THE UNIVERSITY OF SHEFFIELD
DEPARTMENT OF COMPUTER SCIENCE
RESEARCH MEMORANDA CS-03-08

MPhil/PhD Upgrade Report

AN ENHANCED DATA MODEL AND
QUERY ALGEBRA FOR PARTIALLY
STRUCTURED XML DATABASE

SUPERVISORS:
DR. SIOBHÁN NORTH
DEPARTMENT OF COMPUTER SCIENCE
DR. BARRY EAGLESTONE
DEPARTMENT OF INFORMATION STUDIES

YASSER ABDEL KADER

NOVEMBER 2003
Abstract

This report documents a preliminary research study of semi-structured data stored on the Web in general and in collections of eXtensible Markup Language (XML) documents in particular.

The research aims to enhance the performance of storing, querying and retrieving XML databases that contain a combination of structured and semi-structured data (this hybrid structuring can be described as partially structured data), so as to better support classes of application where there is a fixed formal framework for data, but also an ad hoc component.

To this end, the research will seek and validate enhanced data model and query algebra for XML databases. Specifically, this aim is to extend existing structured approaches by adapting a relational query optimization technique to be used in XML database.

This will facilitate manipulating partially-structured data. This hybrid approach is likely to have performance advantages over pure semi-structured database approaches as well as information retrieval approaches. This is achieved by utilising existing optimisation techniques developed for conventional databases to deal with the structured part of the data and not treating data as if it is totally semi-structured or un-structured.

The conclusions of this study motivate a research proposal towards enhancing the performance of storing, querying and retrieving XML-encoded data on the World Wide Web (or Web). The report documents the early stages of implementing this proposal and is submitted to support an application by the author to upgrade from MPhil to PhD registration.
# Table of Contents

ABSTRACT ........................................................................................................................................................... II  
TABLE OF CONTENTS .......................................................................................................................................... III  
LIST OF FIGURES ............................................................................................................................................... V  
LIST OF TABLES ............................................................................................................................................... VI  
1. INTRODUCTION ............................................................................................................................................. 1  
2. BACKGROUND AND BASIC DEFINITIONS ................................................................................................. 3  
   2.1 SEMI-STRUCTURED DATA ......................................................................................................................... 4  
      2.1.1 Semi-Structured Data Characteristics ................................................................................................. 4  
      2.1.2 Semi-Structured Data Motivations ....................................................................................................... 6  
      2.1.3 Semi-Structured Data Models .............................................................................................................. 7  
      2.1.4 Semi-Structured Data Research Issues ............................................................................................... 9  
   2.2 EXTENSIBLE MARKUP LANGUAGE (XML) ............................................................................................. 12  
      2.2.1 Proposed Data Model by W3C ............................................................................................................. 15  
   2.3 SEMI-STRUCTURED DATA AND XML ..................................................................................................... 17  
   2.4 PARTIALLY STRUCTURED DATA: A HYBRID STRUCTURED AND SEMI-STRUCTURED APPROACH  
      .................................................................................................................................................................. 20  
      2.4.1 Motivation for Hybrid Approach ....................................................................................................... 21  
      2.4.2 Advantage of Hybrid Model .............................................................................................................. 22  
      2.4.2.1 Hybrid Model vs. Fully Structured Data Model ........................................................................... 22  
      2.4.2.2 Hybrid Model vs. Semi-Structured Data Model ........................................................................... 22  
   2.5 RELATED WORK ....................................................................................................................................... 22  
   2.5.1 Ozone Motivating Example .................................................................................................................. 23  
   2.5.2 Ozone Design Concept ......................................................................................................................... 24  
   2.5.3 Ozone Implementation .......................................................................................................................... 25  
   2.5.4 Ozone Discussion ................................................................................................................................... 26  
   2.5.5 STORED System ................................................................................................................................... 27  
   2.6 CONCLUSION .......................................................................................................................................... 28  
3 MOTIVATION AND RESEARCH DIRECTIONS ............................................................................................... 28  
   3.1 MOTIVATIONS ........................................................................................................................................... 28  
   3.2 RESEARCH PROBLEMS AND DIRECTIONS ........................................................................................ 30  
   3.3 RESEARCH PROPOSAL OUTLINES ......................................................................................................... 32  
      3.3.1 Research Aims ....................................................................................................................................... 32  
      3.3.2 Research Objectives ............................................................................................................................ 33  
      3.3.3 Methodology ....................................................................................................................................... 34  
   3.4 CONCLUSION .......................................................................................................................................... 34  
4 DATABASE THEORIES AND XML ................................................................................................................ 35
List of Figures

FIGURE 1 SEMI-STRUCTURED DATA AS LABELLED EDGE DIRECTED GRAPH ............ 7
FIGURE 2: STRUDEL ARCHITECTURE .................................................................. 10
FIGURE 3: STRUCTURED DATA FOR THE OZONE EXAMPLE ........................... 24
FIGURE 4: SEMI-STRUCTURED DATA FOR THE OZONE EXAMPLE .................. 24
FIGURE 5: OZONE ARCHITECTURE .................................................................. 26
FIGURE 6: ANSI-SPARC DATABASE ARCHITECTURE ....................................... 41
FIGURE 7: XML TECHNOLOGIES AS DATABASE ARCHITECTURE .................. 42
FIGURE 8: LOGICAL XML DATA MODEL ........................................................... 46
FIGURE 9: PHYSICAL XML DATA MODEL ......................................................... 47
FIGURE 10: GENERAL RESEARCH ISSUES ......................................................... 60
FIGURE 11: SPECIFIC RESEARCH ISSUES ......................................................... 61
FIGURE 12: SAMPLE DATA-CENTRIC XML DOCUMENT .................................... 71
FIGURE 13: GRAPH REPRESENTATION OF DATA-CENTRIC XML DOCUMENT ....... 72
FIGURE 14: SAMPLE DOCUMENT-CENTRIC XML DOCUMENT ......................... 74
FIGURE 15: GRAPH REPRESENTATION OF DOCUMENT-CENTRIC XML DOCUMENT 75
FIGURE 16: SAMPLE HYBRID XML DOCUMENT ............................................... 77
FIGURE 17: GRAPH REPRESENTATION OF HYBRID XML DOCUMENT ............... 78
List of Tables

TABLE 1: DOCUMENT-CENTRIC VS. DATA-CENTRIC XML .............................................. 15
TABLE 2: DIFFERENT XML DATA MODELS PROPOSED BY W3C ............................ 17
TABLE 3: COMPARING XML, OEM, RELATIONAL AND OBJECT-ORIENTED DATA MODELS .......................................................... 20
TABLE 4: OBJECT-ORIENTED ALGEBRAS AND OPERATORS ............................... 40
TABLE 5: RESEARCH TIME TABLE ........................................................................ 63
TABLE 6: DATA MODEL FOR DATA-CENTRIC XML DOCUMENT ......................... 73
TABLE 7: DATA MODEL FOR DOCUMENT-CENTRIC XML DOCUMENT ............... 76
TABLE 8: DATA MODEL FOR HYBRID XML DOCUMENT ........................................ 79
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSI</td>
<td>THE AMERICAN NATIONAL STANDARDS INSTITUTE</td>
</tr>
<tr>
<td>DOM</td>
<td>DOCUMENT OBJECT MODEL</td>
</tr>
<tr>
<td>DTD</td>
<td>DOCUMENT TYPE DECLARATION</td>
</tr>
<tr>
<td>HTML</td>
<td>HYBRID TEXT MARKUP LANGUAGE</td>
</tr>
<tr>
<td>NXD</td>
<td>NATIVE XML DATABASE</td>
</tr>
<tr>
<td>OEM</td>
<td>OBJECT EXCHANGE MODEL</td>
</tr>
<tr>
<td>PDA</td>
<td>PERSONAL DIGITAL ASSISTANCE</td>
</tr>
<tr>
<td>SPARC</td>
<td>STANDARDS PLANNING AND REQUIREMENTS COMMITTEE</td>
</tr>
<tr>
<td>WWW</td>
<td>WORLD WIDE WEB</td>
</tr>
<tr>
<td>XLINK</td>
<td>XML LINKING LANGUAGE</td>
</tr>
<tr>
<td>XML</td>
<td>EXTENSIBLE MARKUP LANGUAGE</td>
</tr>
<tr>
<td>XPATH</td>
<td>XML PATH LANGUAGE</td>
</tr>
<tr>
<td>XPointer</td>
<td>XML POINTER LANGUAGE</td>
</tr>
<tr>
<td>XSchema</td>
<td>XML SCHEMA</td>
</tr>
<tr>
<td>XSL</td>
<td>THE EXTENSIBLE STYLESHEET LANGUAGE</td>
</tr>
<tr>
<td>XHTML</td>
<td>THE EXTENSIBLE HYPER TEXT MARKUP LANGUAGE</td>
</tr>
</tbody>
</table>
1. Introduction

The Web is well established as a global infrastructure for storing and manipulating large collections of structured and semi-structured data. Therefore it provides an alternative to - and complements - structured data resources stored in conventional database systems, such as relational and object-oriented databases. Among Web technologies, the eXtensible Markup Language (XML [Online]) - a subset version of the Standard Generalized Markup Language (SGML [Online]) – is gaining rapid acceptance as the de facto standard for data exchange on the Internet and the Intranet as well as a technology to store structured, semi-structured and un-structured data on the Web. XML can be defined as a markup language that uses a standard plain text format and contains data in user-defined tags presented in a hierarchical way (tree-like structure) (W3Schools [Online]).

XML has a particular tension with the mature database theories. This arises from the similarity between using databases to store and query structured data and using XML to store and query semi-structured data. However, there are basic differences between defining data using markup languages and the more prescriptive and structured techniques used in mature data models such as relational data model. There are barriers that make the direct application of relational query optimisation techniques to XML document collections (or XML databases) a problem (Suciu (2001) and Vianu (2001)). In particular, the lack of pre-knowledge of the structure of the XML documents collection and the lack of a structure common to all XML documents within any specific collection. This problem is the focus of the research.

The above barriers can be eliminated for a class of XML document collections for which there is a common pre-determined structural component in addition to the semi-structured content. This class of documents can be defined as “Partially Structured Data”. The research will focus on that hybrid model. This class of XML documents collection is likely to include a significant number of applications, for example, health care system and multi-product selling system.
This preliminary study reviews the literature on theories and technologies relevant to the storing, querying and retrieving of semi-structured data, and XML documents in particular. It focuses on the notion of an XML document collection as a database, with the aim of investigating how the mature technologies that have been developed for relational databases can be adapted for the management of XML-encoded semi-structured data stored on the Web.

The research argues that to take the full power of XML databases there must be a solid formal data model and query algebra analogues to the relational data model (Codd, E. F. (1970)) or object-oriented data model (Cattell et al. (2000)) and their related query algebra. The aim is to fully utilise this power which defines the semantic of XML queries and opens the way toward query transformation and therefore optimisation.

Therefore, the research seeks to define an enhanced data model and query algebra for XML databases. Specifically, this aim is to extend existing structured approaches. This approach will have two main advantages. Firstly, it will facilitate the combination of structured and semi-structured data (partially structured data), so as to better support classes of application where there is a fixed formal framework for data, but also an ad hoc component. Secondly, this hybrid approach is likely to have performance advantages over pure semi-structured database approaches as well as information retrieval approaches by utilising existing optimisation techniques developed for conventional databases to deal with the structured part of the data and not treating data as if it is totally semi-structured or un-structured.

The report concludes with a proposal for further and more focused research in XML databases. Specifically, by enhancing a formal data model and query algebra for XML to be applicable for partially structured XML documents and then seek to adapt an established relational database query optimisation technique(s) for querying partially structured XML documents.

The overall structure of this report following this introductory section can be summarised as follows:
Section 2 titled “Background and basic definitions”, presents semi-structured data, XML and its related technologies, the tension between the two technologies (semi-structured data and XML) and finally partially structured data. Section 3 titled “Motivations and Research Directions”, is the core section of this report; it identifies the general motivation behind the research and gives details of the proposed research outline. It states the research problem, aims, questions and objectives. It describes the methodology which will be used to achieve the research objectives. Section 4 titled “Database Theories and XML”, discusses in more detail the relation between database theories and XML. It shows the evaluation of database data model and algebra, starting from the relational data model and relational algebra towards the semi-structured data model and algebra. Furthermore, it discusses the database system architecture (The ANSI-SPARC Model) and maps XML technologies into that model. It describes logical and physical XML data model as well as native XML databases. Finally, it shows the requirements for querying web data in general and XML in particular. Section 5 titled “Formal XML Data Model and Query Algebra”, presents a formal data model and query algebra for XML. Section 6 titled “Research Proposal and Conclusions”, concludes this report by discussing the research proposal and the time table toward achieving the research objectives.

