Department of Computer Science

Labelling Dynamic XML Documents: A GroupBased Approach

Submitted for the degree of Doctor of Philosophy

(PhD Thesis)

Alaa Abdulbasit Almelibari

March 2015

Supervisor: Dr Siobhán North

Abstract

Documents that comply with the XML standard are characterised by inherent ordering and their modelling usually takes the form of a tree. Nowadays, applications generate massive amounts of XML data, which requires accurate and efficient query-able XML database systems. XML querying depends on XML labelling in much the same way as relational databases rely on indexes. Document order and structural information are encoded by labelling schemes, thus facilitating their use by queries without having to access the original XML document. Dynamic XML data, data which changes, complicates the labelling scheme. As demonstrated by much research efforts, it is difficult to allocate unique labels to nodes in a dynamic XML tree so that all structural relationships between the nodes are encoded by the labels.

Static XML documents are generally managed with labelling schemes that use simple labels. By contrast, dynamic labelling schemes have extra labelling costs and lower query performance to allow random updates irrespective of the document update frequency. Given that static and dynamic XML documents are often not clearly distinguished, a labelling scheme whose efficiency does not depend on updating frequency would be useful.

The GroupBased labelling scheme proposed in this thesis is compatible with static as well as dynamic XML documents. In particular, this scheme has a high performance in processing dynamic XML data updates. What differentiates it from other dynamic labelling schemes is its uniform behaviour irrespective of whether the document is static or dynamic, ability to determine all structural relationships between nodes, and the improved query performance in both types of document. The advantages of the GroupBased scheme in comparison to earlier schemes are highlighted by the experiment results.

Declaration

I declare that the composition of this thesis and the work within are entirely my own, apart from the specified exceptions. This work has not been put forward for any other degree or professional qualification, except as stated.

Alaa Abdulbasit Almelibari

Acknowledgements

First of all, I wish to praise God for endowing me with the perseverance and dedication to bring this research to fruition.

Secondly, I wish to express my gratitude to my supervisor, Dr Siobhan North, who not only offered me invaluable advice and assistance throughout this research, but also supported me tirelessly. She is an exceptional supervisor and an extraordinary person.

I am beyond grateful to my parents for their unconditional love, support, encouragement and, most importantly, for having faith in me. I would not have been able to overcome the many hurdles and complete this work without their help and guidance and therefore I wish to dedicate my success to them. I am also thankful to my siblings (Abdullah, Mohammed, Ammar and Arwa) for encouraging me throughout my research. I love them more than words can say.

I am incredibly grateful to my husband (Hani) and our amazing little princess (Danah) without whom I could have finished this work a whole year earlier!
You are the most beautiful things in my life and I feel blessed because of you both; you have made me believe in myself. Thank you from the bottom of my heart for giving me the motivation I needed to undertake this work, for being patient and making sacrifices to ensure I had a positive study environment, and for inspiring me to pursue my dreams and ambitions.

I am much obliged to the Saudi Government and Umm-Al-Qura University for offering me the opportunity and funding I needed to undertake higher education studies in the United Kingdom.

Finally, I am highly appreciative of all my friends in Sheffield for their help and encouragement which gave me confidence to persevere with this research.

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Introduction

It has become increasingly important to manage web-based information to keep up with the accelerated pace of the expansion of the Internet. This necessity has promoted the development of XML, which has become the norm for data exchange on the Web (Abiteboul *et al.*, 2000, Assefa and Ergenc, 2012, Champion, 2001, Chang *et al.*, 2012, Choi *et al.*, 2014, Davis *et al.*, 2003, Deng *et al.*, 2013, Härder and Mathis, 2010, Jonge, 2008, Luo *et al.*, 2009, Ogbuji, 2004, Tatarinov *et al.*, 2002, Thimma *et al.*, 2013, Tidwell, 2002, Vakali *et al.*, 2005, W3schools, 2013d, Xu *et al.*, 2012, Zhuang and Feng, 2012). This has resulted in extensive study of XML databases and associated technologies, with an emphasis on data storage, access, retrieval and updating.

The XML labelling scheme is the key to managing XML data competently and rigorously. XML labelling basically means the act of assigning labels or identification nomenclature to nodes in XML trees (Bosak and Bray, 1999). Labelling gives each node a unique identification, it ensures that it is to establish the relationship that exists between any two nodes in an XML tree. At first, the concern of most studies of XML was navigating and retrieving data in static documents, which do not require node labels to have wide-ranging functionality. This is because, well formed XML documents were not considered to require any externally aided approach such as labelling to make them identifiable (Chung *et al.*, 2002, Jiang *et al.*, 2011, Kaushik *et al.*, 2002a). Because the XML documents were considered well formed, they were thought to have the ability to be read and understood by the use of the XML parsers without node labels (Kaushik *et al.*, 2002b, Li and Ling, 2005b, Tatarinov *et al.*, 2002, Wan and Liu, 2008, Wang and Meng, 2005, Zhang *et al.*, 2001).

Nowadays XML is not static, documents change. It is important that dynamic XML documents are managed effectively as the majority of well-developed and popular database products now incorporate XML processing. In the context of dynamic and complex XML documents, labelling becomes essential to aid query processing. Query processing refers to the ability of data retrieval, update, delete and manipulate.

Numerous researchers (Amagasa *et al.*, 2003, Cohen *et al.*, 2010, Eda *et al.*, 2005, Li and Ling, 2005a, Li and Ling, 2005b, Li *et al.*, 2006a, O'Neil *et al.*, 2004, Wu *et al.*, 2004, Xu *et al.*, 2009) have put forward dynamic schemes, but none of these is entirely satisfactory, thus warranting further exploration. In response, this thesis has created a new dynamic labelling scheme entitled *'GroupBased'*, which is primarily geared towards enhancing the performance of both dynamic and static XML documents.

The current chapter presents the research motivation in Section 1.2, the research methodology and hypothesis in Section 1.3, and the scope of the research is described in Section 1.4. The research aims and objectives are outlined in Section 1.5. Moreover, the chapter provides an overview of the structure of the thesis in Section 1.6 before ending with a conclusion in Section 1.7.

1.2 Research Motivation

Overall, this thesis seeks to propose a labelling scheme that supports the effective management of dynamic XML trees. To underscore this motivation, the significance of XML databases and of labelling schemes is discussed in Sections 1.2.1 and 1.2.2 respectively.

1.2.1 The Importance of XML Databases

Data storage, transfer and management are the main functions of XML. As argued by several researchers (Abiteboul *et al.*, 2000, Champion, 2001, Connolly and Begg, 2005, Tidwell, 2002, W3schools, 2013d), XML is advantageous not only because it can be read by people and machines alike, but also because of its flexibility, simplicity and self-definition. Recently, XML's properties of standardisation, and especially its flexibility, have been applied in many contexts, among others, in data mapping, cardinality variations, optional or non-existing structures, have become the catalysts for drawing complex write/read applications, allowing non-uniform data stores, as well as promoting the fusion of data (Abiteboul *et al.*, 2000, Assefa and Ergenc, 2012, Champion, 2001, Choi and Wong, 2014, Chung *et al.*, 2002, Deng *et al.*, 2013, Härder *et al.*, 2007, Jonge, 2008, Liu *et al.*, 2013, Luo *et al.*, 2009, Noaman and Al Mansour, 2012, Tatarinov *et al.*, 2002, Thimma *et al.*, 2013, Tidwell, 2002, Vakali *et al.*, 2005, W3schools, 2013d, Xu *et al.*, 2009, Zhuang and Feng, 2012).

In most industries, business models employ large and constantly developing sets of barely populated attributes (Cunningham, 2006, Duong and Zhang, 2005). Increasingly, firms have come to rely on XML, even going so far as to establish corporations (Bosak and Bray, 1999, Gou and Chirkova, 2007) to create XML schemas compatible with their data modelling requirements. Since many applications demand data flexibility, it is no surprise that XML databases are used with growing frequency not only in collaborative contexts, but also in competitive ones (Loeser *et al.*, 2009). The increasing popularity of XML databases has intensified investigations focusing on enhancing their performance.

1.2.2 The Importance of XML Labelling

Large volumes of data are managed directly in XML data format. The current XML technology is however facing many challenges due to the particularities of data

management in concrete applications (Abiteboul *et al.*, 2000, Bouganim *et al.*, 2004). The issues can be generalised as the need for handling data that is imprecise and uncertain, through application of fuzzy logic, probability and even soft computing (Ma and Yan, 2010).

Increased volumes of data handled by XML document has necessitated the development of XML databases. This is due to the need to manage XML documents since nowadays many applications are using it to store their configurations and data. Such applications include Microsoft Office and Open Office (Barbosa and Bonifati, 2007).

The XML tree structure basically refers to the unique nature in which the XML document is arranged to form a tree which starts at the root, having branches and developing further on to form leaves (Abiteboul *et al.*, 2000, Darugar, 2000, Harold *et al.*, 2004, Ray, 2003, W3schools, 2013k). The XML tree is underpinned by the interconnection of nodes and specific edges. In a typical XML document such as the one given in Figure 1.1, the correspondening XML tree is shown in Figure 1.2.

```
<?xml version="1.0" encoding="UTF-8"?>
<menu>

<drinks>

<hot-drinks>
      <coffee> </ coffee >
       <tea> </ tea >
</ hot-drinks >

<cold-drinks>
       <soft-drink> </ soft-drink >
        <juice> </ juice>
 </ cold-drinks >

 </drinks>

<food>
<rice> </rice>
<chips> </chips>
 </food>

</menu>
```

Figure 1.1: an example of XML document

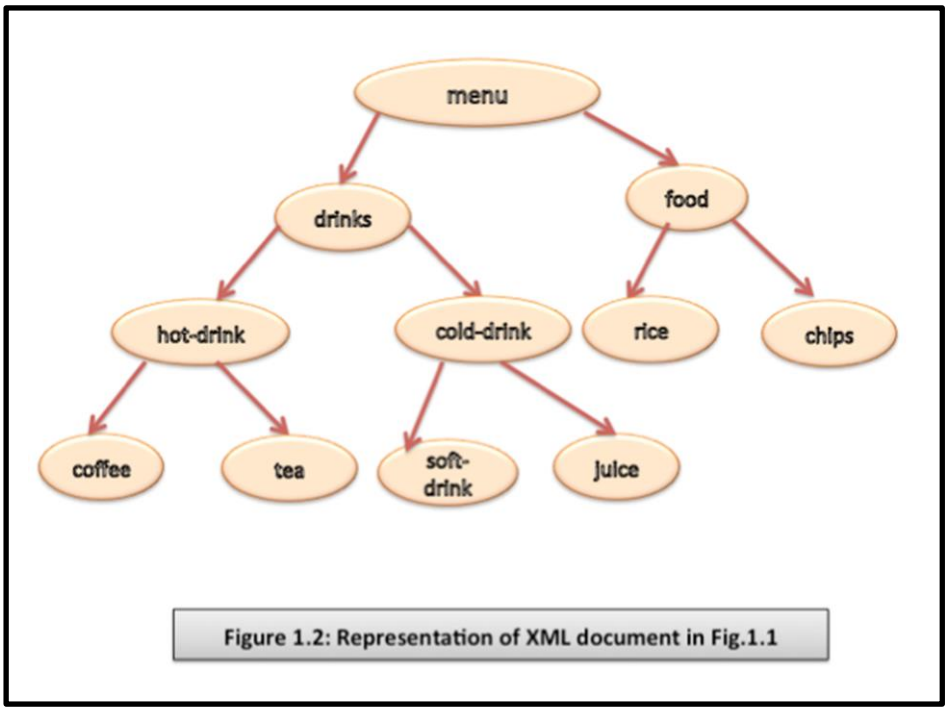


Figure 1.2: Representation of XML document in Fig.1.1

The root element, acting as 'parent' for the other elements, is the starting point of the tree which is node 'menu' in Figure 1.2.

Various relationships and family orientations can be identified within an XML tree. The first of these is the parent/child relationship, which can be identified between a node and any immediate node resulting from it (Wu, Lee & Hsu, 2008). Using Figure 1.2, it can be said that node 'drinks' and node 'hot-drinks' form a parent/child relationship. Another relationship is sibling, which exists between nodes that share the same parent node (Cunningham, 2006); such as node 'drinks' and node 'food' in Figure 1.2. Writing on the ancestor/descendant relationship, Yun and Chung (Yun and Chung) explained that any child that a parent has forms a descendant of the parent. By extension, all children and their siblings are descendants of the parent. Zhang et al. (Zhang *et al.*) also added that it does not really matter how far down the family tree is, all children of children remain the descendants of the parent. From the XML tree in Figure 1.2 therefore, it can be said that 'hot-drink' is a descendant of 'menu'. Using all the explanations given here, any element that comes above another in the family tree is an ancestor. This means

that the node is ancestor to all its descendants; from Figure 2.1, 'drinks' is an ancestor to 'hot-drink', 'cold-drink' and all their children nodes.

The tree structure limits the storage capacity of the XML document. As a result, pointers in the trees occupy most of the storage space. However, a solution to this is to avoid the storage of pointers, and instead, store the tree as a sequence in a link list, and to make use of layers by using them to store the content of every node (Shen *et al.*, 2010).

In XML, the document order is significant and affects the data that is returned by queries. However, in relational databases, the data is stored in tables with rows and columns. The order of the rows in relational data does not give a clue to the ordering of the data (Hunter *et al.*, 2007). As a result, the main reason why XML databases have been so slow to take off could be attributed to the fact that the storage of XML documents on file systems works extremely well (Shen *et al.*, 2010).

To cater for the increasing importance in XML data management, XML labelling schemes were invented and much research has been done to develop more efficient labelling schemes. XML labelling schemes refer to tools, which are basically used to assign unique labels to the nodes in the tree such that constant time is taken in the determination of the relationship between two nodes from the labels. A good labelling scheme is, therefore, measured by how well it determines the relationship between XML elements and how it quickly offers access to the desired data (i.e. provide better query performance) (Haustein and Härder, 2007, Min *et al.*, 2009).

The performance of a query in any database depends on the data being indexed and in XML the indexing process is based on the labelling schemes (Johnson *et al.*, 2012). Thus, XML querying depends on XML labelling in much the same way that relational databases rely on indexes. Labelling schemes permit the identification of structural relationships between elements and attributes (e.g. parent-child,

ancestor-descendant, and document order) based on comparison to their labels. As specified in (Sans and Laurent, 2008), at present there are two major categories of labelling scheme: namely, interval-based schemes and prefix-based (Dewey) schemes. The labelling method of interval-based schemes involves the representation of identifiers as intervals. To establish the connection between two specific nodes, it uses the associated containment information.

In general, the interval-based scheme offers limited information, particularly with regard to the lowest common ancestor (LCA) of a series of nodes. The prefix-based scheme employs a depth-first tree traversal to achieve the direct encoding of the father of a node in a tree as a prefix of its label. Structural relationships can be successfully identified with the prefix-based scheme. In addition, this labelling technique is the preferred option for the query processing of XML keywords (Gou and Chirkova, 2007, Sun *et al.*, 2007, Xu and Papakonstantinou, 2005) that use LCA assessments due to the fact that the labels in the scheme encompass path information. This is discussed further in Chapter 3.

There has been a surge in the need for XML updates thanks to the growing preference for XML as a data exchange format. A labelling scheme supporting solely static XML queries is not enough for XML to become a general standard for data representation and exchange; a labelling scheme that effectively supports dynamic XML trees is also necessary. A dynamic document is one that is continually edited and updated. It may or not have a framework for making these changes. This type of document though, without the proper contextualization can change the content of the document to something very different from the original document. A static document on the other hand does not allow changes to be made (Behrends, 2007). It is written in advance anticipating a particular process. XQuery usually in the form of XML is a functional programming language as well as a query designed to query and change both structured and unstructured data for other data formats (Groppe, 2008). It enables data transfer from virtual or real documents in the wide world web to or from databases providing an interaction that is much needed. A static XML query is concerned with the retrieval of

information and updating the node contents. It does not involve any other changes to the structure of the document (Olteanu, 2005). A dynamic XML query not only retrieves information and updates the content of the document in question; it also inserts new nodes or deletes existing nodes or both often resulting in a change in the document structure.

However, dynamic queries are problematic and difficult to handle because they require the updating of the labels of many nodes simultaneously with the updating of the original XML document to preserve the efficiency of the labelling scheme. This issue has been addressed by a number of researchers (Amagasa *et al.*, 2003, Cohen *et al.*, 2010, Eda *et al.*, 2005, Gou and Chirkova, 2007, Li and Ling, 2005a, Li and Ling, 2005b, Li *et al.*, 2006a, O'Neil *et al.*, 2004, Sun *et al.*, 2007, Wu *et al.*, 2004, Xu *et al.*, 2009).

To prevent re-labelling, earlier researchers left gaps between labels. Drawing on the Dewey labelling scheme, O'Neil et al. (2004) developed the ORDPATH labelling scheme. For initial labelling, this scheme employs positive, odd integers, while for subsequent 'careting-in' insertions it uses negative integers. Due to the gaps left, however, ORDPATH is insufficiently compact and, moreover, the label insertions are made more complicated by the 'careting-in' mechanism. Eliminating initial label gaps, Li et al. (2006a) designed a new labelling scheme for processing updates in XML documents by modifying the labels to be more compact and enhanced update efficiency.

The downside of converting labels into dynamic formats is that it enhances the complexity of updating and querying. Xu et al. (2009) aimed to increase the encoding performance even more by developing two new labelling schemes for dynamic XML trees on the basis of mathematical operations on Dewey elements. Although the performance of the labelling schemes during the updating of XML documents has improved, their labels continue to lack compactness, producing additional storage cost. Furthermore, data querying is time-consuming and the frequent insertion of nodes between two sequential siblings can diminish the

performance of the two labelling schemes. Thus, the capability of XML database management depends on efficient dynamic labelling.

Generally, labelling schemes need to be dynamic such that they can update XML data dynamically and avoid re-labelling or even recalculating of the value of the existing labels (Tian and Georganas, 2002). Efficient schemes have to avoid completely re-labelling in XML updates (Mirabi *et al.*, 2010). They should also be compact, meaning the length of the labels ought to be as small as possible. Finally, they need to facilitate the identification of various relationships existing between the nodes, to be effective (Duong and Zhang, 2008).

All these aspects will be discussed in more details in the next chapters.

1.3 Research Methodology and Research Hypothesis

1.3.1 Research Process

The research's methodology was directly influenced by the research process onion developed by Saunders et al (Collis *et al.*). This research process comprises five major modalities that influence the overall methodology to the research. These five major modalities are research philosophy, research approach, research strategy, time horizons, and data collection methods. The methods followed in the usage of the research process onion have been summarised in the diagram below and subsequently explained in detail.

