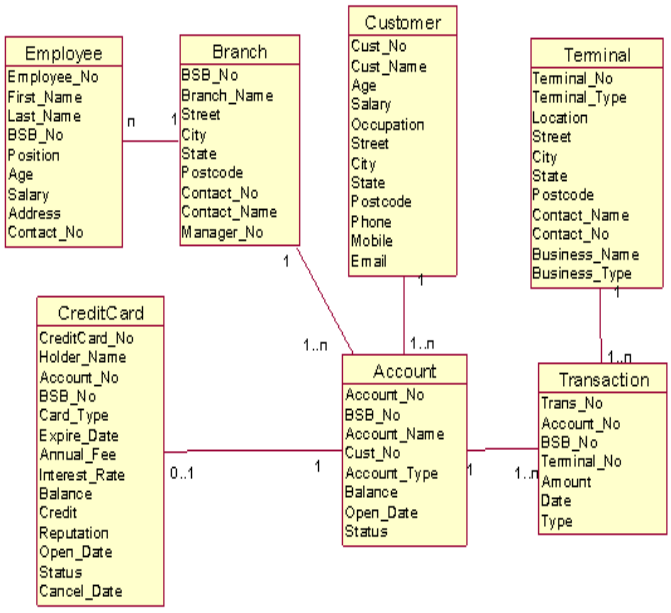



Fig. 1 Example of the Relational Database Schema of a Bank System

We assume that there is a Relational Database which is a typical example of structured data running in the Bank Information System, and separately there is a customer service system collecting and processing the email documents of customer's complaints in XML format which is semi-structured data type. Fig. 1 presents a simple Relational Database Schema of bank system where one can find information such as age, salary ('Customer' table), and the credit card information ('CreditCard' table). However, one cannot discover any information related to customer complaints from the Relational Database.

```
<xs:element name="E-mail">
        <xs:complexType>
          <xs:sequence>
      <xs:element name="To" type="xs:string"/>
      <xs:element name="From" type="xs:string"/>
      <xs:element name="Subject" type="xs:string"/>
      <xs:element name="Body" type="EmailBody"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
<xs:element name="EmailBody">
        <xs:complexType>
          <xs:sequence>
      <xs:element name="Name" type="xs:string"/>
      <xs:element name="Address" type="xs:string"/>
      <xs:element name="Phone" type="xs:string"/>
      <xs:element name="AccountNo" type="xs:string"/>
      <xs:element name="BSBNo" type="xs:string"/>
      <xs:element name="AccountName" type="xs:string"/>
      <xs:element name="OpenBranch" type="xs:string"/>
      <xs:element name="CreditCardNo" type="xs:string"/>
      <xs:element name="TransactionDate" type="xs:string"/>
      <xs:element name="TransactionAmount" type="xs:integer"/>
      <xs:element name="TransactionType" type="xs:string"/>
      <xs:element name="ComplainContent" type="ComplainType"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
```

Fig. 2 Example of the XML Schema of Email

A customer service system is collecting and processing the emails of customer's complaints in XML format (eg. Fig. 2).

Because not all the customers send email with their detail profiles or bank account details, we cannot assume that all the emails will definitely include those key information which can be easily linked to the related records or information in the RDB. Only, when the information from both data sources is linked and integrated, can the example query be answered. The existing data mining methods can only mine structured or semi-structured data sources separately, and hence a new method capable of mining both data sources in a conjoint manner is needed.

## III. PROPOSED METHOD

Fig. 3 displays the general steps of the proposed method, namely Data Pre-processing, Data Linking and Data Mining. We first explain each of these steps at a high level of abstraction and we then go into more detail about the aspects of the Data Linking process.

**Data Pre-processing** includes data extraction, data cleaning and data generalization. The goal of data extraction is to populate the data from the raw database so that the processed data will be more relevant to the user's requirement.

