USING SCHEMA MATCHING IN DATA TRANSFORMATIONFOR WAREHOUSING WEB DATA

Abdelmgeid A. Ali, Tarek A. Abdelrahman, Waleed M. Mohamed

***Abstract**: Data warehousing is one of the more powerful tools available to support a business enterprise, it provides a multidimensional view of data in an intuitive model designed to match the types of queries posed by analysts and decision makers. Schema mapping plays a key role in many different applications, such as schema integration, data integration, datawarehousing, data transformation, E-commerce, peer-to-peer datamanagement, ontology matching and integration, semantic Web. In order to analyze e-commerce and make reasonable business plans, a company's local data is not sufficient. Decision making must also be based on information from suppliers, partners and competitors. This external data can be obtained from the Web in many cases Such XML, but must be integrated with the company's own data, for example, in a data warehouse. To this end, Web data has to be mapped to the star schema of the warehouse. In this paper we propose a semi-automatic approach to support this transformation process. Our approach is based on the use a XML Schema representation of Web data and the existing warehouse schema. Based on this common view we can compare source and target schema to identify correspondences. We show how the correspondences guide the transformation to be accomplished automatically. We also explain the meaning of Data cleaning and apply it on XML web data to restructuring web data according to DW(Data Warehouse) schema, which are the core of the transformation process using XSLT(Extensible Stylesheet Language Transformations) and XPATH(XML Path Language).*

1 Introduction

The end of the 20th Century has seen the rapid development of new technologies such as web-based, communication, and decision support technologies. Companies face new economical challenges such as e-commerce or mobile commerce, and are drastically changing their information systems design and management methods. They are developing various technologies to manage their data and their knowledge. These technologies constitute what is called ―business intelligencell. These new means allow them to improve their productivity and to support competitive information monitoring. In this context, the web is now the main farming source for companies, whose challenge is to build web-based information and decision support systems. Our work lies in this field. Information from the Web has already become of major importance in helping individuals and companies to follow the current development in many areas, analyzing market developments and making smart business decisions. Warehouse ETL (Extraction, Transformation and Loading of data) is an essential part of data warehousing where the data warehousing professional populate data warehouse with information from production databases and flat files (XML web data). In an online book- shop, a data warehouse, for example, can be used to manage business transaction data, such as customer orders and promotions. The implementation of OLAP (Online Analytical Processing) on the data warehouse will help to gain an insight into customer behavior, perform buy and replenish analysis, and design focused promotions. However, in order to analyze market trends and make new business plans, a company's own data is not sufficient, the bookshop manager also needs information from his suppliers, partners, and about his competitors. For example, discount book information or

information about new publications from his competitors is important to help him to better plan own production lines or offer new promotions. Such information can be acquired from the Web. Integrating Web data and a company's data, materializing them in a data warehouse for implementing OLAP on them, making business plans based on them, and mining historic data to deduct business rules will greatly benefit e-commerce.

In this paper we propose a semi-automatic transformation approach based on the comparison of source- schema and target-schema. First, Web data is integrated based on a common structural and semantic basis by using XML Schema Definition (XSD).We refer to this step as the Web data representation and integration phase. Next is the transformation phase. In this step our source has already become XML representing the Web data, while the target schema is the relational data warehouse schema. The transformation task is to map XML data to relational warehouse tables. We create XML schema from XML data and represent DW tables as XSD. Once the source schema and the target schema are viewed as XSDs, we can compare source and target schemas to find correspondences between them. Therefore the transformation from semi-structured or unstructured Web data to well-structured relational data warehouse data can be implemented through XML schema. We define the correspondences between XML schema for web data and warehouse table's schemas in a mapping rule definition file. The mapping process designed as XSLT for all transformation tasks, therefore it can be done automatically. For a new application we only need to specify the correspondences between semantic concepts of new XML web data and column names in new warehouse tables in the mapping rule definitions, while the actual transformation process will not be changed.

2 An Overview of the Process for Warehousing Web Data

2.1 An Application Techniques

Online book shopping is a very active e-commerce area. A large amount of customer orders is produced every day, and can be recorded in a bookshop data warehouse for OLAP and decision making. Figure 1 shows such a customer order. In addition, the book shop manager may also integrate discount book information from his competitors' in this data warehouse, in order to compare pricing schemes, analyze market trends and make new business plans. This information can be obtained from related Web pages. Figure 2 shows discount book information from an online provider given as a HTML page.