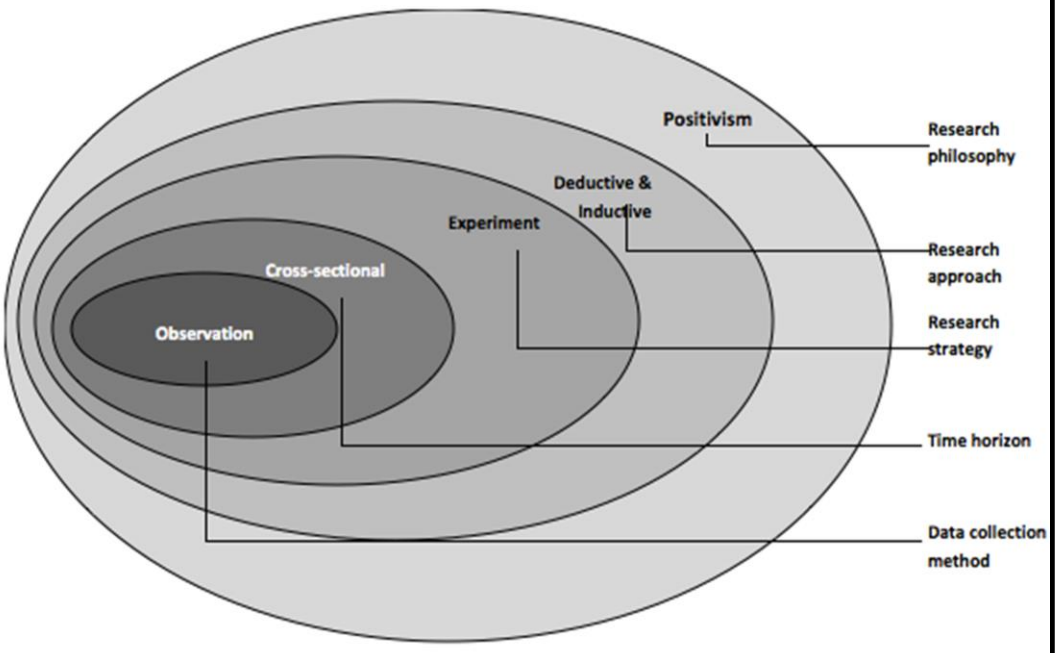


Figure 1.3: Research Process Onion *(Adopted from Saunders et al. (2003))*

1.3.2 Research Philosophy

The research philosophy basically explains how available knowledge influences the research and how the research seeks to develop new lines of knowledge (Diriwächter and Valsiner, 2006). Saunders *et al.*(2003) therefore posited that the research philosophy shows the researcher's overall perception of the way knowledge is constructed. To use and construct knowledge for any research of this nature, three major types of research philosophies may be selected. These are positivism, realism and interpretivism (Remenyi, 1998). The current research made use of the positivism research philosophy. The positivism research philosophy has been explained as an approach to knowledge where the researcher uses scientific reasoning and law-like generalisations in the knowledge construction process (Adams, 2011). This means that using positivism required the research to be based on scientific processes that are generally empirical and evidence based. Thus, this thesis shows how the GroupBased scheme could improve XML labelling (Alstrup and Rauhe, 2002). It was also important to make use of law-like generalisations which could be interpreted in the form of hypotheses that could easily be tested for their validity.

In keeping with the research motivation explained above, the hypothesis that this research seeks to assess is:
Applying a second layer of labels and grouping the nodes based on the parent-child relationship may facilitate node insertions in dynamic XML data in an efficient way, offering inexpensive labels without excessive label size growth rate in which it is easy to maintain structural relationships, as well as improved query performance.

The rationale that influenced the selection and use of positivism was the need to ensuring that the findings that resulted from the study could easily be assessed by other for validity and authenticity. This is because Green, Johnson and Adams (Green *et al.*) saw the positivism research philosophy as a very transparent structure that enhances the replication of findings from the study. Because of the

scientific reasoning and law-like generalisations, it is always possible that the researcher's reasons for drawing conclusions based on the hypothesis can be tested.

1.3.3 Research Approach

The research approach generally describes the means by which the researcher will go about the implementation of the research philosophy. In the current context, the research approach was to establish way the researcher must test the hypothesis set as part of the positivism research philosophy. In research practice, two major research approaches are known; deductive and inductive. Given (2008) explained that the deductive research approach is highly suitable for scientific research as it ensures that the researcher develops a hypothesis and systematically tests it in establishing a theory. This means that for a deductive approach to be used effectively, the researcher must approach data collection from the known to the unknown. This is because the hypothesis is based on what the researcher already knows from a preliminary data collection exercises (Hart, 2008). Based on the hypothesis, the researcher then establishes a theory that is not known or is relatively new in the field of study. Given the fact that the research used the positivism research philosophy which makes the study scientific research, the deductive approach became the main approach that underlined the performance of the research.

Throughout this research, the above hypothesis is tested based on the deductive research approach which originates from a specific case and proceeds to derive generalizations and theories (Jebreen, 2012, Meheus and Nickles, 2009). In deductive research, the propositions made in the beginning in the study only support the truth of the conclusion but do not guarantee it. As such, a deductive researcher conducts the study cognizant that the conclusion might not be true. A deductive proposition helps the researcher to derive universal theories or statements. Strong deductive propositions increase the probability of the conclusion being true but they do not confirm that truth (Khan and Ullah, 2010). Deductive reasoning is chosen because it takes into account the impact of

researcher bias on the outcome of the study (Sans and Laurent, 2008). This is important because different researchers will have different notions and orientations on a given subject, such as in various labelling schemes that cope with dynamic XML documents from different perspectives, and this affects the outcome of studying such a subject (HAMMAWA and SAMPSON, 2011, Stadler, 2004). The fact that deductive research helps in developing a solution to a specific problem is another reason why it is an appropriate approach to use in this research (Khan and Ullah, 2010, Lorenz *et al.*, 2013); as it can also be used to generate recommendations on how to improve various techniques.

Inductive research was however not excluded entirely from the research. Saunders, Lewis and Thornhill (2003) explained an inductive research approach as one in which the researcher formulates the research theory through the critical evaluation of available research variables. This means that instead of using a hypothesis as the main route to forming conclusions and theories, the researcher in an inductive study modifies various research variables based on accumulated findings (Cooper, 2008). Aspects of an inductive approach were also used in the study even though they did not form the main basis on which conclusions were drawn. For example knowledge of what has already been studied on XML labelling schemes in literature was used to set themes or research variables. The review of literature did not become the main basis for drawing conclusions but served as a guide for discussing the researcher's own findings gathered through the deductive approach. In effect, both inductive and deductive approaches were used but with major emphasis and focus on deductive approach so as to maintain the scientific nature of the study.

1.3.4 Research Strategy

The research strategy gives the underling approach used by the researcher in collecting data (Hunter and Leahey, 2008). In this, as many as six possible research strategies are recommended by Saunders *et al* (2003). Of these, experiment was selected as the most appropriate for this research. The major rationale for selecting experiment is due to its direct relationships with the positivism research

philosophy. It should be noted that the positivism research philosophy is appropriate for a scientific study (Robson, 2011). Meanwhile, when used as a research strategy, an experiment requires the researcher to engage in the systematic manipulation of controlled testing with the aim of understanding a causal process (Kasim *et al.*). The aim with which experiment is used as research strategy is to ensure that the researcher can manipulate variables and controls with the aim of measuring any changes that may occur in the variables (Moghaddam and Moballeghi, 2008). In the context of the current study, the researcher was concerned with understanding the behaviours of the GroupBased scheme as a means of improving XML labelling by providing a scheme that deals with insertions without having to re-label or sacrificing the queries' performance, construction time and memory usage. This means that the GroupBased scheme was the independent variable based on which dependent variables including query performance, construction time and memory usage were all tested.

1.3.5 Time Horizon

The time horizon basically shows the duration or period within which the phenomena or variables of the study are experimented on (Sapsford and Jupp, 2006). In the literature, two major time horizons were identified longitudinal time horizon and cross-sectional time. The longitudinal time horizon examines a situation or phenomenon over a given period of time, wherease the cross-sectional time horizon focuses on a particular moment (Dellinger and Leech, 2007). A cross-sectional time horizon was used on this study as design and implementation was developed purposely for the research. These design and implementation specifications ensured that the performance of GroupBased scheme was tested over a very specific time frame to discover the impact of re-labelling on queries' performance, construction time and memory usage.

1.3.6 Data Collection Method

The overall data collection method used was observations. Observation has been described as a systematic collection of data from a research setting or an experiment through visual interpretation of findings (Creswell, 2007). To use

observation as part of the positivism research philosophy where a hypothesis was developed and gradually tested, a number of processes were followed. These processes have been discussed below.

1.3.6.1 Formulating a Tentative Hypothesis

Based on the observations and the patterns identified, a tentative hypothesis is formulated. In this thesis, after analysing the existing XML labelling schemes (see Chapter 3) and providing explanations for the patterns and problems detected, a more general theory was formulated and the research hypothesis (in Section 1.3) was suggested. As a result, a new labelling scheme called 'GroupBased' is proposed (see Chapter 4) that may have the potential to improve the performance of current XML labelling schemes was proposed.

1.3.6.2 Observation and Patterns Identification

The first phase of the deductive approach begins by collecting data that is related to the research area and observes them to highlight any patterns or meaning that can be extracted from them in order to identify the problem under study. In this thesis, the research problems emerged from a critical investigation of the existing XML labelling schemes (see Chapter 3), which results in determining the research aims and objectives.

1.3.6.3 Testing the Hypothesis

In the third phase of the deductive approach the hypothesis is subjected to tests to see whether it is verifiable. This thesis relies on the testing and assessment of an empirical implementation to explore the research hypothesis (see Chapters 5 and 6). This process comprises a number of aspects, including appraisal of the original labelling time and size to determine the extent to which the suggested scheme can be applied in both static and dynamic XML documents; measurement of the length of time necessary for the identification of structural relationships prior to and following insertions; the impact of various types of insertion on the scheme with regard to the size of the labels and time measurements; and the response time for queries prior to and following insertions (see Chapter 7).

1.3.6.4 Develop a Theory

Evaluating the experiments that used to test the hypothesis and their results (see Chapter 8) should help in generalising a theory and determining the main contributions and limitations of this research (see Chapter 9). Obtaining persistent results after several tests would mean that the hypothesis is supported. Inconsistent results would mean that the hypothesis needs to be changed or rejected. However, in the end a general theory or statement ought to be defined that can help explain similar cases (Li *et al.*, 2014, Weinstein, 2010).

1.4 The Scope of the Research

The aim of this thesis is to propose a new XML labelling scheme that may provide better performance in managing dynamic XML data. To test the performance of the scheme, several factors have been analysed, including labelling time, label size, query response time, and managing updates. It must be noted here that the queries in this context refer to those determining the structural relationships between nodes, node access and information retrieval. Furthermore, updates signify that new nodes are inserted, as opposed to mere modification of current node content. However, the scope of this thesis does not extend to XML document parsing and storage mechanisms; i.e. the thesis does not address how the XML document are parsed and how the labels and the data associated with them are stored.

1.5 Research Aims and Objectives

Taking into account the limitations of current labelling schemes (Section 1.2.2) and the research hypothesis (Section 1.3), the following five research objectives were intended to be accomplished by the proposed scheme:

- ***Compatibility with static as well as dynamic XML documents***

  There are strengths and weaknesses to both dynamic and static labelling schemes. In cases where XML documents require regular updating, dynamic

labelling schemes are normally used as the static ones are less efficient due to the number of nodes that have to be re-labelled. Static labelling schemes are usually employed in XML documents that require sporadic or no updating, as dynamic schemes would generate additional encoding cost and make querying less efficient. Therefore, to improve performance, the selection of either static or dynamic schemes should theoretically be made based on the update frequency of the XML documents. However, things are not as straightforward in reality due to the fact that the updating frequency exhibits time-dependent variations; thus making the distinction between static and dynamic XML documents less clear. This increases the difficulty of choosing between a static and a dynamic scheme, the outcome being often different from the initial plan. These issues highlight the importance of creating a labelling scheme that can be applied to static as well as dynamic XML documents.

- ***Efficient identification of all structural relationships***

  Documents to which the XML standard applies follow an inherent order and their modelling takes the form of a tree. Document order and structural information are encoded by labelling schemes to facilitate their exploitation by queries. The encoding of document order is imperative, but a certain variation is permitted in the quantity of structural information the labels contain. To give an example, prefix-based labelling schemes enable the extraction of sibling relationships, but range-based labelling schemes do not.

- ***Cost-efficiency with regard to labelling time and size***

  Time: It is essential that the creation and allocation of labels are time-effective, as otherwise both static and dynamic documents would have lower performance.

Size: This is a key factor underpinning query and updating performance, but it is beyond the scope of this research.

- ***Avoidance of re-labelling and preserving the label quality when processing insertions***

  Using a persistent labelling scheme is ideal, as XML document updates do not necessitate the re-labelling of current labels. As noted by Cohen et al. (Cohen *et al.*), this lowers the cost of updating and enables users to query the modifications brought to the XML data over time.

- ***Improved query performance***

  Accomplishing high query performance depends on the efficient extraction of structural information from labels.

1.6 Structure of the Thesis

In this section the structural organisation of the thesis is described. In a general sense, the thesis is divided into three parts where Chapters 1-3 represent the first part as they introduce the research and discuss the related background and literature. Chapters 4 and 5 represent the second part since they discuss the main idea of the research in detail from both theoretical and practical points of view. The third part consists of Chapters 6-9 which cover the experimental setup, the analysis of the results, evaluation and the thesis conclusion. The description of the thesis chapters is outlined below:

Chapter 1: The title of this chapter is 'Introduction' and it introduces the thesis in general, explaining the research motivations along with its aims and objectives. It also introduces the research hypothesis and outlines the structure of the thesis.

Chapter 2: Entitled 'XML Background', this chapter provides a descriptive illustration of the basic concepts of XML and its parsing mechanisms.

Chapter 3: This chapter discusses the existing XML labelling schemes from a comparative perspective. Thus, its title is 'Related Work in XML Labelling Schemes'.

Chapter 4: GroupBased Labelling Scheme. This chapter discusses the proposed scheme theoretically by describing the underlying structure of the scheme, defining the rules that serve its intended purposes, and validating these rules using simple algebra.

Chapter 5: Design and Implementation. This chapter describes the design and implementation of the GroupBased scheme from a practical perspective based on the rules specified in Chapter 4. Furthermore, justifications for some practical decisions are provided.

Chapter 6: Experimental Framework. This chapter describes the experiments used to evaluate the proposed scheme and their objectives. The platform used and the chosen datasets are specified, and the existing datasets are described.

Chapter 7: This chapter presents the experimental results along with their analysis in order to assess the proposed scheme's performance and scalability. A comparative discussion is provided and graphical illustration is used to support the analysis. Thus, the title of this chapter is 'Results and Analysis'.

Chapter 8: Evaluation. This chapter discusses the experiments and their results from an evaluative point of view. Then the whole scheme is evaluated and its limitations are identified.

Chapter 9: Conclusion. This chapter summarises the whole thesis and discusses the research's main findings, contributions and limitations. Moreover, some

recommendations to improve the proposed scheme's development are presented and the research's future direction is highlighted.

1.7 Conclusion

To conclude, this chapter offered a brief introduction to the thesis. Then, the motivations behind this research were described and the hypothesis was stated. The research aims and objectives were discussed and finally the structure of the thesis was outlined.

Chapter 2: XML Background

2.1 Introduction

The World Wide Web Consortium (W3C) has facilitated data sorting and sharing between applications through the implementation of a standard called eXtensible Markup Language 'XML' (Abiteboul *et al.*, 2000, Bray *et al.*, 2008, Connolly and Begg, 2005), as a result of which application homogeneity is no longer necessary. The popularity of the XML data model is on the rise, because the XML language is not only convenient and simple, but also supports the storage, transfer, display and retrieval of data in both homogeneous and heterogeneous applications (Abiteboul *et al.*, 2000, Anderson, 2008, Bray *et al.*, 2008, Champion, 2001, Connolly and Begg, 2005, Jonge, 2008, Oqbuji, 2004a, Oqbuji, 2004b, Palani, 2011, Powell, 2007, Tidwell, 2002, Vakali *et al.*, 2005, W3c., 2010, W3schools, 2013d, Whatley, 2009). This has led to a surge in the number of XML-supported technologies and applications. Database technology developers have responded to the growing demand for XML data management primarily by upgrading the strategies of XML database management, to include the storage, retrieval and security of XML data. Furthermore, labelling schemes, which encrypt the data related to the XML tree order and structure into highly compact labels, have attracted significant interest. This is a commonly used method of supporting XML data management (Cohen *et al.*, 2010, Li and Moon, 2001, Milo and Suciu, 1999, Silberstein *et al.*, 2005, Tatarinov *et al.*, 2002, Xu *et al.*, 2009). Nonetheless, despite the comprehensive analysis of labelling methods, considerable difficulties have been encountered in developing an appropriate labelling scheme; given its importance for the effective management of XML data, this area is currently intensely researched. As specified in the last chapter, this thesis aims to address the issue by attempting to design a new dynamic labelling scheme and comparing it to other available labelling schemes.

This chapter presents an overview of XML to provide a better understanding of the basic concepts starting with an overview of the XML in section 2.2. Followed by a description of XML syntax in section 2.3. Next, the concept of XML tree structure is explained in section 2.4. XML document type definitions and schema are described in sections 2.5 and 2.6 respectively. Section 2.7, describes the most popular XML query languages. Then, XML parsing techniques are described in section 2.8. The concept of XML databases is briefly discussed in section 2.9. Finally, the chapter concludes in section 2.10.

2.2 XML Overview

Nowadays, the Extensible Mark-up Language (XML) is one of the most commonly employed tools for structured data representation (Abiteboul *et al.*, 2000, Cameron, 2008, Jonge, 2008, Oqbuji, 2004a, Oqbuji, 2004b, Palani, 2011, Thimma *et al.*, 2013, Tidwell, 2002, Vakali *et al.*, 2005, W3c., 2010, W3schools, 2013d, Whatley, 2009). Developed from SGML in 1996, the use of XML was advocated by W3C two years later (Oqbuji, 2004b, Tidwell, 2002, W3c., 2010, Whatley, 2009, Al-Badawi, 2010). What distinguishes XML from HTML is the fact that it is not concerned with appearance control, but with data storage and transfer. XML is advantageous for a number of reasons. As a self-describing language, it enables users to design their own tags, which is a feature which makes it highly flexible (Tidwell, 2002). Moreover, the XML language is straightforward and text-based, with a portable data format (Abiteboul *et al.*, 2000, Harold *et al.*, 2004, Palani, 2011, Ray, 2003, Tidwell, 2002, W3c., 2010, W3schools, 2013d, Whatley, 2009). It can also be exchanged among various applications as it is read by the majority of platforms.

An XML document for employees' information named 'EmpRecordList' is illustrated in Figure 2.1. XML files comprise a range of components, such as elements (e.g. <Emp>), attributes (e.g. DeptNo="D003"), and comments (e.g. <!—

Author Name -- >) (Abiteboul *et al.*, 2000, Tidwell, 2002, W3schools, 2013d, Walsh, 1998). In the following section, each XML file component is presented.

```
<?xml version= "1.0" encoding=UTF-8" standalone="yes" ?>
<?xml:stylesheet type = "text/xsl" href= "Employee_Record.xsl"?>
<!DOCTYPE EMPLOYEES_RECORD SYSTEM "Employee_Record.dtd">

<EmpRecordList>

    <Emp deptNum= "D003"  Sex="M">
       <EmpID>  ERL44 </EmpID>
       <EmpName>
          <Fname> Rayan </Fname> <Lname> Darwin </Lname>
       </EmpName>
       <JOB_POS> Suprrvisor </JOB_POS>
       <DateOfBirth> 26-07-1973 </DateOfBirth>
       <Salary> 35000</Salary>
    </Emp>

    <Emp deptNum= "D001"  Sex="F">
       <EmpID>  ERL26 </EmpID>
       <EmpName>
          <Fname> Tacey </Fname> <Lname> Cutt </Lname>
       </EmpName>
       <JOB_POS> accountant </JOB_POS>
       <DateOfBirth> 09-09-1982 </DateOfBirth>
       <Salary> 20000 </Salary>
    </Emp>

</EmpRecordList>
```

Figure 2.1: XML document represents employee information (EmpRecordList.xml)

2.3 XML Syntax

2.3.1 Elements

In the XML document, data representation is textual. An 'element' encompasses everything circumscribed by matching tags when names are case sensitive (e.g. <Emp> and </Emp> in Fig.2.1) (Abiteboul *et al.*, 2000, W3schools, 2013i). An element or tag represents the fundamental component of the XML document. The start-tag and the end-tag, referred to as markups, are, respectively, the starting and end point of an element (e.g. <EmpName> and </EmpName>) (Abiteboul *et al.*, 2000; Connolly and Begg, 2005). Furthermore, an element can consist of additional element(s), text value(s) or both, and it can also be void. A root element, representing the initial element in the document (e.g. <EmpRecordList>), is a crucial component of any XML document. A 'sub-element' is an element incorporated in another element. For instance, the sub-element of the <Emp> element is <DateOfBirth>. In general, the arrangement of elements (tags) in an XML document must be balanced, while the opening and closing of tags should be diametric (Abiteboul *et al.*, 2000, Connolly and Begg, 2005, Tidwell, 2002, Walsh, 1998, W3schools, 2013i).