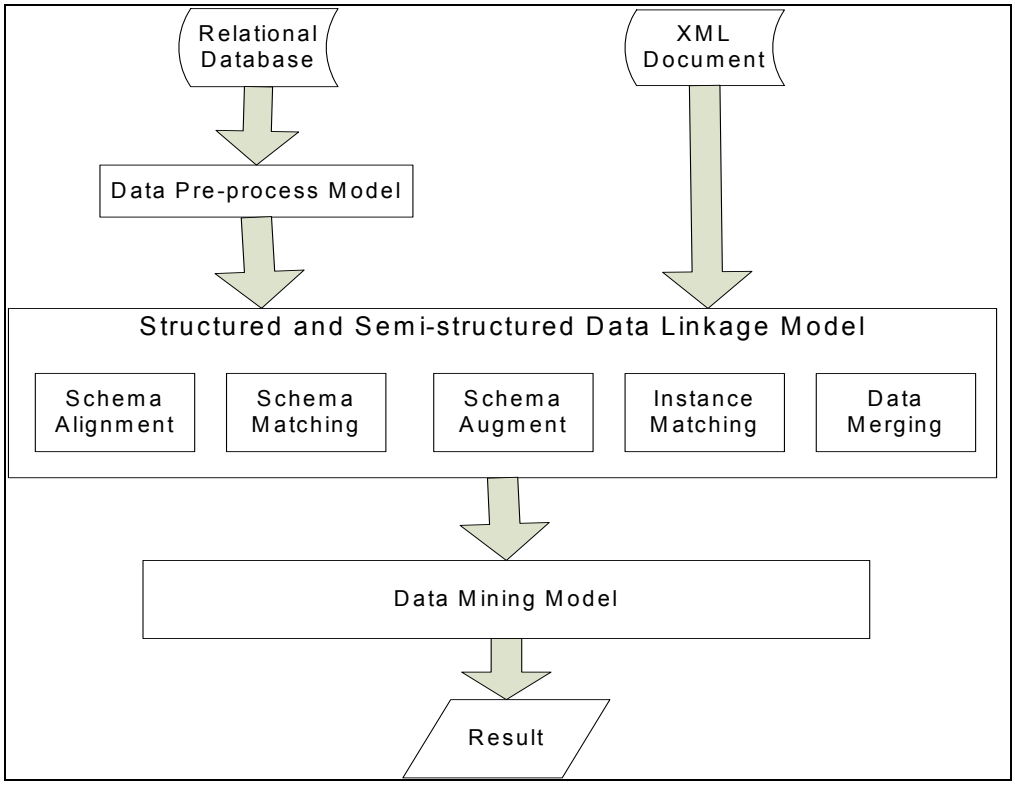



**Fig. 3 General steps of the proposed method**

**Structured and Semi-structured Data Linking** aims to link the relevant data between RDB and XML, and is comprised of 5 sub-steps. Shown in the *Structured and Semi-structured Data Linkage Model* in Fig. 3, it is important to realize that in each of those steps, for example, Schema Matching, we mean matching between the schema for the structured database and the XML schema. This is different from just matching two relational schemas. *Schema Alignment* aims to align the XML schema with the RDB schema so that individual document instances from the XML repository can be aligned with individual records from the structured database. In the whole alignment process, we have 2 different levels at which matching can occur, i.e. logical and instance level. *Schema Matching* is working on the logical level, where the semantic meaning of the tags and attributes in the XML and RDB schema is compared and similar ones are linked together. For this purpose we aim to utilize some of the existing concept matching techniques [11, 12], which can match the concepts using a combination of name similarity, online dictionaries and thesauruses, and descriptive information found in the schema. One may need to consider one-to-one as well as complex matching, as it may be the case that the same aspect of a domain is described by different number of concepts in different data sources. The purpose of *Schema Augment* process is to augment the original XML schema by appending the attributes found in the RDB, that have not been matched to any of the attributes in the XML schema. *Instance Matching* can then be performed to find the values matched from two data sources at the instance level. It may be possible to find multiple instances and record pairs with exactly the same values on both data sources. In this case, for each pair a weight will be calculated depending on the similarity of the original instance and record pair occurring in the data sources. After getting the maximum weight, we can find the best matched instance and record pairs. In fact, this step is also very helpful when there are no matching attributes or tags found from the schemas of both data souces in previous steps. In this case, one can extract some values of crucial attributes in one record from one data source, and use these vaules as keywords to search the contents of the other data source. If a matching attribute from relational schema cannot be found in the XML schema, then one can search for that particular attribute in the textual content of an XML document. For example, if a customer complaint is stored in textual format embedded in the XML document, there is a possibility that some account information may be mentioned in the text and this can reveal which particular customer record from the relational database can be linked to this complaint. This search may produce more than one record whose contents are partially mached with those keywords. This problem can be solved by setting a weight for each candidate record, and determining the best matching record as the record with the highest weight. This involves a process of fuzzy instance matching. The instance matching step enables *Data Merging* to occur where specific records from the RDB are aligned with the corresponding instances in

the XML Document. A new augmented XML database is created by merging the records from the original XML DB with the RDB instances, as defined by the Schema Augment process.