The bookshop data warehouse in our example is based on relational data model. The star schema of simplified bookshop data warehouse is defined in Figure 3. This data warehouse manages e-commerce data of an online book- shop and discount book information from competitors Web pages. E-business data populates the book shop Fact table, while Web data populates the discount fact table. The two fact tables can share several well-conformed dimensions, e.g., Book Dimension and Time Dimension table, or they may also have their own dimension table, e.g., Customer Dimension and Bookstore Dimension.

Figure 1: A customer order

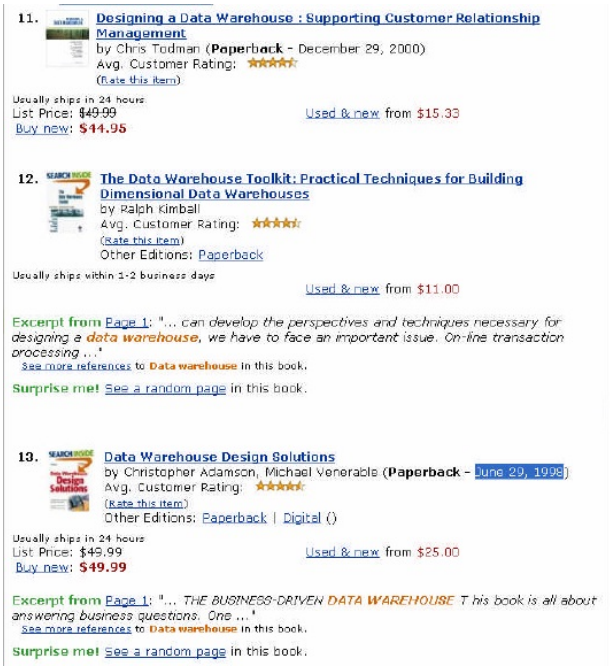

Figure 2: Discount book information from source

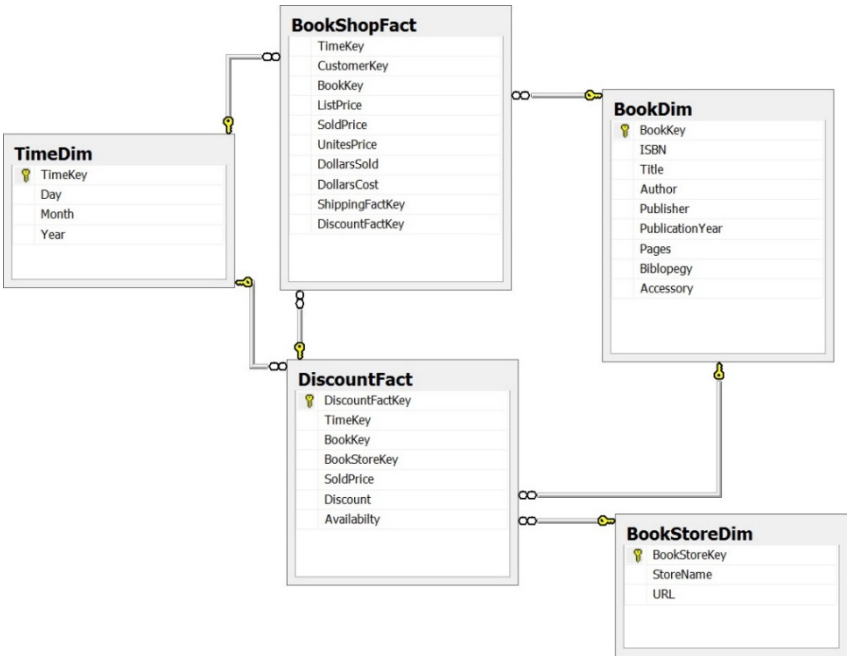

Figure 3: Star schema of the online bookshop data warehouse (OnlineBookStoreDW)

2.2 The Warehousing Web Data Extraction Framework

Our approach framework that provides a platform for integrating Web data and materializing it into a relational data warehouse has been introduced in [2, 9]. The implementation of our framework is outlined in Figure 4. Components such as Transformation Processor, Web Data Wrapper and XML Ontology are located at the organization possessing the data warehouse [3]. Web data sources are available via the Internet. Then web data represented using XML. The framework is based on Dot Net technology and standards like XML schema (XSD), and XPath. Extracted XML file is cleaned and transformed into formal XML format according to the data Warehouse schema. After transformation, the transformed XML file can be loaded into the data warehouse easily depending on loading utilities in the used DBMS product.