2.3.2 Attributes

Formed through the association of a name and a value, attributes provide a more expansive description of an element in XML. The position of an attribute is within the start-tag of the element, after its name (Walsh, 1998; Abiteboul *et al.*, 2000; Tidwell, 2002; Connolly and Begg, 2005; Whatley, 2009). Moreover, single or double quotes are required to delimit the value of an attribute, which is always a string value (Connolly and Begg, 2005; W3School (XML Attributes), 2013). The data of XML more often than not have no use for the information that an attribute supplies; nevertheless, the information is of significance for data management. The following is an example of an attribute (deptNum), representing the department in which a staff member works, within the previously established element (Nolan and Lang):

<Emp deptNum="D003">

The distinction between an element and an attribute in XML is the fact that an attribute cannot be repeated, unlike a sub-element included in the same tag (Abiteboul *et al.*, 2000; Connolly and Begg, 2005).

What is more, despite the fact that the use of attributes or elements is not specified by any rule, there is a general preference for elements over attributes. The reason for this preference is that an attribute has just a single value and therefore it is more challenging not only to expand it, but also to maintain and read it (Abiteboul *et al.*, 2000, Ray, 2003, Tidwell, 2002, Whatley, 2009, W3schools, 2013f).

2.3.3 Comments

In spite of the simplicity and clarity of the XML language, comments are still necessary to elucidate complex code or to include further notes for the writer or reader. Although the location within the XML document of the comments is not fixed, they have to be inserted between <!-- and --> tags. Apart from the literal string '--', all data can take the form of comments. However, an XML processor does not transfer comments to an application (Connolly and Begg, 2005, Harold *et al.*, 2004, Ray, 2003, Tidwell, 2002, Whatley, 2009, W3schools, 2013j).

2.3.4 Ordering

As elements in XML are ordered, the fragments in figure 2.2 are not the same (Abiteboul *et al.*, 2000, Connolly and Begg, 2005):

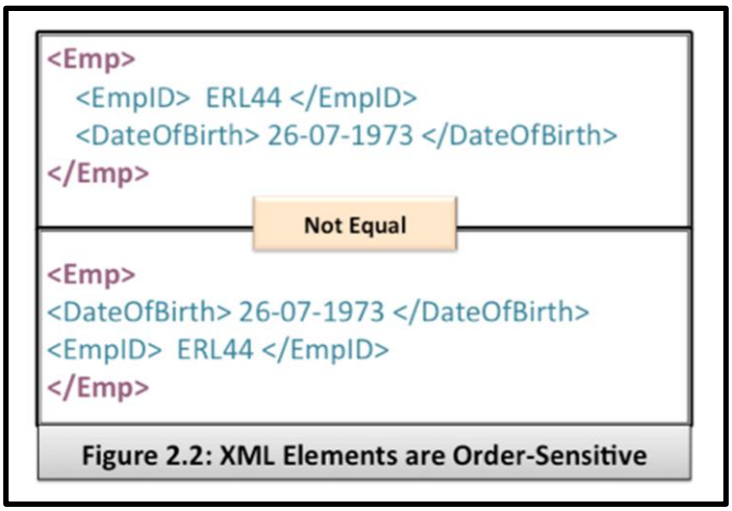


Figure 2.2: XML Elements are Order-Sensitive

On the other hand, attributes in XML are not ordered, meaning that the fragments in figure 2.3 are equivalent (Abiteboul *et al.*, 2000, Connolly and Begg, 2005)

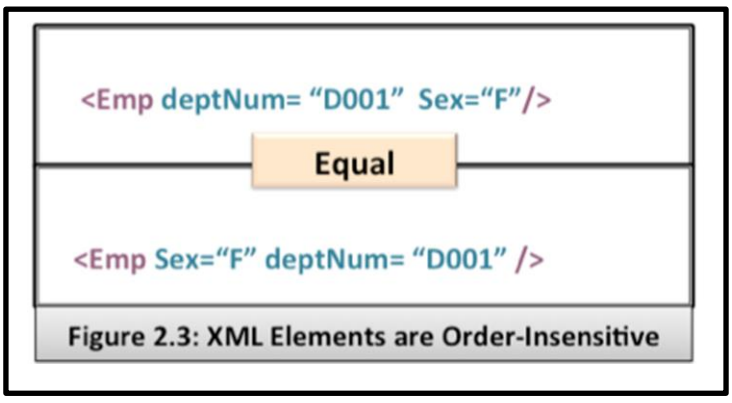


Figure 2.3: XML Elements are Order-Insensitive

2.4 XML Tree Structure

The usual representation of an XML document is that of a tree graph, where it is mandatory for the tree to have a root. Tree branches extend to the lower level from this parent element, depicting additional elements that take the form of nodes (Abiteboul *et al.*, 2000, Darugar, 2000, Harold *et al.*, 2004, Ray, 2003, W3schools, 2013k). The XML data model is underpinned by the interconnection of nodes and specific edges. As noted by Teorey *et al.* (2011), the representation of this tree model assumes the form of structured parent and child relationships.

As previously mentioned, the root element, acting as 'parent' for the other elements, is the starting point of the tree; the additional elements – the child nodes - are conventionally depicted in a lower level. The tree ramifies until the end of the document, as any element may incorporate a sub-element. The tree structure of the document in Figure 2.1 is illustrated in Figure 2.4:

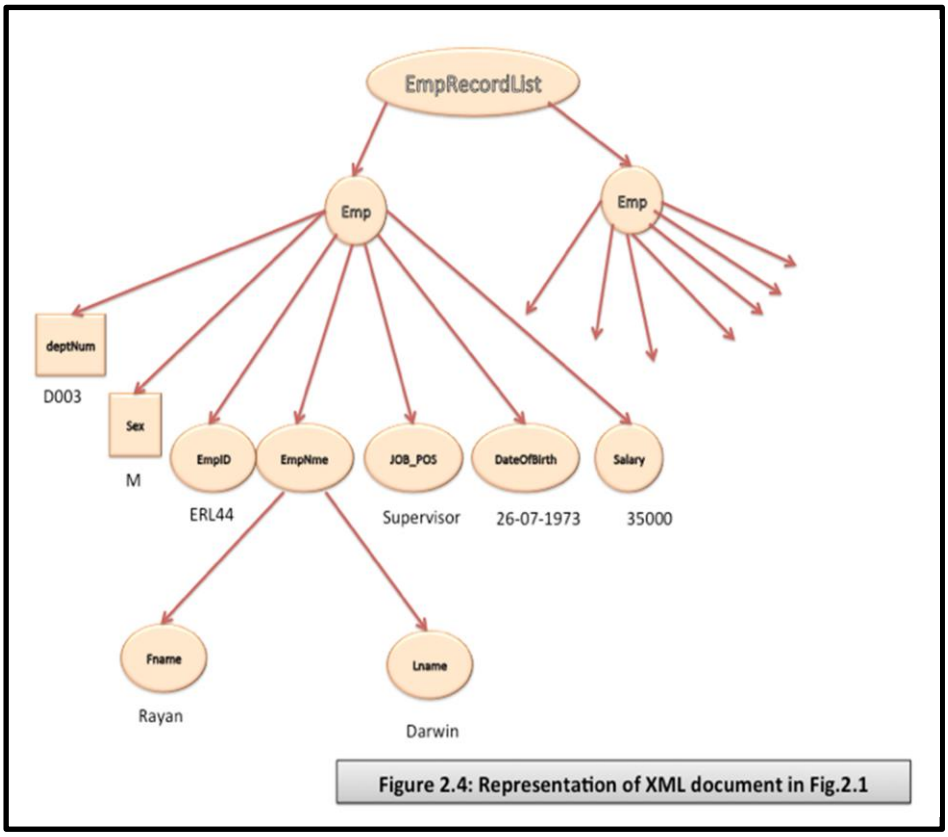


Figure 2.4: Representation of XML document in Fig.2.1

In the next sections, the literature related to methods and standards associated with the XML data model is reviewed.

2.5 Document Type Definitions (DTDs)

The format of an XML document can be outlined by a Document Type Definition (DTD), which indicates, among other things, the names of the elements that can be included in the document, the frequency with which an element can occur in the document, the order of the elements, the connections between elements and the manner of their arrangement, as well as the attributes for every element type (Connolly and Begg, 2005, Elmasri, 2008, W3schools, 2013c, W3schools, 2013a). It can therefore be said that DTD is the grammar that underpins the XML document. The DTD can be defined either within the XML document or as an external file, before being subsequently employed as a reference in the actual XML document (Abiteboul *et al.*, 2000, Chase, 2003, Harold *et al.*, 2004, Lee and Chu, 2000, Molina *et al.*, 2009, Ray, 2003).

Given its capacity to define a data schema and type, the DTD can simulate the relational database schema. However, the relational database schema has an advantage over the DTD, which lacks numerous constraints; for instance, only 'String' data can be declared (Abiteboul *et al.*, 2000, Connolly and Begg, 2005, Elmasri, 2008).

A potential DTD declaration of the 'EmpRecordList' example is presented in Figure 2.5:

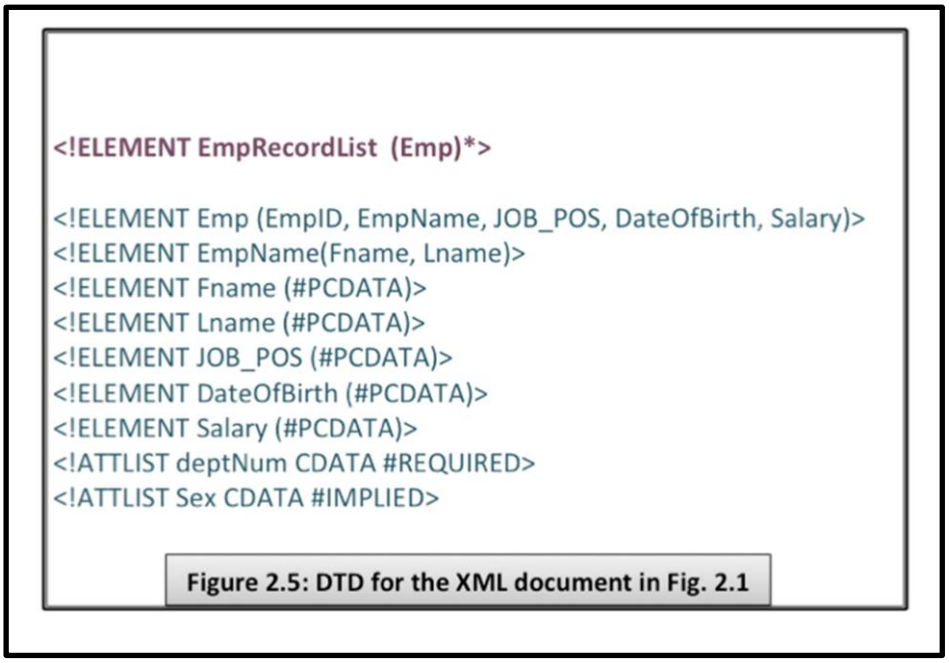


Figure 2.5: DTD for the XML document in Fig. 2.1

2.6 XML Schema

To address the shortcomings of the DTD and provide a more inclusive definition of XML document content, the W3C recommended the implementation of the 'XML Schema' language in May 2001 (Connolly and Begg, 2005, Fallside and Walmsley, 2004). A schema can be defined as a relatively static database description which is formulated during the database design phase (Elmasri, 2008, Molina *et al.*, 2009).

In terms of data types and configuration, the structure of a given XML document is outlined via the 'XML Schema' definition. This involves indicating the manner in which every element is defined, as well as the type of data corresponding to its value. Furthermore, the 'Schema' is actually represented as an XML document, the inherent elements and attributes being used to express the 'Schema'. The 'Schema' is identical to XML with regard to its viewing, editing and processing, as well as the tools necessary to accomplish these procedures (Abiteboul *et al.*, 2000, Connolly and Begg, 2005, Harold *et al.*, 2004, Lee and Chu, 2000, Molina *et al.*, 2009, Radiya and Dixit, 2000, W3schools, 2013b, Waldt, 2010).

Furthermore, the 'Schema' does not have the shortcomings of a DTD as it is more expressive than DTD in terms of supporting various types of data, the domains of the values, and the number of times an element occurs in an XML document (Fallside and Walmsley, 2004).

An XML Schema for the 'EmpRecordList' example is illustrated in Figure 2.6:

```
<xsd:schema  xmlns:xsd= "http://www.w3.org/2001/XMLSchema">

    <xsd:simpleType name = "CHAR_5">
       <xsd:restriction base= "xsd:string">
             <xsd:length value = "5"/>
       </xsd:restriction>
    </xsd:simpleType>

. . . .

    <xsd:complexType name = "EmpInfo">
        <xsd:sequence>
              <xsd:element name= "EmpID" type="CHAR_5" />
              <xsd:element name= "Fname" type="CHAR_16" />
              <xsd:element name= "Lname" type="CHAR_16" />
              <xsd:element name= "JOB_POS" type="CHAR_14" />
              <xsd:element name= "DateOfBirth" type="DATE" />
              <xsd:element name= "Salary" type="DECIMAL_5_2" />
              <xsd:element name= "deptNum" type="CHAR_4" />
               <xsd:element name= "Sex" type="CHAR_1" />
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name = "Table" >
        <xsd:sequence>
              <xsd:element name= "tuple" type="EmpInfo" ninOccurs="0" maxOccurs="unbounded" />

        </xsd:sequence>
    </xsd:complexType>

     <xsd:element name= "EmpRecordList" type="Table"  />

</xsd:schema>
```

Figure 2.6: XML Schema for the XML document in Fig. 2.1

2.7 XML Query Languages

The relevance of database systems in the management of data derives from the procedures for data retrieval, processing, extraction, conversion and integration, which are dealt with on the basis of query language (Boag *et al.*, 2011, Connolly and Begg, 2005, Elmasri, 2008).

SQL cannot be used to query XML data, as they share similarities with semi-structured data. Consequently, XML data have to be queried using specific languages. The most commonly used languages for querying XML data are XPath and XQuery, which are discussed in the following sections.

2.7.1 XML Path Language (XPath)

W3C recommends XML data to be queried with the use of the XML query language XPath, which employs a simple syntax to manipulate the elements and attributes in an XML document (W3schools, 2013m). XPath treats the XML document as a logically ordered tree-structure. There are seven distinct nodes that make up the XPath tree, namely, element, attribute, text, namespace, processing instruction, comment, and the document's root. Every component of the XML document has an equivalent node in XPath (Berglund *et al.*, 2010b, Boag *et al.*, 2007, Connolly and Begg, 2005, W3schools, 2013n). XPath deals with the XML document based on the mechanism which determines the start node as well as the so-called 'location path' from one node to another (Berglund *et al.*, 2010b, Harold *et al.*, 2004, Molina *et al.*, 2009, Ray, 2003, W3schools, 2013l). Similar to the location path in the directory, the 'location path' in XPath comprises a number of steps linked by '/' to identify the location, its root and final destination being the starting and end point, respectively. Every step in the 'location path' is underpinned by the axis and 'node test' pair. The axis denotes the direction of navigation, whereas the 'node test' indicates the node type in the document. In addition to this, a predicate condition may also be present within square brackets, acting as a filter condition to identify a particular node or a nodes with a certain value (Harold and Means, 2002; Ray,

2003; Molina *et al.*, 2009; W3C, 2010; Elmasri and Navathe, 2011Connolly and Begg, 2005; W3School (XPath Syntax), 2013). Table 2.1 shows the thirteen types of axis associated with XPath:

AxisName	Abbreviation (if any)	Purpose
ancestor		refers to all ancestors of the current node
ancestor-or-self		refers to the current node itself and its ancestor nodes
descendant		refers to all descendants of the current node
desendant-or-self	//	refers to the current node itself and its descendant nodes
following		refers to all siblings of the current node
following-sibling		refers to the siblings which are after the current node
namespace		refers to the namespace of the current node
parent	..	refers to the current node's parent
preceding		refers to every node exist before the start tag of the current node
preceding-sibling		refers to all siblings before the current node
self	.	refers to the current node
child	/	refers to al children of the current node
attribute	@	Returning all attributes of the current node

Table 2.1: XPath Axis

To explain how these marks are used, here are some examples of XPath expressions for the EmpRecordList

a) / EmpRecordList/Emp/JOB_POS

The ***JOB_POS*** node, which is attached to the parent ***Emp*** node, can be accessed via this expression.

b) / EmpRecordList /Emp@deptNum

The identifier (***deptNum***) attribute of the ***Emp*** node can be selected through the above expression.

In XPath expressions, the role of predicates, represented in square brackets "[ ]", is to facilitate the identification of particular nodes and values (Berglund *et al.*, 2010b, Connolly and Begg, 2005, Elmasri, 2008, Harold *et al.*, 2004, Molina *et al.*, 2009, Ray, 2003, W3schools, 2013o).

2.7.2 XML Query Language (XQuery)

The W3C Query Working Group has recommended the XML query language XQuery (Boag *et al.*, 2011), developed on the basis of the 'Quilt' XML query language. XQuery is comparable to SQL, in that the representation of a query takes the form of an expression that can undertake functional tasks, while the value of the expression consists of ordered nodes or atomic values (Al-Badawi, 2010). Given that it is an extension of XPath, XQuery has path expressions identical to those of XPath (Al-Badawi, 2010, Connolly and Begg, 2005). The result of the expression is an ordered series of nodes; however, the result may be affected by redundancy due to repetition of the same node with the same name and type. Moreover, similar to the XPath, the XPath axis dictates the direction of movement of every step in the expression. Additionally, predicate condition(s) can be applied to narrow down or exclude nodes in every step (Connolly and Begg, 2005). Several

other new expressions were developed, apart from the XQuery path expression. 'FLWOR' is one such expressions, being an acronym for FOR, LET, WHERE, ORDER BY and RETURN clauses. FOR or LET (multiple clauses are permissible) represent the starting point of the expression, while the subsequent WHERE and ORDER are optional. RETURN is necessarily the end point of the expression (Boag *et al.*, 2011, Cameron, 2008, Connolly and Begg, 2005, Elmasri, 2008, Molina *et al.*, 2009, W3schools, 2013e).

2.7.2.1 FLWOR Expression

- *The FOR and LET clauses:*

  These two clauses bind values and variables. Values may have multiple variables, an association which is known as 'tuple'. The FOR and LET clauses are employed, respectively, with and without the repetition. In addition, multiple FOR or LET clauses may be included in an expression (Connolly and Begg, 2005).

- *The WHERE clause:*

  It contains a predicate, which specifies one or more conditions, to minimise and control the result generated by FOR or LET clause (Connolly and Begg, 2005).

- *The RETURN and ORDER BY clauses:*

  Each FLWOR expression must include a RETURN clause. The evaluation of each tuple is the aim of the application of this clause, while the result of the FLWOR expression is given by the combination of all evaluations (Connolly and Begg, 2005, W3schools, 2013e). The sequence of the resulting tuples is denoted by the ORDER BY clause, when it is included (Connolly and Begg, 2005, W3schools, 2013e).