**Data Mining** step is concerned with applying a frequent subtree mining algorithm to extract the potentially useful patterns from the augmented XML document obtained from the previous step. Once the set of frequent subtree patterns has been extracted the association rules can be easily formed according to user specifications.

In what follows we formalize the main aspects of the Data Linking process which is comprised of five steps. We assume that the XML schema and Relational schema will be provided with the XML documents and Relational Database, respectively.

**Step 1: XML Schema Alignment**

Our current research is to investigate a model to deal with the data mining problem based on RDB and an ideal XML documents which only have simple structure, such as the one displayed in Fig. 2. We can have some definition as follows:

**1)** Let T be a set of tags of XML Schema and n be the number of tags in XML Schema, then

$T = \{ t_1, t_2, \ldots, t_i, \ldots, t_n \}$ where $1 <= i <= n$

**2)** Let A be a set of attributes of Relational Schema and m be the number of attributes in the Relational Schema, then

$A = \{ a_1, a_2, \ldots, a_j, \ldots, a_m \}$ where $1 <= j <= m$

**3)** Let D be the set of instance values and q be the number of instances in the XML document, then

$D = \{ d_1, d_2, \ldots, d_p, \ldots, d_q \}$ where $1 <= p <= q$

Let $d_p$ be the set of values for each tags of the XML schema for the $p_{th}$ instance in XML document and n be the number of tags in XML Schema, then

$d_p = \{ d_p(t_1), d_p(t_2), \ldots, d_p(t_i), \ldots, d_p(t_n) \}$ where $1 <= i <= n$

**4)** Let V be the set of record values and t be the number of records in the Relational Database, then

$V = \{ v_1, v_2, \ldots, v_s, \ldots, v_t \}$ where $1 <= s <= t$

Let $v_s$ be the set of values for each attributes of the Relational schema for the $s_{th}$ record in the Relational Database and m be the number of attributes in the Relational Schema, then

$v_s = \{ v_s(a_1), v_s(a_2), \ldots, v_s(a_j), \ldots, v_s(a_m) \}$ where $1 <= j <= m$

**Step 2: Schema Matching**

With the aligned XML schema and Relational schema, we can match some of the elements and attributes from respective data sources and link them. For the ease of explanation, we currently only consider one-to-one mapping from the tags set T to the attributes set A. During the mapping process, we can utilize the schema matching techniques [11] to find the element and attribute matching with the similar terminology. Additionally, the data type is also concerned with the similarity of elements and attributes.

After matching, there will be two subsets: T' and A' which present the matched tags and attributes from the XML schema and Relational schema. Both subsets have the same size and they are in ordered projection, which means each element in T' has one-to-one projection to each element in A' in sequence.

**5)** Let T' be the set of matched tags found in T and l be the number of matched tags in T', then

$T' = \{ t'_1, t'_2, \ldots, t'_k, \ldots, t'_l \}$ where $1 <= k <= l$, and $T' \in T$

**6)** Let A' be the set of matched attributes found in A and l be the number of matched tags in A', then

$A' = \{ a'_1, a'_2, \ldots, a'_k, \ldots., a'_l \}$ and $A' \in A$

In T' and A', $t'_k$ has one-to-one projection to $a'_k$, which presents like $t'_k \rightarrow a'_k$, where $1 <= k <= l$

**Step 3: Schema Augment**

With the subset A', we can find the rest of elements in set A, which haven't one-to-one projection to the elements in set T. Then, we can append those attributes as new tags to the original XML schema. So, a new XML schema with augments, T'', has been constructed.