2. Background and Basic Definitions

The background and basic definitions for semi-structured data and XML are presented in this section. The research is based on these technologies, so defining them is the first step toward conducting the research. The tension between the two technologies and the partially structured data are also discussed. The main notion that arises is the overlap between these two technologies (semi-structured data and XML) and their relation to each other.

The section is organised as follows. Section 2.1 defines semi-structured data, section 2.1.1 discusses its characteristics, section 2.1.2 discusses the motivations to study it, section 2.1.3 defines its data model and finally section 2.1.4 describes its research issues. In section 2.2, XML - the
emerging de facto standard for encoding semi-structured data on the Web - and its related technologies are described. Section 2.3 discusses the relationship between the semi-structured data and XML followed by section 2.4 which shows the bases for partially structured data (the hybrid structured and semi-structured approach). Section 2.5 discusses related work which includes the description of two systems namely, Ozone (Lahiri et al. 1999) and STORED (Deutsch et al. 1999). Finally, section 2.6 draws a conclusion to this section.

2.1 Semi-Structured Data

Here, the scene for the subsequent study of XML is set, by explaining the characteristics of semi-structured data and the motivation for research in this area.

Data is stored on the Internet, mainly unregulated and/or un-administrated, and is typically in the form of documents, to be read by human. Consequently, the data is often characterised by ad hoc irregularity, incompleteness and complicated structures which can be changed rapidly and unpredictably. This form of data is usually referred to as semi-structured data. (see Abiteboul (1997), Buneman (1997), Suciu (1998) for tutorials on semi-structured data, in general, and see Abiteboul et al. (2000) for a tutorial on semi-structured Web data in particular).

2.1.1 Semi-Structured Data Characteristics

The characteristics of semi-Structured data as described by Abiteboul (1997) are as follows:

- *The structure is irregular:* for example a student name can be stored as a string or as a record (as a first and last name string)

- *The structure is implicit:* if it exists, the structure is embedded inside the document itself and there is no attached external structure. (There are some attempts such as Data Guide in Lore (Goldman and Widom (1997)) that discover the structure from the data itself and store it in a structure called data guide)
• *The structure is partial*: some portions of the data, such as bitmaps, may not have any formal structure, while other portions may.

• *Indicative structure vs. constraining structure*: since the structure does not provide a solid constraint over the semi-structure data, the structure here is indicative rather than constraining (not like the relational model).

• *A priori schema vs. a posterior data guide*: unlike traditional structure data sources in which the structure is designed before the data, in semi-structured data the structure is discovered from the data itself.

• *The schema can be very large*: large irregularity of the data will require a large description of the data structure.

• *The schema may be ignored*: in some queries of a discovery nature, the browsing or information retrieval search is used instead.

• *The schema may evolve rapidly*: because of the nature of the data, the schema can be changed much more rapidly in contrast to the traditional databases.

• *The type data elements are eclectic*: the data types are not precise.

• *The distinction between schema and data is blurred*: there are no border lines between the schema and the data, this due to the schema being embedded inside the data and not putting a hard constraint on the data.

An important consequence of the above characteristics is that, because of the lack of a predictable pre-defined structure for data within a collection, the description of the data, i.e., its schema, is specific to each document or record, and is therefore often contained within the data itself. Thus semi-structured data is “self-describing” or “schema-less”. Also, semi-structured data cannot be directly modelled in a traditional data model, such as the relational data model (Codd 1970); since these require data collections to rigidly conform to a predetermined structure.
### 2.1.2 Semi-Structured Data Motivations

Semi-structured data has become a focus for research. Motivations for this are listed in Buneman (1997) and Suciu (1998), and can be summarised as follows:

- A huge amount of data is stored outside structured database systems in different forms such as structured documents (for example SGML, HTML and XML), legacy systems, scientific data that is stored in a very complex structure and file systems. Such data does not fit into the highly structured data models of database technology, but there is a need for it to be treated as a database.

- Semi-structured data is investigated to facilitate data integration. Although some source data may be highly structured and stored in a database system, other sources may lack this structure and must therefore be stored in an unstructured manner, such as a text format. But because there is a need to integrate these different heterogeneous data sources, a need arises for a highly flexible data format to facilitate the integration of both structured and semi-structured data.

- Techniques are also required for viewing structured data as a semi-structured data for browsing purposes and sometimes ignoring the schema, if it exists, for this purpose.

- The development of XML and XML technologies must be grounded on an understanding of semi-structured Data (see section 2.3).

To summarise the above motivations, a huge amount of semi-structured data already exists that needs to be managed and utilised using database-like facilities. In addition, there is a requirement to integrate and navigate semi-structured data resources. These motivations therefore pose the question: *To what extent can these problems be solved by adapting the maturing theories and technologies of database?*

That question is the focus of the study in the remainder of this report. The next sections show that defining a formal data model and query algebra is a necessary first step towards addressing that question.
2.1.3 Semi-Structured Data Models

Having discussed the characteristics and importance of semi-structured data, this section discusses the ways in which semi-structured data can be modelled.

Semi-Structured data comprises self-explanatory data, which means that associated with data values are labels that represent the semantics of the respective pieces of data. Further, data values are associated with each other by an embedded hierarchy which represents the natural relationship between data. Therefore semi-structured data has a natural representation as a rooted directed graph with labelled edges. In the graph, the leaf nodes represent the data values, the edges represent the embedded hierarchy and the relation between different data, and the edge labels denote the semantics of the data represented by the child node. Figure 1 shows an example of a semi-structured data represented as an example of label-edge graph.

The above graphical representation of semi-structured data, with minor variations, is used widely in research into semi-structured data (such as Lore (McHugh et al. 1997)). One example model, based upon the above labelled graph representation, is OEM (Papakonstantinou et al. (1995). OEM has been
widely adopted in semi-structured data research. As Suciu (1998) observes, “The de facto standard data model for the semi-structured data is the Object Exchange Model (OEM)” (Papakonstantinou et al. (1995)). OEM was originally designed for the Tsimmis data integration project. It was to be used as a very flexible data model to facilitate the data integration between different applications. It was subsequently adapted to provide the theoretical basis for the lore project (McHugh et al (1997)).

In OEM the graph-structure of semi-structured data is represented as nested quadruple structures, comprising the following elements:

- **Label**: which describes what the object represents
- **Type**: the data type of the object’s value, which can be either an atomic type (such as integer or string) or type set.
- **Value**: the value of the object. It can also be an object-Id value where the type is object-Id; this means that links can be provided in the model.
- **Object-Id**: a unique identifier for the object or null

OEM is a logical data model; since it does not specify how data is stored physically. OEM is much simpler than Object-Oriented data model since it supports only object nesting and object identifying. Other features such as classes, methods and inheritance are omitted. Also, labels are used instead of schema in a similar object oriented model (Papakonstantinou et al. 1995) and this makes it simple enough to cope with the frequent changes in semi-structured data.

In an instance of OEM, each stored value is given a label or tag to describe its meaning. Also, the data can be given structure by nesting tagged data values. OEM does not include a schema, separate from the tagged data. Therefore, OEM can be considered as a schema-less and self-describing data model.

To describe an atomic object, such as course name, one could use (“Course Name” (label), “String” (Data Type), “Database Systems” (Value)) and to describe a complex object such as student data, the description might
be ("Student Data" (Label), "Set" (Data Type), {Student Name, Student Address} (Values)) and then ("Student Name" (label), "String" (Data Type), "John Smith" (Value)) and ("Student Name" (label), "String" (Data Type), "Sheffield, UK" (Value)). So, the object itself will consist of the three values (label, data type, value) and object Id, which uniquely identifies the object.

OEM is sufficiently expressive to represent highly structured data such as relational data model and object-oriented data model or semi-structured data or even unstructured data (Abiteboul et al. (2000)). Also, it can represent, in a natural way any missing attributes, duplicate attributes with the same name, different types for the same attribute in different objects and finally, it can represent the same attribute in one object as an atomic one while in another as a record such as a name of a student as one field or as a record with first and last name fields.

OEM provides a complete description of semi-structured data and XML documents. However, if this model is adopted in investigating partially structured data, the structured part of the data will be ignored and treated as totally semi-structured. So, this model is inappropriate for direct use in partially-structured data domain.

2.1.4 Semi-Structured Data Research Issues

Research into semi-structured data complements previous research in conventional databases. Moreover, the invention of the World Wide Web opens a gate for a research related to both database and the web based on a semi-structured data model. The research varies from management and administration issues (such as web site management and general purpose management of semi-structured data), data representation and manipulation (such as web site management, general purpose management of semi-structured data, data conversion, schema specification and schema extraction), performance issues (such as optimisations and formal aspects and indexing), and issues relating to the formal foundations (such as optimisations and formal aspects). This section discusses these research areas (Suciu (1998)):
• **General purpose management of Semi-structured Data** such as Lore system (McHuge et al. (1997) and Abiteboul et al. (1997)) which is a complete database management system for semi-structured data, but it not appropriate for partially structured data because it deals with the data as if it’s totally semi-structured and ignoring the structured part of it.

• **Web Site Management**: Fernandez et al. (1998) developed a system for web site management by separating
  
  o The management of the site’s data
  o The creation and management of the site’s structure
  o The visual representation of the site’s pages

![Figure 2: Strudel Architecture](Fernandez et al. 1998)
STRUDEL first integrates data from any number of heterogeneous data sources into a semi-structured repository. Then it applies a site-definition query to declaratively define the Web site’s structure with the result called site graph which represents both site contents and structure and finally presents the visual presentation in Strudel’s HTML-template language (Figure 2).

- **Data Conversion:** Cluet et al. (1998) proposed a YAT system for data conversion among heterogeneous data sources based on a middleware model which is named tree with ordered and labelled node (that is similar to Semi-structured Data model). YATL (YAT Language), the conversion language, is declarative, rule-based and features enhancement pattern matching customisation mechanism.

- **Schema Specification:** Buneman et al. (1997) proposed a new schema for semi-structured data by presenting both data and schema as edge-labelled graphs. They then studied the analogy between graph database and graph schema and showed how schema can improve the optimisation and decomposition of queries.

- **Schema Extraction:** Another example is the Data Guides (Goldman and Widom 1997), which is one of the novel features of Lore. They are concise and accurate structural summaries of data stored and are created from the data itself. They are used in browsing data, formulating queries, storing information such as statistics and enabling query optimisation. The paper by Buneman et al. (1997), which was previously mentioned in Schema Specification, can be considered also as a schema extraction.

- **Optimisations and Formal Aspects:** Abiteboul and Vianu (1997) introduced a web model as an infinite semi-structured set of objects. They studied declarative query languages (such as first-order logic, Datalog and Datalog with negation) based on that model.

- **Indexing:** McHugh et al. (1998) studied the indexing of semi-structured data in the Lore DBMS and they developed the following indices:
o **Vindex**: a value index over atomic values (such as integer, string and real) based on the type coercible which can be built selectively.

o **Tindex**: a text index locates string atomic values containing specific words or groups of words which can be built selectively.

o **Lindex**: a link index locates the parent of a specific object.

o **Pindex**: a path index for fast access to all objects reachable via given labelled path.

The above research addresses a number of general areas. Each of these areas has potential for valuable research. But, the work proposed here will focus on two of these areas – performance issues and formal foundations. Specifically, the research will investigate these issues with respect to XML-encoded semi-structured data query processing. Accordingly, XML is discussed in the following section.

### 2.2 Extensible Markup Language (XML)

The Extensible Markup Language’s (XML [Online]) roots belong in the document community not to the database community (Vianu 2001). XML as a standard for electronic applications, like electronic commerce (E-Commerce), electronic learning (E-Learning), etc., was a subset version of the Standard Generalized Markup Language (SGML [Online]) modified to enable it to deal with the Internet. SGML is the widely used international standard for text processing defined by the International Organization for Standardization. So, XML is a document format and not a data model (Widom 1999), but there is a great tension between XML and database theories in general and semi-structured data in particular. This section defines XML, which is then followed by a comparison between XML and semi-structured data in the next section.

