CHAPTER 2 LITERATURE REVIEW

Before the design and implementation phase of this proposed system, a brief literature review has been done and is documented in this chapter. This literature review will provide insights on the relationship between XML documents and relational databases and its significant contribution to XML world. However, the review on the popular native XML databases is beyond the scope of this dissertation. Here, the review also covers the leading edge of XML mapping technologies used in integrating (marshalling and un-marshalling) XML documents with heterogeneous relational databases, and the advantages of mapping XML structure to relational databases structure as well as the requirements involved in designing the XML mapper interface. Besides, data binding technologies are essential in this scenario where it is used to bind directly the objects created by the user that represents the actual database structures such as records or fields. As the user iterates through the database records, the system is able to retain the XML structures and able to generate respective XML documents and their DTDs at the end of the process.

2.1 XML and Databases

In this section, a brief elaboration on the distinction between an XML and relational databases is documented. As mentioned above, XML has been the leading edge technology in today’s information technology world, especially involving integration with the traditional RDBMS and also the talk-about native XML databases. According to some comments from related XML and databases web sites, many of them think that the native XML databases will eventually take over the traditional RDBMS. However, until recently
that these assumptions are proven wrong as they started to realise that XML and RDBMS are complementary technologies that work hand in hand to provide better services to customers (Gicqueau, 2002). In fact, recent initiatives and research on these combined technologies have proven that it provides much better and intelligent usage in storing and retrieving data between them. This is because each has its own strengths that reside in different areas as both of them are structured differently. Before proceeding to review the close relationship between XML documents and relational databases, brief description on each of them will be elaborated below.

2.1.1 XML

The definition of XML is ‘XML is an open standard for defining data elements on a Web page and business documents. In contrast to HTML markup language, which defines how elements on a Web page are displayed, XML defines the structured information those elements contain’ (Guardalben and Atre, 2002). XML is also a markup language that was originated from a subset of SGML together with HTML to use over the Internet via web browsers. XML documents can be viewed as containers for information. Within the primary container may be information and more nested containers, which themselves may contain information and more nested containers (Simon, 2000).

An XML document is a database only in the strictest sense of the term. That is, it is a collection of data. As a “database” format, XML has some advantages [as it is meta-data language] because it is self-describing and it can describe data in tree or graph structures. It also has some disadvantages as it is verbose and access to the data is slow due to parsing and text conversion (Bourret, 2002).
From the paragraph taken above, a simple deduction can be made that XML and its surrounding technologies do constitute a simple database format, or rather like the DBMS format. Some of its advantages are the ability to provide data storage and retrieval in XML document format, data schemas via DTDs and XML schemas, query languages via XQuery, XPath, XQL, XML-QL and so on, programming interfaces via DOM and SAX (Bourret, 2002). On the other hand, its disadvantages are lack of features that are vital in any production environment such as efficient data storage and retrieval, indexes, security, transactions, data integrity, performance, multi-user accesses and lastly, performing complex queries across multiple XML storage. Therefore, it has to be coexisting with RDBMS to complement each other.

2.1.2 Relational Database Management Systems (RDBMS)

Relational database technology has been made famous for the past 30 years old since its introduction as database format into IT business world. It is famous for its features such as powerful searching capability using SQL, impressive querying performance using indexes, normalisation without data redundancy, reliability and scalability with its ability to service a large number of users concurrently, transactions feature, security with locking and caching mechanism, large storage capability and so on.
2.2 XML-Relational Mapping Technologies

As mentioned in the previous section, people now are convinced that relational databases are still the most efficient method of storing and retrieving data and also will require exchanging data with some other applications or systems. And it has been years that relational databases are the critical business infrastructure in most organizations. At present, XML documents can be saved on either legacy relational database or native XML database, which is less popular due to its limitations compared to relational database systems (Guardalben and Atre, 2002). On the other hand, XML is known for its natural format as meta-data; that is data describing about data where the user can understand by its self-describing tags without looking for its meanings or additional information. It is used as the lingua franca standard for data exchange between businesses and content management. However, with its lack in the required database transaction features, it is still not readily be used as a storage and retrieval mechanisms in a real relational database format.

Following this, there is a foreseeable growing need in the near future to continue use relational database systems as the ideal data storage and retrieval mechanisms for XML documents because of their scalability, reliability, security, data integrity, and performance issues as well as for being XML-enabled databases providing best of breed services. In spite of this, it is anything but not straightforward to store XML data directly into relational tables because the XML and relational databases approaches are built on different principles (Guardalben and Atre, 2002). Therefore, it is vital to have tools or middleware that is able to map semi-structured XML data to the relational databases with structured data to extend XML supports to customers.
2.2.1 XML-Relational Mapping Features

In order to achieve seamless integration between XML documents and relational databases, the proposed mapping design will require some skills in both RDBMS and XML data design, and clear and direct methods to map each element of the XML schema to relational database schema. The features of the mapping technologies must:

- Be able to define the format of the XML document that will represent the XML view over relational database based on W3C schema or DTD requirements, as they are recommendations from W3C that define XML document metadata information about classes or templates of XML documents.