**7)** Let T'' be the set of tags of the augmented XML schema which is composed of the original XML schema and the unmatched attributes found in the Relational schema, then

$T'' = T \cup \overline{A'}$ where $A' \cup \overline{A'} = A$

So, the new schema T'' combines the original XML schema, T, with $\overline{A'}$.

$\overline{A'}$ = {"age", "salary", "occupation", "mobile", "status", "canceldate", "trans_no"}

**Step 4: Instances Matching**

With the subset T' and A', we can extract the corresponding values from XML documents and Relational database. Sets D' and V' are composed by the values of the tags and attributes which are defined in subset T' and A' from all the instances in XML documents and records in Relational DB. It may be the case that we will find multiple instances and record pairs with exactly the same values on both data

sources. In this case, for each pair a weight will be calculated depending on the similarity of the original instance and record pair occurring in the data sources. After getting the maximum weight, we can find the best matched instance and record pairs.

**8)** Let D' be the set of instance values responding to the tags in T' and q be the number of instances in the XML document, then

$D' = \{ d'_1, d'_2, \ldots, d'_p, \ldots, d'_q \}$ where $1 <= p <= q$

Let $d_p ( t'_k )$ be the value of tag $t'_k$ on pth instance in the XML document and $d'_p$ be the set of values of each tags in T' for the $p_{th}$ instance in the XML document and l be the number of tags in T', then

$d'_p = \{ d_p( t'_1 ), d_p( t'_2 ), \ldots, d_p( t'_k ), \ldots., d_p( t'_l ) \}$ where $1 <= k <= l$, and $d'_p \in d_p$ ($d'_p$ is a subset of $d_p$)

**9)** Let V' be the set of record values responding to the attributes in A' and t be the number of records in the Relational Database, then

$V' = \{ v'_1, v'_2, \ldots, v'_s, \ldots, v'_t \}$ where $1 <= s <= t$

Let $v'_s$ be the set of values for each attributes in A' for the $s_{th}$ records in the Relational Database and l be the number of attributes in A', then

$v'_s = \{ v_s( a'_1 ), v_s( a'_2 ), \ldots, v_s( a'_k ), \ldots, v_s( a'_l ) \}$ where $1 <= k <= l$ , and $v'_s \in v_s$ ($v'_s$ is a subset of $v_s$)

**10)** If $d'x = v'y$ where $1 <= x <= p$ and $1 <= y <= p$, then we can find dx and vy from set D and V which represent the original XML document and Relational DB respectively. If the values are nominal, exact matches will be necessary. If the values are continuously numerical, there will be a range value (e) for limiting the difference of two values ($|d'x - v'y| < e$ which e is a small number).

**Step 5: Data Merging**

By having a set of best matched instance and record pairs with different values, it is easy to locate the original instance in the XML document and the paired record in the Relational database. As we mentioned in step 3, a new XML schema has been constructed and it can be used to merge those values in XML instances and paired Relational database records.

With $d_x$ and $v_y$, $d'_x$ and $v'_y$ where $d'_x = v'_y$, we can find $\overline{v'_y}$ where $v'_y \cup \overline{v'_y} = v_y$

With the augmented XML schema T'', we can migrate $d_x$ and $\overline{v'_y}$ into the T''.

The step 4 is repeated until all the related information from the data sources has been merged.

## IV. Experimental Evaluation

The purpose of this section is to indicate how additional interesting information can be obtained by mining the information contained in a relational and semi structured database in a collective manner. More specifically the aim is to show that the patterns extracted from the augmented document could not be found if each of the data sources was mined separately. At this preliminary stage of our research, the system uses keyword based matching between the concepts rather than the more complex semantic matching. It is common that due to the variety of designs or naming habits of individuals, schemas may be totally different even though they are built for the same concepts or domain. To recognize diverse labels in schemas with similar meanings and make them consistent, semantic matching will be part of our future work where we will in addition utilize online dictionaries or thesauruses, structural information and schema constraints, to enable a more semantic approach (e.g. Gazettes). To solve the problem of how to match the relation or structure of different data sources, we will apply ontology techniques to link structured and semi-structured data by matching not only the concepts but also the relations or structures of the attributes from both kinds of data. The expressiveness, consistency and correctness after schema and data merging has occurred is another aspect that will need to be validated.

