Clustering-based Labelling Scheme - A Hybrid Approach for Efficient Querying and Updating XML Documents

By
Alhadi Ali Klaib

A thesis submitted to the University of Huddersfield in partial Fulfilment of the requirements for the degree of Doctor of Philosophy

School of Computing and Engineering
University of Huddersfield

April 2018

Abstract:

Extensible Markup Language (XML) has become a dominant technology for transferring data through the worldwide web. The XML labelling schemes play a key role in handling XML data efficiently and robustly. Thus, many labelling schemes have been proposed. However, these labelling schemes have limitations and shortcomings. Thus, the aim of this research was to investigate the existing XML labelling schemes and their limitations in order to address the issue of efficiency of XML query performance. This thesis investigated the existing labelling schemes and classified them into three categories based on certain criteria, in order to identify the limitations and challenges of these labelling schemes. Based on the outcomes of this investigation, this thesis proposed a state-of-the-art labelling scheme, called clustering-based labelling scheme, to resolve or improve the key limitations such as the efficiency of the XML query processing, labelling XML nodes, and XML updates cost. This thesis argued that using certain existing labelling schemes to label nodes, and using the clustering-based techniques can improve query and labelling nodes efficiency. Theoretically, the proposed scheme is based on dividing the nodes of an XML document into clusters. Two existing labelling schemes, which are the Dewey and LLS labelling schemes, were selected for labelling these clusters and their nodes. Subsequently, the proposed scheme was designed and implemented. In addition, the Dewey and LLS labelling scheme were implemented for the purpose of evaluating the proposed scheme. Subsequently, four experiments were designed in order to test the proposed scheme against the Dewey and LLS labelling schemes. The results of these experiments suggest that the proposed scheme achieved better results than the Dewey and LLS schemes. Consequently, the research hypothesis was accepted overall with few exceptions, and the proposed scheme showed an improvement in the performance and all the targeted features and aspects.

List of Publications and Research Activities:

Klaib, Alhadi, & Lu, Joan. (Klaib & Lu). *Development of Database Structure and Indexing Technique for the Wireless Response System.* Paper presented at the INFOCOMP 2013, The Third International Conference on Advanced Communications and Computation.

Klaib, Alhadi & Lu, Joan. *Investigation into Indexing XML Data Techniques,* Poster, 5-6 June 2014, Libyan Higher Education Forum, AVision for the Future. London.

Klaib, Alhadi and Lu, Joan (2014) *Investigation into Indexing XML Data Techniques.* In: ICOMP'14 - The 2014 International Conference on Internet Computing and Big Data, 21st - 24th July 2014, Las Vegas, USA.

Article under Review:

Klaib, Alhadi and Venters, Colin. *Clustering-based Labelling Scheme - A Hybrid Approach for Efficient Querying and Updating XML Documents*. [Target Journal – ACM Transactions on Database Systems].

DEDICATION

To my parents, my wife, and my sons who supported me incredibly.

ACKNOWLEDGEMENTS

First of all, all praise and thanks to Allah who endowed me with will, strength, and means to complete this thesis. Without His bounty, grace and mercy this work would have never been accomplished.

I take this opportunity to express my sincere thanks to my supervisor Dr Colin Venters. I am deeply grateful for his genuine guidance without his generous help my thesis would not have taken the final shape.

Finally, I would like to express my deepest gratitude to my mother, my father, my wife, my children, brothers, sisters, and all other relatives and friends, for their emotional and moral support, and also for their love, patience, encouragement and prayers.

List of abbreviations:

Abbreviations	Details
ADG	Approximate DataGuide
DAG	Directed Acyclic Graph
DOM	Document Object Model
DTD	Document Type Definition
GRP	GRoup base Prefix scheme
LLS	Level-based Labelling Scheme
LSDX	Labelling Scheme for Dynamic XML Data
MBench	Michigan Benchmark
NXD	native XML databases
PRIX	Prufer sequence for indexing XML
SAX	Simple API for XML
SQL	Structured Query Language
ViST	Virtual Suffix Tree
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XPath	XML Path Language
XQuery	XML query language

Table of Contents

List of Figures

List of tables

1. Introduction

1.1 Introduction

Extensible Markup Language (XML) has become very significant technology for transferring data through the world of the Internet (Abiteboul, Buneman, & Suciu, 2000; Zhuang & Feng, 2012). Thus, a large amount of research on XML databases and XML technologies such as data retrieval and data update has being carried out (Zhuang & Feng, 2012). XML labelling scheme is an important technique used to handle XML data efficiently and robustly. Labelling XML data is performed by assigning labels to all nodes in that XML document. Every node is provided with a unique label that can be used to build the relationship among nodes in that XML tree (Bosak & Bray, 1999; Elmasri & Navathe, 2016). Initially, the main focus of the XML research was about handling static documents in terms of data retrieval and navigation. (Connolly & Begg, 2015; Jiang, He, Lin, & Jia, 2011; Kaushik, Bohannon, Naughton, & Korth, 2002; Kaushik, Bohannon, Naughton, & Shenoy, 2002). At present, most XML documents are dynamic and it is essential that they are handled efficiently since most database applications nowadays include XML processing. As a result, labelling XML data has become an important task to improve query processing (Cohen, Kaplan, & Milo, 2010). Many dynamic schemes have been proposed (Amagasa, Yoshikawa, & Uemura, 2003; Cohen et al., 2010; Eda et al., 2004; O'Neil et al., 2004a; X. Wu, M. L. Lee, & W. Hsu, 2004b). However, none of these schemes suit all users' requirements and they are only appropriate for certain circumstances. The main challenges are with dynamic XML data since static XML data has been efficiently processed by proposing a number of successful schemes such as the Dewey scheme and containment scheme (Tatarinov et al., 2002; Zhang, Naughton, DeWitt, Luo, & Lohman, 2001). The case with dynamic XML data is different as the XML databases still struggle to manage large numbers of relabelling cases. Many labelling schemes have been proposed for dynamic XML data (C. Li, Ling, Lu, & Yu, 2005; Liu, Ma, & Yan, 2009; O'Neil et al., 2004a; Xu, Ling, & Wu, 2012; Xu, Ling, Wu, & Bao, 2009). Any efficient labelling scheme should provide effective query performance, labelling XML data efficiently, reducing the required relabelling cases, and determining the relationships. Therefore, this research addressed this issue and came up with a new proposal as a new XML labelling scheme. This thesis proposed and evaluated an XML labelling scheme that was developed to provide an improvement to the query processing, reduce the relabelling to the lowest possible level, and update efficiency.

1.2 Motivation and Challenges

XML data has many features over the relational model. For instance, both the structure and the data are incorporated in the XML document. However, relational data is different from XML data in that the structure and data are separate from each other (Ali et al.,

2006; Luk et al., 2002). Thus, this advantage makes XML a suitable method for exchanging data. Moreover, XML data has a flexible model in terms of querying data which is not available in query languages for relational databases such as SQL (Abiteboul, 1997).

The fast growth in XML databases is a consequence of the high demand for efficiency of XML query. Indexing is a major factor in achieving efficient and scalable query processing by reducing the search space and time. Thus, there is much concern about the XML indexing and labelling schemes. However, the existing XML indexing techniques face many challenges (Klaib & Lu, 2014). Storing XML data is an expensive process since the data is stored randomly in the disk. Also, the data is not regular and some elements are missing or repeated. Consequently, these issues lead to inefficient query performance (Chung, Min, & Shim, 2002). Moreover, as the data and the structure are both integrated in the XML document, this makes navigation through the data tree a complicated and challenging task (Haifeng, Hongjun, Wei, & Ooi).

The diversity of structural relationships between different elements in XML documents is one of the key issues that distinguishes XML data and relational data (Che, Aberer, & Özsu, 2006). Figure 1-1 shows a relational databases model. The most common relationships in the XML data model are, parent, child, sibling, ancestor, descendent relationships. These relationships are used to derive other relationships that are identified by X-Path query language (Anders Berglund, 2015). Figure 1-2 shows an example of XML tree.

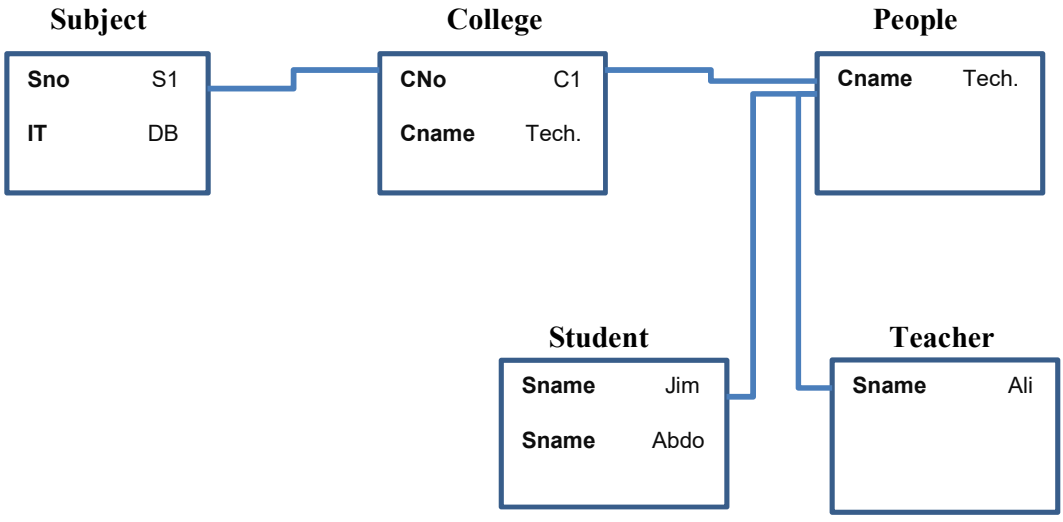


Figure 1-1: relational databases model

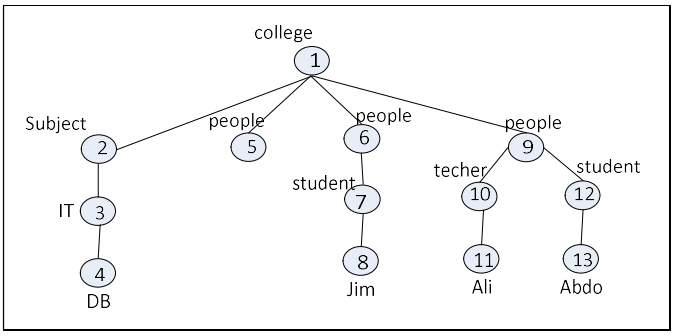


Figure 1-2: XML tree

Furthermore, XML data management is a challenge nowadays because not all data are standard format. Thus, handling this data is a complicated task (Ma, Liu, Hunter, & Zhang, 2010). The growth of XML data leads to the development of XML databases. Managing XML data is required by many applications in order to save their data. Examples of these applications are Microsoft Office and Open Office (Barbosa & Bonifati, 2007).

In general, the efficiency of the performance of any query in a database is based on the indexing. In terms of an XML database, the indexing depends on the labelling scheme (Johnson, Miller, Khan, & Thuraisingham, 2012). Therefore, labelling schemes play the same role that the index plays for relational databases. The key task of the labelling schemes is to identify the structural relationships among the nodes in the XML tree (Sans & Laurent, 2008).

Therefore, an investigation was carried out and determined that one of the most common shortcomings of the existing labelling schemes is the high cost of the updates. To be more specific, the existing labelling schemes lack the ability to adapt deletions and insertions gracefully, even though, many labelling schemes have been proposed to solve these shortcomings. The advantages of one labelling scheme are the disadvantages of another (Klaib & Lu, 2014).

With respect to the classification of labelling schemes, there are three key categories: namely, node indexes, path indexes, and sequence indexes. These categories are explained in the chapter of Previous Work.

Consequently, the main motivation for this research is to improve the performance of XML query processing by developing an XML labelling scheme.

1.3 Main Findings and Contributions

This research contributes to the field of XML data indexing. The main contributions are briefly explained in this section:

1- Study and investigate the XML indexing techniques and determine their limitations. This work was published in a conference (Klaib & Lu, 2014).
2- Propose a novel hybrid XML labelling scheme that presents an improvement to the labelling schemes.
3- The proposed scheme shows improvement on handling both static and dynamic XML documents.
4- The proposed scheme enhances the performance of XML query.
5- The proposed scheme enhances the updating processes by reducing the re-labelling cases to the lowest possible level.
6- The research improves the efficiency of labelling in terms of time and size.
7- The proposed scheme provides fast labelling construction.

1.4 Research Hypothesis:

Based on the research motivation and challenges, the hypothesis that this thesis seeks to evaluate is the following:

H_1: ***Employing a hybrid labelling scheme based on a clustering-based technique improves the efficiency of labelling nodes and query performance of XML data.***

H_0: ***Employing a hybrid labelling scheme based on a clustering-based technique does not improve the efficiency of labelling nodes and query performance of XML data.***

1.5 Aims and Objectives:

The aim of this research is to investigate XML labelling schemes and their limitations and address the challenges and open issues in order to improve the efficiency of indexing XML data.

In order to achieve this aim, the following objectives have been defined:

Objective 1: Carry out a state-of-the-art literature review on the current XML labelling schemes.

Objective 2: Introduce a framework that addresses some limitations and challenges of indexing XML data.

Objective 3: Design and develop this framework.

Objective 4: Evaluate the framework by testing it in order to ascertain whether it has achieved the aim.

1.6 The Hypothesis and The Findings

Four experiments were used to test the hypothesis. The results support the research hypothesis in general. These experiments tested the hypothesis in different aspects.

The labelling nodes experiment illustrated that the hypothesis was supported and the proposed scheme presented better performance in this regard. The second experiment showed that the proposed scheme presented efficient calculation for determining the relationships. The third experiment assessed the query performance using 19 queries used for different purposes. The last experiment presented the ability of the proposed scheme to handle three types of insertions. The results suggest that the proposed scheme shows better performance than the other schemes in this respect. Generally, these results support the hypothesis.

1.7 Research Approach:

1.7.1 Overview

This research started by a state-of-the-art review in XML indexing techniques and their classifications, resulting in a comparison between these techniques which was carried out in order to identify and analyse their pros and cons. Suitable criteria were identified for this comparison. Briefly, these criteria are the following:

1.7.1.1 Measuring how precise and complete the results are.

1.7.1.2 This criterion is used to assess issues associated with the requirement to calculate the relationships among elements; the requirement of structural joins in order to perform a query; and finally, the requirement for further refinement tasks to gain accurate results.

1.7.1.3 Measuring how much the index is scalable. In other words, measure the ability of the index to cope with variety of queries in terms of path lengths.

1.7.1.4 Measuring how many nodes that are updated in order to find out the cost of the update processes.

Having carried out the investigation, there are some limitations and open issues such as the trade-off between the index size and the query performance as whenever the index size increases, then the query performance decreases and vice versa. In addition, the XML data structure is semi-structured and irregular. Therefore, the labelling scheme plays a key role in handling the decrease of the index size and improves the query performance.

Consequently, this research concentrates on improving the performance of XML query processing. As a result, this research proposes a new hybrid approach called clustering-based labelling scheme for indexing XML data.

1.7.2 Methodology of the Proposed Scheme:

Theoretically, the idea of the proposed labelling scheme is to divide the whole data tree into small groups (clusters). This mechanism makes each cluster itself a sub-data tree. The advantage of this mechanism is to reduce the number of required re-labelling cases to the lowest possible level, and as a result improve the efficiency of the query

performance processing. Subsequently, two XML labelling schemes were used to label the nodes and their clusters of a data tree. These two schemes are the Dewey labelling scheme which was used to label the clusters, and the LLS labelling scheme which was used to label the nodes of the data tree.

1.7.3 Data Model and labelling XML Document

The implementation mechanism of this scheme is divided into two stages, see below:

1.7.3.1 Creating and Labelling Clusters:

All nodes are grouped into clusters as follows:

a- Each node and its child nodes are gathered to build a cluster.
b- Only the main root of the document is considered as a cluster itself and without child nodes. This node is labelled number (1).
c- Each cluster is considered a sub-tree.
d- A cluster can have only one or two levels of nodes.
e- Each cluster has at least one node.
f- Each cluster may have a root and child node(s) that is connected with this root. Thus, a cluster must contain at least one node.
g- All clusters, including the main root (not each node), are labelled by the Dewey labelling scheme.

1.7.3.2 Labelling Nodes:

a) Each cluster is treated as a sub-tree and its nodes are labelled separately from other clusters.
b) The LLS labelling scheme is used to label the nodes.
c) If the root of a cluster is a child node of another cluster, which means that this root has already been labelled, then the label of this root will be kept the same.

More description is to be in the following chapter.

1.8 Thesis Structure and Organisation

This section discusses the thesis structure. The thesis includes six chapters. The outlines of these chapters are as follows:

Chapter 2 Extensible Markup Language (XML): This chapter discusses the background of XML technology. An introduction is provided. XML overview and Origins is described including XML syntax, XML document format, XML advantages and disadvantages, and XML structure are explained. Subsequently, many XML technologies are discussed in detail such as: DTD, SAX, XML schema, XML DOM. Furthermore, XML Query languages were described. Data Model is discussed including types of Data model.

Chapter 3 Literature Review and Previous Work: This chapter discusses an overview of labelling schemes and existing approaches in detail. Significance of XML labelling schemes and critique of XML labelling schemes are also explained.

Chapter 4 Methodology and Framework: This chapter discusses the existing approaches and the approach of this research and then determines their advantages and disadvantages. This research approach is explained in detail in this chapter. The structure and labelling nodes technique of the proposed scheme are explained. The design and implementation of the proposed scheme are illustrated in detail. Experimental setup is also explained, including the experiment objectives and types, XML datasets and Benchmarks. Subsequently, data analysis and presentation are discussed in detail, including data generation, and the data analysis approach used for this research is explained.

Chapter 5 Results: This chapter presents the results of the experiments in detail. The results are divided into experiments for static documents and experiments for dynamic documents. The results of the experiments for static documents includes the three experiments used for this part. The results of the experiments for dynamic documents also comprise of inserting new nodes, determining relationships, and query performance for dynamic.

Chapter 6 Discussion: Assessment for the results is provided. The main findings of this assessment are explained. The main advantages and limitation of the proposed scheme are illustrated. The discussion and critical assessment of the results are discussed. The research hypothesis is evaluated in this chapter. The strengths and limitations of the proposed labelling scheme are explained. Finally, the experimental findings are outlined.

Chapter 7 Conclusion: This chapter starts by explaining the introduction chapter in brief. It goes on to explain and summarise the literature review. The Methodology is also illustrated in brief. The Background of XML chapter is reviewed. The Results and Discussion chapter is outlined and the results discussed briefly.

1.9 Summary

This thesis addresses the challenges and problems of indexing XML data. Thus, the problems and limitations of the labelling schemes were reviewed and considered. This chapter provided an introduction to this thesis. Subsequently, the motivation and challenges for this research were addressed; the research hypothesis was identified; the aims and objectives of this research were explained; the research approach was described; and lastly, the thesis organisation was summarised.

2. Extensible Markup Language (XML)

2.1 Introduction

The first use for the XML data was by the W3C to share data among applications and consequently different applications and platforms can communicate much easier (Abiteboul et al., 2000). Since that time, XML is getting more popular for many advantages that this technology offers such as simplicity, supporting data manipulation and data retrieval. As a result, the applications and technologies that rely on XML have increased rapidly (Bray, Paoli, Sperberg-McQueen, Maler, & Yergeau, 2006). As far as the database is concerned, the XML data management has been improved to adopt this spread of XML data. Therefore, this chapter demonstrates an overview of XML fundamentals and the associated technologies such as XML databases, SAX, DTD, and XML schema. This chapter also provides a brief history of XML overview and origins includes XML syntax, XML document format, XML advantages and disadvantages, and XML structure. Section 3.4 discusses XML technologies such as Document Type Definition (DTD), Simple API for XML (SAX), XML schema, and XQuery. Subsequently, section 4.4 discusses XML parsing. Section 4.5 explained XML DOM. XML databases are illustrated in section 4.6. the most common Query languages are discussed in section 4.7. XML schems and data models are explained in sections 4.8 and 4.9 respectively.

2.2 XML Overview and Origins

The World Wide Web Consortium (W3C) recommended the XML in 1998 as a method for saving, retrieving, and exchanging data on the internet. XML has many advantages such as it's a flexible language; so, developers can create their own tags. XML data are portable format. XML is a very simple language. XML is a readable data by most platforms. XML is originally a part of the document community rather than the database one (Elmasri & Navathe, 2016).

2.2.1 XML syntax

The rules of the XML are very logical and simple. XML files contains many parts such as elements, attributes, and comments (Abiteboul et al., 2000).

2.2.1.1 Elements

The element is the essential part and represents the basic component of the XML document. The element contains two tags, namely the opening one and the closing one. Between these two tags is the value of the element. This value might be other elements, or text, or both elements and text. Moreover, elements can include attributes, such as < people teacher = "Ali"> from the figure 2-1, or can be void

(Abiteboul et al., 2000). The elements in a XML document have to be balanced so each opening tag has a closing tag (Abiteboul et al., 2000).

```
<College>
  <subject>
      <tutor>Dave</tutor >
  </ subject >
  <people> </people>
  <people>
      <student>Jim</student >
  </people>
  <people>
  < people  teacher = "Ali">
     <student > Abdo </student>
  </people >
</College>
```

Figure 2-1: An example of XML file

2.2.1.2 Attributes

The purpose of the attributes is to give extra details regarding the element. Generally, this kind of elements is fixed. In addition, the attributes are located into the opening tag and contains the name and the value of the attribute. The value of the attribute is located into two bracket s as this example shows < people teacher = "Ali"> from the figure 2-1. Reference attributes are utilized to refer and link to other elements or the element itself (Abiteboul et al., 2000; Tidwell, 2002).

Attributes are different from elements as the attributes are suitable for static and certain values, whereas the elements are suitable for dynamic values. Attributes are hard to include sub-elements. However, elements can include values and sub-elements. Therefore, it is better to reduce the attributes to the lowest possible level(Abiteboul et al., 2000).

2.2.1.3 Comments

Even though XML is simple and readable, comments are useful for clarification purpose. Providing notes and explaining any complicated code can be achieved by adding comments for the users and developers. Despite the position of the comment is not specified, it must be between tags like this example <!-- comment -->. Comments are not processed by applications (Connolly & Begg, 2015).

2.2.2 Ordering

Elements in XML document must be in order, whereas the attributes do not need to be in order. Figure 2-2 shows two fragments of XML (a) and (b) which are not equal as the elements are in different orders. Figure 2-3 illustrates two fragments of XML (a) and (b) which are equal even the attributes are in different orders (Abiteboul et al., 2000).

<contact> <telNo>457855665 </telNo> <email> sample@gmail.com</ email > </contact >	<contact> <email> sample@gmail.com</ email > <telNo>457855665 </telNo> </contact >
(a) XML elements	(b) XML elements

Figure 2-2: XML elements – (a) not equal (b)

<Name> Fname=Jim" Lname= "Smith" </Name >	<Name> Lname= "Smith" Fname=Jim" </Name >
(a) XML attributes	(b) XML attributes

Figure 2-3: XML elements – (a) equal (b)

2.2.3 XML Document Format

Despite that XML is simple and easy for developers to make their XML documents, still need to obey rules such as:

- There is only one root for a XML document.
- Both starting and ending tags have to match and be case sensitive.
- Brackets are used for the values of the attributes.
- Elements need to be in order, so last opened tag is first closed tag.
- The name of an attribute into an element should not be repeated.

The well-formed XML document must be formatted according to these rules in order to be accepted and processed (Abiteboul et al., 2000; Tidwell, 2002)

2.2.4 XML Advantages and Disadvantages

XML has become a very interest for broadly users and organizations since was recommended in 1998 because of the strengths that it have such as:

- Simplicity: as a clear and plain text style.
- It's independent and standard platform: accepted by most platforms and applications.
- Widely accepted data format: XML is used broadly and can be processed over many different devices such as mobile phone, computers, and tablets.
- Extensibility: XML files can be expanded by adding new tags.

- XML is readable for both human and machine.

Despite these advantages which make XML a suitable method to exchange data over the internet, there are disadvantages which as follows:

- XML does not base on mathematical and theoretical basis as the relational model does.
- XML is connected somehow with many technologies that cause complexity to the use of XML. Examples of these technologies are DTD, XQuery, SAX, XML Schema, DOM, XPath, etc.

To conclude, XML is a strong and widely used technology in spite of the weaknesses of it (Samir Mohammad & Martin, 2009).

2.3 XML Structure

The XML document is illustrated by a tree graph. The tree demonstrates the XML document's structure. The components of an XML document called nodes. These components include elements and attributes. This tree must have a root element which makes the start of the tree. Subsequently, this tree has sub-trees. This tree ends with leaf nodes. (Harold & Means, 2004; Erik T. Ray, 2001).

The tree also used to distinguish the links between the nodes such as parent and sibling relationships. The root node is considered a parent, and all other nodes linked with this node and located in the lower level called child nodes. all nodes that in the same level are considered siblings (Erik T. Ray, 2001). Despite the fact that this structure of the tree is a good advantage as it facilitates the access to the data very easily, it has the problem that it occupies huge space of memory (Tidwell, 2002). The tree graph for the XML file in figure 2-1 was illustrated in the figure 2-4.

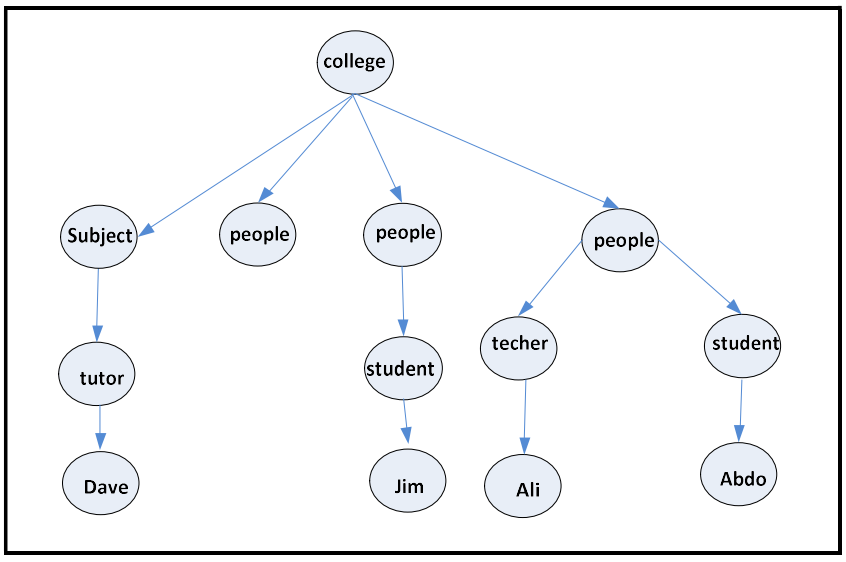


Figure 2-4: the tree graph for the XML document in figure 2-1

2.4 XML Technologies:

The XML surrounded by variety of technologies that linked with it such as DTD, XML schema, SAX, XPath, XQuery, and so on. Some of them are explained later on in this chapter and some are discussed below:

2.4.1 Document Type Definition (DTD): as its name shows, it is used to define the format of XML documents, and it supports one kind of data. The DTD includes the names and order of the elements, attributes, the links between elements, and the way of the elements arranged. (Connolly & Begg, 2015; Elmasri & Navathe, 2007). The DTD can be saved in the XML document or either in a separate file. The DTD is similar to the schema for relational databases in terms of the functionality (Abiteboul et al., 2000).

2.4.2 Simple API for XML (SAX): it is used for parsing XML documents. SAX uses different way from the DOM for parsing XML documents. SAX informs the applications by a stream of parsing processes. The SAX is a good choice for big documents as the whole file does not need to be loaded in the memory (Tidwell, 2002; Vakali, Catania, & Maddalena, 2005).

2.4.3 XML Schema: it was recommended by the W3C in 2001 as a tool in defining the structure of XML documents. It was developed to address the disadvantages of the DTD(Connolly & Begg, 2015). The definition of XML schema provides the structure in terms of the configuration and the kind of data. The schema itself is provided as XML document and it is exactly the same as the XML in terms of its processing, editing, viewing and the tools in performing these operations (Abiteboul et al., 2000). The XML schema supports different types of data and its very expressive scheme, therefore, it is better than the DTD (Connolly & Begg, 2010).

2.4.4 XQuery: the XML query language (XQuery) was recommended by the Query Working Group at the W3C. The XQuery is a development on the Quilt and is similar to the SQL in terms of representing the query. XQuery has the exact path expression in the XPath. In addition, it is a case sensitive language (Al-Badawi, Ramadhan, North, & Eaglestorne, 2012).

2.4.5 XML Parsing

The parser performs a significant task in processing XML documents. Thus, all XML applications are combined with a parser. The purpose of the parser is to split up the XML document and make a representation in the form of a tree or stream. Examples of the parsers are the following: SAX, JDOM, Xerces2, and, DOM. These parsers build XML files (Connolly & Begg, 2015).

2.4.6 XML DOM

Document Object Model (DOM) was recommended by the W3C for handling XML documents. DOM makes an XML document as a tree and stores it in a memory. This tree consists of objects and demonstrates the relationships between them. Every element is represented as a node in the tree (IG, 2005). The DOM has advantages such as; DOM handles XML data easily. DOM eases the access and update processes of data. In addition, DOM offers reading and writing synchronously. In terms of data access, DOM allows random access. Moreover, DOM is a proper environment for XPath and dealing with queries and updates properly. It is also appropriate for number of platforms such as C++, NET, and JAVA. Nevertheless, storing the XML tree in memory occupies big space and takes long time. Therefore, handling big XML documents causes problems (Al-Badawi et al., 2012; Anders Berglund, 2015; Frank, Apel, & Schaebec, 2003).

DOM provides an *Interface* for the *Node* which includes attributes and elements. This I Interface is incorporated when the XML document is processed. There are number of methods which are offered by this Interface such as 'parentNode()', 'childNode()'. The function of the parentNode() is to return the parent of a node (IG, 2005).

2.4.7 XML Databases

XML files are classified into two sorts namely data-centric and document centric. Data-centric is saved in databases since it is highly structured. However, document centric is semistructured since it is textual content (Sun & Wang, 2012).

Since XML is similar to other types of databases in terms of storing and retrieving data, it is understood that XML is a technology can be used to construct databases (Connolly & Begg, 2010; Sun & Wang, 2012). XML has many features of databases such as saving data in XML documents, holds XML schema, DTD, and query languages. Also, XML offers interfaces for instance SAX and DOM. However, XML has shortage of some properties of database systems as recovery system, multiple accesses, and security system (Steegmans, 2005). These disadvantages raise an enquiry if XML can be considered database or not. Thus, heavy researches have been carried out to enhance the XML database.

XML databases are classified into two categories namely enabled XML databases and native XML databases (NXD) (Steegmans, 2005). Enabled XML databases based on the relational database for saving data. The advantage for using this kind is to support the already present applications as huge XML data exist in relational database(El-Aziz & Kannan, 2012; Papamarkos, Zamboulis, & Poulovassilis, 2008). The mapping technique is a standard method that used for converting data from XML form to relational one. Nevertheless, this mechanism has shortcomings such as it is unable to manipulate big XML documents due to the number of joins (Papamarkos et al., 2008). Moreover, this technique does not handle properly the hierarchical structure, nested data, and elements

sequence. Another problem is that information might be lost through the conversion process (Steegmans, 2005; Sun & Wang, 2012).

The second type of XML databases is the native one. This type offers a method to create, manipulate, save, and retrieve XML files. The NXD can be easily looked up and its whole data can be handled since the NXD is compacted (Sun & Wang, 2012). In addition, NXD supports XML query languages; so that improves the efficiency of data retrieval (Steegmans, 2005). NXD is much flexible than enabled XML database. However, the key shortcoming of NXD is that it is not able to handle data in other format than XML (El-Aziz & Kannan, 2012).

NXD are classified into two categories; namely text-based and model-based (Papamarkos et al., 2008). Text-based method supports the XML file as text and save it in a file system in relational database. However, the model-based deals with XML data as objects and represents the file as tree (Steegmans, 2005).

2.4.8 XML Query Languages

XML data can't be queried by SQL since XML data is similar with semi-structured data. Thus, certain query languages used for XML such as XPath and XQuery. More details are below:

2.4.9 XPath

It is an XML Path Language (XPath) and was recommended by the W3C to be used for querying XML data. As its name indicates, path notation is used for navigating cross the XML document. The XPath handles the elements and attributes of an XML document by using a simple syntax. The mechanism of XPath is to recognise the start node and the end node. This path called location path (Anders Berglund, 2015). There are number of relationships that the XPath defines; namely parent, child, sibling, ancestor, and descendant. Regarding the parent and child relationship, it is between two nodes in two neighbouring levels. Parent can have unlimited number of children whereas a child can have only one parent on the above level. Nodes at the same level and sharing the same parent called siblings. Any relationship that moves up from a node to any other node till the root called ancestor. Descendant relationship moves down from a node to any other node till the leaves (Elmasri & Navathe, 2016; Garcia-Molina, 2008). XPath expressions provide the position of elements and attributes in XML document. The XPath syntax comprises number of special marks that used for defining the nodes (Garcia-Molina, 2008; W3C, 2016). The axis indicates the way of the navigation. The 'node test' denotes the node kind in the file(Erik T. Ray, 2001). Table 2-1 illustrates number of types of axis linked with XPath.

TABLE 2-1: XPATH AXIS (W3SCHOOL, 2016)

Axis name	Results
ancestor	Selects all ancestors (parent, grandparent, etc.) of the current node
ancestor-or-self	Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself
attribute	Selects all attributes of the current node
child	Selects all children of the current node
descendant	Selects all descendants (children, grandchildren, etc.) of the current node
descendant-or-self	Selects all descendants (children, grandchildren, etc.) of the current node and the current node itself
following	Selects everything in the document after the closing tag of the current node
following-sibling	Selects all siblings after the current node
namespace	Selects all namespace nodes of the current node
parent	Selects the parent of the current node
preceding	Selects all nodes that appear before the current node in the document, except ancestors, attribute nodes and namespace nodes
preceding-sibling	Selects all siblings before the current node
self	Selects the current node

The following examples of XPath expressions illustrate the way that these axes can be used. These examples are from the figure 2-1 for an XML document.

/college/people/student

The *student* node can be accessed by using this expression. *Student* node is a child node of the *people* node and descendent of *college* node.

/college/people@teacher

The identifier (teacher) attribute of *people* node can be chosen by this expression.

To find certain nodes; predicates are used and put in brackets (Elmasri & Navathe, 2016; Erik T. Ray, 2001). The following example shows how to find all students who their names “Ali”.

College/people

Moreover, axes are used to recognise number of nodes that linked with a certain node (Garcia-Molina, 2008; Harold, Means, & Udemadu, 2004; Erik T Ray, 2003).

There are two kinds of XPath expressions namely relative path and absolute path. The absolute path for a certain node is the full path from the root till the node itself(Harold & Means, 2004; Erik T Ray, 2003).

2.4.10 XQuery:

The XML query language (XQuery) was recommended by the Query Working Group at the W3C. The XQuery is a development on the Quilt and it is similar to the SQL in terms of representing the query. XQuery has the exact path expression in the XPath. In addition, it is a case sensitive language (Al-Badawi et al., 2012). XQuery has exactly the same path expressions with the XPath (Al-Badawi et al., 2012). Generally, the expression result is a group of ordered nodes. Nevertheless, results can be influenced by redundancy because of the duplication of some nodes that hold the same name and kind. Moreover, XQuery is a very flexible language and able to handle complex queries. It is offers FLWOR expression which is abbreviation from FOR, LET, WHERE, ORDER BY, and RETURN clauses. This expression is used to extract data as the SQL does. This expression must start by either FOR or LET; whereas the WHERE and ORDER are non-compulsory clauses. This expression must end by RETURN (Connolly & Begg, 2015; Elmasri & Navathe, 2016; Garcia-Molina, 2008). More details about the FLWOR expression clauses as follows:

- **FOR and LET Clauses:** these clauses link values and variables. FOR clause used to create a loop for repeating steps; whereas LET clause used to identify a variable (Connolly & Begg, 2015).
- **WHERE Clause:** it is used to designate a condition or more in order to reduce and control the result generated by FOR and LET (Connolly & Begg, 2015) .
- **RETURN and ORDER BY Clauses:** every expression has to have a RETURN clause. This clause evaluates every tuple whereas the FLWOR expression result is provided by the grouping all evaluations. With regard to the ORDER BY clause, it is used to sort the sequence of the tuples (Connolly & Begg, 2015).

2.5 XML Schemas:

Schemas are a key issue in XML field. Initially, the W3C recommended the XML schema language in 2001 in order to overcome the disadvantages of the DTD. Schema is defined as a static description for a database that is addressed during the database design phase. XML schema's definition provides the structure of a XML document with respect to configuration and data kinds (Connolly & Begg, 2015; Elmasri & Navathe, 2007). The XML schema is provided as an XML document and it is exactly the same as XML in terms of editing, processing, viewing, and also the tool used to perform these actions. (Abiteboul et al., 2000; Connolly & Begg, 2010). In terms of XML, a schema is intended to save the structure of the file and illustrates the relationships among elements. There are

different kind of schemas used with XML files such as XML schema, RELAX NG, DTD, and Schematron. The most common used schemas are the DTD and XML schema. However, schema is better than the DTD with regard to supporting different kinds of data, domain of the values, and the number of repetition of an element in an XML file (Abiteboul et al., 2000; Elmasri & Navathe, 2016; Fallside & Walmsley, 2004; Garcia-Molina, 2008; Lee & Chu, 2000).

Data Model : the data models for hierarchical structure are classified into four categories based on the structure form as follows (Gou & Chirkova, 2007) :

2.5.1.1 Edge-Labelled

The edges of a XML document represent the nodes which are either elements or attributes. Figure 2-5 shows an example for an XML document. Figure 2-6 illustrates the edge-labelled for the document in the figure 2-5. For instance, *tutor* is an element and *teacher* is an attribute. There are leaf nodes which considered as elements and attributes' values. For instance, *Ali* is a value for the *teacher* attribute and *Abdo* is a value for and *student* element.

```
<College>
  <subject>
      <tutor>Dave</tutor >
  </ subject >
  <people> </people>
  <people>
      <student>Jim</student >
  </people>
  <people>
  < people  teacher = "Ali">
      <student > Abdo </student>
  </people >
</College>
```

Figure 2-5: An example of XML file

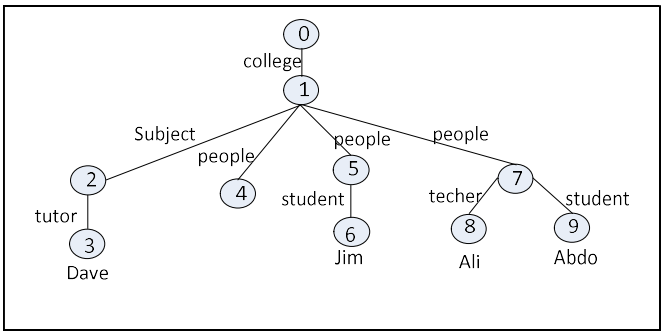


Figure 2-6: Edge-labelled data tree for the XML document in figure 2-5

2.5.1.2 Node-labelled

This model has three parts namely element, attribute, and values. Figure 2-7 shows an example of this model for the XML document in figure 2-5. Nodes in this model represent elements. Unlike the edge-labelled model where the nodes are edges represent elements.

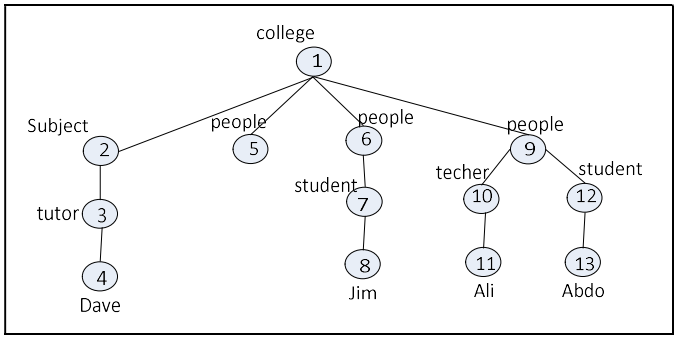


Figure 2-7: Node-labelled data tree for the XML document in figure 2-5

2.5.1.3 Directed Acyclic Graph (DAG)

This model uses the technique ID/IDREF token which is available in the XML language through DTD. This technique used to recognise the sort of an attribute's element. Figure 2-8 is a revised issue of the XML document in figure 2-5.

```
<College>
   <subject ID=1>
         <tutor>Dave</tutor >
   </ subject >
   <people reference=1> </people>
   <people>
         <student>Jim</student >
   </people>
   <people>
   < people teacher = "Ali">
        <student > Abdo </student>
   </people >
 </College>
```

Figure 2-8: the modified XML document with ID/IDREF

The idea of the ID/IDREF is to allow more than one element to refer to a certain node. For instance, node number 2 in figure 2-9. This token plays the role of the keys in relational databases.

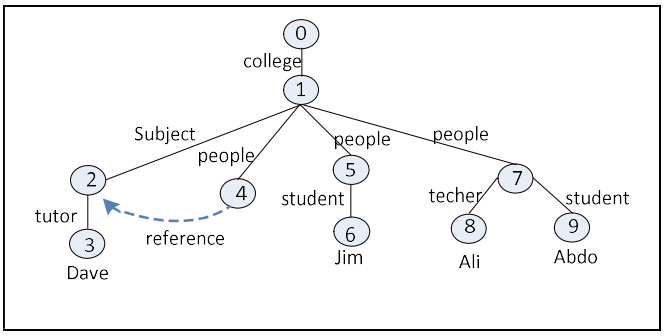


Figure 2-9: Directed acyclic graph

2.5.1.4 Directed Graph with Cycles

This data model is based on the technique that adding another IDREF from an element to another element which creates a circle. Figure 2-10 shows an example.

```
<College>
   <subject ID=1 recommend 2>
         <tutor>Dave</tutor>
   </subject >
   <people ID=2  reference=1>
</people>
   <people>
         <student>Jim</student >
   </people>
   <people>
   < people  teacher = “Ali”>
        <student > Abdo </student>
   </people >
 </College>
```

Figure 2-10: modified XML document with ID/IDREF

An IDREF was added from the *subject* element, in the second line of the figure 2-6 as “recommend=2”, to the *people* element, in the fifth line of the figure 2-10 as “ID=2”, which creates a circle. This model is common in XML. Nevertheless, this technique makes the XML query processing more complicated. Figure 2-11 shows this model.

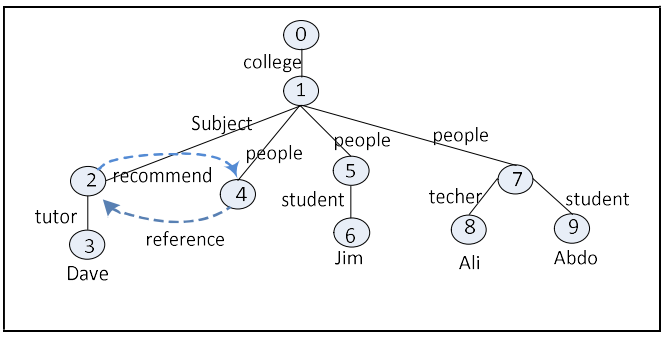


Figure 2-11: directed graph with cycles

2.6 Summary

To summarise, the XML technology is very wide area and covering all of it is beyond this thesis's scope. Thus, the most essential and relevant issues of XML technology were discussed in this chapter. Nevertheless, the covered topics are sufficient and provided a comprehensive review. In more details, the chapter started by considering the history of the origin and development XML. of XML and its and structure. A review inside the XML and its components and organisation has been explained in details such as XML syntax, elements, attributes, and comments. This review gives the researcher an essential and comprehensive understanding for XML structure and history which consequently assist in fulfilling the literature review and the rest of this research. Thus, this review is considered a task to achieve the objective number one.

The advantages and limitations of XML have also been discussed in order to know any general issue that might affect the labelling scheme positively or negatively. Next, the XML structure is an essential topic that any researcher involves in this area needs to know. Thus, a brief description of the XML structure was discussed.

Furthermore, there are many sophisticated technologies around the XML. These technologies conduct different functions. For instance, DTD, SAX, XML schems and so on. Study the most common used technologies and use the relevant ones to this research is a significant task for this research as there is a need to use some of them.

XML parsing is an important task that all XML applications need to accomplish. All XML application have a parser. This research used the DOM to parse XML documents. As a result, sections 4.4 and 4.5 have discussed the XML parsing and XML DOM.

Similar to relational databases, XML database is a crucial part as used for data storage, data retrieval, and data management. Section 4.6 has discussed XML database and its features and differences from traditional databases, as well as the XML databases classifications. Next, XML query languages have been explained in section 4.7. for instance, XQuery and XPath are common used languages in this regards. This research needs to use an appropriate query languages in order to evaluate the proposed scheme. Thus, the XPath was selected for testing purpose. XML schemas is a description of a database. Therefore, it is an important issue that has been discussed and considered in section 4.8.

Finally, data models for hierarchical structure have been discussed in details in section 4.9. it is a very important and relevant topic for this research. The classifications of data hierarchical structure give an understanding for the differences between these structures and their advantages and disadvantages. This understanding assists the researcher in studying the literature review of the existing labelling schemes.

Consequently, this chapter provided wide discussions for relevant fundamental topics. All these topics are linked to the objectives of this research, and therefore, provide a comprehensive understanding for both the researcher and reader.

3. Literature Review and Previous Work

3.1 Introduction

Due to the increase in the importance of XML data management, many researches focus on this area. In addition, labelling schemes are an interesting topic for XML data developers and researchers as this area plays a key role in improving query performance. In order to query XML data, an efficient XML labelling scheme is needed (Edith, Haim, & Tova, 2010). However, even though heavy research on labelling schemes and techniques has been carried out, there is as yet no appropriate labelling scheme for all users' requirements (B. Catania, Maddalena, & Vakali, 2005). Thus, this research focuses on this area to develop a framework that resolves some of the challenges that these existing labelling schemes face.

Therefore, this chapter starts by presenting an overview of the existing labelling schemes, and discusses the existing approaches of labelling schemes and their classifications. The criteria for the evaluation of the labelling schemes used in this research are demonstrated. The classifications of the approaches of labelling schemes based on these criteria are also explained, and the advantages and disadvantages of these approaches are discussed. The significance and critique of the labelling schemes are also explained. The chapter also considers the importance of using the Dewey and LLS labelling schemes. In addition, a review of the testing of the existing labelling schemes is demonstrated, and finally, a summary of the chapter is presented.

3.2 Overview of Labelling Schemes

A considerable amount of research has been conducted on querying and storing XML data since the importance of XML data management has been increasing dramatically. Thus, there is a demand for and focus on producing an efficient labelling scheme. Thus, enhancing XML efficiency by developing a robust XML scheme is mostly achieved by cutting the cost of data searching. The main purpose of the labelling scheme is to encode the information about the XML tree and structure in a very compacted label. The compactness of the labels and the performance speed of the scheme are the key metrics for any labelling scheme. A significant amount of research has been carried out to produce an efficient XML labelling scheme (Elmasri & Navathe, 2016; Garcia-Molina, 2008; Harold & Means, 2004; Harold et al., 2004; Wyke & Watt, 2002).

XML sets the standard rules which are readable by both machine and users. XML has been recognised as a standard means of exchanging data on the WWW. Problems were expected to arise in the use of XML. Eventually these problems appeared and are mostly concerned the query performance (Amato, Debole, Zezula, & Rabitti, 2003; Bruno, Koudas, & Srivastava; B. Catania et al., 2005; Liu et al., 2009); (Abiteboul et al., 2000).

The query definition in this context is to allow users to access the XML data. Consequently, many XML query languages have been proposed. Some of these languages have been developed by the W3C group such as XQuery and XPath (Rousseeuw, Ruts, & Tukey, 1999). These languages have features that assist in overcoming the disadvantages of XML. However, XML query performance still needs to be improved in order to always cope with the growth of the web. Therefore, developing an efficient indexing system has been carried out for many years (O'Neil et al., 2004a, 2004b). The existence of labelling schemes is an important issue for XML indexing sin ce it allows queries to avoid going through the whole XML document as Murata et al. said (Murata, Laurent, & Kohn, 2000). T

hey also stated that it is a significant step to assign a label for each node in the XML document in order to be able to determine the relationship among these nodes. The function of the nodes is to play the role of unique IDs for the components on the XML documents. Thus, labelling is required in order to execute structural queries which can only be performed by using the index (C. Li et al., 2005). As a result, the query process does not require access to the whole XML document, so that makes the query process efficient and quicker (Barbara Catania, Ooi, Wang, & Wang, 2005). The size of the index is mainly based on the size of the labels; thus, many researches have concentrated on proposing labelling schemes to achieve this goal. Some other researches have proposed schemes that were developed to assist in path indexing and numbering schemes to ease the XML query.

Some other schemes have focused on the ease of the function. For instance, a scheme was proposed by Bruno (Bruno, Koudas, & Srivastava, 2002) to ease the compact method of the results of a query path and then the gathering of these paths produces the final combination for a twig query. This scheme was enhanced by Lu (W. Wang et al., 2005) to support queries efficiently. The improvement was on handling twig queries.

3.3 Existing Approaches

XML labelling schemes can be classified into different categories based on the criteria and aspects in which they are evaluated. First of all, this classification is based on the position or location of the index residence. The position can be either in the main memory as a temporary index or on the hard disk, called a disk-based index (Samir Mohammad & Martin, 2009). The former has the advantage of fast response as it avoids the input and output expenses. However, it has the disadvantage that it lacks scalability for large index files. Second, another category was classified by Sans and Lauren (Sans & Laurent, 2008) into two classes, namely interval scheme and prefix-based scheme, also called Dewey schemes. This classification was improved with the addition of two other schemes, namely multiplication-based schemes and Vector-based labelling schemes (Almelibari, 2015). Third, in this category, classification is based on the type of document that is indexed. There are two classes, namely, the index data-centric

document, and the index document-centric XML database (Gang, Chirkova, & Chirkova, 2007). Four, indexing schemes can be evaluated on the basis of the structural relationships in the index. The classification of this category is usually divided into three classes, as follows: 1) node indexes; 2) path indexes; 3) sequence indexes (Bruno et al.; Edith et al., 2010). Consequently, the former classification which was based on the structural relationships is the most relevant one for this research as the structure of the index and the relationships between the nodes are the main concern. Therefore, a deep review and evaluation were carried out in this research as shown below.

3.3.1 Common Criteria for the Evaluation of Indexing Labelling Schemes

The ideal method for evaluating a XML indexing scheme is to compare it with other schemes using some of the criteria that are suitable for all such schemes. There are a number of common criteria that are used to compare indexing schemes such as: accuracy, adaptability, precision, scalability, response time, and type of supported queries (Samir Mohammad & Martin, 2009). This research used the most relevant of them in order to evaluate the structural labelling indexes. These criteria were also chosen as the most helpful ones for users to select the best labelling scheme for their requirements. The selection is carried out by determining the features that these labelling schemes support. The following are these criteria:

3.3.1.1 Retrieval Power: This means the precision of the index scheme is based on the completeness and accuracy of the result. In other words, how much the return results of the labelling scheme are precise and complete.

3.3.1.2 Processing Complexity: This step covers a few issues, such as the requirement of structural joins. In order to improve the performance of a query operation; there is a need to minimize the number of joins. Other issues include the processing cost, and the need to compute the relationships between elements.

3.3.1.3 Scalability: Large indexes involve many input and output operations. Thus, this increases the query processing time.

3.3.1.4 Update Cost: There are two kinds of updates, namely, inserting a node and inserting a subtree. The nodes in a tree index need to be kept organized in a particular way to reflect all kinds of relationships. These relationships have to be preserved if a new node is inserted into the tree. Thus, the index has to reflect its location with respect to these relationships, which makes the case more complex, especially if the scheme has no spaces for the new node.

3.3.2 The Structural Relationships-based XML Labelling Schemes

This section reviews and discusses this category of XML labelling scheme for the most common ones as follows:

3.3.2.1 Node Indexing Schemes:

Node indexes store values that represent the locations of the nodes of the XML tree structure. These values are used to find a particular node's parents, child, sibling, ancestor and descendent. These values are represented by numbers and used to resolve simple and twig queries. Generally, the most commonly used node schemes are the interval (also known as region) labelling scheme and the prefix (also known as path) scheme. Further details about these schemes can be found in the following sections (Al-Badawi, North, & Eaglestone, 2007; Al-Badawi, North, & Eaglestone, 2010; Al-Badawi et al., 2012; M. El-Sayed, K. Dimitrova, & E. Rundensteiner, 2005; M. El-Sayed, K. Dimitrova, & E. A. Rundensteiner, 2005; D. Fisher, Lam, Shui, & Wong; Härder, Haustein, Mathis, & Wagner, 2007; Q. Li & Moon, 2001; Maghaydah & Orgun) (S. Mohammad, Martin, & Powley).

3.3.2.1.1 Prefix Labelling Scheme:

This type of scheme generates code containing two fragments which are the prefix part and the actual-code. The prefix part encodes the previous node code which is the parent of the node. The actual-code encodes the order of the node in the tree. There are many examples of this scheme, and the most popular on e is the Dewey labelling scheme. This labelling scheme has two parts in each node except in the root node. This code is called the Dewey code (D. K. Fisher, Lam, Shui, & Wong, 2006; Lu & Ling, 2004; Scott & SCOTT, 1998). The code at each node has two parts. The first part is an increasing number that reflects the location of the node. The second part is the Dewey code which is the parent's code. These parts are separated by a dot like this ".". The root code has only one part since it has no parent. Figure 3-1 shows an example of a Dewey labelling scheme (Härder et al., 2007; Lu, Ling, Chan, & Chen, 2005; Scott & SCOTT, 1998; Tatarinov et al., 2002; Wei, Haifeng, Hongjun, & Jeffrey Xu).

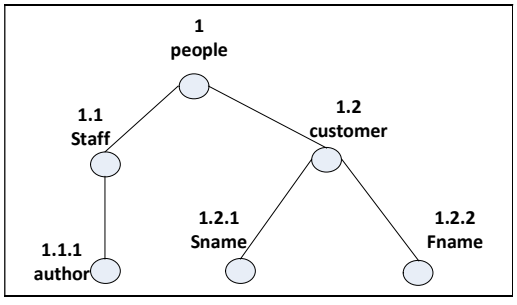


Figure 3-1 Dewey Labelling Scheme

A number of researchers (D. Fisher et al.) developed a dynamic labelling method that can be used with Dewey labels with identifiers of size 0(log n) where "n" is the size of the database. All labelling schemes including Dewey need 0(n) bits per label (Edith et al., 2010). Many prefix schemes have been developed such as:

Dewey encoding (Tatarinov et al., 2002). Other examples of prefix labelling schemes are the following: Labelling Scheme for Dynamic XML Data (LSDX) developed by Duong and Zhang (Duong & Zhang, 2005). In this method, the numbers and letters are combined. This scheme supports the ancestor/descendent relationship and the sibling between nodes. Lu and Ling (Lu & Ling, 2004) proposed a labelling scheme called GRoup base Prefix (GRP) which contains two parts. The first part is the group ID. The second part is the group prefix. Both the Labelling Scheme for Dynamic XML data (LSDX) and GRP schemes are firm and immutable as the label sizes of these schemes can reach 0(n) bits per label. The ORDPATH labelling scheme was developed by a number of researchers (O'Neil et al.). The Dewey labelling scheme is a prefix labelling scheme that was developed by Tatarinov and colleagues (Tatarinov et al., 2002). There is a big similarity in terms of the structure between the Dewey labelling scheme and other prefix labelling schemes such as ORDPATH and LSDX (Almelibari, 2015).

Some researchers (C. Li & Ling, 2005) proposed a labelling scheme called *ImprovedBinary* as an enhancement to the LSDX and other schemes that were developed by Duong and Zhang (Duong & Zhang, 2005). More details about the Dewey labelling scheme are presented later on in this chapter.

- **Advantages of Prefix Labelling Scheme**

The prefix labelling schemes have many advantages such as supporting the query processing of XML data. This type of schemes was defined by researchers (Alstrup & Rauhe, 2002) as perfect schemes for the fast update of XML data. Other researchers stated that the majority of prefix schemes are able to represent ancestor–descendant relationships along with the sibling relationship between nodes (Amato et al., 2003; Yun & Chung, 2008). Knowing the depth of the XML tree is a significant factor for improving the query processing (Meuss & Strohmaier, 1999). Moreover, the unique code given to each level of the nodes in the LSDX labelling scheme is used to ascertain the depth of the XML tree (Duong & Zhang, 2005). This feature provides this scheme an efficiency in terms of retrieving and updating data (Hou, Zhang, & Kambayashi, 2001). In addition, the Dewey labelling scheme can define the path between the root and the node as the label has the parent label (Duong & Zhang, 2008). Therefore, the prefix labelling scheme provides the structural details with regard to all relationships mentioned in the previous chapter (Tatarinov et al., 2002).

The ImprovedBinary scheme is a binary string. It has the feature that supports the updates by requesting no relabelling or recalculations. Regarding the ORDPATH scheme, this is a strong scheme in terms of handling updates and insertions as this scheme allocates odd numbers to parent nodes whereas even numbers are allocated to the inserted nodes (O'Neil et al., 2004a).

- **Disadvantages of Prefix Labelling Scheme**

There are a number of limitations to prefix labelling schemes that cause problems when using these schemes: first, the non-tree edges, which are nodes or edges that do not appear in the normal tree relationship. This problem with the building of non-tree edge relationships was discovered by Yun (de l'Adour; Yun & Chung, 2008). This issue causes a weakness in labelling non-tree edge relationships. In order to solve this problem, deterministic tree labels should be used with prefix labelling schemes. However, using non-tree labels means extra time needs to be spent in executing additional tasks such as providing extra storage (Fennell, 2013). Some researchers have already claimed that prefix labelling schemes need more improvement in terms of query processing (Hou et al., 2001).

The prefix labelling scheme also has the disadvantage of not handling complex XML files properly as these schemes can't allocate extensive labels. This is due to the fact that XML files have deep and long paths (Bosak & Bray, 1999; Murata et al., 2000; Tatarinov et al., 2002). In addition, the prefix labelling scheme can't handle the dynamic updates properly so as a result many researchers perceive this issue to be a challenge (Brenes, Wu, Van Gucht, & Santa Cruz, 2008).