XML is a markup language in a standard plain text format. It contains structured or semi-structured data in verbose user-defined tags presented in a hierarchical way (tree-like structure) (W3Schools [Online]). The main advantage of XML is the separation of the data from how this data is formatted and the internationalisation features of it since it uses Unicode. So,
users can display XML data in any electronic device such as computers, mobile phones and personal digital assistant (PDA). Moreover, users can make very complex queries from multiple XML data sources in comparison to the simple text searches available in Hyper Text Markup Language (HTML [Online]) and simple, fixed and parameterised queries in traditional databases (Widom (1999)).

XML is not a replacement for HTML, since its goal is to describe data vs. HTML, whose goal is to format data. Also, XML is not a replacement for traditional databases, although XML structures data and give syntax and meaning to the data. This is sort of database that stores data in a structured way. Also using different technologies XML data can be queried, transformed, etc using the XML programming interface. But XML - in Bourret’s (2002) points of view - lacks a lot of things available in database, like efficient data storage, indexes, data redundancy, security, transactions, data integrity, multi-user access, triggers and queries across multiple documents. Mertz (2001) compared the relationship between XML and the different data models (hierarchical, relational and object-oriented models). He argues that XML is something of a hybrid between the hierarchal data model (because of the tree structure of XML document) and between object models (as a data model since XML consists of nodes and these nodes can contain heterogeneous data). Although XML can represent any relational data model, it is “less natural” compared with the relational data model in the way that the relational model represents data in flatted tabular form vs. hierarchal representation of XML document.

XML can be seeing from different points of view. As described by Ray (2001 p.2), it is a protocol that allows the containment and management of data, it is a family of technologies that can store, format and filter data, and finally it is a philosophy that seeks maximum usefulness and flexibility for data in a structure form. XML has become in a very short period¹ the base for data presentation and data exchange between heterogeneous applications on the Internet and Intranet and the keyword for nowadays integration between different businesses processes. XML documents are used either as a

¹ The first XML standard was in 1998.
container to store semi-structured data or a media to exchange data between heterogeneous application. The World Wide Web Consortium (W3C [Online]) maintains the XML as a standard as an essential web technology.

The widely acknowledged advantages of XML include:

- Its simple format,
- The internationalisation capability (use of Unicode),
- Platform independence,
- Extensibility,
- Human readability as well as machine readability,
- The large investment in XML applications that already exists.

Therefore, these advantages make XML particularly appropriate as a way to store web semi-structured data.

XML document collections can be categorized into two main types (Bourret (2002)), Data-Centric and Document-Centric documents. The former uses XML as a method of data transport and is mainly designed for machine representation of highly structured data, such as product lists, inventories and student courses. The second type uses XML to store documents with less regular or semi-structure data and is mainly designed for human consumption, such as book contents, emails and advertisements. Some documents can be described as a hybrid between them; for example, a product contains some properties such as price and product name with another less structured data such as product specifications. Table 1 summarize the differences between data-centric and document-centric XML document (Kim et al. (2002))

A complementary part of XML file is the technology related to it. The Document Type Definition (DTD) and XML Schema are used to define the structure of an XML file, and any valid XML file must be validated against DTD or XML Schema file definition. Extensible Stylesheet Language can transform XML to another format like HTML and also can select a part of it and sort it. Document Object Model (DOM) is the programming interface to an XML file, used to access and manipulate an XML file.
Instead of representing XML as its natural structure (tree structure), we must seek more intuitive and meaningful presentations of XML. For example, XSL or any other transformation means must be used to convert the XML data to a better format to present it (for example to convert it to HTML, XHTML …). XML provides a better way to search web documents compared with the text search and Information retrieval techniques currently used to search HTML Web documents.

### 2.2.1 Proposed Data Model by W3C

The World Wide Web Consortium (W3C Online) proposed different models for XML document. These tree data models describe the document structure. Table 2 (Salminen and Topma (2001) compares between the XML Information set, Xpath data model, Document Object Model (DOM) and Xquery 1.0 and Xpath 2.0 data model.

<table>
<thead>
<tr>
<th>Purpose</th>
<th>XML Info Set</th>
<th>Xpath 1.0</th>
<th>Dom 1.0 Level 2.0</th>
<th>XQuery 1.0 and XPath 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>To refer to the information stored in the XML document</td>
<td>To address parts in XML document (works as a query language but for one document only)</td>
<td>To access and update the contents and the structure of documents dynamically</td>
<td>To define precisely the information contained in the input to an XSLT / XQuery processor.</td>
</tr>
<tr>
<td>Purpose</td>
<td>XML Info Set</td>
<td>XPath 1.0</td>
<td>DOM 1.0 Level 2.0</td>
<td>XQuery 1.0 and XPath 2.0</td>
</tr>
<tr>
<td>---------</td>
<td>--------------</td>
<td>-----------</td>
<td>-------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>What is modelled?</td>
<td>XML</td>
<td>XML</td>
<td>XML or HTML</td>
<td>Collection of XML or parts</td>
</tr>
<tr>
<td>No of Nodes</td>
<td>11</td>
<td>7</td>
<td>12</td>
<td>8</td>
</tr>
</tbody>
</table>
| Node Type | 1. Document  
2. Element  
3. Attribute  
4. Processing Instruction  
5. Unexpanded Entity Reference  
6. Character  
7. Comment  
8. The Document Type Declaration  
9. Unparsed Entity  
10. Notation  
11. Namespace | 1. Root  
2. Element  
3. Text  
4. Attribute  
5. Namespace  
6. Processing instruction  
7. Comment | 1. Document  
2. Document Fragment  
3. Document Type  
4. Entity  
5. Notation  
6. Element  
7. Attr (Attribute)  
8. Processing Instruction  
9. Comment  
10. Entity Reference  
11. CDATASection  
12. Text | 1. Document  
2. Element  
3. Attribute  
4. Text  
5. Namespace  
6. Processing instruction  
7. Comment  
8. Reference |
<table>
<thead>
<tr>
<th>Purpose</th>
<th>XML Info Set</th>
<th>Xpath 1.0</th>
<th>Dom 1.0 Level 2.0</th>
<th>XQuery 1.0 and XPath 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTD validity required</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes If DTD exist, If not then not required</td>
</tr>
</tbody>
</table>

Table 2: Different XML Data Models Proposed by W3C (Salminen and Topma (2001))

The previous table shows that there is no clear well defined formal data model by the W3C for XML document. Each model tries to solve a specific problem and to satisfy a specific requirement. Defining a formal data model for an XML document is a prerequisite for query transformation and optimisation, so defining that model is discussed in more details in section 5.

2.3 Semi-Structured Data and XML

This section compares XML, the semi-structured data model (Object Exchange Model) and the traditional database data models (relational and object-oriented data models). Despite the difference between XML and OEM, it is more natural to model XML as OEM rather than relational or object-oriented data model. The following comparison by Abiteboul (1997, Abiteboul et al. 2000) defines the difference between them

- **Natural**: XML is a document markup format; it is not a data model. While OEM is a data model.

- **Origin**: The XML is a subset version of the Standard Generalized Markup Language (SGML [Online]) so it can be enabled to deal with the Internet. This means that the root of the XML comes from the document community. It differs from OEM, which was originally inspired by the database community to be used for data integration between heterogeneous data sources.

- **Schema**: XML may or may not have associated schema, the “well formatted” condition of the XML document that the tags must be nested
correctly, while the “Valid XML” is a document that is well formatted and conforms to a DTD or XML Schema. While the semi-structured data’s schema is discovered from the data itself (self-describing), although some attempts such as Data Guide of Lore (Goldman and Widom 1997) discovers the schema from the semi-structured data. In both two cases the schema may be ignored for browsing purposes. This contradicts both the relational and object-oriented data models, where the schema must be fixed, fully defined prior to data entry.

- **Structure**: both XML and OEM have irregular, implicit and may be partial structure, the best way to model XML is as node-edge graph, while the OEM is as a labelled-edge graph. This difference is minor compared with the regular, explicit and highly structured nature of relational and object-oriented data models.

- **Order**: a major difference between XML as a document format that preserves the order and other database data models which by definition ignore the order. For example the relational model is based on the set theory and by definition the set is not ordered. The object-oriented model re-introduces the order of the data into its model by avoiding sets and using other types such as list and arrays.

- **Mixing Elements and Text**: this is purely a document feature in the XML documents that inside a text there may be another element. For example `<address> West Street <city> Sheffield </city> </address>`. There is no analogy to these features in any of the other database data models.

- **Constructs**: another feature of the XML document is that it may contain constructs such as comments, processing instruction (PI) that gives some instruction regarding the XML document to the receiving application, the start line of the XML document (`<?xml version="1.0" encoding="ISO-8859-1"?>`), CDATA which contains text which will be ignored by the parser and not (`<![CDATA[any text contains for example <, >, /, … or any other special character]]>`), or Escape Characters such as “<” which can not be used inside elements and are replaced by
entity references (&lt). There are no analogy for these constructs in OEM and the relational data model. For the object-oriented data model some analogy exists by using of the methods (compared with processing instructions)

- Reference: XML elements can be referenced by defining “Id” attribute with an element and then using IDRef and IDRefs in another element to reference this element. OEM has an analogy by creating an edge from one node to another node. The foreign key in the relational model and OID and relationship in object-oriented model can be seen as some sort of referencing although the reference in XML and OEM is much richer than in database data models.

The following table summarise the comparison between XML, OEM, relational and object-oriented Data models.

<table>
<thead>
<tr>
<th>Feature</th>
<th>XML</th>
<th>OEM</th>
<th>Relational</th>
<th>Object-Oriented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin</td>
<td>Document Community</td>
<td>DB Community</td>
<td>DB Community</td>
<td>DB Community</td>
</tr>
<tr>
<td>Schema</td>
<td>Optional</td>
<td>Post Data</td>
<td>Fixed, defined prior to data</td>
<td>Fixed, defined prior to data</td>
</tr>
<tr>
<td></td>
<td>May be ignored</td>
<td>May be ignored</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structure</td>
<td>Irregular, Implicit, may be partial</td>
<td>Irregular, Implicit, may be partial</td>
<td>Regular, Explicit,</td>
<td>Regular, Explicit,</td>
</tr>
<tr>
<td></td>
<td>Node-Edge Graph</td>
<td>Labelled-Edge Graph</td>
<td>Tables</td>
<td>Objects</td>
</tr>
<tr>
<td>Order</td>
<td>Ordered</td>
<td>Not Ordered</td>
<td>Not Ordered</td>
<td>Using lists and Arrays</td>
</tr>
<tr>
<td>Mixing Elements and Text</td>
<td>Available</td>
<td>Not Available</td>
<td>Not Available</td>
<td>Not Available</td>
</tr>
</tbody>
</table>
Feature | XML | OEM | Relational | Object-Oriented
--- | --- | --- | --- | ---
Constructs | Comments, Processing Instruction, Start with an optional construct (<?xml version ="1.0"?>), CDATA, Entities (&lt = <) | No Analogy | No Analogy | Using methods analogy with PI
Reference | XML Reference (IDRef, IDRefs) | No Analogy | Foreign Keys | OID and Relationships
Nesting | Available | Available | N/A | Available

Table 3: Comparing XML, OEM, Relational and Object-Oriented Data Models

As seen from this comparison, the XML is more naturally related to OEM model rather than the conventional database data models.

2.4 Partially Structured Data: A Hybrid Structured and Semi-Structured Approach

In the previous three subsections, after describing both semi-structured data and XML as well as comparing them, the question that arises is how can we combine the fully structured data with the semi-structured data and XML into one data model to allow a hybrid approach appropriate to a wide class of application either online on the Internet or offline in local Intranet? How can we take the advantage of both the worlds? It is clear that each of the fully structured data models, relational and object oriented, has many advantages such as the efficient way the data stored and queried. On the other hand, semi-structured data and XML are flexible enough to model any real world data and navigate it in an easy manner without the need for a fixed and solid...
data structure. So, this section describes the combination of these concepts, which is the basis of the proposed research area.

This section focuses on the motivation to study the hybrid model and the natural of the problem and the proposed solution by others in the related work section. The design of the proposed enhanced data model for XML database, which is the core of the research proposal, will be discussed later.