In addition, there are some XML documents containing a large amount of unstructured text as the value of one element, such as text from email, survey, complaint, and so on. This may reduce the effectiveness of the currently proposed method, because a large amount of useful information may be found in this part of documents. To solve this problem, we will search the text content and find out the interesting keywords which can be matched with the data in the Relational Database. A detailed discussion of this process and its evaluation is left for our future work.

The data source used is a publicly available datasets obtained from XML Data Repository (http://www.cs.washington.edu/research/projects/xmltk/xmldata/www/repository.html). We choose the data of TPC-H Relational Database Benchmark from Transaction Processing Performance Council (TPC), because there is some common information among the data files. One XML document (*Lineitem.xml*) is selected as the XML data source, while the rest is transferred into a relational database as is shown in Fig. 4. *Lineitem.xml* contains 15 elements and the XML schema is shown in Fig. 5.

The criteria used detect the element "ORDER_ID" and "ORDERKEY" as the keyword pair to connect the XML document and the RDB. Hence, the XML document is matched with the data in the RDB, and combined into a new augmented XML document. The schema of the augmented XML document is shown in Fig. 6.

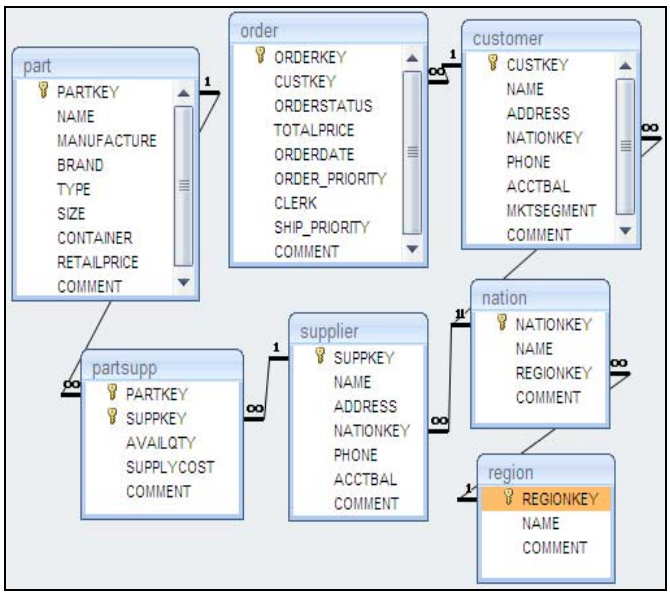



Fig. 4 Relational Database Relationship View

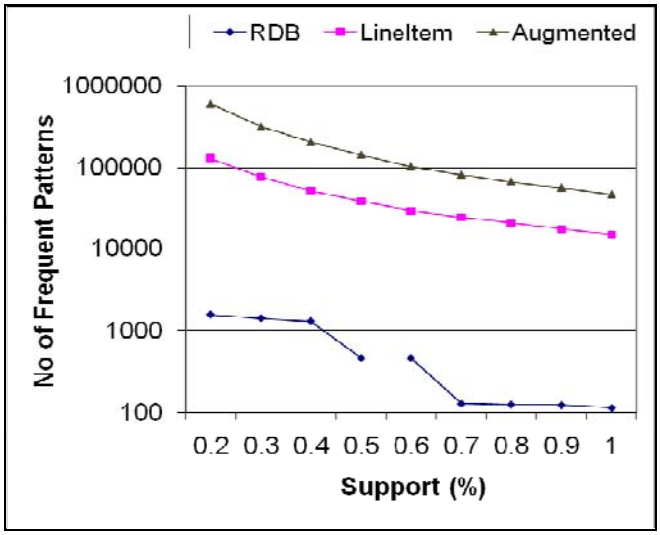
```
<xs:element name="Lineitem" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ORDER_ID" type="xs:int"/>
      <xs:element name="PART_ID" type="xs:int"/>
      <xs:element name="SUPP_ID" type="xs:int"/>
      <xs:element name="LINE_ID" type="xs:int"/>
      <xs:element name="QTY" type="xs:string"/>
      <xs:element name="AMT" type="xs:string"/>
      <xs:element name="DISCOUNT" type="xs:decimal"/>
      <xs:element name="TAX" type="xs:decimal"/>
      <xs:element name="RETURN_FLAG" type="xs:string"/>
      <xs:element name="STATUS" type="xs:string"/>
      <xs:element name="SHIP_DETAIL">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="DATE">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="SHIP_DATE" type="xs:string"/>
                  <xs:element name="COMMIT_DATE" type="xs:string"/>
                  <xs:element name="RECEIPT_DATE" type="xs:string"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="SHIP_STRUCT" type="xs:string"/>
            <xs:element name="SHIP_MODE" type="xs:string"/>
```