3.3.2.1.2 Interval Labelling Scheme:

A number of researchers have proposed different interval labelling schemes with different quality and performance. This type of scheme is also known as Region-based Encoding. The idea of this scheme is to attach two values to each node which are the startID and the endID. The startID is used to save the node ID for first element or attribute. The endID is used to store the end of the attribute (X. Wu, M.-L. Lee, & W. Hsu, 2004a). There are common examples of this kind of labelling scheme which uses the same technique for labelling nodes such as the (Beg, End) which is also called the containment labelling scheme, introduced by Zhang et al. (Zhang et al., 2001). Another example of interval labelling schemes is the (Pre, Post) labelling scheme proposed by Dietz (P. F. Dietz, 1982). The (Order, Size) scheme was developed by Li and Moon (Q. Li & Moon, 2001) as an interval scheme. More details about some of these types of labelling schemes are as follows:

I The (Beg, End) labelling scheme allocates two numbers to each node based on its sequential traversal order. The mechanism of this scheme is to assign the "Beg" number according to the sequential location in the XML document to all elements including the root, and every element, attribute of an element, value of an attribute, and value of an element. The process of assigning values of the "End" number starts when the process of "Beg" reaches the end of an attribute or an attribute value. This start value must be the same as the next sequential value (P.

Dietz; Duong & Zhang, 2005; Kha, Yoshikawa, & Uemura; Silberstein, He, Yi, & Yang, 2005). Figure 3-2 illustrates an example of this scheme.

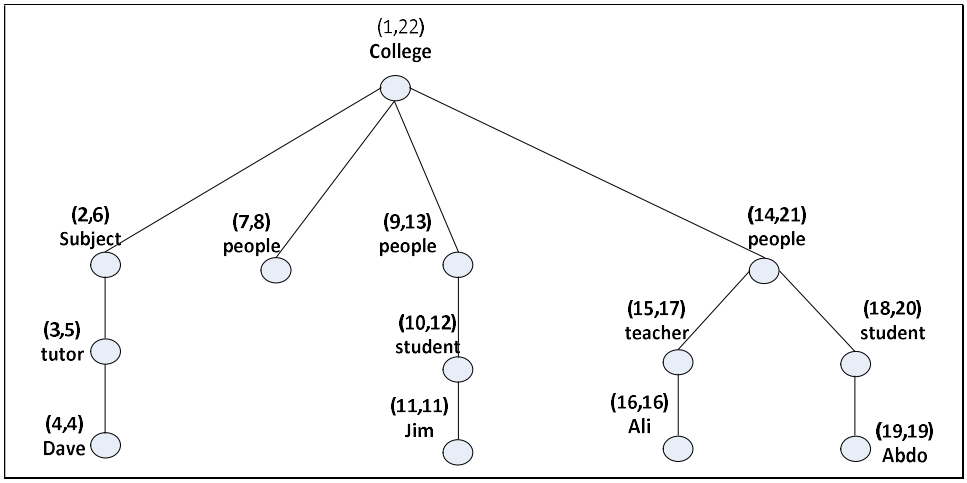


Figure 3-2: Containment (Beg, End) labelling Scheme

The scheme (Beg, End) can be used to answer both twig query and path query by making use of the relational database management system. This can be achieved by using "structural joins". To answer a query, the relationships between any couple of nodes in a path in this query are investigated individually as the granularity of this indexing scheme is determined at the level of each node. Therefore, this provides a precise and complete answer for the query. XML queries shred XML documents into tables of relational databases with fixed schema (Label, Beg, End, Level, Flag, and Value) (Gang et al., 2007). Table 3-1 represents the relational table of shredded XML documents of the above node tree.

TABLE 3-1: A NODE TABLE OF THE XML DATA IN FIGURE 3-2

Label	Beg	End	Level	Flag (Type)	Value
Subject	2	6	2	Element	Null
Tutor	3	5	3	Value	Dave
People	7	8	2	Element	Null
.....	...	...	....	.....	.....
People	14	21	2	Element	Null
Student	18	20	3	Attribute	Abdo

Silberstein et al. (Silberstein et al., 2005) developed dynamic labelling schemes for interval indexes. The dynamic labelling schemes allow relabelling of the schemes. The interval labelling scheme is the most used scheme for fixed encoding. Such examples propose leaving spaces between the values in order to

add new nodes. In case of adding new nodes, there is a need for re-numbering or other solution.

Cohen et al. (Edith et al., 2010) argued that persistent labelling needs 0(n) bits per label where n is the size of the tree. The interval labels size is used to measure the complexity as this size determines the total size of the index. It is preferable to keep the used number of bits small as this can allow the index to reside in the main memory.

II Li and Moon (Q. Li & Moon, 2001) developed the (Order, Size) labelling scheme. Each *Order* and *Size* has a certain job. The *Order* one is based on a traversal of pre-order, whereas the *Size* part is an estimation of the number of child or descendent nodes for a given node. The advantage of this mechanism is that this labelling scheme leaves space for any case of adding or inserting nodes in order to avoid relabelling of the data-tree as relabelling can cause delay.

- **Advantages of Interval Labelling Schemes**

The space between the start and end in an interval labelling scheme helps very much in creating the relationships between ancestor and descendant, or between parent and child (Cunningham, 2006). Moreover, one of the main advantages is that interval labelling schemes support the XML tree since the tree labels are efficient and unlike the non-tree labels. This is due to that the interval labelling scheme being able to gain the description of the relationships between the child and parent (Fallside & Walmsley, 2004; O'Neil et al., 2004a; Wu et al., 2004b).

- **Disadvantages of Interval Labelling Schemes**

Interval labelling schemes cannot cope with some cases such as dynamic updates which are not always supported (Tatarinov et al., 2002), particularly when the space between any two nodes is not enough for the inserted nodes (Cooper, Sample, Franklin, Hjaltason, & Shadmon, 2001). This problem is due to this type of labelling scheme being based on the technique of a limited space (interval) assigned to the nodes (Duong & Zhang, 2005). Consequently, the interval labelling scheme is efficient whenever the inserted nodes are within the assigned space between two nodes (Amato et al., 2003). Due to the frequent updates on XML documents, updates on dynamic XML documents is a significant issue (Cooper et al., 2001). Furthermore, the worst limitation is that the relabelling process occurs only under restricted circumstances (Harold et al., 2004).

3.3.2.1.3 Conclusion:

Both prefix labelling scheme and interval labelling scheme perform well in certain cases and have advantages and disadvantages. With regard to the relabelling process, the prefix labelling scheme performs well. Therefore, many users prefer to use this

type of labelling scheme whenever they want to reduce relabelling to the minimum. However, the interval labelling scheme is better than the prefix scheme in terms of the storage space cost. In addition, this scheme is costly in terms of updates (Wu et al., 2004b; Zhang et al., 2001) (Y. Chen, Mihaila, Bordawekar, & Padmanabhan, 2005; Wu et al., 2004b; Yang, Fontoura, Shekita, Rajagopalan, & Beyer; Zhang et al., 2001). Based on the criteria for evaluating the existing labelling scheme, and in terms of the retrieval power, both the prefix and interval labelling schemes deliver precise results as they return no false answer. Concerning the processing complexity, two sub criteria were used in this regard, namely, the relationship computation and the data kind. The relationship computation of the interval is fixed which means that the relationship between a couple of nodes can be calculated in fixed time. The relationship computation of the prefix labelling schemes is directly proportional to the depth increase. Regarding the data kind, the interval schemes support numerical data kind, whereas the prefix schemes support string ones. With respect to the scalability, the interval schemes provide a linear increase and the prefix schemes provide an exponential increase. Finally, the prefix schemes are better for the update than the interval schemes.

3.3.2.2 Graph Indexing Scheme (Path scheme):

This type of labelling scheme (also known as a summary index) is a structural path summary. It is used to enhance query performance, particularly for single path queries, by producing a path summary for XML data in order to accelerate the process of query evaluation. However, it also has the ability to solve twig queries with extra effort of multiple joins processes. There are a number of existing graph schemes such as DataGuide (Goldman & Widom, 1997, 1999); Index Fabric (Cooper et al., 2001); APEX (Chung et al., 2002); D(K)-index (Q. Chen, Lim, & Ong, 2003); (F+B)K-index (Kaushik, Bohannon, Naughton, & Korth); and F&B-index (Abiteboul et al., 2000; Gang et al., 2007). Graph schemes are classified into different categories and according to different criteria (Chung et al., 2002; Cooper et al., 2001; Yoshikawa, Amagasa, Shimura, & Uemura, 2001). Examples of these classifications are the following: Polyzotis and Garofalakis (Polyzotis & Garofalakis, 2002) classified the graph schemes according to exactness. This classification divided the schemes into exact schemes and approximate schemes. Examples of exact schemes are: strong Data Guide, 1-index, disk-based F&B-index, Index Fabric, and F&B-index. Examples of approximate schemes are A(K)-index, approximate Data Guide, D(K)-index, and (F+B)K-index (Polyzotis & Garofalakis, 2002).

There is another category that classifies graph schemes into two classes, namely, path schemes (also known as P-index), and twig schemes (also known as T-index). Path schemes are suitable for simple path queries; examples for this scheme are DataGuide

and 1-Index. Twig schemes are suitable for twig queries such as F&B-index (Gang et al., 2007).

Samir and Martin (Samir Mohammad & Martin, 2009) classified the graph schemes into determinism and bisimilarity. This classification is based on the consideration of properties such as path determinism and bisimilarity. The paths of the tree are considered to be deterministic paths in the determinism schemes. The bisimilarity has two sub classes: forward and backward. Thus, deterministic schemes provide uniqueness of paths, which is appropriate for single path queries. However, non-deterministic schemes provide uniqueness of elements and are suitable for twig queries as both forward and backward they provide precise results (S. Mohammad, 2011). As the term 'path indexes' is used to refer to different kinds of schemes, this classification uses determinism to classify schemes. More details are the following:

3.3.2.2.1 Deterministic Graph Scheme:

The Strong DataGuide was proposed by Goldman and Widon (Goldman & Widom, 1997). Strong DataGuides have the ability to give complete and precise results for both simple parent/child path queries and ancestor/descent path queries. Regarding twig queries, Strong DataGuides are complete but not precise (Kaushik et al.) (Q. Chen et al., 2003; Goldman & Widom, 1999).

The Approximate DataGuide (ADG) has solved the problem of the large size of the Strong Data Guide since this scheme shows large size in some cases (Goldman & Widom, 1999).

Cooper et al. proposed the Index Fabric in order to solve the problem of scalability (Cooper et al., 2001). The Index Fabric is theoretically like the Strong Data Guide as the size may enlarge dramatically. Moreover, the Index Fabric is complete for both path and twig queries. However, it is precise for the path but not for the twig (Samir Mohammad & Martin, 2009).

3.3.2.2.2 Non-deterministic Graph Schemes with Backward Bisimilarity:

There are a number of these indexing schemes such as: 1-index, A(K) index, and D(K) index. These indexes are based on backward bisimilarity. The 1-index was proposed by Milo and Suciu (Milo & Suciu, 1999) in order to decrease the size of the structural summary. The 1-index divides the data nodes of a document into similar classes based on their backward bisimilarity.

3.3.2.2.3 Non-deterministic Graph Schemes with Forward and Backward Bisimilarity:

This is the only kind of graph index that has the ability to support twig queries. Examples of this type of scheme are the following: the F&B-index, $(F+B)^K$-index, and the disk

based F&B-index (Su-Cheng & Chien-Sing); (Kaushik et al.). The F&B-index was proposed by Abiteboul et al. (Abiteboul et al., 2000). It is different from the A(K)-index and D(K)-index as this scheme is based on the incoming and outgoing paths' bisimilarity for all nodes. Thus, it is a twig structural index scheme. (Kaushik, Shenoy, Bohannon, & Gudes, 2002) developed the (F+B) k-index. This scheme is an improved release of the F&B-index. The (F+B)k-index scheme controls the size of the F&B-index by identifying the value of the "K" (Gang et al., 2007).
The Disk-based F&B-index was proposed by (W. Wang et al., 2005). This index scheme has provided additional properties and criteria. The Disk-based F&B-index is an integration of 1-index and F&B-index in a new clustered Disk-based F&B-index which is then saved on the disk (Samir Mohammad & Martin, 2009).

3.3.2.2.4 Summary

Some graph labelling schemes are appropriate for small XML documents such as 1-index and strong DataGuide; whereas, some others are appropriate for large XML data such as disk-based F&B-index. Some of these labelling schemes support single path queries such as 1-index, A(K)-index, and D(K)-index. However, the F&B-index and disk-based F&B-index support twig queries. Based on the criteria for this research for evaluating the labelling schemes, the following is a summary of a comparison between the three classes of graph schemes in the previous classification:
Regarding the retrieval power, the three classes return precise and complete answers for the path query. The results for twig query are not precise except the non-deterministic forward & backward bisimilar which is precise and all these classes return complete results.
As for the processing complexity, all three classes are not complex for path queries as they require no joins; whereas they are complex for twig queries except for non-deterministic forward and backward bisimilar. With respect to the scalability, both non-deterministic schemes grow linearly. The deterministic scheme grows linearly for the tree data only.

3.3.2.3 Sequence Indexing Scheme:

This kind of index converts both XML documents and queries into structure sequences. Sequence indexes put the values and the structures of XML data together into an integrated index structure. This structure is used to evaluate both path and twig queries efficiently, answering a query, making a string sequence that matches the sequence of the data with the query. This technique reduces the need for joins to evaluate twig queries (Wei et al.; Wei, Haifeng, Hongjun, & Jeffrey Xu, 2003). Regarding the classification of sequence schemes, they are classified into two types, according to the importance of the tree mapping direction, which are the following: top-down sequence indexing schemes, and bottom-up sequence indexing schemes.

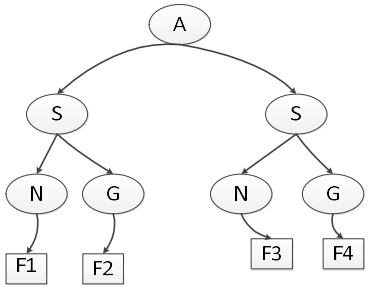
3.3.2.3.1 Top-down Sequence Indexing Schemes

Wang, Park, Fan and Yu (2003) proposed the ViST (Virtual Suffix Tree). This scheme is based on the B+ tree (H. Wang, Park, Fan, & Yu, 2003). In addition, the ViST has the disadvantage of weakening the query operations due to the large number of nodes being checked. This is due to the ViST being used as a top-down sequence. Thus, the size of the index becomes very large when dealing with large XML documents since the top elements are added into the sequence. This is the main disadvantage of the ViST scheme.

3.3.2.3.2 Bottom-up Sequence Indexing Schemes

The PRIX (Prufer sequence for indexing XML) is an example of a bottom-up sequence index. This indexing scheme does a good job in decreasing the query processing time. Since the ViST has a problem of scalability as mentioned above, Rao and Moon propose the PRIX as another method that uses a bottom-up sequence to solve the scalability problem with the ViST (Rao & Moon, 2004).

Some studies show that these two indexing schemes have a weakness in terms of precision, retrieving data, and processing complexity (H. Wang & Meng, 2005; H. Wang, Park, Fan, & Yu). However, the sequence indexing schemes have some advantages, such as 1) the ability to expect the results of the query; 2) using the complete query tree as one component in order to avoid the cost of joint operations. Figure 3-3 illustrates an example of this indexing scheme.



Label	Map
Students	A
PGR	S
Name	N
Group	G
Omar	F1
XDIR	F2
James	F3
SE	F4

Figure 3-3: Sequence-based indexing examples

From Figure 3-3, the XML tree is changed into a sequence as follows: T= (A), (S,A), (N,AS), (F1,ASN), (G,AS), (F2,ASG), (S,A), (N,AS), (F3,ASN), (G,AS), (F4,ASG).

Despite the sequence indexing technique having the advantage of speeding up the query pre-evaluation and getting rid of the increase in structural joins, this technique also has two disadvantages, namely, 1) the sequence of XML produced by one of this technique's algorithms has to be reconstructed quite often; 2) this technique uses a hashing algorithm to encode the textual values, which makes the hash list increase very fast, therefore the indexing becomes very slow (Bremer & Gertz, 2006; Meng, Jiang, Chen, & Wang, 2004; S. Mohammad, 2011; Prasad & Kumar, 2005; Rao & Moon, 2004, 2006; Stein, 2007).
To summarize, generally, most top-down and bottom-up sequence schemes return non-precise and non-complete results. As for scalability, the top-down grows exponentially whereas the bottom-up grows linearly. Regarding the process complexity, the top-down requires too many expensive joins and the bottom-up has very complicated processes.

3.3.3 Level-Based Labelling Scheme (LLS):

The LLS is based on the levels of the nodes in XML trees and the summary of an XML tree. This labelling scheme was developed to combine the advantages of both the interval and prefix labelling schemes, and to avoid their disadvantages. Therefore, this labelling scheme supports the processing of simple queries and twig queries. The LLS is based on the levels of the elements in XML trees. The element labels and values are firmly fixed with a structural summary so the method provides efficient query processing. This labelling scheme shows high performance in comparison to existing labelling schemes. The LLS has shown some advantages such as: the LLS maintains the best features and advantages of interval labelling and prefix labelling schemes. Second, the LLS provides a constant size of the label s regardless of the data-tree depth. Path information is easily available since the root path of an element can be produced from the labels. Third, LLS is based on the levels of the tree. Knowing the level at which a node is located can speed up query processing by improving the search space at an early evaluation stage (Samir Mohammad & Martin, 2009, 2010; S. A. Mohammad, 2011).

3.3.3.1 XML Data Model:

The XML document is modelled as trees. An *XML tree* is a directed ordered graph G. G=(*R,VR,VL,E,tagg,labelg,T*).

R is the root node. *VR* is the set of internal nodes which include the elements and attributes excluding the root. *VL* is the set of leaf nodes. *VL* = (VE∪VT), that is *VL* consists of the empty leaf node *VE* (for empty element), and the set of value (text) leaf nodes *VT*. Nodes in *VR* and *VL* are tagged through the *tagg* function (the extra *g* stands for the graph G). *VR* and *VE* nodes are tagged according to the tag of the elements or attributes they represent. Nodes in *VT* have the same tag as their *VR* parent nodes. Internal nodes *VR* have to have one or more child nodes, which could be *VR* and/or *VL* node(s). *E* is a set of child–parent edges, *E* = {e1, e2,..., ei} that connects all nodes of *VR* and *VL* to form a tree. (S. Mohammad, 2011; Samir Mohammad & Martin, 2010).

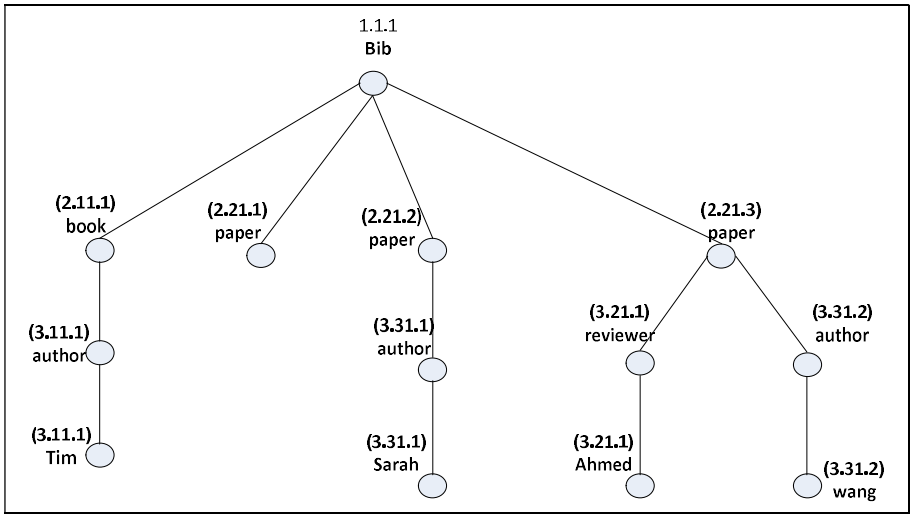


Figure 3-4: An LLS labelled tree representation of the XML document

3.3.3.2 Path Summary:

All nodes of an XML data-tree *G* can be summarized by a summary *S*, in a way that all node paths of *G* that have the same tag path *t* are represented by exactly one tag path *t* in *S*. Each tag path *t* of *S* is a tag path of at least one node path *n* of *G* (Samir Mohammad & Martin, 2010). Figure 3-4 shows an example of the LLS Labelling Scheme. And figure 3-5 shows an example of the summary of the LLS Labelling Scheme.

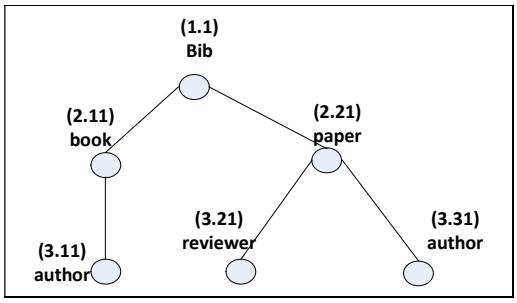


Figure 3-5: The summary S of the XML data-tree G

3.3.4 Dewey Labelling Scheme:

The Dewey labelling scheme is also called Dewey code labelling. It is one of the prefix labelling schemes. This labelling scheme was introduced for general knowledge classification (Scott & SCOTT, 1998). Tatarinov and colleagues (Tatarinov et al., 2002) first used this labelling scheme for XML tree-shaped data. Each node is related with a vector of numbers that reflect the node-ID path from the root to the designated node. Moreover, this labelling scheme is classified as a node index and a path index since each

node is represented as a complete path from the root to the indexed node (Samir Mohammad & Martin, 2009).

A prefix matching operation on the index string is carried out in order to determine whether a relationship of a parent–child or an ancestor–descendent exists. In a certain data-tree, node *x* is an ancestor of node *y* if the label of node *x* is a substring of the label of node *y*. Figure 3-6 shows an example of a Dewey Labelling Scheme. In this figure, node (0.3) is an ancestor of node (0.3.1.0). It is obvious that the Dewey labelling scheme does not require any additional information in order to evaluate the parent–child relationship. Thus, the Dewey labelling scheme is different from the (*Beg, End*) labelling scheme. For example, it is easy to see that node (0.3) is the parent of node (0.3.1) (Samir Mohammad & Martin, 2009, 2010).

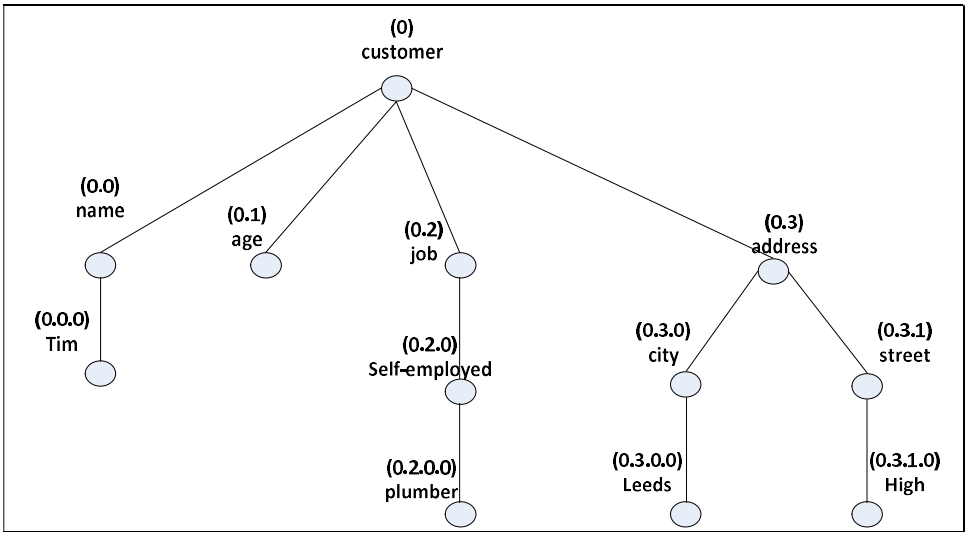


Figure 3-6: Dewey labelling scheme

Furthermore, also there is no need for any further information, for example level number or parent ID, to work out the sibling relationship, and it can be done in the same way as the parent–child relationship. The Dewey labels make direct support for the sibling relationship. In a specific tree, node *x* and node *y* are siblings if nodes *x* and *y* have the same number of components in their labels (call it *n*) and *x*.prefix = *y*.prefix, where the prefix length is equal to *n* minus one. For example, node (0.3.0) and node (0.3.1) are siblings. One of the Dewey label's advantages is that it is a good labelling scheme in terms of updating and it is better than the (*Beg, End*) labels as, when a new node is inserted, only the nodes in the sub-tree rooted at the following sibling need to be updated (Tatarinov et al., 2002). Nevertheless, one of the disadvantages of the Dewey labelling scheme is that its storage size enlarges with the depth of the tree. Additionally, as the depth increases, it becomes more costly to find the parent–child or the ancestor–descendent relationship between any two arbitrary nodes because the string prefix matching becomes longer (Samir Mohammad & Martin, 2009).

3.3.5 Importance of Using the Dewey and LLS Schemes in the Proposed Scheme

Having investigated the existing labelling schemes and defined their shortcomings, the LLS labelling scheme was chosen as it is a state-of-the-art labelling scheme. It has the advantages of both the interval and prefix labelling schemes (S. A. Mohammad, 2011). Briefly, the LLS scheme is based on the levels of the nodes in XML trees and the summary of an XML tree. In addition, it has some other advantages such as the data type is numerical, the size of the label is constant, and the relational computation is also constant (Samir Mohammad & Martin, 2010).

With respect to the Dewey labelling scheme, it was selected because it is simple and straightforward to implement. In addition, this scheme facilitates the sibling relationships. It is also a simple one for updates (S. A. Mohammad, 2011).

3.3.6 Significance of XML Labelling Schemes

The XML databases and XML query are not as mature as the relational databases that have been in existence for decades. However, the research areas of indexing and querying XML data have recently become very active (Al-Khalifa et al., 2002; Gang et al., 2007). The fast growth in XML data has led to this high demand for querying this data. Moreover, indexing is one of the techniques used to achieve high query performance and to retrieve data very fast (Vakali et al., 2005). One of the main shortcomings of indexing XML data is that there is always a trade-off between the size and efficiency of the index. In other words, indexes can be large in order to provide high performance; or small size with weak performance (Guoren et al., 2003). Another common problem with indexing XML data is that the update operations are usually expensive (Samir Mohammad & Martin, 2010). Therefore, this research concentrates on how we can improve the performance of query processing by enhancing the updates operations.

3.3.7 Critique of XML Labelling Schemes

In terms of precision, some labelling schemes may return non-precise results such as the sequence schemes and non-deterministic with backward bisimilarity. However, non-deterministic forward and backward bisimilarity mostly returns precise results. Moreover, node schemes always return precise results. Regarding completion, the node and graph schemes mostly return complete results, whereas, the sequence schemes return incomplete results. As for process complexity, the node schemes require structural joins and thus are complex. The sequence schemes don't require any structural joins. Finally, the graph scheme may require structural joins for twig queries but not for path queries. Therefore, the node schemes are the weakest schemes in this regard. Concerning growth of the size, all three kinds of labelling schemes grow both linearly and exponentially. Table 3-2 shows a summary of the main finding of the literature review.

Nowadays XML applications need to support dynamic XML documents and query them as high demand on the update of XML data. Thus, labelling schemes should consider this feature. Many examples have been proposed and discussed above. (Lindner, Mesiti, Türker, Tzitzikas, & Vakali, 2004; O'Neil et al., 2004a). The key focus in all these schemes was on avoiding or reducing the number of required relabelling nodes. These schemes came up with different mechanisms to achieve this goal. Another technique used to improve the query performance is compacting the labelling schemes.

TABLE 3-2:SUMMARY OF THE RESULTS FOR THE EVALUATION OF THE LABELLING SCHEMES

No.	Criteria	Node index technique	Graph index technique	Sequence index technique
1	Retrieval power (precision)	Yes	Yes/no (yes for path & no for twig)	No
2	Processing complexity	No	Yes/no (join required)	Yes
3	Scalability	Yes	Yes	Yes
4	Update cost	Yes	Yes	Yes

Based on the above review, the main features and challenges that the labelling scheme face are as follows: supporting the updates and relabelling are significant features of state-of-the-art labelling schemes (D. K. Fisher et al., 2006). Updates are frequently needed over the internet processes in the present era, and therefore, this feature needs to be considered in a perfect labelling scheme. This issue is one of the main challenges that the existing labelling schemes face as many of them may not support this matter. It is considered as a limitation if a labelling scheme does not support dynamic querying (Fennell, 2013).

The time needed to determine the relationships between the nodes is a crucial issue as sometimes it can be too long and as a consequence the querying process will be inefficient (Cunningham, 2006). The first step in query processes is the determination of the relationships required by many of the labelling schemes. In fact, labels are used to create the relationships among nodes which are needed for the query processes (Harold & Means, 2004). The most common relationships are the node level, ancestor/descendent, and parent/child. Harold (Harold & Means, 2004) noted that determining these last three relationships can take a long time.

The ideal scheme should support various kinds of insertions of nodes and this is an important advantage (D. K. Fisher et al., 2006; Gou & Chirkova, 2007). The uniform insertion has been considered as an important kind of insertion for any state-of-the-art labelling scheme (Alstrup & Rauhe, 2002). The measurement of spent time of the insertion performance and the size of the labels after insertion might be used to determine

whether the labelling scheme is ideal for uniform insertions (Cooper et al., 2001). Ordered skewed insertions should also be considered in an ideal labelling scheme (Cohen et al., 2010). Finally, the random skewed insertion is a crucial one for any model labelling scheme. This kind of insertion is used with some labelling schemes that can't cope with random insertions as they have fixed node structures (Cohen et al., 2010). More details are discussed in the methodology chapter.

Having considered the above review, this research intends to propose a state-of-the-art XML labelling scheme. The goal of this proposed scheme is to address the four issues that were just discussed above, namely, supporting update processes, supporting labelling XML data, supporting various insertions, and finally supporting the spent time for determination of the relationships.

3.4 Review on the Testing of Existing labelling Schemes:

Existing labelling schemes have been tested in different techniques based on the aspects that will be assessed such as performance, scalability, and efficiency. Most of the labelling schemes are compared using different measures such as initial labelling, label size, creating labelling time, and the cost of updating. Usually the proposed scheme is compared with one or more existing labelling schemes to show the improvements that this proposed scheme can deliver. Different experiments are designed according to the testing aims for testing. In order to compare a labelling scheme with others, the queries that these schemes support should be considered. A review on the testing of most relevant and common labelling schemes will be discussed below.

3.4.1 Testing the LLS scheme: the (*Beg, End*) was used in the experiments to show the improvement in the LLS. Two datasets were used to test the LLS which are as follows: the DBLP computer science Bibliography dataset and the XMark dataset. Furthermore, four kinds of XML queries were used to evaluate these schemes. These kinds of queries are the most common ones used. The technique of mapping to relational database tables was used to evaluate XML queries. Each query was run twenty times and then the average was taken.

3.4.2 Testing the DDE scheme: three datasets, namely, XMark, Treebank and Nasa were used to test this scheme against two other labelling schemes. The two labelling schemes tested against the DDE were ORDPATH and Compact DDE (CDDE). The executed experiments are as follows: initial labelling, querying static document, querying dynamic document, and processing updates. The processing updates included uniform insertions and skewed insertions which were classified into order skewed insertions and random skewed insertions.

3.4.3 Testing the LSDX scheme: the XMark was used as a dataset for generating XML data. This scheme was tested against the GRP scheme (by Lu and Ling, 2004) and the SP scheme (by Cohen, Kaplan and Milo, 2002). The experiments that were

performed are the following: length of labels, time used to generate labels, insertion and deletion time.

3.4.4 Testing the ORDPATH scheme: as for the dataset, the XMark and XMach-1 were used to test this scheme. Different measures were used such as arbitrary insertions, insert-friendly IDs, ORDPATH length. This scheme was compared with the Dewey order and others for evaluation purposes.

The most common technique used for testing XML labelling schemes is to compare the proposed scheme against other relevant schemes. Therefore, as the proposed scheme is based on the LLS and Dewey schemes, the assessment of the proposed scheme is performed by comparing this scheme against these two schemes. So, the proposed scheme and the others were implemented. This implementation was necessary as the LLS and Dewey labelling schemes are not available as source code. More details are provided in the Methodology chapter.

3.5 Summary

The labelling schemes represent the indexes in XML data. Labelling XML data plays a main role in enhancing the efficiency of querying XML data. In fact, indexing XML data must represent the structure of the XML data tree, so it can support XML queries. Moreover, the existing labelling schemes have different limitations.

This chapter reviewed the existing XML labelling schemes and considered their structures and techniques for supporting query processes based on identified criteria. These criteria were used to evaluate the targeted labelling schemes.

Consequently, the main findings are as follows: all labelling schemes have advantages that make each one of them eligible for certain requirements of users. Thus, no one of these labelling schemes is capable of meeting all user's requirements. However, one of the main problems that these labelling schemes face is the trade-off between the size of the index and the performance efficiency. So, whenever the size of the scheme increases, then the performance of the query decreases. Moreover, the review also found that inserting and deleting nodes requires relabelling some nodes accordingly. Li and Moon stated that even though the early labelling schemes have been amended, they still suffer from the same limitations (Q. Li & Moon, 2001). Another issue that the existing labelling schemes have is scalability. The update cost is also a common problem that labelling schemes encounter. The cost here is measured by the occupied storage space. In other words, the increase of the required number of relabelling nodes causes deficiency.

The LLS and Dewey labelling schemes were considered for use in the new state-of-the-art scheme that this research will propose. The LLS scheme was chosen as it is state-of-the-art and has a number of advantages. For instance, the used data type is numerical, and the size of the labels is steady. As for the Dewey scheme, it is a pre-fix labelling scheme

which shows the ability to manage a cluster-based scheme. It is also a simple scheme so can easily be implemented.

Consequently, all researches try to achieve a labelling scheme that can be stored in the minimum possible space while maintaining the retrieval power at the same level. Similarly, and based on this literature review, this research considers coming up with a state-of-the-art labelling scheme that might achieve the aim of the research. As a result, this research proposes a hybrid scheme which is a clustering-based labelling scheme.

4. Methodology and Framework

4.1 Introduction

XML database management systems rely on labelling schemes as an essential factor for the XML query processing. By using a suitable labelling scheme, retrieving and indexing of XML data can be done efficiently. The scheme reflects the structure of the XML tree in order to carry out the process. Each node in this kind of processing of the XML tree is assigned a unique label. The structural relationships can be extracted by comparing these unique label nodes (Jayanthi & Tamilarasi, 2011).

This chapter starts by explaining the fundamentals of the proposed labelling scheme such as the idea of this scheme and the mechanism for implementing it, as well as the data model. It continues by describing how the update cost should be affected in this proposed scheme; subsequently, the design of the proposed scheme is illustrated including the implementation of the LLS and the Dewey labelling schemes for evaluation purposes. The experimental setup is an essential section in this chapter. This section discusses the objectives and types of the experiments. It also includes a review of the existing datasets and benchmarks in ord er to adopt an appropriate one for the testing. Section 3.4 concerns the testing, which discusses the testing platform and environment, and then explains the testing objectives. Subsequently, this section also demonstrates the measurement and metrics and testing plan. Finally, this chapter finishes by discussing the summary of this chapter and what has been achieved.

4.2 Existing Approaches and our Approach, a Comparison and Justification

Based on the literature review of XML labelling schemes, number of approaches have been proposed and developed as solutions for indexing XML data. As has been stated, each approach has advantages and limitations and at the same time has a specific technique. This section discusses and compares the most common techniques used in these labelling schemes to resolve the targeted issues, in order to ascertain the limitations of these techniques and justify the improvement that the proposed scheme might provide.

Based on the classifications of these labelling schemes in the literature review, first, let's start with the node schemes. The main idea of this technique is to save numbers that represent the positions of the nodes in the tree. Thus, accessing any node can be achieved by using these numbers. This technique is suitable for both single and twig queries.

The node schemes were classified into two groups, which are the prefix-based and interval labelling schemes. The following provides details for these groups and their own labelling scheme.

The main idea of a prefix-based scheme is labelling the father of a node as a prefix of the node itself (Runapongsa, Patel, Jagadish, Chen, & Al-Khalifa, 2006). Many prefix-based schemes have been proposed such as the Dewey scheme, LSDX scheme, GRP scheme, and ORDPATH scheme, each of which has it is own technique and targets certain issues. The Dewey is the most commonly used one. More details about the techniques of these schemes are as follows:

- Dewey labelling scheme: this scheme was classified as a node scheme and path index at the same time. It has advantages such as the ability to manage the clustering nodes into groups, the simplicity of implementation of the scheme, high performance and ease of updating. This scheme returns precise and complete results for queries. However, because the label is non-constant, so the size of the label enlarges as the depth of the data tree increases (Scott & SCOTT, 1998; Tatarinov et al., 2002).
- ORDPATH labelling scheme: this is the same technique as the Dewey scheme but the difference is that this scheme uses odd numbers to label child nodes, whereas even numbers are used for inserting new nodes (O'Neil et al., 2004a).
- GRoup base prefix (GRP): the technique of this scheme is to employ two divisions: group ID and group prefix.
- Dynamic XML data (LSDX): this technique is to combine numbers and letters in the label that support relationships such as siblings and ancestor–descendant. It is a good technique in terms of update and data retrieval.

There are a number of interval schemes such as (Beg, End) scheme and (Order, Post) scheme. The idea of this technique is to attach two values to each node which are the startID and the endID. The startID is used to save the node ID for first element or attribute. The endID is used to store the end of the attribute.

The technique of a Graph indexing scheme is based on producing a path summary for the XML data. The path summary accelerates the process of query evaluation in order to improve query performance. There are a number of Graph schemes that have been proposed. Different techniques were employed in these schemes. These techniques and their classifications were addressed in the literature review.

Regarding the techniques of Sequence indexing schemes, they convert both XML documents and queries into structure sequences. Sequence indexes put the values and the structures of XML data together into an integrated index structure. This structure is used to evaluate both path and twig queries efficiently, answering a query, making a string sequence that matches the sequence of the data with the query. The problem with most sequence techniques is that they return non-precise and non-complete answers.

Concerning the technique of dividing the nodes into groups and introducing sub-trees, this idea has many advantages which are discussed in section 3.1.

This research, as many others in this field, tries to introduce a new technique for labelling XML data that can occupy the minimum storage space and at the same time keep the query performance efficiency at the best level.

Therefore, this research proposes a labelling scheme that takes advantage of some of the existing schemes, using the clustering-based technique in order to introduce a technique that achieves a better solution than others. Consequently, this technique is based on the following main points:

- The Dewey labelling scheme was selected for labelling the clusters due to the advantages of this scheme such as the ability to manage the clusters, the ease of updating, and because it performs well.
- The LLS labelling scheme was selected for labelling the nodes of the clusters. The advantages of this scheme are that it uses numerical data and the size of the labels is steady.
- The clustering-based technique was used due to the great advantages of this technique. It reduces the required relabelling cases and as a consequence improves the inserting new nodes processes, with easy determination of the relationships.

All three of these ideas were intended to present the state of the art scheme that would achieve the aim of this research.

4.3 The Data Models of the Existing Labelling Schemes:

There are various kinds of techniques used for labelling XML documents as discussed in the Literature Review chapter. Some of them are not suitable for dynamic XML data. A review of these models is discussed in this section and an explanation of why this research selected a particular one and rejected others.
The model of the prefix technique does not handle complex XML files properly and handles the dynamic updates (D. Fisher et al.). As for the models of the interval schemes, due to the limited space between nodes, this model is not suitable for dynamic updates and is very costly (Fallside & Walmsley, 2004; O'Neil et al., 2004a; Wu et al., 2004b).
Regarding the data model of the sequence schemes, these schemes do not return precise results. They are also weak in terms of retrieving data and processing complexity (H. Wang & Meng, 2005; H. Wang et al.). This model must be reconstructed often and has become very slow due to using the hashing technique (Bremer & Gertz, 2006; Meng et al., 2004; S. Mohammad, 2011; Prasad & Kumar, 2005; Rao & Moon, 2004, 2006; Stein, 2007). These issues conflict with the objectives of the proposed scheme.

The model of the LLS scheme is based on the levels of the tree. This feature makes the query process of this scheme very fast. Additionally, the model of this scheme has a fixed size of labels regardless of the size of the tree. The Dewey model is a very common one and has been studied and adopted in many later schemes. It is a very robust scheme that supports the structure of the index. The decision to choose the LLS and Dewey schemes

was considered carefully based on these issues. Moreover, for the reasons mentioned above, this research also carefully rejected other data models.
Regarding the significance of this proposed scheme for the real world, the experiments of this research were executed to compare the performance of this scheme against the other two schemes. However, using the XMArk dataset implies that the data can represent the real world as it is a well-known dataset used to test XML schemes. Having said that, performing further experiments in order to test the proposed scheme against a dataset from the real world such as DBLP would be recommended to test this scheme for the real world. Furthermore, the tests for the experiments were repeated twenty times as this number has been used in testing a number of labelling schemes such as the Labelling Dynamic XML Document and the LLS scheme.

4.4 Research Approach

The objective of this research is to produce a framework that can be a state-of-the-art technology. Therefore, based on the in-depth review in the previous chapter, different techniques are used to overcome the limitations. These techniques were studied in the literature review chapter, each of which has some certain limitations, as well as advantages.

This research came up with the idea of developing a state-of-the-art approach. The idea of this approach is to propose a hybrid labelling scheme using two existing labelling schemes, namely, LLS and Dewey labelling schemes in labelling the targeted XML documents. Also, this scheme is based on using the grouping technique which has already been used before. More details in the following section, number 4.4.1.

The idea of the proposed scheme is based on clustering nodes in order to ease the determination of the child–parent and sibling relationships as the child–parent relationship is available in each XML document; furthermore, these relationships also ease the process of inserting new nodes. The parent–child clustering-based technique helps in dealing with a small tree rather than the entire XML tree (Kaplan, Milo, & Shabo, 2002). It was also found that the parent–child clustering technique supports the labelling process, and is more efficient than the simple tree in terms of the accuracy and spent time for query processing (Gusfield, 1997).
One of the features of a clustering-based technique is that it uses two labels for every node as this idea eases the procedure of labelling nodes that share the same cluster; whereas the label of the cluster is used to link that cluster with the entire tree. This feature assists in determining the relationships among nodes that form different clusters (Xu et al., 2009). It was stated that a scheme is developed to provide fast identification of relationships, as this feature helps in the optimising of query processing (Q. Li & Moon, 2001).

Therefore, these advantages support the proposed scheme, which consequently helps in enhancing the query processing and other targeted features of the proposed scheme such as improving the process of labelling XML documents.

4.4.1 The Mechanism and Data Model of the Proposed Scheme:

The idea of the proposed labelling scheme is to divide the whole data tree into small groups (clusters). This mechanism makes each cluster itself a sub-data tree. The advantage of this mechanism is to reduce the number of required relabelling cases to the lowest possible level, and as a result improve the efficiency of the query performance processing. Subsequently, two XML labelling schemes were used to label the nodes and their clusters of a data tree. These two schemes are the Dewey labelling scheme which was used to label the clusters; and the LLS labelling scheme which was used to label the nodes of the data tree.

4.4.1.1 Labelling XML Document:

The implementation of this task is divided into two stages as follows:

4.4.1.1.1 Creating and labelling clusters:

All nodes are grouped into clusters. The mechanism for achieving this task is as follows:

A- Each node and its child nodes are gathered to build a cluster.
B- Only the main root of the document is considered as a cluster itself and without child nodes. This node is labelled number (1).
C- Each cluster is considered a sub-tree.
D- A cluster can have only one or two levels of nodes.
E- Each cluster has at least one node.
F- Each cluster may have a root and child node(s) that is connected with this root. Thus, a cluster must contain at least one node.
G- All clusters, including the main root (not each node), are labelled by the Dewey labelling scheme.

4.4.1.1.2 Labelling Nodes:

All nodes are labelled according to the following mechanism:

a) Each cluster is treated as a sub-tree and its nodes are labelled separately from other clusters.
b) The LLS labelling scheme is used to label all nodes.
c) If the root of a cluster is a child node of another cluster, which means that this root has already been labelled, then the label of this root will be kept the same.

Figure 4-1 shows an example of an XML data file. Figure 4-2 illustrates the proposed scheme for the XML data document in Figure 4-1.

```
<College>
  <subject>
      <tutor>Tim</tutor >
  </ subject >
  <people> </people>
  <people>
      <student>Sarah</student >
  </people>
  <people>
  < people teacher = "Ahmed">
      <student > Wang </student>
  </people >
</College>
```

Figure 4-1: An example of XML data document

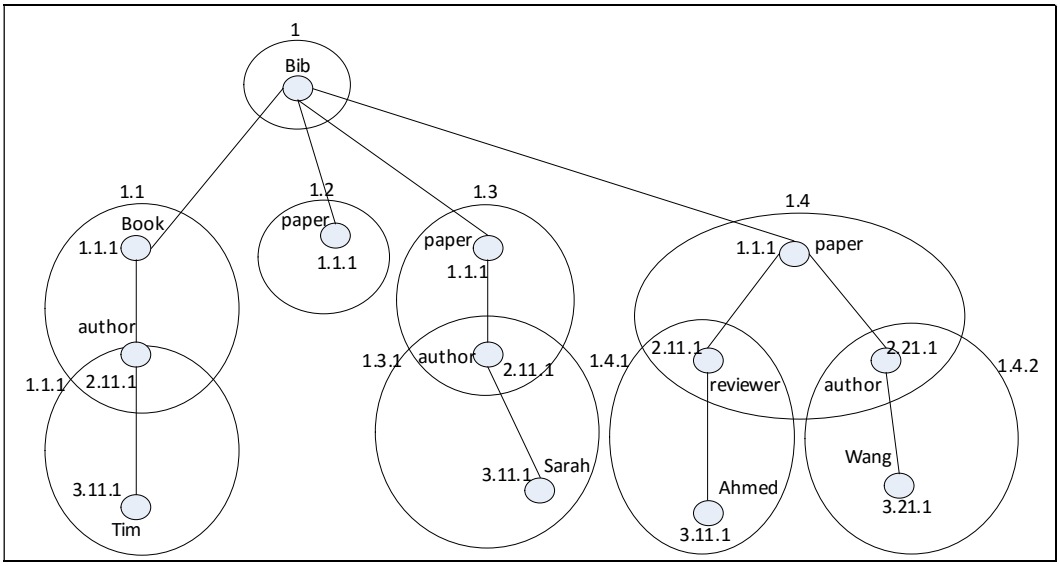


Figure 4-2: The proposed labelling scheme for the XML document in figure 4-1

The following definitions and explanations discuss the structures and relationships in the proposed scheme:

Definition 1: a *cluster* consists of one node or a set of nodes that includes a root and the rest are child nodes for this root. So, this sub-tree in a cluster has only one or two levels of nodes.

Definition 2: two nodes are from the same cluster if, and only if, their cluster's labels are the same.

$\therefore$ ***n1*** node and ***n2*** node $\in$ ***C1*** cluster $\Rightarrow$ their cluster's labels are the same.

4.4.1.1.3 Determining Node's Level:

Definition 3: the detail of the level of each node is extracted from the label of the node as follows:
The level of a node is the first component of the label of this node +1.

$$\therefore \text{Node's level} = 1^{st} \text{ component of the label} + 1$$

$$\therefore \text{Node's level} = (\text{lev.valueTable}+1)$$

Lev = a field (column) name in a table in the database.
valueTable = is the name of a table in the database.

For example, as shown in Figure 4-3, the level of the node '*Student'* (for which its label is: 1.1.1) is:

$$1+1=2.$$

4.4.1.1.4 Determining Label Order:

There are two cases in this matter which are as follows:

- **Label Order for Nodes from the same Cluster:**
  The order is determined by using the node labels only. The following definition shows how this order can be determined:

Definition 4:
Node 1 is before node 2 if, and only if, one of the two following conditions applies:
Condition 1: the level of the first node is before the level of node 2

$$\text{Level 1} < \text{Level 2}$$

$$\text{n1.Lev.summary} < \text{n2.Lev.summary}$$

n1 = node 1, n2 = node 2
Lev = a field (column) name in a table in the database.
Summary = is a table name in the database.
Condition 2: the second component of the node's label of the first node is smaller than the second component of the node's label of the second node. The following definition clarifies this explanation:

$$\text{n1.perLev.summary} < \text{n2.perLev.summary}$$

perLev = a field (column) name in a table in the database.
n1 = node 1, n2 = node 2.
Summary = is a table name in the database.

- **Label Order for Nodes from Different Clusters:**

Definition 5: a *cluster* consists of one node or a set of nodes that includes a root and the rest are child nodes for this root. So, the sub-tree in a cluster has only one or two levels of nodes.

4.4.1.2 Inserting new Nodes:

Inserting a new node will affect the corresponding index structure of the database which is the labels. The following examples show various scenarios for inserting new nodes and show how the proposed scheme copes with different scenarios:

4.4.1.2.1 Inserting a Node Between two Nodes:

There are two cases in this scenario:

a) Firstly, if the inserted node is a child of the main root, then this node is inserted within the second level of the data tree. In this case, a new cluster is created and then both the new cluster and the new node are labelled. Moreover, all the right clusters need to be relabelled. This is one of the worst scenarios as the number of required relabelling processes is higher than most of the other scenarios. Figure 4-3 illustrates an example of such insertion; when the inserted new node is within the second level of the data tree, a new cluster must always be created.

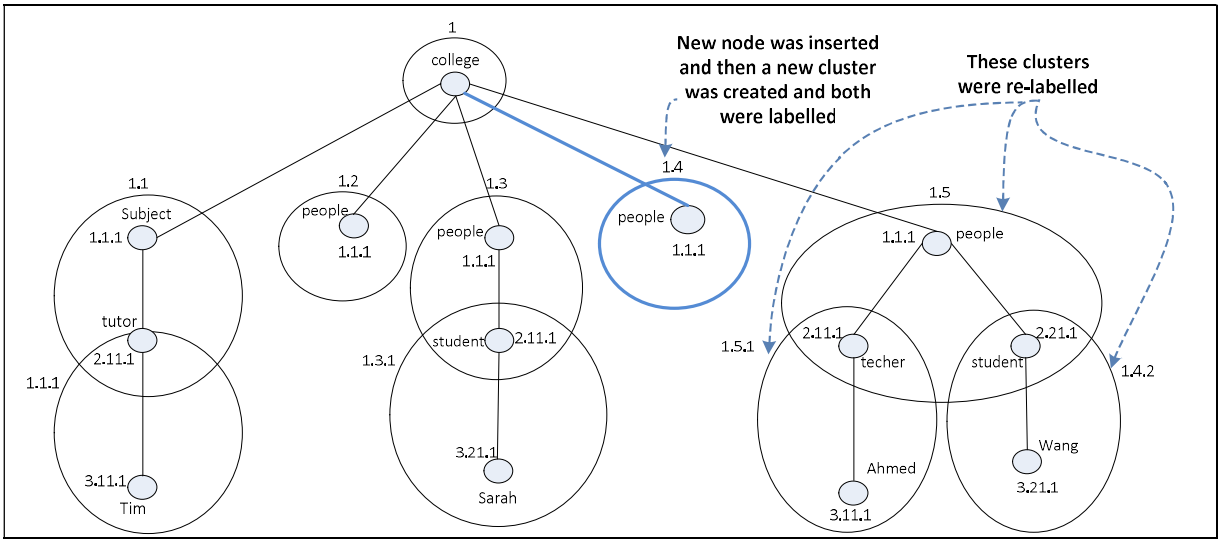


Figure 4-3: Inserting a new node in the second level

b) Secondly, if the inserted node is a child node within a cluster, then all the right siblings' nodes in that cluster, if there are any, need to be relabelled. Figure 4-4 shows an example.

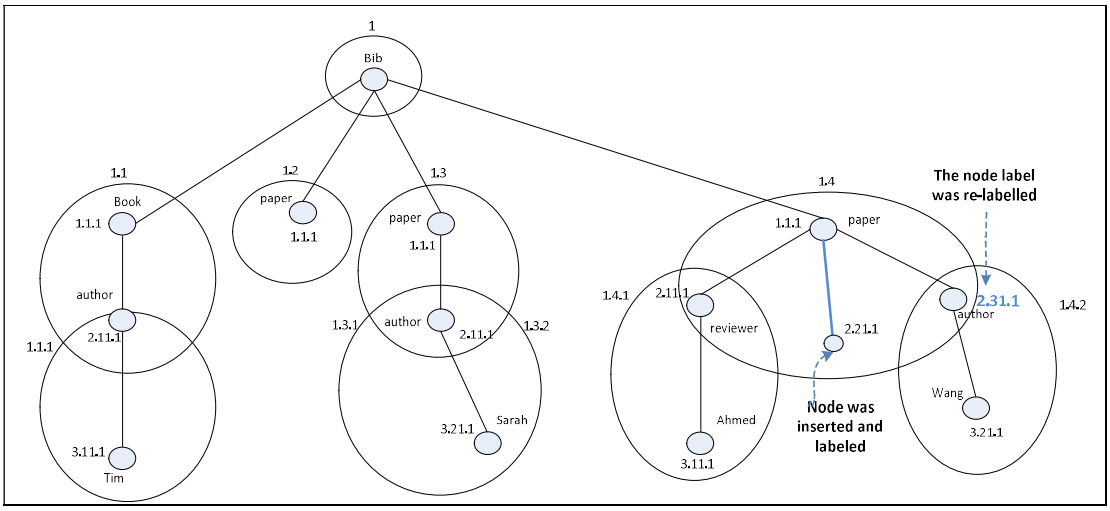


Figure 4-4: Inserting a new node inside a cluster

4.4.1.2.2 Inserting a Node in the Rightmost Side:

This scenario is one of the most efficient updates as there is no required relabelling. There are also two cases in this scenario as follows:

a) If the inserted node is a child of the main root and inserted to the rightmost, then a new cluster is created. Both the new node and new cluster are labelled. For instance, in Figure 4-5, the ***'lecture'*** node has been inserted and labelled (1.1.1), and then a new cluster has been created and labelled (1.6). There are no required relabelling processes in this case.

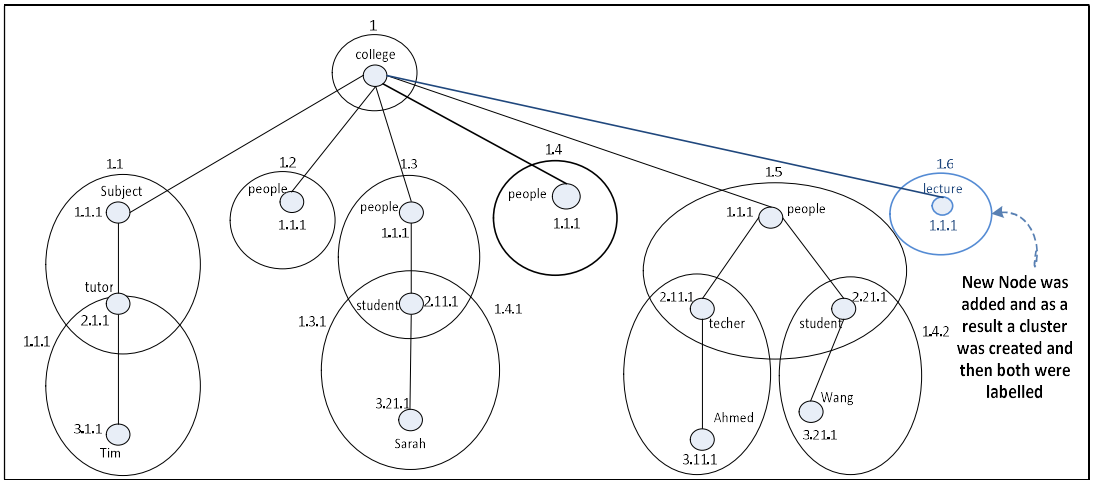


Figure 4-5: Inserting a new node in the second level and rightmost side

b) If the node is inserted within a cluster (not in the second level of the tree) and at the rightmost in this cluster, then there is no need for any relabelling operations. For example, the node *'James'* is inserted and labelled (3.21.2) as Figure 4-6 shows.

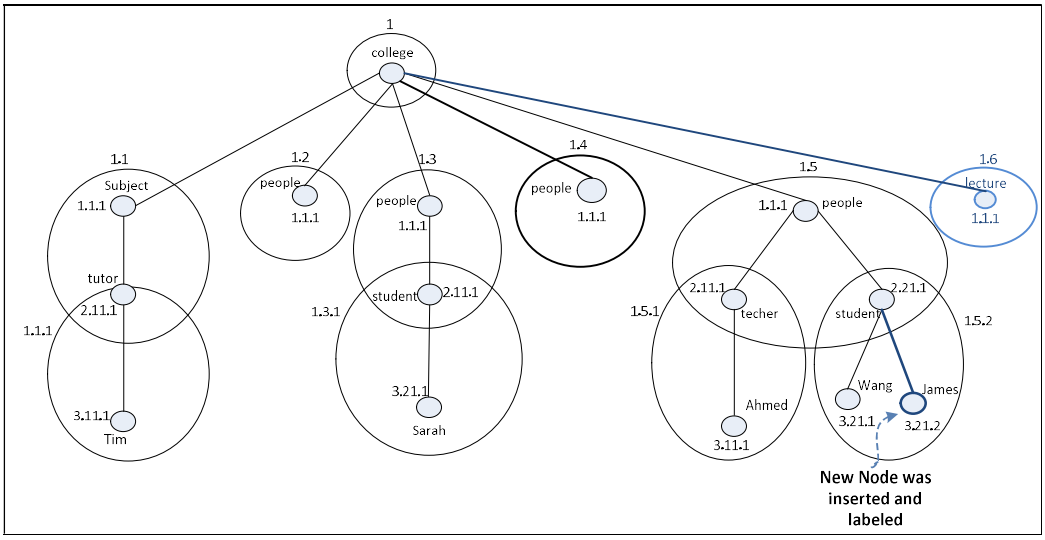


Figure 4-6: Inserting a new node in the rightmost side not in the level two

4.4.1.2.3 Inserting a Node in the Leftmost Side:

There are two cases in this scenario as follows:

a) If the new node is inserted in the leftmost side and this node is a child of the main root, then a new cluster is first created. Subsequently, this cluster is labelled according to the Dewey scheme by reducing the second component of the first cluster's label by one. Consequently, the node is inserted in this cluster and labelled by the LLS scheme. In Figure 4-7, the node '*syllabus'* has been inserted in the leftmost side in the second level and labelled (1.1.1) and a cluster is created and labelled (1.0). There is no further required relabelling process. In case more nodes are inserted in the leftmost sides then the second component is reduced by one for every inserted node.