- Be based on declarative approach that is flexible without having the application codes whenever there are any XML mapping definition change during design time.

- Be able to define an abstraction layer that is consistent and similar from XML data to relational data by mapping the relationships between XML data and RDBMS data (Guardalben and Atre, 2002). This is done by defining relationships between XML element/attribute and relational table field definitions of a relational database.
• Be able to define the mapping relationships information in XML document format so that it can be parsed by any XML software.

• Provide a GUI mapper to generate the mapping definition files without requiring the users manually deal with complex mapping relationships (Guardalben and Atre, 2002).

• Provide database transaction features such as commit and rollback transactions when inserting or updating the XML data to relational databases.

• Generate SQL statements automatically after the integration process in XML format as it is easier to let the mapping engine do the tedious work for generating suitable SQL queries required to create or manipulate the database.

• Have a mapping engine to reduce a specific vendor lock and be independent of heterogeneous relational database systems, using standard database driver such as JDBC, ODBC or OLEDB (Guardalben and Atre, 2002).

• Be robust and flexible so that it can be easily integrated with other XML tools such as the XPATH and XQuery language where they can search and query semi-structured data in XML documents.

2.2.2 XML-Relational Mapping Requirements

In order to perform an effective XML-relational mapping process, there are three important criteria that need to be taken in consideration. They are mapping definitions, tools used in mapping processes and the capability to integrate query, selection, and transformation of XML processors which will be explained in the following sub-sections (Guardalben and Atre, 2002).
2.2.2.1 Mapping Definition

It is important that XML mapping definitions be direct references from W3C Schema or DTD standards of W3C requirement list. Thus, they should include link element and attribute definitions to fields from database table and column catalog. This can be done through object-relational mapping schema. In addition, to be consistent, the mapping definitions’ syntax should be XML-based in nature. At present, there is still no direct standard in defining the mapping definitions for XML to relational tables.

In spite of several recommendations made on the mapping definitions, they can be categorized into two general approaches where the users can choose from as well as they are supported by W3C until a new standard comes along. Thus the choice between these two is rather subjective and depends on the users. They are based on the following approaches:

- Element-attribute annotation approach where the XML syntax is derived from W3C Schema syntax which is annotated with mapping extensions (Guardalben and Atre, 2002). In this approach, the W3C schema to be mapped contains additional elements as either annotations or new mapping elements by means of additional namespaces.

- Type-refinement approach where the XML syntax is derived from W3C Schema syntax which is extended with new schema types (Guardalben and Atre, 2002). These new types are derived from the ones defined in the schema to be mapped and contain additional required information to establish the mapping rules.
However, no matter what mapping definition approach the users select in their implementation, they should take into consideration of the following vital features provide:

- Clear links between element/attribute definitions in the schema definition and table fields in the database catalog column datasets. It can be done by using the object-relational mapping schema whereby the element/attribute is mapped to object views, and subsequently these objects will be mapped to the relational tables and fields.

- Database referential integrity constraints to allow proper retrieval and storage of complex hierarchical information (Guardalben and Atre, 2002).

- Optional database-table information, such as column data types, precision, scale, primary and foreign key, auto-incremental flag and so on. This is important when the user wish to import XML data into existing tables or new tables in relational database.

- Support for complex mapping in XML syntaxes for mapping SQL expressions on XML elements or XML expression on SQL fields.

- Support for parametrical predicates expression (Guardalben and Atre, 2002). This is useful and convenient to specify constraints on the size of the XML record set, during mapping design-time, when retrieving XML data and storing XML data.

- Support for XML aggregation (Guardalben and Atre, 2002). It is a tedious mapping process to combine hierarchical information in XML format when the users wish to retrieve data from relational tables, especially when the parent elements group more than one child element together; that is, when many child elements share the same parent element. Typically, this scenario happens as a result of SQL join queries on the database server.
• Additional useful information, such as database connectivity details especially when the mapping is to be done only for specific DBMS servers.

• Advance support for XQuery and XPath features in retrieving, querying XML data from XML documents. So far, there is still little effort in mapping definitions that support these W3C recommendations as it is still new in its area.

2.2.2.2 Mapping Tools

Despite the increasing maturity and improvement in mapping technologies, there will always be a need for data modeling changes. Consequently, it is difficult to produce and maintain the syntax consistency of the mapping definitions. In fact, it has to rely on manual editing, instead of the mapping rules (Guardalben and Atre, 2002). As a result, it is preferably that these mapping tools should be designed in XML-based and automated with a user-friendly graphical interface that can assist its users to generate and edit complex mapping instructions. Although at present, it is still lacking in these mapping tools to produce standard mapping definitions processes.