The following subsection discusses the motivation behind this hybrid model followed by advantage of this model over the purely structured and semi-structured data models. The related work section describes a very similar approach, which is very much related to this research scope and finally a general discussion and conclusions.

2.4.1 Motivation for Hybrid Approach

One of the great disadvantages of relational database is its simple data structure (a table or a relation) that can not represent all the real complex world data such as scientific data – although this is sometimes considered as advantage for its simplicity and ease of understanding. The third generation of database data models which is the emergence of object oriented data model and object relational data model take one step forward in this direction, but this was not enough. Since much more flexibility is needed to represent a class of data that can be changed rapidly and unexpectedly, semi-structured data. On the other hand, this flexibility comes with the cost of less than optimal ways to store and query these classes of data. So, a hybrid model is needed to take the good features for both the two models. This means that this model will be a container of both structured and semi-structured data allowing links from both sorts of data to each other, preserving the structured parts of the data, storing and querying it in an optimised way as if it were purely structured data taking advantage of decades of research on the optimisation techniques for structured data as well as adding the capability to store the semi-structured part of data as efficient as possible allowing the of this data without full knowledge of its structure.

To conclude, the proposed hybrid system accepts both structured and semi-structured data, stores it as its natural data model, taking the of both of
them but it also providing links between them so users can query and navigate both of them simultaneously.

2.4.2 Advantage of Hybrid Model

The Hybrid model tries to take the advantage of both structured and semi-structured data models. These advantages are discussed compared with purely structured and semi-structured in the next two subsections. It can be considered that the advantage of one model is the disadvantage of the other model.

2.4.2.1 Hybrid Model vs. Fully Structured Data Model

The fully structured data model can not represent semi-structured data in an easy manner. This yields a complex structure that the user must be aware of in order to query the data. Also, any small change of the data structure - which easily can happened because of the nature of the semi-structured data - could yield a big evolution of the schema. And structured data does not provide the navigational search approach to its data.

2.4.2.2 Hybrid Model vs. Semi-Structured Data Model

Purely semi-structured data model such as Lore (McHugh et al. (1997)) does not take into consideration the structure part of the data if it exists. This means that it deals with this part as if it were semi-structured without taking advantage of its strong structure and typing toward optimisation either for storing or querying. Also, the semi-structured approach alone does not take into consideration the huge amount of research already developed in the highly structured data models.

2.5 Related Work

A hybrid system Ozone (Lahiri et al. (1999)) is described in this section in details. It integrates structured and semi-structured data by extending the structured object database model ODMG as well as the Object Query Language OQL (named OQL₅) to be able to store and query semi-structured data, which is based on the Object Exchange Model OEM (Papakonstantinou et al. (1995)) model and Lorel Language (Abiteboul et al. (1997)).
The idea behind the Ozone system is very simple; it extended the ODMG model by a new built-in class type OEM. And then allows crossover points from the structured part of the data toward the semi-structured part and vice versa. And it extends the semantics of OQL to cover this extension by introducing new semantic operations to OQL that allow this crossover. The syntactics of the OQL$^S$ are identical to OQL and the semantics is also identical in the case of a query on the only fully structured part of the database. The following subsections explore Ozone in more details by describing its motivating example, and then its design concepts followed by implementation model.

2.5.1 Ozone Motivating Example

Ozone uses an example of a simplified on-line broker that sells products on behalf of different companies. This shows a very good example of a hybrid data since some information such as products number and name and companies’ details are fully structured while product info and reviews about products are semi-structured due to the fact that each type of product can have details which are specific to it and not applicable to other product types. For example, computers may have processor attribute while monitors may have screen size attribute. Figure 3 shows the design of the structure part of the data, while figure 4 shows how could be the semi-structured part for the product info attribute. This reveals how a link could be made from the structure part to the semi-structured though the product info attribute and also the reverse through the ‘competing’ attribute.
Figure 3: Structured Data for the Ozone Example  
(Lahiri et al. 1999)

Figure 4: Semi-Structured Data for the Ozone Example  
(Lahiri et al. 1999)

And finally, the semi-structured part is flexible that it can show a competing product such as “System 12” which is not part of the structured database as a complex OEM object.

2.5.2 Ozone Design Concept

The previous example shows how both structured and semi-structured data can be linked with each other. In this section, some design concepts of the Ozone system are discussed.
As described before, adding the new OEM class allows the storing of semi-structured data into the ODMG model. This OEM class can be classified into two categories, OEMcomplex and OEMatomic representing complex and atomic OEM objects respectively. OEMcomplex is a collection of (label (as string), value (as OEM object)) pairs. And due to the XML represents an ordered graph; there are two subclasses for this collection as OEMcomplexset for the unordered collection and OEMcomplexlist for the ordered one.

OEMatomic represents atomic values, which are also represented by subclasses. For example OEM_integer encapsulate the type integer represents OEM (integer) objects. And so on for the other atomic types such as OEM_real, OEM_string, and OEM_Boolean … etc. OEM_object can be used to reference any other class inside the database and to be the crossover point from semi-structured to structured part since the object class is a super class of all the classes inside the database, but for performance reasons, and instead of making it general which leads to determining the type of the class at run time, Ozone defines an OEM class for each class in the database to be used as a reference to that class such as OEM_product and OEM_company. And finally for the atomic OEM objects encapsulating non-atomic literals values (tuple, set, list…etc), which could be represented by equivalent complex objects. But also for performance reasons, Ozone defines additional OEM subclasses encapsulating the types of non-atomic literal types that most probably are used in queries.

2.5.3 Ozone Implementation

Figure 5 describes the architecture of the Ozone system. The class manager will be extended by a predefined group of classes for the OEM class type as well as some specific problem classes such as OEM_product. Prior to running, an OQL$^S$ query goes through pre-processor, to transform it to the intermediate OQL query which resolves the crossover points if any exist. Then the results go though postprocessor before appearing to the user. Also there is an Ozone Loader to directly load bulk of data into the database.
2.5.4 Ozone Discussion

Ozone is a very good example of combining of structured and semi-structured data. As stated by Lahiri et al. (1999) “we know of no previous research that has explored the integration of structure and semi-structured data as exhibited by Ozone”. This means that a lot of research is needed in this direction.

Some of the future extension of the Ozone have been described in their paper, these include:

Ozone performance: to study further the optimisation of this hybrid system

Object-relational Ozone: to define a similar semi-structured extension to the object-relational data model

Applications: to study further applications that can take advantage of the hybrid approach.

Other points that can arise from the previous sections are:
There is no formal mathematical model and query algebra defined for this hybrid approach. Such a model is essential to defining the semantics of the queries which will open the gate for query transformation and optimisation.

Ozone also is very specific about the problem that it is applied to. For example, it defines a class called OEM_product, which is only applicable for this problem. So is there is a general way so this hybrid model can act as a higher abstraction level for any application? And is there an algorithm that can automatically transfer the user requirements into a complete system?

What about the extension of pure relational model itself to deal with hybrid model?

These directions and questions open the way toward the more detailed discussion of the research aim and objective in section 3.

2.5.5 STORED System

The idea behind the STORED (Semi-structured TO Relational Data) System (Deutsch et al. 1999) is to define a declarative query language that specify storage mapping from semi-structured data model (an ordered version of the OEM data model (Papakonstantinou et al. 1995)) to a relational data model and an overflow graph. Using data mining, the mapping tries to exploit any patterns in the data and then create the equivalent relational structure to store this data, and because that part of the data can not be fitted into the relational structure, it is stored in an overflow graph. When a query is executed, it is re-written into a query over the relational store. Any update over the semi-structured data is also re-written into update over the relational store. The two possible application of this system are:

- To store and manage efficiently existing semi-structured data sources.
- To convert relational sources into a semi-structured format such as XML.

Although this system’s initial mapping of semi-structured data into the relational data model may yield a good result, with time, the performance will degrade, because of changes in both the data and the query mix; this will
require a new mapping to be generated. The data model used is not clear since it is a mixture of both the relational model and an overflow graph.

### 2.6 Conclusion

Discussing the basic definitions of semi-structured data, XML and partially structured data shows the great overlap between these technologies and how they can be related to each other. It provides the ground work for the next step in the research which is described in the next section. The next section shows the motivation behind the research and discusses the research problems and directions followed by an outline of the research proposal.

### 3 Motivation and Research Directions

This section presents the motivation, the specific research problem, the research design and the methodology to be addressed by the research. This proposal addresses issues that emerged in the previous sections, in which the tension between technologies and models for XML document collections and databases were identified.

This section is structured as follows. Section 3.1 discusses the motivation for the research proposal. Section 3.2 presents an outline for the research problems and directions. Section 3.3 discusses in detail the proposed research outline; thus, divided into 3 subsections. Section 3.3.1 discusses the aims of the research, section 3.3.2 discusses the objectives of the research and finally section 3.3.3 describes the methodology which is used to achieve the research objectives. And finally the conclusion and the way ahead are discussed in section 3.4.

#### 3.1 Motivations

The motivation behind the research proposal is rooted in issues relating to semi-structured data and XML document collections in particular. This section revisits these two areas and introduces the notion of partially structured data and associated data management problems. The latter are the focus of the proposed research.
As discussed in section 2.2, XML is highly important nowadays. A number of widely acknowledged advantages for XML were discussed, which include:

- Its simple format,
- The internationalisation capability (use of Unicode),
- Platform independence,
- Extensibility,
- Human readability as well as machine readability,
- The large investment in XML applications that already exists.

Section 2.1 shows the huge amount of semi-structured data already existing and the need for this data to be treated like databases, and to be integrated and navigated. The motivations for studying semi-structured data have been discussed in more detail in that section (Buneman (1997) and Suciu (1998)), but it can be summarized as:

- Modelling the data on the web as a semi-structured data.
- Modelling the huge amount of data that is stored outside structured database systems in different forms such as structured documents, legacy systems and scientific data.
- Facilitating data integration.
- The ability to view structured data as semi-structured data for browsing purposes.

So, based on these motivations, and adding to them the many applications that need to deal with large numbers of XML documents in a database manner, the importance of the research issue is highlighted. Examples of applications that deal with partially structured data are:

- Health care system: the structured part for example includes the name, address and date of birth of patients and the un-structured part include the description of the illness or a doctor visit.
• Multi-products selling system: the structured part includes the name, price of the product while the un-structured part includes the specification of that product.

These motivations are the basis for the research and lead to the following discussion regarding the research problems and directions.

3.2 Research Problems and Directions

The Web data can be classified as a semi-structured data, it can by no means fit into the rigid structured data proposed by the conventional models such as relational model or object-oriented data model (although some legal data or business data can fit such as payroll system and flights timetable). The previous section has provided a brief overview of XML as a vehicle for describing and representing such semi-structured data. Also it identified the tremendous need for database-like activities to organise, store, query, restructure and manipulate large collections of XML documents in an efficient way (Suciu (2001) and Vianu (2001)). If we try to model semi-structured data into a conventional data model such as a relational DBMS, it will yield a non-efficient way to store this data because of its irregularity and incompleteness and the lack of any schema and structure. Thus there is a need for a different way to model this data and manipulate it.

Various attempts to provide this facility have been reviewed. These were broadly classified as native XML database technologies, relational XML database approaches, and those using object and object-relational database features. A general weakness of the reviewed attempts is the lack of coherent formal data models for XML document collections. This is a necessary prerequisite to a rigorous treatment of data management issues. The exceptions are some formal data models for XML which are referenced later in this section and discussed in more details in section 4.

The main advantage of relational databases and the one that made it survive and dominate the database market since its invention by Codd in 1970 (Codd 1970) and the first commercial database management system based on it “System R” by IBM in 1974, is the strong formal mathematical model that it is based upon. Specifically, the relational data model, relational algebra and
calculus provide the relational database with a formal model of data structures and operations that has made the development of efficient and optimised query plans and execution possible. Defining similar formal models for XML databases is problematic because it requires the handling of semi-structured data. The lack of a pre-determined constraining structure means that conventional representational and query optimisation approaches developed for conventional databases cannot be applied directly to XML databases.