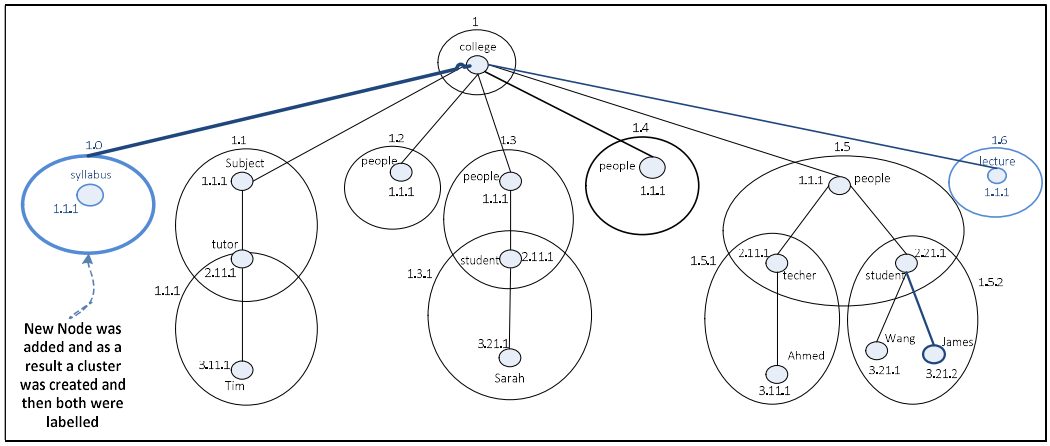


Figure 4-7: Inserting a new node in the second level and leftmost side

b) The second case is to insert a new node in the leftmost side and at any level except the second one. Thus, this node will be within a cluster. All the right-

side nodes of the inserted node in this cluster need to be relabelled. Figure 4-8 shows an example where the node *'David'* has been inserted and labelled and the right-side nodes have been relabelled.

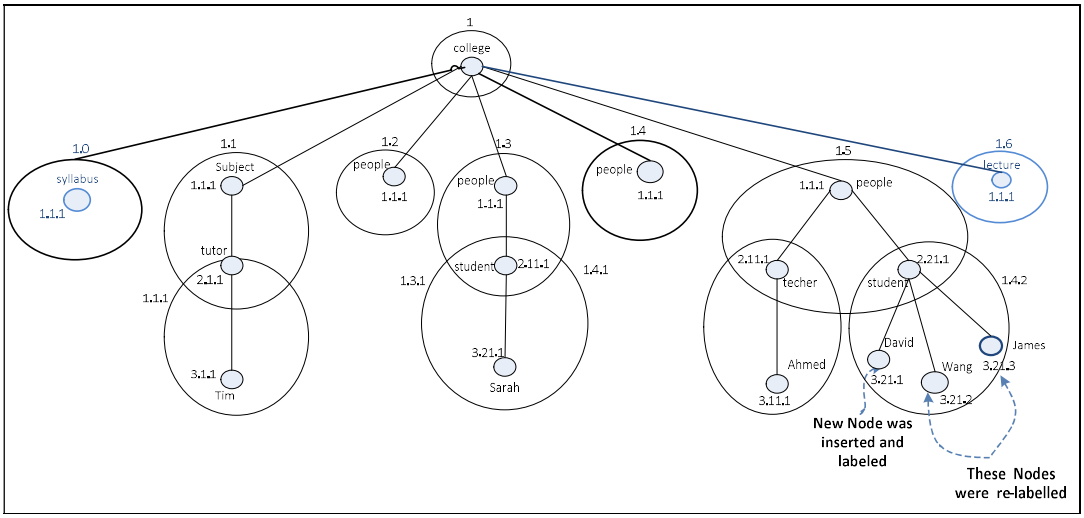


Figure 4-8: Inserting a new node in the leftmost side within a cluster

4.4.1.2.4 Inserting a Node Below a Leaf Node:

In this case, the new inserted node is not a child of the main root. Therefore, a new cluster needs to be created, and this cluster will contain this node and its parent node. The new inserted node and the new cluster are labelled. A new node *'DB'* was inserted as a chi

ld node for the node *'syllabus'* and a new cluster was created. Subsequently, the cluster and node were labelled as Figure 4-9 illustrates this example.

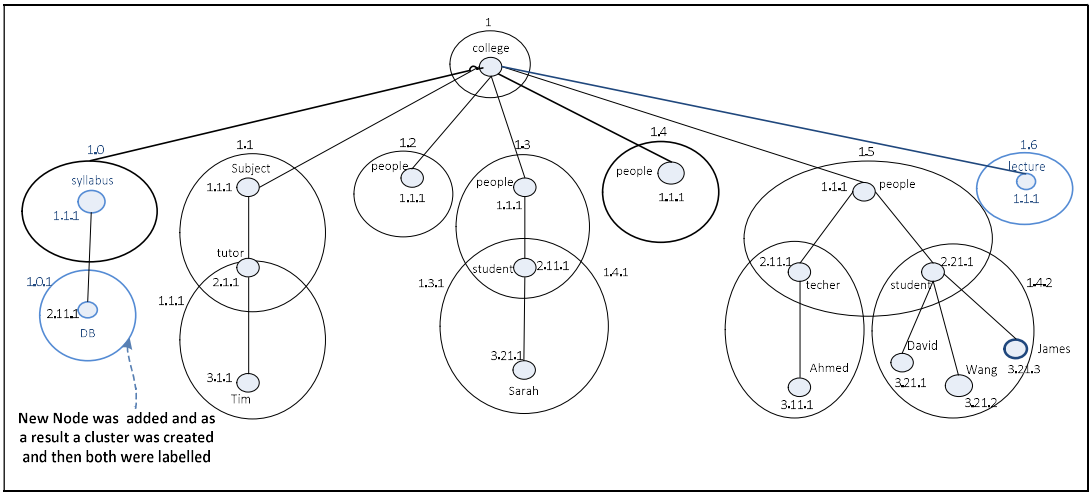


Figure 4-9: The updated labelling scheme structure

4.4.1.3 Update Cost of the Proposed Labelling Scheme:

Update processes that take place in the XML documents affect the databases and indexes. Thus, the proposed scheme aims to reduce the update cost to the lowest possible level. Reducing the number of required relabelling processes is the main technique to reduce the update cost. Using the clusters technique helps in reducing these relabelling processes. The following number of scenarios for the update cost for the proposed scheme against the LLS scheme and the Dewey labelling scheme show how the update cost was improved:

- **Scenarios for Comparing the Proposed Scheme with other Labelling Schemes:**

Figure 4-10 demonstrates an example of inserting a new node. This node is inserted into a cluster that has only one node and the root is a child of the main root. The figure also shows the required relabelling nodes for the proposed scheme and the LLS and Dewey labelling schemes. In this scenario, the proposed scheme can provide better performance than the others as there is no need for any relabelling in the proposed scheme, whereas relabelling operations are required in the other two schemes. The result of this scenario is a common case with the proposed scheme.

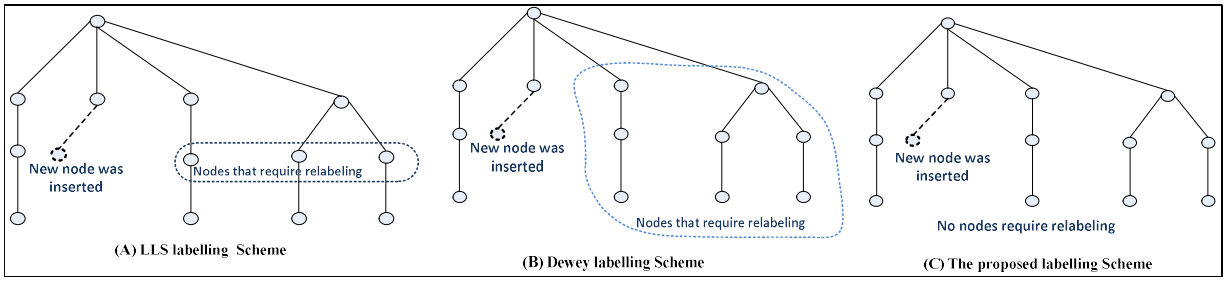


Figure 4-10: A scenario for comparing the proposed labelling schemes with the LLS and Dewey schemes

The worst scenario for the proposed scheme occurs when a new node is inserted between two nodes in the second level. Thus, a new cluster needs to be created and then some of the same level clusters need to be relabelled. In this case the relabelling operations in the proposed scheme will be the same as the LLS scheme, and better than the Dewey scheme. However, this scenario is an uncommon case. Figure 4-11 shows an example of this scenario.

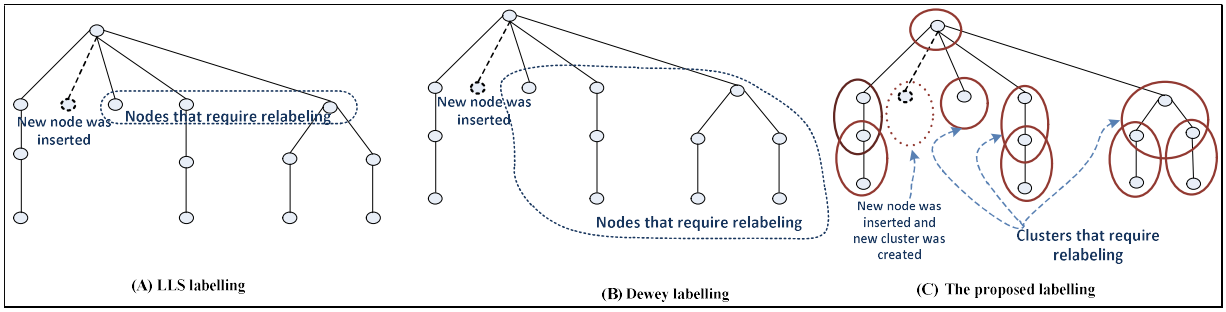


Figure 4-11: The worst case relabelling scenario for LLS and Dewey, and the proposed labelling schemes

4.5 Design and Implementation

Figure 4.12 shows the design of the proposed scheme and the platform and how the proposed scheme works. As can be seen, the first step is to parse the XML document. The DOM was selected for this platform because it is the most suitable option as it is a commonly used parser and straightforward to apply. The XML DOM is a standard means for accessing and manipulating XML documents. It has also already been employed by researchers for designing a labelling scheme (Almelibari, 2015).

Java 1.8 was used for implementing the algorithms for the proposed scheme and the LLS and Dewey labelling schemes. MySQL Workbench was used for building the databases and as a database management system to save the data for all schemes.

The second step is to label each node and any linked cluster. More details are given below.

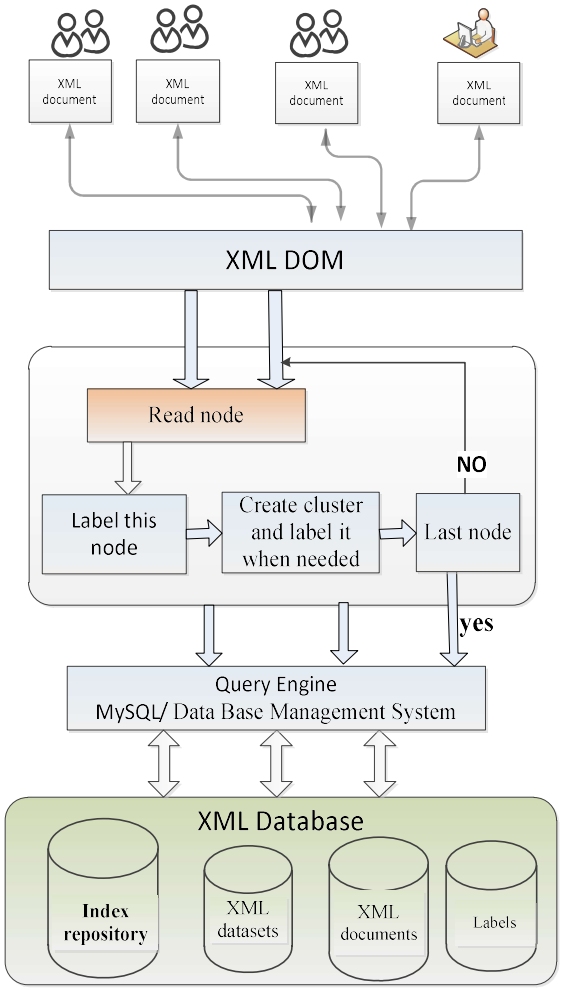


Figure 4-12: Design of the proposed scheme and the platform

4.5.1 Labelling XML Documents:

The first step is to read the XML document and save all nodes in a *nodelist*. The following Figure 4-13 shows the pseudo code for this function.

```
1-  If (mainNode !=null) {
2-      Level=0
3-      Depth=0
4-    nodeList = get all nodes into the nodeList
5-    Passing the nodeList to the indexer
6-  Invoking the indexer
```

Figure 4-13: Pseudo code for getting all nodes in a node list

- **Labelling Nodes and Clusters:**

Inserting a new node has different types of cases such as the node is a first child for its parent, or if the node is inserted between two existing nodes and so on. The algorithm below shows the general steps for inserting new nodes:

Algorithm: inserting new node

Input: a new node to be inserted somewhere in the data tree.
Output: the new node is inserted.

1 Determine the location of the new node.
2 Insert the new node in the assigned position.
3 Check the position of the new node with the cluster.
4 Create cluster if required.
5 Label the new cluster if any was created.
6 Relabel any cluster(s) that require relabelling.
7 Label the new node accordingly.
8 Relabel any old node(s) in the new node's cluster if any require relabelling.

Figure 4-14 demonstrates the pseudo code for labelling new nodes and clusters and taking into account all possible cases.

4.5.2 Implementing the LLS and the Dewey Labelling Schemes:

The evaluation of the proposed scheme is based on testing the proposed scheme against the LLS scheme and the Dewey labelling scheme. Thus, the implementation of both the LLS and Dewey labelling schemes was required as they are not available either as open source code or on the shelf software.

```
1-  If level =1 { //This part only for level 1
2-                Start_level = start level + ".1.1"
3-                 Last_Main_Index = "1.1" }
4-  //Compare current and old node names
5-  // and save  the current node name into the variable
6-   Current_node_name = null
7-   If (Current_node_name!= null & != empty) {
8-         Current_node_name= nodeName}
     // check if the node is duplicated then label it in the proper way
9-   If ( old node is not empty ) {
    //Then compare old node name with current node name and also compare current depth
    level with last depth level
10-   If (oldNodeName = currentNodeName & both nodes from the same level) {
11-        LastValue = LastValue+ 1 // Add the last component + 1
12-        Start_level = Last_Main_Index + "." +
          }
13- Else
14- If ( level > 1) { //To make sure this code doesn't work with Level 1
15-           //Save entire indexing into Last_Main_Index  } } }
16- Else { //if old name is empty
17- If ( level > 1) { //To make sure this code doesn't work with Level 1
18-  Last_Main_Index  } } //Save entire indexing into
    // handles labelling all children nodes
    // check if any node name is duplicated
19- If (ParentNodeName = currentNodeName) & parentNodeLevel =level) {
20- lastValue= lastValue +1 // add the last component +1
21- nodeLabel= Last_Main_Index + lastValue  }
    //This part save current depth level into last depth level
22- lastXML_depth_level= level
23- If ( level = 1) {//This code only handles depth level 1
24-     If (parentNodeName != null)
25-        parentNodeLevel = level-1  //here parent node level is 0
26-        parentNodeIndex = 0  }//This is main node
27-    If (parentNodeName != null)
28- parentNodeLevel = level-1
29- oldNodeName = currentNodeName
30- if ( node_has_children){// To make sure if current node have any children
31- nodeList = get_node_children
32- while ( the next node exists) {
33-    child_node = currentNode }
```

Figure 4-14: Pseudo code for labelling nodes and clusters

With regard to the LLS, the original code that was supplied by the author of the LLS scheme is incomplete and was not documented at all. The author also confirmed that he can't provide a description of the code documentation. Thus, a new implementation of the LLS scheme was considered easier to accomplish than rectifying the missing parts in the original software.

The Dewey labelling scheme is not available as open source code since it is a theory that other researchers have used as a base to implement their interpretations of it in software. The Dewey labelling scheme was implemented using Java as a coding language. The implementation was according to the data model. A ***nodelist*** was used to save labels of the nodes on it (Tatarinov et al., 2002). Since the Dewey labelling scheme is just a theory and not implemented practically by the founder, it is thus not possible to validate it practically.

As a result, both schemes were implemented by the author of the proposed scheme as bespoke software. This implementation was as accurate as possible and according to their data models, in order to prove the validity. With respect to the technologies, Java1.8 was used to code the algorithms; *nodelists* were used to save the labels of the nodes. The LLS scheme founder used *XMark* and DBLP as datasets to test the LLS labelling scheme (S. A. Mohammad, 2011).

4.5.3 Validating the LLS Scheme

XMark datasets were used to prove the validity of the implementation of this scheme. The Xpath was used as query language and the same types of queries were used as well. However, the specifications of the hardware and the software that was used by the founder of the LLS scheme to test it are slightly different from the ones used now to validate the implemented LLS scheme. Thus, minor differences in the results are expected and acceptable. Having executed these queries and compared their results with the LLS scheme founder's results, the results are almost identical and therefore the implementation was accepted. Figure 4-15 shows a comparison of the results for both the original LLS scheme and the implemented LLS scheme using the same four queries that used in testing the original LLS scheme which were referred as Q1,.. Q4 in the figure below.

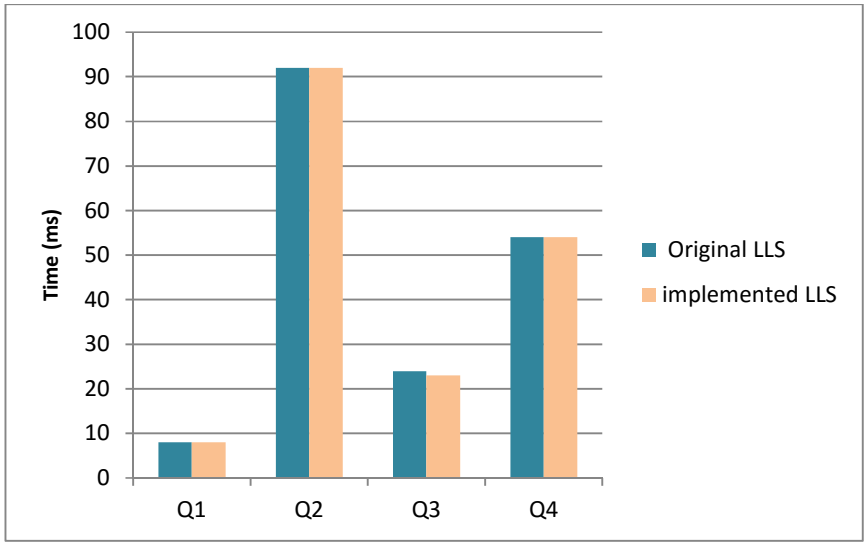


Figure 4-15: The results for the original LLS scheme and implemented LLS scheme

4.6 Experimental Setup

Testing the performance of the proposed scheme is an important task in order to assess this scheme. Thus, a number of experiments were designed and executed to test several aspects and features of the proposed scheme. These experiments were executed on the proposed scheme as well as the LLS and Dewey labelling schemes in order to compare the results. This section discusses the details of the experiments and the datasets used in these experiments.

4.6.1 Objectives of the Experiments:

The experiments were designed and implemented to meet the objectives of the proposed scheme which are the following:

1- The key aim of the experiments is to assess the performance, scalability, and efficiency of the proposed scheme.
2- Evaluating the process of labelling XML documents: the labelling process is measured according to two factors, namely, time required for labelling the document; and second, the size of the growth of the labels, and how these factors are affected by the document size.
3- Evaluating the query performance by carrying out different kinds of queries on the labelled documents before and after insertions.
4- Handling insertions: testing the scheme scalability in handling different types of insertions.

4.6.2 Types of Experiments:

1- Labelling the XML document: this experiment was carried out to evaluate the process of labelling the nodes and the clusters of the XML document. Two aspects were used to evaluate the scheme, which are time for labelling nodes and size of the growth of the labels.
2- Time for updates: these updates include inserting new nodes and deleting existing nodes. This experiment is conducted to evaluate the ability to handle different types of insertion.
3- Determining different relationships: this experiment is for assessing the time to determine the relationships.
4- Performance of the queries: this experiment is for evaluating the query response time before and after insertions. A number of different queries were carried out. These queries are used to test different features and aspects. The XMark queries were chosen to test the schemes. More details about the XMark and why it was chosen are given in the next section.

4.6.3 XML Datasets and Benchmarks:

There are many XML datasets available. Some of them are from real life datasets and others are from artificial datasets. These two types of datasets are used to evaluate the XML labelling schemes (Schmidt et al., 2001). An investigation into the most broadly used XML datasets is carried out in order to choose the most appropriate dataset(s) for testing the proposed scheme. More detail about these datasets and benchmarks as follows:

1] **Actual XML Datasets:** these datasets are based on real data. They are also called production/ existing/ experimental XML datasets. Examples of them are: SwissPort, DBLP Computer Science Bibliography, University Courses, Auction Data, NASA, Treebank, Protein Sequence Database, SIGMOD Record, Mondial, and TPC-H Relational Database Benchmark (Schmidt et al., 2001).

2] **Benchmark/Standard Datasets:** also, called artificial datasets. These benchmarks were developed for the purpose of evaluating queries. XML benchmarks are categorised into two groups, namely, micro benchmarks and application benchmarks. Micro benchmarks are used to evaluate specific parts of a system whereas application benchmarks are used to evaluate the performance of an XML database in general (Barbosa, Mendelzon, Keenleyside, & Lyons, 2002; Kanda Runapongsa, 2006; Mlýnková, 2008; Schmidt et al., 2001). Examples of them are: XMark Benchmark, TPox benchmark, Michigan Benchmark, XBench benchmark, and XMack-1 Benchmark. These XML benchmarks are intended for both query processing and storing data (Schmidt et al., 2001). More details about the most used datasets and benchmarks as follows:

- **XMark Benchmark:** this was proposed in 2002 by Schmidt et al. (Schmidt et al., 2002) and is mainly used to evaluate XML applications. It is one of the most used XML benchmarks nowadays. The XMark has a data generator called *xmlgen*. This generator can produce an artificial XML document that is based on the DTD of an internet database. This generator is available free of charge at the XMark project website. XMark can recreate an XML database in different sizes. Thus, users can generate their own datasets that are appropriate for their requirements. In addition, the XMark datasets are used to evaluate the system performance efficiently. Twenty queries are included in XMark and they are used to assess different aspects of searching in databases. These queries do not include the update processes (Schmidt et al., 2002).

- **XOO7 Benchmark:** this was introduced by Carey et al. (Carey, DeWitt, Kant, & Naughton, 1994) and is called Object Oriented RDBMS benchmark (OO7). It was,

subsequently, implemented on XML data by Li et al. (Q. Li & Moon, 2001). The XML dataset that XOO7 creates is a separate XML file. This file is created in three different sizes – small, medium and large. This XML dataset has only up to five levels and offers twenty-three queries. These queries only handle search operations (Carey et al., 1994). The XOO7 benchmark can be downloaded free of charge from its website (Bressan et al., 2001).

- **Michigan Benchmark:** also called MBench. Introduced by Runapongsa et al., this benchmark was designed as a micro benchmark in order to evaluate specific system components (Runapongsa et al., 2006). This benchmark's dataset has forty-six queries and seven update processes (Mlýnková, 2008). It comes as an XML file that includes a number of nodes starting from 728,000 nodes and up to ten times more. In addition, this dataset has limited depth which is sixteen levels, whereas the width is changeable. With regard to the queries, this benchmark has thirty-one queries that handle and evaluate many different features of databases containing update operations (Runapongsa et al., 2006).

- **XML Data Management Benchmark (XMach-1):** this was introduced by (Böhme & Rahm, 2003) and supports multiusers. The dataset of the XMach-1 includes a large number of XML files with sizes between 2 KB and 100 KB. The number of levels is limited up to six levels. The query set has eleven queries, three of them for update processes and the other eight queries for search processes. This benchmark is supported by web applications and is comprised of four parts, namely XML database, server, loader and client. The application servers provide XML document handling. The loaders handle the processes of detecting and loading the XML data from the database. The clients query and retrieve XML data (Böhme & Rahm, 2001). Figure 4-16 shows the components of the XMach-1.

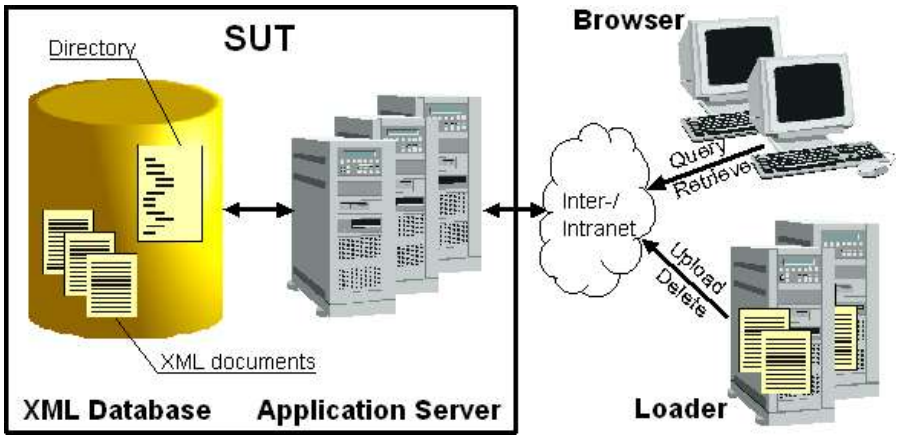


Figure 4-16 : XMach-1 components (Böhme & Rahm, 2001)

- **XBench Benchmark:** this creates a very wide range of XML files such as data centric and text centric. The database of this benchmark could be a single or multi XML file. The XBench is provided with a generator to generate XML data. This generator is built on ToXgene data generator and can generate different sizes of files from 10 MB to 10 GB. This benchmark has twenty queries that support search only without update (Benjamin B Yao, Özsu, & Keenleyside, 2003; Benjamin Bin Yao, Ozsu, & Khandelwal, 2004).

- **TPoX Benchmark:** TPoX stands for Transaction Processing over XML. This benchmark was designed to evaluate the whole system. The schema of the benchmark controls the size of the XML files. Regarding the XML database, it includes many small XML files with sizes between 2 KB and 20 KB. This benchmark has seventeen queries that mainly focus on updates (Nicola, Kogan, & Schiefer, 2007).

Having investigated these datasets, the XMark benchmark has been chosen for the testing experiments. The XMark benchmark helps both implementers and users to obtain insights into the XML storage. XMark was chosen to test the proposed scheme for the following reasons: first and most importantly, the XMark was used to evaluate the performance of the LLS scheme by the founder. Thus, it would be appropriate to use the same dataset to evaluate the proposed scheme and compare the results (Samir Mohammad & Martin, 2010). Secondly, this benchmark is widely used to test XML queries and XML database performance (Almelibari, 2015). Moreover, XMark is a good choice since it has many features such as providing a document generator to create documents in different sizes. Thus, users can generate datasets that are appropriate for their requirements (Schmidt et al., 2002). Also, this benchmark provides a binary version of the XMark that can be run as an independent platform on any operating system. XMark provides a broad range of queries – twenty-one in total. These queries are designed to evaluate different aspects of the datasets. They are divided into groups based on their goals and purposes. Table 4-1 shows these groups (Schmidt et al., 2002). Query number 10 of the XMark queries was eliminated since it is irrelevant to the proposed scheme as this query is used to translate the results into another language.

To conclude, the above XMark features offer great choice for evaluating the proposed scheme.

TABLE 4-1: XMARK BENCHMARK QUERIES (SCHMIDT ET AL., 2002)

No	*Query number*	*Group name*	*Description*
1	Q1	Exact match	Return the name of the person with ID 'person0'.
2	Q2	Ordered access	Return the initial increases of all open auctions
3	Q3		Return the first and current increases of all open auctions whose current increase is at least twice as high as the initial increase
4	Q4		List the reserves of those open auctions where a certain person issued a bid before another person.
5	Q5	Casting	How many sold items cost more than 40?
6	Q6	Regular path expression	How many items are listed on all continents?
7	Q7		How many pieces of prose are in our database?
8	Q8	Chasing references	List the names of persons and the number of items they bought. (joins person, closed auction)
9	Q9		List the names of persons and the names of the items they bought in Europe. (joins person, closed auction, item)
10	Q11	Joins on values	For each person, list the number of items currently on sale whose price does not exceed 0.02% of the person's income.
11	Q12		For each person with an income of more than 50,000, list the number of items currently on sale whose price does not exceed 0.02% of the person's income.
12	Q13	Reconstruct portions of the original XML document	List the names of items registered in Australia along with their descriptions.
13	Q14	Full text	Return the names of all items whose description contains the word 'gold'.
14	Q15	Path traversals	Print the keywords with emphasis in annotations of closed auctions.
15	Q16		Return the IDs of the sellers of those auctions that have one or more keywords emphasised.
16	Q17	Finding missing elements	Which persons don't have a homepage?
17	Q18	Function	Convert the currency of the reserves of all open auctions

		application	to another currency.
18	Q19	Sorting	Give an alphabetically ordered list of all items along with their location.
19	Q20	Aggregation	Group customers by their income and output the cardinality of each group.

4.7 Testing

The overall purpose of testing the proposed scheme is to ensure it achieved objectives. The experiments were executed twenty times for each query and then the average was taken in order to gain as accurate results as possible. This number of executions for each experiment was chosen as a fair number to get an accurate result by calculating the average of these twenty executions. Other researchers used the same number for testing similar labelling schemes (Almelibari, 2015).

The items that need to be tested are: the proposed scheme, the LLS labelling scheme, and the Dewey labelling scheme. Furthermore, there are certain features that need to be tested, namely the query performance, efficiency of labelling XML documents, efficiency of scalability, and functionality of the proposed scheme. The testing technique is based on running nineteen queries of the XMark dataset to test the target schemes in order to compare the results and ascertain the achievement of the proposed scheme.

4.7.1 Testing Platform and Environment

An appropriate platform and testing environment for certain experiments is an important task in order to gain accurate results. The following issues and tasks were considered and carried out:

1- Use appropriate XML datasets and benchmarks: as discussed above, a review of the most commonly used XML datasets for testing the labelling scheme had already been carried out, and as a consequence, the XMark was chosen for the testing. More details are given above in section 3.2.3.

2- Identify the environment of the experiments: all experiments were carried out on a desktop with the following specifications:
   - Processor: Intel Core i5
   - CPU 3.20 GHz
   - RAM: 8GB

   With respect to the software:
   - Windows 7 Enterprise.
   - MySQL Workbench 6.3 as a database management system.
   - Java 8 as a programming language for coding the algorithms.

This software and hardware was used for the implementation and testing stages. They were selected carefully as the most appropriate specifications for the testing requirements.

4.7.2 Data Analysis and Presentation:

This process starts from the stage of data collection as raw data moving to significant findings. The raw data of the experiments is the selected XML datasets which is the XMark benchmark.

4.7.2.1 Mechanism of Generating Data:

The data raw is processed by the algorithm of the proposed labelling scheme. This process labels and manipulates the raw data. The raw data here is XML documents. These raw data include eleven files in different sizes which generated by the XMark Benchmark. Table 4-2 below shows these data files and their purposes. The XMark-1 file is the smallest file (1 MB) and was used to test all experiments because it facilitates the observation of the changes when updates to the document take place. The other ten files started from 10 MB an d reached 100 BM. The sizes of these files differed by 10 MB. These files were generated in these different sizes in order to test the capability of the proposed scheme to manage the scalability and other targeted features. These files were used to measure the time and size for labelling XML documents.

TABLE 4-2: XMARK FILES FOR EXPERIMENTS

File name	***Size***	***Targeted experiments***
XMark-1	1 MB	All experiments
XMark-2	10 MB	Measuring time and size for labelling XML documents
XMark-3	20 MB	
XMark-4	30 MB	
XMark-5	40 MB	
XMark-6	50 MB	
XMark-7	60 MB	
XMark-8	70 MB	
XMark-9	80 MB	
XMark-10	90 MB	
XMark-11	100 MB	

Briefly, the processing of data raw produces indexes that are stored in database files. These indexes are the labels of XML documents. The results of this processing are represented in the spent time of certain procedures and the size of certain files. The simple figure 4-17 below shows the stages of generating data.

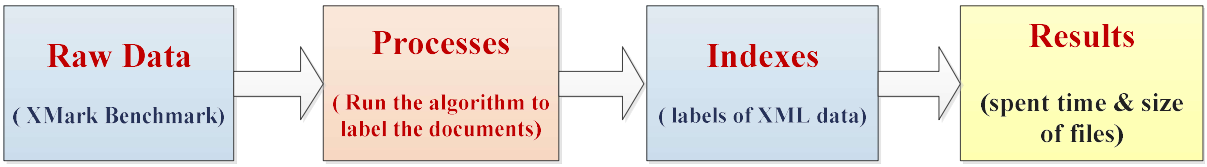


Figure 4-17: process of generating data

4.7.2.2 Statistical Data Analysis Procedures:

The presentation of the results in graphs is a crucial task as it offers a descriptive view for the differences among the targeted schemes for the testing. Moreover, most of the results of the experiments are doubtful and hard to be sure about it. Thus, in terms of statistics, the hypothesis test is widely accepted as a process to achieve trusted answers (Currell, 2015). Furthermore, it was suggested that calculating the statistical significance helps in clarifying the differences between the results of two schemes as obtaining the statistical significance. The null hypothesis is used to measure the statistical significance. The null hypothesis is tested by checking if the targeted schemes are equal, otherwise the alternative hypothesis is accepted (Benjamini, 1988).

Briefly, hypothesis testing (a. k. a. *t* testing) is used to evaluate the research question and the research hypothesis. However, as the targeted schemes for the testing are non-parametric. Thus, the t-test is not suitable for this case and the p-value would be the alternative. Mainly, the p-value assists the researchers to test their hypotheses. Moreover, the p-value is also used as the probability of achieving a result that the same or much more than the one was observed. Thus, p-value is broadly used (Currell, 2015; Gray, 2013).

4.7.2.3 Calculate the Statistical Significance: Statistical significance means that the results of the experiments are not by chance or randomly. Thus, the statistical significance was chosen for this research as a statistical approach to test the hypothesis. The statistical significance in this research concentrates on measuring the performance in terms of time and size.

The box plot has been recognised as a suitable method for numerical data presentation as quartiles, as well as descriptive statistics. The box plot technique as a statistical procedure is also used to obtain the p-values. The interpretation of the figures that the box plot provided is very straightforward since the p-values are used to find the significance between the tests. (Nuzzo, 2016). This interpretation is straightforward due to the utility of p-values in finding the significance between the tests. The p-value is calculated by setting a null hypothesis (H0) for the targeted schemes in order to test the statistical significance. In this regard, scientists usually set the significance level for the experiments is 0.05, which means that only at most 5 percent of the results are by chance

or randomly can be accepted. The justification for choosing the box plot is that it is a suitable way for non-parametric data.

The time and size of populations of eleven XML documents which showed in table 4-2 above are calculated. These eleven populations are measured for every variable such as time of labelling XML documents, size of labelling XML documents, determining different relationships, query performance, and so on.

The SPSS (Statistical Package for the Social Sciences) will be used as a data analysis tool. Initially, this software was introduced for social sciences as its name implies. However, nowadays this software is very well known cross many fields. All the box plots are generated by the SPSS. the Wilcoxon rank sum test will be used to calculate the p-value. The Wilcoxon rank sum test is a convenient in figuring out the statistical significance based on the p-value (Rousseeuw et al., 1999).

4.8 Summary

This chapter described the state-of-the-art proposed scheme including the mechanism and design, as well as implementation and the experimental setup. In more detail, the chapter was divided into four main sections, namely, research approach, experiment setup, testing, and summary.

In section 4.1 an introduction was provided. In section 4.2 the existing approaches and our approach were discussed and compared. The data models of existing labelling schemes were discussed in section 4.3. Moreover, section 4.4 started by demonstrating the mechanism of labelling XML documents, and how this scheme can create clusters and then label them as well as their nodes according to the theory and data model of this scheme. This section also discussed the data model and how the proposed scheme can determine the node order and the inserting new nodes process. Furthermore, this section explained the update cost of the proposed scheme and then illustrated examples for inserting new nodes and how the proposed scheme handles them. Subsequently, section 4.5 demonstrated the design and implementation of the proposed scheme, including the algorithms and the pseudo code. Section 4.5.2 considered the implementation of the LLS and Dewey labelling schemes for evaluation purposes, followed by section 4.5.3 which explained the validation of the LLS and Dewey labelling schemes. In section 4.6, the experimental setup including experiment objectives and types were discussed. In addition, this chapter discussed the testing stage in section 4.7.

The mechanism used in the proposed scheme which is based on dividing the whole data tree into small sub-trees called clusters is a supportive part of enhancing the efficiency of the scheme. The basic idea of this mechanism has already been used in developing a labelling scheme by Almelibari (Almelibari, 2015). However, some differences occurred

due to the nature of the aim and objectives of this research; for instance, labelling these clusters and their nodes by using two existing labelling schemes, which were selected carefully and based on justifications, as discussed in the previous chapter. Having carried out these tasks, objective 2 was achieved successfully.

The stage of design and implementation of the proposed scheme is a crucial stage. All steps were considered and justified carefully, from designing the proposed scheme to implementation of this design and all other required components such as implementing the LLS and Dewey labelling schemes. Therefore, objective 3 of this research was achieved successfully.

Section 4.6 discussed the experimental setup. In more detail, the testing objectives and types of experiments were explained in sections 4.6.1 and 4.6.2 respectively. A deep review of the most common XML datasets and benchmarks in order to pick up one or more of these datasets for experimental purposes were discussed in section 4.6.3.

Section 4.7 includes the testing stage. This section discussed the testing platform and environment, testing objectives, and finally testing measurements and metrics in sections 4.7.1, 4.7.2, and 4.7.3 respectively. This stage is a part of the setup for the evaluation stage which has been carried out successfully.

5. Results

5.1 Introduction

The results are illustrated and discussed in this chapter. These are the results of the experiments that were described in the previous chapter. These experiments were developed in order to assess a number of features and functions of the proposed scheme such as time required for labelling clusters and nodes, time for defining the relationships among nodes, size of the labels, and so on. These experiments were designed to test the proposed labelling scheme as well as the LLS and Dewey labelling schemes in order to compare the results.

Previous research that tested labelling schemes found that there is a fluctuation between the results of certain repeated experiments (Fennell, 2013). Therefore, results of experiments that come from just a few repeated experiments could be inaccurate and unreliable (Murata et al., 2000). Similar research repeated each experiment twenty times and their average was calculated (Almelibari, 2015). Therefore, the experiments of this research were repeated the same number of times in order to gain accurate results.

With respect to the datasets, XMark benchmark was chosen for testing the experiments as explained in the previous chapter. Different sizes of files were created to test different experiments as table 4.2 shows. Different file sizes were used to assess whether size can influence the calculated time and size.

The experiments were carried out to test both static and dynamic documents separately as follows:

5.2 Experiments for Static Documents

5.2.1 Labelling XML Documents:

The idea of this experiment is to compute the time needed for assigning labels to the XML data nodes. These labels are assigned according to the proposed labelling scheme. In addition, the growth of the label's size is computed by using different file sizes.

5.2.1.1 Measuring Time:

This is calculated according to the method of the proposed scheme for labelling nodes. Thus, there is a rapid increase in time when the file size increases by 10 MB. Figure 5.1 illustrates the time needed for labelling XML files by using the proposed scheme.

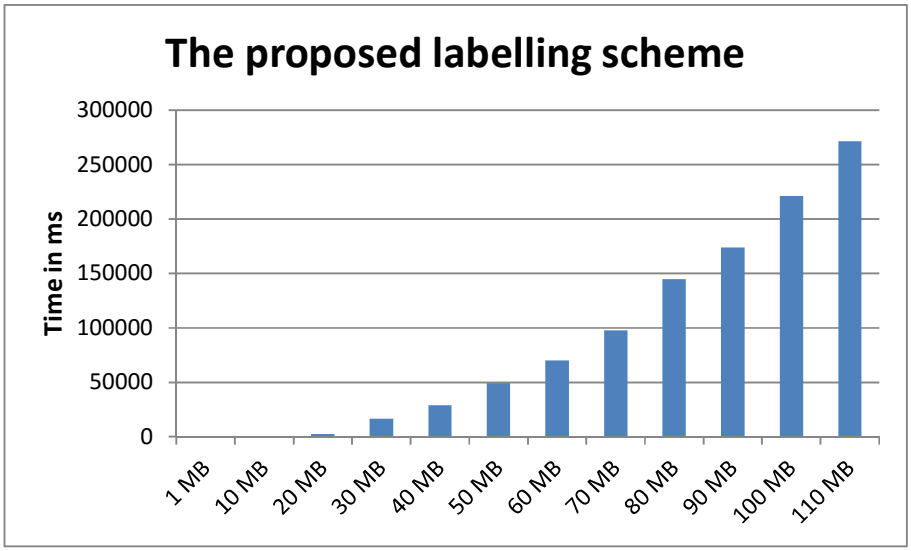


Figure 5-1: time needed for labelling XML files by the proposed scheme.

As the above figure shows, there is a rapid increase of time when the file size increases by 10 MB. Executing this experiment on the LLS scheme also shows a rapid increase between the time and file size as shown in figure 5-2.

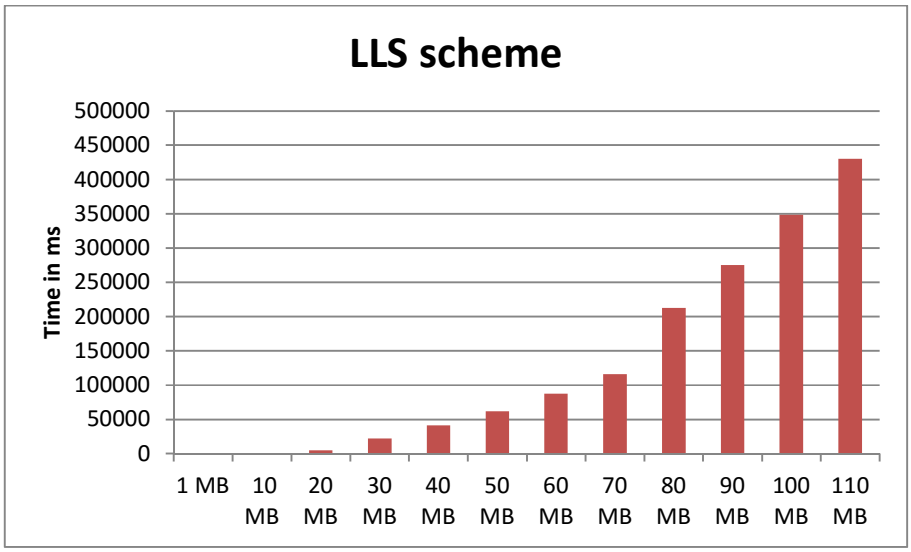


Figure 5-2: time needed for labelling XML files by using the LLS scheme.

Executing this experiment on the Dewey labelling scheme shows a similar increase to the previous two experiments, as figure 5.3 shows.

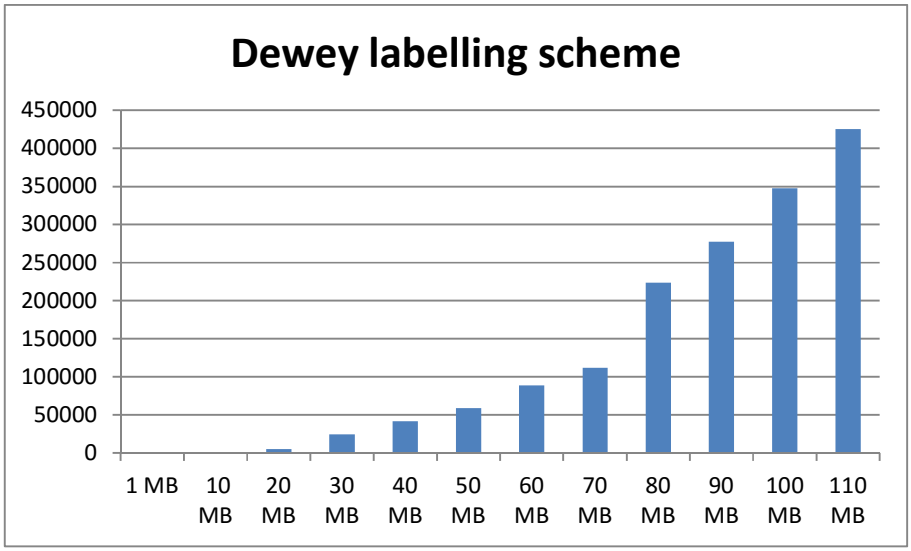


Figure 5-3: time needed for labelling XML files by using the Dewey scheme.

All labelling schemes demonstrate a great time increase. Figure 5.4 shows a comparison for the results for all these schemes. The spent times for labelling XML files by the LLS and Dewey labelling schemes are longer than the spent time for labelling XML files by the proposed scheme.

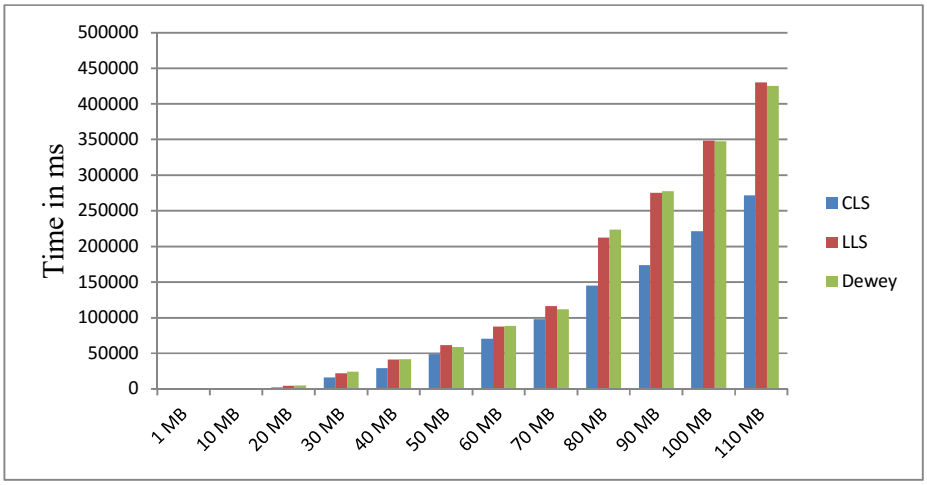


Figure 5-4: time needed for labelling XML files.

5.2.1.2 Measuring Size: observing the increasing size of the labels while labelling XML documents shows that all schemes have a significant increase in the size of the labels. However, these increases are different from each other. The proposed scheme could not beat the other two schemes in this regard, and had a higher increase. This increase is due to the method that the proposed scheme used to label the nodes and clusters, which means that two labels need to be saved for each node, and as a result a bigger space in the memory is equipped. Figure 5.5 shows these results.

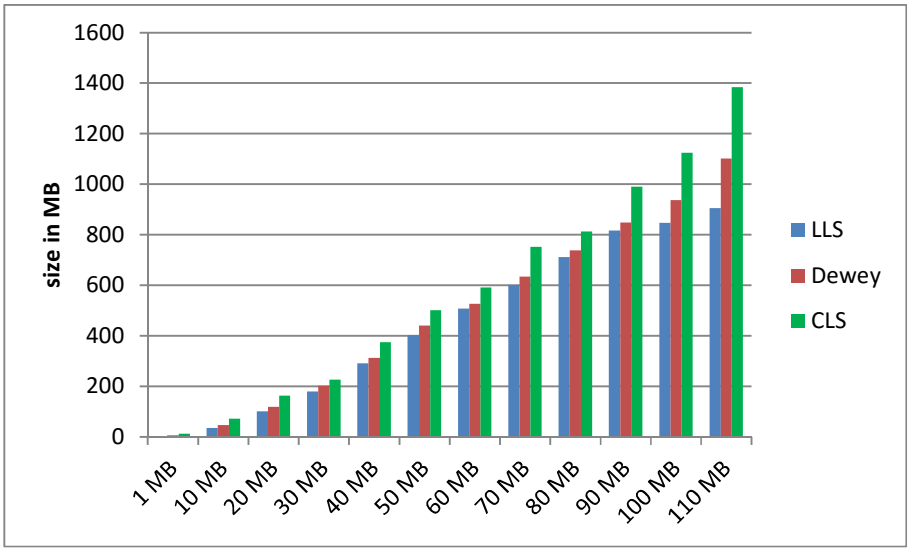


Figure 5-5: the growth of the size of labelling XML files for all schemes.

5.2.1.3 Statistical Description of the Results:

The proposed labelling scheme shows difference in spent time for labelling XML documents from the LLS and Dewey schemes. Eleven populations were used to measure the time for labelling nodes; these populations were illustrated in figure 5-5 in the Methodology chapter. Consequently, the box plots illustrate the distributions that were moved to the proposed scheme, which means that the proposed scheme shows better performance in this regard.

The p-values results gained from testing the eleven different sizes of populations show the results are below the significance level, which is 0.05. As a result, the null hypothesis was rejected and the alternative hypothesis was supported. Therefore, the proposed scheme achieved faster performance than the LLS and Dewey labelling schemes.

Figures from 5-6 to 5-25 show the box plots and p-values for this testing.

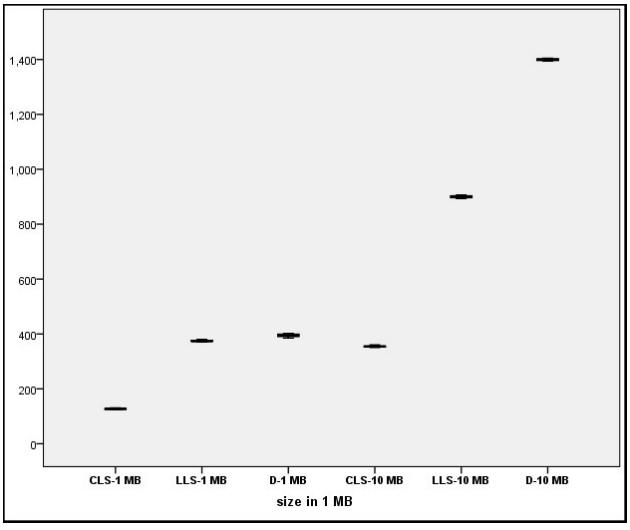


Figure 5-6: Box plots between the CLS & LLS & Dewey schemes using XML files of size 1 MB and 10 MB for labelling XML documents.

p-value @ 1 MB between CLS & LLS schemes =	0.041
p-value @ 1 MB between CLS & Dewey schemes =	0.043
p-value @ 10 MB between CLS & LLS schemes =	0.043
p-value @ 10 MB between CLS & Dewey schemes =	0.042

Figure 5-7: the p-value between the CLS, LLS and Dewey schemes using XML files of size 1 MB and 10 MB for labelling XML documents.

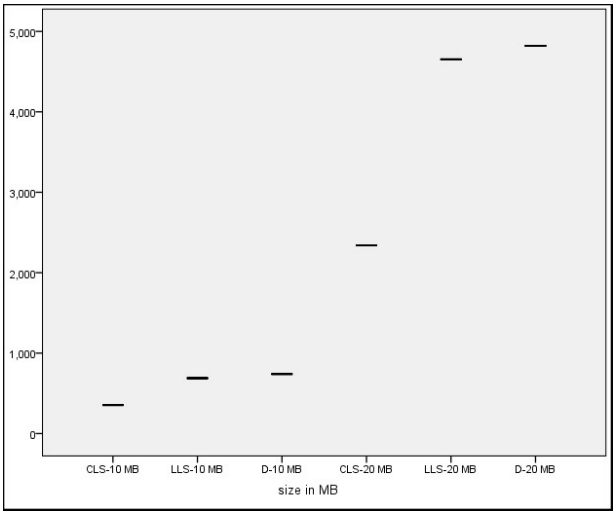


Figure 5-8: Boxplot between CLS, LLS, & Dewey schemes using XML files of size 10 MB and 20 MB for labelling XML documents.

p-value @ 20 MB between CLS & LLS schemes =	0.045
p-value @ 20 MB between CLS & Dewey schemes =	0.047

Figure 5-9: the p-value between the CLS, LLS and Dewey schemes using XML files of size 20 MB XML files for labelling XML documents.

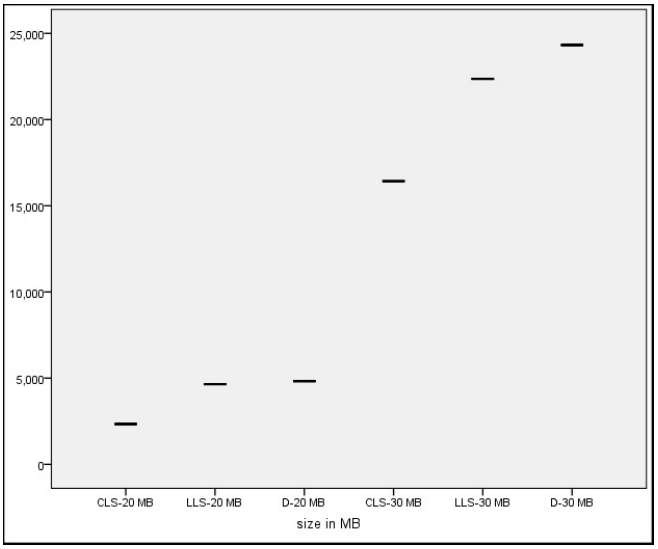


Figure 5-10: Box plot between CLS, LLS, & Dewey schemes using XML files of size 20 MB and 30 MB for labelling XML documents

p-value @ 30 MB between CLS & LLS schemes =	0.004
p-value @ 30 MB between CLS & Dewey schemes =	0.004

Figure 5-11: the p-value between the CLS, LLS and Dewey schemes using XML files of size 30 MB for labelling XML documents.

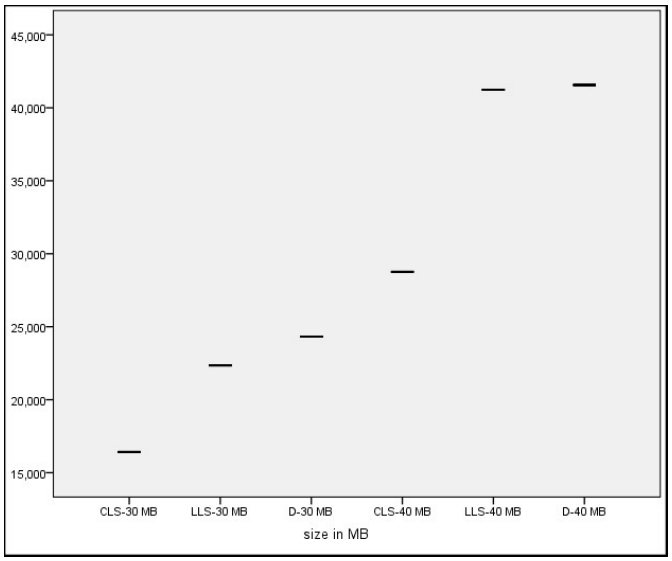


Figure 5-12: Box plot between CLS, LLS & Dewey schemes using XML files of size 30 MB and 40 MB for labelling XML documents.

p-value @ 40 MB between CLS & LLS schemes =	0.018
p-value @ 40 MB between CLS & Dewey schemes =	0.018

Figure 5-13: the p-value between the CLS, LLS and Dewey schemes using XML files of size 40 MB for labelling XML documents.

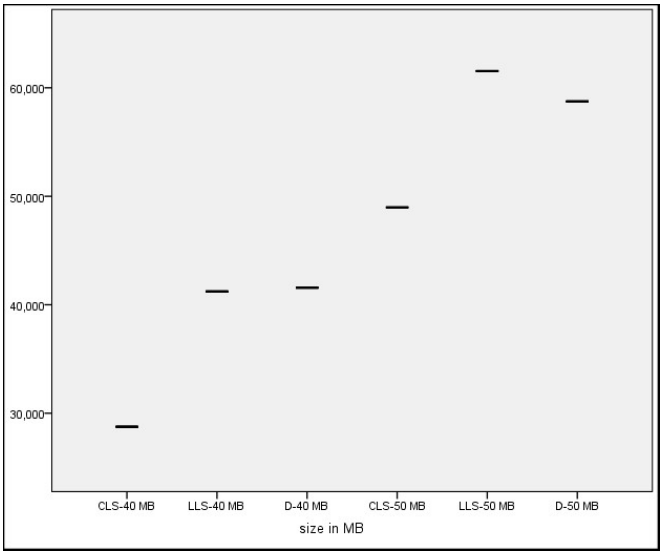


Figure 5-14: Boxplot between CLS, LLS & Dewey schemes using XML files of size 40 MB and 50 MB for labelling XML documents.

p-value @ 50 MB between CLS & LLS schemes =	0.007
p-value @ 50 MB between CLS & Dewey schemes =	0.006

Figure 5-15: the p-value between the CLS, LLS and Dewey schemes using XML files of size 50 MB for labelling XML documents.

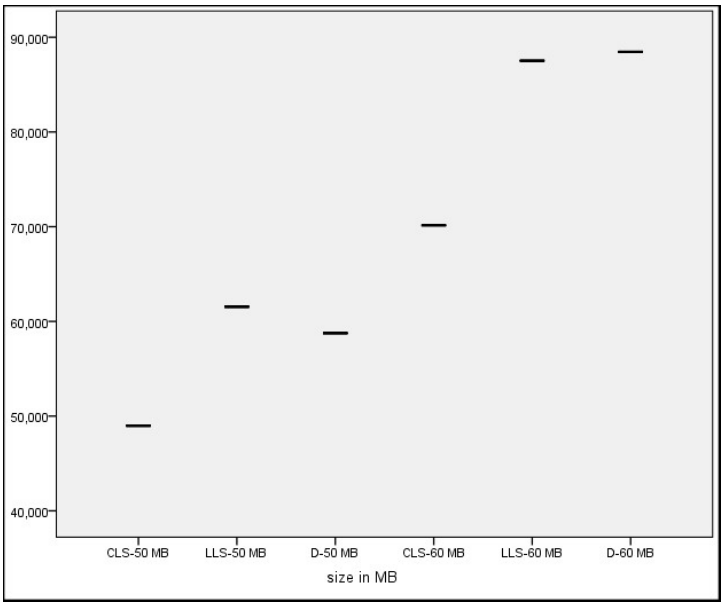


Figure 5-16: Boxplot between CLS, LLS & Dewey schemes using XML files of size 50 MB and 60 MB for labelling XML documents.

p-value @ 60 MB between CLS & LLS schemes =	0.007
p-value @ 60 MB between CLS & Dewey schemes =	0.007

Figure 5-17: the p-value between the CLS and & LLS schemes using XML files of size 60 MB for labelling XML documents.