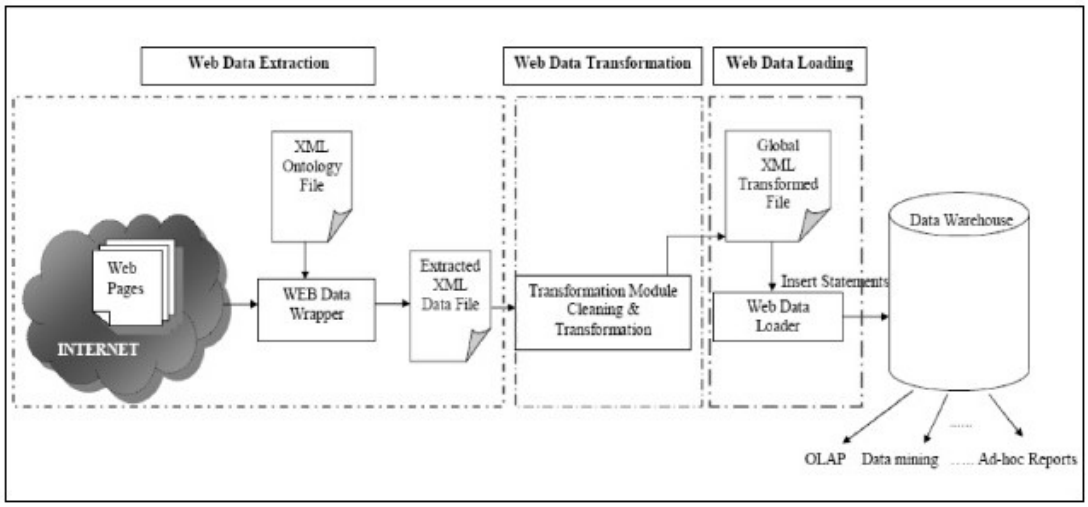

Figure 4: System architecture for warehousing Web data

Integrating Web data and a company's data, materializing them in a data warehouse for implementing OLAP on them, making business plans will greatly benefit e-commerce. Although powerful tools for extracting, transforming and loading (ETL-tools) data from source operational systems into a data warehouse are available today for commercial products. On the other hand, there is a growing need to integrate also external data, such as market information, into these systems [4].

The web data integration process in the wearhousing framework is divided into Three Major processes:

- At first, Web data extraction[5,6], wrappers are generated according to the specific structure depending on the designer requirements in XML Ontology file, and then the wrapper extracts desired information from HTML Web pages to a structured XML format that can be queried and processed by other applications[7]. Wrappers can be generated by non-technical personnel. Wrappers are typically generated by employees with the relevant business expertise for the project, e.g. from a company's marketing department.
- In a second process, web data transformation, XML data generated by wrappers is processed by the transformation module. A wrapper retrieves the web data automatically. Additionally, the transformation module can combine, transform and re-format data from different wrappers.
- Finally, web data loading process, the framework delivers the structured extracted information in the desired XML format to other Business Intelligence systems or data warehousing environments.

The data given in Figure 2 can be represented as XML file have root BookOffers. Figure 5 shows a sample of the integrated data corresponding to the books of Figure 2.

3 XML Schema Representation

As introduced in Section 2, XML file is used in our system as a Web data representation model. We create Schema from XML file in Figure 2. Figure 6 shows this representation.

A relational database can also be represented as XML Schema. The data warehouse in our system is based on the relational data model and can be represented as a set XML Schema's. The table name is the root element, child elements of the root are columns of the table. Figure (7, 8) gives an example of the XSD representation of two data warehouse tables shown in Figure 3

```
<?xml version="1.0" encoding="UTF-8"?>
- <BookOffers>
  - <BookOffer id="1">
      <StoreName>SourceA</StoreName>
      <URL>http://www.bookshop.com</URL>
      <OfferDate>2013-01-01</OfferDate>
      <SoldPrice>28.50</SoldPrice>
    - <Book>
        <ISBN>0764531999</ISBN>
        <Title>XML:Extensible Markup Language</Title>
        <Author>Harold, Elliotte Rusty</Author>
        <Publisher>IDG Books</Publisher>
        <PublicationYear>1998</PublicationYear>
        <Pages>426</Pages>
        <Biblopegy>paperback</Biblopegy>
        <Accessory>CD Rom</Accessory>
      </Book>
      <ListPrice>39.99</ListPrice>
      <Discount>29</Discount>
      <Availabilty>true</Availabilty>
    </BookOffer>
  - <BookOffer id="2">
      <StoreName>SourceA</StoreName>
      <URL>http://www.bookshop.com</URL>
      <OfferDate>2013-01-01</OfferDate>
      <SoldPrice>34.95</SoldPrice>
    - <Book>
        <ISBN>0070411034</ISBN>
        <Title>Web Warehousing and Knowledge Management</Title>
        <Author>Mattison, Rob</Author>
        <Publisher>McGraw-Hill</Publisher>
        <PublicationYear>1999</PublicationYear>
        <Pages>576</Pages>
        <Biblopegy/>
        <Accessory/>
      </Book>
      <ListPrice>49.00</ListPrice>
      <Discount>29</Discount>
      <Availabilty>false</Availabilty>
    </BookOffer>
</BookOffers>
```