Thus, in seeking to apply database theory and technology to the XML document collections, adaptations are required to provide a basis for more rigorous treatment for XML databases. This adaptation should allow for the definition of the XML structure, manipulations and integrity rules for XML document collections. Further, it can be noted that research into XML databases is in anticipation of patterns of usage which have yet to emerge (data-centric XML vs. document-centric XML (Bourret (2002)), XML with or without Schema (DTD, XML Schema …) (W3C Online)). Therefore, in order to focus the research, such that issues of performance can be addressed, it is necessary to make assumptions relating to the types of data and modes of use that the research will address. These assumptions are based on current evidence, emerging and likely trends, and those features of XML document collections that are in most need to adaptations of solutions developed for conventional databases.

Accordingly, the research focuses on XML document collections where there is some consistent structuring, but within which there are elements of ad hoc or semi-structured data. The research refers to this class of semi-structured data as partially structured data. This means that the research focuses on a hybrid model, which is mainly data-centric but with part of it as a document-centric XML and with a prescriptive schema for at least the data-centric part.

Although there are some attempts to define a formal data model and algebra for XML and semi-structured data, there is no widely accepted formal notation for that. These attempts can be categorized as follow:
• Model and algebra for XML data (such as a formal data model and algebra for XML (Beech et al. 1999), Lore (McHugh and Widom 1999), W3C XQuery data model (Online 2003), XAL (Frasincar et al. 2002), SAL (Beeri and Tzaban 1999)),
• Data model and algebra for Document-Centric XML (Kim, S.W. et al. 2002).
• Physical Algebra for XML (Paparizos et al. 2002).

The research will utilise and enhance a model in the first category, namely; model and algebra for XML data, so as to better address the data-centric aspects of the above problem. This can be viewed as a key to adapting conventional database technology for partially structured data. The formal model adopted is defined and discussed in section 5.

3.3 Research Proposal Outlines

In the previous section, the research problems and directions are presented. This section defines a research proposal by defining the research aims, research objectives and the methodology, which is to be used to achieve the research objectives.

3.3.1 Research Aims

The broad aim of the research is as follows:

To seek ways of efficiently managing collections of XML documents with some common prescribed sub-structure – these are referred to as partially structured data.

The approach that will be taken to achieve this aim is as follows: the research seeks a formal XML database model and algebra - analogous to conventional database models and algebras – to define the semantic and syntactic structures of partially structured XML document collections. This model is then used to investigate theories and techniques to achieve efficient storage, querying, manipulation and restructuring of XML document collections, based upon adaptation of those developed for relational databases.
Therefore, the research questions are as follows:

- How can partially structured XML document collections be formally defined, such that both semantic and syntactic structures can be defined as a data model?
- How can manipulations on partially structured XML document collections be formally defined as algebra?
- How can the proposed data model and algebra be used for query plan optimisation?

3.3.2 Research Objectives

In order to address the above aims and questions, the objectives are as follows:

- To gain a critical understanding of current data models for representing XML data.
  
  This critical understanding will be achieved through a review of the literature. The aim is to identify a taxonomy of the different approaches, a comparison of their features, and an evaluation of their relative strengths and weaknesses with respect to, completeness of capture of the semantics of XML documents, the availability of strategies for efficient storage, and the potential for taking advantage of optional XML meta data, such as XML schemas, DTDs, etc.
  
- To define a formal representation of XML documents
  
  The formal representation is based upon conclusions of the preceding review of the literature.
  
- To gain a critical understanding of current XML query languages.
  
  This review will be primarily through a literature survey of XML query language research, supplemented by a review of the literature on interfaces to databases and query language theory. The aim is to identify limitations of current systems, to elucidate the ways in which the semantics of XML documents can potentially be exploited by a query language, and to derive
guidelines for good query language design, which suit the Web / semi-structured data situation.

- To define desirable features of an XML query language.
- To formally define XML data query algebra with sufficient expressiveness, with respect to those desirable features defined.
- To seek an implementation of the above formalisms by mapping to existing database technology.
- To seek an analysis and evaluation of the implementation, in comparison with alternative approaches.

The next section discusses the methodology, which will be used to achieve the research objective.

3.3.3 Methodology

Based on the previous research objectives and to achieve the research aims, the following methodology will be used:

- Study the existing mature database theories specially the ones related to data model and algebra and query optimisation techniques.
- Select a mapping between formal XML data model and adapt the data model to cope with the partially structured data.
- Conduct an experiment in order to verify and validate the chosen data model to test the performance of it especially in storing and querying a large collection of partially structured XML documents.

3.4 Conclusion

The motivation behind the research discussed lead to a discussion of research problems and directions which was then followed by a research proposal outline. The research is of high importance and value in the field of the XML database. The following part addresses in more details the problems already discussed by studying database theories and XML, followed by a study of a formal data model and algebra in section 5.
4 Database Theories and XML

The objective of this section is to discuss the evolution of database theories toward storing, querying and managing XML documents. As discussed in the previous sections, there is a great overlap between XML and databases in general and semi-structured data in particular, and there is a great need to apply database theory to store, query and manage a collection of XML documents. This section discusses in depth the evolution of database theories (specifically data models and query algebra as a query language) and how this effects XML documents in general and partially structured XML document - as this is the main aim of the research - in particular.

This section is organized as follows: in section 4.1: database theories and XML are discussed. Then section 4.1.1 shows the evolution of database data models and algebra namely the relational data model and algebra in section 4.1.1.1, the extended relational algebra in section 4.1.1.2, the object-oriented data model and algebra in section 4.1.1.3 and finally the semi-structured data model and algebra in section 4.1.1.4. Then section 4.2 studies the database system architecture, with a focus on the ANSI-SPARC model in section 4.2.1, the following section 4.2.2 covers how XML technologies can be mapped to the ANSI-SPARC model. The logical and Physical XML data model is discussed in section 4.2.3 and 4.2.4, and the native XML data base in section 4.2.5. The requirements for querying web data in general are described in section 4.3 and the conclusion is drawn in section 4.4.

4.1 Database Theories and XML

XML and associated technologies have a number of aspects in common with database theories and technologies, which motivates this discussion. Specifically, both are concerned with semantically meaningful representation of data, storage, manipulation and data integration. However, as Vianu (2001) observes, XML is a subset of SGML, so it has its root from the document community, not the database community, which leads to a number of differences as well as difficulties in combining both. Consequently, the aims of XML technologies have tended to address requirements and perspectives specific to the Internet and document communities. Because of
that, XML is widely used to store data on the web and to exchange data between homogenous and heterogeneous sources on and off the Internet as well as the Intranet. On the other hand, database technology has an extensive and mature research and technological base, which begs the question, can approaches successfully applied for databases, be usefully adapted for use with XML documents and can the two technologies be used effectively in combination? The need for adaptation arises from the fact that XML data is naturally semi-structured data, which makes it difficult to directly apply the strongly structured database theories (such as relational or object-oriented data models).

Database theories and technologies are grounded in data models to represent the data, as well as query language, which is based on that data model to query data. The query languages can be based on either (Abiteboul et al. (1995)):

- **Algebra**: such as relational algebra which consists of a simple algebraic operation for manipulating relations to construct answers to queries
- **Calculus**: such as relational calculus which is logic based and is a variant of the predicate calculus of first order logic
- **Logic programming**: such as the Datalog (for Database Logic), which can be used as logic programming without function symbols.

The three techniques yield similar results in representing the semantics of querying relational data, which allows query transformation and therefore better query optimisation and manipulation. So, the following discussion is focused on the query algebra as a query language tool.

The traditional database models such as the relational data model and its associated relational algebra are based on a well-structured data. This is not the case with XML data. This makes applying this data model directly to XML impossible without adaptation, also as Widom (1999) noted, XML is not a data model, rather, it is a document format. Therefore, a prerequisite for exploring the relationship between XML and databases is to seek a data model into which XML documents can be mapped. And then seek a query algebra that can be used over that data model as a querying language.
Therefore our general motivation question is as follows:

- How can XML be mapped as a database like data model? And what possible query language and algebra can be proposed?

This question leads to a number of specific directions that need more exploration.

- Is it better to map XML as a semi-structured data such as Object Exchange Model (which is more common sense, since XML data is semi-structured by nature)?

- Or what is the benefit of mapping XML to a traditional database-like model and how this will affect the database like efficient query process instead of the navigational process supplied by semi-structure data model? This leads to a study of the relationship between XML and existing data models (relational, object oriented and object relational)

- Or is there a need for new data model and query algebra to represent syntactic and semantic of XML document and to efficiently query it?

- How can XML technologies be mapped into the three levels Database Architecture defined by ANSI-SPARC? How can the data Independent concept be applied to data stored in XML format?

The following sections seek to answer the above questions

4.1.1 The evolution of database data models and algebras.

Hereafter, a review of the different database data models and algebras is briefly overviewed. The starting point is the relational data model and relational algebra then going through the extended relational algebra, the object-oriented data model and algebra and finally the semi-structured data model and algebra. This review is inspired by the Algebraic languages for XML database (Chinwala 2001)

4.1.1.1 Relational Data Model and Relational Algebra

In 1970, Codd invented the relational data model and algebra. This was the starting point for the relational database which dominates the database market nowadays. The relational data model depends on a single data
structure which is a table that consists of tuples (rows or records) to represent real world objects and columns (attributes or fields) with the same data type. A field must contain an atomic (scalar) value, this is known as the first normal form (or 1NF) restriction. A relation (or table) is a set of tuples (records) with the same layout. Relational algebra is used to provide a way to query data stored in the relational data model. It consists of five main operators as follows (Ullman 1988) where R and S represent two relations:

- **Union (∪)**: The union of two relations with the same arity R and S is the set of tuples that are in R or S or both.
- **Difference (−)**: The difference of two relations with the same arity R and S is the set of tuples that are in R but not in S.
- **Cartesian product (X)**: If R is of arity k₁, and S is of arity k₂ then R x S is the set of all possible (k₁ + k₂) tuples where the first k₁ components form a tuple in R and the last k₂ components form a tuple in S.
- **Projection (π)**: is a unary operation that chooses and maybe rearranges some or all of the attributes of a relation.
- **Selection (σ)**: is a unary operation that selects a subset of the relation’s tuples based on a given criteria.

Other operators can be derived from the basic operator such as:

- **Intersection (∩)**: can be represented using the difference operation as follows R ∩ S = R − (R − S)
- **Quotient (÷)**: Let r be the arity of R and s the arity of S, and r > s and s <> 0, then R ÷ S is the set of (r − s) – tuples a₁,...,ar-s where for all s-tuples a₁,s+1,...,ar in S the tuple a₁,...,ar is in R.
- **θ - Join**: θ is an arithmetic comparison operator (=, <>, <, >, <=, >=) then R ▷◁ S equal the Cartesian product of R x S where the i_th component of R stands in relation θ to the j_th component of S.
- **Natural – Join (▷◁)**: when only R and S have columns that are named by attributes, then R ▷◁ S can be calculated as follows: compute R x
S, then for each attribute A that names both a column in R and S select row where R.A = S.A and finally remove S.A and rename R.A to be A.

Using the previous operations, the relational algebra is rich enough to express a rich query language over data represented in the relational model.

4.1.1.2 Extended Relational Algebra

The difference between extended relational data model and the relational data model is that the former allows an attribute value not to be scalar. This means it could be another relation (nested relation). This is called non-first normal form (or $\text{NF}^2$) relations. The basic operators used in relational algebra are used in extended algebra with some modification in selection to allow set comparators and to allow algebraic expressions to appear in selection condition. The projection operator allows algebraic expressions to appear in projection lists to facilitate retrieval of nested data from a relation. New operators are added such as nest (specify a name of attribute that will replace specified number of attributes in a single relation) and unnest (inverse of nest) (Chinwala 2001).

4.1.1.3 Object-Oriented Data Model and Algebra

The object-oriented data model and algebra extends the complex object models with the notions of object identity, inheritance and methods. The main difference between the object-oriented algebra on one hand and relational and extended relational algebra on the other is that the former requires sophisticated restructuring operations which open the door to use the functional programming languages to perform these complex restructuring operations. Table 4 summarizes these operators for three object-oriented algebras.
<table>
<thead>
<tr>
<th>Algebra</th>
<th>Operators</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENCORE/EQUAL</td>
<td>select, image, project, union, intersection, difference, flatten, duplicate_elimination, coalesce, nest, unnest</td>
<td>1990</td>
</tr>
<tr>
<td>AQUA</td>
<td>apply, select, exists, forall, mem, set, choose, group, dup_elim, nest, unnest, convert, join, tup_join, outer_join, union, intersect, diff, multiset</td>
<td>1993</td>
</tr>
<tr>
<td>QAL</td>
<td>select, union, differ, intsc, unnest, group, apply, tuple, close, apply_at</td>
<td>1998</td>
</tr>
</tbody>
</table>

Table 4: Object-oriented algebras and operators (Chinwala 2001)

In general, there is a lack of a well-defined set of common operators for the proposed object-oriented query algebra (such as ENCORE/EQUAL (Shaw and Zdonik 1990), Aqua (Leung et al. 1993), and QAL (Savnik et al. 1998)) except for the operators derived from relational algebra (Chinwala 2001).