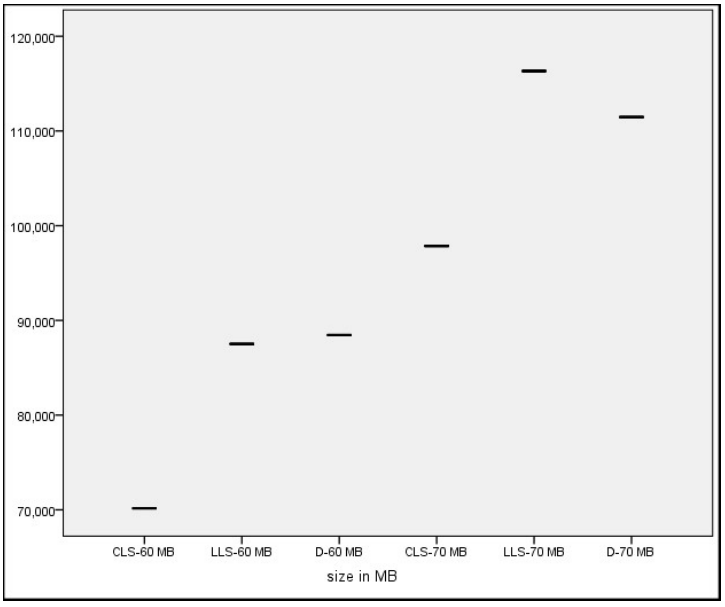


Figure 5-18: Boxplot between CLS, LLS & Dewey schemes using XML files of size 60 MB and 70 MB for labelling XML documents.

p-value @ 70 MB between CLS & LLS schemes =	0.006
p-value @ 70 MB between CLS & Dewey schemes =	0.007

Figure 5-19: the p-value between the CLS, LLS & Dewey schemes using XML files of size 70 MB for labelling XML documents.

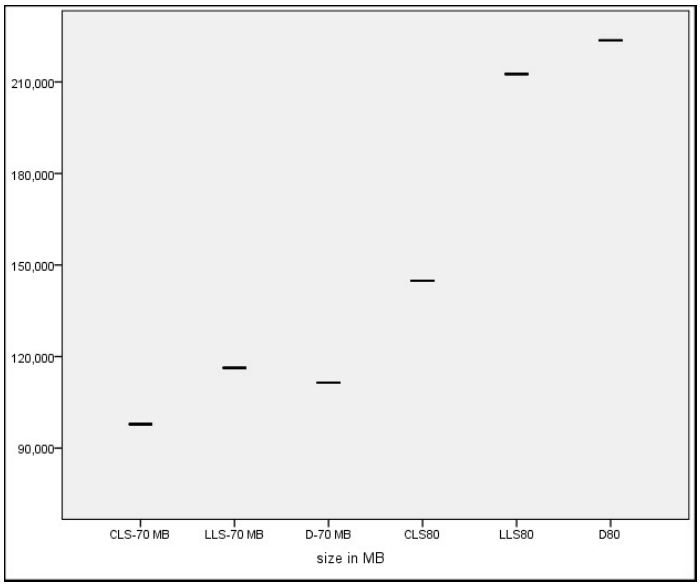


Figure 5-20: Boxplot between CLS, LLS & Dewey schemes using XML files of size 70 MB and 80 MB for labelling XML documents.

p-value @ 80 MB between CLS & LLS schemes =	0.012
p-value @ 80 MB between CLS & Dewey schemes =	0.007

Figure 5-21: the p-value between the CLS, LLS & Dewey schemes using XML files of size 80 MB for labelling XML documents.

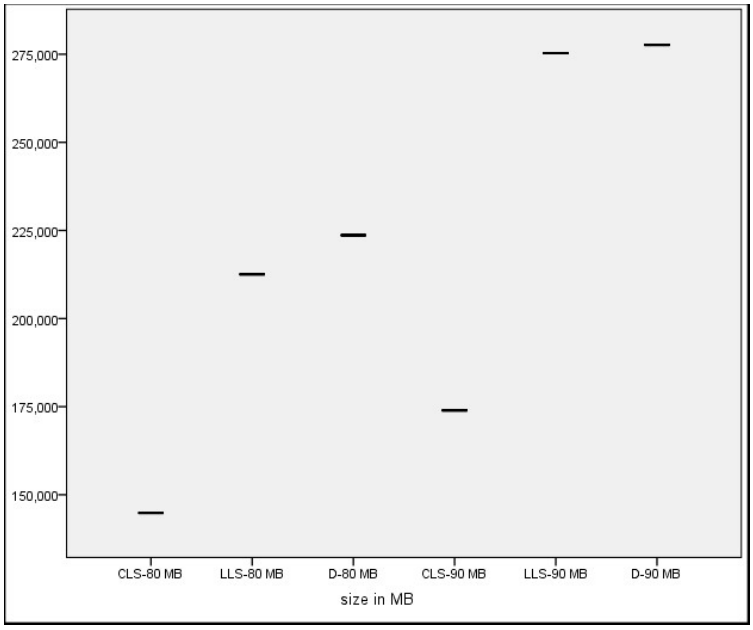


Figure 5-22: Boxplot between CLS, LLS & Dewey schemes using XML files of size 80 MB and 90 MB for labelling XML documents.

p-value @ 90 MB between CLS & LLS schemes =	0.007
p-value @ 90 MB between CLS & Dewey schemes =	0.005

Figure 5-23: the p -value between the CLS, LLS and Dewey schemes using XML files of size 90 MB for labelling XML documents.

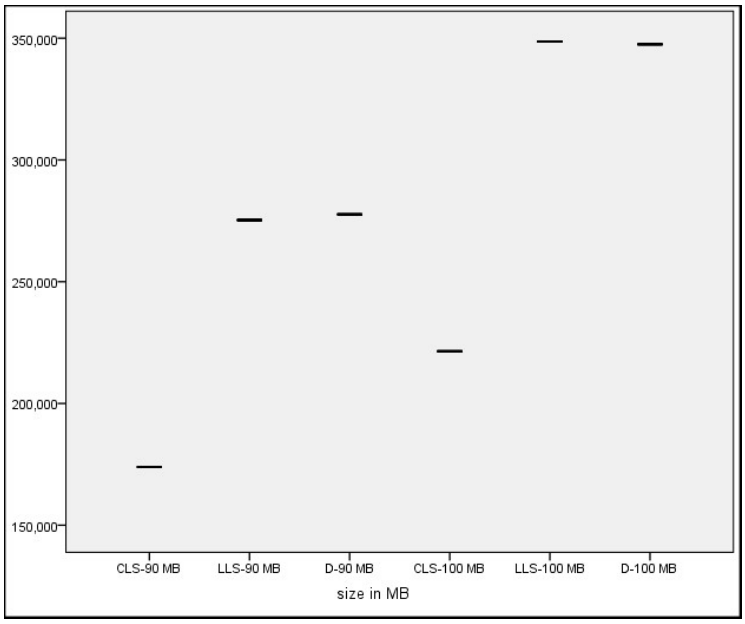


Figure 5-24: Boxplot between CLS, LLS & Dewey schemes using XML files of size 90 MB and 100 MB for labelling XML documents.

p-value @ 100 MB between CLS & LLS schemes =	0.004
p-value @ 100 MB between CLS & Dewey schemes =	0.007

Figure 5-25: the p-value between the CLS, LLS and Dewey schemes using XML files of size 100 MB for labelling XML documents.

5.2.2 Determining Different Relationships:

This experiment computed the calculation time of the relationships between two nodes using the labels between these nodes. Five aspects and relationships were measured in this experiment, namely: sibling, parent/child, ancestor/descendant, order, and level. More details are the following:

a) Sibling measurement: this is used to decide whether two nodes or more share the same parent or not.
b) Parent/child and ancestor: these relationships are calculated to define whether two nodes are parent and child linked.
c) Ancestor/descendant measurement: these relationships are calculated to define whether two nodes are ancestor and descendant linked.
d) Order measurement: this is used to define the order between certain nodes.
e) Level measurement: this one defines the level of a node in an XML data-tree.

Figure 5-26 shows the spent time for determining the relationships of the proposed scheme. It is observed that the spent time to compute the level of nodes is the shortest, whereas the spent time to compute the parent/child is the longest.

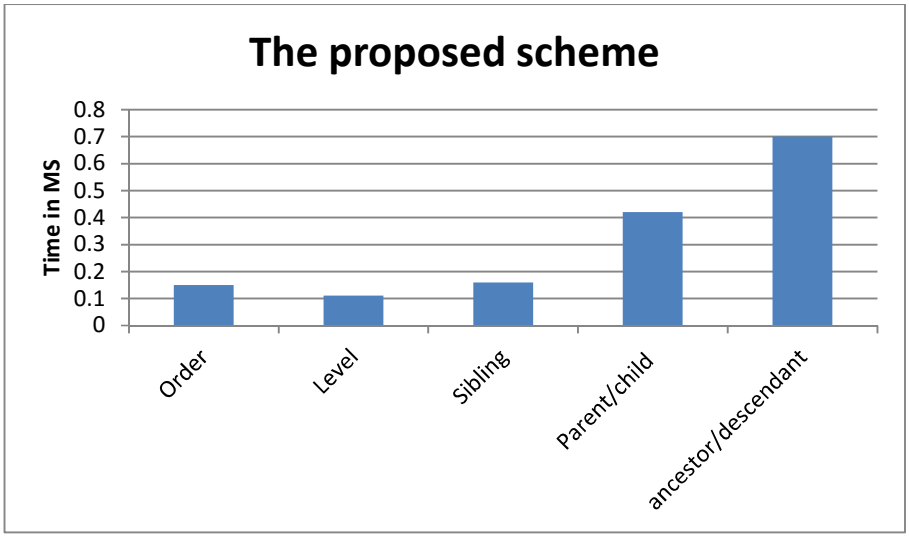


Figure 5-26: spent time for determining the relationships of the proposed scheme.

Figure 5.27 shows the spent time for determining the relationships of the LLS labelling scheme. The spent time to compute the order of the nodes is the shortest, whereas the spent time to compute the ancestor/descendant is the longest.

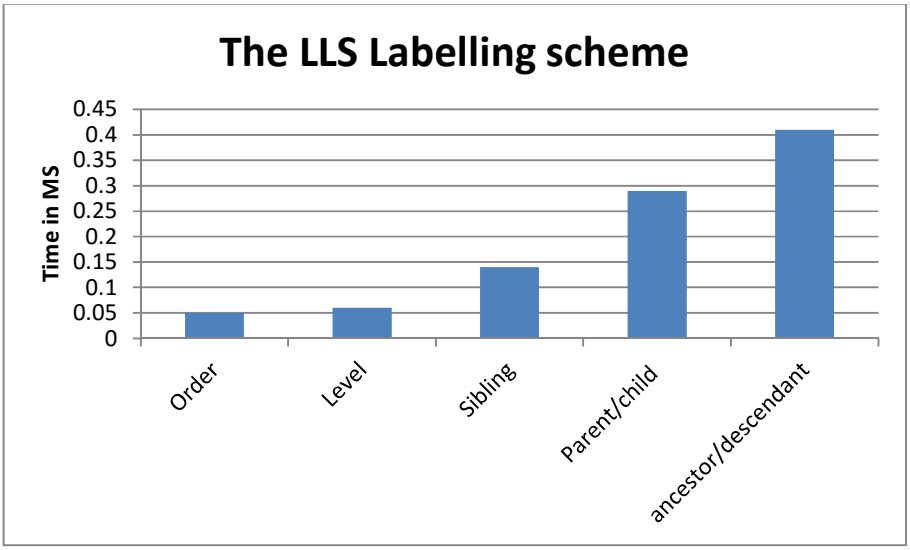


Figure 5-27: spent time for determining the relationships of the LLS scheme.

Figure 5.28 illustrates the spent time for determining the relationships of the Dewey labelling scheme. Again, the spent time to compute the order of the nodes is the shortest, whereas the spent time to compute the ancestor/descendant is the longest.

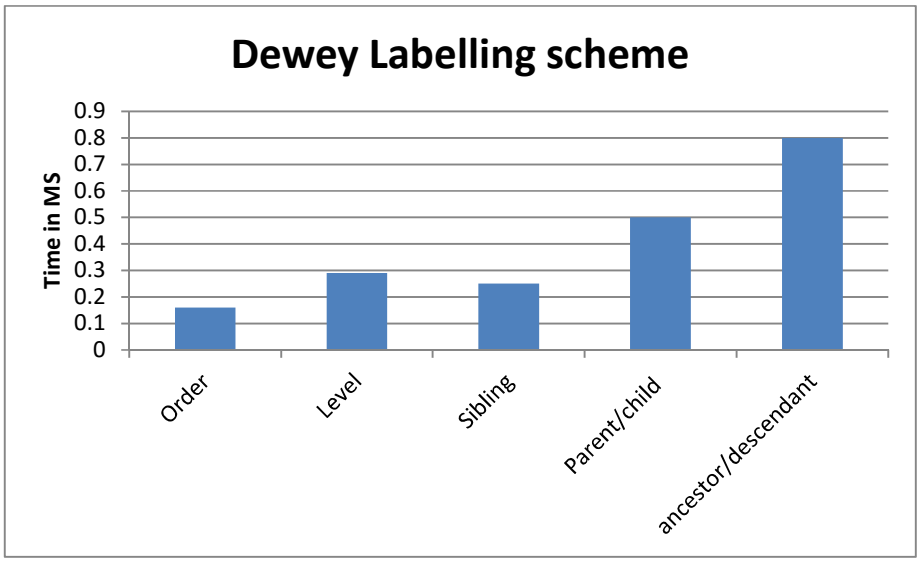


Figure 5-28: spent time for determining the relationships of the Dewey labelling scheme.

A comparison of the results, as in figure 5.29, shows that the proposed labelling scheme achieved better results than the LLS and the Dewey labelling schemes in four relationships, namely, order, level, sibling, and parent/child; whereas the Dewey scheme achieved best result in the ancestor/descendant relationships.

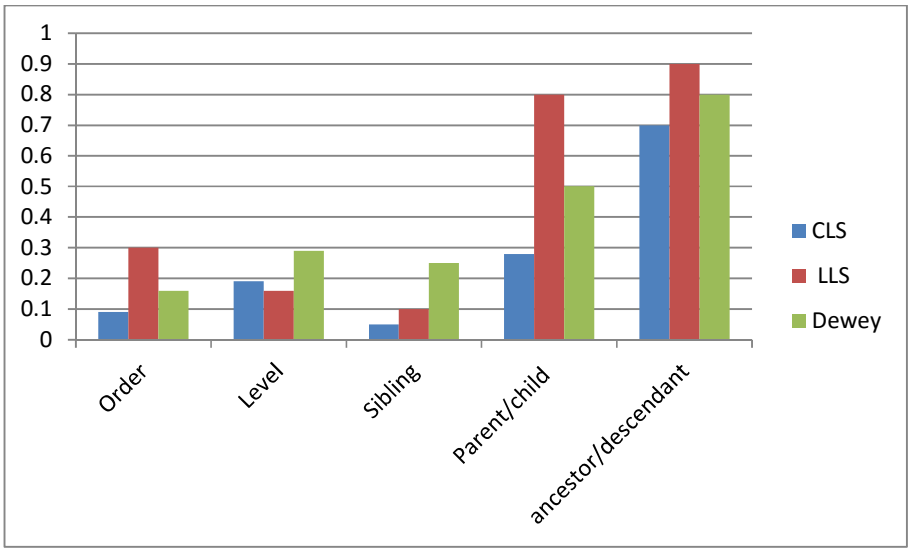


Figure 5-29: the time spent for determining the relationships.

With respect to the proposed scheme, the sibling measurement illustrates the shortest time; whereas the ancestor/descendant relationship is the longest time. The second shortest time is the level calculation which is 0.19 millisecond. The order and the parent/child relationship results are 0.09 and 0.28 respectively. As far as the LLS scheme is concerned, the sibling measurement demonstrates the shortest time

consumption, whereas the parent/child relationship consumed the longest time. Regarding the Dewey scheme, the calculation of nodes' level is the longest time. The parent/child is the shortest time.

Statistical Description of the Results:

The time needed to figure out the relationships between two nodes was calculated. These relationships were discussed in detail in the Methodology chapter. The populations that were examined are, namely, order between nodes, level of nodes, sibling relationships, parent/child relationships, and ancestor/descendent relationships. Having considered the five box plot figures below, the distribution shows a shift to the proposed scheme, which means that this scheme is faster than the others. The p values were calculated by the Wilcoxon rank-sum test, and all the results are far lower than the significance level. Thus, the null hypothesis was rejected and the alternative hypothesis was accepted, and as a result, the proposed scheme is faster than the other schemes.

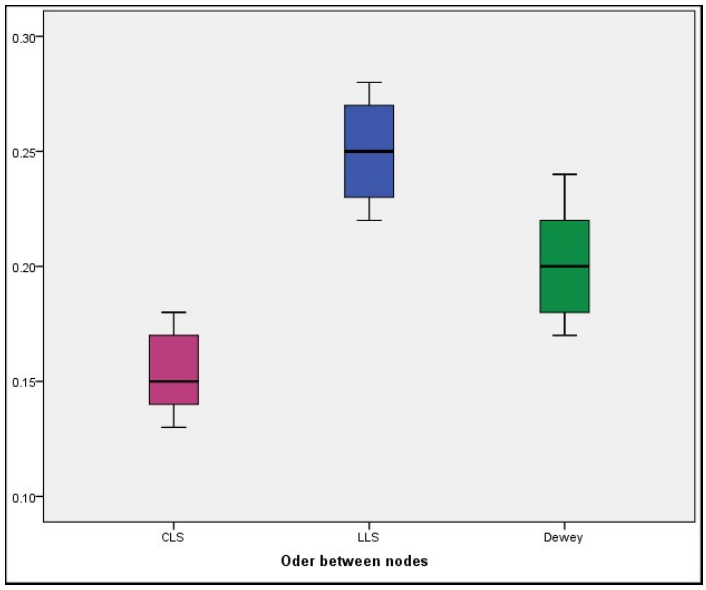


Figure 5-30: Boxplot between CLS, LLS & Dewey schemes when determining the Order between two nodes.

p-value between CLS & LLS schemes =	0.018
p-value between CLS & Dewey schemes =	0.017

Figure 5-31: the p-value between the CLS and & LLS schemes when determining the Order between two nodes.

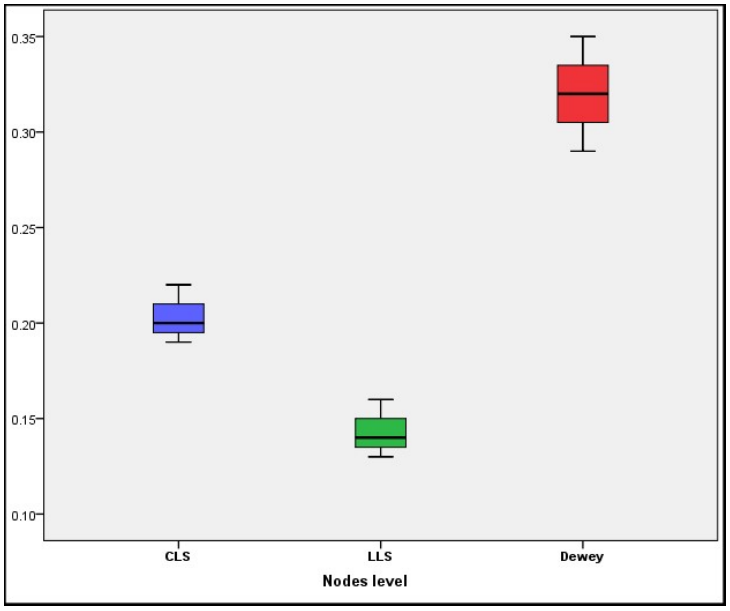


Figure 5-32: Boxplot between CLS, LLS & Dewey schemes when determining the nodes' Level.

p-value between CLS & LLS schemes =	0.017
p-value between CLS & Dewey schemes =	0.008

Figure 5-33: the p-value between the CLS, LLS & Dewey schemes when determining the nodes' Level.

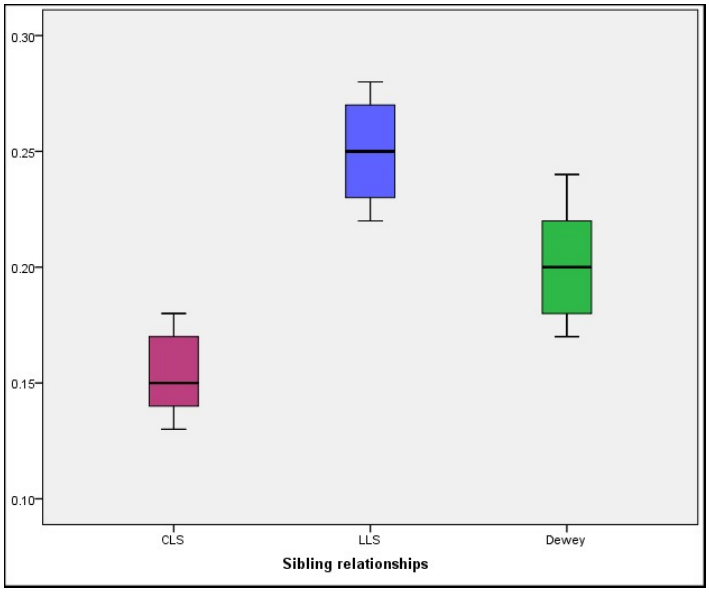


Figure 5-34: Boxplot between CLS & LLS & Dewey schemes when determining the sibling relationships.

p-value between CLS & LLS schemes =	0.008
p-value between CLS & Dewey schemes =	0.017

Figure 5-35: the p-value between the CLS, LLS and Dewey schemes when determining the sibling relationships.

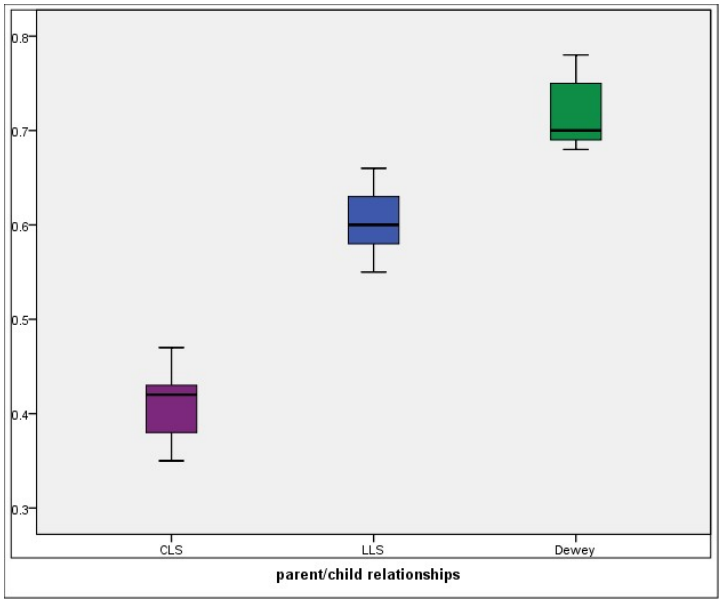


Figure 5-36: Boxplot between CLS, LLS & Dewey schemes when determining the Parent/Child relationships.

p-value between CLS & LLS schemes =	0.017
p-value between CLS & Dewey schemes =	0.016

Figure 5-37: the p-value between the CLS, LLS and Dewey schemes when determining the Parent/Child relationships.

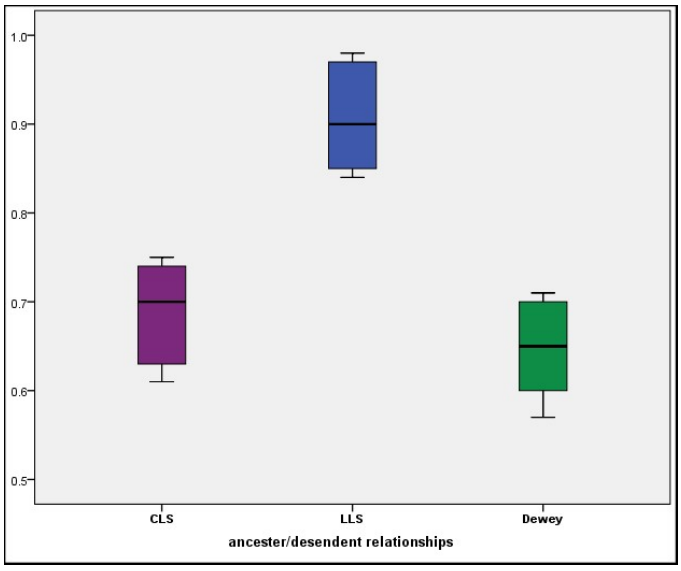


Figure 5-38: Boxplot between CLS, LLS & Dewey schemes when determining the Ancestor/Descendent relationships.

p-value between CLS & LLS schemes =	0.017
p-value between CLS & Dewey schemes =	0.018

Figure 5-39: the p-value between the CLS, LLS and Dewey schemes when determining the ancestor/descendant relationships.

5.2.3 Query Efficiency Measurement:

This experiment was carried out to measure query efficiency and performance before and after insertions. As stated in the Methodology chapter, the XMark queries were chosen to test the proposed scheme against the LLS and Dewy schemes. These queries are illustrated in table 4-1 in the Methodology chapter.

As table 4-2 in the Methodology chapter shows, different sizes of files were created to test and measure different aspects. The number of these files is eleven. The sizes of these files are between 1 MB and 100MB size. The *XMark-1* file was chosen to evaluate all queries. This file is 1 MB size which is the smallest one. The XMark-1 file was selected because it facilitates monitoring and observing the changes easily when updates of the document take place. The other files started from 10 MB and reached 100 BM. The sizes of these files differed by 10 MB. Table 5.1 shows the results for testing the proposed scheme and the other two labelling schemes using the *XMark-1* file.

TABLE 5-1: THE RESULTS OF THE QUERIES FOR TESTING ALL SCHEMES

No	***Query number***	Purpose	Proposed scheme	LLS scheme	Dewey scheme
1	Q1	Exact match	45	57	70
2	Q2	Ordered access	146	252	261
3	Q3		229	272	288
4	Q4		121	163	161
5	Q5	Casting	37	43	50
6	Q6	Regular path expression	52	59	78
7	Q7		16	14	22
8	Q8	Chasing references	2174	3273	3421
9	Q9		173	213	198
10	Q11	Joins on values	256	317	288
11	Q12		207	348	374
12	Q13	Reconstruct portions of the original XML document	1875	2239	2381
13	Q14	Full text	45	61	58
14	Q15	Path traversals	92	136	157
15	Q16		85	108	149
16	Q17	Finding missing elements	126	73	151
17	Q18	Function application	67	83	80
18	Q19	Sorting	437	148	621
19	Q20	Aggregation	152	206	264

Figures 5.40, 5.41, and 5.42 show the results of query performance for the three targeted schemes.

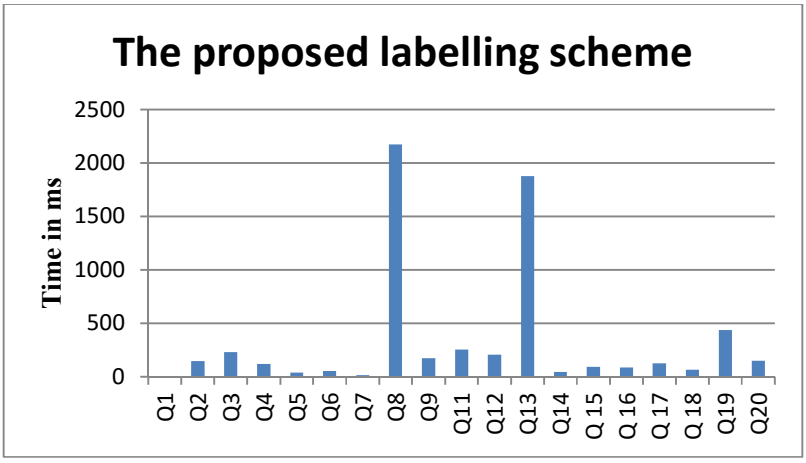


Figure 5-40: query performance for the proposed scheme.

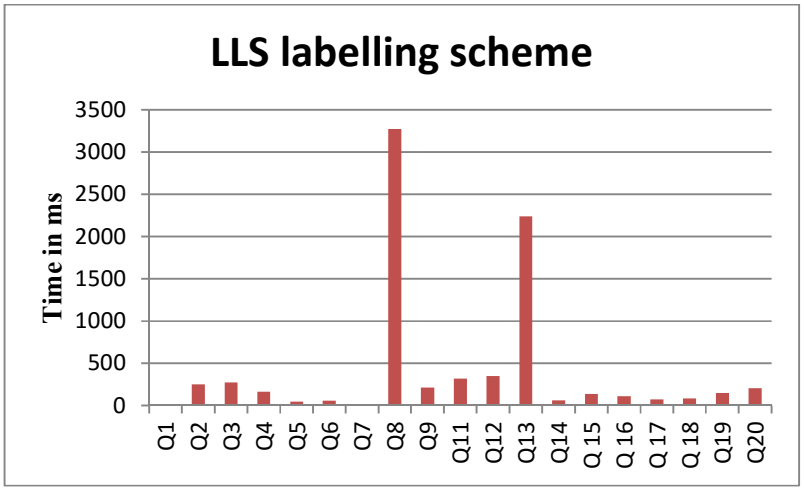


Figure 5-41: query performance for the LLS scheme.

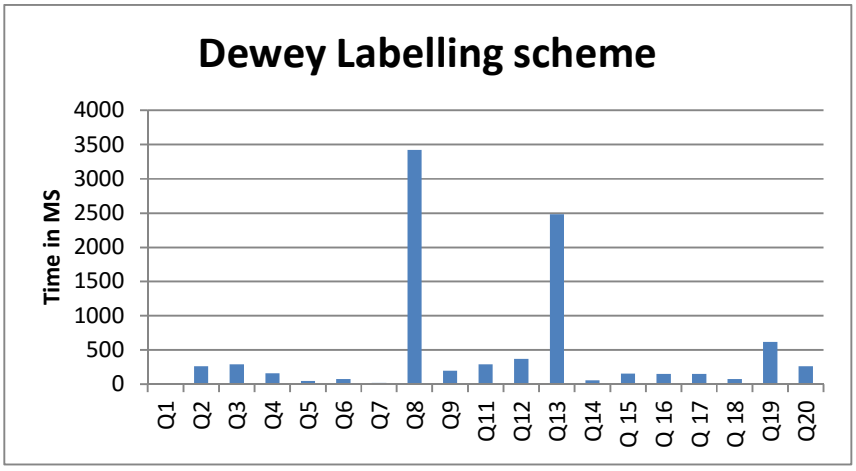


Figure 5-42: query performance for the Dewey labelling scheme.

As seen in table 5.1 and the figures above, the results vary among each other. In order to clarify and discuss these results in more detail, they were divided into four groups based on the spent time. Table 5.2 shows these groups.

TABLE 5-2: GROUPS OF THE QUERIES BASED ON THEIR SPENT TIME.

no	***Group name***	***Time range***	***Queries***
1	G1	From 0 till 100 ms	Q1, Q5, Q6, Q7, and Q14.
2	G2	From 101 till 206 ms	Q16, Q15, Q18, Q4, Q17 and Q20
3	G3	From 207 till 500 ms	Q9, Q12, Q3, Q2, Q11 and Q19
4	G4	From 501 till 3500 ms	Q8 and Q13

Regarding the proposed scheme results, as seen in table 5.1, query (7) achieved the shortest spent time which is 16 milliseconds. On the other hand, query (8) achieved the longest spent time at 2174 milliseconds. This query is used to test the database ability to reconstruct parts of the original XML document.

The results for all schemes that are based on the query purpose are as follows: The Exact match (Q1) spent quite a short time which between 45 and 70 milliseconds. The order access queries (Q2, Q3, and Q4) deal with large amounts of data and therefore quite a long time was spent on these queries – between 121 and 288 milliseconds. Similarly, between 173 and 3421 milliseconds were spent on the chasing references queries (Q8 and Q9). Moreover, a long time was also spent on the joins on values queries (Q11 and Q12) – between 207 and 374 milliseconds. Time spent on the reconstructed portions of the original XML document query (Q13) was between 1875 and 2381 milliseconds. Finally, for the rest of the queries, time spent was: casting – between 37 and 50 milliseconds; regular path expression – between 14 and 78 milliseconds; full text – between 45 and 61 milliseconds; path traversals – between 85 and 157 milliseconds; finding missing elements – between 73 and 151 milliseconds; function application – between 67 and 83 milliseconds; and aggregation queries – between 152 and 264 milliseconds. Figures from 5.43 until 5.46 show the spent times using the proposed labelling scheme based on the groups of queries.

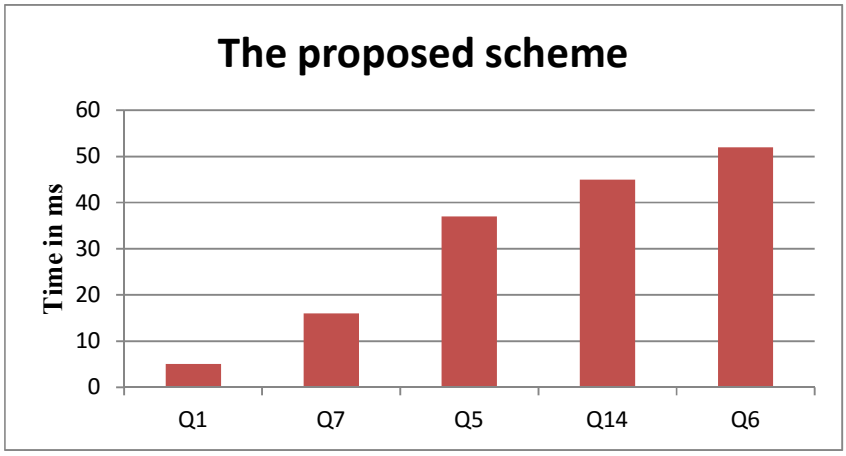


Figure 5-43: query performance of the proposed scheme for group G1.

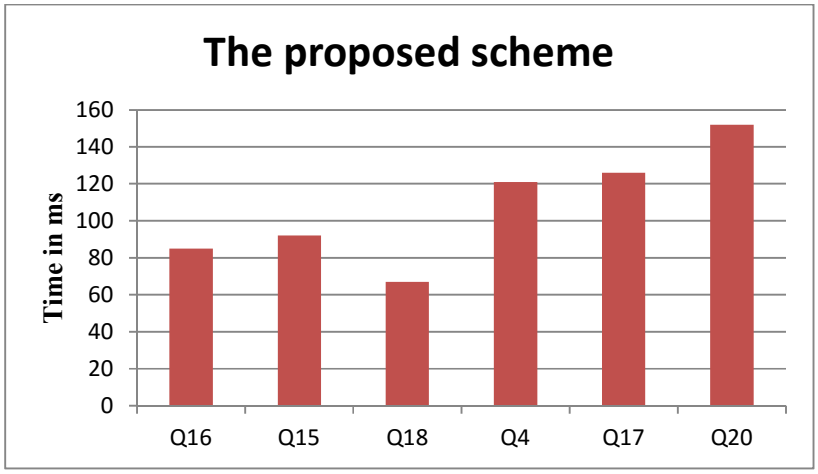


Figure 5-44: query performance of the proposed scheme for group G2.

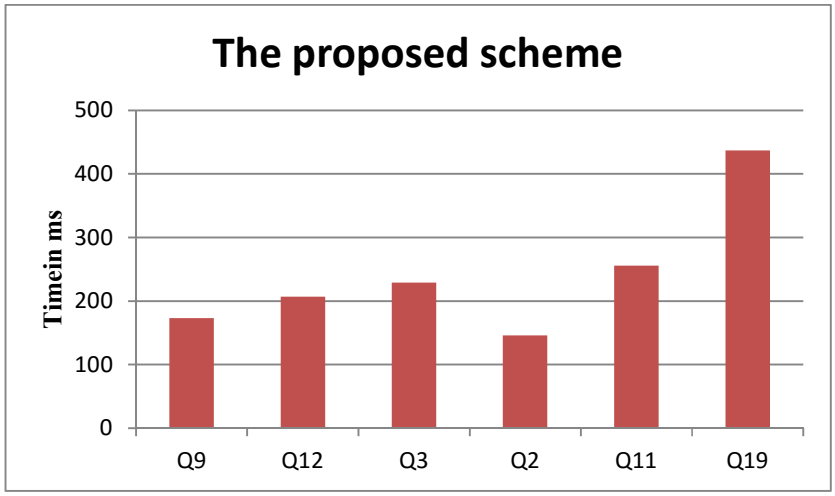


Figure 5-45: query performance of the proposed scheme for group G3.

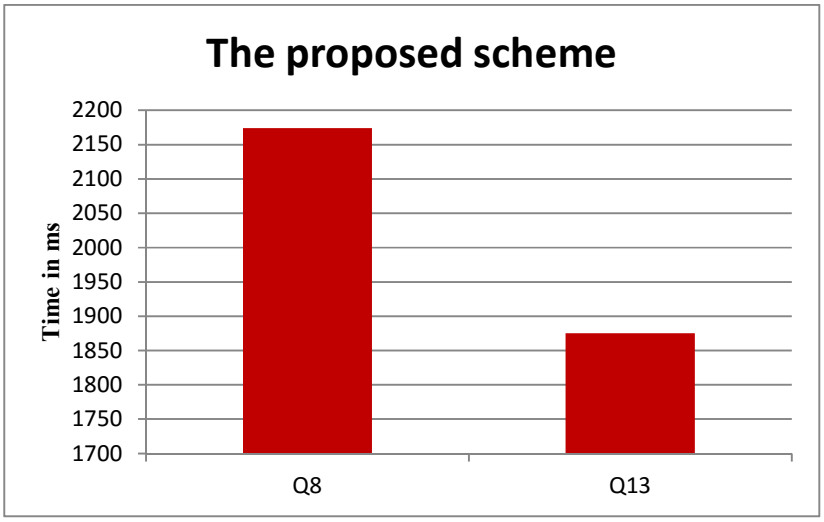


Figure 5-46: query performance of the proposed scheme for group G4.

Figures from 5.47 until 5.50 show the spent times using the LLS labelling scheme based on the groups of queries.

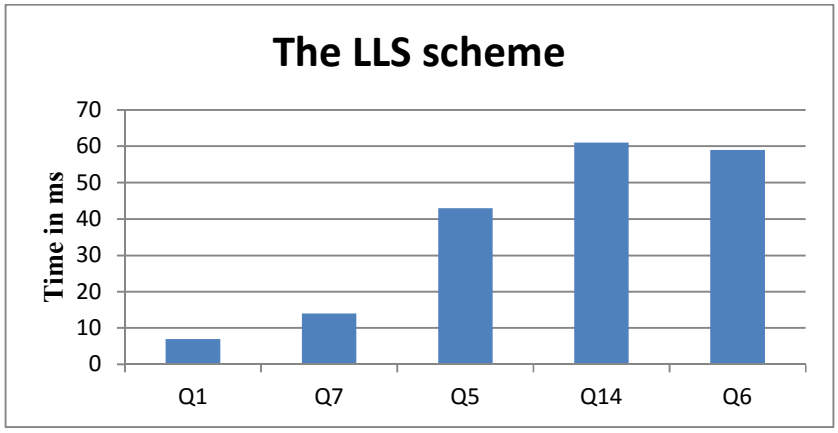


Figure 5-47: query performance of the LLS scheme for group G1.

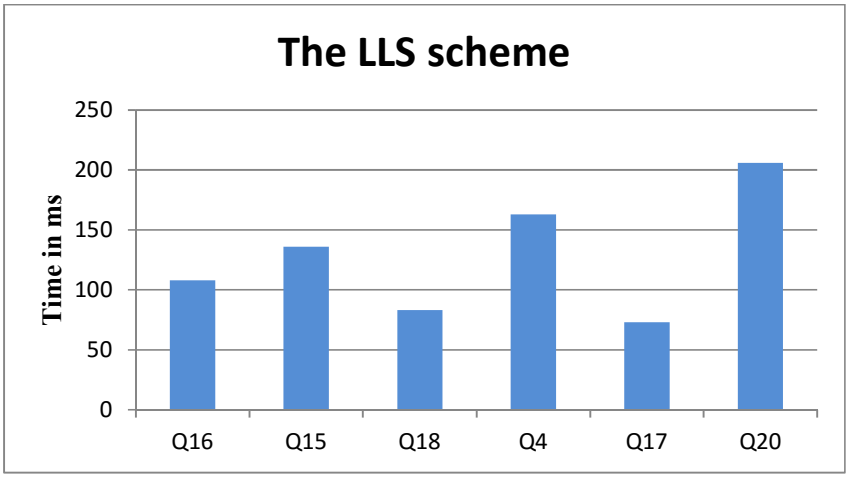


Figure 5-48: query performance of the LLS scheme for group G2.

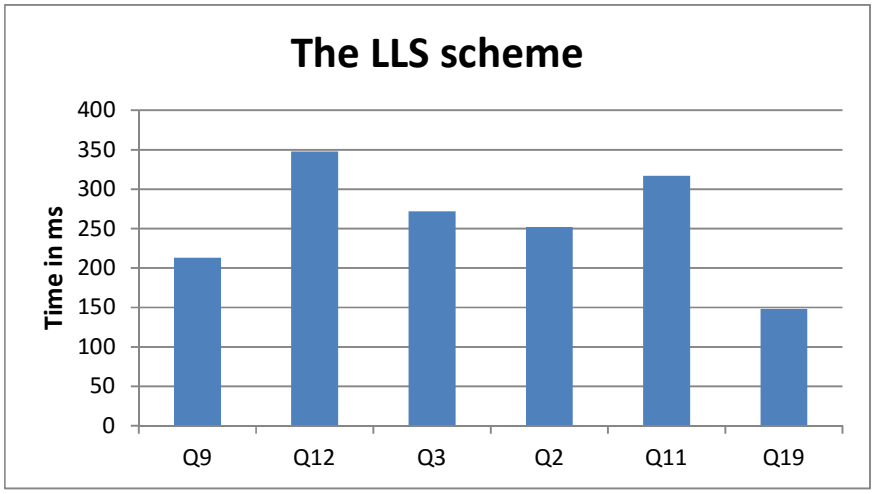


Figure 5-49: query performance of the LLS scheme for group G3.

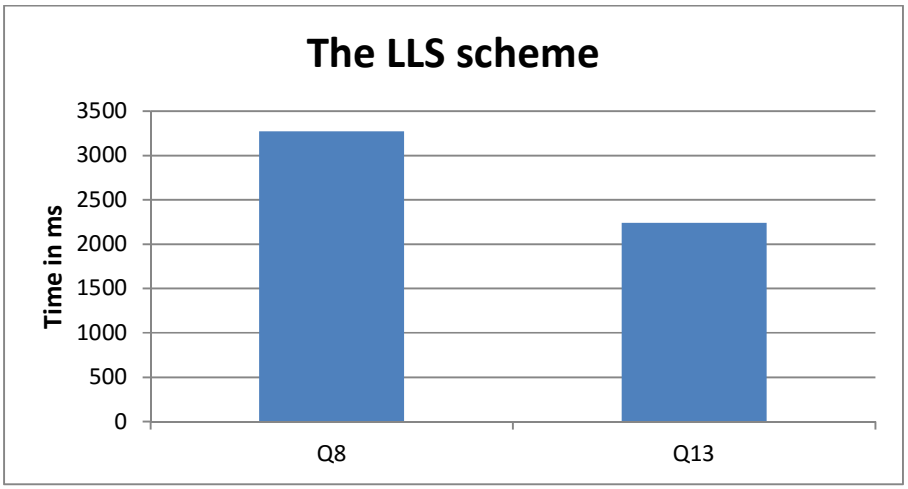


Figure 5-50: query performance of the LLS scheme for group G4.

Query Q8 is the longest spent time with 3273 milliseconds. The second longest spent time is Q13 at 2239 milliseconds. The shortest one is query 7 with 14 milliseconds. So, with regard to the longest and shortest spent time, the proposed scheme and the LLS are the same.

Figures from 5.51 until 5.54 show the spent times using the LLS labelling scheme based on the groups of queries.

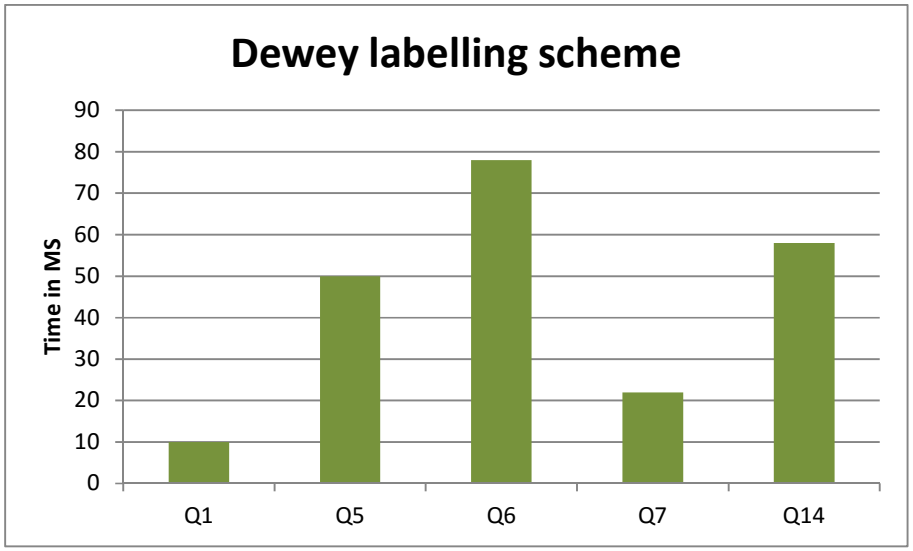


Figure 5-51: query performance of the Dewey labelling scheme for group G1.

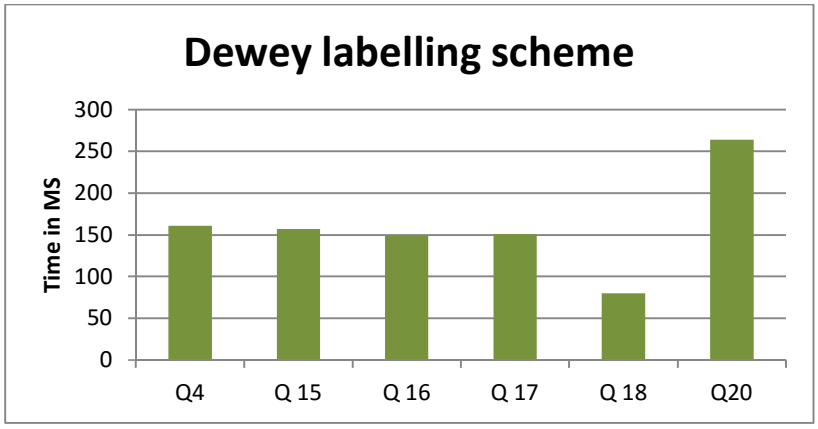


Figure 5-52: query performance of the Dewey labelling scheme for group G2.

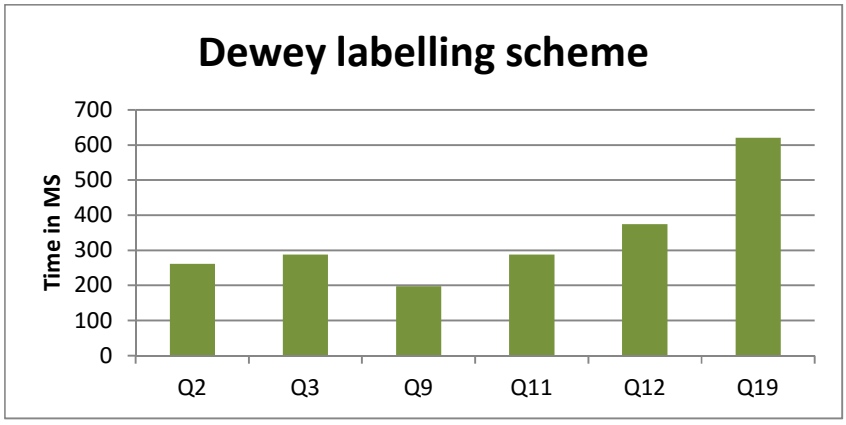


Figure 5-53: query performance of the Dewey labelling scheme for group G3.

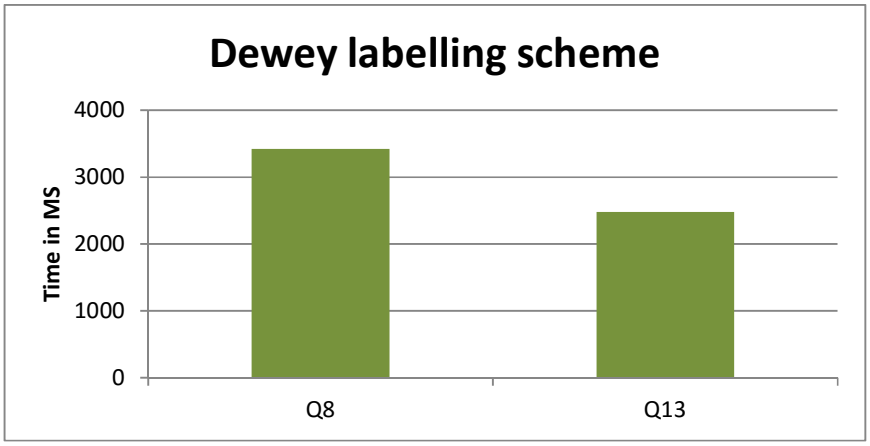


Figure 5-54: query performance of the Dewey labelling scheme for group G4.

- **Comparing the Results:**

The proposed scheme achieved the best results in sixteen queries. It failed in queries number 7, 17, and 19: query number (7) which is *Regular Path Expression* query with time spent 16 milliseconds; query number (17) which is *Finding missing elements* with time spent 126 milliseconds; query number (19) which is *Sorting* query with time spent 437 milliseconds. The LLS scheme achieved the best results in these queries as follows: time spent for query 7 is 14 milliseconds; time spent for query 17 is 73 milliseconds; time spent for query 19 is 148 milliseconds. Figure 5.55 shows all these results for all schemes. Moreover, figures from 5.56 to 5.59 illustrate these results based on designated groups in table 5.2.

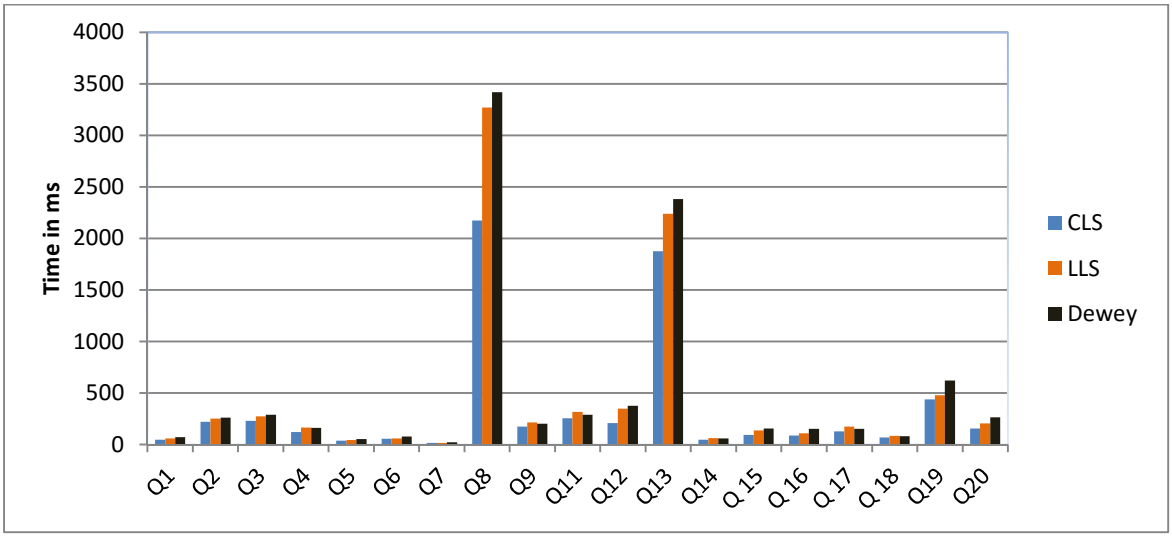


Figure 5-55: query performance of all schemes.

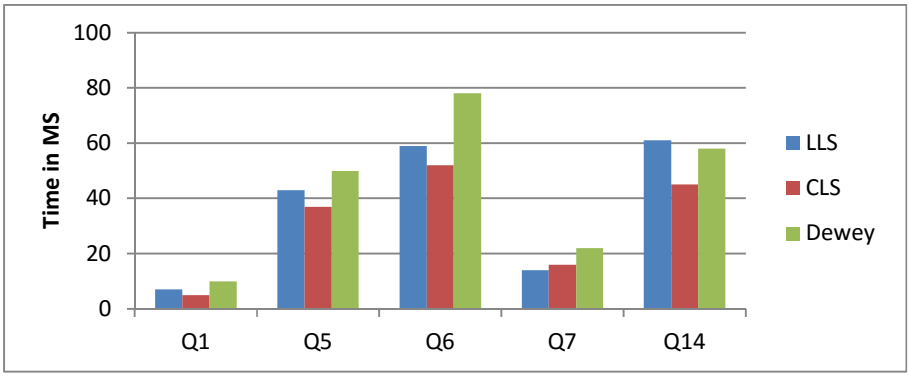


Figure 5-56: query performance of all schemes for group G1.

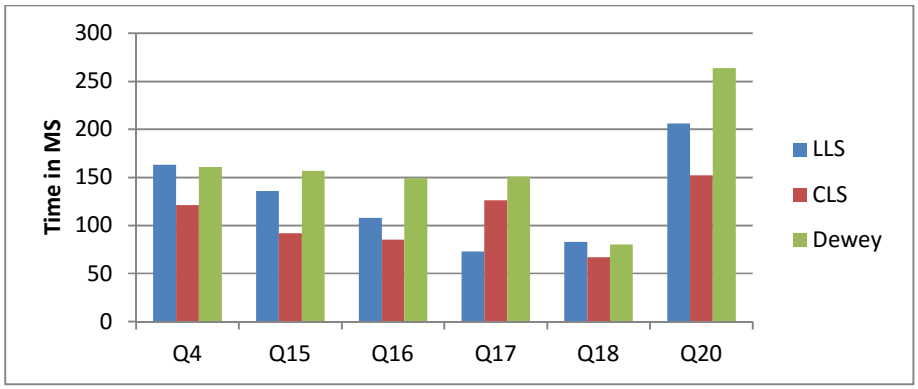


Figure 5-57: query performance of all schemes for group G2.

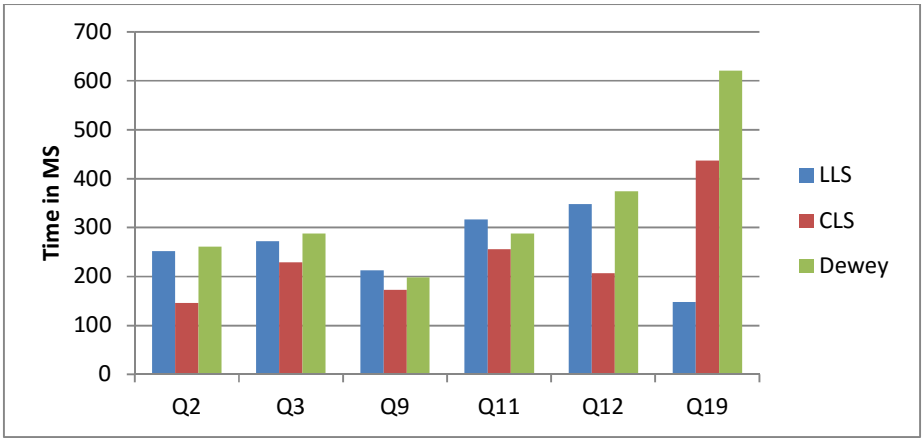


Figure 5-58: query performance of all schemes for group G3.

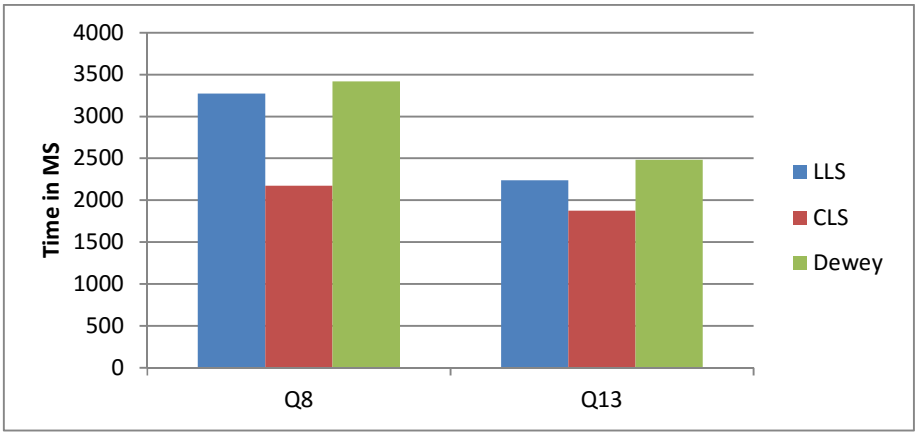


Figure 5-59: query performance of all schemes for group G4.

- **Statistical Description of the Results:**

As discussed in the Methodology chapter, nineteen queries were provided by the XMark Benchmark to test different aspects of XML data. A box plot for each of these queries was created as shown below. All these box plots, except the box plot for query 7, show that the proposed labelling scheme delivered the best performance. In query 7 the LLS scheme shows better performance than the proposed scheme. With regard to the p-values, all p values between the proposed scheme and others scheme for each query were calculated. Thus, 36 p values were calculated for 19 queries. All the results are far lower than the significance level. Thus, the null hypothesis was rejected and the alternative hypothesis was accepted. Therefore, the proposed scheme has better results in 16 out of 19 queries.

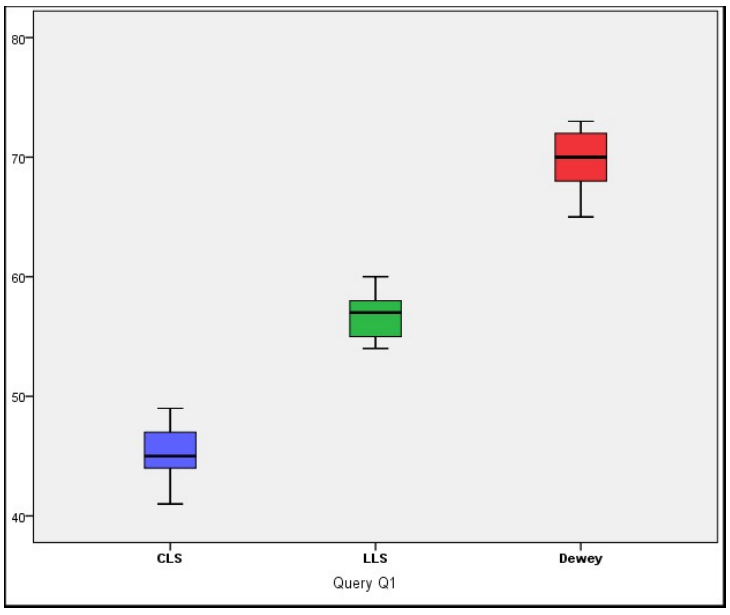


Figure 5-60: Boxplot between CLS, LLS & Dewey schemes when evaluating query 1.

p-value between CLS & LLS schemes =	0.006
p-value between CLS & Dewey schemes =	0.005

Figure 5-61: the p-value between the CLS, LLS and Dewey schemes when evaluating query 1.