Figure 5: Extracted Web data in XML representation

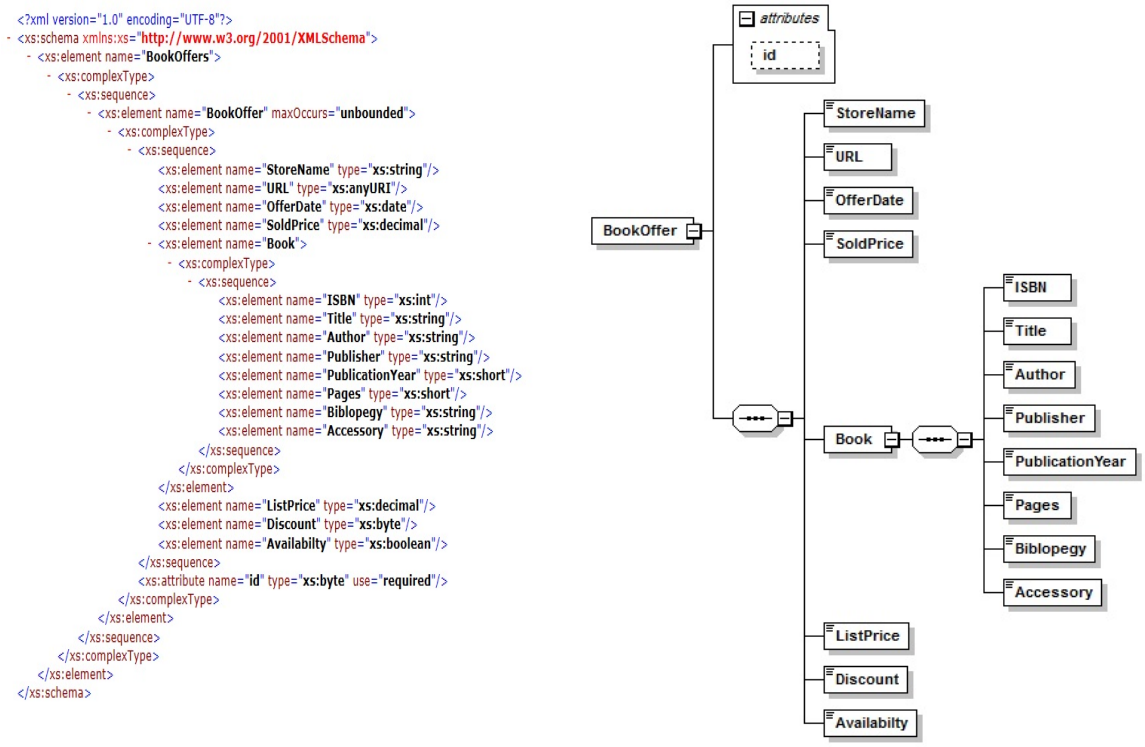

Figure 6: XSD representation for XML Web data

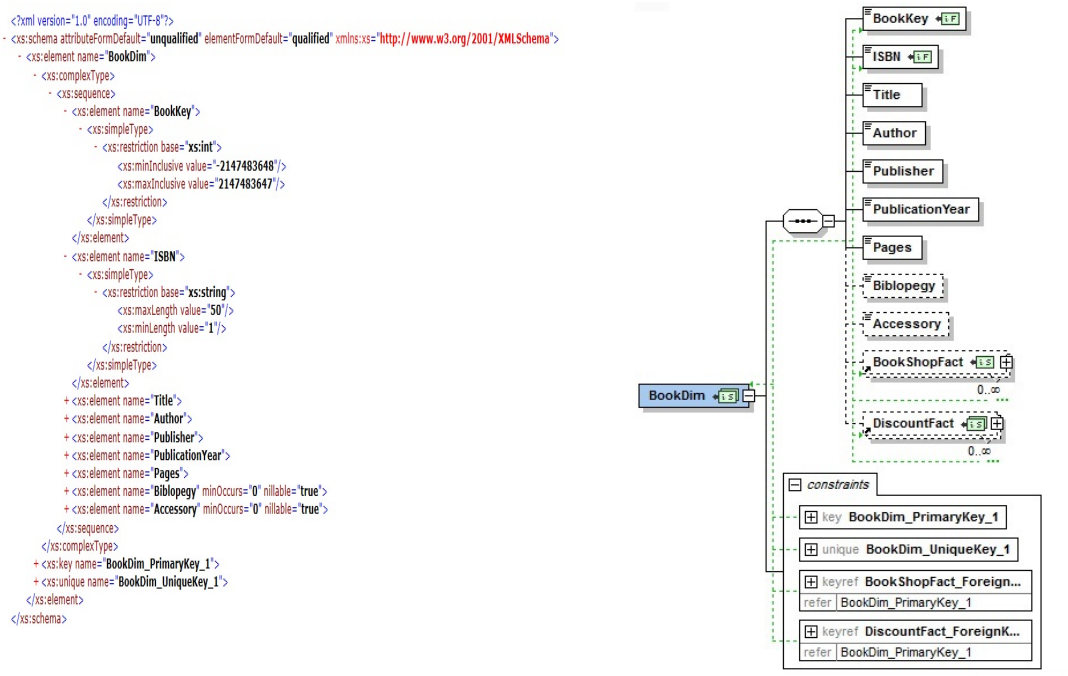

Figure 7: XSD representation of warehouse BookDim table

Figure 8: XSD representation of warehouse DiscountFact table

4 Structure Warehouse Tables through Restructuring XSD Representation

4.1 Mapping Rule Definition

Based on the XSD representations in Figures (6, 7, 8) we can observe that the schema of source and target have correspondences. Constructing a warehouse table can be achieved by generating XML schema. We specify correspondences explicitly in a mapping rule definition file, where we specify not only the schema correspondences between XML schema of web data and columns of warehouse tables schema, but also define the mapping rules from