To explain how these clauses are used, here are some simple examples of FLOWR expressions for the EmpRecordList document (See Fig. 2.1)

a) ***Return all employees with salary more than £25,000***

```
LET $EmpSalary := £25,000
RETURN doc("EmpRecordList.xml")//Emp[Salary > $EmpSalary ]
```

b) ***Return all male employees at the department D003***

```
FOR $E IN doc("EmpRecordList.xml")//Emp
WHERE $E /@ deptNum="D003" AND $E /@ Sex="M"
RETURN $E/EmpID
```

c) ***Return all departments that have less than 20 employees***

```
<SmallDepartments>
FOR $D IN distinct-values (doc("EmpRecordList.xml")//Emp@deptNum)
LET $E := doc("EmpRecordList.xml")//Emp[@deptNum=$D]
WHERE count($E) < 20
RETURN
        <deptNum> {$D/text()} </deptNum>
</SmallDepartments>
```

2.8 XML Parsing

The parser plays an essential role in XML file processing, and therefore all XML-based applications incorporate it. The parser is designed to break down the XML text and generate a representation in the shape of a tree or stream. DOM, SAX, JDOM and Xerces2 are just some of the parsers employed in the construction of XML files. Among these, the most commonly used are DOM and SAX, which form the focus of the following sections.

2.8.1 Document Object Model (DOM)

This API represents a 'tree-based' model and the view of the data that employ DOM is from an object-oriented perspective. W3C proposed DOM as a standard of managing XML documents with the use of specific techniques and classes. During the parsing process, based on the DOM interface, the representation of the XML document takes the form of a tree; moreover, this process is performed for the entire document at the same time (Abiteboul *et al.*, 2000, Al-Badawi, 2010, Connolly and Begg, 2005, Eriksen, 2004, Hégaret *et al.*, 2005, Whitmer, 2004).

As observed by Frank *et al.* (2003), DOM facilitates the navigation, access and manipulation of XML data. In addition, it enables not only traversal in any direction, but it also permits the concurrent performance of reading and writing processes, and based on the tree structure, it affords random access to XML data. What is more, DOM provides an appropriate context for XPath (Berglund *et al.*, 2010a), while also managing queries and updates (Al-Badawi, 2010. The platforms that support this parser include .NET, C++ and Java (Zhang, 2006).

DOM defines a 'Node' interface that comprises the sub-classes Element, Attribute and Character-Data, which are applied during XML file processing. The Node interface supplies several techniques through which the components of each node can be accessed; these techniques include the 'parentNode()', which returns a parent node of a particular node, and the 'childNode()', which returns all child nodes for the requesting node (W3schools, 2013h, W3schools, 2013g).

However, despite its efficiency in facilitating rapid access and processing of the nodes, the DOM interface has a significant drawback, in that all the objects of the tree, including their structure, are uploaded onto the computer's memory; hence, any memory limitations may have a negative impact on the performance of the interface (Al-Badawi, 2010, Harold, 2002).

2.8.2 Simple API for XML (SAX)

An 'event-based' API, the SAX interface is an alternative to DOM and is the product of collaborative work undertaken on the XML-DEV mailing list. Each event corresponds to an element in the XML document and therefore the sequence of events emulates that of the elements. Compared to DOM, SAX is straightforward, rapid and highly efficient at parsing because does not store the XML tree in memory and therefore facilitates the parsing of large XML documents (Abiteboul *et al.*, 2000, Brownell and Megginson, Connolly and Begg, 2005, Idris, 1999, Megginson, 2001, Project, 2013a).

As SAX is event-based, the tree is not constructed in the memory; rather, it reports the event, such as the start and end tags of an element, straight to the application during the parsing of the XML file. However, this makes reading the XML data without manipulation difficult. It offers only a top down traversal and ordered access to data, thus restricting navigation and making back navigation completely impossible (Abiteboul *et al.*, 2000, Al-Badawi, 2010, Connolly and Begg, 2005, Project, 2013b).

The DOM and SAX parsers both have advantages as well as disadvantages. The system requirements constitute the determining factor in the selection of one or the other. In this thesis, the parser chosen for the implementation stage was DOM because its application is straightforward. However, DOM's storage limitations mean that its use reduces the range of the scalability test when large XML databases are assessed. This issue is addressed later on in Chapter 5 (Section 5.).

2.9 XML Databases

There are two types of XML files: data-centric and document-centric. In a data-centric XML file, data are highly structured and usually stored in databases. By contrast, in a document-centric XML file, the textual content is semi-structured, as is the case with books (Bourret, 2005, Noaman and Al Mansour, 2012, Noaman and Almansour, 2012, Sun and Wang, 2012). This research applies only to data-centric XML files because of its link to databases application.

The issue of whether or not XML is a database has been intensely discussed. Similar to other types of databases, XML is capable of data storage and retrieval and therefore can be perceived to be a technology that facilitates the construction of databases (Bourret, 2005, Noaman and Al Mansour, 2012, Sun and Wang, 2012). Moreover, it displays numerous properties common in databases, including storage of data in XML files, possession of schemas (DTD and XML Schemas) and query languages (XPath and XQuery), as well as the provision of interfaces thanks to programming languages like DOM and SAX. On the other hand, several properties of database management systems, including update, multi-access, recovery and security, are not exhibited efficiently by XML (Bourret, 2005, Noaman and Al Mansour, 2012, Steegmans, 2004). It is these shortcomings that are at the root of the debate as to whether XML should be considered to be a database. Responding to the shortcomings, many researchers have attempted to improve the XML's database like characteristics. In line with such attempts, this research generally seeks to enhance the dynamic update of XML databases.

Enabled XML database and native XML database are the two existing categories of XML databases (Bourret, 2005, Elmasri, 2008, Molina *et al.*, 2009, Papamarkos *et al.*, 2009, Steegmans, 2004). The first category relies on traditional databases like relational databases to store data, and its primary use is in supporting current applications, as many XML files have already been stored in relational databases (Abd El-Aziz and Kannan, 2012, Papamarkos *et al.*, 2009, Steegmans, 2004). Employing standard approaches, enabled XML databases achieve the transfer of

data from the XML structure to the relational structure with the help of mapping methods (SAXProject). However, it has some weaknesses. Papamarkos *et al.* (2009) indicated that, because of the number of joins, it is inefficient at managing large XML files. Furthermore, it does not take into account the hierarchical structure, nested data and sequence of elements. It may also lose information during the conversion process (Bourret, 2005, Noaman and Al Mansour, 2012, Steegmans, 2004, Sun and Wang, 2012)

A native XML database has as its basic unit an XML file, and therefore it constitutes a suitable method for managing XML databases (Fiebig *et al.*, 2002, Steegmans, 2004, Sun and Wang, 2012). Due to the fact that it is compact, it can be searched with ease and its content can be managed (Bourret, 2005, Sun and Wang, 2012). Additionally, native XML databases enhance the efficiency of retrieval as they supports XML query languages (Bourret, 2005, Papamarkos *et al.*, 2009, Steegmans, 2004, Sun and Wang, 2012). It also has greater flexibility than enabled XML database (Bourret, 2005). The inability of the native XML database to provide data in formats other than XML constitutes its greatest weakness (Abd El-Aziz and Kannan, 2012, Bourret, 2005). There are two types of native XML database: text-based and model-based (Bourret, 2005, Papamarkos *et al.*, 2009). The XML file is managed by the text-based type in the form of text and stored as a file in file systems or as a CLOB/BLOB in relational databases. By contrast, XML data are managed by the model-based type as objects, while file representation takes the form of a tree, like in DOM (Bourret, 2005, Harold *et al.*, 2004, Noaman and Almansour, 2012, Staken, 2001, Steegmans, 2004, Sun and Wang, 2012). Only native XML databases are relevent in this thesis.

2.10 Conclusion

This chapter briefly covered the fundamental topics of XML technology, as it is a huge subject and cannot be totally covered in this limited chapter. However, the described topics provide adequate background and introduction to XML before exploring the XML labelling technology in the next chapter since it is the concern of this thesis.

Chapter 3: Related Work on XML Labelling Schemes

3.1 Introduction

Native XML storage and query support have been the focus of much research due to the growing significance of managing XML data. This task is made more challenging by the ordered tree-structured model of the data, which offers extensive semantic content. To query XML data, there is need to adopt an effective and efficient labelling scheme. XML tree order and structural information, such as parent/child or ancestor/descendant are encoded into highly compact labels by labelling schemes; the result of significant research in the recent past. It is important to note that the metrics for a labelling scheme are the compactness of the encoded labels and the speed of the algorithm for both creation and use of the labels. To develop an efficient labelling scheme that can handle an ordered tree-structured data model, various scholars have focused on the aim of developing a labelling scheme that is efficient and effective in handling both static and dynamic XML documents and these approaches are discussed below.

In the introductory chapter of this thesis, there were specific objectives which defined the motivation of this study, and its goals. The first area of literature relevant to this goal is an overview of labelling schemes. The second part of the literature review presents and discusses other labelling schemes that have commonly been used with XML documents. The first theme is different from the second because in the first, only the overall approach to the functionality of the schemes is presented but in the second, there is more detail of the schemes reviewing their strengths, weaknesses and limitations. By so doing, it exposes the research challenges. Lastly, the literature review will identify the weaknesses and limitations of other labelling schemes to propose alternative ideas for new scheme which helps to address the identified weaknesses and limitations.

Section 3.2 of the chapter provides an overview of the labelling schemes, while Section 3.3 presents common labelling schemes used for XML data along with their strengths and weaknesses, such as prefix-based schemes (Section 3.3.1), interval-based schemes (Section 3.3.2), multiplication-based schemes (Section 3.3.3) and vector-based schemes (Section 3.3.4). A summary of the major XML labelling schemes is provided in Section 3.3.5. Section 3.4 discusses the characteristics to be seen in any ideal scheme. Section 3.5 summarises the literature review and Section 3.6 concludes the chapter.

3.2 Labelling Schemes: An Overview

Four major schemes are overviewed in this section. These are prefix-based schemes, interval-based schemes, multiplication-based schemes and vector-based schemes. After the overview, these will be discussed in later sections of the chapter.

Data representation and information exchanges over the web have increased remarkably over the past decade. To ensure that there is a universal query language that is used in the performance of these web activities, eXtensible Mark-up Language (XML) has emerged as a common data format which defines the rules used for encoding documents in a way that can be considered as both human-readable and machine-readable (Amato *et al.*, 2003). Since XML has been accepted as a standard of exchanging data on the Internet, the improvement of its efficiency through development of robust management schemes has been identified as a potential method of reducing the cost of data searching (Bruno *et al.*, 2002, Catania *et al.*, 2005, Liu *et al.*, 2009, Lu *et al.*, 2005, Sun *et al.*, 2007, Xu and Papakonstantinou, 2005). Murata *et al.* (2009) lamented that regardless of the universal acclamation given to XML, some irregularities may arise from its usage in its most original format. Most of the irregularities have been found to focus on the query function of XML (Zhang *et al.*, 2001). Query as used in this context refers to the permission granted to the human and machine users in establishing contact with the base of the XML document (Abiteboul *et al.*, 2001). In the light of this, a

number of query languages have been developed, particularly by W3C group to be used for XML. Two of these are XPath and XQuery, which have declarative queries and path expressions characteristics (Rousseeuw *et al.*, 1999). These characteristics help to overcome the irregularities of XML. But even with these two query languages, Yun *et al.* (2008) still contended that the need to increase query performance remained necessary to make the functionality of XML over the web useful and effective. As a solution to the quest for an increase in query performance, the creation of effective indexing has been developed over the years (O'Neil *et al.*, 2004). Duong and Zhang (2008) noted that these indexes work mainly by allowing queries to bypass the need to scan a whole table of results.

It is based on the functionality of the all important index that the issue of labelling schemes arises, where the study's major emphasis is on dynamic labelling scheme. Goldman and Widom (1997) explained dynamic labelling scheme (henceforth referred simply to as labelling scheme) as dynamic data used in XML format being extracted from a strange database and placed in a deserved XML format. The presence of labelling schemes have been noted to be important for index functioning because as Murata *et al.* (2009) observed to ensure that the index can function by allowing queries to bypass the entire scanning process, noted it is vital to have a unique label assigned to each node in the XML trees in a way that makes it easy to determine the relationship between any two given nodes. The nodes are basically the identification parameters given to components on the XML tree. In this context relationship means relation such as ancestor- descendant relationship or sibling. The labelling is therefore needed to allow structural queries that can be answered only by the use of index (Yu *et al.*, 2005). What this implies in this case is that the need to access the actual documents is eliminated, making the whole query process fast and effective (Wang *et al.*, 2003). Because the creation of the index is largely based on the presence of the unique labels assigned to each node in the XML tree, several researchers have focused their attention on the development of labelling schemes that are used to achieve this purpose. It is important to note that the various forms of labelling schemes work with path indexing and numbering schemes to facilitate the query process for XML data. The motivation of

this study is however focused on labelling schemes due to their unique roles in the indexing processes.

Several schemes have been proposed to help in making the function of labelling easier in both the contexts of computer and human user of XML documents. For instance Bruno (2002), developed a method consisting of connected stacks. This method facilitates the compact representation of the partial results of a query path and the combination of these paths yields the final matches for a twig query. With the query established therefore, the labelling is further facilitated as because the query in itself is a clue to what the label should be (Bruno *et al.*, 2002). This method has also been advanced by Lu (2005), in processing of twig queries. It was mainly successful due to its efficiency in supporting queries with the help of wildcards and branching nodes which are used in labelling processes.

The identification of structural relationships in data elements such as parent–child, ancestor–descendant and document order is achieved through a comparison of the labels. Sans and Lauren (2008), categorise labelling schemes into two: interval-based (range-based) and prefix-based schemes. Prefix-based schemes are also referred to as Dewey schemes. In interval-based schemes, the identifiers are represented as intervals. To determine the associated link between two nodes, the scheme relies on the containment information. Prefix-based schemes on the other hand employ a depth-first tree traversal to directly encode the parent of a node in a tree as a prefix to its label. This implies that interval-based schemes are likely to yield considerably more limited information than the prefix-based schemes. For example, information regarding the Lowest Common Ancestor (LCA) for a group of nodes is hardly ever provided by interval labels. By contrast, structural relationships can be effectively identified based on the prefix-based scheme. Thus, prefix-based schemes has also become the primary choice for query processing of XML keywords as its labels comprise path information (Gou and Chirkova, 2007, Sun *et al.*, 2007, Xu and Papakonstantinou, 2005). This is of significant relevance for LCA assessment.

The position of Sans and Lauren (2008), who categorised labelling schemes into two broad categories have however been advanced with the introduction of other newer schemes. Other labelling schemes that have been identified include multiplication-based and vector-based labelling schemes. In multiplication-based schemes, the nodes in a XML document are labelled by multiplying atomic numbers (Kha *et al.*, 2002, Wu *et al.*, 2004). In vector-based schemes, this is done by vector orders as derived from mathematics (Xu *et al.*, 2007, Xu *et al.*, 2012, Xu *et al.*, 2009).

All these labelling schemes have significant strengths and weaknesses. It is important to point out that to harness specific strengths possessed by different schemes, hybridization of the labelling schemes has been tried. A hybrid scheme integrates the approaches of different schemes with the aim of developing a scheme with the strengths of several schemes (Haw and Lee, 2009, Yun and Chung, 2008).

3.3 Common labelling schemes used for XML data

As indicated earlier, labelling schemes are highly relevant in the use of XML data as they optimise query retrieval by providing a quick way to determine the relationships that exist between nodes (Zhang *et al.*, 2001). This section of the review is dedicated to the presentation and discussion of some of the existing labelling schemes.

Labelling schemes can be divided into prefix-based schemes (Section 3.3.1), interval-based schemes (Section 3.3.2), multiplication-based schemes (Section 3.3.3) and vector-based schemes (3.3.4).

3.3.1 Prefix-based Labelling Schemes

3.3.1.1 Structure and Description

A prefix-based scheme has the characteristic of directly encoding the father of a node in an XML tree as the prefix of its label (K., 2006). Several prefix-based schemes are have been proposed. They include are Dewey encoding (Tatarinov *et al.*, 2002), LSDX Duong *et al.*(2005), ORDPATH O'Neil *et al.*(2004), and Cohen *et al.* (2010). Of these prefix based schemes, there has been extensive study of Dewey encoding, making it possible to refer to it as the embodiment of prefix labelling schemes in general (Wang *et al.*, 2003). Typical of prefix-based schemes, the Dewey encoding (Tatarinov *et al.*, 2002) is structured such that each node has a label that represents the path from the document's root (Harold, 2004). Of the identified labels, each of them stands for the local order of an ancestor node present in the document's root. In the labelling process, nodes that have the same number of delimiters in their label are assigned to the same level (Wu *et al.*, 2004). The explanation to this is that such nodes with same number of delimiters in their labels are siblings and thus do not require a differentiated labelling processing as their outcomes will be the same. Figure 3.1, illustrates the Dewey scheme.

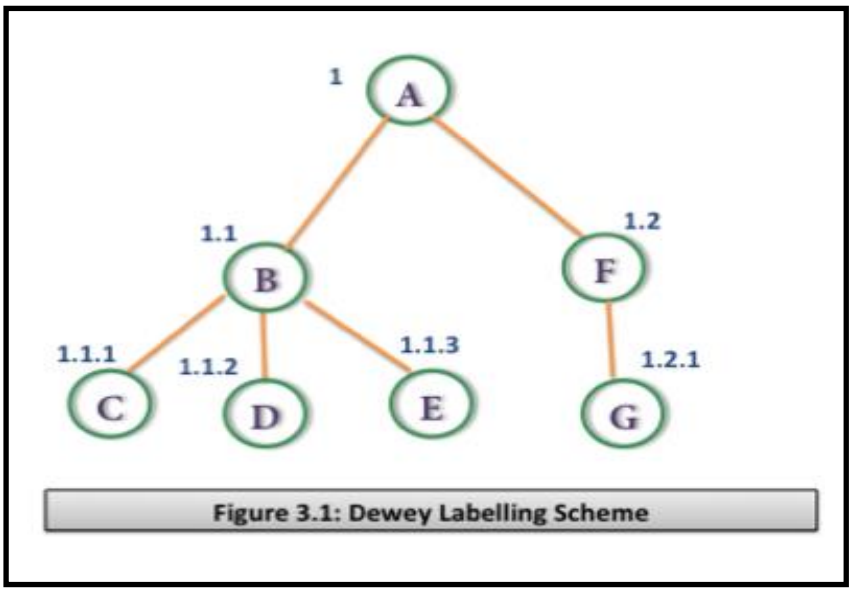


Figure 3.1: Dewey Labelling Scheme

The structural information between two given Dewey labels, *deweyA* : $da_1.da_2 ...da_m$ and dewey*B* : $db_1.db_2 . . . db_n$, can be extracted based on the following rules:

- *Ancestor/Descendant*. For dewey*A* to be the ancestor of *deweyB*, **m<n** and **$da_1 = db_1$, $da_2 = db_2$,...,$da_m = db_m$**.

- *Parent/Child*. For *deweyA* to be the parent of *deweyB*, *deweyA* **must be an ancestor of *deweyB* and m = n-1**.

- *Sibling*. For *deweyA* to be the sibling of *deweyB*, the parent labels of *deweyA* and *deweyB* must match, in other words, **if m=n and $a_1 = b_1$, $a_2 = b_2$, ..., $a_{m-1} = b_{m-1}$.**

For example, from Figure 3.5, node *B (1.1)* is a prefix of node *D (1.1.2)* and therefore *B* is an ancestor of *D*. Furthermore, node *F (1.2)* is compatible with the parent label of node *G (1.2.1)* and thus *F* is the parent of *G*.

The structure of the Dewey scheme strongly resembles other prefix-based schemes including ORDPATH and Labelling Scheme for Dynamic Xml data (LSDX) which were developed by O'Neil *et al.* (2004)and Duong *et al.* (2005) respectively. This is because in the ORDPATH scheme, Thonangi (2006) noted that a child or descendant of a given parent are represented by odd numbers while insertions are given even numbers. Meanwhile in the Dewey scheme also, parent and child nodes are given different non-identical numerical formats of identification. LSDX also has the capability of combining numbers and letters to label each tree as shown in Figure 3.2.