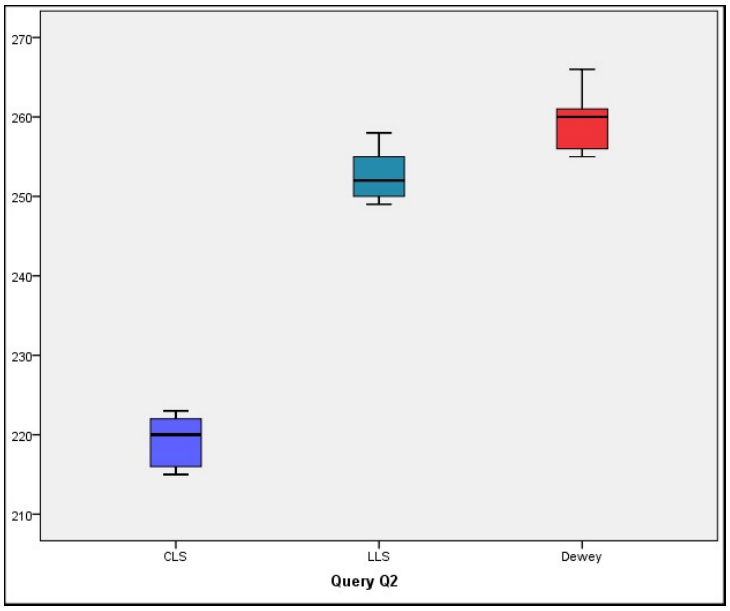


Figure 5-62: Boxplot between CLS, LLS & Dewey schemes when evaluating query 2.

p-value between CLS & LLS schemes =	0.042
p-value between CLS & Dewey schemes =	0.042

Figure 5-63: the p-value between the CLS, LLS and Dewey schemes when evaluating query 2.

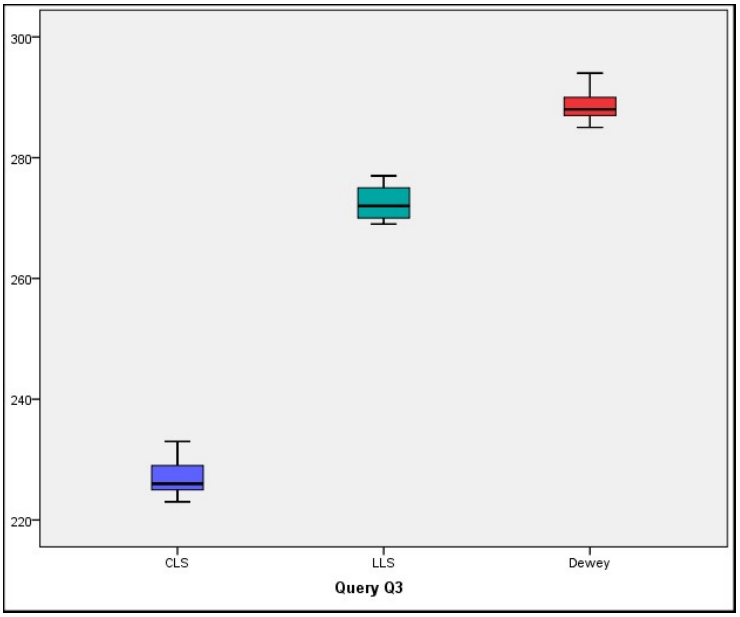


Figure 5-64: Boxplot between CLS, LLS & Dewey schemes when evaluating query 3.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.043

Figure 5-65: the p-value between the CLS, LLS and Dewey schemes when evaluating query 3.

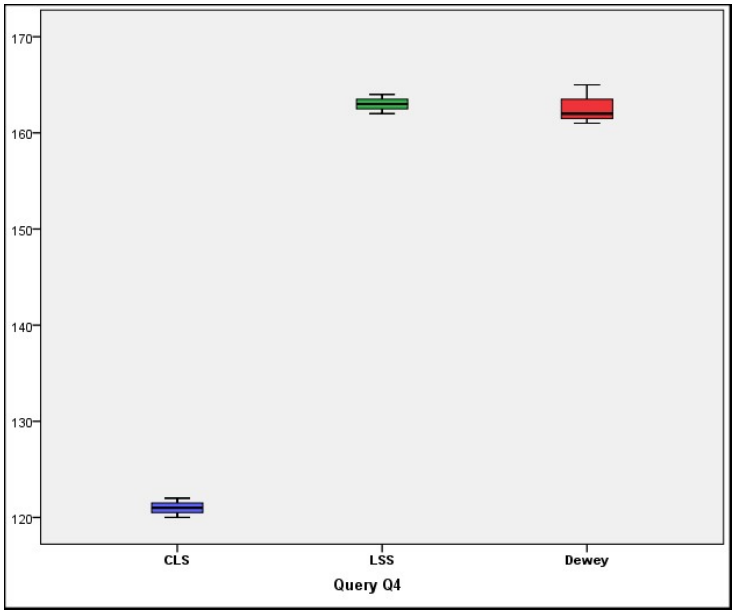


Figure 5-66: Boxplot between CLS, LLS & Dewey schemes when evaluating query 4.

p-value between CLS & LLS schemes =	0.040
p-value between CLS & Dewey schemes =	0.042

Figure 5-67: the p-value between the CLS, LLS and Dewey schemes when evaluating query 4.

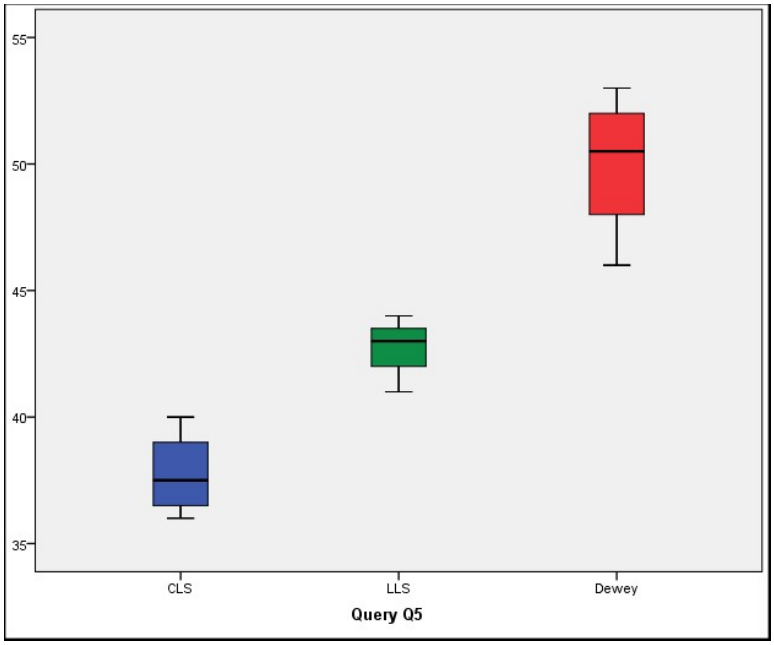


Figure 5-68: Boxplot between CLS, LLS & Dewey schemes when evaluating query 5.

p-value between CLS & LLS schemes =	0.068
p-value between CLS & Dewey schemes =	0.068

Figure 5-69: the p-value between the CLS, LLS and Dewey schemes when evaluating query 5.

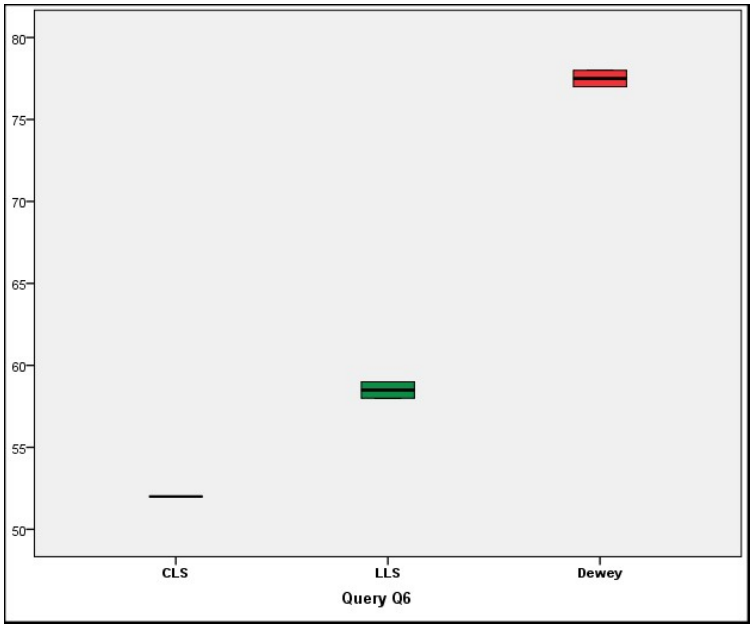


Figure 5-70: Boxplot between CLS, LLS & Dewey schemes when evaluating query 6.

p-value between CLS & LLS schemes =	0.037
p-value between CLS & Dewey schemes =	0.043

Figure 5-71: the p-value between the CLS, LLS and Dewey schemes when evaluating query 6.

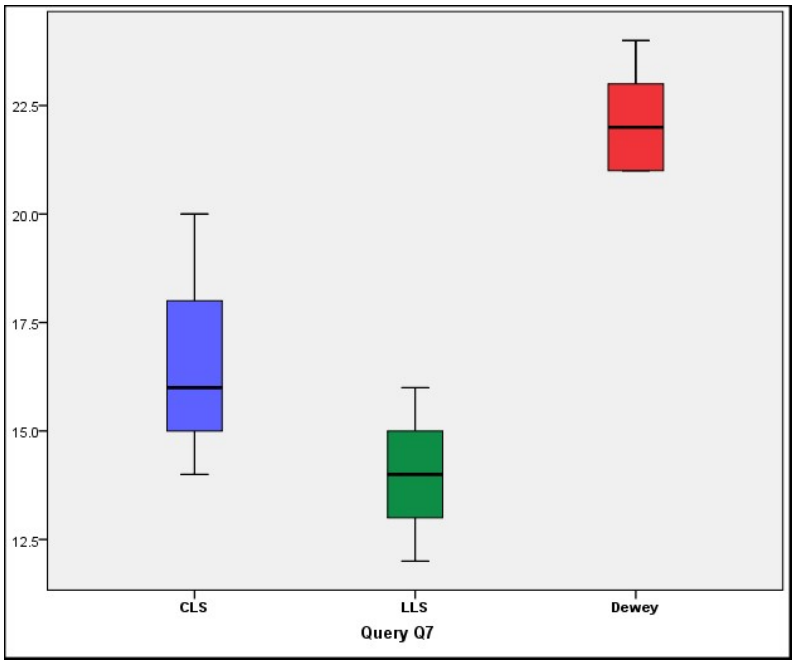


Figure 5-72: Boxplot between CLS, LLS & Dewey schemes when evaluating query 7.

p-value between CLS & LLS schemes =	0.042
p-value between CLS & Dewey schemes =	0.041

Figure 5-73: the p-value between the CLS, LLS and Dewey when evaluating schemes 7.

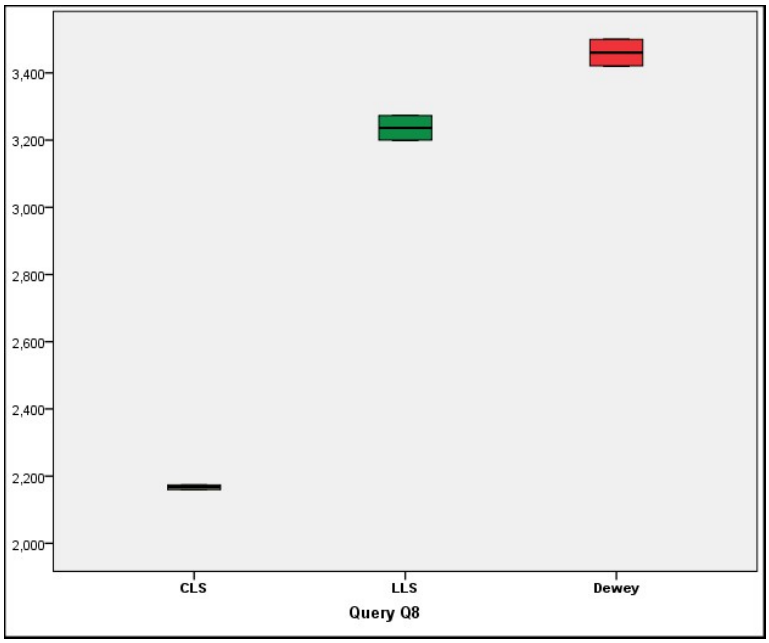


Figure 5-74: Boxplot between CLS, LLS & Dewey schemes when evaluating query 8.

p-value between CLS & LLS schemes =	0.044
p-value between CLS & Dewey schemes =	0.046

Figure 5-75: the p-value between the CLS LLS and Dewey schemes when evaluating query 8.

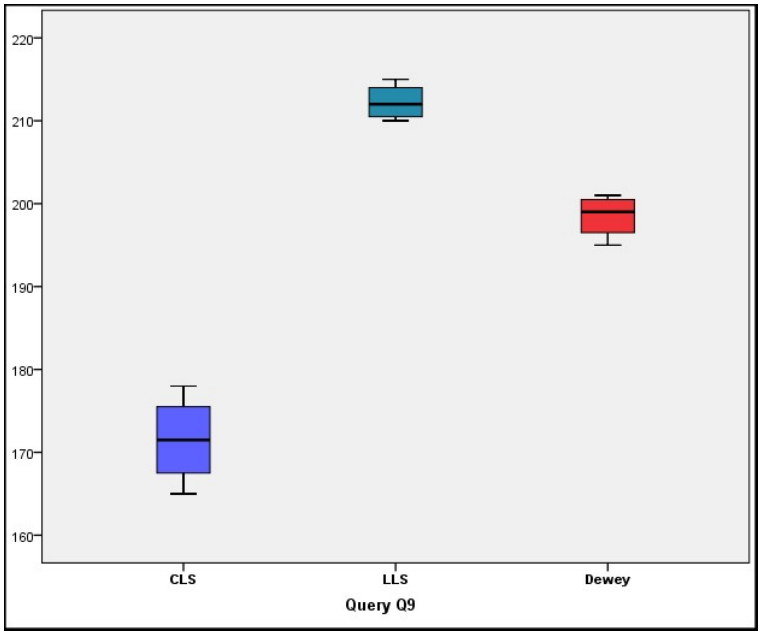


Figure 5-76: Boxplot between CLS & LLS & Dewey schemes when evaluating query 9.

p-value between CLS & LLS schemes =	0.038
p-value between CLS & Dewey schemes =	0.043

Figure 5-77: the p-value between the CLS, LLS and Dewey schemes when evaluating query 9.

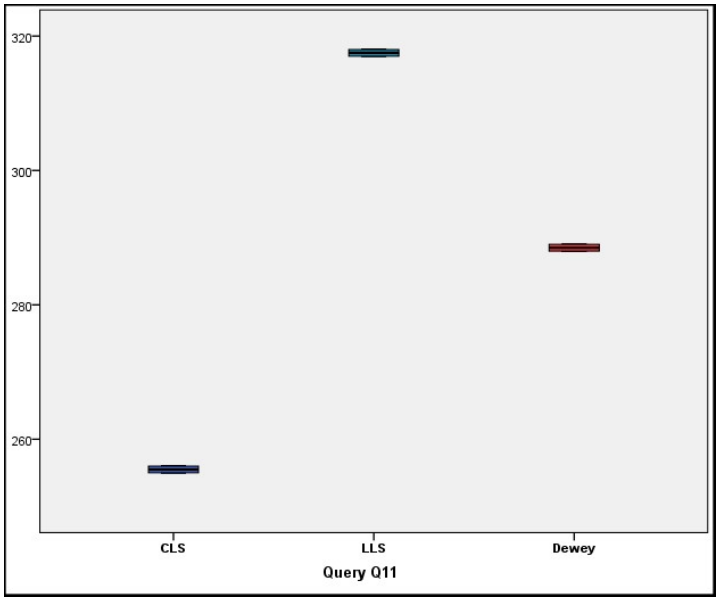


Figure 5-78: Boxplot between CLS, LLS & Dewey schemes when evaluating query 11.

p-value between CLS & LLS schemes =	0.045
p-value between CLS & Dewey schemes =	0.032

Figure 5-79: the p-value between the CLS, LLS and Dewey schemes when evaluating query 11.

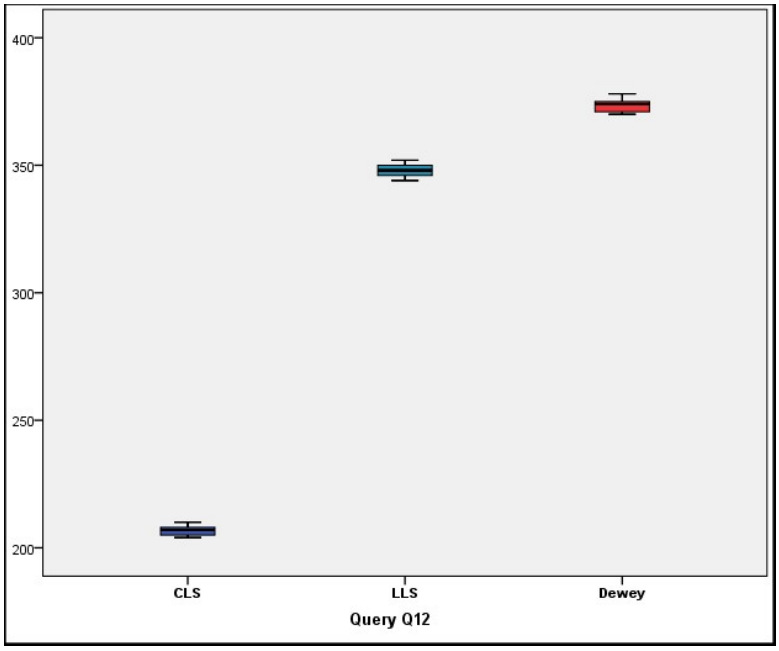


Figure 5-80: Boxplot between CLS, LLS & Dewey schemes when evaluating query 12.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.043

Figure 5-81: the p-value between the CLS, LLS and Dewey schemes when evaluating query 12.

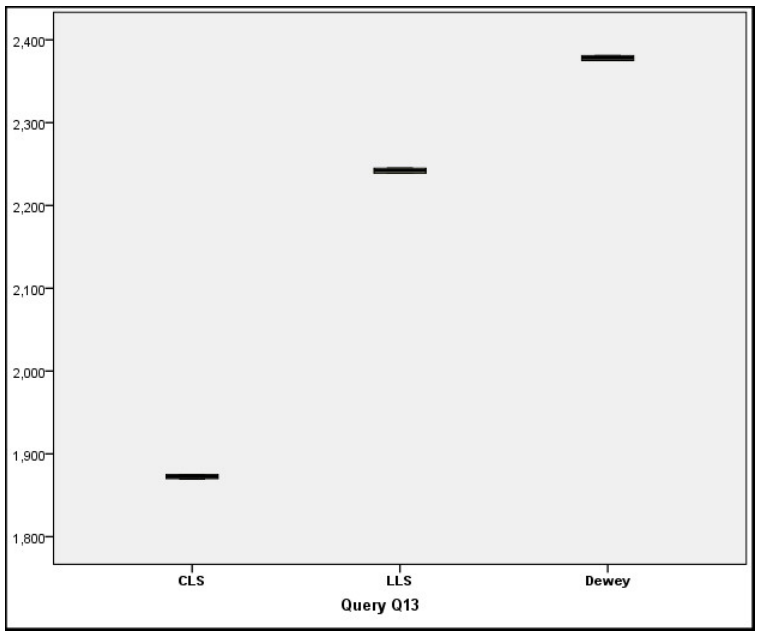


Figure 5-82: Boxplot between CLS & LLS & Dewey schemes when evaluating query 13.

p-value between CLS & LLS schemes =	0.044
p-value between CLS & Dewey schemes =	0.046

Figure 5-83: the p-value between the CLS, LLS and Dewey schemes when evaluating query 13.

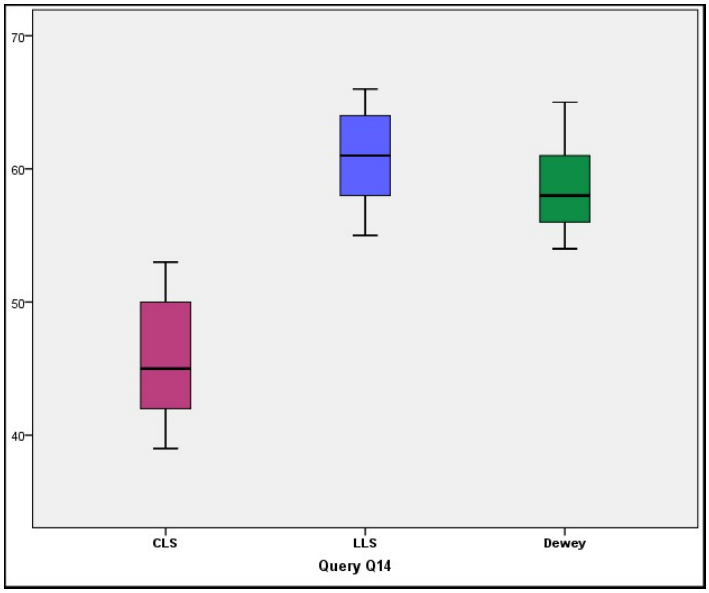


Figure 5-84: shows Boxplot between CLS & LLS & Dewey schemes when evaluating query 14.

p-value between CLS & LLS schemes =	0.045
p-value between CLS & Dewey schemes =	0.041

Figure 5-85: the p-value between the CLS, LLS and Dewey schemes when evaluating query 14.

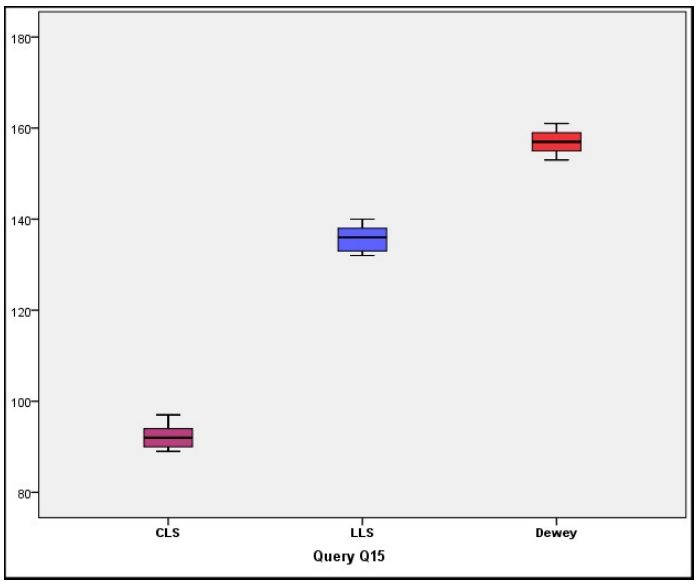


Figure 5-86: Boxplot between CLS, LLS & Dewey schemes when evaluating query 15.

p-value between CLS & LLS schemes =	0.038
p-value between CLS & Dewey schemes =	0.035

Figure 5-87: the p-value between the CLS, LLS and Dewey schemes when evaluating query 15.

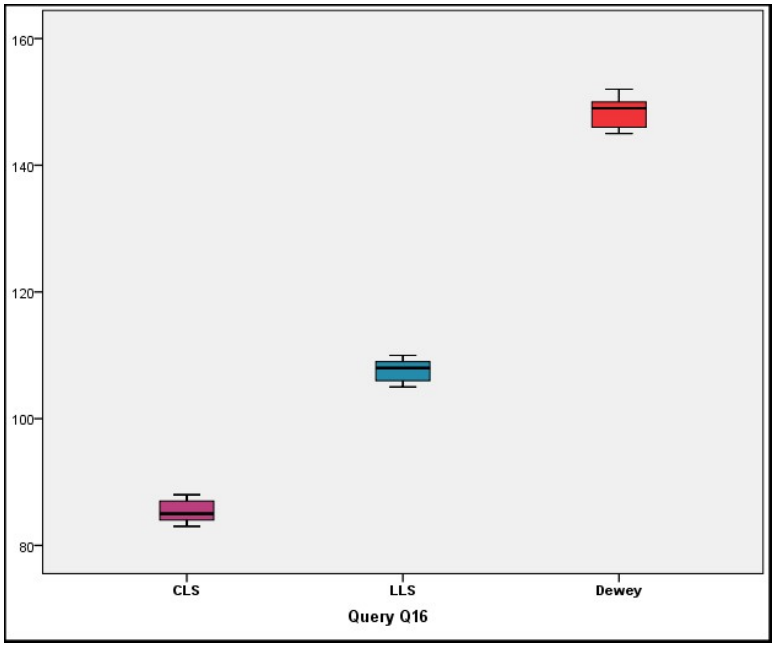


Figure 5-88: Boxplot between CLS, LLS & Dewey schemes when evaluating query 16.

p-value between CLS & LLS schemes =	0.034
p-value between CLS & Dewey schemes =	0.041

Figure 5-89: the p-value between the CLS, LLS and Dewey when evaluating query 16.

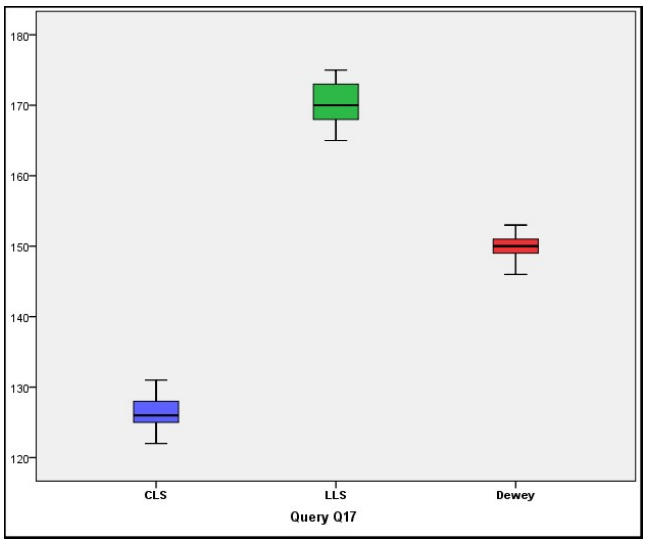


Figure 5-90: Boxplot between CLS, LLS & Dewey schemes when evaluating query 17.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-91: the p-value between the CLS, LLS and Dewey schemes when evaluating query 17.

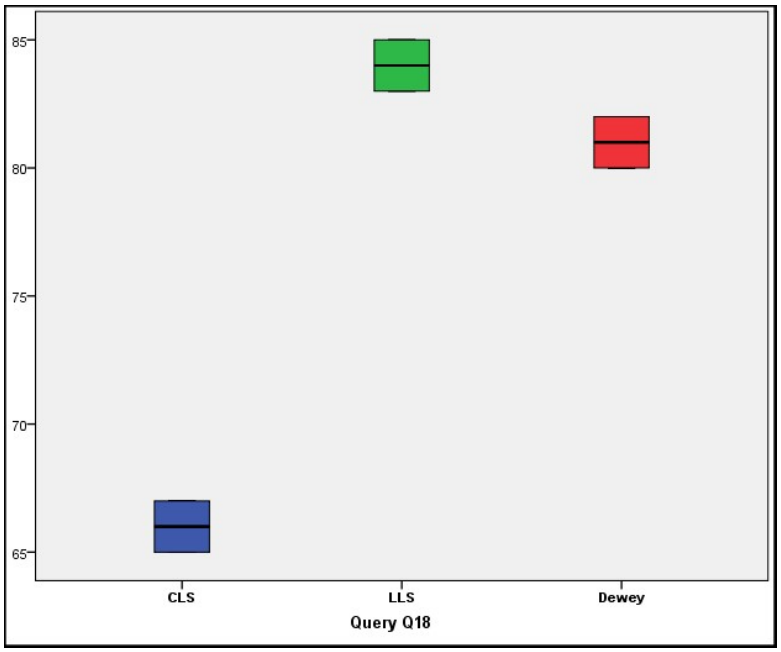


Figure 5-92: Boxplot between CLS, LLS and Dewey schemes when evaluating query 18.

p-value between CLS & LLS schemes =	0.047
p-value between CLS & Dewey schemes =	0.044

Figure 5-93: the p-value between the CLS, LLS and Dewey schemes when evaluating query 18.

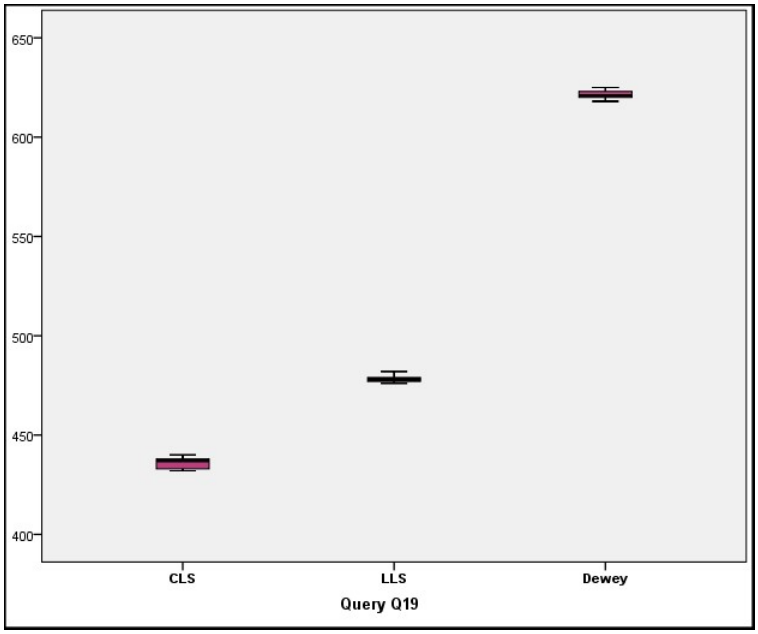


Figure 5-94: Boxplot between CLS & LLS & Dewey schemes when evaluating query 19.

p-value between CLS & LLS schemes =	0.041
p-value between CLS & Dewey schemes =	0.042

Figure 5-95: the p-value between the CLS and & LLS schemes when evaluating query 19.

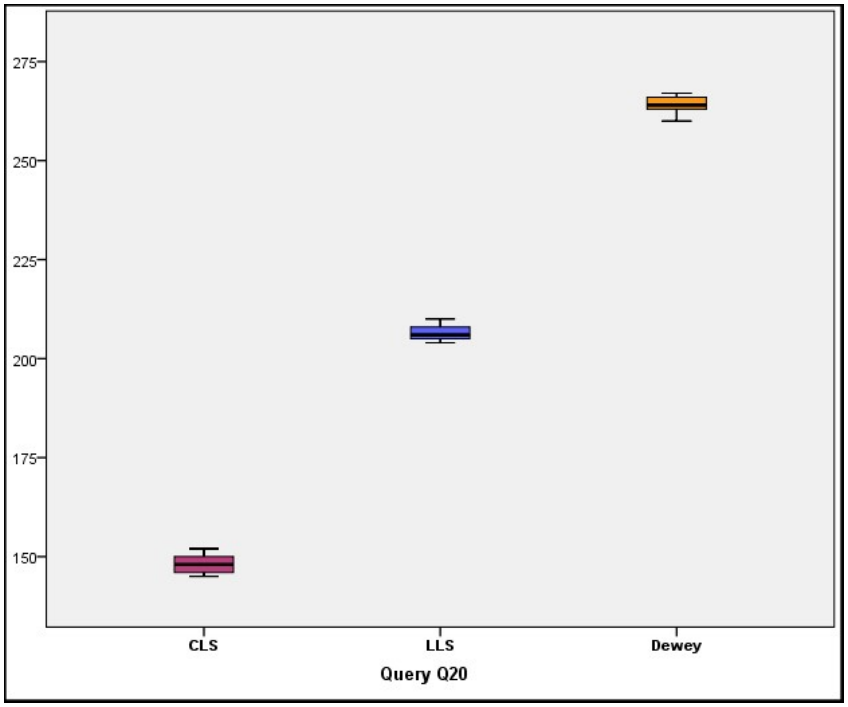


Figure 5-96: Boxplot between CLS, LLS & Dewey schemes when evaluating query 20.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-97: the p-value between the CLS, LLS and Dewey schemes when evaluating query 20.

5.3 Experiments for Dynamic Documents:

Since developing a labelling scheme that handles the dynamic documents is one of the goals of the proposed scheme, these experiments were intended to measure any updates on the dynamic XML documents. The experiments were executed to assess the proposed scheme's ability to deal with different kinds of insertion. Thus, the experiments were divided into three types of groups as follows:

- Experiments for inserting new nodes.
- Experiments for determining different relationships.
- Experiments for query performance.

5.3.1 Inserting New Nodes:

Labelling schemes should be able to handle diversity of insertions as a significant aspect (D. K. Fisher et al., 2006). Thus, different types of insertions need to be executed and tested. Uniform is an important type of insertion that needs to be performed. The time spent and size of the labels are measured in this insertion (Alstrup & Rauhe, 2002). Ordered skewed is another type of insertion. It is used to insert new nodes before and after a specific node times (Edith et al., 2010). Random skewed is another kind of insertion. New nodes are inserted between other nodes randomly. This type of insertion tests the flexibility of the scheme to deal with random insertions as some schemes have difficulty in doing so (C. Li, Ling, & Hu, 2006).

5.3.1.1 Uniform Insertions

The method of this insertion is as follows; a new node is inserted between two sequential nodes and then a new cluster is created when required. Subsequently, the spent time and the sizes of the labels are measured before and after the inserting process. Consequently, all eleven generated *XMark* files were used to test this insertion in order to ascertain how the proposed scheme copes with different sizes of XML data.

First, the time spent for implementing the insertions increases as the XMark file increases due to the growth in the number of insertions. The proposed scheme achieved the shortest spent time in all queries compared with the other two schemes. Figure 5.98 demonstrates the results for all schemes.

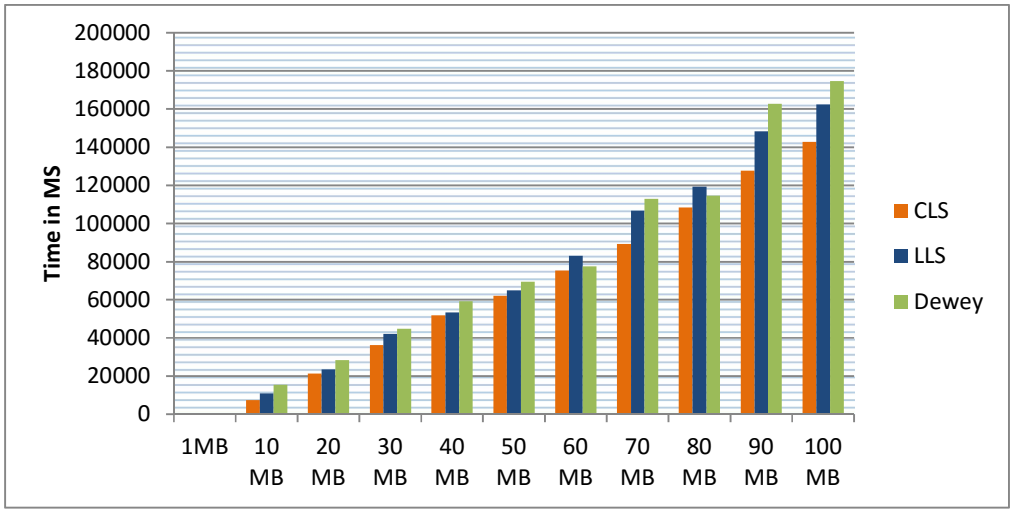


Figure 5-98: spent time for uniform insertions

Second, the size of the labels was also affected by the insertions as XMark files grow. The proposed scheme produced bigger sizes than the LLS and Dewy labelling schemes. Figure 5.99 illustrates these results.

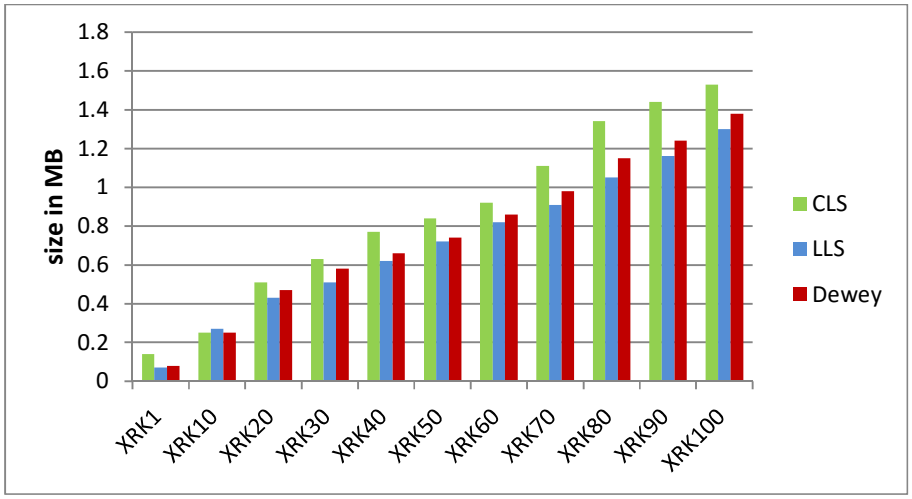


Figure 5-99: size of the labels for uniform insertions

- **Statistical Description of the Results:**

Since the idea of uniform insertions is to insert new nodes between two respective nodes, it was considered by researchers that the features of certain respective nodes may affect the efficiency (McGill, Tukey, & Larsen, 1978). Therefore, this test was carried out to test the influence of the targeted scheme on time when executing uniform insertions. Thus, the test of uniform insertions was accomplished based on the population of the eleven *XMark* files.

Based on the generated box plots for this testing, illustrated below, all these results show that the proposed labelling scheme is faster than the others. In order to find the statistical significance, p-values were calculated in all tests and as a result, all cases are lower than the significance level. Thus, the null hypothesis was rejected and the alternative

hypothesis was accepted. Consequently, the difference in the performance of the proposed scheme and the other schemes in terms of labelling nodes has a direct influence on time.

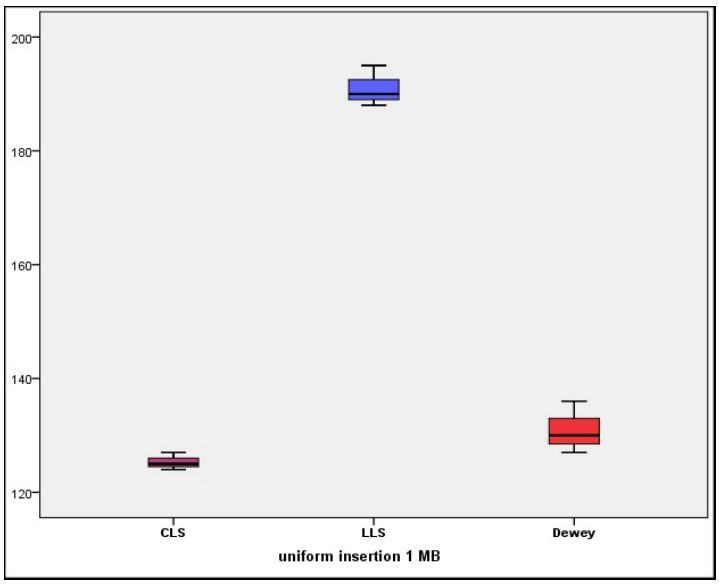


Figure 5-100: Boxplot between CLS, LLS & Dewey schemes when executing the Uniform insertion using 1 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-101: the p-value between the CLS and & LLS schemes when executing the Uniform insertion using 1 MB file.

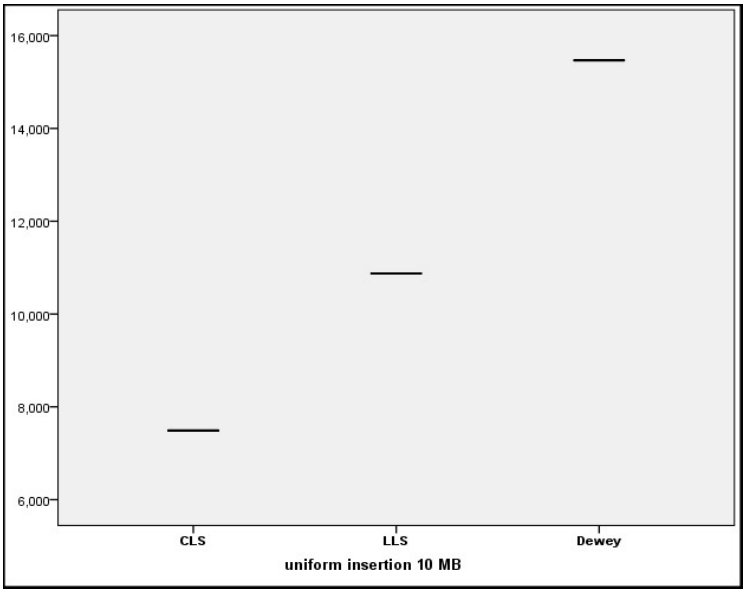


Figure 5-102: Boxplot between CLS, LLS & Dewey schemes when executing the Uniform insertion using 10 MB file.

p-value between CLS & LLS schemes =	0.046
p-value between CLS & Dewey schemes =	0.041

Figure 5-103: the p-value between the CLS, LLS and Dewey schemes when executing the Uniform insertion using 10 MB file.

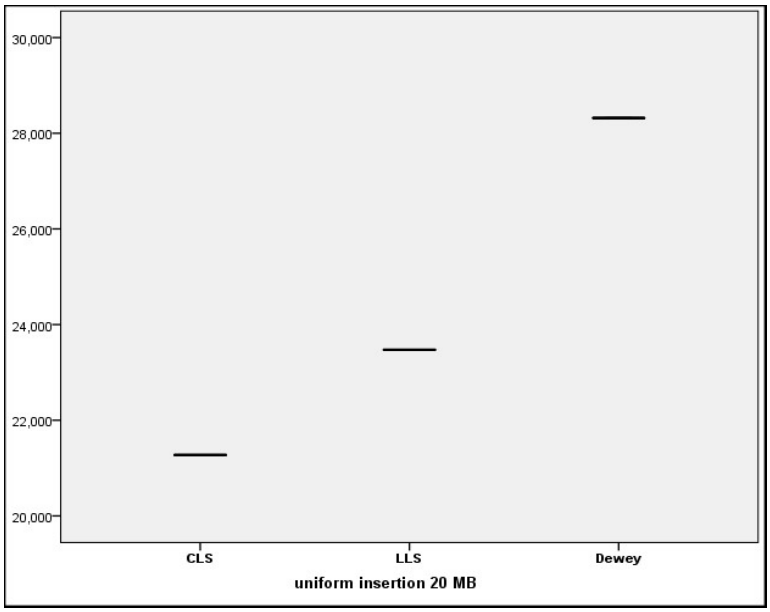


Figure 5-104: Boxplot between CLS, LLS & Dewey schemes when executing the Uniform insertion using 20 MB file.

p-value between CLS & LLS schemes =	0.033
p-value between CLS & Dewey schemes =	0.029

Figure 5-105: the p-value between the CLS, LLS and Dewey schemes when executing the Uniform insertion using 20 MB file.

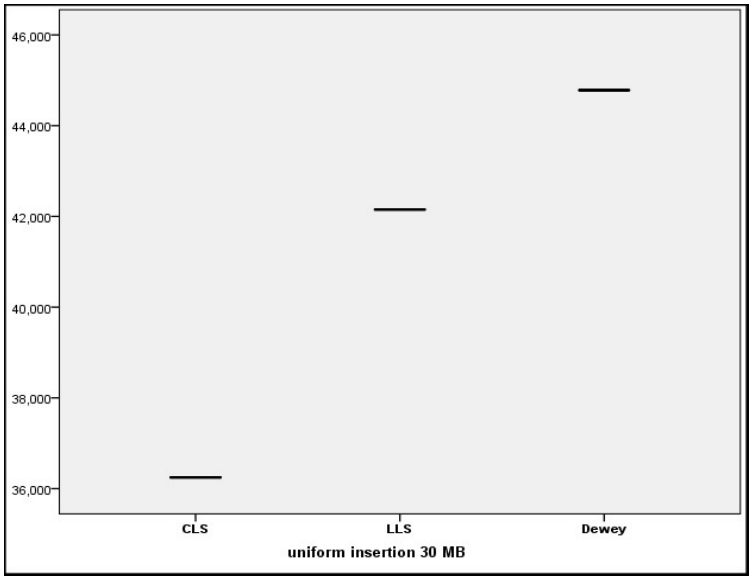


Figure 5-106: Boxplot between CLS & LLS & Dewey schemes when executing the Uniform insertion using 30 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-107: the p-value between the CLS and & LLS schemes when executing the Uniform insertion using 30 MB file.

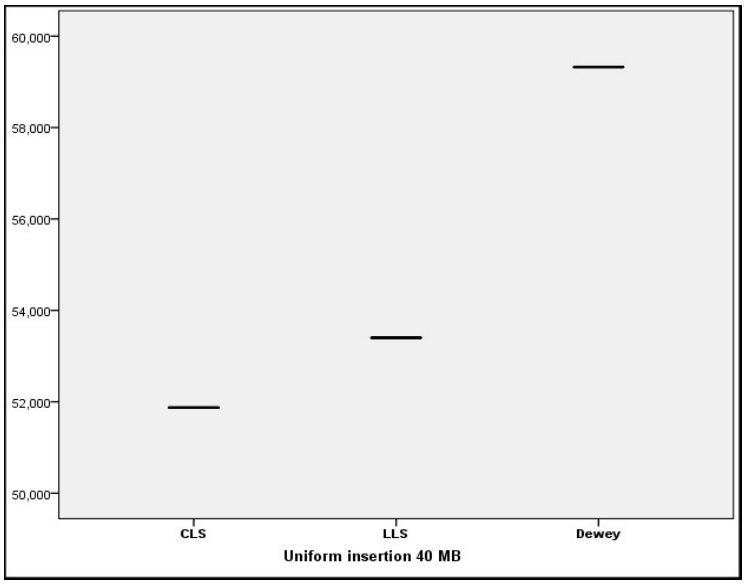


Figure 5-108: Boxplot between CLS & LLS & Dewey schemes when executing the Uniform insertion using 40 MB file.

p-value between CLS & LLS schemes =	0.032
p-value between CLS & Dewey schemes =	0.036

Figure 5-109: the p-value between the CLS, LLS and Dewey schemes when executing the Uniform insertion using 40 MB file.

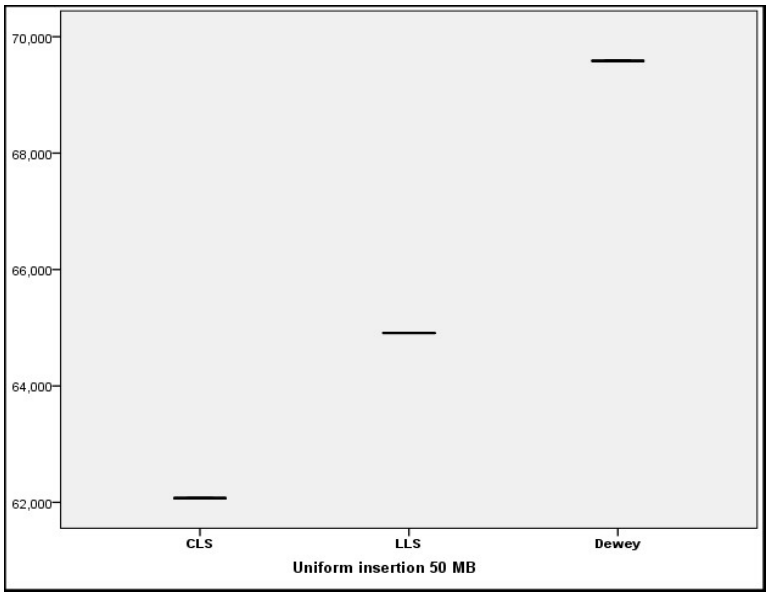


Figure 5-110: Boxplot between CLS & LLS & Dewey schemes when executing the Uniform insertion using 50 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-111: the p-value between the CLS, LLS and Dewey schemes when executing the Uniform insertion using 50 MB file.

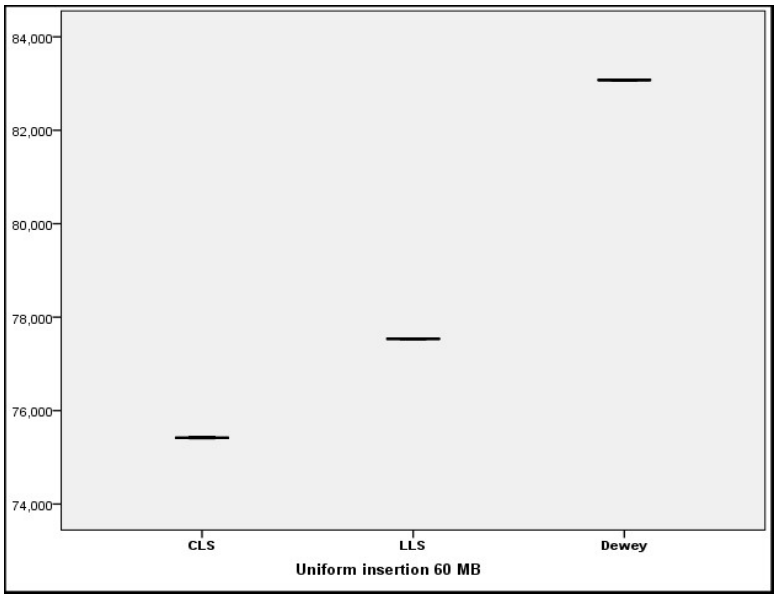


Figure 5-112: Boxplot between CLS, LLS & Dewey when executing the Uniform insertion using 60 MB file.

p-value between CLS & LLS schemes =	0.024
p-value between CLS & Dewey schemes =	0.031

Figure 5-113: show the p-value between the CLS, LLS and Dewey schemes when executing the Uniform insertion using 60 MB file.

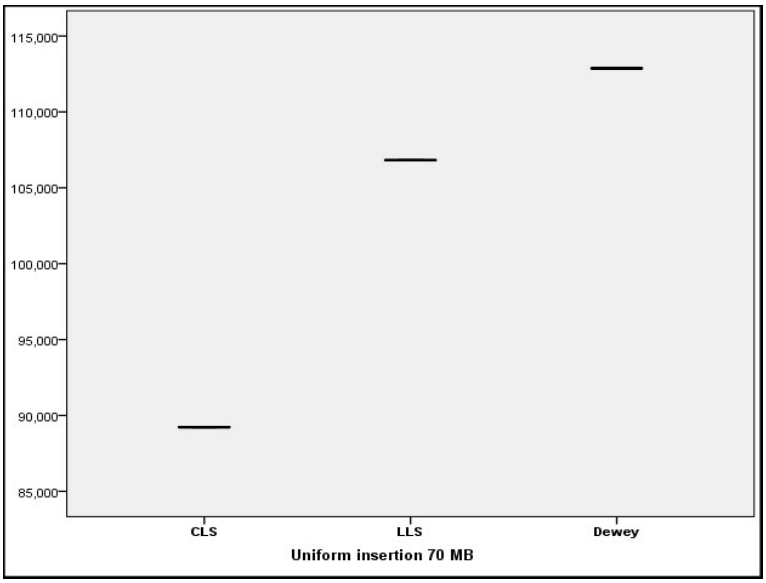


Figure 5-114: Boxplot between CLS, LLS & Dewey schemes when executing the Uniform insertion using 70 MB file.

p-value between CLS & LLS schemes =	0.025
p-value between CLS & Dewey schemes =	0.033

Figure 5-115: the p-value between the CLS, LLS and Dewey schemes when executing the Uniform insertion using 70 MB file.

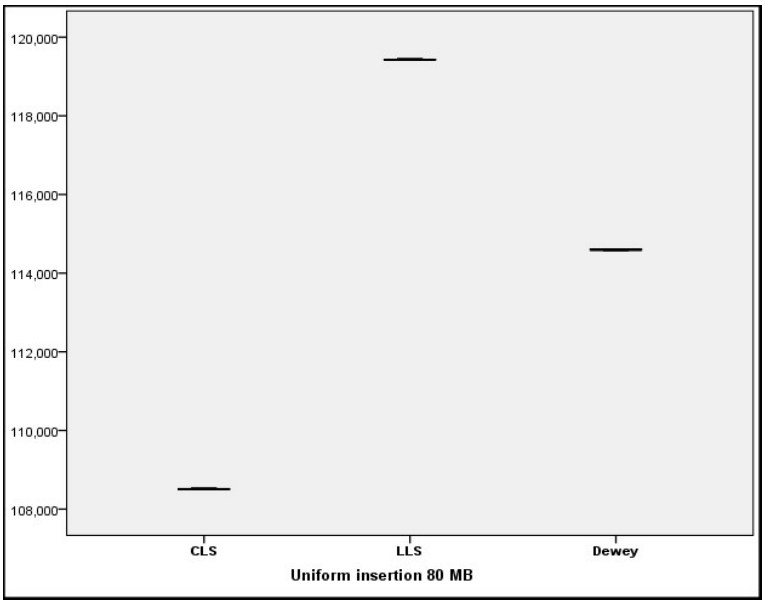


Figure 5-116: Boxplot between CLS, LLS & Dewey schemes when executing the Uniform insertion using 80MB file.

p-value between CLS & LLS schemes =	0.026
p-value between CLS & Dewey schemes =	0.019

Figure 5-117: the p-value between the CLS, LLS and Dewey schemes when executing the Uniform insertion using 80 MB file.

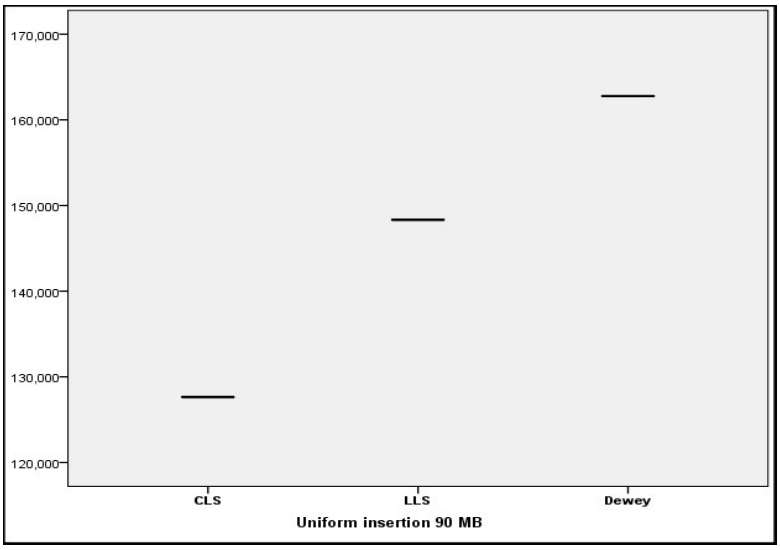


Figure 5-118: Boxplot between CLS, LLS & Dewey schemes when executing the Uniform insertion using 90 MB file.

p-value between CLS & LLS schemes =	0.040
p-value between CLS & Dewey schemes =	0.036

Figure 5-119: the p-value between the CLS, LLS and Dewey schemes when executing the Uniform insertion using 90 MB file.

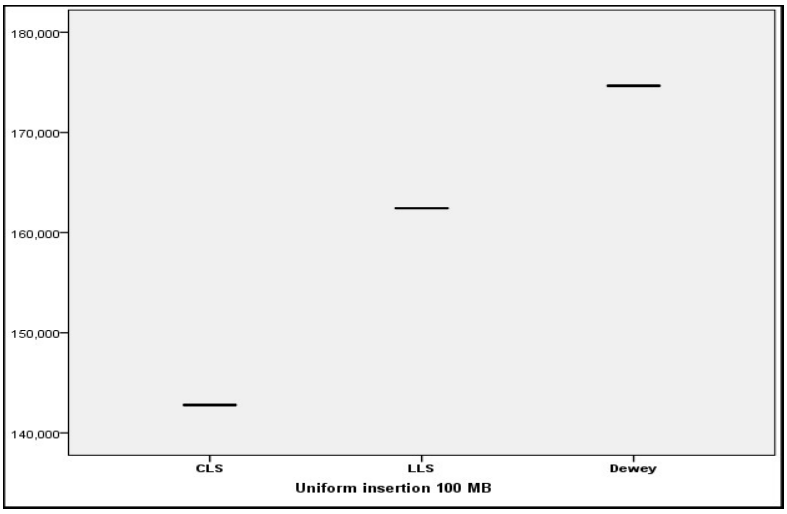


Figure 5-120: Boxplot between CLS, LLS & Dewey schemes when executing the Uniform insertion using 100 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-121: the p-value between the CLS, LLS and Dewey schemes when executing the Uniform insertion using 100 MB file.

5.3.1.2 Ordered Skewed

a) First, regarding the time spent, the XMRK1 file was used to test this experiment. The number of insertions was used to monitor the changes. These numbers start from one thousand up to ten thousand. Figure 5.122 shows these results. As can be seen, the results of the three schemes are almost identical. However, the proposed scheme achieved slightly better results in nine out of ten cases (2, 3, 4, 5, 6, 7, 8, 9 and 10 thousand). The LLS scheme achieved better results in number one thousand insertions.

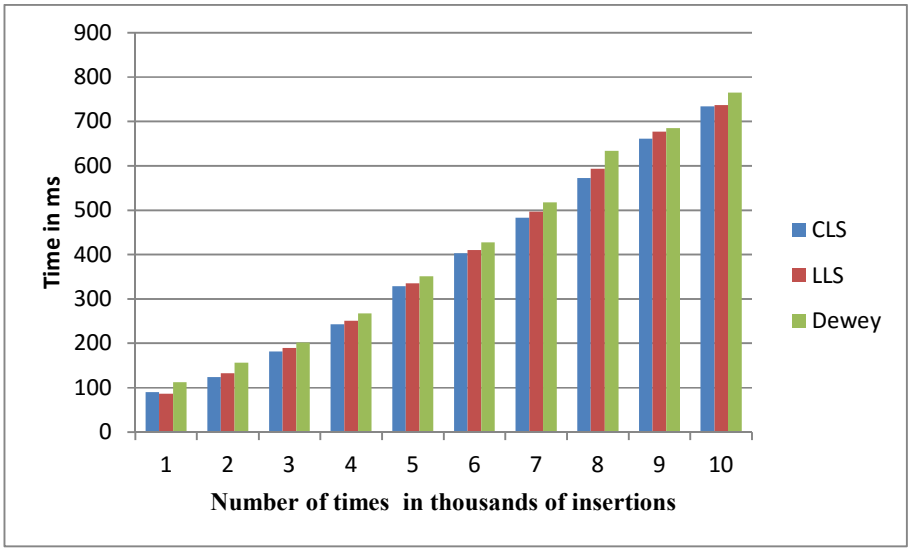


Figure 5-122: Execution time for all schemes

b) Second, with respect to the size of the labels, the proposed scheme and LLS scheme achieved the same results and better than the Dewey scheme in one testing (7 thousand). However, the proposed scheme achieved best results in nine other experiments. Figure 5.123 shows these results.