4.1.1.4 Semi-Structured Data Model and Algebra

Currently, there is no standard accepted algebra for semi-structured data and XML, but there are number of attempts to define a formal data model and algebra for XML and semi-structured data, which can be classified as:

- Data-Centric XML model: model and algebra for XML data, such as A formal data model and algebra for XML (Beech et al. 1999), Lore (McHugh and Widom 1999), W3C XQuery data model (Online 2003), XAL (Frasincar et al. 2002), SAL (Beeri and Tzaban 1999)
- Document-centric XML model (Kim et al. (2002))
- Physical XML Algebra (Paparizos et al. (2002))

The chosen data model is discussed in more details in section 5 of this report.

4.2 Database System Architecture

In the previous sections, different database data models and algebra were overviewed; the aim of this section is to establish similarities and differences in the general architectures of XML-based systems and databases.

40
system architectures. The following sub sections describe the three levels of architecture of database systems. It is followed by the application of this architecture to XML-enabled systems.

4.2.1 The ANSI-SPARC Model

As in Figure 6 the ANSI-SPARC defined the database system as a three levels architecture (Tsichritzis and Klug Eds. (1978)). The architecture of the Database System consists of the Internal (Physical) level, which is concerned with how data is physically stored, the Conceptual (Logical) level which is concerned with how data is modelled and the relationship between data, and finally, External (User View) which is concerned with how different users see the data.

Data Dictionary contains meta-data about the data stored in the database system. Two mappings are needed; between internal schema and conceptual schema this relates the logical record to the physical record, and between external schema and conceptual schema which relates names in the user’s view to the relevant names in the conceptual schema.

![Figure 6: ANSI-SPARC Database Architecture](image)
Data independence means that changing the lower levels will not effect upper levels. For example, Physical data independence means that changing the way data is actually stored such as using different file organization will not affect the conceptual level. And logical data independence means that changing to the conceptual schema will not have an effect on the other users not concerned with these changes (Ullman 1998).

### 4.2.2 XML Technologies as ANSI-SPARC Model

As the three level architecture of database systems discussed in the previous section, there are two important data models we can map XML to; Logical and Physical. The External level is a very important one as well, since it introduces the problem of restructuring XML Documents. It does not provide a new data model. For example XSL can transform any XML document to a different format and display it on different devices such as computers, PDA (Personal Digital Assistant), mobiles phones, etc…which is a way for restructuring XML. This is one of the important advantages of XML.

![Figure 7: XML Technologies as Database Architecture](image)
Figure 7 shows a direct mapping for XML technology (since XML itself is not a Database System) as the same three levels architecture.

The Internal (Physical) level is a plain text file as the definition of XML Document. The Conceptual Model is the basic model of the XML document as a labelled ordered tree (with labels on nodes), possibly with data values associated to the leaves (Vianu (2001)) and External (User View) can be either another XML or HTML or XTHML. The transformation between the XML conceptual model and user views can be done using different technologies such as XSL, XPath and XQuery. Logical data independence which allows changes in conceptual schema without affecting existing external schema is perfectly possible in XML since it is naturally extensible and a structure can be added or deleted at any time. If we stick to storing XML documents as text files (as the definition of XML document), then Physical data independence will not exist since there will not be any changes in how the XML document is stored. But, if a model is proposed to map XML document into DBMS then there will be some sort of physical data independence but one of the main issues in storing XML in DBMS is to be sure that physical data independence is maintained.

The following sub sections review the literature on data models for the representation and manipulation of XML documents. The review covers both logical and physical (storage) models, within the various architectural contexts that researchers are exploring.

4.2.3 Logical XML Data Model

The logical XML data models are mainly concerned with the logical data structures for representing XML documents and operations for their retrieval and manipulation. Figure 8 is a classification of the different logical XML data models. The possible ways is either to use the structured data models or the semi-structured data model such as the Lore system (McHugh et al. (1997)) or a hybrid one such as the Ozone system (Lahiri et al. 1999) or STORED system (Deutsch et al. 1999). Using the structured models, this can be classified as follows:
• Using native XML databases (Staken, K. (2001)): which will be discussed in section 4.2.5.

• Using the Structured-mapping Approach (Klettke and Mayer (2001), Kudrass and Conrad (2002), Runapongsaa et al. (2002), and Shanmugasundaram et al. (1999)): in this technique, the DTD is mapped to equivalent relational schema. And XML queries are mapped also to SQL queries. This technique is good in the data-centric XML documents but is not applicable in document-centric XML documents.

• Using the Model-mapping Approach (Kudrass (2002), Yoshikawa and Amagasa (2001), Shimura et al. (1999) and Florescu and Kossmann (1999)): this technique maps XML to a generic schema, it models XML nodes and edges not the XML data itself. The disadvantage of this approach is that when querying the data, it can produce a huge number of internal joins.

• Opaque Approach (Kudrass (2002)): this technique stores XML as CLOB. So the XML document is stored inside the relational table as one data item. The disadvantage of it is that when dealing with this XML document, it must be loaded every time into the memory.

• Extended Entity-Relation Model: in this technique we either extend the relational model by adding a new abstract XML data type (Klettke and Mayer (2001) and Runapongsaa and Patel(2002)), such as some commercial database or add storing of the semantic of XML (as in ERX Model (Psaila 2002). The problem in the former one is that there is no standard to deal with this new abstract data type in the different implementations of these commercial databases.

4.2.4 Physical XML Data Model

The Logical XML data models, reviewed in the preceding section, provide formalisms for languages and interfaces with which to query and manipulate XML documents. This section reviews complementary issues, i.e., how can those logical models be physically represented such that they can be
executed with sufficient effectiveness and efficiency within the intended scenario.

As shown in figure 9 the two possible ways to store XML documents physically is either using a file system or using a Database Management System.

4.2.5 Native XML Database

XML:DB Initiative ([Online]) defined Native XML Databases (NXD) as follows:

- They define a (logical) model for an XML document - as opposed to the data in that document - and store and retrieve documents according to that model. At a minimum, the model must include elements, attributes, PCDATA, and document order. Examples of such models are the XPath data model, the XML Infoset, and the models implied by the DOM and the events in SAX 1.0.

- A NXD has an XML document as its fundamental unit of (logical) storage, just as a relational database has a row in a table as its fundamental unit of (logical) storage.

- Is not necessary to have any particular underlying physical storage model. For example, the NXD can be built on a relational, hierarchical, or object-oriented database, or use a proprietary storage format such as indexed, compressed files.
Figure 8: Logical XML Data Model
Figure 9: Physical XML Data Model

As Staken (2001) summarized this definition, the Native XML databases mainly store XML documents and its components, the basic unit to deal with the database is the XML document and NXD may not be a standalone database and it can be built over any other DBMS and it may not store XML in a native text format.

4.3 Requirements for Querying Web Data

In this section, the requirements of the query language for web documents will be overviewed.

There are a number of factors effecting query XML Data. The main one is how XML Data is stored; if this data is stored using the file system or XML data servers versus a relational or object-oriented database. In the first case indices are used to speed the access to this data and in the second, XML queries are translated to SQL or OQL language then executed and then the result is translated back to XML data.

Another important factor is the different views of query languages between document point of view and database point of view. As Maier (1998) noticed, queries can be described as a powerful “find” command in one or more documents or it can allow richer forms for reference to portions of a document, while from the Database point of view an XML query is “a tool for structural and content-based query over the world-wide XML database”. The key point here is the granularity of the results, from the Database viewpoint the results can be grouped or data can be divided to arbitrary levels vs. the fixed level of granularity offered by the document retrieval (Abiteboul 2000)
There are a number of main requirements for XML Query Languages, (for example, Maier 1998, XML Query Requirements 2001 [Online], Bonifati and Ceri (2000) and Fernandez et al. (1999)). Most of these derive from a list of fundamental requirements by Maier (1998). Maier’s list addresses a number of issues, i.e., implementation strategy, the design characteristics of the query language, and its expressiveness. In summary, Maier’s requirements can be categorized as follows:

- **Orthogonal**
  - XML Output: The result of the XML query will be an XML document. This also allows Computationally (Abiteboul 2000) as the result of the query can be used as input for another query and the result must be in the same data model.

- **Expressiveness and completeness**
  - Query Operations: The Query language must perform the following operations
    - Selection: Select subset of the document or the whole document based on content, structure or attributes.
    - Extraction: Removing particular elements of a document.
    - Reduction: Removing selected sub-elements of an element.
    - Restructuring: Constructing a new set of element instances to hold queried data.
    - Combination: combine more than one element into one.
  - Preserve Order and Association: as XML is an ordered document, the query language must preserve this order in the results.
  - Mutually Embedding with XML: XML can contain queries statement while query statement can contain XML elements.
• Xlink (Online) and Xpointer (Online) Cognizant: a query language should provide for following XLinks and XPointers.

• Namespace Alias Independence: The query language should not dependent on namespace aliases local to an XML document

• Support for New Data types
  o Suitable for Metadata

• Implementation
  o Server-side Processing: Queries can be executed remotely on the server with no dependence on resources in its creation context for evaluation.
  o Programmatic Manipulation: the query language can be easily created by applications rather than writing by the users or programmers.
  o XML Representation: the representation of the language will be in XML, so no need for a special mechanism to store the queries statements.

• Schema Features
  • No Schema Required: If the schema is not known, XML Query must depend on the self-describing feature of XML Document.
  • Exploit Available Schema: But if the schema is known, the Query language must make use of it for error detection

The Query Language Semantics: The language must have a clear definition to allow efficient processing.

• Precise Semantics: this is to allow determining result structure, equivalence and containment.

• Compositional Semantics: the query expression should be the same wherever it appears.
However, the above features do not address in detail, the expressiveness of XML query languages, for example, in the way in which the notion of relational completeness does for relational languages. To do so, it is first necessary to establish some formal description of the manipulations that an XML query language should be able to apply to an XML database. Such formalism will have a role equivalent to that of relational algebra for relational databases. It is then possible to evaluation the expressiveness of any XML query language, in terms of those expressions in the formalism that the language can also express.

4.4. Conclusion

This section shows the evolution of data models from relational to semi-structured data models and also the different XML data models. One of the important things that allow relational databases to survive from 1970 until now is that it is based on a formal mathematical data model. So, in the next section, a formal data model and query algebra will be defined for an XML database taking into consideration the sub-class in which the data is partially-structured.

5 Formal XML Data Model and Query Algebra

Following the discussion in the previous section regarding the relation between XML and databases and the discussion in section 3 regarding the research aims and objective which shows how important is to map XML into a formal data model with query operations and query algebra, this section defines the formal data model to map XML into and an XML algebra based on that data model. This section reviews the different data model first, and then describes in more details the chosen data model, and finally it defines the XML algebra based on that model.

This section is organized as follow: Section 5.1 discusses the formal data model for XML, introducing it first, then in section 5.1.1, a brief overview of the proposed data model, in section 5.1.2 the selected formal data model is described in more details. Section 5.1.3 describes the three examples to clarify this model which will be found in Appendix A. In section 5.2, an XML
algebra based on the model is described in detail. And finally section 5.3 presents the Conclusion.

5.1 A Formal Data Model for XML

In this section, a formal data model and query algebra for XML databases is presented. This preliminary work aims to provide a formal basis for a rigorous investigation of query optimisation and transformation for XML databases. Specifically, the aim of the proposed investigation is to study a subclass of XML databases, containing XML document collections with a common prescribed sub-structure, in addition to the ad hoc structure that characterises semi-structured data. I refer to this subclass as partially structured data. In this investigation, I will seek to apply, possibly in an adapted form, query optimisation techniques that have been developed for relational databases. Consequently, the formalism presented, is defined specifically to allow (partial) transformation between the XML data model and the relational model, and is adapted from the XML data model in Beech et al. (1999).