Currently, there are many software-mapping tools available in the market, and most of them implemented almost similar approaches. For example, most of them contain at least two tree-views or rather list-views of the displayed XML data when used in the XML-relational mapping processes. Alternatively, it can be shown as one tree-view represent a basic tree representation of a W3C Schema such as DTD document or XML document, whereas another tree-view provides a virtual XML views of a DBMS catalog tree, such as the qualifiers/databases or tables/columns. Moreover, these tools can include some facilities
to link one element from one view to the next such as from XML documents to relational tables. Other common approaches include drag-and-drop, link activation dialog boxes, and variations of cut-and-paste features. As for the future advance features, the proposed tool can support XML or SQL expressions or scripting for JavaScript or VBScript using an intelligent editor to check and debug for invalid syntaxes or expressions.

2.2.3 Integration of Query, Selection and Transformation XML Processors of XML Views over Relational Databases

As described earlier, the mapping process is a vital methodology to present and define XML views over subsets of relational databases. Here, the software developers have to switch from a database-centric development approach to one that is based entirely on the XML paradigm. Thus, there will be a growing need to use the standardised XML processors, such as XQuery, XPath and XSLT. These XML processors are important to further query, filter, and transform the XML views that obtained from relational data by means of mapping technologies.

In spite of the potential improvement in XML processors, there is one obvious challenge of using XML processors to map XML views over relational databases is that XML views are not materialized fully unless it is cached in memory, stream, or file. On the other hand, this may become a major bottle-neck issue if the XML view obtained from the mapping process is large. This is because either the cached view is too large for disk memory or disc space, or the time needed to post-process the XML view by an XML processor is too long. To overcome these problems, there are several strategies has been suggested. However, two options currently being used by commercial products are shown below:
• Implement a mapping-aware XML processor. For example, these processors can program to understand mapping definitions and translate XML processor statements into native SQL statements. Then the same SQL statement is merged with the one generated by the mapping processor. However, it is only at ad-hoc approach to the solution, but it is also the most labor-intensive. This is because XML processors are complex components to maintain once there are new updated standards.

• Use standard XML processors from third-party software vendors or from open source. The software developer will face the difficulty mentioned earlier if they choose to use off-the-shelf XML processors to perform their post-processing operations after the mapping has already taken place, which is the limited cache size issue. Therefore, they need to first reduce the size of the XML record set generated by the mapping process, and then post-process it by means of any third-party XML processor. In order to reduce the size of the XML record set, they can perform an initial parsing of the XML processor query statement to extract only the relevant predicate information which is the SQL where conditions to append to the SQL generated by the mapping process.

2.2.4 XML-Relational Mapping Advantages

The advantages of using XML mapping technologies to integrate XML-based documents with relational databases are shown below:

• XML is metadata language which has self-describing ability to represent structured data without having its readers looking for any additional information; it is text-based, human and computer readable, platform independent, cost-effective and is also an open standard from W3C (Gicqueau, 2002).
• It is a popular, natural and persistent method used to share data in heterogeneous database systems across the network, either inside the organisation or with business partners, especially to exchange transactional data between business partners or incompatible systems in B2B world.

• Software developers and system analysts who are using this XML-based interface do not require to know the details of SQL programming skills, all they need to know are some basic knowledge of XML concepts in order to use the interfaces. This is because the data from the relational databases is displayed directly as virtual XML document views.

• Standard software components or tools that are used in XML software development such as XML parser, XML filters or even XML processors can seamlessly coexist with the existing relational databases without having to re-structure any existing system (Guardalben and Atre, 2002).

• Both XML documents and relational databases have intrinsic differences, so any tool used in the integration will not be able to gather the implicit semantics of the relational tables and fields (Guardalben and Atre, 2002). Therefore, with the design-time mapping specifications, it can provide the necessary flexibility required to explicitly define these hidden data semantics.
2.3 XML versus Relational Data Structure

Before proceeding to the design of implementing XML mapping definitions to relational databases in the next chapter, it is necessary to perform a brief study on these two different data representation structures in order to be able to accurately define the tree structure from XML documents.