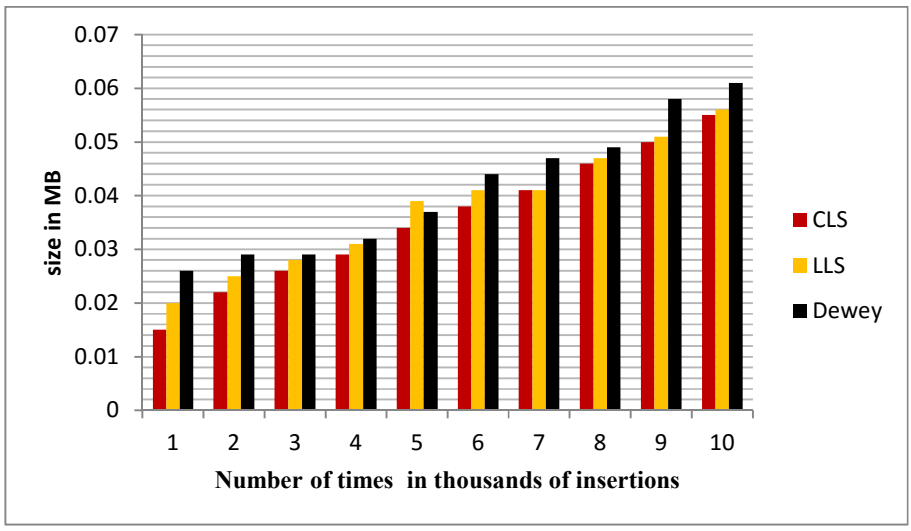


Figure 5-123: size of the labels for all schemes

- **Statistical Description of the Results:**

As explained above, this type of insertion is based on inserting new nodes repeatedly before or after a particular node. For this reason, this testing concentrated on increasing the number of inserted nodes as shown in the previous two figures. Thus, the number of populations that were used for the Wilcoxon rank-sum test is ten. These results were illustrated as box plots below. These plot boxes show that the proposed scheme is faster than the other schemes in all cases except the first test where the LLS was faster.

Having ran the Wilcoxon rank-sum test, the null hypothesis was rejected as the p values were lower than the significance level. As a result, the alternative hypothesis was accepted which means that the difference was significant and had influenced time.

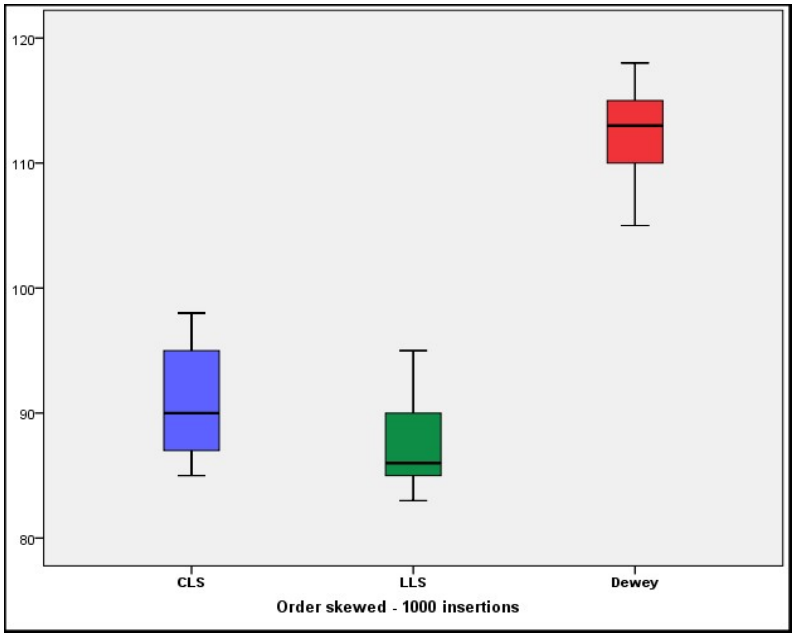


Figure 5-124: Boxplot between CLS, LLS & Dewey schemes when executing the Ordered Skewed insertion using 1000 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-125: the p-value between the CLS, LLS and Dewey schemes when executing the Ordered Skewed insertion using 1000 MB file.

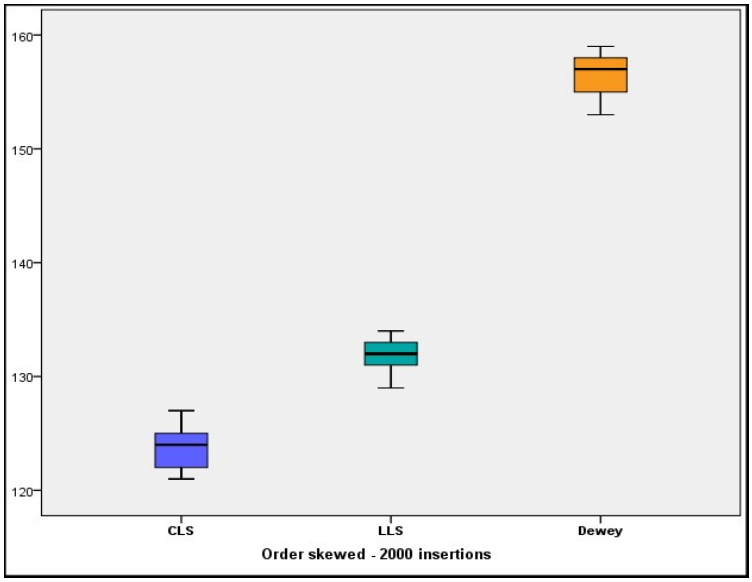


Figure 5-126: Boxplot between CLS & LLS & Dewey schemes when executing the Ordered Skewed insertion using 2000 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-127: the p-value between the CLS, LLS and Dewey schemes when executing the Ordered Skewed insertion using 2000 MB file.

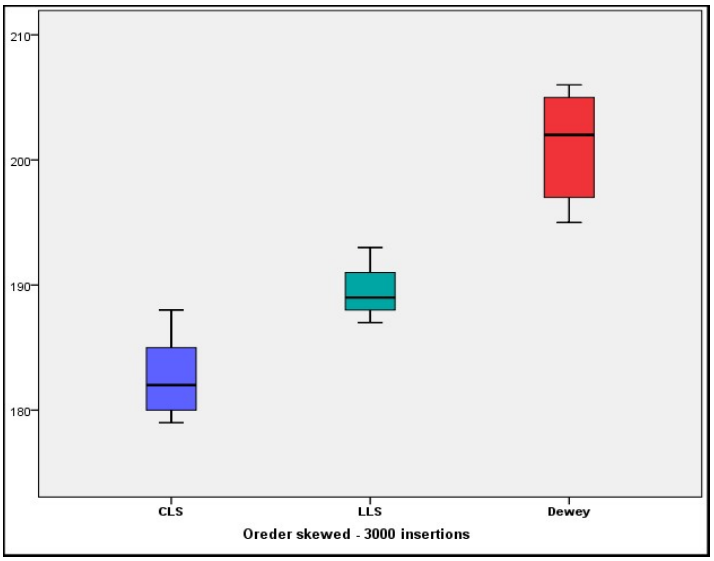


Figure 5-128: Boxplot between CLS, LLS & Dewey schemes when executing the Ordered Skewed insertion using 3000 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-129: the p-value between the CLS, LLS and Dewey schemes when executing the Ordered Skewed insertion using 3000 MB file.

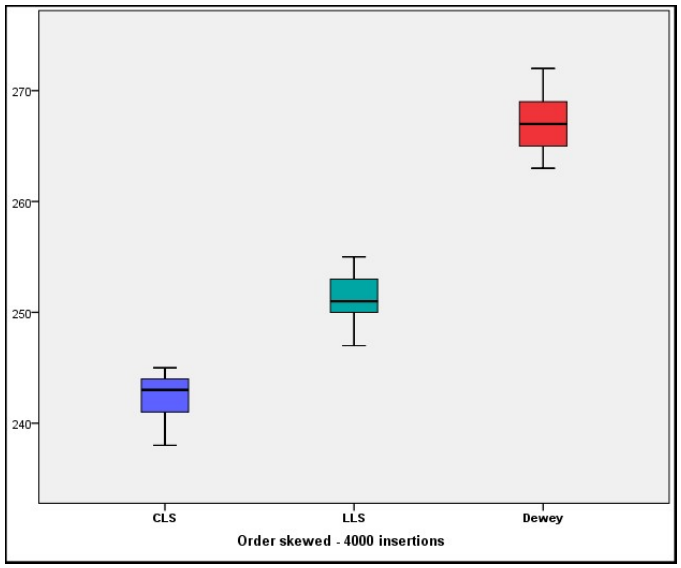


Figure 5-130: Boxplot between CLS, LLS and Dewey schemes when executing the Ordered Skewed insertion using 4000 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-131: the p-value between the CLS, LLS and Dewey schemes when executing the Ordered Skewed insertion using 4000 MB file.

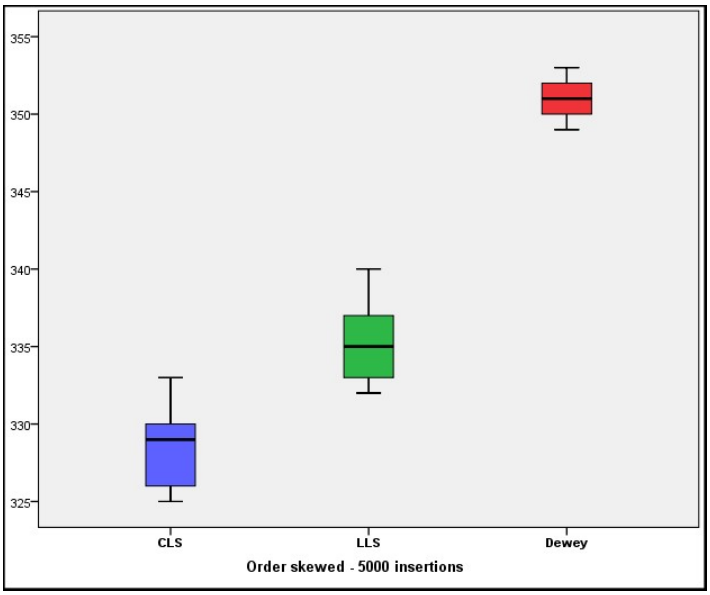


Figure 5-132: Boxplot between CLS, LLS & Dewey schemes when executing the Ordered Skewed insertion using 5000 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-133: the p-value between the CLS, LLS and Dewey schemes when executing the Ordered Skewed insertion using 5000 MB file.

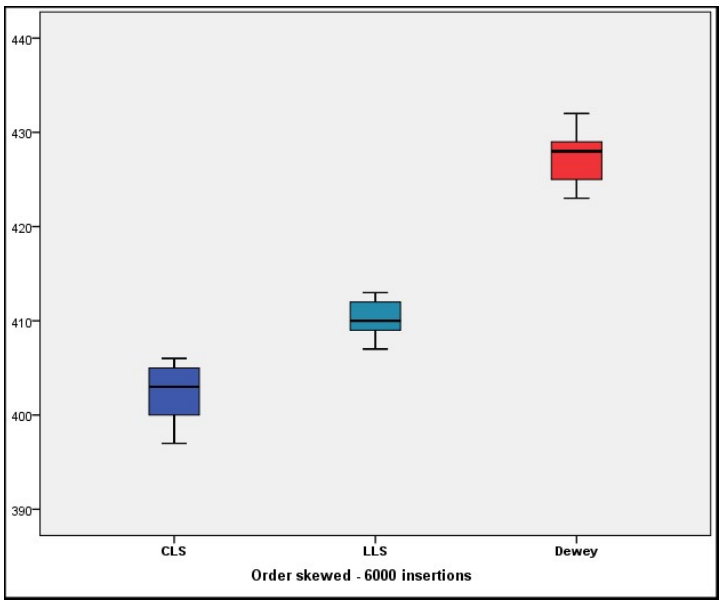


Figure 5-134: Boxplot between CLS & LLS & Dewey schemes when executing the Ordered Skewed insertion using 6000 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-135: the p-value between the CLS, LLS and Dewey schemes when executing the Ordered Skewed insertion using 6000 MB file.

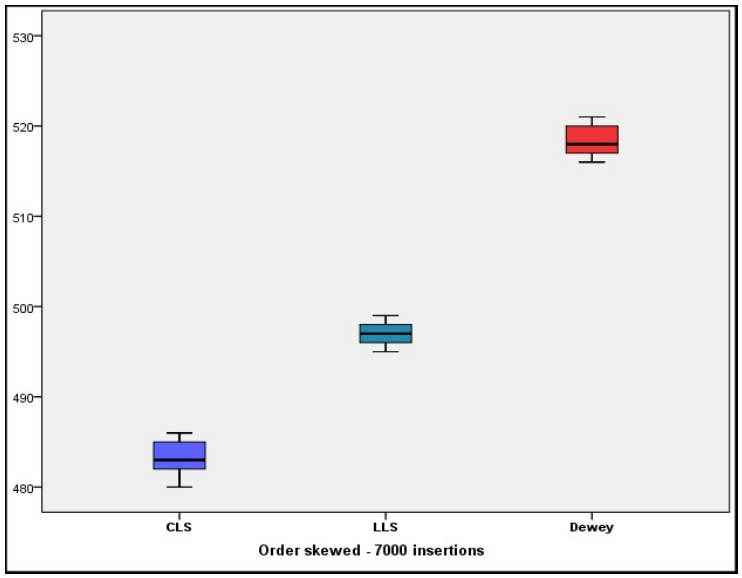


Figure 5-136: Boxplot between CLS, LLS & Dewey schemes when executing the Ordered Skewed insertion using 7000 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-137: the p-value between the CLS, LLS and Dewey schemes when executing the Ordered Skewed insertion using 7000 MB file.

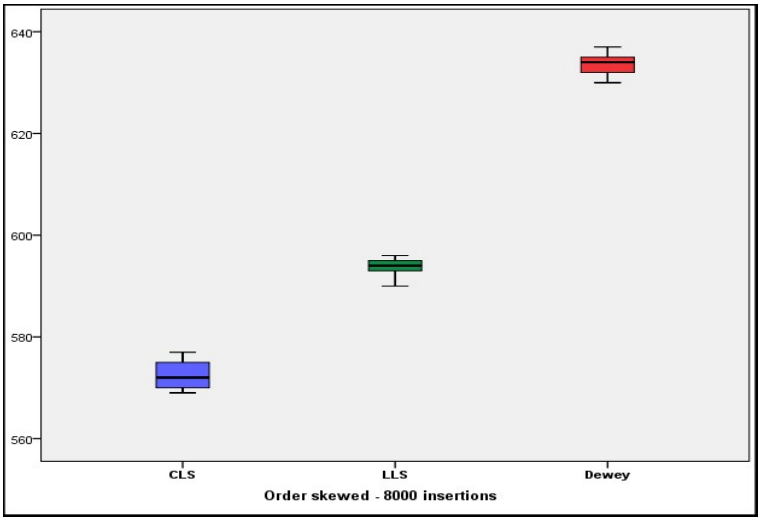


Figure 5-138: Boxplot between CLS, LLS & Dewey schemes when executing the Ordered Skewed insertion using 8000 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-139: the p-value between the CLS, LLS and Dewey schemes when executing the Ordered Skewed insertion using 8000 MB file.

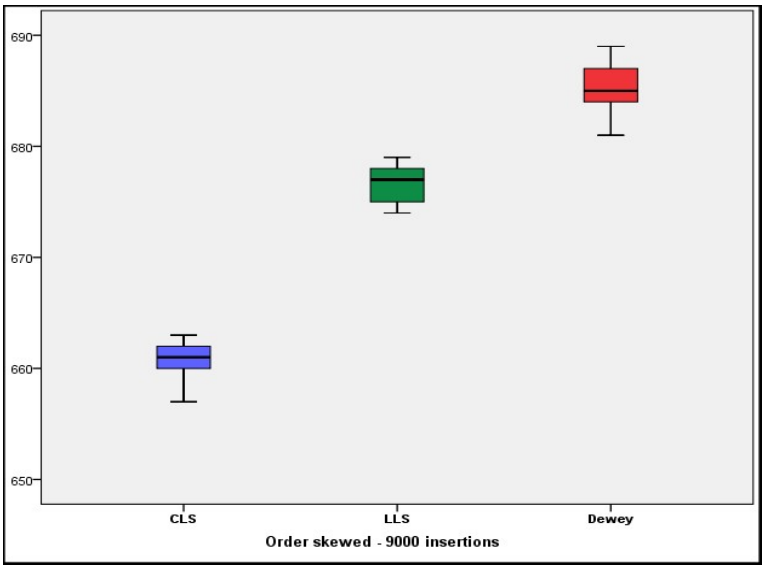


Figure 5-140: Boxplot between CLS, LLS & Dewey schemes when executing the Ordered Skewed insertion using 9000 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-141: the p-value between the CLS, LLS and Dewey schemes when executing the Ordered Skewed insertion using 9000 MB file.

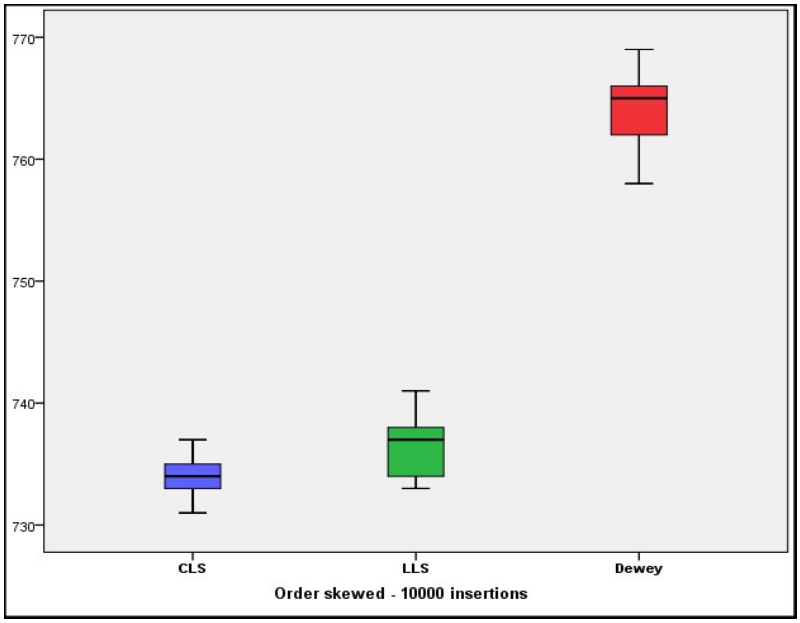


Figure 5-142: Boxplot between CLS & LLS & Dewey schemes when executing the Ordered Skewed insertion using 10000 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-143: the p-value between the CLS, LLS and Dewey schemes when executing the Ordered Skewed insertion using 10000 MB file.

5.3.1.3 Random Skewed

a) With regard to the time spent, again the XMRK1 file was used to test this experiment and the number of insertions was used to monitor the changes. The proposed scheme achieved better results in nine out of ten (2, 3, 4, 5, 6, 7, 8, 9 and 10 thousand) experiments. However, the LLS scheme achieved better results in one experiment (1 thousand). Figure 5.144 shows these results.

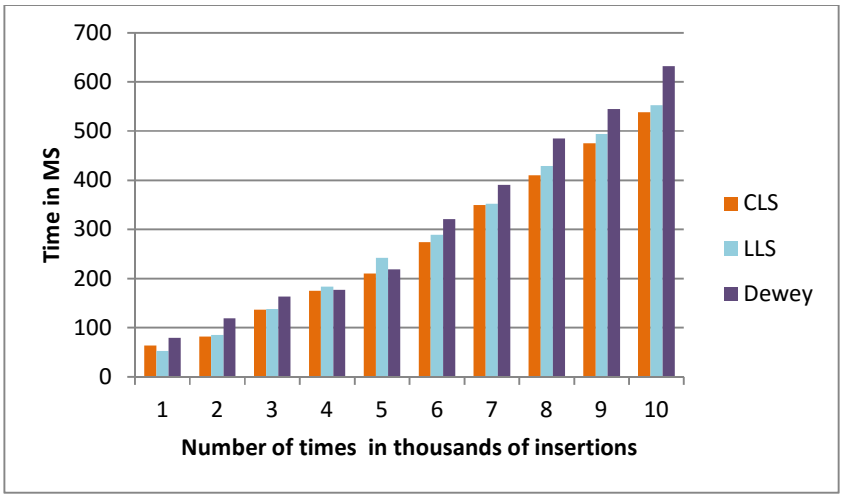


Figure 5-144: Execution time for random insertions.

b) With respect to the size of the labels, again, the XMRK1 file was used for this experiment. However, the proposed scheme has not achieved better results in all experiments. The LLS scheme has achieved best results in 1, 2, 5, 6, 7, 8, 9, and 10; whereas, the Dewey scheme was the best one in queries 3 and 4. Figure 5.145 shows these results.

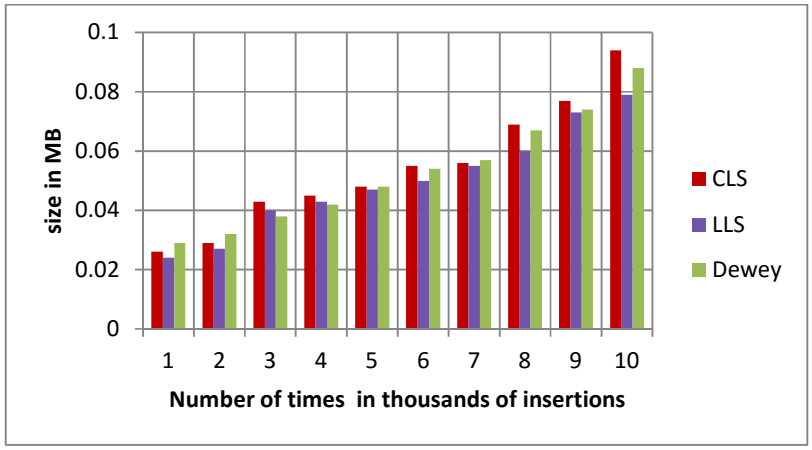


Figure 5-145: size of the labels for random Skewed insertions.

- **Statistical Description of the Results:**

Similar to the ordered skewed insertions, the populations used for testing were based on time rather than size. The box plots that were executed show that the distribution was moved to the proposed scheme all cases except the first and seventh cases. As a result, the proposed scheme showed better performance in terms of labelling XML documents.

Regarding the Wilcoxon rank-sum test, all tests show that the p values were lower than the significance level which is 0.05. Thus, the null hypothesis was rejected and, thus, the alternative hypothesis was accepted, so, the proposed scheme shows the possibility of enhancing the random skewed insertion.

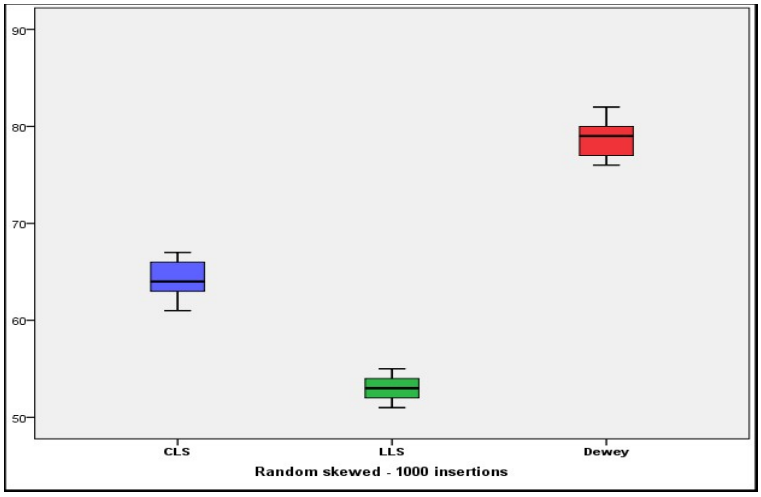


Figure 5-146: Boxplot between CLS, LLS & Dewey schemes when executing the Random Skewed insertion using 1000 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-147: the p-value between the CLS, LLS and Dewey schemes when executing the Random Skewed insertion using 1000 MB file.

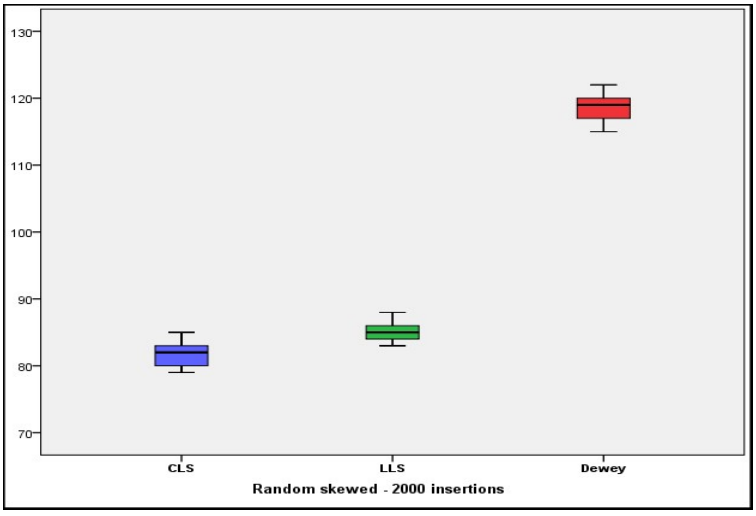


Figure 5-148: Boxplot between CLS, LLS & Dewey schemes when executing the Random Skewed insertion using 2000 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-149: the p-value between the CLS, LLS and Dewey schemes when executing the Random Skewed insertion using 2000 MB file.

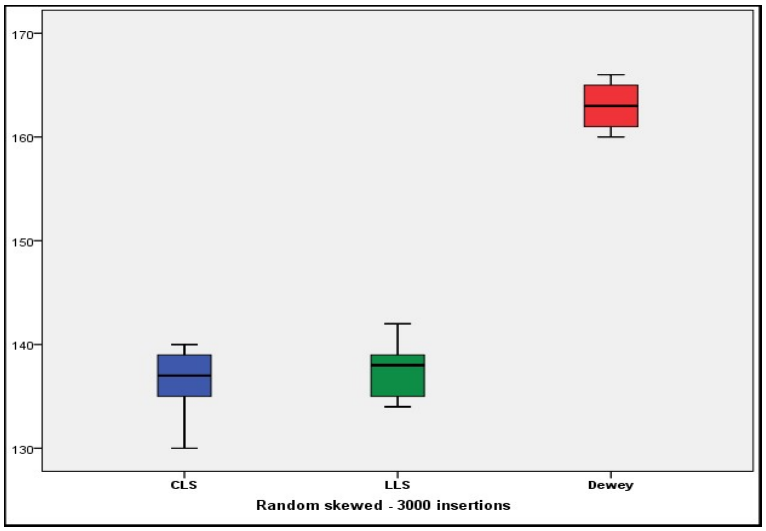


Figure 5-150: Boxplot between CLS & LLS & Dewey schemes when executing the Random Skewed insertion using 3000 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-151: the p-value between the CLS, LLS and Dewey schemes when executing the Random Skewed insertion using 3000 MB file.

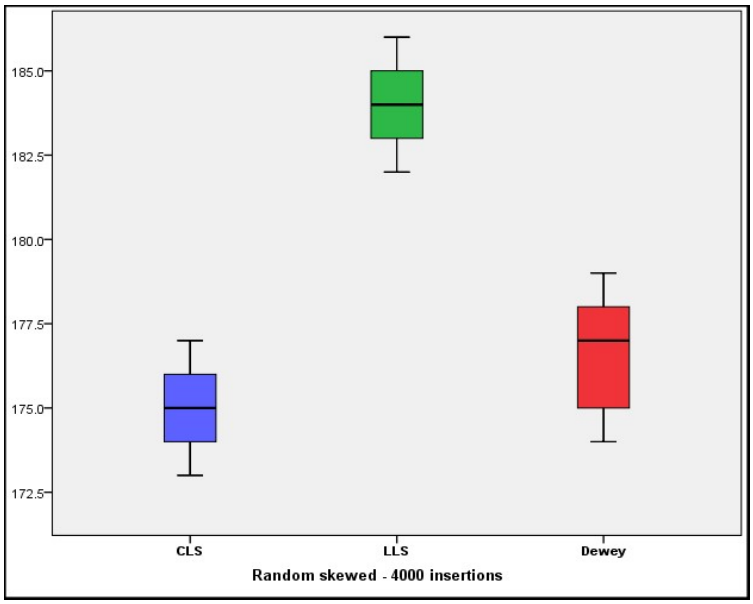


Figure 5-152: Boxplot between CLS, LLS and Dewey schemes when executing the Random Skewed insertion using 4000 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.043

Figure 5-153: the p-value between the CLS, LLS and Dewey schemes when executing the Random Skewed insertion using 4000 MB file.

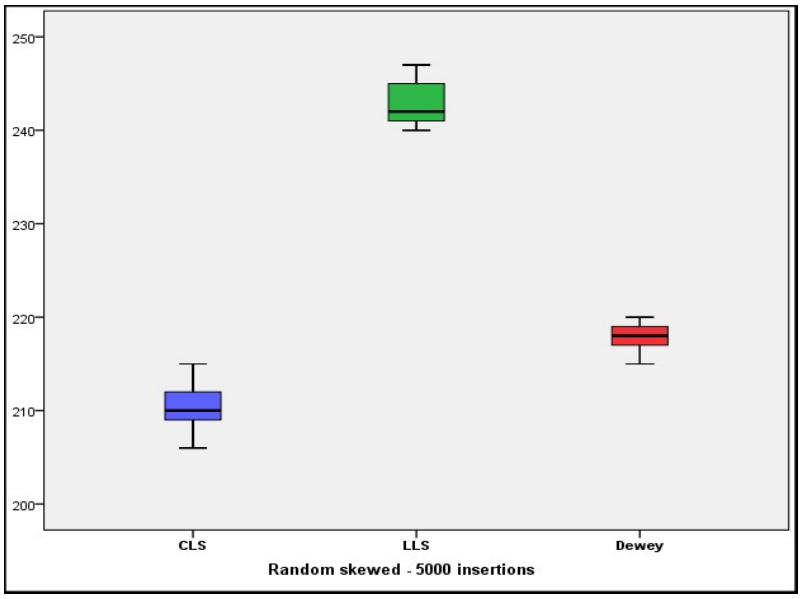


Figure 5-154: Boxplot between CLS & LLS & Dewey schemes when executing the Random Skewed insertion using 5000 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.043

Figure 5-155: the p-value between the CLS, LLS and Dewey schemes when executing the Random Skewed insertion using 5000 MB file.

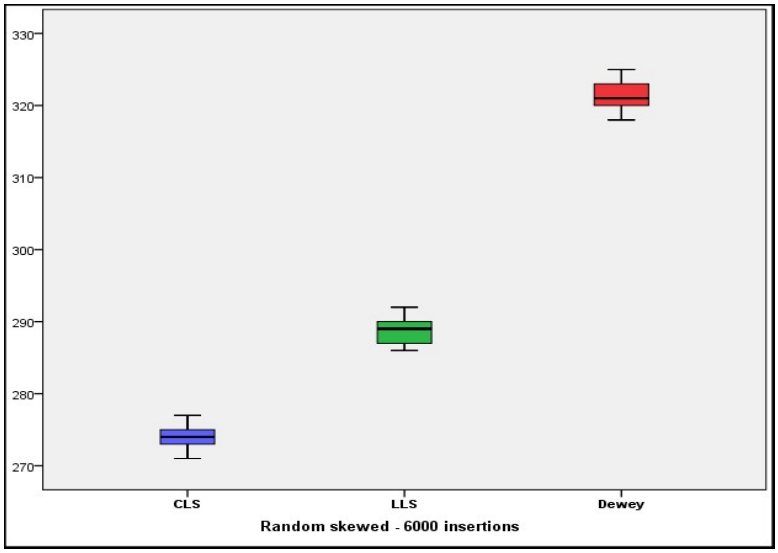


Figure 5-156: Boxplot between CLS, LLS & Dewey schemes when executing the Random Skewed insertion using 6000 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.043

Figure 5-157: the p-value between the CLS, LLS and Dewey schemes when executing the Random Skewed insertion using 6000 MB file.

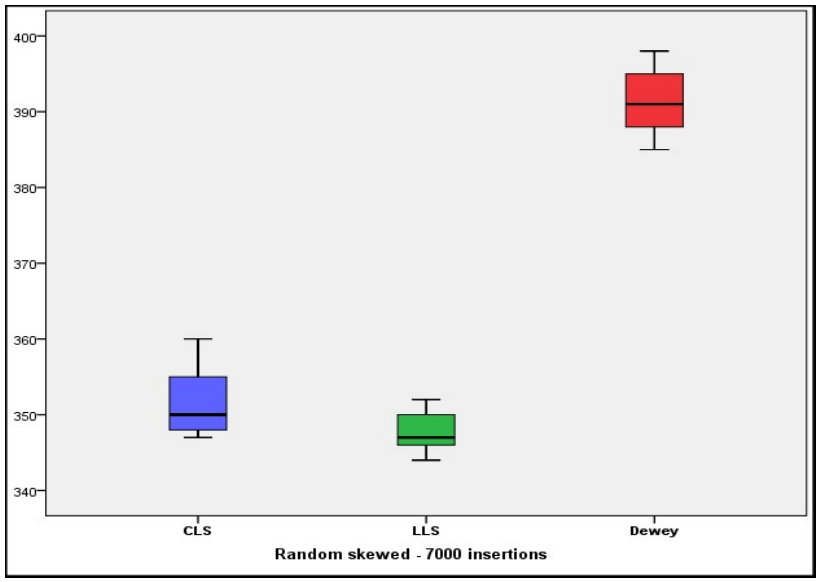


Figure 5-158: Boxplot between CLS, LLS & Dewey schemes when executing the Random Skewed insertion using 7000 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.043

Figure 5-159: the p-value between the CLS, LLS and Dewey schemes when executing the Random Skewed insertion using 7000 MB file.

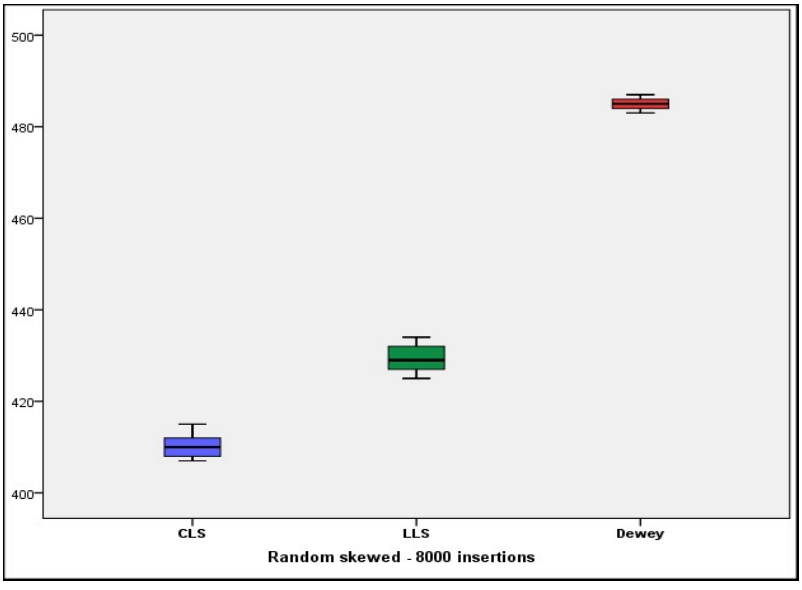


Figure 5-160: Boxplot between CLS, LLS & Dewey schemes when executing the Random Skewed insertion using 8000 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-161: the p-value between the CLS, LLS and Dewey schemes when executing the Random Skewed insertion using 8000 MB file.

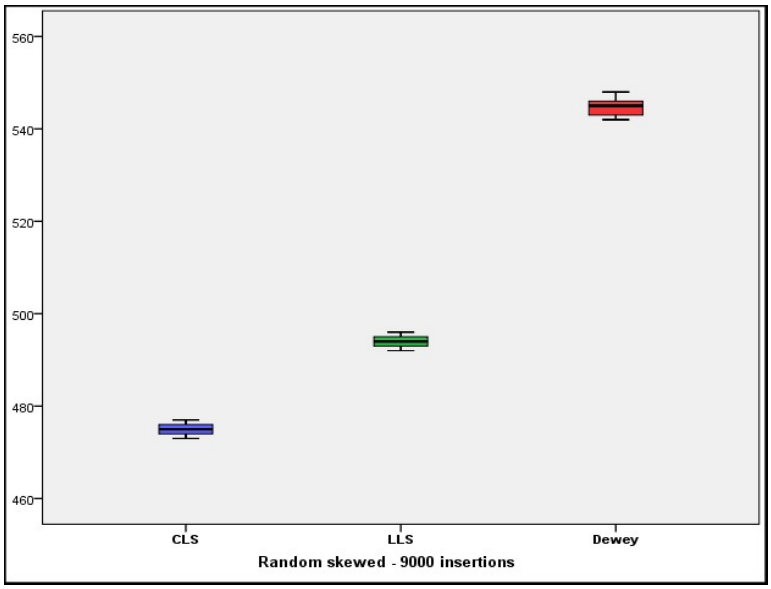


Figure 5-162: Boxplot between CLS, LLS & Dewey schemes when executing the Random Skewed insertion using 9000 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-163: the p-value between the CLS, LLS and Dewey schemes when executing the Random Skewed insertion using 9000 MB file.

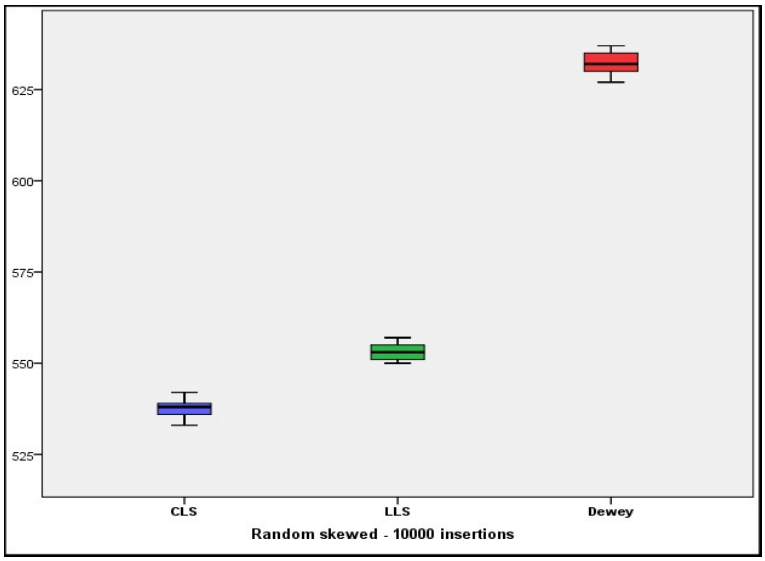


Figure 5-164: Boxplot between CLS, LLS & Dewey schemes when executing the Random Skewed insertion using 10000 MB file.

p-value between CLS & LLS schemes =	0.043
p-value between CLS & Dewey schemes =	0.042

Figure 5-165: the p-value between the CLS, LLS and Dewey schemes when executing the Random Skewed insertion using 10000 MB file.

To conclude, in terms of the static XML documents, the proposed scheme shows an improvement in the labelling of XML documents over the LLS and Dewey labelling schemes. The response time and the size of the labels are influenced by the XML document size. Therefore, the proposed scheme achieved better results in most cases for testing different aspects and for different sizes, except for a small number of cases where the LLS and Dewey schemes achieved slightly better results than the proposed scheme.

5.3.2 Determining Relationships:

These experiments test the scheme in terms of determining relationships. Therefore, this experiment observes the time spent for determining relationships after inserting new nodes as follows:

- **Determining Relationships after Uniform Insertions:**

  The proposed scheme shows the best results in determining relationships after inserting new nodes between two nodes. Figure 5.166 shows the results.

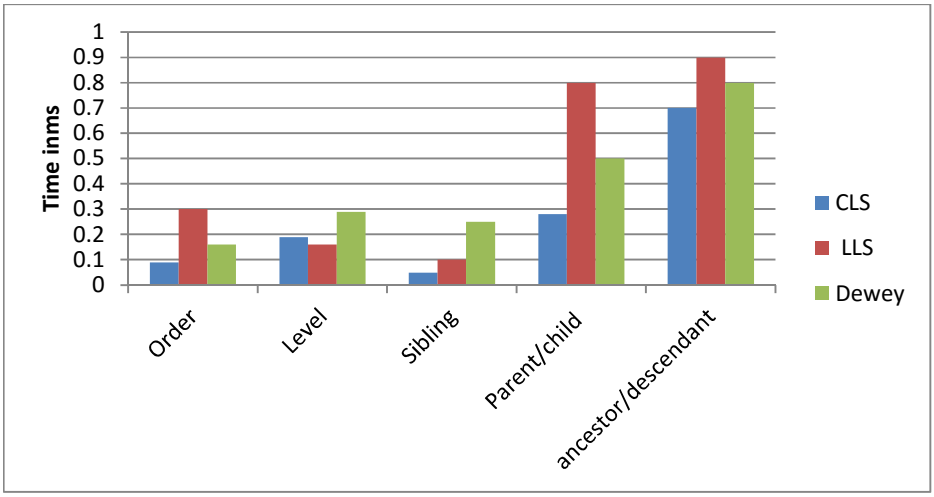


Figure 5-166: relationships after uniform insertions.

- **Statistical Description of the Results:**

The identified relationships for the experiments were tested to measure the needed time for the performance after the uniform insertions. Five populations were used for this testing according to the identified relationships. The five box plots below show that the proposed scheme was better in determining the relationships after the uniform insertion except the testing of the level of nodes where the LLS was better.

Concerning the Wilcoxon rank sum test, all the p values in all relationship cases are far lower than the significance level, and thus show strong evidence that the performance difference was significant and influenced by time.

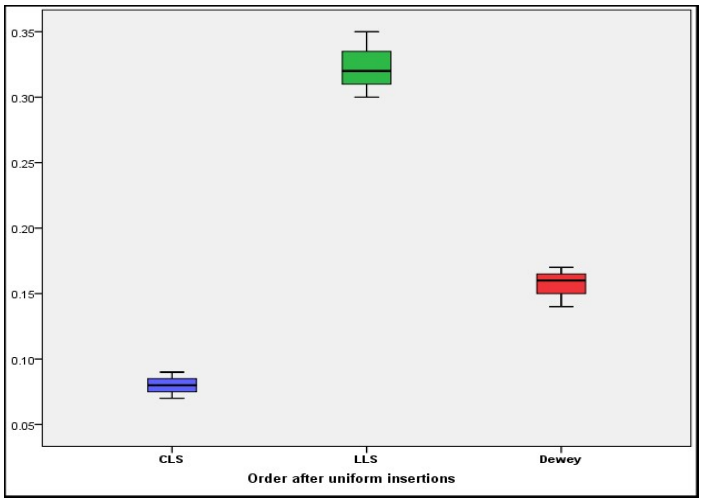


Figure 5-167: Boxplot between CLS, LLS & Dewey schemes when determining the Order after uniform insertion.

p-value between CLS & LLS schemes =	0.016
p-value between CLS & Dewey schemes =	0.016

Figure 5-168: the p-value between the CLS, LLS and Dewey schemes when determining the Order after uniform insertion.

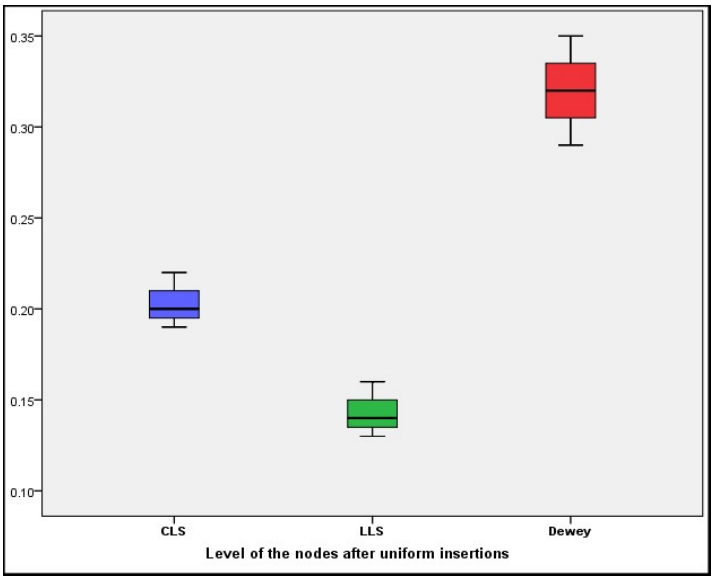


Figure 5-169: Boxplot between CLS, LLS & Dewey schemes when determining the Nodes' level after uniform insertion.

p-value between CLS & LLS schemes =	0.005
p-value between CLS & Dewey schemes =	0.003

Figure 5-170: the p-value between the CLS, LLS and Dewey schemes when determining the Nodes' level after uniform insertion.

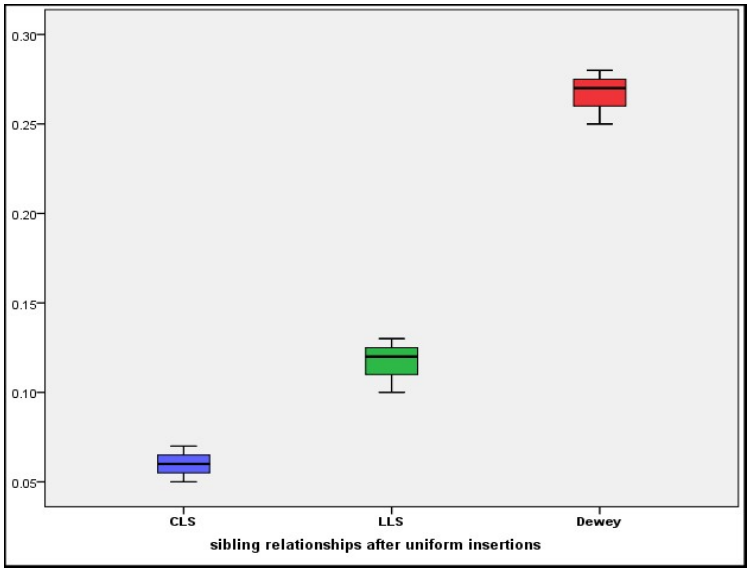


Figure 5-171: Boxplot between CLS, LLS & Dewey schemes when determining the sibling relationships after uniform insertion.

p-value between CLS & LLS schemes =	0.004
p-value between CLS & Dewey schemes =	0.004

Figure 5-172: the p-value between the CLS, LLS and Dewey schemes when determining the sibling relationships after uniform insertion.

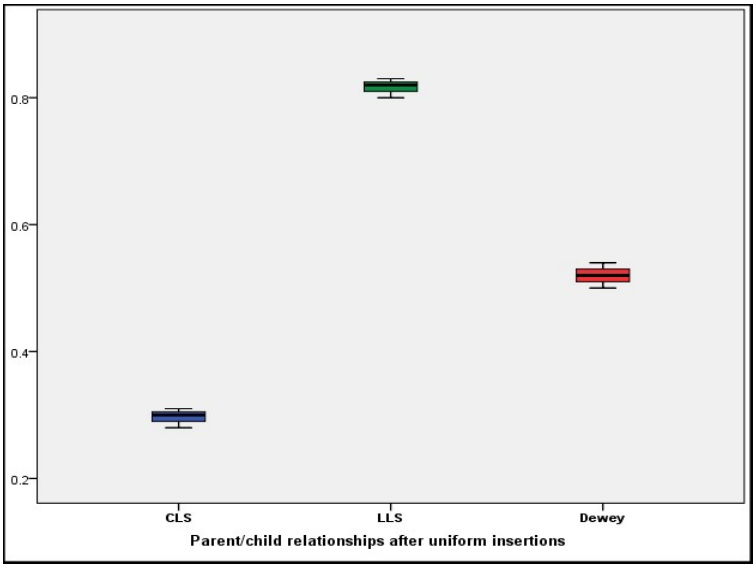


Figure 5-173: Boxplot between CLS, LLS & Dewey schemes when determining the Parent/Child relationships after uniform insertion.

p-value between CLS & LLS schemes =	0.003
p-value between CLS & Dewey schemes =	0.006

Figure 5-174: the p-value between the CLS, LLS and Dewey schemes when determining the Parent/Child relationships after uniform insertion.

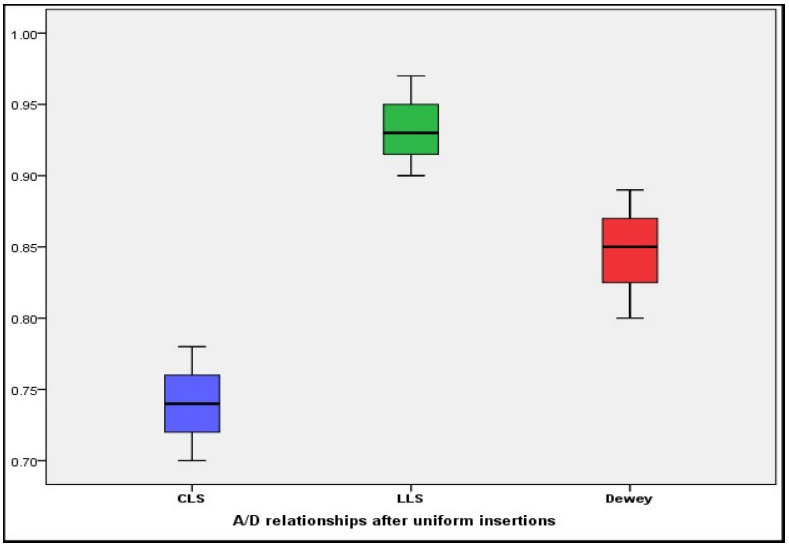


Figure 5-175: Boxplot between CLS, LLS & Dewey schemes when determining the A/D relationships after uniform insertion.

p-value between CLS & LLS schemes =	0.004
p-value between CLS & Dewey schemes =	0.004

Figure 5-176: the p-value between the CLS, LLS and Dewey schemes when determining the A/D relationships after uniform insertion.

- **Determining Relationships after Ordered Skewed Insertions**

  The proposed scheme achieved better results in all experiments except the ancestor/descendant in which the Dewey scheme shows better results. Figure 5.177 shows the results.

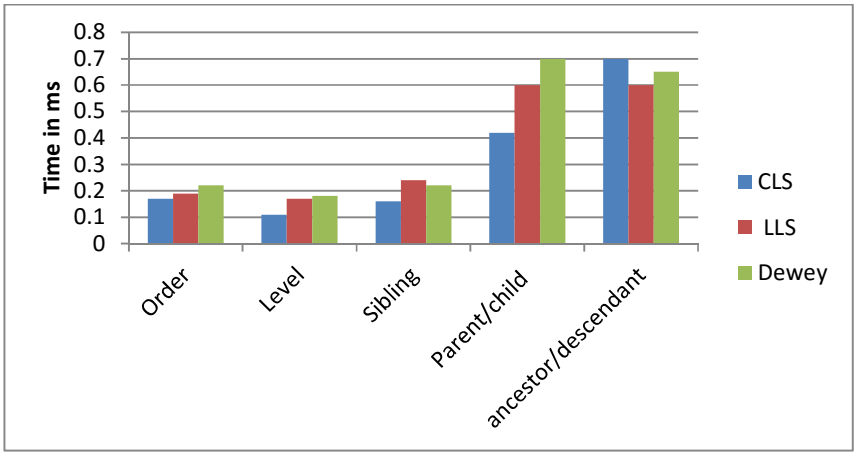


Figure 5-177: relationships after ordered skewed insertions.

- **Statistical Description of the Results:**

All the identified relationships were tested after the order skewed insertion in order to measure the performance efficiency for the targeted schemes in determining the relationships. The number of populations were tested in this experiment is five. The five box plots below show that the distributions were shifted to the proposed scheme in four

of them, whereas, the box plot for the ancestor/descendant was shifted to the LLS scheme. Therefore, this shift in the fourth box plot indicates the proposed scheme was better than the other schemes, whereas, the shift in the fifth box plot indicates that the LLS was faster than the others.

With respect to the Wilcoxon Rank-Sum test, the results are very similar to each other which far lower than the significance level. Thus, the null hypothesis was rejected and the alternative hypothesis accepted. This is strong evidence that supports the significance of the relationship.

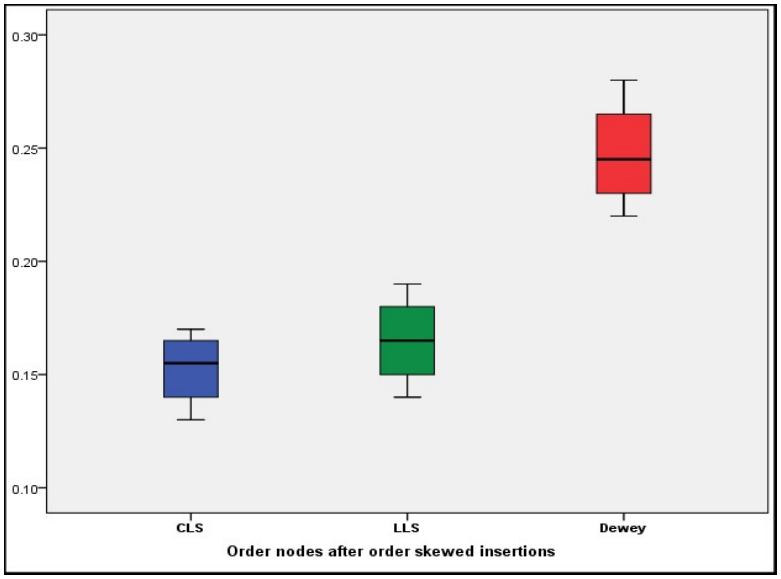


Figure 5-178: Boxplot between CLS & LLS & Dewey schemes when determining the order nodes after Ordered skewed insertion.

p-value between CLS & LLS schemes =	0.018
p-value between CLS & Dewey schemes =	0.017

Figure 5-179: shows the p-value between the CLS, LLS and Dewey schemes when determining the order nodes after Ordered skewed insertion.

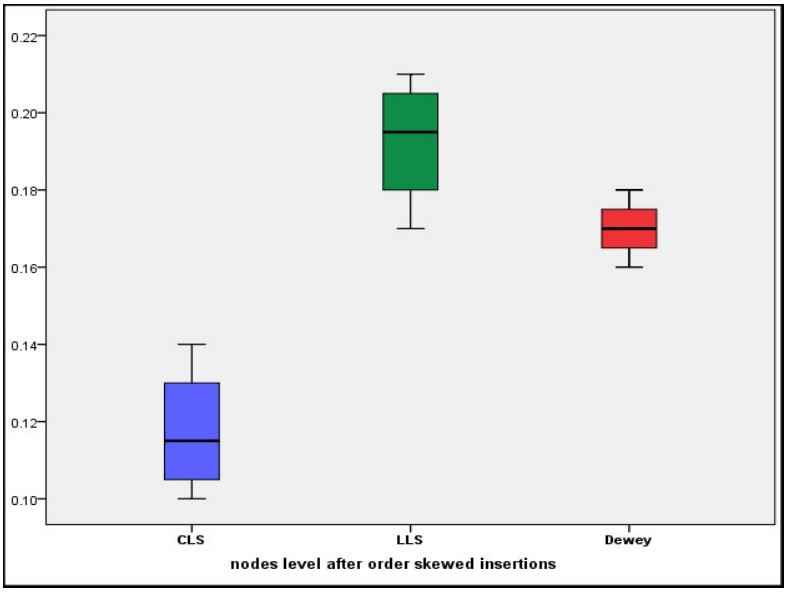


Figure 5-180: Boxplot between CLS, LLS & Dewey schemes when determining the Nodes' level after Ordered skewed insertion.

p-value between CLS & LLS schemes =	0.017
p-value between CLS & Dewey schemes =	0.014

Figure 5-181: the p-value between the CLS, LLS and Dewey schemes when determining the Nodes' level after Ordered skewed insertion.

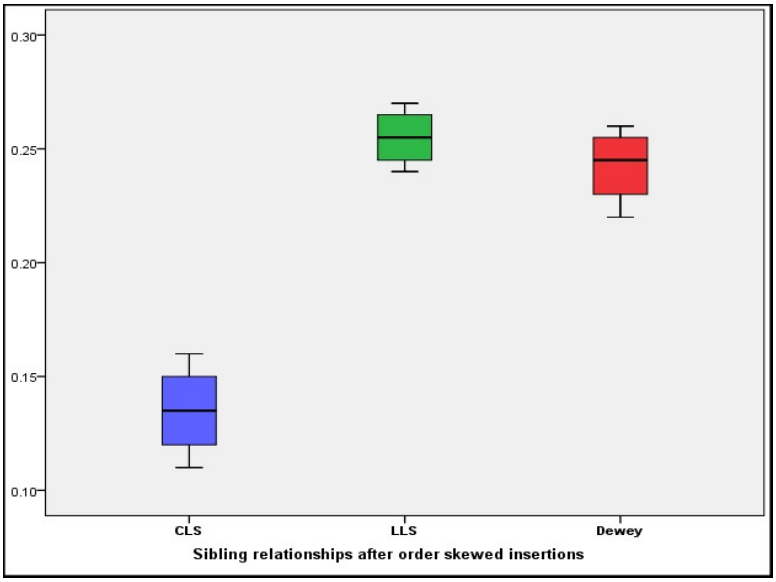


Figure 5-182: Boxplot between CLS, LLS & Dewey schemes when determining the Sibling relationships after Ordered skewed insertion.

p-value between CLS & LLS schemes =	0.017
p-value between CLS & Dewey schemes =	0.017

Figure 5-183: the p-value between the CLS, LLS and Dewey schemes when determining the Sibling relationships after Ordered skewed insertion.