data values of XML data to values of columns,Figure 9 show this mapping. Our mapping rule definitions are written by using XSLT and XPATH. When defining correspondences between XML and warehouse tables in XSLT file.

The data transformation process typically consist of multiple steps where each step may perform schema- and instance-related transformation (mappings)[8]. In metadata and data warehouse, a data transformation converts a set of data values from the data formate of a source schema into the data formate of a destination schema.

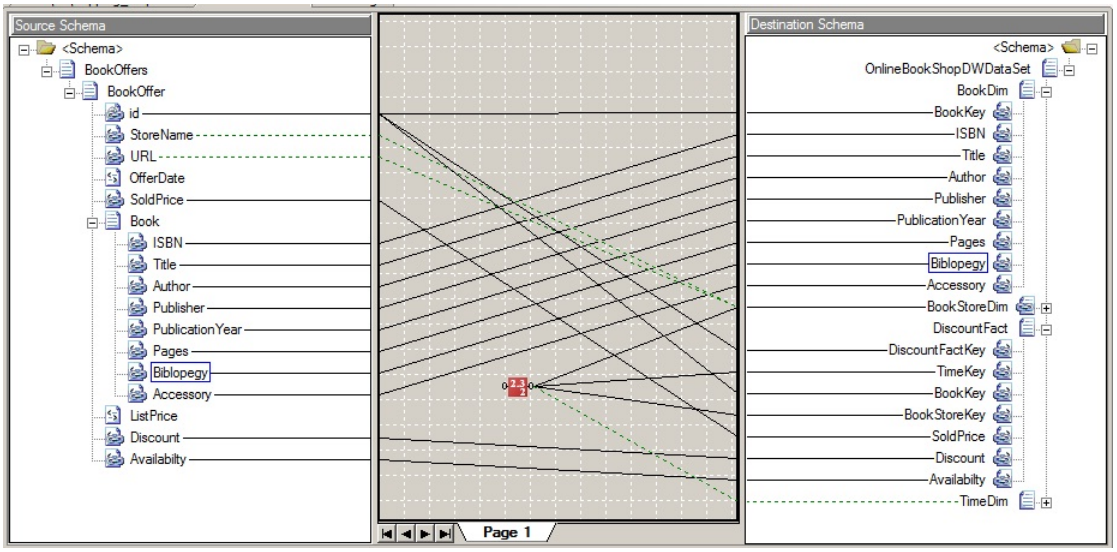

Figure 9: Mapping File between Source schema Book.xsd and target Warehouse OnlineBookshopDW

The following XSLT file (xslt1) show the correspondence between XML Schema (source data-figure 6) and the schema of Book Dimension table (Target data-Figure 7).