5.1.1 A Review of XML Data Models

XML is defined as a document format and not a data model. In particular, the XML definitions (XML [Online]) lack operations that can manipulate XML data, structures and queries. However, definition of a formal data model for XML is a hot research topic, and a number of studies into XML databases propose data models. In this section, these proposals are reviewed, and their suitability as the basis for an enhanced data model for XML that can define partially structured XML data, is assessed.

XML data model can be briefly categorized into two main classes:

- Relational-based data models (invented by Codd 1970). In this type of model, the data structure used in this model is the relation (table). The data are modelled using tables and relations between tables and each other.

- Graph-based data model (e.g., Beech et al. 1999). The data structure used in this type of model is the graph, consisting of nodes (vertices)
and edges that represent the relation between different nodes. The graph could be cyclic (e.g., XAL (Frasincar et al. 2002)) or acyclic (tree) (e.g. TAX: A tree algebra for XML (Jagadish et al. 2001)), depending on how the reference edge (IDREF and IDREFS) in XML document is represented. An example of a model that works both ways is Lore (McHugh and Widom 1999). There are number of proposals defining graph-based formal models such as “A formal data model and algebra for XML” (Beech et al. 1999), Lore (McHugh and Widom 1999), W3C XQuery data model (Online 2003), XAL (Frasincar et al. 2002), SAL (Beeri and Tzaban 1999), A data model and algebra for Document-Centric XML (Kim, S.W. et al. 2002), a Physical Algebra for XML (Paparizos et al. 2002).

Both main approaches are relevant to this study, since the two approaches aim to capture the semantics of XML document collections, and to seek ways of applying relational query optimisation techniques. The graph-based models provide the most direct and natural formalism for XML documents, since these directly describe the hierarchical composition of an XML document, as nested tagged elements, and links within and between documents, whereas, relational-based models offer a direct link with relational database theory and query optimisation in particular. So, each one of these models has some advantages and disadvantages regarding the modelling of partially structured data as follow:

Relational: it is too rigid to fit the semi-structured part in the XML document, but it is the optimum for the structured part. It has well-defined theories regarding query optimisation and operations. Also, there are some major differences such as the XML data is ordered while the relation is a set, and therefore is unordered. The handling of duplicate as well as non-complete data is another major difference between the two.

Graph-based: graphs are a natural representation of XML documents. Since XML is based upon nested tagged nodes (represented by graph vertices or nodes) that contains the data in elements or attributes and inter-document links, intra-documents links represented by the graph edges. So, the graph-based is the commonly used data model for XML data since it can
represent its complex structure. But there is no solid query optimisation techniques compared with the relational one.

So, based on that, the Graph-based is the appropriate way to define a data model that can represent XML document. In the next section a formal data model is described in more details to represent XML documents, and then the following section describes the operation and XML algebra based on that model.

5.1.2 Formal XML Data Model

The chosen data model is based on “A formal data model and algebra for XML (Beech et al. 1999)” for the following reasons:

- It is a very well defined and simple data model.
- It can represent both data-centric as well as document-centric collections of XML documents.
- It is based on XML Infoset (Online 2001) which is a W3C recommendation that defines a set of specifications needed to refer to the information in an XML document. Also XQuery is based on XML Infoset.
- It defines a number of operations to deal with collections of XML documents.
- Although it does not define optimisation strategies or physical algebra operations, this gives room to developing such optimisation techniques that suits the partially structured data.
- A number of formal models such as XAL (Frasincar et al. 2002) were inspired by it.
- There are no great differences between the different graph-based data models.

So, based on the previous criteria, the research is based on that data model. This model can be briefly described as a node-labelled and directed graph where the graph nodes (V) are either element or a data value and the graph edges are directed element containment edges (E) or attribute edge (A)
or reference edge (R). And finally (O) which represents a set of ordering relation of child element within a parent element.

5.1.2.1 Detailed Description of the Formal XML Data Model

In this section, the formal XML data model is described in more detailed as follow: (Beech et al. 1999)):

- A node-labelled directed ordered graph
- V: The graph vertices (or nodes) are either an element or a data value. Each vertex must have a parent (another element vertex) with the exception of the root as a special case which has fictitious root vertex. Each vertex has a unique, immutable and system generated identifier. This is different than the ID which may be used for internal referencing between XML elements.
- The graph directed edges are one of three different types.
  - E: a set of directed element containment edges. It relates a parent element to a child which could be another element where the name of the edge is the generic identifier of the child name or a data value with a special name ~data, a comment with a special name ~comment or a processing instruction with a special name ~PI
  - A: a set of directed attribute edges. It relates an element to its attribute data value.
  - R: a set of referenced edges. It relates an element to another referenced element via IDREFs or IDREFSs or XLinks, URLs or other reference mechanisms.
- O: represents a set of ordering relation of child element within a parent element. It represents the total order among all edges while it does not represent order among different types nor elements with different parent vertex. This order is defined for the three different types of edges as follows:
o E: the order of the children as they appear within the parent element
o A: not defined, since the XML attributes are un-ordered
o R: The order as they were in the document in the case of IDREFS

- Vertices have two basic properties:
  o value: returns the system generated identifier for this element, and in the case of value vertex it returns its value
  o type: element in the case of an element vertex and the data type of the value in the case of value vertex

- Element vertices derived properties (based on edge and order information)
  o gi: name (with namespace qualified if relevant) of this vertex
  o parent: vertex parent
  o referredby: set of vertices that reference this vertex through a reference edge
  o childelements: set of all element containment edges from this vertex
  o attributes: set of all attributes edges from this vertex
  o references: set of all reference edges from this vertex

- E: has the following basic properties
  o Parent: returns the from vertex
  o Child: returns the to vertex
  o Name: returns the name of the edge
  o Type: returns E

- A: has the following basic properties
  o Parent: returns the referring vertex
  o Name: returns the name of the edge
• Value: the attribute value
• Type: always A

• R: has the following basic properties
  • Parent: returns the \textit{from} vertex
  • Child: returns the \textit{to} vertex
  • Type: $R_x$ where $R$ indicate that this is a reference and $X$ indicates the kind of that (XLink, URI, …)
  • Refedge: the set of attribute or element edges that provides the reference information

• E, A, R has the following derived properties
  • Next: returns the following edge
  • Previous: returns the previous edge

• O: it has the following basic properties are
  • $e$: return the current edge
  • successor: the successor of current edge

So, briefly based on the previous definitions, a \textbf{Formal Data Model} can be defined as follow:

• A graph $G = (V, E, A, R, O)$ represents the data model for XML elements. Where
  • $V = V_{\text{element}} \cup V_{\text{int}} \cup V_{\text{string}} \cup …$
  • $E$: represents the set of directed element containment edges
  • $A$: represents the set of directed attribute edges
  • $R$: represents the set of directed reference edges
  • $O$: represents the total order between edges of a particular class $E$, $A$ or $R$, that connect parent element to its children
5.1.3 Examples of Formal XML Data Model

In Appendix A, there are three examples showing how to model XML document using the previous data model. The first example is a data-centric XML document, the second is a document-centric XML document and the last one is a hybrid a partially structured document. These XML documents were extracted - with some changes - from the department of Information Studies, University of Sheffield web site. (Online)

5.2 XML Algebra

The XML algebra is based on the previous data model (Beech et al. 1999). Its goal is to operate on a collection of XML documents, allowing the selection of a whole document or a part of a document based on specific criteria, and restructuring the results as a new XML document. It allows also joins between XML documents. It deals with XML’s graph structure, heterogeneity of types, ownership vs. reference and finally the order of the document. It also allows composability and therefore transformation and optimization. Its operation can be summarized as follow:

• **Navigation Operations**: the “follow” operation $\phi$, it starts with a set of vertices and then follows edges of a given type (E, A, R, or any) and with a given name returning a set of edges. And to get a set of vertices it must be composed with a child operation. The “inverse follow” operation $\phi_{inv}$ also takes the edge type and name as parameters and a set of vertices, it returns all the edges of that type and name that lead to the specified set of vertices.

• **Selection Operation**: the “selection” operation $\sigma$ allows selection of a given collection based on a given criteria, it returns a collection where is the criteria is true. Vertices and edges properties can be used to construct the selection criteria as well as standard comparison operations ($=, <>, <, >, <=, >=$) and Boolean operations (and, or, not). The selection operation supports also Existential and Universal qualification.
• **Join Operation:** when two documents are queried with a join condition between them, the Cartesian product is calculated. Then, for the true join condition only between the two vertices, a virtual reference edge is created and can be used as a normal reference edge during the evaluation of the query only.

• **Construction Operations:** are used in building a new fragment of a document based on the selection conditions. The “expose” operation returns the fragment of a document identified by navigation operations in conjunction with selection operation. The “return” operation returns copy of the fragment of a document identified by navigation operations in conjunction with selection operation. The “create edge” and “create vertex” are used to create new fragment of XML by attaching edges and vertices to the root and recursively attaching edges and vertices to the attached vertices.

• **Other Operations:**
  o “∑ sort” which orders a set of edges
  o “χ unordered” which indicates that the order is not important and this can help in the query optimization
  o “µ map” which applies a specified function to a collection of edges or vertices. It does not include the input collection in the result collection (Unlike the kleene star *)
  o “* kleene star” operator is used to indicate the possibility of infinite repetition of a function. It can for example allow the navigation among paths repeatedly until reaching a fixed point.
  o “δ distinct” which eliminates duplicates from a set of vertices or edges.
  o “pick” transforms an element of a collection into a singleton
  o “flatten” flattens any collection of collections of any nesting depth into a flat collection
• "group by" operator is used to create element and attribute vertices in the result that aggregate or summarize information from a group of similar vertices or edges.

After describing the basic operations of this XML algebra, the next sections explore these operations further and build on them to reach a proper query transformation and optimization techniques for a collection of partially structured XML documents.

5.3 Conclusion

A formal data model and algebra is described for a collection of XML document. This formal definition is the basis of the research into allowing better storing techniques and query optimisation for XML database.

In the next section the research proposal will be described giving a detailed description of the research directions.

6. Research Proposal and Conclusion

After discussing the basis of the research in the previous sections, this section concludes the report by presenting in summary the general and the specific research issues as well as the time table to conduct the research.

Figure 10 shows the general research issues, which can be described as follows: data can be categorised as highly structured, semi-structured or un-structured. Conventional database technologies are quite efficient when it comes to dealing with the highly structured data but poor in dealing with semi-structured and un-structured data. On the other hand, the emerging web data management is good in dealing with the semi-structured and un-structured data, while poor in dealing with highly structured data. Therefore, the need arises to combine both these technologies. The result is an XML databases. Further, the graph shows how XML can be stored either using a conventional database or file system. These different approaches were discussed in section 4.
Conventional DB (Mature theory and technology)

Web Data Management (New technology)

Highly Structured Data

Semi-Structured Data

Un-Structured Data

Structured Data Models

Semi-Structured Data Models

Un-Structured Data Models

XML Database

Native XML DB

Use Objects or OODBMS

Use Relational or Object Relational DBMS

Hybrid Approach (Ozone/STORED)

Labelled Tree or Graph (Lore)

File System

Store XML as relations

Opaque Approach

Store XML as CLOB

Extended Entity-Relation Model

Structured-mapping Approach

Model-mapping Approach

Store the Semantic of XML (as in ERX Model)

Add XML as a new Abstract Data type

Figure 10: General Research Issues
Figure 11 shows the specific research issues. It follows the research aim stated in section 3

“To seek ways of efficiently managing collections of XML documents with some common prescribed sub-structure – these are referred to as partially structured data.”

Figure 11: Specific Research Issues
Therefore, the hypothesis that the research will test is: If there is a prescribed structural component common to all documents within an XML database. Relational query optimisation techniques can then be adapted and applied to produce enhanced XML querying.

To test the hypothesis, there is a need to establish the following:

- Identification of the primitive operations, Meta information and heuristics that are applied in standard relational database querying.
- Definition of an XML data model which has structures and operations analogous to those of the relational model.
- Extension of that model, to include prescribed common structure.
- Definition of a relational representation of an XML document collection which exploits structural pre-knowledge.
- Translation of XML to relation representation (exploiting pre-structural knowledge)
- Translation of XML queries to standard SQL to query the relational representation for the structured part.
- Translation of XML queries to Xpath to query the semi-structured part.
- Evaluate in comparison with existing approaches.