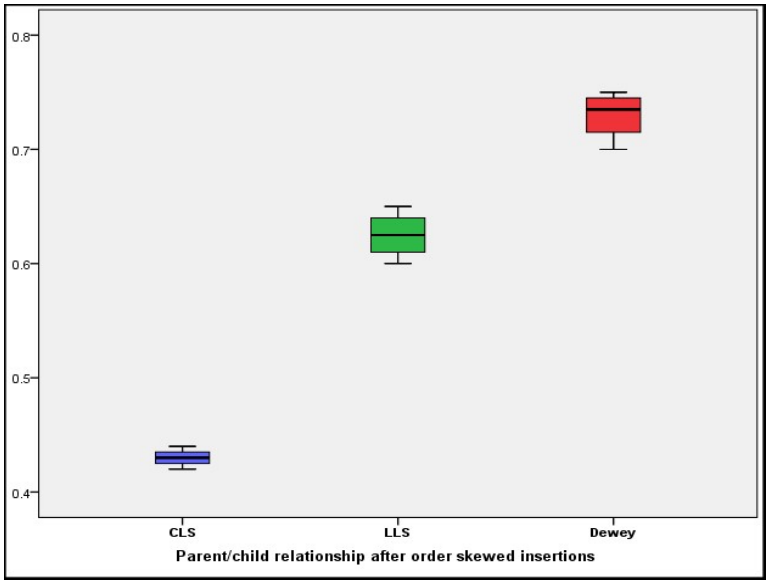


Figure 5-184: Boxplot between CLS, LLS & Dewey schemes when determining the Parent/Child relationships after Ordered skewed insertion.

p-value between CLS & LLS schemes =	0.017
p-value between CLS & Dewey schemes =	0.016

Figure 5-185: the p-value between the CLS, LLS and Dewey schemes when determining the Parent/Child relationships after Ordered skewed insertion

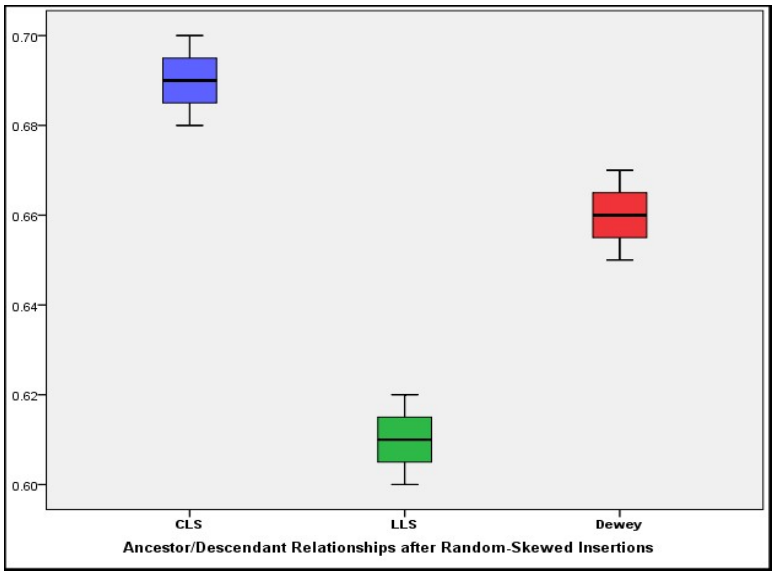


Figure 5-186: Boxplot between CLS, LLS & Dewey schemes when determining the A/D relationships after Ordered skewed insertion.

p-value between CLS & LLS schemes =	0.015
p-value between CLS & Dewey schemes =	0.016

Figure 5-187: the p-value between the CLS, LLS and Dewey schemes when determining the A/D relationships after Ordered skewed insertion.

- **Determining Relationships after Random Skewed:**

  This experiment is to insert new nodes between two other nodes randomly: the proposed scheme achieved better results in all experiments as can be seen in figure 5.188.

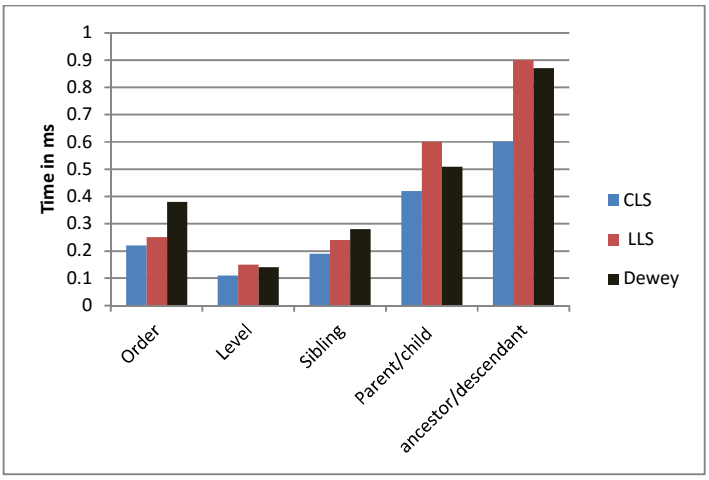


Figure 5-188: relationships after random skewed insertions.

- **Statistical Description of the Results:**

Again, an experiment for random skewed was executed in order to determine the relationships. This experiment is similar to the previous one as five populations were tested. Box plots were generated as shown below. As a result, the proposed scheme was faster in all five cases than the other schemes.

With regard to the significance, all p values for these five tests were far lower than the significance level, and therefore, the null hypothesis was rejected and the alternative hypothesis accepted. This is an implication that the performance difference was significant and influenced by time.

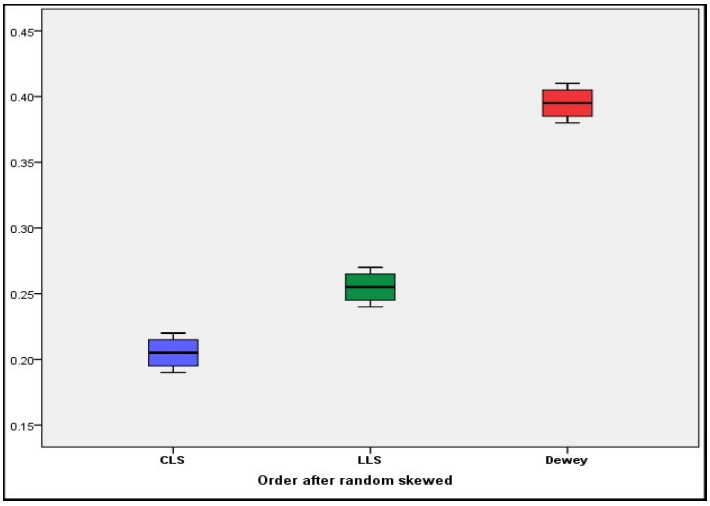


Figure 5-189: Boxplot between CLS, LLS & Dewey schemes when determining the Order after Random skewed insertion.

p-value between CLS & LLS schemes =	0.015
p-value between CLS & Dewey schemes =	0.017

Figure 5-190: the p-value between the CLS, LLS and Dewey schemes when determining the Order after Random skewed insertion.

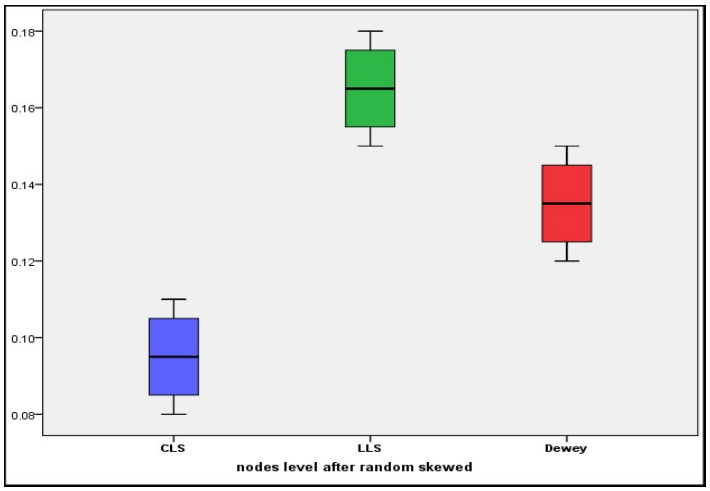


Figure 5-191: Boxplot between CLS, LLS & Dewey schemes when determining the Nodes' level after Random skewed insertion.

p-value between CLS & LLS schemes =	0.017
p-value between CLS & Dewey schemes =	0.014

Figure 5-192: show the p-value between the CLS, LLS and Dewey schemes when determining the Nodes' level after Random skewed insertion.

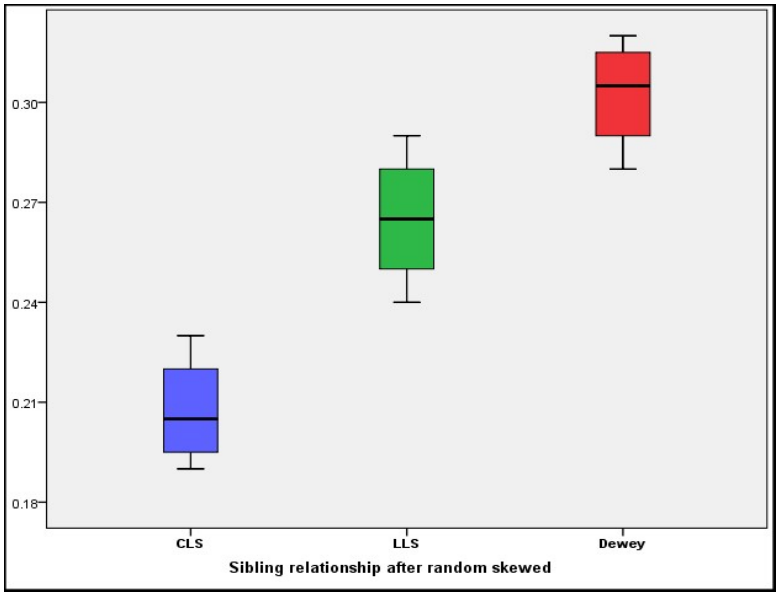


Figure 5-193: Boxplot between CLS, LLS & Dewey schemes when determining the Sibling relationships after Random skewed insertion.

p-value between CLS & LLS schemes =	0.011
p-value between CLS & Dewey schemes =	0.015

Figure 5-194: the p-value between the CLS, LLS and Dewey schemes when determining the Sibling relationships after Random skewed insertion.

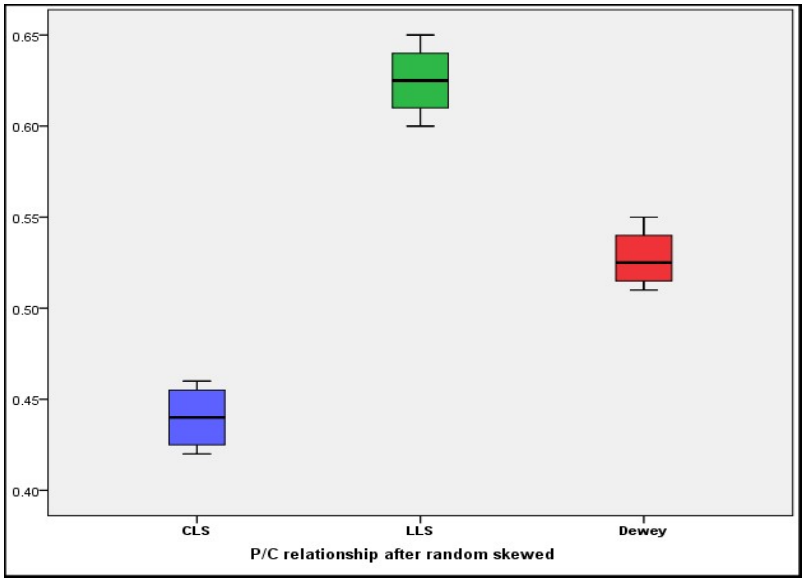


Figure 5-195: Boxplot between CLS, LLS & Dewey schemes when determining the P/C relationships after Random skewed insertion.

p-value between CLS & LLS schemes =	0.015
p-value between CLS & Dewey schemes =	0.014

Figure 5-196: the p-value between the CLS, LLS and Dewey schemes when determining the P/C relationships after Random skewed insertion.

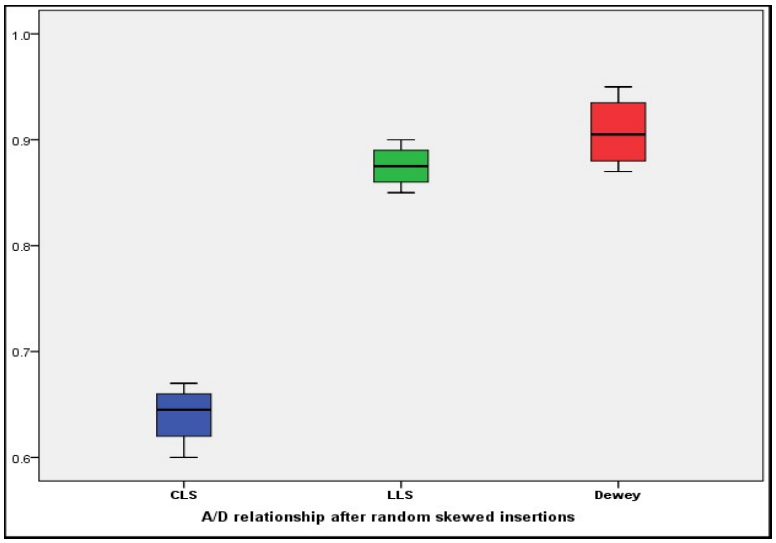


Figure 5-197: Boxplot between CLS, LLS & Dewey schemes when determining the A/D relationships after Random skewed insertion.

p-value between CLS & LLS schemes =	0.017
p-value between CLS & Dewey schemes =	0.017

Figure 5-198: the p-value between the CLS, LLS and Dewey schemes when determining the A/D relationships after Random skewed insertion.

5.3.3 Query Performance for Dynamic Documents:

This experiment was carried out to assess the same query performance that was used for the static document. However, the assessment this time is after insertions. Figures from 5.199 till 5.205 illustrate the results.

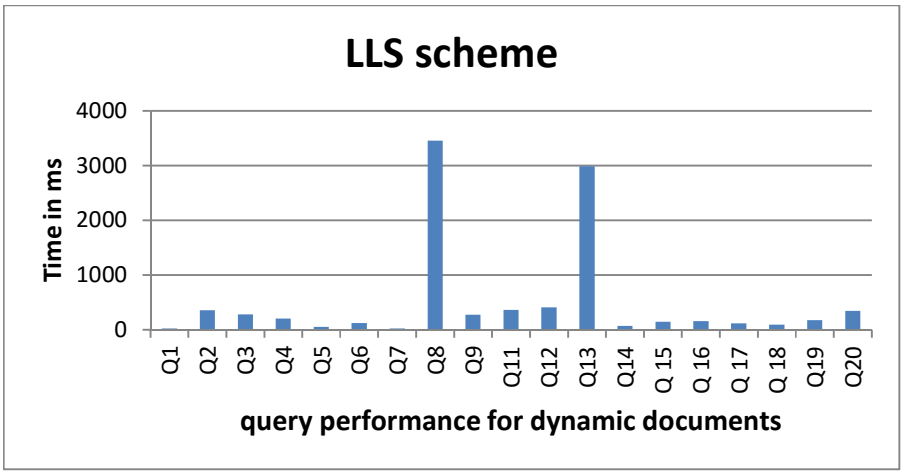


Figure 5-199: query performance for the LLS scheme.

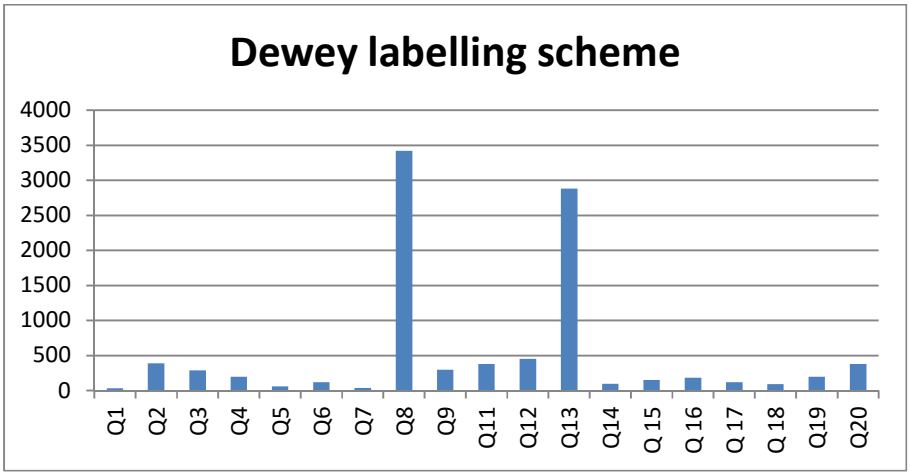


Figure 5-200: query performance for the Dewey labelling scheme.

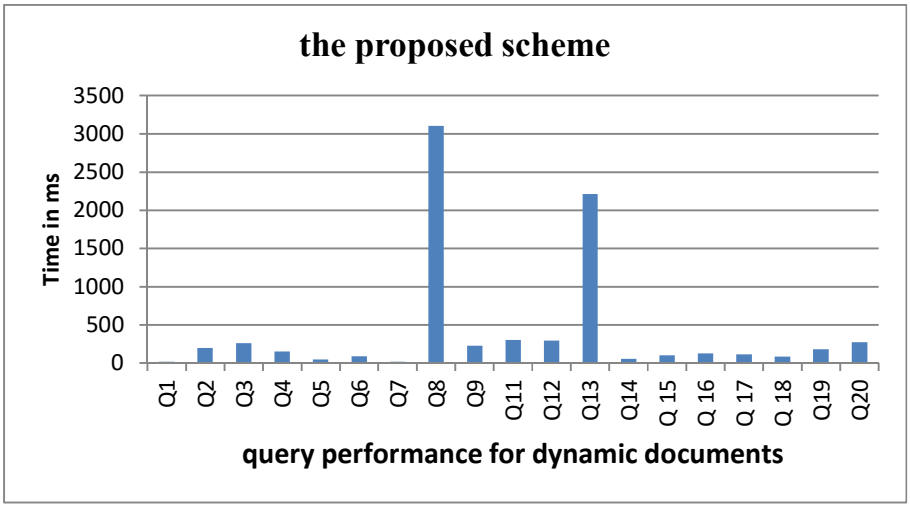


Figure 5-201: query performance for the proposed scheme.

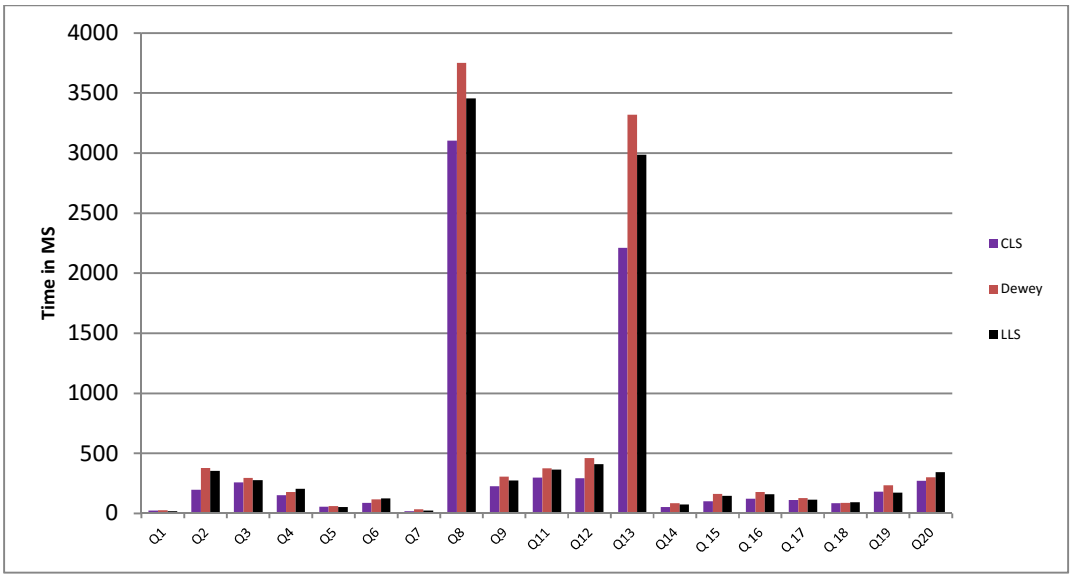


Figure 5-202: query performance for all schemes for dynamic documents.

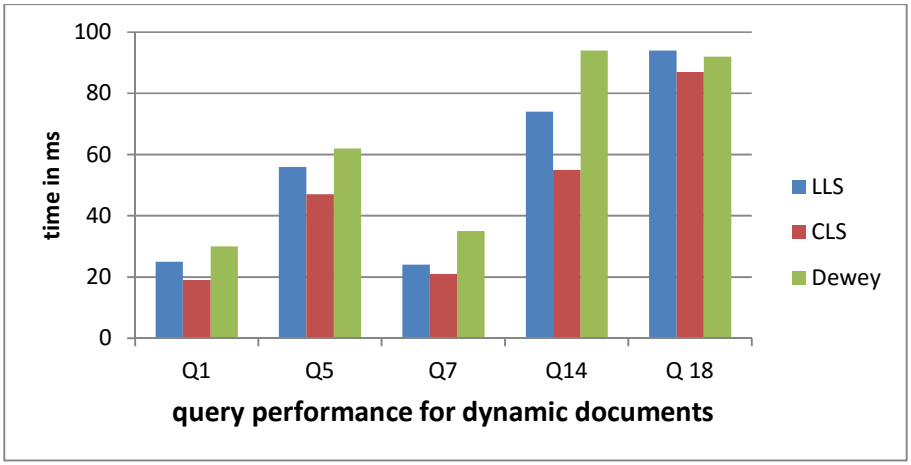


Figure 5-203: query performance for all schemes for group 1.

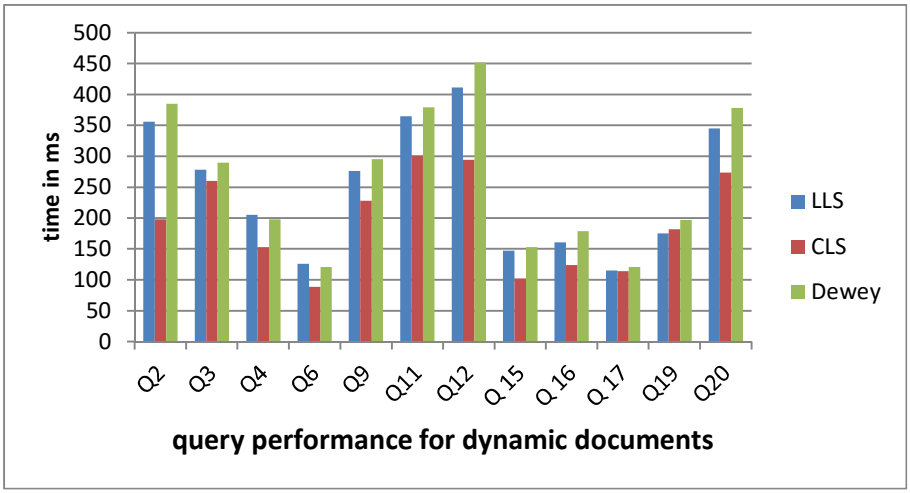


Figure 5-204: query performance for all schemes for group 2.

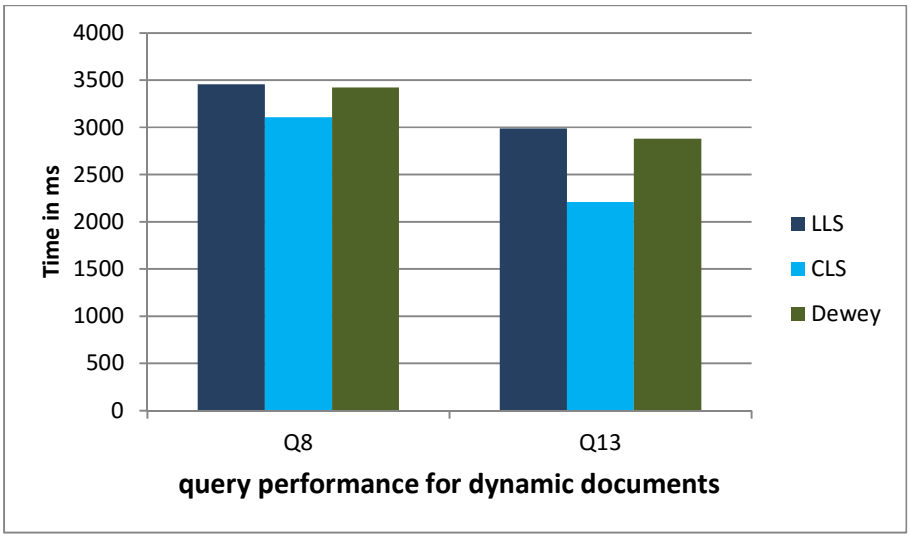


Figure 5-205: query performance for all schemes for group 3.

The proposed scheme presented best spent time and response times in eighteen queries out of twenty. The LLS scheme achieved best result in queries (1, 5).

The query performance on dynamic documents presented a better spent and response time compared to the results of the static documents. However, the LLS scheme achieved better results than the proposed scheme in two queries in the static documents, whereas the LLS scheme has achieved better results just in one query in the dynamic documents. Therefore, the proposed scheme demonstrated more efficiency with dynamic documents than the LLS scheme.

- **Statistical Description of the Results:**

The same nineteen queries used for static documents were used now for dynamic documents. There are two cases where the Dewey scheme showed more efficiency than the proposed scheme which are query one and query five. However, in all the other seventeen queries, the proposed scheme was more efficient than the others. Nineteen Box plots below show the results. P values for these nineteen cases show that all of them rejected the null hypothesis except two cases where the results are far higher than the level of significance – query five and query eighteen. Therefore, in all cases where the null hypothesis was rejected there is strong evidence for claiming that the proposed scheme is faster than the others.

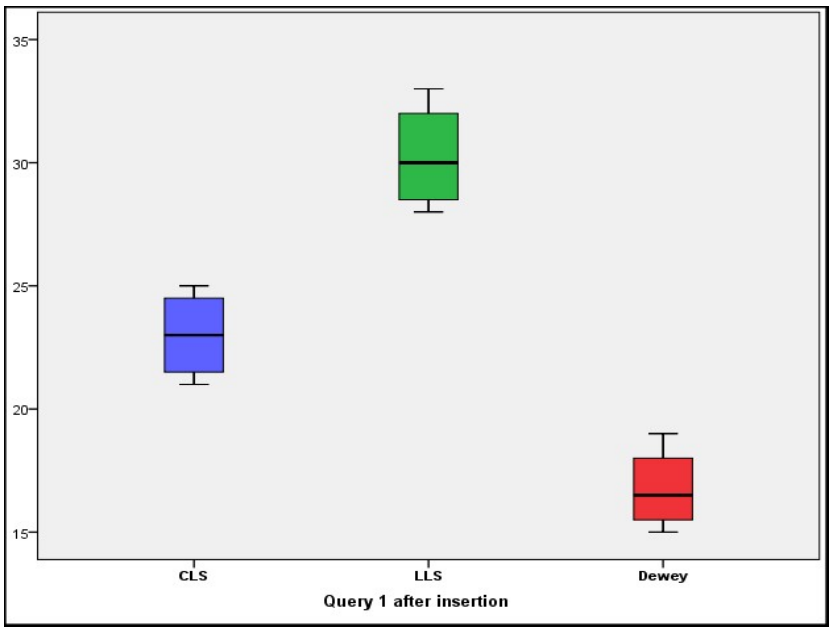


Figure 5-206: Boxplot between the CLS, LLS and Dewey schemes when executing query 1 after insertion.

p-value between CLS & LLS schemes =	0.017
p-value between CLS & Dewey schemes =	0.014

Figure 5-207: the p-value between the CLS, LLS and Dewey schemes when executing query 1 after insertion.

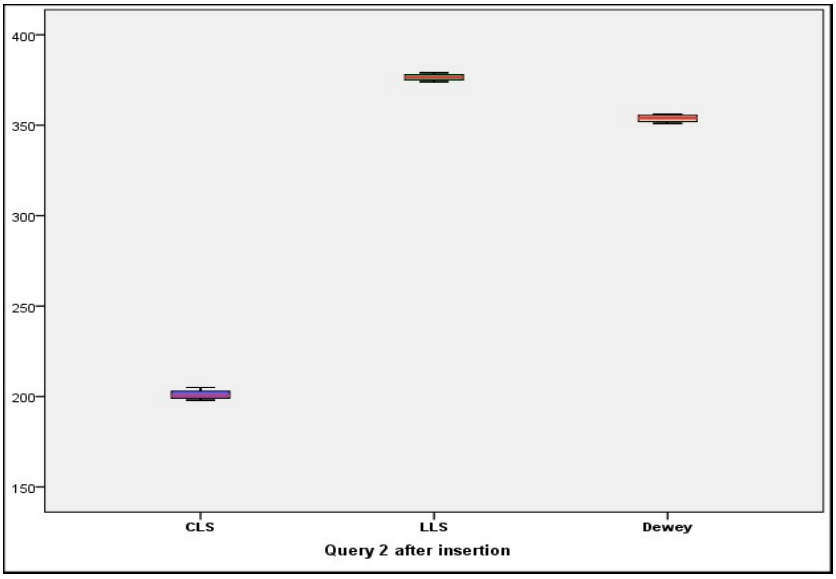


Figure 5-208: Boxplot between the CLS, LLS and Dewey schemes when executing query 2 after insertion.

p-value between CLS & LLS schemes =	0.017
p-value between CLS & Dewey schemes =	0.017

Figure 5-209: the p-value between the CLS, LLS and Dewey schemes when executing query 2 after insertion.

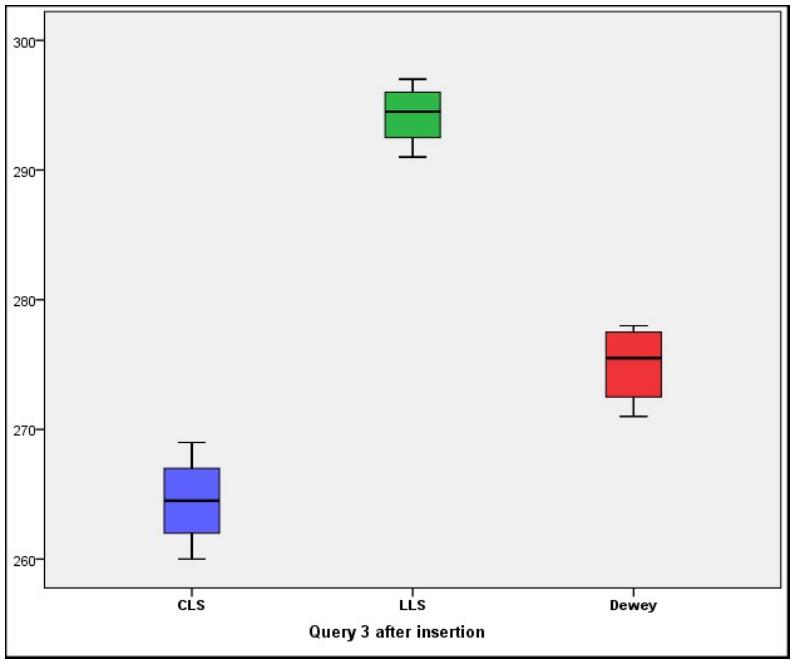


Figure 5-210: Boxplot between the CLS, LLS and Dewey schemes when executing query 3 after insertion.

p-value between CLS & LLS schemes =	0.017
p-value between CLS & Dewey schemes =	0.017

Figure 5-211: the p-value between the CLS, LLS and Dewey schemes when executing query 3 after insertion.

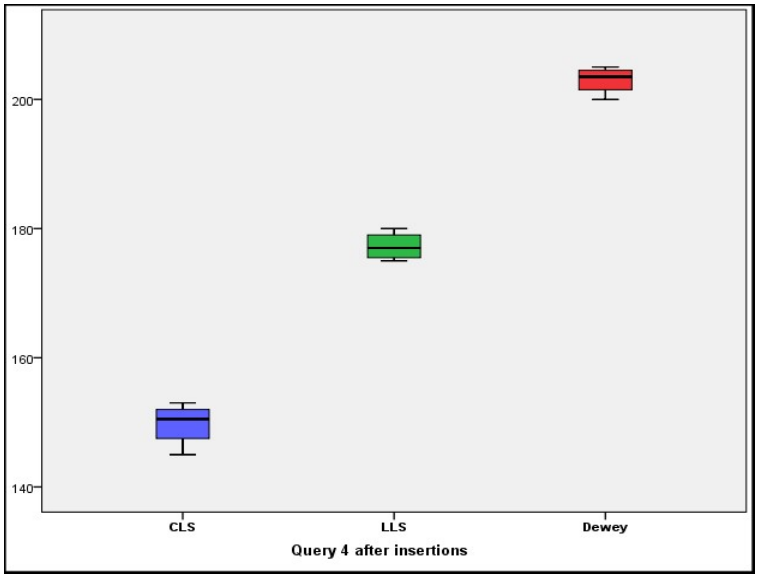


Figure 5-212: Boxplot between the CLS, LLS and Dewey schemes when executing query 4 after insertion.

p-value between CLS & LLS schemes =	0.017
p-value between CLS & Dewey schemes =	0.016

Figure 5-213: show the p-value between the CLS, LLS and Dewey schemes when executing query 4 after insertion.

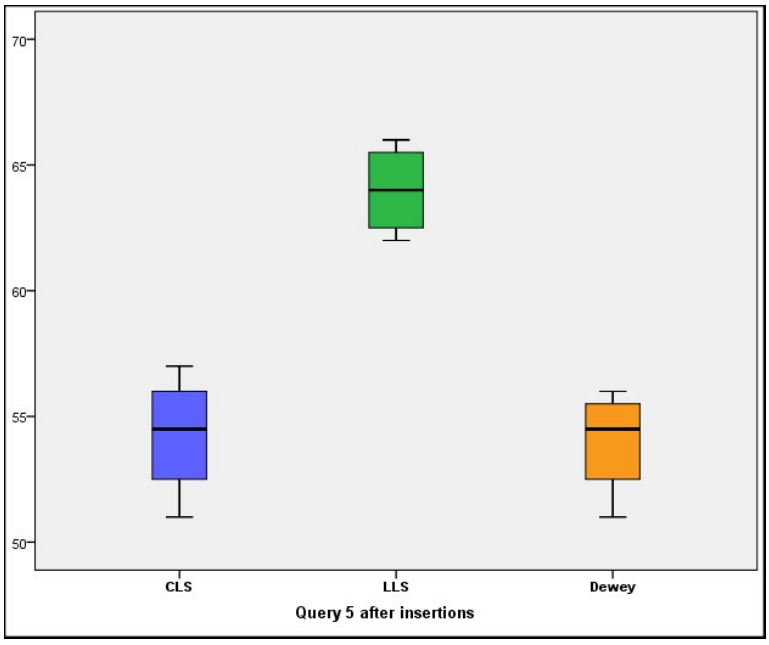


Figure 5-214: Boxplot between the CLS, LLS and Dewey schemes when executing query 5 after insertion.

p-value between CLS & LLS schemes =	0.017
p-value between CLS & Dewey schemes =	0.317

Figure 5-215: the p-value between the CLS, LLS and Dewey schemes when executing query 5 after insertion.

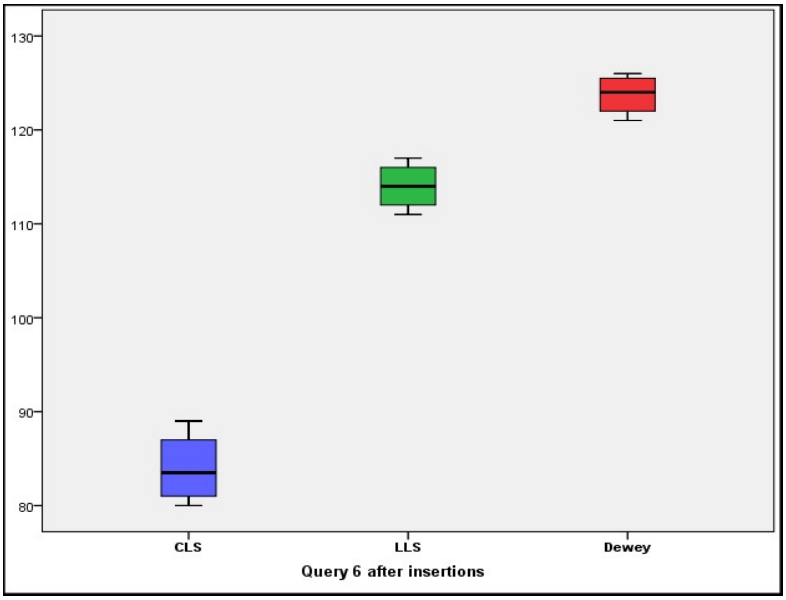


Figure 5-216: Boxplot between the CLS, LLS and Dewey schemes when executing query 6 after insertion.

p-value between CLS & LLS schemes =	0.016
p-value between CLS & Dewey schemes =	0.016

Figure 5-217: the p-value between the CLS, LLS and Dewey schemes when executing query 6 after insertion.

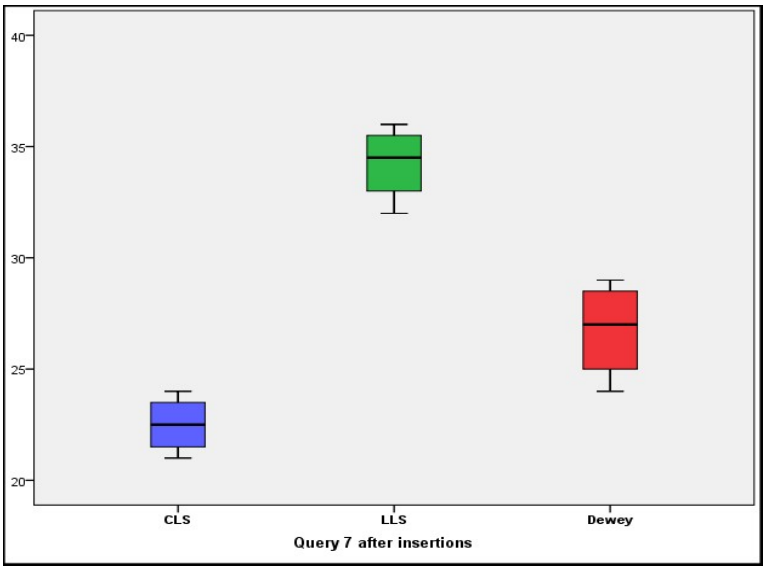


Figure 5-218: Boxplot between the CLS, LLS and Dewey schemes when executing query 7 after insertion.

p-value between CLS & LLS schemes =	0.016
p-value between CLS & Dewey schemes =	0.017

Figure 5-219: the p-value between the CLS, LLS and Dewey schemes when executing query 7 after insertion.

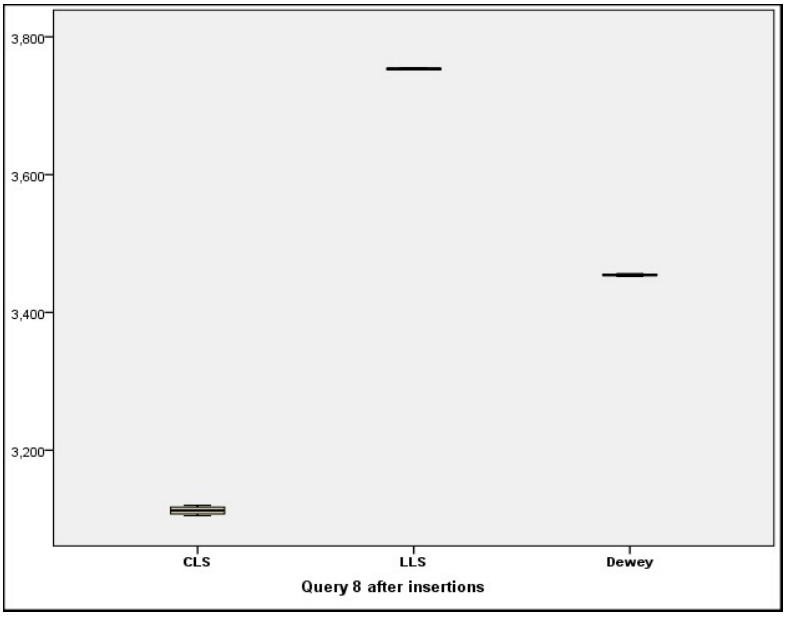


Figure 5-220: Boxplot between the CLS, LLS and Dewey schemes when executing query 8 after insertion.

p-value between CLS & LLS schemes =	0.017
p-value between CLS & Dewey schemes =	0.017

Figure 5-221: the p-value between the CLS, LLS and Dewey schemes when executing query 8 after insertion.

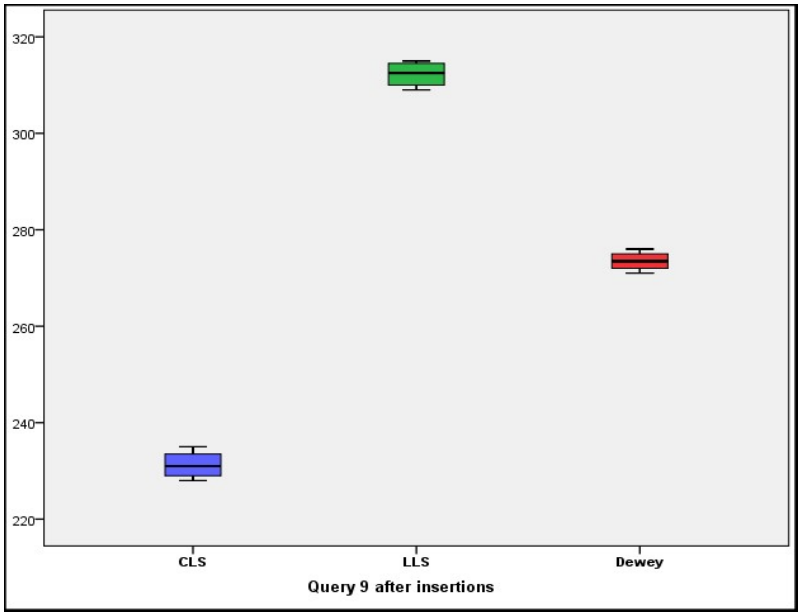


Figure 5-222: Boxplot between the CLS, LLS and Dewey schemes when executing query 9 after insertion.

p-value between CLS & LLS schemes =	0.017
p-value between CLS & Dewey schemes =	0.017

Figure 5-223: the p-value between the CLS, LLS and Dewey schemes when executing query 9 after insertion.

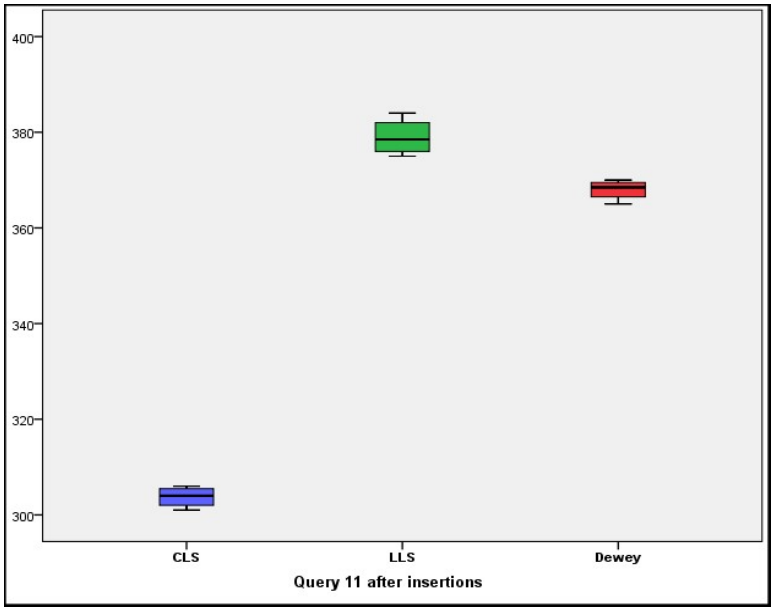


Figure 5-224: Boxplot between the CLS, LLS and Dewey schemes when executing query 11 after insertions.

p-value between CLS & LLS schemes =	0.017
p-value between CLS & Dewey schemes =	0.014

Figure 5-225: the p-value between the CLS, LLS and Dewey schemes when executing query 11 after insertions.

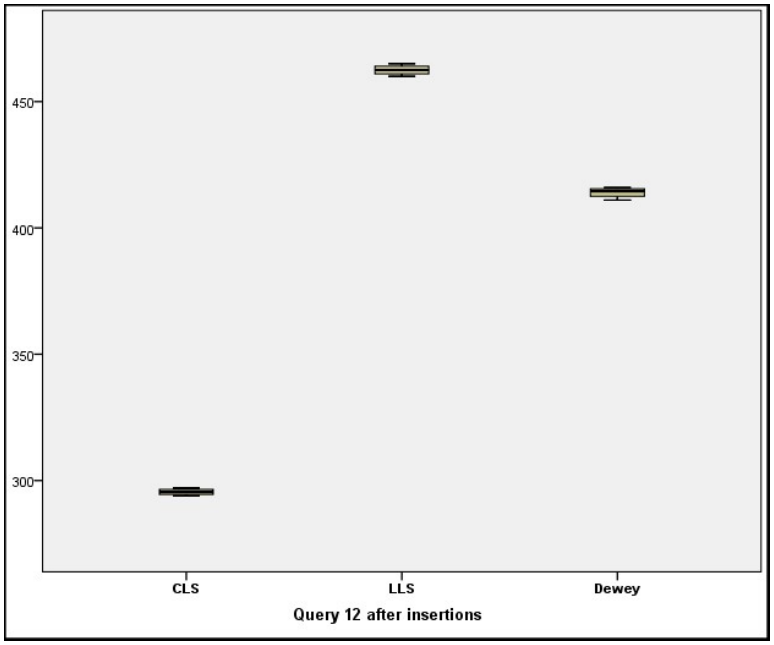


Figure 5-226: Boxplot between the CLS, LLS and Dewey schemes when executing query 12 after insertions.

p-value between CLS & LLS schemes =	0.016
p-value between CLS & Dewey schemes =	0.011

Figure 5-227: the p-value between the CLS, LLS and Dewey schemes when executing query 12 after insertions.

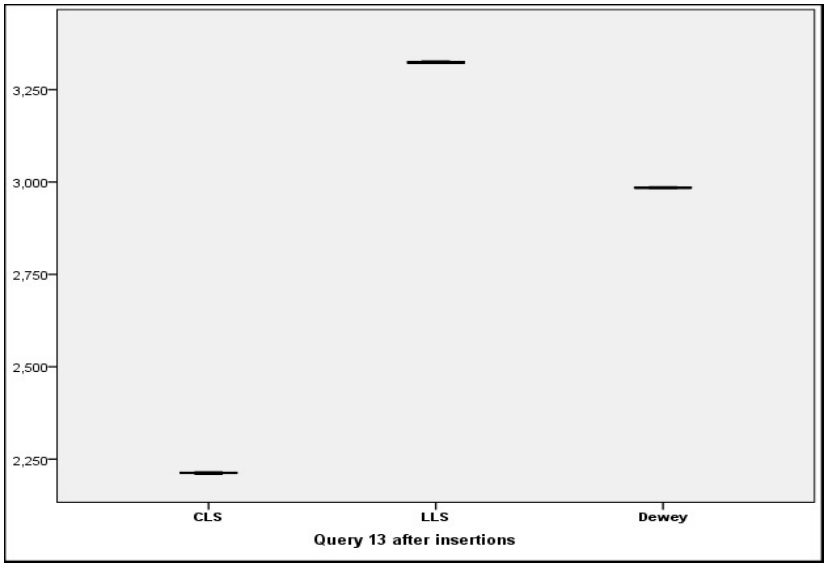


Figure 5-228: Boxplot between the CLS, LLS and Dewey schemes when executing query 13 after insertions.

p-value between CLS & LLS schemes =	0.017
p-value between CLS & Dewey schemes =	0.017

Figure 5-229: the p-value between the CLS, LLS and Dewey schemes when executing query 13 after insertions.

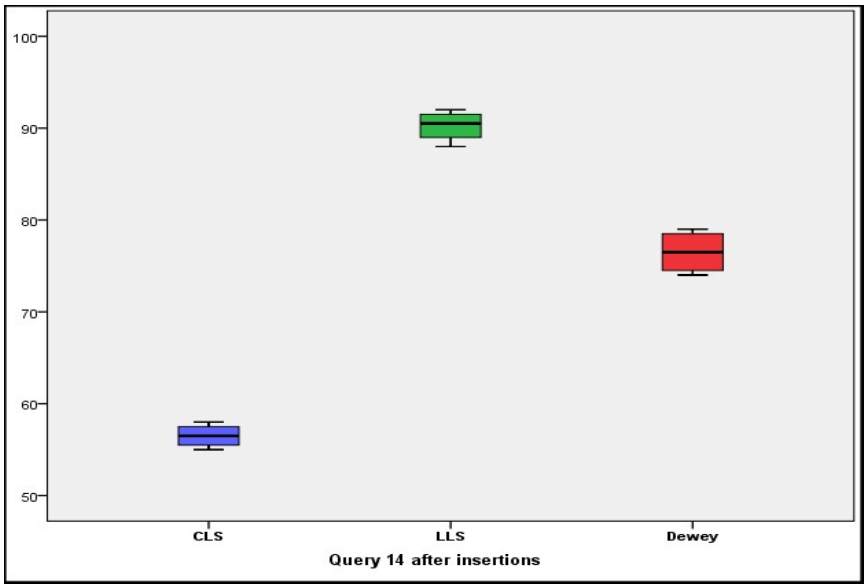


Figure 5-230: Boxplot between the CLS, LLS and Dewey schemes when executing query 14 after insertions.

p-value between CLS & LLS schemes =	0.011
p-value between CLS & Dewey schemes =	0.015

Figure 5-231: the p-value between the CLS, LLS and Dewey schemes when executing query 14 after insertions.

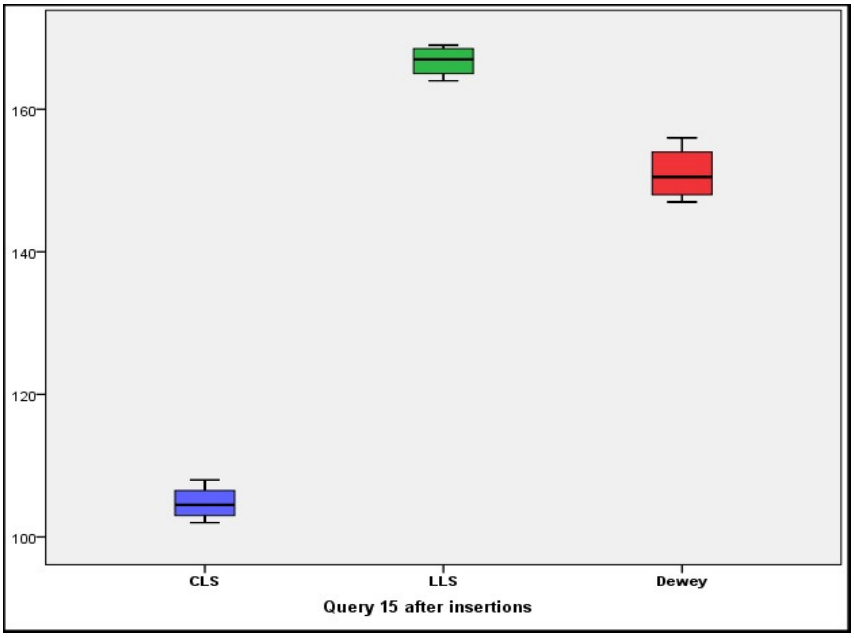


Figure 5-232: Boxplot between the CLS, LLS and Dewey schemes when executing query 15 after insertions.

p-value between CLS & LLS schemes =	0.017
p-value between CLS & Dewey schemes =	0.017

Figure 5-233: the p-value between the CLS, LLS and Dewey schemes when executing query 15 after insertions.

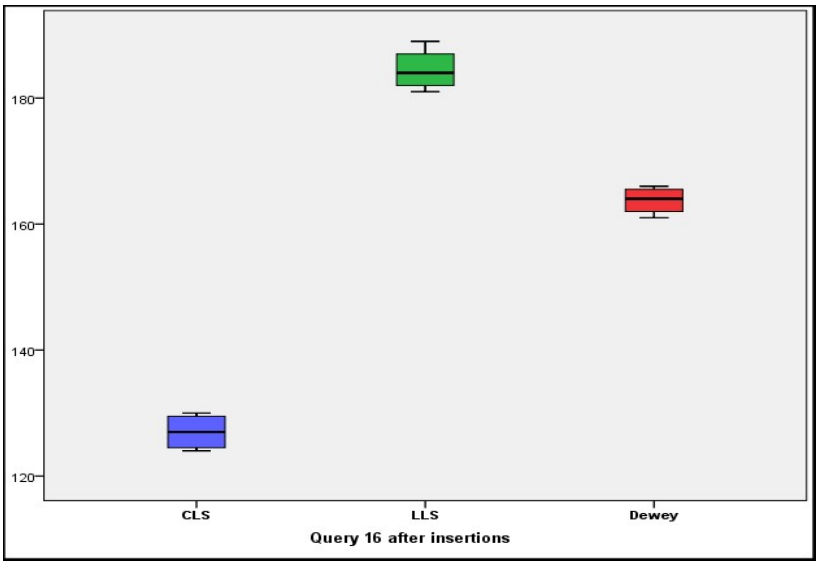


Figure 5-234: Boxplot between the CLS, LLS and Dewey schemes when executing query 16 after insertions.

p-value between CLS & LLS schemes =	0.017
p-value between CLS & Dewey schemes =	0.016

Figure 5-235: the p-value between the CLS, LLS and Dewey schemes when executing query 16 after insertions.

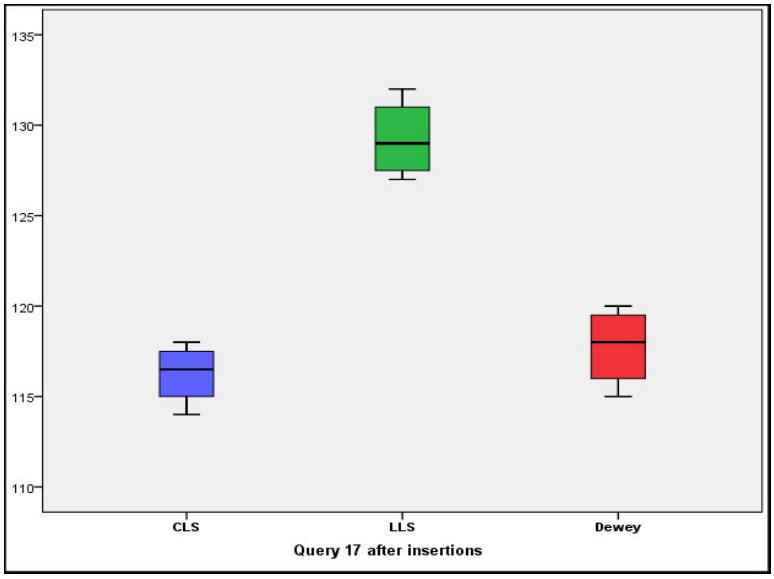


Figure 5-236: Boxplot between the CLS, LLS and Dewey schemes when executing query 17 after insertions.

p-value between CLS & LLS schemes =	0.017
p- between CLS & Dewey schemes =	0.015

Figure 5-237: the p-value between the CLS, LLS and Dewey schemes when executing query 17 after insertions.

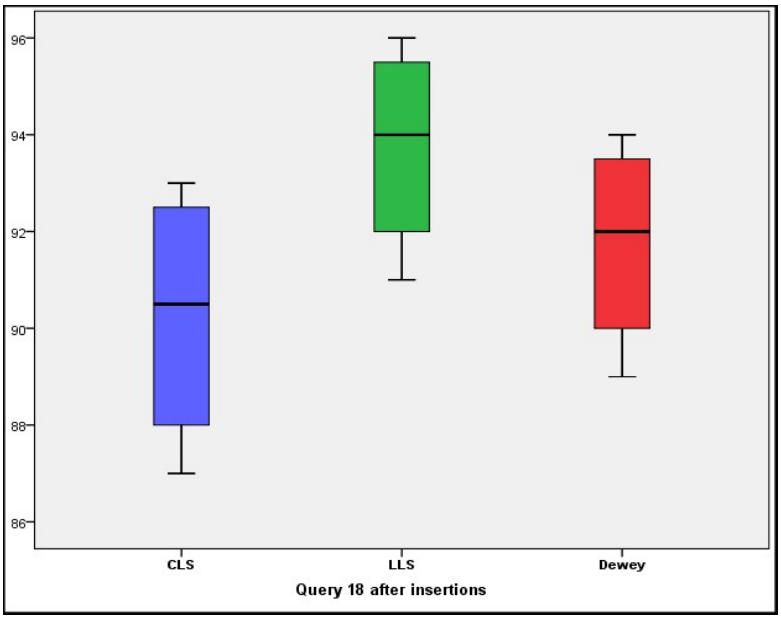


Figure 5-238: Boxplot between the CLS, LLS and Dewey schemes when executing query 18 after insertions.

p-value between CLS & LLS schemes =	0.015
p-value between CLS & Dewey schemes =	0.730

Figure 5-239: the p-value between the CLS, LLS and Dewey schemes when executing query 18 after insertions.

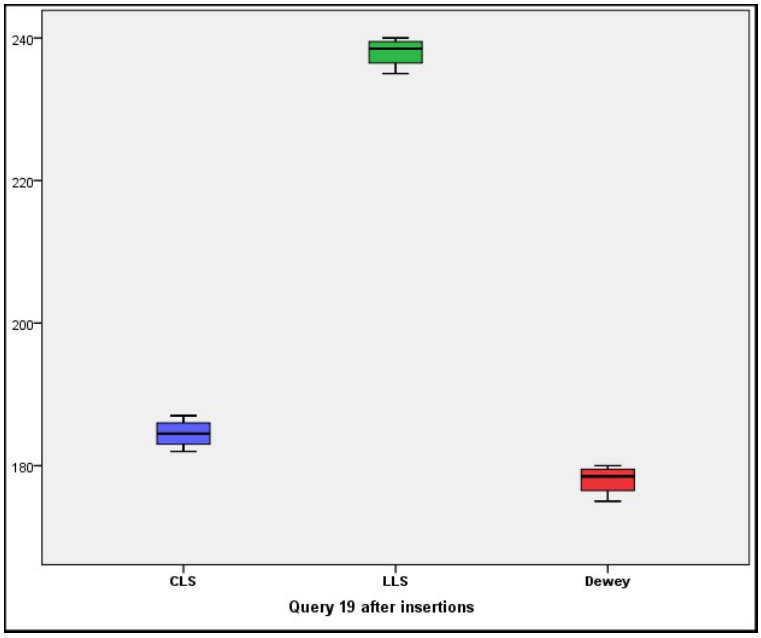


Figure 5-240: Boxplot between the CLS, LLS and Dewey schemes when executing query 19 after insertions.

p-value between CLS & LLS schemes =	0.015
p-value between CLS & Dewey schemes =	0.015

Figure 5-241: the p-value between the CLS, LLS and Dewey schemes when executing query 19 after insertions.