Fig. 5 XML Schema of Lineitem.xml

```
<xs:element name="ORDER_ID" type="xs:int"/>
<xs:element name="ORDER_DATE" type="xs:string"/>
<xs:element name="CUST_ID" type="xs:short"/>
<xs:element name="CUST_NATION" type="xs:string"/>
<xs:element name="CUST_REGION" type="xs:string"/>
<xs:element name="PART_ID" type="xs:short"/>
<xs:element name="SUPP_ID" type="xs:byte"/>
<xs:element name="SUPP_NATION" type="xs:string"/>
<xs:element name="SUPP_REGION" type="xs:string"/>
<xs:element name="LINE_NUMBER" type="xs:byte"/>
<xs:element name="QTY" type="xs:string"/>
<xs:element name="AMT" type="xs:string"/>
<xs:element name="DISCOUNT" type="xs:decimal"/>
<xs:element name="TAX" type="xs:decimal"/>
<xs:element name="RETURN_FLAG" type="xs:string"/>
<xs:element name="LINE_STATUS" type="xs:string"/>
<xs:element name="SHIP_DETAIL">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="DATE">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="SHIP_DATE" type="xs:string"/>
            <xs:element name="COMMIT_DATE" type="xs:string"/>
            <xs:element name="RECEIPT_DATE" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="SHIP_STRUCT" type="xs:string"/>
      <xs:element name="SHIP_MODE" type="xs:string"/>
```

Fig. 6 Augmented XML Document Schema

Prior to mining the augmented XML, some data pre-processing takes place in order to obtain simpler and more meaningful results. Duplicate attributes are discarded, eg. “CUST_ID” and “CUST_NAME” come from RDB, but can be seen as redundant in the patterns obtained from the augmented document, and would only make the patterns more complicated and larger in number while there is no additional meaning implicated by them. Furthermore, the values of the continuous attributes such as ‘QTY’ and ‘AMT’ are grouped into ranges (discretized) in order to reduce the number of diverse values. Otherwise it is very unlikely that any association can be made with continuous attributes, as they would have so many different values that they would not occur in the extracted frequent subtree set.




| Support (%) | RDB | LineItem | Augmented |
|---|---|---|---|
| 0.2 | 1569 | 128339 | 474707 |
| 0.3 | 1425 | 76623 | 239238 |
| 0.4 | 1309 | 50901 | 154184 |
| 0.5 | 464 | 38517 | 106216 |
| 0.6 | 464 | 29421 | 74424 |
| 0.7 | 129 | 24057 | 58508 |
| 0.8 | 126 | 20413 | 47516 |
| 0.9 | 124 | 17301 | 40120 |
| 1 | 115 | 14736 | 32806 |

Fig. 7 Number of frequent patterns extracted from different documents

To mine the RDB we use the Apriori [3] implementation by Bodon [6], while for mining of XML documents we use the IMB3-Miner algorithm [4] for mining of ordered induced subtrees. Ordered induced subtrees preserve the left-to-right ordering among the sibling nodes and preserve all the parent-child relationships in the mined tree database. This type of subtree is sufficient to capture the interesting relationships among the data objects. All the documents consist of 60175 records and the number of frequent patterns (itemsets for RDB, subtrees for XML) detected for varying support thresholds (displayed in the percentages of the database records) are displayed in Fig. 7. The reason for choosing such small support thresholds is that in this particular example the RDB does not contain many frequently occurring patterns, and for the frequent patterns from the RDB to contain reasonable long patterns, the support threshold had to be quite low. Even the patterns that occur in at least 0.8% of the database do not consist of more than 2 items. As is evident from Fig. 6, many more patterns can be found when the RDB and *LineItem.xml* are merged together and mined conjointly.