Using the methodology described in section 3, the research will follow the above steps, and the time table to achieve the research objectives and aims is as follow:

<table>
<thead>
<tr>
<th>Item</th>
<th>Duration</th>
<th>Date From</th>
<th>Date To</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPhil/PhD upgrade report</td>
<td>14 Months</td>
<td>10/2002</td>
<td>11/2003</td>
</tr>
<tr>
<td>Study the existing mature database theories, especially the ones related to data model and algebra and query optimisation techniques.</td>
<td>2 Months</td>
<td>12/2003</td>
<td>1/2004</td>
</tr>
</tbody>
</table>
Adjust the selected formal XML data model to cope with the partially structured data. | 2 Months | 2/2004 | 3/2004 |
---|---|---|---|
Conduct an experiment in order to verify and validate the chosen data model to test the performance of it especially in storing and querying a large collection of partially structured XML documents. | 2 Months | 6/2004 | 7/2004 |
• Generating the test data (design a software to generate the sample data to be used in the testing) | 1 Month | 4/2004 | 4/2004 |
• The core system to query the partially structured XML | 2 Months | 6/2004 | 7/2004 |
  o Implementation | 2 Months | 12/2004 | 1/2005 |
Results analysis and comparison to existing techniques | 6 Months | 4/2005 | 9/2005 |
Thesis writing

**Table 5: Research Time Table**
References


DTD – Document Type Declaration http://www.w3c.org/TR/REC-xml#dt-doctype [Accessed 1.11.2002]


HTML - Hypertext Markup Language [http://www.w3c.org/MarkUp](http://www.w3c.org/MarkUp)  [Accessed 1.10.2002]

Information Studies Department at the University of Sheffield [http://www.shef.ac.uk/~is/](http://www.shef.ac.uk/~is/)  [Accessed 31.03.2003]


Lahiri, T., Abiteboul, S. and Widom, J. *Ozone: Integrating Structured and Semistructured Data*. Proceedings of the Seventh International Conference on Database Programming Languages, Kinloch Rannoch,
Scotland.


Runapongsa, K, Patel, J. M. (2002) Storing and Querying XML Data in Object-Relational DBMSs EDBT 2002 Workshop on XML-Based Data


Shaw, G. M. and Zdonik, S. B. (1990) A Query Algebra for Object Oriented Databases Proceedings of the 6th International Conference on Data Engineering, Los Angeles, California, USA.


W3 Schools http://www.w3schools.com [Accessed 10.10.2002]


XHTML The Extensible Hyper Text Markup Language http://www.w3.org/MarkUp [Accessed 1.10.2002]


XQuery 1.0 and XPath 2.0 Data Model– W3C Working Draft 02 May 2003 http://www.w3.org/TR/xpath-datamodel/ [Accessed 1.8.2003]

XSL - The Extensible Stylesheet Language Family http://www.w3.org/Style/XSL/ [Accessed 1.11.2002]
Appendix A

Examples of Formal XML Data Model

In this appendix, three examples show how to model XML into a formal data model.

Example 1: A Data-Centric XML document

Figure 12 shows an example of a data-centric XML document. Then figure 13 shows the graph representation of this XML document and finally the table 6 shows how this document mapped using the previous data model. This document was taking - with some changes - from the department of Information Studies, University of Sheffield web site (Online).

```xml
<University>
  <Department>
    <Name>Information Studies</Name>
    <ResearchGroup>
      <Name>Computational Informatics Research Group</Name>
      <Director>Peter Willett</Director>
      <Focus>database management systems</Focus>
      <AcademicStaff HeadofResearchArea="Barry Eaglestone">
        <StaffName>Barry Eaglestone</StaffName>
        <StaffName>Angela Lin</StaffName>
        <StaffName>Miguel Nunes</StaffName>
      </AcademicStaff>
    </ResearchGroup>
  </Department>
</University>

Figure 12: Sample Data-Centric XML Document
Figure 13: Graph Representation of Data-Centric XML Document
<table>
<thead>
<tr>
<th>Edg</th>
<th>Name</th>
<th>Parent</th>
<th>Child</th>
</tr>
</thead>
<tbody>
<tr>
<td>e₁</td>
<td>“University”</td>
<td>Vₐ root</td>
<td>Vₐ University</td>
</tr>
<tr>
<td>e₂</td>
<td>“Department”</td>
<td>Vₐ University</td>
<td>Vₐ Department</td>
</tr>
<tr>
<td>e₃</td>
<td>~data</td>
<td>Vₐ Department</td>
<td>“Information Studies”</td>
</tr>
<tr>
<td>e₄</td>
<td>“ResearchGroup”</td>
<td>Vₐ Department</td>
<td>Vₐ ResearchGroup</td>
</tr>
<tr>
<td>e₅</td>
<td>“Name”</td>
<td>Vₐ ResearchGroup</td>
<td>Vₐ RGName</td>
</tr>
<tr>
<td>e₆</td>
<td>~data</td>
<td>Vₐ RGName</td>
<td>Computational Informatics Research Group</td>
</tr>
<tr>
<td>e₇</td>
<td>“Director”</td>
<td>Vₐ ResearchGroup</td>
<td>Vₐ RGDirector</td>
</tr>
<tr>
<td>e₈</td>
<td>~data</td>
<td>Vₐ RGDirector</td>
<td>Peter Willett</td>
</tr>
<tr>
<td>e₉</td>
<td>“Focus”</td>
<td>Vₐ ResearchGroup</td>
<td>Vₐ RGFocus</td>
</tr>
<tr>
<td>e₁₀</td>
<td>~data</td>
<td>Vₐ RGFocus</td>
<td>database management systems</td>
</tr>
<tr>
<td>e₁₁</td>
<td>“AcademicStaff”</td>
<td>Vₐ ResearchGroup</td>
<td>Vₐ RGAcademic</td>
</tr>
<tr>
<td>e₁₂</td>
<td>“StaffName”</td>
<td>Vₐ RGAcademic</td>
<td>Vₐ StaffN1</td>
</tr>
<tr>
<td>e₁₃</td>
<td>~data</td>
<td>Vₐ StaffN1</td>
<td>Barry Eaglestone</td>
</tr>
<tr>
<td>e₁₄</td>
<td>“StaffName”</td>
<td>Vₐ StaffN2</td>
<td>Angela Lin</td>
</tr>
<tr>
<td>e₁₅</td>
<td>~data</td>
<td>Vₐ StaffN3</td>
<td>Miguel Nunes</td>
</tr>
<tr>
<td>e₁₆</td>
<td>“StaffName”</td>
<td>Vₐ RGAcademic</td>
<td>Vₐ StaffN3</td>
</tr>
<tr>
<td>e₁₇</td>
<td>~data</td>
<td>Vₐ StaffN3</td>
<td>Miguel Nunes</td>
</tr>
<tr>
<td>e₁₈</td>
<td>“HeadofResearch Area”</td>
<td>Vₐ RGAcademic</td>
<td>Barry Eaglestone</td>
</tr>
</tbody>
</table>

Table 6: Data Model for Data-Centric XML Document
2.1.2 Example 2: A Document-Centric XML document

Figure 14 shows and example of a document-centric XML document. Then figure 15 shows the graph representation of this XML document and finally the table 7 shows how this document mapped using the previous data model. This document was also taking -with some changes- from the department of Information Studies, University of Sheffield web site.

< Department webaddress="http://www.shef.ac.uk/uni/academic/I-M/is/">
  <Name>Department of Information Studies </Name>
  <Description>
    <Para>Welcome to the Department of Information Studies World Wide Web pages where you will find information about the Department. Information about</Para>
    <List>
      <Item xlink::HREF="../people/people.html">the staff</Item>
      <Item xlink::HREF="../courses/index.html">our degree courses</Item>
    </List>
    Thank you for visiting our web site
  </Description>
</Department>

Figure 14: Sample Document-Centric XML Document
Welcome to …
Thank you …

- Our degree courses
- the staff

Figure 15: Graph Representation of Document-Centric XML Document
<table>
<thead>
<tr>
<th>Edge</th>
<th>Name</th>
<th>Parent</th>
<th>Child</th>
</tr>
</thead>
<tbody>
<tr>
<td>e₁</td>
<td>“Department”</td>
<td>Vₚᵣoot</td>
<td>V₂₅_department</td>
</tr>
<tr>
<td>e₃</td>
<td>“Name”</td>
<td>V₂₅_department</td>
<td>V₂₅_departmentName</td>
</tr>
<tr>
<td>e₄</td>
<td>~data</td>
<td>V₂₅_departmentName</td>
<td>“Department of Information Studies”</td>
</tr>
<tr>
<td>e₅</td>
<td>“Description”</td>
<td>V₂₅_department</td>
<td>V₂₅_description</td>
</tr>
<tr>
<td>e₆</td>
<td>“Para”</td>
<td>V₂₅_description</td>
<td>V₂₅_para</td>
</tr>
<tr>
<td>e₇</td>
<td>~data</td>
<td>V₂₅_para</td>
<td>“Welcome to …”</td>
</tr>
<tr>
<td>e₈</td>
<td>“list”</td>
<td>V₂₅_para</td>
<td>V₂₅_list</td>
</tr>
<tr>
<td>e₉</td>
<td>“Item”</td>
<td>V₂₅_list</td>
<td>V₂₅_item1</td>
</tr>
<tr>
<td>e₁₁</td>
<td>~data</td>
<td>V₂₅_item1</td>
<td>“the staff”</td>
</tr>
<tr>
<td>e₁₂</td>
<td>“Item”</td>
<td>V₂₅_list</td>
<td>V₂₅_item2</td>
</tr>
<tr>
<td>e₁₄</td>
<td>~data</td>
<td>V₂₅_item2</td>
<td>“our course data”</td>
</tr>
<tr>
<td>e₁₅</td>
<td>~data</td>
<td>V₂₅_para</td>
<td>“Thank you…”</td>
</tr>
<tr>
<td>e₂</td>
<td>“webaddress”</td>
<td>V₂₅_department</td>
<td>“<a href="http://www.shef.ac.uk/uni/academic/I-M/is/%E2%80%9D">http://www.shef.ac.uk/uni/academic/I-M/is/”</a></td>
</tr>
<tr>
<td>e₁₀</td>
<td>“xlink”</td>
<td>V₂₅_item1</td>
<td>“../people/people.html”</td>
</tr>
<tr>
<td>e₁₃</td>
<td>“xlink”</td>
<td>V₂₅_item2</td>
<td>“../courses/index.html”</td>
</tr>
</tbody>
</table>

Table 7: Data Model for Document-Centric XML Document
2.1.3 Example 3: A hybrid XML document

The following example is a combination of the previous two examples showing a link between structured part of the document to unstructured part (such as <Description>) and showing also a link between unstructured part to structured part (such as <Head of Department>). Figure 16 shows XML document. Then figure 17 shows the graph representation of this XML document and finally the table 8 shows how this document mapped using the previous data model.

<University>
  <Department webaddress="http://www.shef.ac.uk/uni/academic/I-M/is/">
    <Name>Information Studies</Name>
    <Description>
      <HeadofDepartment IDREF="PW"/>
      <Para>Welcome to the Department of Information Studies World Wide Web pages where you will find information about the Department. Information about
        <List>
          <Item xlink:href="../people/people.html">the staff</Item>
        </List>
        Thank you for visiting our web site
      </Para>
      <Description>
        <Staff>
          <Name Id="PW">Peter Willett</Name>
          <Name Id="BE">Barry Eaglestone</Name>
          <Name Id="MN">Miguel Nunes</Name>
        </Staff>
        <ResearchGroup>
          <Name>Computational Informatics Research Group</Name>
          <Director IDREF="PW"/>
          <Focus>database management systems</Focus>
          <AcademicStaff>
            <HeadofResearchArea IDREF="BE"/>
            <StaffName IDREFS="BE,MN"/>
          </AcademicStaff>
        </ResearchGroup>
      </Description>
    </Description>
  </Department>
</University>

Figure 16: Sample Hybrid XML Document
Figure 17: Graph Representation of Hybrid XML Document
Table 8: Data Model for Hybrid XML Document