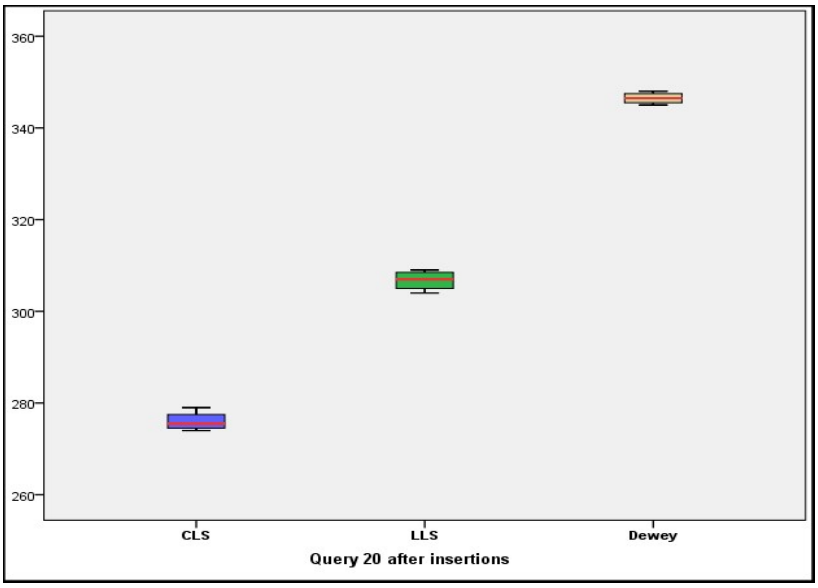


Figure 5-242: Boxplot between the CLS, LLS and Dewey schemes when executing query 20 after insertions.

p-value between CLS & LLS schemes =	0.017
p-value between CLS & Dewey schemes =	0.014

Figure 5-243: the p-value between the CLS, LLS and Dewey schemes when executing query 20 after insertions.

5.4 Summary of the Results

This chapter illustrates the experiments for both static and dynamic documents. The experiments of static documents presented the enhancement of the proposed scheme on the LLS and Dewey schemes in terms of labelling XML documents. Subsequently, the determination of the relationships was assessed in terms of the time needed using the XMark queries.

With respect to dynamic documents, first of all, the ability to handle insertions was tested and as a result the proposed scheme was compared with the LLS scheme. The time for inserting new nodes and the growth of the size of the labels were assessed. Secondly, the relationships were assessed again in order to determine any changes after the insertions. Thirdly, all queries were again executed to evaluate the spent time.

Graphs were used to present and describe the results, and then the statistical significance test was executed, in which the box plot and Wilcoxon rank-sum test were used, to obtain the p-values. All these tasks were used to evaluate the proposed scheme properly and to achieve robust results.

To summarise, the proposed scheme shows efficiency in all aspects, except a few cases when otherwise shown. Briefly, these aspects are the following: labelling nodes of XML documents, determining the identified relationships, and query performance for both static and dynamic XML documents. More details and discussion will be provided in the following chapter.

6. Discussion and Critical Assessment

6.1 Introduction

The aim of this chapter is to discuss the results and the main findings of the testing experiments. This discussion is to link the literature review to the objectives of this research in order to assess whether these results are different from what others have found and whether they have achieved the aim and objectives of this research. The chapter is also intended to identify the effective characteristics of the proposed scheme as well as any limitations. This task is significant as it can be used as a basis for considering the future work for this research.

As chapter 4 states, the determination of whether the proposed scheme is better than the other schemes or not is one of the goals of the experiments of this research. Consequently, the results are discussed and illustrated in that chapter. These discussions will be presented in the following sections:

6.2 Discussion and Assessment of the Experiments

The proposed scheme and other schemes were tested to evaluate certain features such as labelling XML documents, determining the relationships, and querying performance for both static and dynamic XML documents. Discussion and assessment of the results of these experiments are as follows:

6.2.1 Discussion and Assessment of the Labelling XML Documents

There are two main features that were assessed in this experiment; namely the spent time, and the size of the labels of the nodes and clusters. First, the experiment's results of spent time in the previous chapter show that the proposed scheme achieved better performance over the other schemes. Second, with regard to measuring the size of the labels, the proposed scheme was not able to achieve better results than the other schemes. The justification for this failure is due to the method that the proposed scheme used to label the nodes and clusters which allocates two labels for each node. In addition, there is always a trade-off between the query performance and the size of the indexes.

6.2.2 Discussion and Assessment of the Determination of the Relationships

This experiment was intended to check the determination speed of the relationships. The experiment was executed twice in order to assess both static and dynamic XML documents. It was successfully designed and executed, and, therefore, the proposed scheme shows the best results, beating the other schemes in all relationships.

The achieved results from these experiments met the expected goals. In terms of the static XML documents, the results the proposed scheme were faster than the others in all results. In addition, the results of the dynamic documents are also faster than the

others with a few exceptions in which the proposed scheme did not achieve better results.

Since the Prefix labelling schemes, of which the Dewey labelling scheme is one, support the A/D and sibling relationships, the proposed scheme showed better results than the Dewey scheme in this regard. This implies that the proposed scheme can produce better performance than the prefix-based labelling schemes in terms of determination of relationships.

6.2.3 Discussion and Assessment of the Query Efficiency

As explained in the previous chapter, nineteen of the XMark queries were used to test the targeted schemes for both static and dynamic documents. The discussion for both experiments are as follows:

With respect to the static XML documents, the results in general show that the proposed scheme achieved the expected results, with a few exceptions. These exceptions are three queries, namely number 7, 17, and 19.

Regarding the dynamic documents, all the results met the expectations except query number 19 where LLS achieved a better result than the proposed scheme.

To summarise, the proposed scheme has shown an improvement over the other schemes in terms of the efficiency of the query performance. However, since the only dataset was used for these experiments is the XMark Benchmark, the proposed scheme can be tested further using other datasets that can offer more complex queries, since the XMark Benchmark has limitations such as update queries and comparing results from different datasets (Almelibari, 2015).

As far as the query efficiency of other labelling schemes is concerned, prefix schemes such as Dewey, LSDX, and ORDPATH support query processing, however, some researchers have already claimed that prefix labelling schemes need more improvement in terms of query processing (Hou et al., 2001). Moreover, graph schemes handle the path query efficiently as they all return precise and complete answers.

6.2.4 Discussion and Assessment of Inserting new Nodes

The goal of this experiment is to assess the capability of the proposed scheme to handle a variety of insertions. This experiment was only intended for dynamic documents. The measure was the insertion time and the size of the labels after the insertions. There were three different types of insertions used, namely, uniform insertion, ordered skewed, and random skewed. The discussion of the results of this experiment is based on the measures as follows:

Regarding the time for inserting new nodes, the proposed scheme has achieved better results than the other schemes in all times of intended insertions with a few exceptions. As explained in the Methodology chapter, the proposed scheme has taken

advantage of the clustering-based technique in inserting new nodes to simplify many scenarios of the insertions. Therefore, the proposed scheme has beaten the other schemes in most insertions.
Concerning the size of the labels, as the previous chapter illustrated, the results of the proposed scheme in this regard are: the proposed scheme has beaten the other schemes in uniform and ordered skewed insertions with a few exceptions; however, the proposed scheme failed to beat the others in random skewed insertions.
Regarding the functions of inserting new nodes and updates in the other labelling schemes, the prefix schemes were defined as brilliant for fast updates. Intervals offer a space that making the insertion is an easy task by avoiding relabelling. However, this space is limited and therefore this scheme can't cope with intensive insertions.

6.3 Discussion and assessment of the proposed scheme and the main findings

This assessment and discussion is according to the research hypothesis mentioned in the introduction chapter, see below:

H_1: ***Employing a hybrid labelling scheme based on a clustering-based technique improves the efficiency of labelling nodes and query performance of XML data.***

H_0: ***Employing a hybrid labelling scheme based on a clustering-based technique does not improves the efficiency of labelling nodes and query performance of XML data.***

The proposed scheme was developed in order to enhance the XML labelling. This goal was intended to be achieved by mainly reducing the relabelling cases to the lowest possible level. This reduction should not compromise the performance. Moreover, robust and reliable assessment for the proposed scheme would request a comparison for this proposed scheme against the LLS and Dewey schemes on which the implementation and testing were accomplished, in order to perform this comparison.
There are general main qualities that any XML labelling scheme should obtain. Examples of these qualities are query performance, storage space, and labelling time. It should be possible to enhance these qualities (Fennell, 2013).

In general, the results of this research support the research hypothesis. With regard to the performance of the proposed scheme, there are three features that the hypothesis tested. These features are the following; first, the ability of the scheme to insert new nodes efficiently; second, the ability of the scheme to reduce the required relabelling

cases to the lowest possible level; and finally, the ability of the proposed scheme to enhance the performance of the query.

Concerning the ability to insert new nodes into the dynamic documents and determining the relationship efficiently, new node insertions should be achieved very fast to be efficient. The previous chapter illustrated and explained the results of the insertions in detail. Consequently, these results support the first feature of the hypothesis, however, as in some cases such as Random Skewed insertions, the proposed scheme did not achieve best results. Regarding the determination of the relationships, the results of static documents support the first part of the hypothesis strongly, whereas the results of dynamic documents support the same part of the hypothesis, however, as in some cases such as determining the level and the A/D relationships in dynamic documents, the proposed scheme did not beat the other schemes.

Regarding the second feature, this part is assessed by an experiment that calculates the cost of labelling XML documents in terms of time and size. The results as shown in chapter 4 demonstrated that the occupied space in the memory by the proposed scheme is larger than others, which implies that the hypothesis in this regard is rejected. This failure is due to the structure of the labels where the label of the proposed scheme has two parts whereas the others have one part. Regarding the spent time, the proposed scheme shows best results.

The third feature concerns the query performance. This feature is measured by the response time. The assessment was to compare and analyse the results in order to find out how efficient is the proposed scheme against the other schemes. There were 19 queries of the XMark used for testing and the proposed scheme outperformed the other schemes in all queries except query 19. This result confirms that the hypothesis in this respect is accepted.

To summarise, based on the discussion above, the whole hypothesis is accepted, and therefore, the proposed scheme confirmed that it was designed and worked as planned, even though it failed in a few tests. Moreover, the main outcome of these discussions of the results is that the proposed scheme has shown improvement in some aspects such as the function of labelling schemes' determination of the relationships. Yet Cunningham claimed that the time and size of labelling XML data are important functions that are related to efficiency (Cunningham, 2006). Thus, this research has focused on these features in the proposed scheme.

6.4 Strengths and Limitations of the Proposed Labelling Scheme

The proposed scheme has confirmed that it provides better performance than the other tested schemes in most aspects with little exceptions where observed. However, the

technique of the clustering-based scheme is a complicated one as one researcher stated (Almelibari, 2015). This technique has already been used with different utility of existing labelling schemes. The calculations and implementation of this technique is the main limitation of this scheme since the implementation of two labels for each node is a complex process.

Regarding of the significance of this proposed scheme for the real world, the experiments of this research were executed to compare the performance of this scheme against the other two schemes. However, using the XMArk dataset implies that the data can represent the real world as it is well-khown dataset used to test XML schemes. Having said that, performing further experiments in order to test the proposed scheme against a dataset from the real world such as DBLP would be recommended to test this scheme for the real world. Furthermore, the tests for the experiments were repeated ten time as this number has been used in testing number of labelling schemes such as the Labelling Dynamic XML Document and the LLS scheme.

6.5 Clustering technique and improving the query process:

This thesis argues that using the clustering-based techniques can improve query processing performance in the real world and labelling nodes efficiency. This technique has already been used because it has many advantages such as it decreases the re-labelling cases and therefore improves the insertion process. Improve the determination of the relationships (Kaplan et al., 2002). In addition, the clustering technique has two labels for every node, and this feature improves the process of labelling nodes that from the same cluster; and the label of the cluster is used to connect that cluster with the whole tree. This feature helps in determining the relationships between the nodes that form different clusters. It is also simplify many scenarios of inserting new nodes (Xu et al., 2009). However, despite these features of the clustering technique, it has the disadvantage that implementing two labels is a complicated process (Almelibari, 2015). It was also observed that the parent–child clustering technique improves the labelling process, and better than the simple tree with regard to the accuracy and spent time for query processing (Gusfield, 1997)

6.6 Experimental Findings

Regarding the outcomes of the experiment of labelling XML documents, in terms of time consumption, it is noted that the time increases dramatically as the size increases as well; however, the proposed scheme still consumes less time than others. It is also noted that the size of a file can influence the time of labelling as they have positive correlation. Thus, this indicates that the proposed scheme is better than other labelling schemes

7. Conclusion

7.1 Introduction

XML has become a dominant technology for transferring data cross the internet. Thus, the XML database has attracted many researchers for enhancing XML data storage and retrieval. Labelling XML data as an indexing technique for improving query efficiency is a crucial part in XML database management. Labelling XML data is achieved by assigning labels to the nodes of the XML data. Each label is unique and can be used for the nodes randomly, and to construct the relationships between the nodes (Abiteboul et al., 2000; Bosak & Bray, 1999; Elmasri & Navathe, 2016). Thus, the aim of this research was to investigate the existing XML labelling schemes and their limitations in order to address the issue of efficiency of XML query performance. Regarding the types of XML data, there are two types, namely static and dynamic. Handling static data is achieved more easily than dynamic. At present, most XML documents are dynamic ones. Handling dynamic XML data is a complicated issue. One of the limitations that some labelling schemes have is that the size of the labels is too small. This feature is not appropriate for a dynamic labelling scheme as they cannot manage all the relationships between nodes (Q. Li & Moon, 2001). However, dynamic labelling schemes occupy a large storage space, usually inefficient in query performance (Cohen et al., 2010; Duong & Zhang, 2005, 2008). Therefore, many labelling schemes have been proposed. However, none of them is suitable for all users' requirements. The advantage of one of them is a disadvantage for another one. So, every labelling scheme is appropriate for certain cases (Amagasa et al., 2003; Cohen et al., 2010; Eda et al., 2004). This thesis argued that using certain existing labelling schemes to label nodes, and using clustering-based techniques can improve the query and labelling of nodes efficiency. This research has considered the challenges associated with indexing XML data. These challenges are the following: query efficiency, labelling XML data effectively, occupied index storage. Therefore, this research proposed a labelling scheme based on the technique and idea mentioned above. Subsequently, this proposal was achieved through the implementation of this scheme. The evaluation was also through testing the proposed scheme against the other two schemes used in building this scheme, namely the Dewey and LLS schemes. A data analysis was carried out to determine the main findings. Consequently, this chapter summarises the main outcomes that have been discussed in the previous chapters in this thesis, the main limitations of this research, and potential future work.

XML labelling schemes are classified into different categories based on the criteria and aspects in which they are evaluated. First, this category was classified according to the location of the index residence in the computer storage, which can be either in the main

memory as a temporary index or on the hard disk, each of which has its own advantages and disadvantages (Samir Mohammad & Martin, 2009). The former has the advantage of fast access as it avoids the input and output expenses. However, it has the disadvantage that it lacks scalability for large index files. Regarding the hard disk-based one, the advantage of this category is the scalability, and the disadvantage is the slow access. Second, this category was classified by Sans and Lauren (Sans & Laurent, 2008) into two classes, namely interval scheme and prefix-based scheme. This category was enhanced by adding two other schemes, namely multiplication-based schemes and Vector-based labelling schemes (Almelibari, 2015). Third, this category was classified based on the kind of indexed document; these classes are: the index data-centric document, and the index document-centric XML database (Gang et al., 2007). Four, this category was classified based on the structural relationships and into three classes, namely node indexes, path indexes, and sequence indexes (Bruno et al.; Edith et al., 2010). Therefore, the former category based on the structural relationships is the most relevant one for this research as the structure of the index and the relationships between the nodes are the key concerns for this research. Thus, an intensive review and evaluation for the up-to-date existing labelling schemes were conducted in this research. Certain criteria were used for evaluating these schemes to compare the schemes among each other; and many criteria were used for comparing the labelling schemes, such as accuracy, adaptability, precision, scalability, response time, and type of supported queries (Samir Mohammad & Martin, 2009).

This research employed the most relevant of these criteria to assess the structural labelling schemes. These criteria were also selected to assist users in choosing the most appropriate labelling scheme for their requirements. Thus, the appropriate labelling schemes are selected by determining the features that these labelling schemes support. These criteria are the following: first, retrieval power; which means how complete and accurate are the results that a labelling scheme returns. Second, processing complexity, this complexity is measured based on the required structural joins, the processing cost, and the required calculation of the relationships between nodes. Third, the scalability of the labelling scheme increases the query processing time. Fourth, the update cost is one of the key issues that determine the quality of the labelling scheme and is also a main challenge for researchers nowadays. Any labelling scheme must take care of the update cost as the main function for dynamic documents. The quality of the update cost is based on the number of required relabelling cases when the update is occurred.

The classification that this research used to review the existing schemes is the labelling scheme that based on structural relationships. This classification is as follows: first, node indexing scheme. The idea of this kind of scheme is to save values that represent the locations of the nodes in the XML tree structure. These values are used to determine the relationships of a node such as parent, child, sibling, ancestor and descendent. Generally, the most commonly used node schemes are the interval (also known as region) labelling

scheme and the prefix (also known as path) scheme (Al-Badawi et al., 2010; Al-Badawi et al., 2012; Samir Mohammad & Martin, 2009).

The type of prefix-based labelling scheme creates codes that includes two parts which are the prefix part and the actual-code. The prefix part encodes the previous node code which is the parent of the node. The actual-code encodes the order of the node in the tree. Examples of this kind of scheme are the Dewey labelling scheme, Labelling Scheme for Dynamic XML Data (LSDX), GRoup base Prefix (GRP), and The ORDPATH labelling scheme(Duong & Zhang, 2005; D. K. Fisher et al., 2006; Lu & Ling, 2004; O'Neil et al., 2004a). The prefix labelling scheme has advantages such as, supporting the query processing of XML data, and supporting the retrieval and updates of the data. However, these schemes are inefficient for handling extensive labels, and also for handling complex data (Alstrup & Rauhe, 2002; de l'Adour; Yun & Chung, 2008). The second type of node schemes is the interval labelling scheme, this kind of scheme is also known as Region-based Encoding. The mechanism of this scheme is to attach two values to each node which are the startID and the endID. The startID is used to save the node ID for first element or attribute. The endID is used to store the end of the attribute (Wu et al., 2004a).

The most commonly used examples of the interval labelling scheme are the following: the containment labelling scheme (a. k. a. Beg, End), proposed by Zhang et al. (Zhang et al., 2001), and the (Pre, Post) labelling scheme, developed by Dietz (P. F. Dietz, 1982). The third example is the (Order, Size) scheme which was proposed by Li and Moon (Q. Li & Moon, 2001) as an interval scheme. The interval schemes have some advantages such as the space between the start and end in an interval labelling scheme gives good support to creating the relationships between ancestor and descendent, or between parent and child (Cunningham, 2006). Moreover, interval labelling schemes support the XML tree since the tree labels are efficient and not like the non-tree labels. This is because the interval labelling scheme has the ability to obtain the description of the relationships between the child and parent (Fallside & Walmsley, 2004; O'Neil et al., 2004a; Wu et al., 2004b). However, interval labelling schemes cannot support some cases such as dynamic updates which are not always supported (Tatarinov et al., 2002), particularly when the space between any two nodes is not enough for the inserted nodes (Cooper et al., 2001). Consequently, the interval labelling scheme is efficient whenever the inserted nodes are within the assigned space between two nodes (Amato et al., 2003). The second type of labelling scheme is the Graph Labelling scheme, (also known as a summary index). It is used to improve query performance, especially for single path queries, by producing a path summary for XML data to accelerate the process of query evaluation. Nevertheless, it also has the ability to solve twig queries with extra effort of multiple joins processes. There are a number of existing graph schemes such as DataGuide (Goldman & Widom, 1997, 1999); Index Fabric (Cooper et al., 2001); APEX (Chung et al., 2002); D(K)-index (Q. Chen et al., 2003); (F+B)K-index (Kaushik et al.); and F&B-index (Abiteboul et al., 2000; Gang et al., 2007). Graph schemes were classified into different categories and

based on different criteria (Chung et al., 2002; Cooper et al., 2001; Yoshikawa et al., 2001). Examples of these classifications are the following: Polyzotis and Garofalakis (Polyzotis & Garofalakis, 2002) classified the graph schemes according to exactness. This classification divided the schemes into exact schemes and approximate schemes. Examples of exact schemes are: strong Data Guide, 1-index, disk-based F&B-index, Index Fabric, and F&B-index. Examples of approximate schemes are A(K)-index, approximate Data Guide, D(K)-index, and (F+B)K-index (Polyzotis & Garofalakis, 2002). There are other classifications which have been discussed in the literature review, and discussing them in detail here is beyond this chapter's scope. To summarise the graph labelling schemes, some of them are suitable for small XML documents such as 1-index and strong DataGuide; whereas, others are suitable for large XML data such as disk-based F&B-index. Some of these labelling schemes support single path queries such as 1-index, A(K)-index, and D(K)-index. However, the F&B-index and disk-based F&B-index support twig queries. The third type of labelling scheme is the sequence one; this type of labelling scheme changes both XML documents and queries into structure sequences. Sequence labelling schemes save the values and the structures of XML data together into an integrated index structure. This structure is used to evaluate both path and twig queries efficiently, answering a query, making a string sequence that matches the sequence of the data with the query. This technique reduces the need for joins to evaluate twig queries (Wei et al.). Concerning the classification of sequence schemes, they are classified into two categories based on the importance of the tree mapping direction, which are the following: top-down sequence indexing schemes, and bottom-up sequence indexing schemes. Discussing these categories is beyond this chapter's scope, and they were discussed in the literature review chapter.

According to the literature review of this research, many labelling schemes have been developed as solutions for indexing XML data. Each labelling scheme has advantages and disadvantages. The advantage of one scheme is a disadvantage of another scheme. Therefore, this research, as many others in this field, tried to introduce a new approach for labelling XML data that can handle the main issues that the labelling schemes encounter. Consequently, a state-of-the-art labelling scheme was developed. It is a hybrid labelling scheme that takes some of the advantages of these two existing schemes. The clustering-based technique is also used in this proposed scheme to improve the functionalities and reduce the problematic issues to the lowest possible level. Thus, this labelling scheme is based on this mechanism; the Dewey labelling scheme was selected for labelling the clusters due to the advantages of this scheme such as the ability to manage the clusters, and the ease of the updating process. The LLS labelling scheme was selected for labelling the nodes of the clusters. The advantages of this scheme are that it uses numerical data and the size of the labels is steady. The clustering-based technique was used due to the great advantages of this technique. It reduces the required relabelling cases and as a consequence enhances the inserting new nodes processes, with easy

determination of the relationships. Thus, all these issues mentioned above were intended to present into the proposed scheme of this research.

The mechanism of the proposed scheme is based on clustering nodes in order to facilitate the determination of the child–parent and sibling relationships. Moreover, these relationships also support the process of inserting new nodes. The parent–child in this technique assists in handling a small tree rather than the entire XML tree (Kaplan et al., 2002). It was also found that the parent–child clustering technique supports the labelling process, and is more efficient than the simple tree in terms of the accuracy and spent time for query processing (Gusfield, 1997). This mechanism offers advantages such as using two labels for each node as this idea facilitates the process of labelling nodes that share the same cluster; whereas the label of the cluster is used to link that cluster with the entire tree. This feature assists in determining the relationships among nodes that form different clusters (Xu et al., 2009). However, using two labels for each node has the disadvantage that the occupied storage space is larger than for others that use one label. Thus, these advantages support the proposed scheme and improve it in aspects such as query processing, and improving the process of labelling XML documents. Furthermore, the evaluation of the proposed scheme is based on testing it against the LLS scheme and the Dewey labelling scheme. Therefore, the implementation of both the LLS and Dewey labelling schemes was required as they are not available either as open source code or on the shelf software. Thus, both schemes were implemented by the author of the proposed scheme as bespoke software. This implementation was as accurate as possible according to their data models, in order to prove the validity.

Testing the performance of the proposed scheme is the following task. It is performed in order to evaluate this scheme. Therefore, a number of experiments were designed and carried out to test targeted aspects and features of the proposed scheme. These experiments were executed on the proposed scheme as well as the LLS and Dewey labelling schemes in order to compare the results.

These experiments were designed and implemented to test the proposed scheme based on the objectives of this research. Thus, the objectives of these experiments are the following: the main objective of the experiments is to evaluate the performance, scalability, and efficiency of the proposed scheme. Second, evaluating the process of labelling XML documents. The factors used for measuring the scheme are the time required for labelling the document, and the size of the growth of the labels. Third objective is evaluating the query performance by executing different types of queries on the labelled documents before and after insertions. Fourth objective, testing the scheme's ability to handle different kinds of insertions. Based on the objectives of the experiments, the types of the experiments are the following: labelling the XML document: this experiment was carried out to assess the process of labelling the nodes and the clusters of the XML document. Second, time for updates: these updates include inserting new nodes and deleting existing nodes. This experiment is carried out to assess the ability to handle different types of insertion. Third, determining different relationships: this experiment is

for evaluating the time to determine the relationships. Fourth, evaluating the performance of the queries: this experiment is for evaluating the query response time before and after insertions. Different kinds of queries were executed. These queries are used to test different features and aspects. The queries of the XMark benchmark were selected to test the schemes. Regarding the XML dataset and benchmark used for testing the targeted schemes, there are many XML datasets available nowadays. Some of them are from real life datasets (a. k. a. Actual XML datasets) and others are from artificial datasets (a. k. a. Benchmark/standard datasets). These two types of datasets are used to evaluate the XML labelling schemes (Schmidt et al., 2001). A review into the most commonly used XML datasets is performed in order to select the most suitable dataset(s) for testing the proposed scheme. Regarding real life datasets, these datasets are based on real data. Examples of them are: SwissPort, DBLP Computer Science Bibliography, University Courses, Auction Data, NASA, Treebank, Protein Sequence Database, SIGMOD Record, Mondial, and TPC-H Relational Database Benchmark (Schmidt et al., 2001). Concerning the artificial datasets,

these benchmarks were designed for the purpose of evaluating queries. XML benchmarks are categorised into two groups which are, micro benchmarks and application benchmarks. Micro benchmarks are used to evaluate specific parts of a system whereas application benchmarks are used to evaluate the performance of an XML database in general (Barbosa et al., 2002; Kanda Runapongsa, 2006; Mlýnková, 2008; Schmidt et al., 2001). Examples of these benchmarks are: XMark Benchmark, TPox benchmark, Michigan Benchmark, XBench benchmark, and XMack-1 Benchmark. These XML benchmarks are intended for both query processing and storing data (Schmidt et al., 2001). Having reviewed these datasets, the XMark benchmark was selected for the testing experiments. XMark was chosen to test the proposed scheme for the following reasons: first and most importantly, the XMark was used to evaluate the performance of the LLS scheme by the founder. Therefore, it would be appropriate to use the same dataset to evaluate the proposed scheme and compare the results (Samir Mohammad & Martin, 2010). Secondly, this benchmark is widely used to test XML queries and XML database performance (Almelibari, 2015). Furthermore, XMark is a good option since it has many features such as providing a document generator to create documents in different sizes. Thus, users can generate datasets that are appropriate for their requirements (Schmidt et al., 2002). Also, this benchmark provides a binary version of the XMark that can be run as an independent platform on any operating system. XMark provides a broad range of queries – twenty-one in total. These queries are designed to evaluate different aspects of the datasets.

The aim of testing the proposed scheme is to ensure it achieved the objectives. The experiments were performed twenty times for each query and then the average was taken in order to gain as accurate results as possible. This number of executions for each experiment was chosen as a fair number to get an accurate result by calculating the

average of these twenty executions. Other researchers also used the same number for testing similar labelling schemes (Almelibari, 2015). The schemes that were tested are: the proposed scheme, the LLS labelling scheme, and the Dewey labelling scheme. Furthermore, there are particular features that need to be tested, namely the query performance, efficiency of labelling XML documents, efficiency of scalability, and functionality of the proposed scheme. The testing technique is based on running nineteen queries of the XMark dataset to test the target schemes in order to compare the results and ascertain the achievement of the proposed scheme.

The following stage in this thesis is the data analysis and presentation. This process started from the stage of data collection as raw data moving to significant findings. The raw data of the experiments is the selected XML datasets which is the XMark benchmark. This raw data was processed by the algorithm of the proposed labelling scheme. This process labels and manipulates the raw data. The raw data here is XML documents. These raw data include eleven files in different sizes which are generated by the XMark Benchmark. These files were generated in different sizes (1MB, 10MB, 20MB, 30MB, … , 100MB) in order to test the capability of the proposed scheme to manage the scalability and other targeted features. These files were used to measure the time and size for labelling XML documents. In brief, the processing of raw data created indexes that were stored in database files. These indexes are the labels of the XML documents.

The presentation of the results in graphs is an important task to describe the differences between the targeted schemes for the testing. In terms of statistics, the hypothesis test is widely accepted as a process to achieve trusted answers (Currell, 2015). Furthermore, it was suggested that calculating the statistical significance helps in clarifying the differences between the results of two schemes as obtaining the statistical significance. The null hypothesis (H0) is used to measure the statistical significance. The null hypothesis is tested by checking whether the targeted schemes are equal, otherwise the alternative hypothesis is accepted (Benjamini, 1988). Regarding the results of the experiments of the static documents, they presented the improvement of the proposed scheme on the LLS and Dewey schemes in terms of labelling XML documents. Subsequently, the determination of the relationships was evaluated in terms of the time needed using the XMark queries. With regard to dynamic documents, first, the ability to handle insertions was tested and as a result the proposed scheme was compared with the LLS scheme. The time for inserting new nodes and the growth of the size of the labels were assessed. Secondly, the relationships were assessed again in order to determine any changes after the insertions. Thirdly, all queries were again executed to evaluate the spent time. Graphs were used to present and describe the results, and then the statistical significance test was executed, in which the box plot and Wilcoxon rank-sum test were used to obtain the p-values. All these tasks were used to evaluate the proposed scheme

properly and to achieve robust results. To summarise, the proposed scheme shows efficiency in all aspects, except a few cases when otherwise confirmed. Briefly, these aspects are the following: labelling nodes of XML documents, determining the identified relationships, and querying performance for both static and dynamic XML documents.

Concerning the testing of the research hypothesis, the results of this research support the research hypothesis with few exceptions. First, with respect to the performance of the proposed scheme, there are three features that the hypothesis tested. These features are the following: first, the ability of the scheme to insert new nodes efficiently; second, the ability of the scheme to reduce the required relabelling cases to the lowest possible level; and finally, the ability of the proposed scheme to enhance the performance of the query. Second, regarding the ability to insert new nodes into the dynamic documents and determining the relationship efficiently, new node insertions should be achieved very fast to be efficient. The results support the first feature of the hypothesis with exceptions, as in some cases such as Random Skewed insertions, the proposed scheme did not achieve best results. Regarding the determination of the relationships, the results of static documents support the first part of the hypothesis strongly, whereas the results of dynamic documents support the same part of the hypothesis with exceptions, as in the cases of determining the level and the A/D relationships in dynamic documents, the proposed scheme did not beat the other schemes. Regarding the second feature which is the ability of the scheme to reduce the required relabelling cases to the lowest possible level, this part is assessed by an experiment that calculates the cost of labelling XML documents in terms of time and size. The results as shown in chapter 4 demonstrated that the occupied space in the memory by the proposed scheme is larger than the others, which implies that the hypothesis in this regard is rejected. This failure is due to the structure of the labels where the label of the proposed scheme has two parts, whereas the others have one part. Regarding the spent time, the proposed scheme shows best results. The third feature concerns the ability of the proposed scheme to enhance the performance of the query. This feature is measured by the response time. The assessment was to compare and analyse the results in order to find out how efficient the proposed scheme is against the other schemes. There were 19 queries of the XMark used for testing and the proposed scheme outperformed the other schemes in all queries except query 19. This result confirms that the hypothesis in this respect is accepted. To conclude, the whole hypothesis is accepted, and thus, the proposed scheme confirmed that it was designed and worked as planned, even though it failed in a few tests. Moreover, the main issue is that the proposed scheme showed improvement in some aspects such as the function of labelling schemes' determination of the relationships. Yet Cunningham claimed that the time and size of labelling XML data are important functions that are related to efficiency (Cunningham, 2006). Thus, this research has focused on these features in the proposed scheme.

7.2 Potential Future Work

As every labelling scheme, the proposed scheme has some limitations that can be considered further. Using the clustering-based technique is an idea already used with different labelling schemes. The main issue that might be considered for future work in the proposed labelling scheme is the storage space that the labels occupied which as the experiments showed are larger than expected. The idea of using multiple existing labelling schemes in proposing a new hybrid scheme is a very flexible technique. Employing this idea can offer opportunities to utilise different existing labelling schemes and introduce new labelling schemes that can enhance the proposed scheme in this regard. It would be suggested that one of the future directions is to enhance the proposed scheme in terms of avoiding the cases of relabelling by introducing a new labelling scheme with the chance for relabelling cases when the update that occurs is zero.

Further investigation to ascertain a way to reduce the size of the labels would be suggested for future work. As far as the experiments and testing are concerned, further testing can be useful by testing the proposed scheme with more advanced datasets and benchmarks in order to determine whether the proposed scheme can cope with the complex scenarios.

7.3 Summary

This research developed an XML labelling scheme called Clustering-based Labelling scheme. The aim of this scheme is to improve certain functions that many existing labelling schemes suffer from. Such functions are query processing, update processing, and labelling MXL data. This proposed scheme has been tested and evaluated successfully. Thus, the proposed scheme considered the reduction of the relabelling cases to the lowest possible level. It has also shown an improvement in terms of query performance and inserting new nodes process. Therefore, the aim of this research was achieved, despite some limitations as considered in the previous section.

8. References

Abiteboul, S. (1997). *Querying semi-structured data*: Springer.

Abiteboul, S., Buneman, P., & Suciu, D. (2000). *Data on the Web: from relations to semistructured data and XML*: Morgan Kaufmann.

Al-Badawi, M., North, S., & Eaglestone, B. (2007). Classifications, Problems Identification and a New Approach. Sheffield University.

Al-Badawi, M., North, S., & Eaglestone, B. (2010). The 3D XML benchmark (Vol. 1, pp. 13-20).

Al-Badawi, M., Ramadhan, H. A., North, S., & Eaglestorne, B. (2012). A performance evaluation of a new bitmap-based XML processing approach over RDBMS. *International Journal of Web Engineering and Technology, 7*(2), 143-172.

Al-Khalifa, S., Jagadish, H. V., Koudas, N., Patel, J. M., Srivastava, D., & Yuqing, W. (2002, 2002). *Structural joins: a primitive for efficient XML query pattern matching*.

Ali, M. S., Consens, M., Gu, X., Kanza, Y., Rizzolo, F., & Stasiu, R. (2006). *Efficient, effective and flexible XML retrieval using summaries.* Paper presented at the International Workshop of the Initiative for the Evaluation of XML Retrieval.

Almelibari, A. A. (2015). *Labelling Dynamic XML Documents: A GroupBased Approach.* (PhD), Sheffield University, Sheffield Retrieved from http://etheses.whiterose.ac.uk/8729/1/FinalThesis_Almelibari_100203910.pdf

Alstrup, S., & Rauhe, T. (2002). *Improved labeling scheme for ancestor queries.* Paper presented at the Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms.

Amagasa, T., Yoshikawa, M., & Uemura, S. (2003). *QRS: A robust numbering scheme for XML documents.* Paper presented at the Data Engineering, 2003. Proceedings. 19th International Conference on.

Amato, G., Debole, F., Zezula, P., & Rabitti, F. (2003). *YAPI: Yet another path index for XML searching.* Paper presented at the International Conference on Theory and Practice of Digital Libraries.

Anders Berglund, S. B., Don Chamberlin , Mary F. Fernández, Michael Kay, Jonathan Robie, Jérôme Siméon. (2015, 7/9/2015). XML Path Language (XPath) 2.0 (Second Edition). Retrieved 3/9/2016, 2016, from https://www.w3.org/TR/xpath20/

Barbosa, D., & Bonifati, A. (2007). *Database and XML Technologies: 5th International XML Database Symposium, XSym 2007, Vienna, Austria, September 23-24, 2007, Proceedings* (Vol. 4704): Springer Science & Business Media.

Barbosa, D., Mendelzon, A., Keenleyside, J., & Lyons, K. (2002). *ToXgene: a template-based data generator for XML.* Paper presented at the Proceedings of the 2002 ACM SIGMOD international conference on Management of data.

Benjamini, Y. (1988). Opening the Box of a Boxplot. *The American Statistician, 42*(4), 257-262.

Böhme, T., & Rahm, E. (2001). *XMach-1: A benchmark for XML data management.* Paper presented at the Datenbanksysteme in Büro, Technik und Wissenschaft.

Böhme, T., & Rahm, E. (2003). Multi-user evaluation of XML data management systems with XMach-1 *Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web* (pp. 148-159): Springer.

Bosak, J., & Bray, T. (1999). XML and the second-generation Web. *Scientific American, 280*(5), 89-93.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & Yergeau, F. (2006). Extensible markup language (XML). *World Wide Web Consortium Recommendation REC-xml-19980210. http://www. w3. org/TR/1998/REC-xml-19980210, 16*, 16.

Bremer, J.-M., & Gertz, M. (2006). Integrating document and data retrieval based on XML. *The VLDB Journal, 15*(1), 53-83.

Brenes, S., Wu, Y., Van Gucht, D., & Santa Cruz, P. (2008). *Trie Indexes for Efficient XML Query Evaluation.* Paper presented at the WebDB.

Bressan, S., Dobbie, G., Lacroix, Z., Lee, M. L., Li, Y. G., Nambiar, U., & Wadhwa, B. (2001). *XOO7: Applying OO7 Benchmark to XML Query Processing Tools.* Paper presented at

the Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM).
Bruno, N., Koudas, N., & Srivastava, D. (2002). *Holistic twig joins: optimal XML pattern matching*.
Bruno, N., Koudas, N., & Srivastava, D. (2002). *Holistic twig joins: optimal XML pattern matching.* Paper presented at the Proceedings of the 2002 ACM SIGMOD international conference on Management of data.
Carey, M. J., DeWitt, D. J., Kant, C., & Naughton, J. F. (1994). A status report on the OO7 OODBMS benchmarking effort. *ACM SIGPLAN Notices, 29*(10), 414-426.
Catania, B., Maddalena, A., & Vakali, A. (2005). XML document indexes: a classification. *IEEE Internet Computing, 9*(5), 64-71.
Catania, B., Ooi, B. C., Wang, W., & Wang, X. (2005). *Lazy XML updates: laziness as a virtue, of update and structural join efficiency.* Paper presented at the Proceedings of the 2005 ACM SIGMOD international conference on Management of data.
Che, D., Aberer, K., & Özsu, M. T. (2006). Query optimization in XML structured-document databases. *The VLDB Journal, 15*(3), 263-289.
Chen, Q., Lim, A., & Ong, K. W. (2003). *D (k)-index: An adaptive structural summary for graph-structured data.* Paper presented at the Proceedings of the 2003 ACM SIGMOD international conference on Management of data.
Chen, Y., Mihaila, G., Bordawekar, R., & Padmanabhan, S. (2005). *L-Tree: a dynamic labeling structure for ordered XML data.* Paper presented at the Current Trends in Database Technology-EDBT 2004 Workshops.
Chung, C.-W., Min, J.-K., & Shim, K. (2002). *APEX: An adaptive path index for XML data.* Paper presented at the Proceedings of the 2002 ACM SIGMOD international conference on Management of data.
Cohen, E., Kaplan, H., & Milo, T. (2010). Labeling dynamic XML trees. *SIAM Journal on Computing, 39*(5), 2048-2074.
Connolly, T. M., & Begg, C. (2015). *Database Systems: practical approach to design, implementation, and management*. Harlow: Pearson Education Limited.
Connolly, T. M., & Begg, C. E. (2010). *Database systems: a practical approach to design, implementation, and management*. Boston, Mass: Addison-Wesley.
Cooper, B. F., Sample, N., Franklin, M. J., Hjaltason, G. R., & Shadmon, M. (2001). *A fast index for semistructured data.* Paper presented at the VLDB.
Cunningham, L. A. (2006). Language, Deals, and Standards: The Future of XML Contracts. *Wash. UL Rev., 84*, 313.
Currell, G. (2015). *Scientific data analysis*. Oxford: Oxford University Press.
de l'Adour, I. d. P. A Persistent Labelling Scheme for XML and tree Databases.
Dietz, P. (1982). *Maintaining order in a linked list*.
Dietz, P. F. (1982). *Maintaining order in a linked list.* Paper presented at the Proceedings of the fourteenth annual ACM symposium on Theory of computing.
Duong, M., & Zhang, Y. (2005). *LSDX: a new labelling scheme for dynamically updating XML data.* Paper presented at the Proceedings of the 16th Australasian database conference-Volume 39.
Duong, M., & Zhang, Y. (2008). *Dynamic labelling scheme for XML data processing.* Paper presented at the OTM Confederated International Conferences" On the Move to Meaningful Internet Systems".
Eda, T., Sakurai, Y., Amagasa, T., Yoshikawa, M., Uemura, S., & Honishi, T. (2004). *Dynamic range labeling for XML trees.* Paper presented at the International Conference on Extending Database Technology.
Edith, C., Haim, K., & Tova, M. (2010). LABELING DYNAMIC XML TREES. *SIAM Journal on Computing, 39*(5), 2048.
El-Aziz, A. A., & Kannan, A. (2012). *Storing XML documents and XML policies in Relational Databases.* Paper presented at the Proceedings of the 2nd International Conference on Computer Communication and Informatics (ICCCI).
El-Sayed, M., Dimitrova, K., & Rundensteiner, E. (2005, 2003). *Efficiently supporting order in XML query processing*.

El-Sayed, M., Dimitrova, K., & Rundensteiner, E. A. (2005). Efficiently supporting order in XML query processing. *Data & Knowledge Engineering, 54*(3), 355-390.
Elmasri, R., & Navathe, S. (2007). *Fundamentals of database systems*. Boston, Mass: Pearson Addison-Wesley.
Elmasri, R., & Navathe, S. (2016). *Fundamentals of database systems* (Vol. Global;Seventh;). Upper Saddle River: Pearson.
Fallside, D. C., & Walmsley, P. (2004). XML schema part 0: primer second edition. *W3C recommendation, 16*.
Fennell, P. (2013). Extremes of XML. *XML LONDON 2013*.
Fisher, D., Lam, F., Shui, W., & Wong, R. (2003). *Efficient ordering for XML data*.
Fisher, D. K., Lam, F., Shui, W. M., & Wong, R. K. (2006). *Dynamic labeling schemes for ordered XML based on type information.* Paper presented at the Proceedings of the 17th Australasian Database Conference-Volume 49.
Frank, T., Apel, M., & Schaebec, H. (2003). *Web integration of gOcad using a 3d-Xml application server.* Paper presented at the Proceedings of the 23rd Gocad Meeting, Nancy, France, June.
Gang, G., Chirkova, R., & Chirkova, R. (2007). Efficiently Querying Large XML Data Repositories: A Survey. *IEEE Transactions on Knowledge and Data Engineering, 19*(10), 1381-1403.
Garcia-Molina, H. (2008). *Database systems: the complete book*: Pearson Education India.
Goldman, R., & Widom, J. (1997). Dataguides: Enabling query formulation and optimization in semistructured databases.
Goldman, R., & Widom, J. (1999). *Approximate dataguides.* Paper presented at the Proceedings of the Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats.
Gou, G., & Chirkova, R. (2007). Efficiently querying large XML data repositories: A survey. *IEEE Transactions on Knowledge and Data Engineering, 19*(10), 1381-1403.
Gray, D. E. (2013). *Doing research in the real world*: Sage.
Guoren, W., Mengchi, L., Yu, J. X., Bing, S., Ge, Y., Jianhua, L., & Hongjun, L. (2003). *Effective schema-based XML query optimization techniques*.
Guoren, W., Mengchi, L., Yu, J. X., Bing, S., Ge, Y., Jianhua, L., & Hongjun, L. (2003, 2003). *Effective schema-based XML query optimization techniques*.
Gusfield, D. (1997). *Algorithms on strings, trees and sequences: computer science and computational biology*: Cambridge university press.
Haifeng, J., Hongjun, L., Wei, W., & Ooi, B. C. (2003). *XR-tree: indexing XML data for efficient structural joins*.
Härder, T., Haustein, M., Mathis, C., & Wagner, M. (2007). Node labeling schemes for dynamic XML documents reconsidered. *Data & Knowledge Engineering, 60*(1), 126-149.
Harold, E. R., & Means, W. S. (2004). *XML in a nutshell* (Vol. 3rd). Beijing;Farnham;: O'Reilly.
Harold, E. R., Means, W. S., & Udemadu, K. (2004). *XML in a Nutshell* (Vol. 8): O'reilly Sebastopol, CA.
Hou, J., Zhang, Y., & Kambayashi, Y. (2001). *Object-oriented representation for XML data.* Paper presented at the Cooperative Database Systems for Advanced Applications, 2001. CODAS 2001. The Proceedings of the Third International Symposium on.
IG, W. C. D. (2005). Document Object Model (DOM). Philippe Le Hégaret. Retrieved 6/92016, 2016, from https://www.w3.org/DOM/
Jayanthi, P., & Tamilarasi, A. (2011). A new way of generating reusable index labels for dynamic XML. *International Journal of Computer Science and Information Technologies, 2*(2), 602-606.
Jiang, Y., He, X., Lin, F., & Jia, W. (2011). An encoding and labeling scheme based on continued fraction for dynamic XML. *Journal of Software, 6*(10), 2043-2049.
Johnson, J. R., Miller, A., Khan, L., & Thuraisingham, B. (2012). *Extracting semantic information structures from free text law enforcement data.* Paper presented at the Intelligence and Security Informatics (ISI), 2012 IEEE International Conference on.

Kanda Runapongsa, J. M. P., H. V. Jagadish, Yun Chen, Shurug Al-Khalifa (2006). The Michigan Benchmark. Retrieved 12/6/2014, 2016, from http://dbgroup.eecs.umich.edu/mbench/description.html

Kaplan, H., Milo, T., & Shabo, R. (2002). *A comparison of labeling schemes for ancestor queries.* Paper presented at the Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms.

Kaushik, R., Bohannon, P., Naughton, J., & Korth, H. (2002). *Covering indexes for branching path queries*.

Kaushik, R., Bohannon, P., Naughton, J. F., & Korth, H. F. (2002). *Covering indexes for branching path queries.* Paper presented at the Proceedings of the 2002 ACM SIGMOD international conference on Management of data.

Kaushik, R., Bohannon, P., Naughton, J. F., & Shenoy, P. (2002). *Updates for structure indexes.* Paper presented at the Proceedings of the 28th international conference on Very Large Data Bases.

Kaushik, R., Shenoy, P., Bohannon, P., & Gudes, E. (2002). *Exploiting local similarity for indexing paths in graph-structured data.* Paper presented at the Data Engineering, 2002. Proceedings. 18th International Conference on.

Kha, D. D., Yoshikawa, M., & Uemura, S. (2001). *An XML indexing structure with relative region coordinate*.

Klaib, A., & Lu, J. (2013). *Development of Database Structure and Indexing Technique for the Wireless Response System.* Paper presented at the INFOCOMP 2013, The Third International Conference on Advanced Communications and Computation.

Klaib, A., & Lu, J. (2014). *Investigation into Indexing XML Data Techniques.* Paper presented at the Proceedings on the International Conference on Internet Computing (ICOMP).

Lee, D., & Chu, W. W. (2000). Comparative analysis of six XML schema languages. *Sigmod Record, 29*(3), 76-87.

Li, C., & Ling, T. W. (2005). *An improved prefix labeling scheme: a binary string approach for dynamic ordered XML.* Paper presented at the International Conference on Database Systems for Advanced Applications.

Li, C., Ling, T. W., & Hu, M. (2006). *Reuse or never reuse the deleted labels in XML query processing based on labeling schemes.* Paper presented at the International Conference on Database Systems for Advanced Applications.

Li, C., Ling, T. W., Lu, J., & Yu, T. (2005). *On reducing redundancy and improving efficiency of XML labeling schemes.* Paper presented at the Proceedings of the 14th ACM international conference on Information and knowledge management.

Li, Q., & Moon, B. (2001). *Indexing and querying XML data for regular path expressions.* Paper presented at the VLDB.

Lindner, W., Mesiti, M., Türker, C., Tzitzikas, Y., & Vakali, A. (2004). *Current Trends in Database Technology-EDBT 2004 Workshops: EDBT 2004 Workshops PhD, DataX, PIM, P2P&DB, and ClustWeb, Heraklion, Crete, Greece, March 14-18, 2004, Revised Selected Papers* (Vol. 3268): Springer.

Liu, J., Ma, Z., & Yan, L. (2009). *Efficient processing of twig pattern matching in fuzzy XML.* Paper presented at the Proceedings of the 18th ACM conference on Information and knowledge management.

Lu, J., & Ling, T. W. (2004). Labeling and querying dynamic XML trees *Advanced Web Technologies and Applications* (pp. 180-189): Springer.

Lu, J., Ling, T. W., Chan, C.-Y., & Chen, T. (2005). *From region encoding to extended dewey: On efficient processing of XML twig pattern matching.* Paper presented at the Proceedings of the 31st international conference on Very large data bases.

Luk, R. W. P., Leong, H. V., Dillon, T. S., Chan, A. T. S., Croft, W. B., & Allan, J. (2002). A survey in indexing and searching XML documents. *Journal of the American Society for Information Science and Technology, 53*(6), 415-437.

Ma, J., Liu, W., Hunter, A., & Zhang, W. (2010). Soft computing in XML data management.

Maghaydah, M., & Orgun, M. A. (2006). *Labeling XML nodes in RDBMS*.

McGill, R., Tukey, J. W., & Larsen, W. A. (1978). Variations of box plots. *The American Statistician, 32*(1), 12-16.

Meng, X., Jiang, Y., Chen, Y., & Wang, H. (2004). *XSeq: an indexing infrastructure for tree pattern queries.* Paper presented at the Proceedings of the 2004 ACM SIGMOD international conference on Management of data.
Meuss, H., & Strohmaier, C. M. (1999). *Improving Index Structures for Structured Document Retrieval.* Paper presented at the BCS-IRSG Annual Colloquium on IR Research.
Milo, T., & Suciu, D. (1999). Index structures for path expressions *Database Theory—ICDT'99* (pp. 277-295): Springer.
Mlýnková, I. (2008). Xml benchmarking: Limitations and opportunities: Technical Report, Department of Software Engineering, Charles University, Czech Republic.
Mohammad, S. (2011). *INDEX STRUCTURES FOR XML DATABASES.* (Dissertation/Thesis). Retrieved from http://hud.summon.serialssolutions.com/2.0.0/link/0/eLvHCXMwY2AwNtIz0EUrE4wSDQ3SjBOB8W1gapCcZppqlJRkZphommKYZJaYmJiKcjATUgHvJsTAIJonymDr5hri7KELvQAiPgWY25JT40FHIxdCmNAxj3iYiuTEeENgUznezBiYSMUYeBNB68XzSsD7yIIAzigvJg

www.summon.com
Mohammad, S., & Martin, P. (2009). XML structural indexes: Citeseer.
Mohammad, S., & Martin, P. (2010). *LLS: level-based labeling scheme for XML databases.* Paper presented at the Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research.
Mohammad, S., Martin, P., & Powley, W. (2011). *Relational universal index structure for evaluating XML twig queries*.
Mohammad, S. A. (2011). *Index structures for XML Databases.* Queen's University.
Murata, M., Laurent, S. S., & Kohn, D. (2000). XML media types.
Nicola, M., Kogan, I., & Schiefer, B. (2007). *An XML transaction processing benchmark.* Paper presented at the Proceedings of the 2007 ACM SIGMOD international conference on Management of data.
Nuzzo, R. L. (2016). The Box Plots Alternative for Visualizing Quantitative Data. *PM&R, 8*(3), 268-272. doi: 10.1016/j.pmrj.2016.02.001
O'Neil, P., O'Neil, E., Pal, S., Cseri, I., Schaller, G., & Westbury, N. (2004). *ORDPATHs: insert-friendly XML node labels*.
O'Neil, P., O'Neil, E., Pal, S., Cseri, I., Schaller, G., & Westbury, N. (2004a). *ORDPATHs: insert-friendly XML node labels.* Paper presented at the Proceedings of the 2004 ACM SIGMOD international conference on Management of data.
O'Neil, P., O'Neil, E., Pal, S., Cseri, I., Schaller, G., & Westbury, N. (2004b, 2004). *ORDPATHs: insert-friendly XML node labels*.
Papamarkos, G., Zamboulis, L., & Poulovassilis, A. (2008). Xml Databases.
Polyzotis, N., & Garofalakis, M. (2002). *Structure and value synopses for XML data graphs.* Paper presented at the Proceedings of the 28th international conference on Very Large Data Bases.
Prasad, K. H., & Kumar, P. S. (2005). *Efficient indexing and querying of XML data using modified Prüfer sequences.* Paper presented at the Proceedings of the 14th ACM international conference on Information and knowledge management.
Rao, P., & Moon, B. (2004). *PRIX: Indexing and querying XML using prufer sequences.* Paper presented at the Data Engineering, 2004. Proceedings. 20th International Conference on.
Rao, P., & Moon, B. (2006). Sequencing XML data and query twigs for fast pattern matching. *ACM Transactions on Database Systems (TODS), 31*(1), 299-345.
Ray, E. T. (2001). Learning XML: Creating self-describing data. *IEEE Micro, 21*(2), 95. doi: 10.1109/MM.2001.918031
Ray, E. T. (2003). *learning XML*: " O'Reilly Media, Inc.".
Rousseeuw, P. J., Ruts, I., & Tukey, J. W. (1999). The bagplot: a bivariate boxplot. *The American Statistician, 53*(4), 382-387.
Runapongsa, K., Patel, J. M., Jagadish, H., Chen, Y., & Al-Khalifa, S. (2006). The Michigan benchmark: towards XML query performance diagnostics. *Information Systems, 31*(2), 73-97.

Sans, V., & Laurent, D. (2008). Prefix based numbering schemes for XML: techniques, applications and performances. *Proceedings of the VLDB Endowment, 1*(2), 1564-1573.
Schmidt, A., Waas, F., Kersten, M., Carey, M. J., Manolescu, I., & Busse, R. (2002). *XMark: A benchmark for XML data management.* Paper presented at the Proceedings of the 28th international conference on Very Large Data Bases.
Schmidt, A., Waas, F., Kersten, M., Florescu, D., Carey, M., Manolescu, I., & Busse, R. (2001). Why and how to benchmark XML databases. *ACM SIGMOD Record, 30*(3), 27-32. doi: 10.1145/603867.603872
Scott, M. L., & SCOTT, M. L. (1998). *Dewey decimal classification*: Libraries Unlimited.
Silberstein, A., He, H., Yi, K., & Yang, J. (2005). *BOXes: Efficient maintenance of order-based labeling for dynamic XML data.* Paper presented at the Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on.
Steegmans, B. (2005). *XML for DB2 Information Integration*: Books24x7. com.
Stein, B. (2007). *Principles of hash-based text retrieval.* Paper presented at the Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval.
Su-Cheng, H., & Chien-Sing, L. (2008). *Evolution of Structural Path Indexing Techniques in XML Databases: A Survey and Open Discussion*.
Sun, L., & Wang, H. (2012). A purpose-based access control in native XML databases. *Concurrency and Computation: Practice and Experience, 24*(10), 1154-1166.
Tatarinov, I., Viglas, S. D., Beyer, K., Shanmugasundaram, J., Shekita, E., & Zhang, C. (2002). Storing and querying ordered XML using a relational database system. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 204-215.
Tidwell, D. (2002, 07 August 2002). Introduction to XML. Retrieved 11/8/2016, 2016, from http://www.ibm.com/developerworks/xml/tutorials/xmlintro/xmlintro.html
Vakali, A., Catania, B., & Maddalena, A. (2005). XML data stores: emerging practices. *IEEE Internet Computing, 9*(2), 62-69.
W3C. (2016). Introduction to XML. Retrieved 8/9/2016, 2016, from http://www.w3schools.com/xml/xml_whatis.asp
Wang, H., & Meng, X. (2005). *On the sequencing of tree structures for XML indexing.* Paper presented at the Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on.
Wang, H., Park, S., Fan, W., & Yu, P. (2003). *ViST: a dynamic index method for querying XML data by tree structures*.
Wang, H., Park, S., Fan, W., & Yu, P. S. (2003). *ViST: a dynamic index method for querying XML data by tree structures.* Paper presented at the Proceedings of the 2003 ACM SIGMOD international conference on Management of data.
Wang, W., Jiang, H., Wang, H., Lin, X., Lu, H., & Li, J. (2005). *Efficient processing of XML path queries using the disk-based F&B index.* Paper presented at the Proceedings of the 31st international conference on Very large data bases.
Wei, W., Haifeng, J., Hongjun, L., & Jeffrey Xu, Y. (2003). *PBiTree coding and efficient processing of containment joins*.
Wei, W., Haifeng, J., Hongjun, L., & Jeffrey Xu, Y. (2003, 2003). *PBiTree coding and efficient processing of containment joins*.
Wu, X., Lee, M.-L., & Hsu, W. (2004a). *A prime number labeling scheme for dynamic ordered XML trees.* Paper presented at the Data Engineering, 2004. Proceedings. 20th International Conference on.
Wu, X., Lee, M. L., & Hsu, W. (2004b). *A prime number labeling scheme for dynamic ordered XML trees.* Paper presented at the Data Engineering, 2004. Proceedings. 20th International Conference on.
Wyke, R. A., & Watt, A. (2002). *XML schema essentials*: Wiley.
Xu, L., Ling, T. W., & Wu, H. (2012). Labeling dynamic XML documents: an order-centric approach. *IEEE transactions on knowledge and data engineering, 24*(1), 100-113.
Xu, L., Ling, T. W., Wu, H., & Bao, Z. (2009). *DDE: from dewey to a fully dynamic XML labeling scheme.* Paper presented at the Proceedings of the 2009 ACM SIGMOD International Conference on Management of data.

Yang, B., Fontoura, M., Shekita, E., Rajagopalan, S., & Beyer, K. (2004). *Virtual cursors for XML joins*.

Yao, B. B., Özsu, M. T., & Keenleyside, J. (2003). Xbench-a family of benchmarks for xml dbmss *Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web* (pp. 162-164): Springer.

Yao, B. B., Ozsu, M. T., & Khandelwal, N. (2004). *XBench benchmark and performance testing of XML DBMSs.* Paper presented at the Data Engineering, 2004. Proceedings. 20th International Conference on.

Yoshikawa, M., Amagasa, T., Shimura, T., & Uemura, S. (2001). XRel: a path-based approach to storage and retrieval of XML documents using relational databases. *ACM Transactions on Internet Technology, 1*(1), 110-141.

Yun, J.-H., & Chung, C.-W. (2008). Dynamic interval-based labeling scheme for efficient XML query and update processing. *Journal of Systems and Software, 81*(1), 56-70.

Zhang, C., Naughton, J., DeWitt, D., Luo, Q., & Lohman, G. (2001). On supporting containment queries in relational database management systems. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 425-436.

Zhuang, C., & Feng, S. (2012). *Full tree-based encoding technique for dynamic XML labeling schemes.* Paper presented at the International Conference on Database and Expert Systems Applications.