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- This file was generated by Waleed -->
- <xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:fn="http://www.w3.org/2005/xpath-
  functions" exclude-result-prefixes="fn xs xsi xsl">
  <xsl:output method="xml" encoding="UTF-8" indent="yes" />
 - <xsl:template match="/BookOffers">
  - <BookDim>
   - <xsl:attribute name="xsi:noNamespaceSchemaLocation" separator="">
      <xsl:sequence
       select="'C:/Users/admin/Desktop/BOOKON~1/Mapping/OnlineBookStoreDW.xsd'" />
     </xsl:attribute>
   - <xsl:for-each select="BookOffer">
      <xsl:variable name="var_BookOffer" select="." />
     - <xsl:for-each select="Book">
      - <xsl:for-each select="$var_BookOffer/@id">
         <xsl:variable name="var_id_byte" as="xs:byte" select="xs:byte(.)" />
       - <BookKey>
          <xsl:sequence select="xs:int($var_id_byte)" />
         </BookKey>
       </xsl:for-each>
      - <xsl:for-each select="ISBN">
         <xsl:variable name="var_ISBN_int" as="xs:int" select="xs:int(.)" />
       - <ISBN>
          <xsl:sequence select="xs:string($var_ISBN_int)" />
         </ISBN>
       </xsl:for-each>
      + <xsl:for-each select="Title">
      + <xsl:for-each select="Author">
      + <xsl:for-each select="Publisher">
      + <xsl:for-each select="PublicationYear">
      + <xsl:for-each select="Pages">
      + <xsl:for-each select="Biblopegy">
      + <xsl:for-each select="Accessory">
       </xsl:for-each>
     </xsl:for-each>
   </BookDim>
  </xsl:template>
</xsl:stylesheet>
```

The following XSLT file(xslt2) show the correspondence between XML Schema (source data-Figure 6) and the schema of Discount Fact table (Target data-Figure 8).

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- This file was generated Waleed -->
- <xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:fn="http://www.w3.org/2005/xpath-
  functions" exclude-result-prefixes="fn xs xsi xsl">
  <xsl:output method="xml" encoding="UTF-8" indent="yes" />
- <xsl:template match="/BookOffers">
  - <DiscountFact>
    - <xsl:attribute name="xsi:noNamespaceSchemaLocation" separator="">
        <xsl:sequence
          select="'C:/Users/admin/Desktop/BOOKON~1/Mapping/OnlineBookStoreDW.xsd'" />
      </xsl:attribute>
      <xsl:variable name="var_const" as="xs:decimal" select="1" />
    - <xsl:for-each select="BookOffer">
      - <xsl:for-each select="@id">
          <xsl:variable name="var_id_byte" as="xs:byte" select="xs:byte(.)" />
        - <DiscountFactKey>
            <xsl:sequence select="xs:int($var_id_byte)" />
          </DiscountFactKey>
        </xsl:for-each>
      - <TimeKey>
          <xsl:sequence select="xs:int($var_const)" />
        </TimeKey>
      - <xsl:for-each select="@id">
          <xsl:variable name="var_id_byte" as="xs:byte" select="xs:byte(.)" />
        - <BookKey>
            <xsl:sequence select="xs:int($var_id_byte)" />
          </BookKey>
        </xsl:for-each>
      - <BookStoreKey>
          <xsl:sequence select="xs:int($var_const)" />
        </BookStoreKey>
      + <xsl:for-each select="SoldPrice">
      + <xsl:for-each select="Discount">
      + <xsl:for-each select="Availabilty">
        </xsl:for-each>
    </DiscountFact>
  </xsl:template>
</xsl:stylesheet>
```

5 Mapping Rules

1. Composition relationships of concepts:

The relationships between a XML elements and its ComplexType sequence (in Figure 6) can be understood as part of relationships. When mapping a complex XML data to warehouse tables, we must distinguish between two cases: First, if the complex semantic sequence populates only a dimension table, the hierarchy of the tree will be flattened, see mapping definition (xslt1). Second, when the complex XML data populates a fact table as well as dimension tables, the element must be decomposed and sequence elements are separately mapped to facts in the fact table and attributes in dimension tables, such as in the case of (xslt2).

2. Identifying attributes or surrogate keys:

In the XML Source each complex element is identified through a single or multiple identifying attributes, similar to key attributes in the relational model. For instance, in Figure 5 a complex semantic element *BookOffer* is identified by its attributes: *StoreName, URL, OfferDate, Price* and *Book* which are complex element. A complex sequence element of concept *Book* is identified by only the attribute *ISBN*. Two complex elements of the same ontology concept are identified by the same set of attributes. We can use surrogate keys in data warehouses, as already shown in Figure 3. An additional column for a surrogate key is created when a warehouse table is designed. The system can generate a unique key for these tables when transforming the corresponding data.