2.3.1 XML Data Structure

According to its structure definition, 'XML data is best represented by a tree structure, and relationships between elements are expressed mostly by containment' (Gicqueau, 2002). This tree structure consists of nodes and nested nodes, where the nodes having relationships as parents and children, are represented by XML elements or tags, which have their own attribute(s) and sometimes also other elements. There are XML rules that declare the details of each element, their order and the attributes that they can take. Subsequently, these XML specifications that define these rules are specified using DTD. DTD will assist any application that wishes to parse XML documents to verify that they are conformed to their DTDs. However, due to DTD’s limitations in data type declarations, XML schema was introduced to give a more flexibility to software developers. The degree of complexity of XML tree is represented by their DTDs or XML schemas. As a result, it is the DTD or XML schema that characterises the XML elements and attributes (Gicqueau, 2002).
2.3.1.1 Document Type Definition (DTD)

Document Type Definition is a file that defines the rules and validates the structure that all XML documents must conform on their XML elements, attributes and sequence of these elements displayed. Further information regarding DTD is defined as part of original XML 1.0 specification from W3C. There are also many downloadable DTDs on the web sites that can be used without writing any DTD from scratch. On the other hand, it is not necessarily that a XML document comes with a DTD file as there is an extra bandwidth burden. This is because XML is considered much simpler than the traditional SGML, where the XML processor is able to process the well-formed XML documents and their functionality for usage by other systems or presentation via web browsers.

2.3.1.2 XML Schema

Similarly, XML schema is defined by W3C. As mentioned above, XML schemas have been introduced because of the limitations found in the traditional DTD. XML schemas define the characteristics of classes of objects or rather the way the objects in an XML document are marked up. Presently, these XML schemas have begun to replace the popular DTDs to overcome some of the limitations found in DTDs (Gicqueau, 2002). It will probably become the de facto standard to represent XML specifications. In contrast to DTDs, which use a different syntax, XML schemas are also XML documents themselves that can be easily viewed using XSL stylesheet schema. Moreover, they support a rich set of data types and element constraints compared to DTDs, and these are very significant when defining relational data. Furthermore, XML schemas also allow repeating definition of XML elements, which was not previously possible with DTDs (Gicqueau, 2002). As a result,
since XML schema's advantages outweighed DTD's advantages in this review, it is recommended best to use XML schemas in the mapping design of XML to relational tables.

2.3.2 Relational Database Structure

On the contrary, a relational database is a database schema that is defined by different database entities such as tables and fields to meet business requirements (Gicqueau, 2002). They store data in the form of related tables where a single database can consist of unlimited number of related tables. Generally, a relational database is represented by a set of tables that are linked together by several types of tables' relationships such as one-to-one, one-to-many, or many-to-many relationships, where they will answer all the business requirements and the required data normalisations. Database normalisation has been a complex issue because all the data must be free of non-redundancy where at any one time that each non-key data can only be present in only one place. In addition, the relational database provides support for a wide variety of data types. Conceptually, each table in a database is made of a fixed collection of columns, which are also known as fields that correspond to the attributes of the data model. Whereas, rows or also known as records, can occur an unlimited number within each table that usually consists information on an entity. A unique primary key is defined in each of these tables that enables other tables to refer to via a foreign key from within the table that holds the same value (Gicqueau, 2002). On the other hand, this foreign key, in turn, is defined to be the primary key for the same tables that linked to the latter foreign key.
2.4 Java Technology

In this section, Java technology is briefly described here as it will be the main programming language used in the development of the proposed system. Java technology is known as both a programming language and a platform. Java technology is best known among software developer communities as “Code Once and Run Anywhere”. Java is a high-level programming language that can be characterised by the following features taken from Sun Microsystems Online Java™ Tutorial:

- Object-oriented
- Simple
- Distributed
- Architecture neutral
- Portable
- Robust
- Secure
- High performance
- Multithreaded
- Dynamic
- Interpreted

Java platform consists of two main components, which are Java Virtual Machine and Java Application Programming Interface (Java API). Java API has a large collection of ready-
made software components that provide many useful capabilities, including support for complex XML programming. As a result, Java technology is selected to use as the programming language to marshal and un-marshal between any XML document and any relational database because Java APIs provide software developers simple and convenient way to do this via its Java Database Connectivity (JDBC), Java API for DOM, SAX, Data Binding, and XML Parser.

2.5 Conclusion

In conclusion, the literature review for this dissertation have performed and documented in this chapter, such as the relationship between XML and relational database systems, a brief overview on their different data representation structure, and XML-Relational database mapping technologies. The reviews done on the XML and relational databases are of significant to this project in order to provide insight knowledge of these two complementing technologies. This is because it is a tedious task in integrating XML data, which is semi-structured in nature, with structured data such as relational databases, as both of these technologies are very different in nature. However, the enabling technologies proposed to be used in this project are also not of less importance as they will be the driving factor in the implementation of a working system. In this context, without suitable selection of programming language, modeling tool, and system design and analysis processes, it might result in the failure of the implementation of the proposed XML-based middleware interface to integrate with relational databases. Therefore, these reviews will be served as a basis and conceptual knowledge used in the later development phases of this project, especially in the system design and analysis as explained in the following chapters.