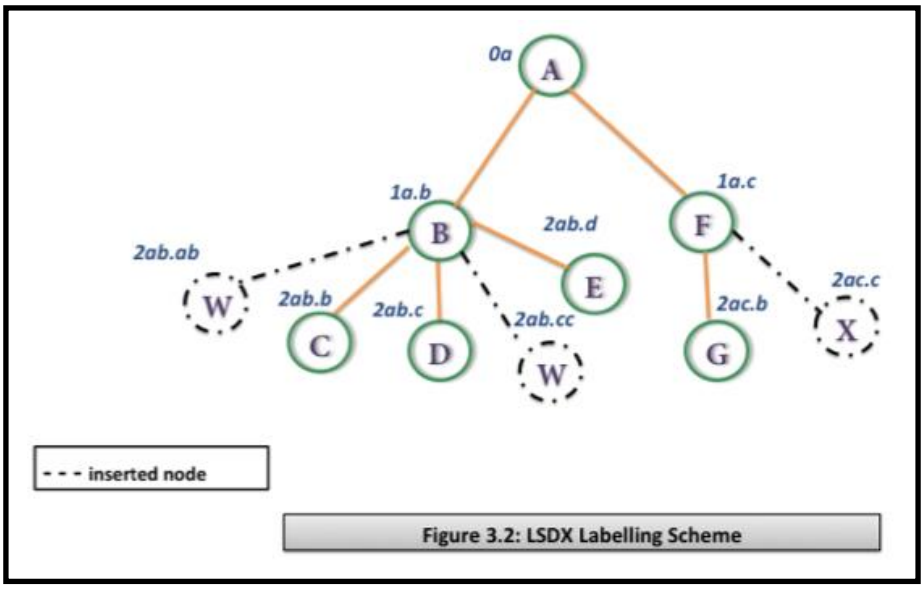


Figure 3.2: LSDX Labelling Scheme

Based on the figure above, it would be noted that Cohen *et al.* (2010) uses a similar structure but this was not represented due to the similarities involved. Moreover, it is the Dewey encoding that has been extensively studied, making it possible to make reference to it as the embodiment of prefix labelling schemes in general.

3.3.1.2 Strengths of Prefix-based Labelling Schemes

The use of prefix-based schemes has been associated with a number of strengths or merits when used as the major labelling scheme to facilitate query processing of XML data. In the first place, Duong and Zhang (2005) posited that prefix-based schemes such as the LSDX act as a persistent labelling scheme that does not require re-labelling of existing labels before it can support the demand for updating XML data. It was for this reason that Alstrup and Rauhe (2002) described prefix-based schemes as being ideal for facilitating fast update of XML data.

Yun and Chung (2008) also mentioned that most forms of prefix-based schemes can handle the representation of ancestor-descendant relationships together with the sibling relationships that exist between nodes. This way it is possible to establish the relationship between any two nodes merely by viewing their unique codes (Amato *et al.*, 2003). This implies that there is much efficiency when using

the prefix-based scheme for querying purposes. Meuss and Strohmaier (1999) also touched on the important role that knowledge of the depth of XML tree plays in facilitating query processing for XML data, stating that such knowledge ensures different node's relatives are given different preferential labelling. Meanwhile, Duong and Zhang (2005) defended their LSDX scheme by stating that it has the ability to show the depth of the tree used in the XML document. This is done mainly by the unique code that is assigned to each level of the nodes. This can be said to be a multi-variant strength that ensures that the tasks of retrieving, inserting, deleting or updating documents is done with so much ease (Hou *et al.*, 2001).

There are other strengths with prefix-based schemes that have mainly been attributed to the Dewey label and ORDPATH. For example Duong and Zhang (2008) indicated that a Dewey label has the capability to single handedly determining the path from the root to an element. This is because it integrates the parent label with its own order. Already, it has been noted that the prefix-based scheme has the ability to provide structural information involving ancestor-descendant relationship, parent-child relationship and sibling relationship (Tatarinov *et al.*, 2002) as described in the previous section.

As an improvement to Duong and Zhang schemes (2005, 2008), Li and Lang (2005a) developed the 'ImprovedBinary' scheme which is a different prefix-based scheme designed to allocate unique and permanent labels to nodes by employing bit strings combined with a recursive algorithm as shown in Figure 3.3.

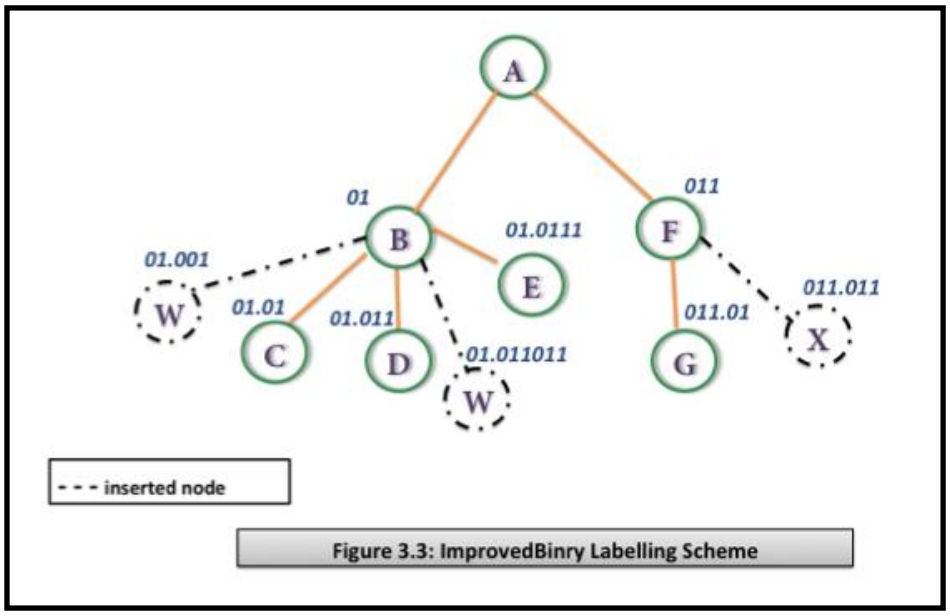


Figure 3.3: ImprovedBinry Labelling Scheme

Writing on the ORDPATH scheme, Duong and Zhang (2005) noted that the scheme has the strength of being very effective in managing updates and insertions. This is due to the fact that in the case of insertions, odd numbers are assigned to parent nodes alone while insertions are labelled with even numbers. One major characteristic with the prefix-based scheme is its potential to function on a group basis through the formation of group-based prefix (GRP) labelling scheme (Wang *et al.*, 2003). Once this is done, it is possible to tap the functional strengths and merits associated with the different schemes that are brought together to form the GRP (Gabillon and Fansi, 2006). In such cases, the GRP combines a group ID and a group prefix.

3.3.1.3 Weaknesses and Limitation

The strengths and merits identified above notwithstanding, there are very specific weaknesses of the use of prefix-based schemes that make their use problematical for query processing of XML data. One such limitation of the prefix-based schemes was identified by Yun and Chung (2008) in the formation of non-tree edges for use in the creation of structural relationship among nodes. The non-tree edges have been explained to be nodes or edges that do not appear in the spanning tree used in a typical tree relationship (Gabillon and Fansi, 2006). This is because apart from

the difficulty associated with the construction of non-tree edge relationships for prefix-based schemes; the resulting non-tree relationships have been noted to lack the strength of deterministic tree label characters. This shows an extensive weakness of prefix-based schemes in the construction of non-tree edge relationships (Boag *et al.*, 2007). To avoid the weaknesses involved, Fennell (2013) recommended the need to apply only deterministic tree labels when using prefix-based schemes. But once the non-tree labels have been used, it can be expected that additional time will be spent in performing such extra tasks such as making provisions for additional storage that will make up for the lapses or serve as backup to the functions that the deterministic tree labels would have played (Duong and Zhang, 2005). Again additional tasks may be required with respect to query processing. Meanwhile, efficiency with time is crucial in the labelling processing. It is not surprising that Hou *et al.* (2001) claimed the prefix-based scheme required special effort to achieve query processing.

Another limitation of the use of prefix-based scheme such as Dewey Encoding (Tatarinov *et al.*, 2002) is the inability to assign extensive labels. Elaborating on this, Murata *et al.* (2009) explained that such limitations show up most when dealing with complex XML documents. This is because these complex XML documents are made up of longer paths than may be seen in simpler XML documents (Bosak and Bray, 1999). These long paths are formed as vector paths in Dewey Encoding as a means of establishing an ancestor-descendant relationship (Cunningham, 2006). The behaviour of such complex XML documents in producing longer paths makes the assessment of extensive labels to a node unfeasible. This is mainly due to the time needed to perform the assessment of extensive labels, where only selected labels or less complex XML documents could be deemed to achieve effective assessment. Wu *et al.* (2008) also opined that in prefix-based schemes, the support for dynamic update is often highly complicated. This makes most researchers avoid dynamic updates. Dynamic updates come with their own benefits, which are lost when using prefix-based labelling schemes. Whenever a dynamic update is started, changes to the parent label causes adjustments to both the child and descendant labels (Fisher *et al.*, 2006). Because when the parent

label is altered, the ancestor's labels will be inherited throughout the document (Yu *et al.*, 2005). As expected, once this is done, the overall updating processing will be complicated, leading to reduce efficiency.

3.3.2 Interval-based Labelling Schemes

3.3.2.1 Structure and Description

An interval-based labelling scheme has a structure where the identifiers of all the nodes are allocated as the start and end position numbers. These are positive numbers distributed in the depth first traverse of the data tree that forms as part of the label numbers (Wu *et al.*, 2004). This process takes place so as to make the identification process possible by constructing an explicit structural relationship between all the nodes. According to Duong and Zhang (2005), this scheme is so called the interval-based labelling scheme because there is an interval created within the nodes, which joins directly with the parent or ancestor to create a parent-child relationship.

Interval-based labelling schemes have been described in a number of papers where researchers have independently identified unique interval-based labelling schemes with different qualities and functionality. There are three common forms of interval-based labelling schemes; the containment labelling scheme proposed by(Zhang *et al.*, 2001), the pre-post labelling proposed by (Dietz, 1982) and the order/size scheme proposed by (Li and Moon, 2001) (2001).

Every element node in a containment labelling scheme is assigned a label with a start, end, and level format, in which a range bounded by start and end comprises all its descendant ranges as shown in Figure 3.4.

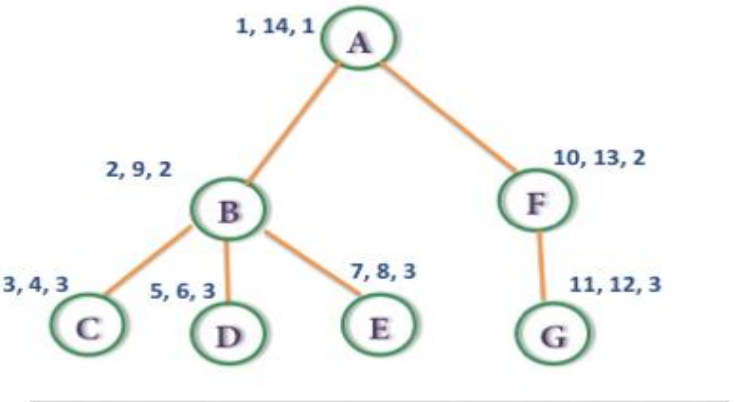


Figure 3.4: Containment Labelling Scheme

From the figure, it can be noted that the element node 'A' is assigned the start format, with 'C, D, E, and G' assigned the end format based on the containment labelling scheme.

Where there are three values for each node, each of these is given a value either as pre or post of the node Zhang *et al.* (2001). This pre or post value generally represents the position of the node, say *A,* whether in a pre-order or post-order of the traversal of the tree. From this point on therefore, there is a change from containment to pre/post labels. In the pre/post labelling scheme, every label has a pre, post and level format; pre-representing the ordinal number of the element node in a pre-order traversal sequence, while post is the ordinal number of the element node in a post-order traversal sequence, as shown in Figure 3.5.

From the figure given below, the level in both labelling schemes refers to the level of the element node in the XML tree. The structural information that can be extracted from two given containment labels, A ($start_1$, end_1, $level_1$) and B ($start_2$, end_2, $level_2$), is as follows:

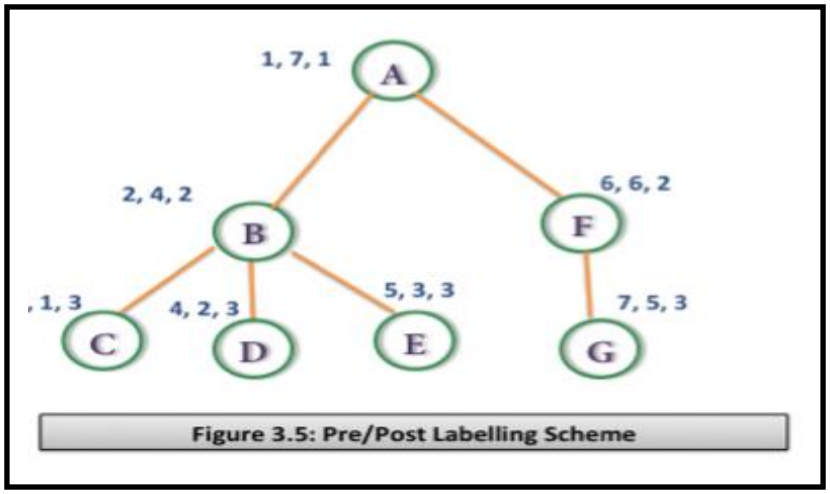


Figure 3.5: Pre/Post Labelling Scheme

- <u>Ancestor/Descendant (AD)</u>. *B* is the descendant of *A* ***if and only if*** $start_1 < start_2 < end_2 < end_1$, which can be reduced to the more simple form ($start_1 < start_2 < end_1$). The simplification is justified as ($start_1 < start_2 < end_1 < end_2$) is not possible as it would signify the improper nesting of the elements.

- <u>Parent/Child (PC).</u> *B* is a child of *A* ***if and only if*** *B* is a descendant of *A* and $level_1 = level_2 - 1$.

For example, in Figure 3.1, *1<5<14*, node *A (1, 14, 1)* is an ancestor of node *D (5, 6, 3)*. Furthermore, *2 < 5 < 9* and *2 =3-1*, node *B (2, 9, 2)* is the parent of node *D (5, 6, 3).*

It is possible to extract AD and PC relationships from pre/post labels with the following:

In the case of two given pre/post labels $A\ (preorder_1, postorder_1, level_1)$ and $B\ (preorder_2, postorder_2, level_2)$, the condition $\boldsymbol{preorder_1 < preorder_2}$ and $\boldsymbol{postorder_2 < postorder_1}$ must be fulfilled for A to be an ancestor of B. Unlike the condition of the containment labelling scheme, this condition cannot be subjected to simplification.

For example, from Figure 3.2, given that *2<3* and *1<4*, node *B (2, 4, 2)* is an ancestor of node *C (3, 1, 3)*.

In Li and Moon (2001), the order/ size scheme of labelling is described to be a triplet. The size of a node in a scheme is the property that determines the number of children the scheme can hold (Li and Moon, 2001). The order on the other hand is the format in which the labelling is performed. This implies that there are three aspects of the node that are taken into consideration. In this case the order/size scheme is seen to be similar to the containment labelling scheme that takes the node's order and order + size.

Figure 3.6, shows an example of order/size labelling scheme.

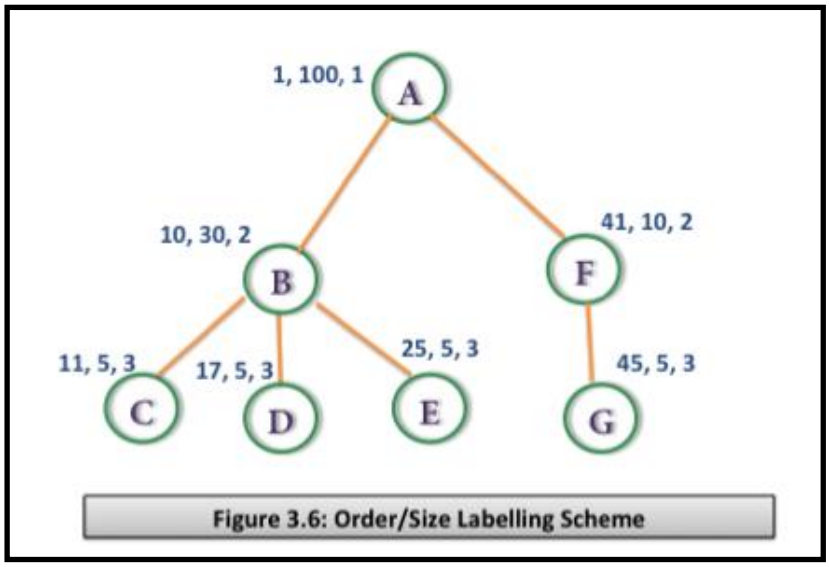


Figure 3.6: Order/Size Labelling Scheme

3.3.2.2 Strengths of Interval-based Labelling Schemes

In the interval-based scheme, it is possible to have labels which can function perfectly as start position number and end position number as depicted in figures 3.5 and 3.6. This ability was classified by Eda *et al.* (2005) to be a major merit when dynamism is important concern in the query processing. This is because when there is a tree label and the same branch of the tree is present in the query process, the start and end position numbers could both be used simultaneously or

interchangeable when the node is traversed back from the branch of the tree (Tatarinov *et al.*, 2002). What is more, the interval that is created between the start-position and end-position in interval-based labelling schemes plays a significant role in establishing both ancestor-descendant relationship and parent-child relationship (Cunningham, 2006). Indeed unlike the prefix-based scheme, the interval-based scheme supports XML tree and this is a major advantage because tree labels have been said to be more efficient than non-tree labels (O'Neil *et al.*, 2004). The basis for the support of XML tree is in the ability of interval-based scheme to take both start and end position numbers, which are often used to describe child and parent relationships on the tree. Fallside and Walmsley (2004) explained that not only does the interval-based scheme support the tree label but that there are both pre and post labelling schemes to which each label is assigned a pre, post and level format to be a pre-representation of the ordinal number of the element node found in the pre-order traversal sequence. Already, the pre/post labelling scheme has been explained and so it can be expected that while there is a pre-order traversal sequence with the pre-order, the post-order found in the ordinal number of the element node is also traversed in a sequence independently (Wu *et al.*, 2004).

3.3.2.3 Weaknesses and Limitation

There are circumstances in which the use of interval-based labelling scheme may be challenging. Similar to the problems with prefix-based scheme, Tatarinov *et al.* (2002) noted that dynamic update is not supported in some cases of interval-based schemes. Specially, when there are more nodes inserted than the interval allocated between the existing nodes (Cooper *et al.*, 2001). This limitation can be attributed to the structure and functioning of the interval-based labelling scheme where greater part of its workability and functionality is dependent on the interval allocated to the nodes. The interval between the start position number and end position number is allocated when the node is traversed back from the same branch of the tree such as a child and its sibling (Duong and Zhang, 2005). This simply shows the efficiency of the interval allocated between the existing nodes

over the inserted nodes. As a result, dynamic update will either be slowed or entirely halted when more nodes are inserted than the interval between the existing nodes (Amato *et al.*, 2003). This is because in this case, it is the inserted nodes rather than the interval allocated that determines the outcome of the dynamic updating process. The frequency of changes of XML documents have been noted to be a major reason why dynamic updating is important and must be considered to take place on a regular basis (Cooper *et al.*, 2001).

Another limitation identified in the literature on interval-based labelling schemes had to do with the fact that re-labelling takes place only under extreme circumstances. In the opinion of Harold (Harold, 2005), this is actually the worst case scenario with interval-based schemes, raising considerable overhead for the scheme and almost nullifying its potential as the most compact labelling scheme.