3. One XML to one/many columns:

When mapping objects of an ontology concept to table columns we must distinguish between three cases. In the easiest case we have a direct 1:1 mapping, i.e., a concept is directly mapped to a column, and the values of the corresponding XML Shema are directly assigned to the corresponding table columns. In the second case, we must calculate the values of a column by applying a specified function on the respective XML element. For

instance, *US dollar* is converted to *pound EG*, for this a conversion function will be used in the corresponding mapping rule in (xslt2):

```
- <xsl:for-each select="SoldPrice">
    <xsl:variable name="var_SoldPrice_decimal" as="xs:decimal" select="xs:decimal(.)" />
  - <SoldPrice>
      <xsl:sequence select="($var_SoldPrice_decimal) * (xs:decimal('6.50'))" />
    </SoldPrice>
  </xsl:for-each>
```

4. Finally

When values of a XML element must be decomposed to multiple table columns (1:n mapping), decomposition functions have to be applied to calculate suitable values. For example, OfferDate in Figure 5 is in the form of "MM DD, YYYY". In the Time Dimension of the data warehouse in Figure 3 we have Day, Month, and Year columns. Therefore, we must use de- composition functions, like:

getYearfromDate("01, 01, 2013") ⟶ "2013" , to generate values for these columns:

```
- <xsl:for-each select="BookOffer">
  - <xsl:for-each select="OfferDate">
      <xsl:variable name="var_OfferDate_date" as="xs:date" select="xs:date(.)" />
    - <Day>
        <xsl:sequence select="xs:int(fn:day-from-date($var_OfferDate_date))" />
      </Day>
    </xsl:for-each>
  </xsl:for-each>
- <xsl:for-each select="BookOffer">
  - <xsl:for-each select="OfferDate">
      <xsl:variable name="var_OfferDate_date" as="xs:date" select="xs:date(.)" />
    - <Month>
        <xsl:sequence select="xs:int(fn:month-from-date($var_OfferDate_date))" />
      </Month>
    </xsl:for-each>
  </xsl:for-each>
- <xsl:for-each select="BookOffer">
  - <xsl:for-each select="OfferDate">
      <xsl:variable name="var_OfferDate_date" as="xs:date" select="xs:date(.)" />
    - <Year>
        <xsl:sequence select="xs:int(fn:year-from-date($var_OfferDate_date))" />
      </Year>
    </xsl:for-each>
  </xsl:for-each>
```

5. Default values:

When we map XML Data to the tables of the data warehouse, we sometimes do not have values for all attributes. Thus, it may be necessary to use some default value in these places, such as *"Paperback"* as the default value of *Bibliopegy* (the art of bookbinding). For example, we can use the following definition:

```
- <Biblopegy>
  - <xsl:choose>
    - <xsl:when test="((('0' != xs:string(Book/Biblopegy)) and ('false' != xs:string
        (Book/Biblopegy))) and fn:boolean(xs:string(Book/Biblopegy)))">
        <xsl:sequence select="xs:string(Book/Biblopegy)" />
      </xsl:when>
    - <xsl:otherwise>
        <xsl:sequence select="'Paperback'" />
      </xsl:otherwise>
    </xsl:choose>
  </Biblopegy>
```

6 Experimental Results

Traditional mapping tools, particularly those that are designed specifically for Electronic Data Interchange (EDI) transactions, follow the same generic model:

1. The source is parsed from beginning of file to end of file.
2. Data is extracted from the source in the order that it is encountered.
3. Mapping rules are constructed as the source is parsed.
4. Data is pushed to an output file structure based on the structure and content of the source.

The better traditional engines are two-pass, meaning that they parse the source twice. The first time they extract and store all of the source data in a manner that makes any data accessible at any time in the rest of the process. The second time they go through the source to build the rules of the map. Most of the traditional engines that are not two-pass provide means, such as variables, in which the mapper may store data for later processing.

In this paper, maps work in exactly the opposite fashion. The mapping rules are built not from the source but from the target as the following scenario:

1. The mapping engine traverses the target from beginning to end.

2. Mapping rules are constructed and executed as links are encountered on the output side.

3. Data is extracted from the source when a link is encountered in the target.

4. Data is pulled into the file structure based on the structure of the output schema.

These are very high-level models of traditional and mapping engines but suffice to show that the basic flow of the map differs between them.

After applying the XSLT files(in sections 4 and 5) to web data (source)we get the target output in XML formate XML file for each table in warehouse Figure 10 and Figure 11.