When analysing the extracted pattern we have observed that some associations found in the patterns from the augmented document could not be detected from the patterns found in each of the data sources separately. For example, at support 0.7 we find that the only frequent 3-itemset in RDB is: supp_nation(UNITED_STATES)_cust_region(AMERICA)_supp_region(AMERICA), which itself does not contain any useful information. When looking at the extracted frequent pattern set from the *LineItem.*xml there is really no information that can be linked with this pattern as may be the case for lower support thresholds, when the common item "SUPP_ID(x)" may be found. However, in the frequent pattern set from the augmented document a 5-pattern is found: cust_region(AMERICA)_supp_nation(UNITED_STATES)_supp_region(AMERICA)_return_flag(N)_line_status(O). Please note that for the ease of comparison we have not included the hierarchical organization of the pattern, but have just highlighted the additional information obtained by mining the augmented document.

These results are consistent among the different support thresholds, in the sense that whatever pattern is obtainable from RDB a longer version of it can be found in the augmented document that contains additional associations. This is particularly evident when the support threshold is even further increased. For example when the support is set to 10%, there are only 4 frequent 1-itemset patterns extracted from RDB which cannot be linked in any way to frequent patterns from the Lineitem.xml. On the other hand, the frequent patterns extracted from the augmented document are always larger in number and length, and contain the collective information and associations that could not be detected if each of the data sources was mined separately. These results indicate the importance of being able to link related information across structured and semi-structured data sources in order to detect collective knowledge patterns that can potentially reveal further insights to the user.

## V. Conclusion

The work presented in this paper has demonstrated the importance of data mining structured and semi-strcutured data in a conjoint manner. A framework was proposed to link the related information among the data sources together, in order to merge them into an augmented database that will contain more complete and combined information. As experimentally demonstrated, mining this merged database can reveal patterns and associations that otherwise would not be detected. In our future work the method will be accompanied with semantic concept matching, to enable a more sophisticated approach to data merging.

## References

[1] R. Blumberg & S. Atre, "The Problem with Unstructured Data. *DM Review Magazine*, February, 2003

[2] M. L. Brodie, "Computer science 2.0: a new world of data management", In *Proc. of the 33rd international Conference on Very Large Data Bases*, Vienna, Austria, September 23-27, 2007.

[3] R. Agrawal, T. Imielinski, A. Swami, "Mining association rules between sets of items in large databases", In *Proc. of the ACM SIGMOD Conference on Management of Data*, Washington, DC, USA, 1993, pp. 207-216.

[4] H. Tan, F. Hadzic, T.S. Dillon, E. Chang, "State of the art of data mining of tree structured information", *CSSE Journal*, vol. 23, no 2, March, 2008.

[5] R. Agrawal & R. Srikant, “Fast algorithm for mining association rules”, In *Proc. of VLDB Conf. 1994*, Santiago de Chile, Chile, 1994, pp. 487-499.

[6] F. Bodon, “A fast apriori implementation”, In *Proc. of IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, Florida, USA, Nov 19-22, 2003.

[7] H. Tan, F. Hadzic, T.S. Dillon, L. Feng, E. Chang, “Tree Model Guided Candidate Generation for Mining Frequent Subtrees from XML”, *ACM Transactions on Knowledge Discovery from Data,* Volume 2, Issue 2, July 2008.

[8] Y. Chi, S. Nijssen, R.R. Muntz, J.N. Kok, “Frequent Subtree Mining--An Overview” *Fundamenta Informaticae, Special Issue on Graph and Tree Mining*, vol. 66, No. 1-2, 2005, pp. 161-198.

[9] P. McBrien and A. Poulovassilis. “A Semantic Approach to Integrating XML and Structured Data Sources”. In *Proc. CAiSE'01, Interlaken, June 2001*. Springer-Verlag LNCS 2068, pp 330-345.

[10] C. Beeri and T. Milo. “Schemas for integration and translation of structured and semi-structured data”. In *Proc. of ICDT’99*, 1999.

[11] H. Do and E. Rahm. “COMA: a system for flexible combination of schema matching approaches”. In *Proc. of the 28th VLDB*, Hongkong, China, 2002.

[12] P. Shvaiko & J. Euzenat. “A survey of schema-based matching approaches”, *Journal on Data Semantics IV, 2005*.