Because the prefix labelling scheme has an advantage in terms of re-labelling, most researchers have either selected to use prefix labelling schemes above interval-based scheme when their priority is to minimise re-labelling or have used a combination of a prefix labelling schemes and an interval-based labelling schemes in order to attain better re-labelling functionality (Sean, 2006). Also writing on the weaknesses associated with interval-based labelling scheme, Duong and Zhang (2005) noted that re-labelling becomes difficult because of the introduction of a node after a consecutive sibling cannot be computed from the existing solution. The result of this is that re-labelling is always automatically triggered as illustrated in Figure 3.7, which shows the number of nodes that required re-labelling when a new node is inserted.

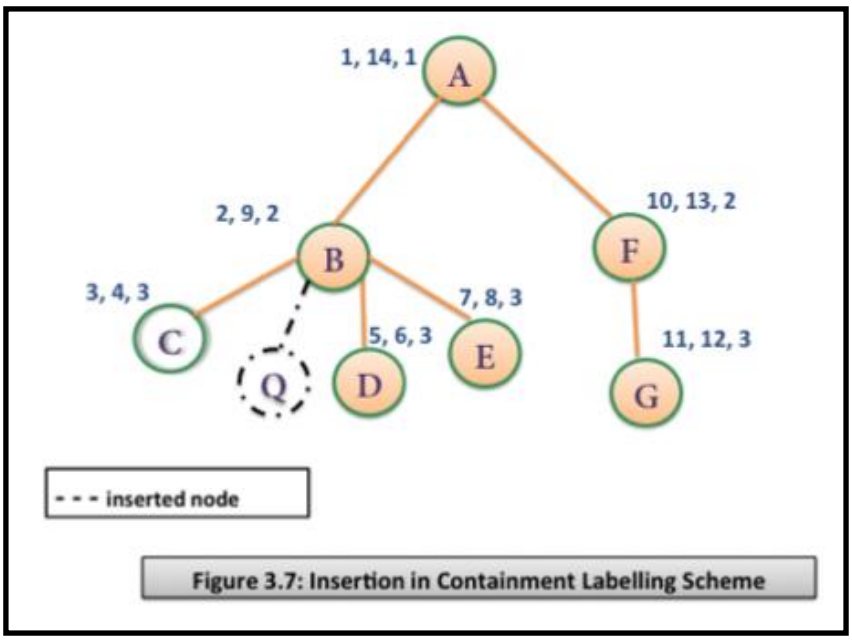


Figure 3.7: Insertion in Containment Labelling Scheme

From the figure above, it is seen that when 'Q' is inserted as a new node, automatic relabeling is triggered with 'C', indicating that there is only one node required for relabelling.

Meanwhile, Gou and Chirkova (2007) were of the opinion that when engaging in query processing, the automatic offset of re-labelling means that the user of the document cannot have any control over the process. Certainly, an alternative labelling scheme that avoids the stress of using a combined scheme in achieving that would be beneficial.

3.3.3 Multiplication-based Labelling Schemes

3.3.3.1 Structure and Description

A major descriptive characteristic of multiplication-based labelling schemes is the positioning and numbers used in identifying their nodes. This is because Zhang *et al*. (2001) saw that multiplication-based labelling schemes exhibit nodes that are determined based on the use of atomic numbers. When it comes to the determination and computation of the relationship between nodes however, it is not these numerical labels that are used (Wu *et al.*, 2004). There are a number of multiplication-based labelling schemes that have been used in optimising query processing for XML. Common among these are the identifier labelling scheme which was used by Kha *et al* (2002) and the prime number labelling scheme used by Wu *et al.*(2004). Like other labelling schemes, the multiplication-based scheme such as the prime number labelling scheme that is often used along with directed acyclic graph (Sjoberg *et al.*) makes use of parents, siblings, ancestors, and descendants' nodes (Schmidt *et al.*, 2002). One unique feature of the multiplication-based scheme however has to do with the fact that it has additional relations such as children and nearest common ancestor (NCA) (Yun and Chung, 2008). In such DAG, the NCA is the lowest node that has two independent nodes on the tree as ancestors. For example given two nodes X and Y, in a tree or DAG, the NCA is the lowest node that possesses both X and Y as ancestors (Boag *et al.*, 2003). Multiplication-based schemes function mainly based on the formation of index structures which are reorganised when there is vertex updated during query processing (Thonangi, 2006). Duong and Zhang (2005) observed a unique behaviour in multiplication-based schemes where before the creation of node labels, schemes such as the prime number labelling scheme allocate unique prime numbers as the labels to every node. The number that results from this then becomes the combination of the self-label of the node and its parent's label. In effect, the self-label which is created that becomes the unique path based on which XML nodes are identified (Meuss and Strohmaier, 1999).

3.3.3.2 Strengths of Multiplication-based Labelling Schemes

When compared to other labelling schemes, Yun and Chung (2008) asserted that the multiplication-based scheme has the ability of facilitating simultaneous processing outcomes when used in indexing tree or graph structured data. The primary need for having labelling schemes is to avoid expensive join operations when undertaking transitive closure computations as part of indexing. To achieve this, it is expected that such qualitative outcomes including determinacy, compaction, dynamicity, and flexibility will all be achieved (Li *et al.*, 2006b). Even though other forms of labelling schemes such as prefix-based schemes may successfully achieve all these outcomes, doing so simultaneously has always been a major challenge. But when such data structures as DAGs are introduced to represent subsumption hierarchies in multiplication-based schemes such as the prime number labelling scheme, it then becomes possible to achieve the preferred outcomes in a simultaneous manner.

Another strength with the use of multiplication-based labelling schemes is the fact that they are able to create tree edge relationships, which is absent in other labelling schemes such as prefix-based scheme (Gou and Chirkova, 2007). Meanwhile, when there is a non-tree relationship leading to a non-tree label, the labelling process does not have the strength of the deterministic tree label characters (Wu *et al.*, 2004). This means that the multiplication-based schemes which comes with a tree edge relationship and tree label do not need any special storage and additional efforts to facilitate the query processing (Amagasa *et al.*, 2003). On the whole, the multiplication-based scheme can be said to have a very rich re-labelling ability for updates as the tree labels trigger such abilities.

3.3.3.3 Weaknesses and Limitation

From the description and structure given above, there are number of limitations to the multiplication-based labelling scheme. In the first place, Harold (2004) saw the multiplication-based scheme as being costly in its computation processes. This is because multiplication-based labelling schemes make use of such complex

labelling parameters as atomic numbers and prime numbers (Cormen *et al.*, 2001). Gabillon and Fansi (2006) also saw a situation where multiplication-based schemes function based on subsumption hierarchies when they are applied in applications such as OO programming, software engineering and knowledge representation. Once this is done, a growing number and volume of DAGs are needed in the systems to support the demands that are expected to make the appropriate index structures for XML query processing functional (Zhang *et al.*, 2001).

These are all processes that come together to make the use of multiplication-based labelling schemes based on costly computations which may be discouraging for most novices who attempt to use XML data. In a related development, Hou, Zhang & Kambayashi (2001) saw that the costly computation processes associated with the use of multiplication-based labelling schemes makes it very difficult to apply to large scale XML documents. The reason for this assertion is that its computation process tends to make the size of resulting nodes very large and therefore impacting on labelling negatively. The reason for this is that the more computation processes are undertaken, the larger the size of the resulting scheme as internal updating takes place internally (Zhang *et al.*, 2001). This is why it is always difficult to apply multiplication-based labelling schemes on large-scale XML documents (Eda *et al.*, 2005).

There is a unique characteristic of multiplication-based labelling schemes that is often debated in literature as to whether it constitutes a strength or a weakness. This has to do with the ability of the multiplication-based scheme to establish a global order based on document order and the mapping of the self-label that are involved in the functioning of the node labels (Yu *et al.*, 2005). These global orders are formed on the basis of the 'simulation congruence' (SC) value as used by Harder *et al.* (2007). Tatarinov *et al.* (2002)opined that as far as the fact that global order makes the multiplication-based labelling scheme integrative and universal, it counts as an strength. However, this point is vehemently disagreed with other researchers such as O'Neil *et al.* (2004) who lamented that the only condition

under which the global orders become viable and useful is when simulation congruence (SC) value stay small. This is because the SC value has the potential of preventing scheme sizes from becoming very large when internal updating processes are going on. However, there was evidence with a study by (Schmidt *et al.*, 2002) who saw that the SC value rarely stays small because the list of SC values employed to determine the global ordering come in five nodes. As a result of this situation, the SC value that results in global orders has been identified to produce storage and maintenance that is very costly, especially in large XML documents (Wu *et al.*, 2004). The latter school of thought, that the limitations and demerits that the global orders produce far outweigh the benefits that are expected from them.

In addition to the points above, Cormen, Leiserson, Rivest and Stein (2001) saw the multiplication-based scheme as being slow in processing and implementation. This is largely due to the fact that when undertaking insertions and deletions with the various multiplication labelling schemes, there is the need to engage in recalculation of greater parts of the SC values. In one such instance, Fallside & Walmsley (2004) found that almost half of the SC values that are used with the Euler's quotient function are recalculated. In such instances, query processing will be very slow. Such lengthy processing, even though it may result in accurate results undermines the updating processing and updating frequency. Supporting this perspective, Tatarinov *et al.* (2002) observed that in the most basic form, updating of XML requires the computation of existing labels. But in cases where insertion and deletion of labels also demand recalculations, it would be expected that the updating process will not only demand computing of its existing labels but re-computations of the labels.

The re-computation process alone takes so much time, that it makes the use of multiplication-based labelling schemes inappropriate when time or efficiency is a priority in the query processing for XML data (Arion *et al.*, 2007).

3.3.4 Vector-based Labelling Schemes

3.3.4.1 Structure and Description

Zhang *et al.* (2001) described the vector-based scheme as having a static structure and requiring a global rebuilding of labels triggered by the occurrence of an update. This static nature notwithstanding, it has been found to have the ability to be intrinsically ordered and typically modelled as a tree (Bosak and Bray, 1999). The result is that vector-based schemes are able to function with documents that obey the XML standards. The ability to intrinsically order and model a tree ensures that like most other labelling schemes, the vector-based scheme encode not only the document order but also the structural information within the document (Yun and Chung, 2008). It is important so that the queries can exploit the labels without accessing the original XML file. Xu *et al.* (2009, 2007) was one of the authors who placed particular emphasis on the mechanisms that make the vector-based scheme functional in XML document query processing. It was explained that a vector code is a binary tuple that is expressed as (x, y), x being greater than zero. In the case of two vector codes, A: (x_1, y_1) and B: (x_2, y_2), the relationship $\frac{x_1}{y_1} \leq \frac{x_2}{y_2}$ must exist for vector A to precede vector B in the vector order. When a new vector C is inserted between vectors A and B, the vector code of C takes the form (x_1+x_2, y_1+y_2). Since the relationship $\frac{x_1}{y_1} \leq \frac{x_1+x_2}{y_1+y_2} \leq \frac{x_2}{y_2}$ is valid, the vector order is A<B<C (Xu *et al.*, 2007).

3.3.4.2 Strengths of Vector-based Labelling Schemes

One important strength or advantage of the use of vector-based schemes is that they are easily compatible with other labelling schemes (Gou and Chirkova, 2007). For example Xu *et al.*, (2009) discovered that it is possible to use the vector-based labelling approach together with interval-based and prefix-based labelling schemes. This possibility is an important strength with vector-based labelling scheme. This is because as has been seen with other labelling schemes discussed earlier and those that will be discussed later, there are always limitations with individual labelling schemes. The effect of these limitations on practical

experimentation and use of labelling schemes in XML documents is that functional outcomes with query processing that are limited may never be able to take place. In such situations, it is only expected that other labelling approaches may be used in addition to the substantive approach in achieving the limited function (Cunningham, 2006).

Unfortunately though, it is not always the case that labelling schemes allow this opportunity of being combined with other schemes. It is therefore very important that in vector-based schemes, it is possible to introduce other approaches, particularly the interval-based and prefix-based labelling schemes. Xu *et al.* (2012, 2009) actually discovered that some very specific schemes that are perfectly compatible with vector order approach are Dewey ID encompass Dynamic Dewey (DDE) and Compact-DDE (CDDE). Xu *et al.* (2012) also indicated that application of the vector labelling scheme to the interval containment-labelling scheme is possible but this happens largely as a V-containment and not an independently formed scheme.

3.3.4.3 Weaknesses and Limitation:

The strength discussed above gives an impressive outlook for the use of vector-based labelling scheme. However there are some key weaknesses and limitations with the use of vector-based labelling schemes in query processing of XML documents. In the first place, (Harold, 2004) lamented that even though it is a positive development that vector-based scheme can work with other labelling schemes, this comes with a major resultant challenge. During the combination process, the path may grow very large, having a significant effect on speed and updating efficiency (Gou and Chirkova, 2007). In what is a slight contradiction to the assertion just made, Tatarinov *et al.* (2002) argued that this increase in path growth does not always happen but is highly dependent on a number of factors including the complexity of the XML document in question. The latter therefore opines that when the XML document is complex, it is possible to achieve the combined labelling scheme approach without any difficulty with size and speed. Having said this, it must be acknowledged that the complexity with which web

activities and document reading are performed lately makes it necessary to make provisions for complex XML documents in any endeavour with query processing (Rousseeuw *et al.*, 1999).

The fact that there may be path growth with the use of vector-based labelling schemes depending on the complexity of the XML document cannot be overlooked as a limitation. This limitation also serves as a motivation for further work on the labelling scheme by emphasising on the need to achieve combined use of internal schemes without any effect on the growth or size of the path.

Another major weakness with the use of the vector-based labelling scheme as observed by Thonangi (2006) was the fact that there is often an outcome from the combined labelling scheme approach where re-labelling is prevented. Zhang *et al.* (2001) however emphasised that prevention of re-labelling only occurs when the need to undertake frequent updating contexts arises. It would however be unwise assume that in a typical modern setup the need to frequently update can be excluded in a typical query-processing scheme. It is in the light of this that the need to ensure that the necessary provisions for re-labelling under any circumstance with updating contexts is important.

Another weakness associated with the use of vector-based labelling scheme is that the identification of relationships between nodes requires computation (Cohen *et al.*, 2010). These computations may deter ordinary users of XML documents as they are mostly complex to implement. As it was mentioned in the initial sections of the chapter, XML documents are designed to be read by both human users and machine users of web content. If the exclusive reading is centred on machine users then it would be expected that the complicity with the identification of relationship between nodes which comes about due to the use of computations will not be a major challenge.

3.3.4.4 DDE Labelling Schemes

Xu *et al.,* (2009) created a new labelling scheme known as Dynamic Dewey (DDE) on the basis of the Dewey labelling scheme. The DDE scheme is capable of managing both static and dynamic XML documents. The scheme consists of labels that take the form of sequences of components to constitute a unique path from the document root to a node. In the case of a DDE label $\boldsymbol{a_1.a_2. \ldots .a_m}$, the parent label is $\boldsymbol{a_1.a_2. \ldots .a_{m-1}}$ while the local order is $\boldsymbol{a_m}$. Figure 3.8 illustrates that the DDE scheme and the Dewey scheme are identical with regard to the initial labelling.

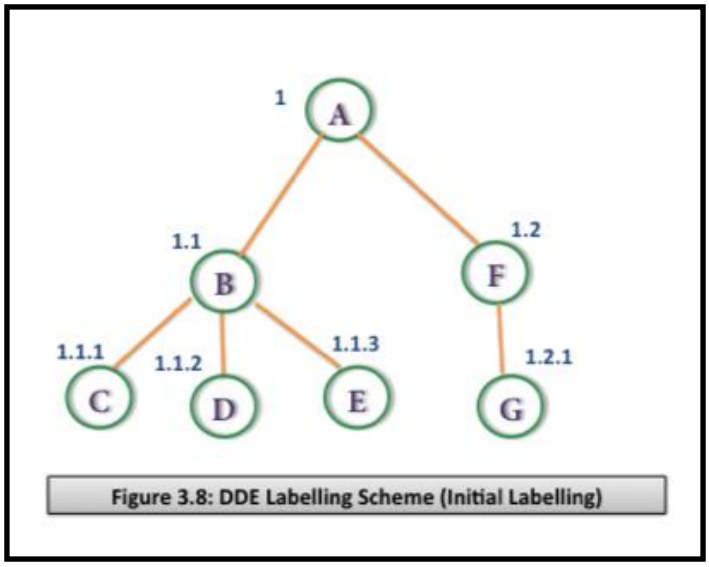


Figure 3.8: DDE Labelling Scheme (Initial Labelling)

The fact that the initial component of a DDE label is invariably a positive number is taken into account by DDE label ordering. This is clearly valid for the initial labels as the first component of all of them is 1.

As in the Dewey scheme, the level information is automatically stored by a DDE label as its number of components. Arbitrary insertions and deletions do not affect the validity of this property.

In the case of two DDE labels, *ddeA:* $a_1.a_2. \ldots .a_m$ and *ddeB:* $b_1.b_2. \ldots .b_n$, the labels exhibit the following properties:

- For *ddeA* to be an ancestor of *ddeB*, m<n and $\frac{a_1}{b_1} = \frac{a_2}{b_2} = \cdots = \frac{a_m}{b_m}$
- For *ddeA* to be the parent of B, *ddeA* has to be an ancestor of *ddeB* and m=n-1.
- For *ddeA* to take precedence over *ddeB* in document order, $A <_{dde} B$ where the following conditions must apply so that $A <_{dde} B$:
  - $m < n \; and \frac{a_1}{b_1} = \frac{a_2}{b_2} = \cdots = \frac{a_m}{b_m}$
  - $k \leq \min(m, n)$, $such\ that \frac{a_1}{b_1} = \frac{a_2}{b_2} = \cdots = \frac{a_{k-1}}{b_{k-1}} \; and \; a_k \times b_1 < b_k \times a_1$
- For *ddeA* to be a sibling of *ddeB*, m=n and $\frac{a_1}{b_1} = \frac{a_2}{b_2} = \cdots = \frac{a_{m-1}}{b_{m-1}}$

Since 1 is the first component of all Dewey labels, the above properties of the DDE labels can be applied in the scenario where $a_1 = b_1 > 0$ the same way as the Dewey labels. Given that the initial DDE labels are identical to Dewey labels, the scheme can be applied to static documents. However, the DDE labelling scheme was developed to avoid re-labelling in dynamic XML documents during the process of update. Figure 3.9 illustrates the ability of the DDE labelling scheme to handle several insertions in XML documents:

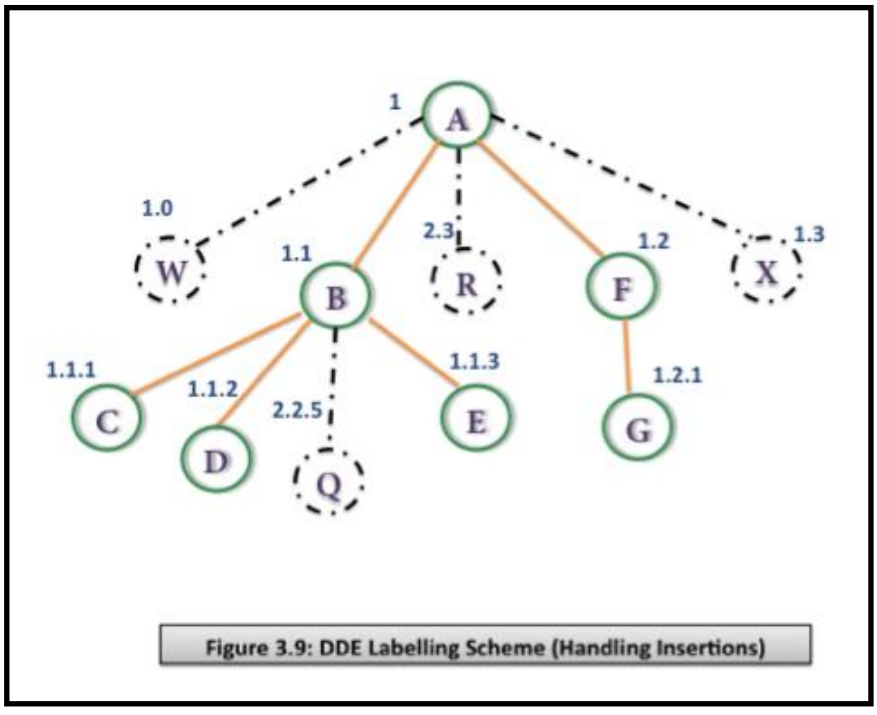


Figure 3.9: DDE Labelling Scheme (Handling Insertions)

3.3.5 Summary of major labelling schemes

However, a labelling scheme that supports only querying static XML would be disadvantageous. This is because although relationships of the nodes are efficiently determined by a static labelling scheme, dynamic updates which are essential to performance are not provided in static schemes (Lu, 2013). In Mesiti (2004), the combination of several schemes just to get the different advantages would not be realistic due to the storage space cost that it comes with. To counter this challenge, there is need to develop a robust XML labelling scheme that is applicable for both static and dynamic XML documents. To improve on the effectiveness of the dynamic XML scheme, many studies have focused on the need to maintain efficiency (Amagasa *et al.*, 2003, Cohen *et al.*, 2010, Eda *et al.*, 2005, Li and Ling, 2005a, Li and Ling, 2005b, O'Neil *et al.*, 2004, Wu *et al.*, 2004, Xu *et al.*, 2009). In all these cases the main idea was to eliminate or substantially reduce the need for re-labelling all the nodes. One such application of the reduction is in Mesiti (2004), who proposes the development of a dynamic labelling system that performs sparse labelling. In this method, a number of nodes around the updated position are randomly selected and labelled. Not all the nodes are re-labelled in the database thus the cost of bulk labelling and the re-labelling of the entire group of nodes is avoided. The reason for this is to allow for effective processing of the selected nodes within the time frame given for the update to take place as labelling several nodes within a limited time frame may affect effectiveness (Xu *et al.*, 2009). These earlier approaches meant that at the initial level, gaps would be created and then their main concern remained developing methodologies to maintain the efficiency of the schemes despite the gaps.