The following figures show sample from output as taget schema formate.


```
<?xml version="1.0" encoding="UTF-8" ?>
- <BookDim xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="C:/Users/waleed/Desktop/BOOKON~1/Mapping/OnlineBookSt
    <BookKey>1</BookKey>
    <ISBN>764531999</ISBN>
    <Title>XML:Extensible Markup Language</Title>
    <Author>Harold, Elliotte Rusty</Author>
    <Publisher>IDG Books</Publisher>
    <PublicationYear>1998</PublicationYear>
    <Pages>426</Pages>
    <Biblopegy>paperback</Biblopegy>
    <Accessory>CD Rom</Accessory>
    <BookKey>2</BookKey>
    <ISBN>70411034</ISBN>
    <Title>Web Warehousing and Knowledge Management</Title>
    <Author>Mattison, Rob</Author>
    <Publisher>McGraw-Hill</Publisher>
    <PublicationYear>1999</PublicationYear>
    <Pages>576</Pages>
    <Biblopegy>paperback</Biblopegy>
    <Accessory />
  </BookDim>
```


Figure 10: Target XML data for BookDim


```
<?xml version="1.0" encoding="UTF-8" ?>
- <DiscountFact xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="C:/Users/waleed/Desktop/BOOKON~1/Mapping/Online
    <DiscountFactKey>1</DiscountFactKey>
    <TimeKey>1</TimeKey>
    <BookKey>1</BookKey>
    <BookStoreKey>1</BookStoreKey>
    <SoldPrice>185.25</SoldPrice>
    <Discount>29</Discount>
    <Availabilty>true</Availabilty>
    <DiscountFactKey>2</DiscountFactKey>
    <TimeKey>1</TimeKey>
    <BookKey>2</BookKey>
    <BookStoreKey>1</BookStoreKey>
    <SoldPrice>227.175</SoldPrice>
    <Discount>29</Discount>
    <Availabilty>false</Availabilty>
  </DiscountFact>
```


Figure 11: Target XML data for DiscountFact

The last step of warehousing process, load this result to warehousing tables (fact and dimention tables).

7 Conclusions

In this paper, we propose a semi-automatic transformation approach for materializing Web data into a data warehouse. In this approach we map the XML Schema (source data) and the star schema of data warehouse tables(tareget data) to identify semantic correspondences between them. These correspondences are explicitly described as mapping rules. The transformation can be accomplished automatically via XSLT. The mapping definition file between the web data schema and warehouse tables schema has been created. Some transformation functions and it's conversion has been used to cope up with warehouse tables. In this paper, we put some modifications for the wearhouse design in bookshop data warehouse [1], since the fileds in dimension table(BookstoreDim) must be fixed, we move the avaliability column to the fact table(DiscountFact), this modifications has been done to keep the warehouse attributes. From Experimental result, we observe that the mapping in our approach is faster than the other traditional mapping tools.

References

[1] Y. Zhu, C. Bornhövd, and A. Buchmann. Data Transformation for Warehousing Web Data. In the proceedings of 3rd Intl. Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems(WECWIS 2001),June 2001, San Jose, USA.

[2] Y. Zhu, C. Bornho¨ vd, D. Sautner, A. P. Buchmann: Materializing Web Data for OLAP and DSS, WAIM'00, Shanghai,China, 2000

[3] L. Zamboulis and A. Poulovassilis and J. Wang. Ontology Assisted Data Transformation and Integration. In Proc. Workshop on Ontologies-based Techniques for DataBases in Information Systems and Knowledge Systems (ODBIS'08 at VLDB'08), pp. TBC, 2008.

[4] Arnaud Sahuguet and Fabien Azavant, "Building intelligent web applications using lightweight wrappers", Data Knowledge Engineering,36(3):283–316, 2001.

[5] Xiaofeng Meng, Dongdong Hu, Chen Li, "Schema-Guided Wrapper Maintenance for Web-Data Extraction", In Proceedings of WIDM 2003 New Orleans, Louisiana, November 7-8, 2003.

[6] Ling Liu, David Buttler, James Caverlee, Calton Pu and Jianjun Zhang, "A Methodical Approach to Extracting Interesting Objects from Dynamic Web Pages", College of Computing, Georgia Institute of Technology, USA, 2005.

[7] R. Baumgartner, O. Frlich, G. Gottlob, P. Harz, M. Herzog, and P. Lehmann. Web data extraction for business intelligence: the lixto approach. In Proc. of BTW 2005, 2005.

[8] E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. IEEE Data Engineering Bulletin, 23(4):3‑13,2000.

[9] J. M. Perez, R. B. Llavori, M. J. Aramburu, T. B. Pedersen, Integrating data warehouses with web data: A survey, IEEE Trans. Knowl. Data Eng. 20 (7)(2008) 940-955.

Information about authors

Abdelmgeid A. Ali Associate Professor, Dept. of Computer Science, Al - Minia University,. Egypt

Tarek A. Abdelrahman Assistent Professor, Dept. of Computer Science, Al - Minia University, Egypt

Waleed M. Mohamed Database administrator, Al - Minia University, Egypt