O'Neil *et al.* (2004) is an example of a scheme that leaves such gaps in the initial nodes. It describes a scheme called the ORDPATH labelling scheme. For initial labelling, this scheme solely employs positive, odd integers, while for subsequent insertions it uses negative integers. However, ORDPATH is not compact due to the gaps left, while the label insertions are made more complex by the insertion

mechanism that it uses. To enhance the compactness of labelling schemes and increase update performance, as well as to avoid having to leave gaps in the initial labels when processing updates in XML documents, Li *et al.* (2006a) developed a labelling scheme that involved converting the labels from their original format to a different ones, which are the updated versions.

The complexity of updating and querying is increased by the conversion of labels into dynamic formats, thus raising the cost of labelling. More recently, Xu *et al.* (2009) sought to enhance the encoding performance by developing two new labelling schemes for encoding dynamic XML trees on the basis of the mathematical operations of Dewey components. The Dewey component is a labelling scheme that has been tailored to perform in both static and dynamic XML documents (Warfield, (2010)). Although updating XML documents was demonstrated to improve the performance of the labelling schemes somewhat, the labels within them remained verbose, therefore increasing the cost of storage. Additionally, the insertion of nodes between two sequential siblings may make any of the four major labelling schemes which are prefix-based, interval-based, multiplication-based, and vector-based schemes inefficient.

To overcome this gap, it is expected that any new labelling scheme surfaces which ensures that even if the difficulty to overcome re-labelling persists, then the need for re-labelling all the nodes will be eliminated as part of the scheme. This is however something that cannot be guaranteed when using the interval-based scheme (Amagasa *et al.*, 2003).

3.4 Functional characteristics of ideal labelling schemes

The review and discussion on existing labelling schemes have clearly outlined the strengths and weaknesses associated with these schemes. But to have a better understanding of how the existing labelling schemes serve the purpose of XML labelling, it is important to review literature on what is seen as expected functional characteristics of labelling schemes. By so doing it will be possible to determine the extent of gaps that exists with the existing labelling schemes, especially their

weaknesses. In this section of the literature review, some important functional characteristics that are expected from labelling schemes to ensure they take advantage of the strengths identified, whilst overcoming the weaknesses are reviewed and discussed.

3.4.1 Time needed to determine the different relationships

When creating query processes, Rusty (Harold, 2004) noted that there is the need to ensure that the relationships between nodes can be established using labels. The authors of existing labelling schemes, researchers and developers of the schemes are silent on the time used in determining the different relationships that exists between the nodes. This is not to say that the functionalities of various relationships such as ancestor/descendant and parent/child relationships are not stressed. However, in the course of actually identifing the relationship, it is important that the time needed to determine the relationship is made very clear (Cunningham, 2006). The reason for emphasising time is that the timing used to establish the relationship could go a very long way to affect the overall efficiency of the labelling process (Rousseeuw *et al.*, 1999). This is because most labelling schemes would require the relationships to be determined before other procedures can follow in the query processing.

There is also the level, which explains the node's level within the XML tree where the document root level is given as one (Amagasa *et al.*, 2003). This relationship is also normally established even though Cunningham (2006) saw that order and level are very difficult to establish in most known schemes. Three other relationships that are very common with the previously discussed schemes are ancestor/descendant (AD), parent/child (PC) and lowest common ancestor (LCA). Together, these three relationships are determine based on specified rules. For this reason, Harold (Harold, 2004) identified these three relationships as some of the most time consuming when talking about time needed to determine the different relationships. Meanwhile, existing labelling schemes such as the vector-based schemes highly rely on individual relationship establishment between these three

relatives (Li and Moon, 2001). Finally there is the sibling relationship, which determines whether two nodes share the same parent node (Cooper *et al.*, 2001).

3.4.2 Queries' performance before and after insertions

Ideally, this could be said to be the most important outcome with any labelling scheme. This is because the overall goal of having a labelling scheme together with other schemes like indexing and numbering schemes is to ensure that query processing for XML data is facilitated (Li *et al.*, 2006a). In effect, the extent and level of query performance can be said to be the overall representation of the efficacy of the labelling scheme. Most of the labelling schemes discussed did not show much prospects when it comes to query performance for the XML data. Several factors account for this, including the structure and procedural functions of these schemes (Xu *et al.*, 2007). Most interval-based labelling schemes which have been discussed above can certainly be said to fail in addressing this expected characteristic. This is because these interval-based labelling schemes have a structure that makes it overwhelmingly complex to achieve efficient query performance (Milo and Suciu, 1999). To ensure that the query performance is not negatively affected, there are some obstacles that Eda *et al.* (2005) felt must be overcome. One of these is the need to avoid re-computation in the context of frequent updates. Again, Fennell (2013)indicated that the query performance must show a relatively constant competence at all levels of insertion, that means both before and after insertion.

A number of researchers have clearly outlined the parameters that may be used in determining query performance before and after the insertions. One of these was suggested to be the query response time before and after the insertions (Cunningham, 2006). By implication, it is expected that both before and after the insertions, the query response will not be seriously degraded. One other determinant of performance that was proposed in the literature was the need to have as wide a range of queries tested as possible (Gou and Chirkova, 2007).

The vector-based scheme was criticised for having a re-labelling performance that is compromised when the complexity of the XML document is high. It is against this backdrop that the need to ensure that as many queries with different complexities and objectivity are evaluated as possible.

Also writing on the examination of query performance, Yoshikawa, Amagasa & Uemura (2003) suggested that to obtain the best result with the query performance of any two given schemes before and after insertions, it is important that the evaluation of all the queries involved in the schemes are performed on the same platform.

3.4.3 Scheme's ability to handle different types of insertion

To activate query processing in an XML document, there is the need to perform several types of insertions of nodes (Gou and Chirkova, 2007). Some of these may be new nodes while others may be existing nodes. On the whole, Fisher *et al.* (2006)posited that one important characteristic feature that can be in any model labelling scheme is the ability of the scheme to handle different types of insertions. Meanwhile, from the beginning of the review, it will be noted that mention of insertion handling ability by the other labelling schemes has not been discussed. This exhibits a significant gap in literature. Because the ability to handle different types of insertion is important, some common types of insertions that are attributed to XML documents have been reviewed.

Alstrup and Rauhe (2002)mentioned uniform insertions as one of the types that must be handled effectively by any model labelling scheme. Uniform insertions mean insertions made on new nodes found between any two consecutive nodes. That is, when there are two existing nodes, an insertion of a new node made in-between the two forms a uniform insertion. Using the cases of multiplication-based scheme and prefix-based schemes as example, it can be noted that schemes' ability to handle uniform insertions has often been triggered by the numbering systems used by these. To effectively identify the ability to handle uniform

insertion, the time spent in executing the insertions and the new label's size after the insertion may be measured (Cooper *et al.*, 2001).

Ordered skewed insertions were the second types of insertions that ought to be handled by any ideal labelling scheme. The description of an ordered skewed insertion is that the insertion is done before or after a particular node repeatedly (Cohen *et al.*, 2010). This means that when there is an existing node and several new nodes are introduced either before or after the existing node, the scheme should exhibit an ability to handle the resulting insertion. Cormen, Leiserson, Rivest & Stein (2001) argued that the number of insertions made will be influential in determining the ease with which the scheme can handle the resulting insertion. Most forms of existing labelling schemes have failed to address this phenomenon because for them, there is a high rate of efficiency reduction when it comes to increasing the number of nodes that are introduced as part of the ordered skewed insertions.

The third type of insertion that was identified in the literature was random skewed insertion. Like the name suggests, random skewed insertion is said to occur when new nodes are randomly inserted between existing nodes (Li *et al.*, 2006b). The reason the scheme's ability to handle random skewed insertions will be said to be very important is that some of the existing labelling schemes have a fixed node structures that makes it difficult to introduce random insertions (Sean, 2006). An example of this is a prefix-based scheme.

3.4.4 New labelling scheme that is appropriate to support dynamic update

Throughout the review and discussion of other labelling schemes, two factors or parameters for ascertaining the strength and weaknesses of the schemes were re-labelling and dynamic update. These two have further been highlighted as being important for any labelling scheme that supports dynamic update (Fisher *et al.*, 2006). Writing on the subject, Alstrup and Rauhe (2002) noted that most modern usage of the World Wide Web requires frequent updates of the XML documents

which are used as the universal language between the web and human users. Because of this, an ideal labelling scheme will be one that supports dynamic updating, no matter what new nodes are added (Bosak and Bray, 1999). One serious challenge with most existing schemes is that even though they may allow updating, this is only limited to static XML documents (Amagasa *et al.*, 2003). Based on this understanding, it can be reiterated that a labelling scheme that exhibit a high sense of compatibility with dynamic updates whilst embracing updates with static XML is needed.

Fennell (2013) opined that a labelling scheme that supports only querying of static XML would be disadvantageous in this case. This is because although relationships of the nodes are efficiently determined by a static labelling scheme, the dynamic update that is essential to the issue of performance is not provided in static schemes (Lu, 2013). It is not surprising that even with some of the earlier labelling schemes such as LSDX, attention has been on the need to creating a labelling scheme which supports the process of updating XML data without having to re-label the existing labels (Xu *et al.*, 2007). This is because if existing labels have to be replaced before updating can take place, then the overall performance will be significantly slower (Cohen *et al.*, 2010).

3.5 Summary of the review

On the whole, the literature review revealed that these other labelling schemes have unique structures and characteristics that make it possible for them to perform the roles in query processing as far as the use of labelling is concerned (Milo and Suciu, 1999). Even more, most of these have a strengths which make the selection of one form of labelling scheme over the other possible, based on the specific goal that a researcher or an experimenter may be aiming to achieve.

For example, the prefix-based labelling scheme has the strength of forming a group based scheme, while the multiplication-based schemes have the strength of achieving simultaneous processing outcomes without having to implement

individual schemes to come out with the processing outcomes or factors (McCreight, 1976, Milo and Suciu, 1999).

The literature review has also outlined the weaknesses and limitations of existing labelling schemes. Some recurring weaknesses such as issues with updating efficiency, insertion of new nodes and maintaining performance of queries performance before and after insertion were found with almost all existing labelling schemes. It was rightly appreciated in literature that older labelling schemes have constantly been updated and improved but most of these weaknesses persist (Li and Moon, 2001). A typical example of such weakness was the need to engage in regular re-computing in the XML data whenever there was a deletion and/or insertion (Goldman and Widom, 1997). Apart from the fact that the review showed that such a re-computing processes was expensive, it was also time consuming and thus degraded efficiency.

On the whole, four major expectations were set and reviewed. The first was the time needed for the initial labelling process to the take place along with the label's size. The second was on the time needed to determine the different relationships existing in the nodes, while the third focused on queries' response time before and after insertions took place. The final metric is that any proposed scheme must have the ability to handle different types of insertion (Kaplan *et al.*, 2002). The next chapter of the thesis is presents the proposed scheme in detail.

3.6 Conclusion

In this chapter, the various aspects of efficiency and effectiveness of the different existing schemes have been discussed. As demonstrated in the discussion, the proposed labelling scheme will borrow the strengths of several schemes that have been reviewed by several studies.

Meanwhile, the fact that there is massive and active human usage in web processing as manifested through the use of XML documents cannot be

underestimated. There is therefore a motivation to work out on a new labelling scheme which makes the identification of relationships between nodes less dependent on computations.

The schemes discussed in this chapter also form a basis for evaluating any proposed scheme. The specific weaknesses that have been identified in these schemes and their mitigation form strong grounds for the selection of the modalities in dealing with the related issues of schemes like re-labelling in the case of an update. The characteristics that the proposed labelling scheme aims to provide are highlighted.

Chapter 4: GroupBased Labelling Scheme for Dynamic XML Databases

4.1 Introduction

Labelling a node of an XML document to reflect the structure is an important process that helps in indexing and retrieving XML data effectively. However, designing a dynamic labelling scheme which can handle insertions of new nodes without the need to re-label the existing labels, as well as taking the size of the labels and the query performance into consideration, is a challenging task; this was mentioned earlier in the literature chapter (Ch. 3).

This chapter presents the principles of the dynamic labelling scheme proposed in this thesis before the design and implementation details are examined in Chapter 5. In this chapter, an overview is given in Section 4.2. Then, Section 4.3 illustrates how the initial labels are allocated and how the different relationships are determined. Section 4.4 describes how insertions are handled and how different relationships are preserved. A validation of the relationships using algebra is shown in Section 4.5. Finally, in Section 4.6, the chapter ends with a general conclusion that leads to the following chapter which discusses the scheme from the point of view of implementation.

4.2 An Overview of the Scheme

The proposed scheme is based on the parent-child grouping to facilitate faster identification of parent-child and sibling relationships, based on a simple comparison. Parent-child grouping was also selected due to the fact that all XML documents come with this type of relationship (Goldman and Widom, 1997). Again, parent-child and sibling grouping facilitate smoother insertions of new nodes, given the fact that in this form of grouping only a simple tree structure will be dealt with rather than the whole tree (Kaplan *et al.*, 2002). Gusfield (1997) also observed that when dealing with parent-child groupings, labelling can be thought of as being easier, faster and more accurate as it deals with a simple tree structure. The simple structure has to do with a root node and its direct child nodes. The advantage of allowing smoother insertion builds on the prefix GroupID labelling scheme but does not restrict the number of nodes that can be inserted.

Another critical characteristic of the scheme is that it uses two labels for each node in order to facilitate the processing of nodes within the same group using their simple local labels. Where as the global label is used to connect a group to the whole tree, which helps in identifying relationships between nodes, which belong to different group (Milo and Suciu, 1999). Based on existing schemes such as the DDE labelling scheme (Xu *et al.*, 2009), in this scheme to create the first part global label has its advantage as it facilitates the insertion without re-labelling of the existing labels. It also ensures the identification of all relationships.

What is more, the scheme is designed to allow fast identification of relationships. This is because as fast as the relationships between nodes can be determined, the query processing will be optimised (Li and Moon, 2001).

This labelling scheme is divided into two parts. Each label has a local and a global part. The local label, which is given to each node, can be duplicated although not within a group whereas the global label uniquely identifies a group of local labels.

This is generated based on the Dynamic Dewey labelling scheme (Xu *et al.*, 2009). Xu's scheme is a Dewy labelling scheme (Tatarinov *et al.*, 2002) used when the document is static; i.e. when no insertions have occurred. Each node has local and global labels and they are assigned as follows:

- Every node except the root node is grouped with its child nodes and is given a global label.

- The local label is assigned for each node within a group starting from the parent node; then, the child nodes are labelled in a serial order.

- The root node refers to the document root. And the child nodes for a specific node are the immediate child nodes without the grandchildren or further descendants (i.e. the nodes that have direct parent/child relationship with a specific node)

4.3 The Initial Labelling

Firstly, '1' is assigned to the document root as its global and local labels. Then two phases of the process are performed.

Phase 1:

This starts by grouping every node and its child nodes to form a sub-tree. Each sub-tree is given a global label, which consists of two components if the node is not a child of the root. The first component is the Dewey label. The second one is either the number of the child node, starting from left to right *1,2,3…*i^{th} where i^{th} is the last child node; or it is information about a new inserted node when random skewed insertion has occurred. This second component of the global label is used to preserve the document order after insertions have been made. More details are provided in the next section.

Phase 2:

This phase involves assigning a local label to each node where all the document root's child nodes have the same local label, which is '1.0'. Then, the local label of the first child within a group is calculated by incrementing its parent's local label by one; the next sibling node's local label is then derived by adding one to the previous sibling local label and so on until the end of the document. Figure 4.1.1 shows the initial labelling of the scheme and the nodes' full labels are presented in Figure 4.1.2 and Table 4.1

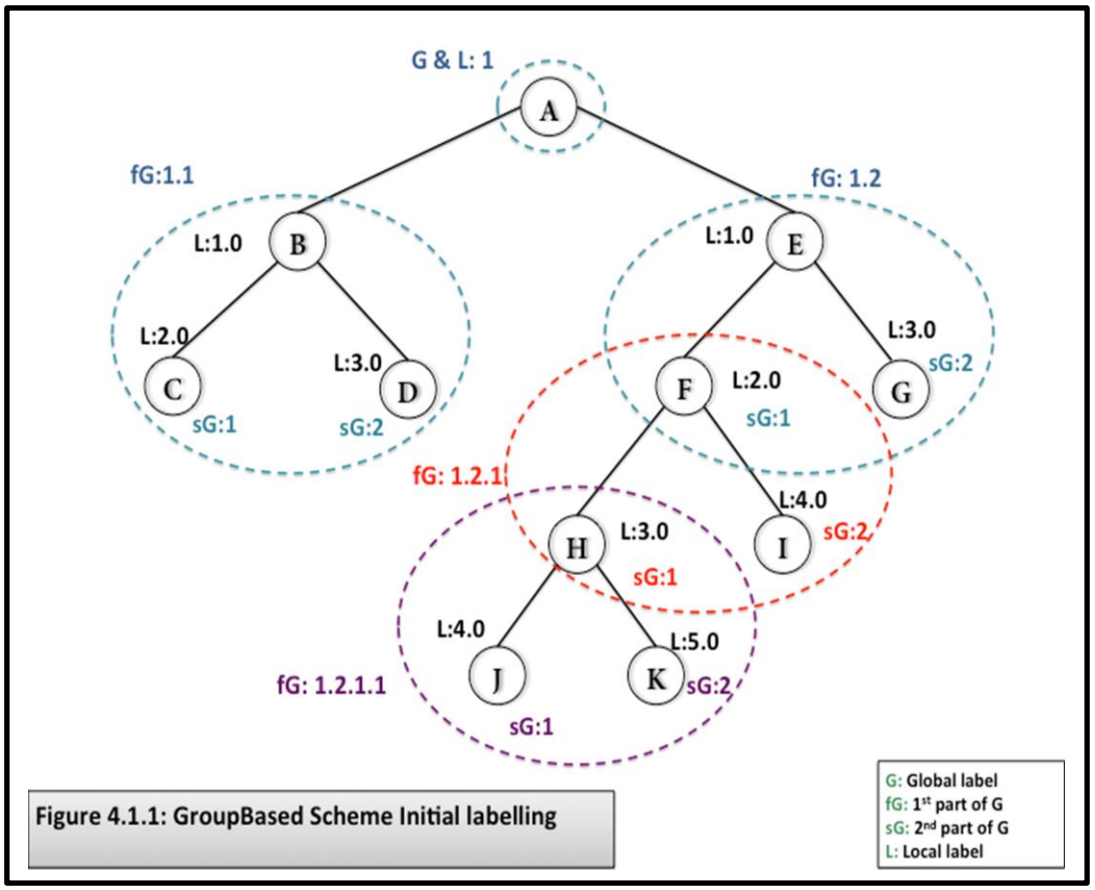


Figure 4.1.1: GroupBased Scheme Initial labelling

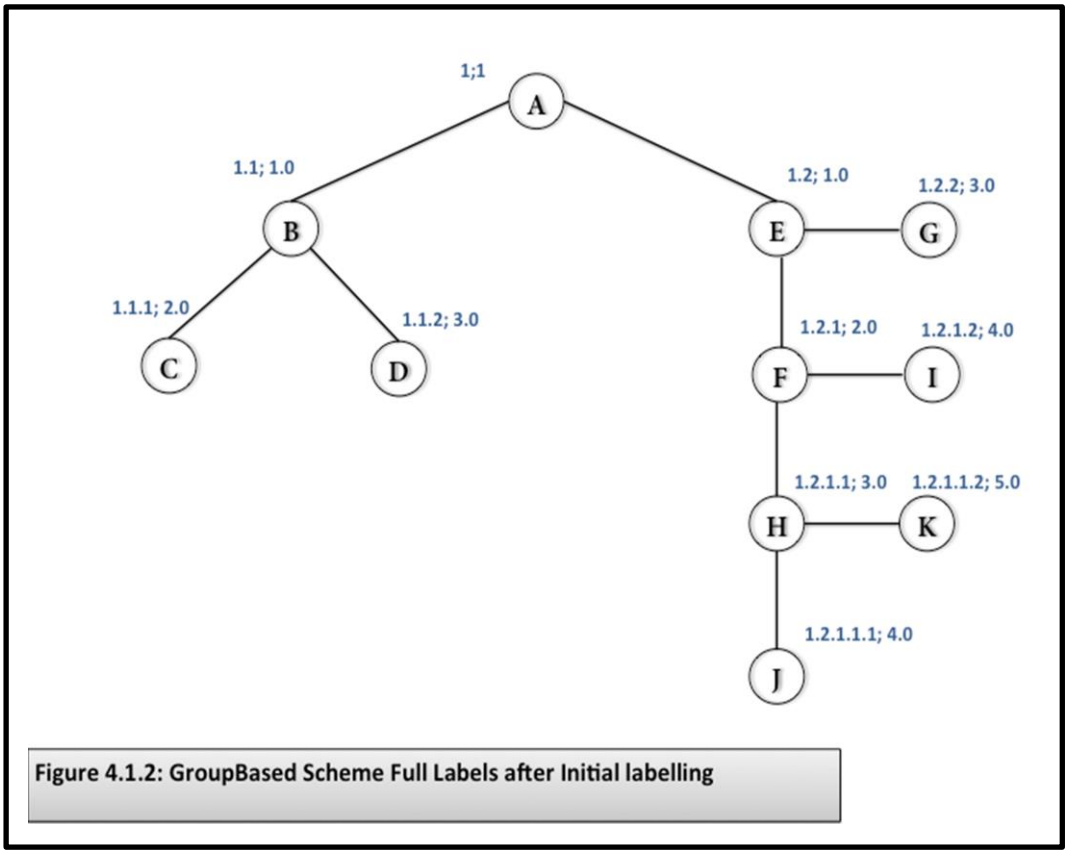


Figure 4.1.2: GroupBased Scheme Full Labels after Initial labelling

Node Name	1st part of the Global Label (Mesiti *et al.*)	2nd part of the Global Label (SG)	Complete Global Label	Local Label (L)
A	1	null	1	1
B	1.1	null	1.1	1.0
C		1	1.1 , 1	2.0
D		2	1.1 , 2	3.0
E	1.2	null	1.2	1.0
F		1	1.2, 1	2.0
G		2	1.2, 2	3.0
H	1.2.1	1	1.2.1 , 1	3.0
I		2	1.2.1, 2	4.0
J	1.2.1.1	1	1.2.1.1 , 1	4.0
K		2	1.2.1.1 , 2	5.0

Table 4.1: The GroupBased Scheme Initial labelling for XML tree in Fig.4.1

As seen in Figure 4.1.1: A node can belong to two groups, which seems to overlap. However, this is not an issue because when such overlap occurs, the node can be treated in two ways as required to handle the situation. The first is to handle the overlapped node as a child node in a group, while the second is to handle it as a root node in another group (Zhang *et al.*, 2001). For example, given a node H which belongs to a group with label 1.2.1 as child node, this same node is also a root in

another group which is labelled 1.2.1.1. In this example, the group label where H is a root can be seen to have been assigned by simply concatenating the H node's first part of the global label (Mesiti *et al.*) where it appears as child node and its second part of the global label (SG). Thus, from this explanation, two nodes belong to the same group based on the following definition:

Definition 1:

n_1 and $n_2 \in$ **same group** *if, and only if, one of the two following conditions holds:*

1. Their 1st part of their global labels are the same
2. The 1st part of the global label of one of them was extrapolated from the global labels of the other one.

e.g. from Fig 4.1.1 and Table 4.1, 'J' ,'K' and 'H' ∈ same group because the FGs of 'J' and 'K' are the same which is 1.2.1.1. And was extrapolated by concatenating the FG and SG of 'H'; ie: "12.1" +".1" .

4.3.1 The Scheme's Properties

Given two nodes, n_1, n_2, with $level_1$, $level_2$ as their levels (the level refers to the level of the element node in the XML tree), and with labels A and B, where global labels are $a_1.a_2...a_m$, i^{th} *and* $b_1.b_2...b_n$, , j^{th} respectively and their local labels are La_1**.** La_2 and Lb_1**.** Lb_2, the label properties can be defined as in the following:

- <u>*Node Level:*</u>

  The level information of each node can be derived from its global label as follows:

  The level is the number of components in the first part of the node's global label plus 1 if the second part of the global label exists; i.e. if the SG equals null, the level is the number of component in FG.

  e.g. As shown in Fig. 4.1.1 and Table 4.1, the level of node 'B' is (Kasim *et al.*), whereas the level of node 'J' is (5) based on their global labels, as shown in Table 4.1.

- *Label Order:*

Case 1: label order between nodes within the same group; i.e. the first parts of their global label are equal or the global label of one of them forms the first part of the global label of the other. In this case, the order is based on the nodes' local labels and can be simply determined as follows:

Definition 2:

n_1 (is before) n_2 if, and only if, one of the two following conditions holds:

C_1: $level_1 < level_2$

C_2: $level_1 = level_2$ and *$La_1.La_2 < Lb_1.Lb_2$*

e.g. from Fig. 4.1.1 and Table 4.1, node 'B' with level (Kasim *et al.*) is before node 'D' as its level is (3), and node 'H' is before 'I' as 'H' local (3.0)< 'I' local (4.0).

Case 2: label order between nodes within a different group. In this case, the order is based on the first part of nodes' global labels and is determined using the DDE pre-order definition (Xu *et al.*, 2009) which states that:

Given two DDE labels, *ddeA*: $a_1.a_2.\ldots.a_m$ and *ddeB*: $b_1.b_2.\ldots.b_n$

***A $<_{dde}$ B if and only if one of the following apply*:**

- $m < n$ *and* $\frac{a_1}{b_1} = \frac{a_2}{b_2} = \cdots = \frac{a_m}{b_m}$
- $k \leq \min(m,n)$, *such that* $\frac{a_1}{b_1} = \frac{a_2}{b_2} = \cdots = \frac{a_{k-1}}{b_{k-1}}$ *and* $a_k \times b_1 < b_k \times a_1$

Adopted from (Xu *et al.*, 2009)

e.g 1. from Fig. 4.1.1 and Table 4.1, node 'C' $<_{DDE}$ node 'G' because :
FG for node 'C' → 1.1 & FG for node 'G' → 1.2 , thus,
As their FGs consist of two components → the minimum (2,2) = 2 such that *k* can be equal 2 as it <= 2, such that $\frac{a_1}{b_1} = \frac{a_2}{b_2} = \cdots = \frac{a_{k-1}}{b_{k-1}} \rightarrow \frac{1}{1} = \frac{1}{1}$ and a_1=1, a_k=1 , b_1=1, b_k=1 → $1 \times 1 < 2 \times 1 \rightarrow 1 < 2$ and the 'G' $<_{DDE}$ 'C' is false as a_1=1, a_k=2 , b_1=1, b_k=1 → $2 \times 1 < 1 \times 1 \rightarrow 2 > 1$

e.g 2. Assuming node 'W', which is not present in the tree, is the first child of node 'I', its FG will be 1.2.1.2 and from Fig. 4.1.1 and Table 4.1, node 'J with FG 1.2.1.1$<_{DDE}$ node 'Q' because :
As their FGs consist of four components → the minimum (4,4) = 4 such that *k* can be equal 4 as it <= 4, such that $\frac{a_1}{b_1} = \frac{a_2}{b_2} = \cdots = \frac{a_{k-1}}{b_{k-1}} \rightarrow \frac{1}{1} = \frac{2}{2} = \frac{1}{1}$ and a_1=1, a_k=1 , b_1=1, b_k=2 → $1 \times 1 < 1 \times 2 \rightarrow 1 < 2$

- *Ancestor/Descendant (AD) Relationship:*

Definition 3:

n_1 (is ancestor of) n_2 if, and only if, one of the two following conditions holds:
C_1: $level_1 < level_2$ and *m <= n: n_1 & n_2 ∈ same group*
C_2: $level_1 < level_2$ and *m < n* ,such that, *n_1* global label → FG.SG → $a_1.a_2....a_m$ and *n_2* global label → FG.SG → $b_1.b_2....b_n$
where $\frac{a_1}{b_1} = \frac{a_2}{b_2} = \cdots = \frac{a_m}{b_m}$.

If *n_1* is the document root, which means *m=1*, AD is true by default as the document root is ancestor to any other node.

e.g. from Fig 4.1.1 and Table 4.1, 'F' is an ancestor of 'H' as they are both in the same group and 'F' level < 'H' level → (*C_1 applies*). However, based on (*C_2*),'E' is an ancestor of 'K' such that, global label of 'E' is 1.2 (as 'E' doesn't have SG) and the global label of 'K' is 1.2.1.1.2 which is the result of concatenate 'E' FG and SG. Thus, $\frac{a_1}{b_1} = \frac{a_2}{b_2} = \cdots = \frac{a_m}{b_m} \rightarrow \frac{1}{1} = \frac{2}{2}$

- *Parent/Child (PC) Relationship:*

  Case 1: n_1 & n_2 ∈ *same group.* The following definition applies.
  Definition 4: n_1 *(is parent of)* n_2 *if, and only if,* n_1 *(is ancestor of)* n_2 *under* C_1 *of Definition 2.*

  Case 2: n_1 & n_2 ∉ *same group* where n_1 is the root document and n_2 its child, such that, n_2 level = 2.

  e.g. from Fig. 4.1.1 and Table 4.1, node 'E' is the parent of node 'G' as 'E' is an ancestor of 'G' based on C_1 of *Definition 2*. Furthermore, node 'A' is the parent of node 'E' as 'A' is the root document and 'E' level =2.

- *Sibling Relationships:*

  Definition 5:
  n_1 & n_2 are siblings if, and only if, one of the two following conditions holds:

  C_1: $level_1 = level_2$ and n_1 & n_2 ∈ *same group (i.e.* $a_1.a_2...a_m = b_1.b_2...b_n$*):*
  a.FG /b.FG =1.
  C_2: $level_1 = level_2$ =2 *(i.e. the root document is their parent).*

  e.g. from Fig. 4.1.1 and Table 4.1, node 'C' & node 'D' are siblings as they belong to the same group and their levels are equal. Node 'B' and node 'E' are also siblings as their level =2.

- *Lowest Common Ancestor (LCA):*

  Definition 6:
  The LCA between n_1 and n_2 is n_3 where the global label of n_3 is $d_1.d_2...d_k$, i^{th} and n_3 is an ancestor of both nodes and one of the following two conditions also apply:

C_1: $\frac{d_1}{b_1} = \frac{d_2}{b_2} = \cdots = \frac{d_k}{b_k}$: k is the *min (m,n).*

C_2: n_1 , n_2 and n_3 $\in$ *same group :* $a_1.a_2...a_m = b_1.b_2...b_n = d_1.d_2...d_k$ *: && * n_3 is the parent node *of* n_1 and n_2

*e.g*1. from Fig. 4.1.1 and Table 4.1, the LCA between node 'J' and node 'I' is node 'F' where their global labels (FG *concatenate SG (if SG exists)* are *1.2.1.1.1 , 1.2.1.2 and 1.2* respectively, and 'F' is the ancestor of both 'J' and 'I' from ***Definition 3***. C_1 applies such that, the number of components in 'J' and 'I' global labels is 5 and 4 respectively → the minimum (5,4) = 4; and the following is true $\frac{1}{1} = \frac{2}{2}$.

*e.*g2. Node 'B' is the LCA between node 'C' and node 'D' based on C_2 where the first part of their global labels are equal and 'B' is an ancestor of both of them based on C_1 *of* ***Definition 3***.

4.4 Handling Insertions

This section shows how the re-labelling of the existing nodes is avoided during different types of insertion (Figures 4.2-4.6).

- *Leftmost Insertion:* Insert a new node, n_x, before n

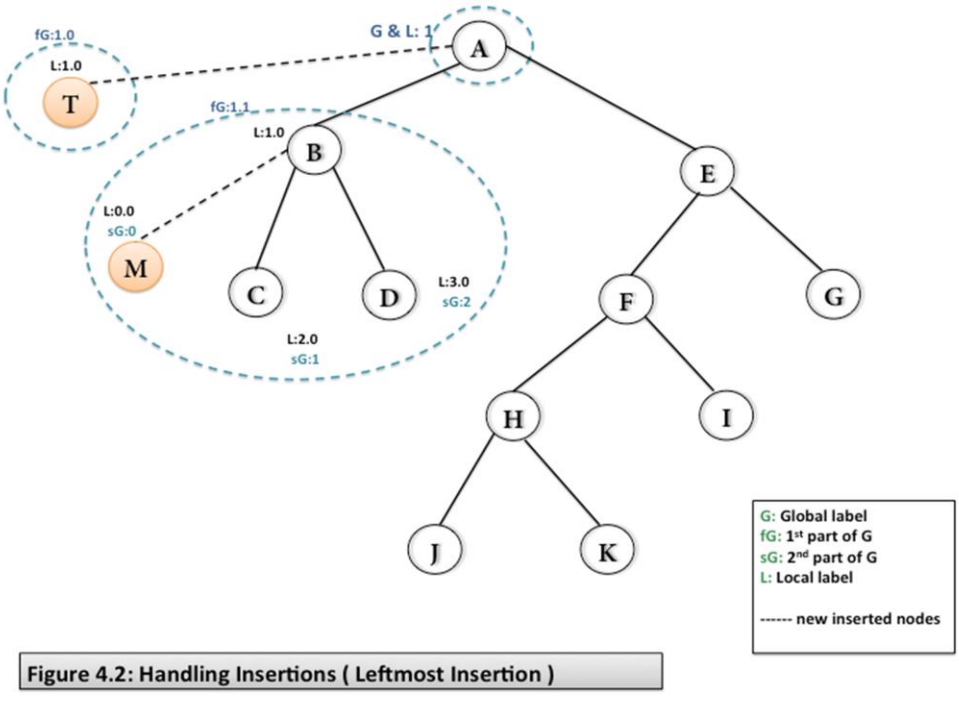


Figure 4.2: Handling Insertions (Leftmost Insertion)

Case 1: If n is a child of the document root, a new group is created where n_x is the root and the first part of the global label is set based on the DDE scheme's leftmost insertion mechanism, as described below:

DDE Leftmost Insertion:

As *A* is the first child of a node, when a new node is inserted before the node *A*: $a_1.a_2.\ldots.a_m$ where the label of the new node will be $a_1.a_2.\ldots.(a_m-1)$.

Adopted from (Xu *et al.*, 2009)

As with the other child nodes of the document root, the local label of this n_x is '1.0' and the second part of its global label is ***null***: *e.g.* node 'T' in Fig. 4.2.

Case 2: In the case of inserting n_x within a group where *n* is the first child, the node's local label is calculated by decrementing the first component of the *n* local by one. Then, if the resulting local equals the local of the parent node, the same component must be decremented again. However, the node's global label is set as follows:

- *FG:* is the same as the FG of the node *n* because they are siblings and elements within the same group.
- *SG:* is calculated by decrementing the SG of *n* by one:

e.g. node 'M' in Fig. 4.2.

- *Rightmost Insertion:* Insert a new node, n_x, after *n*

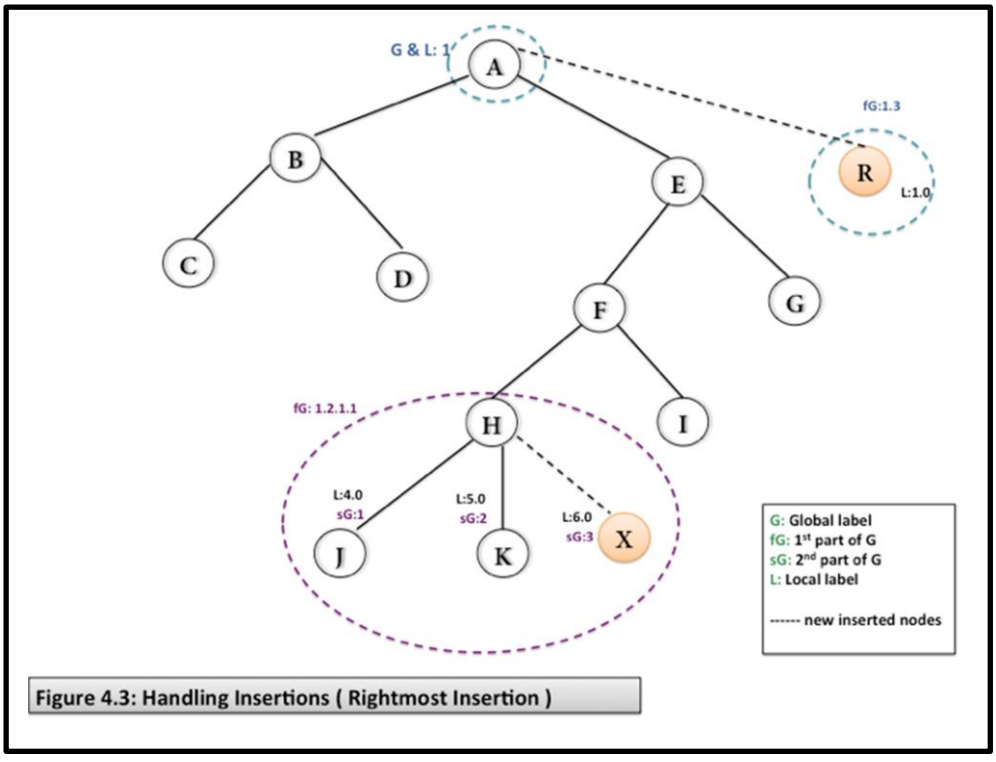
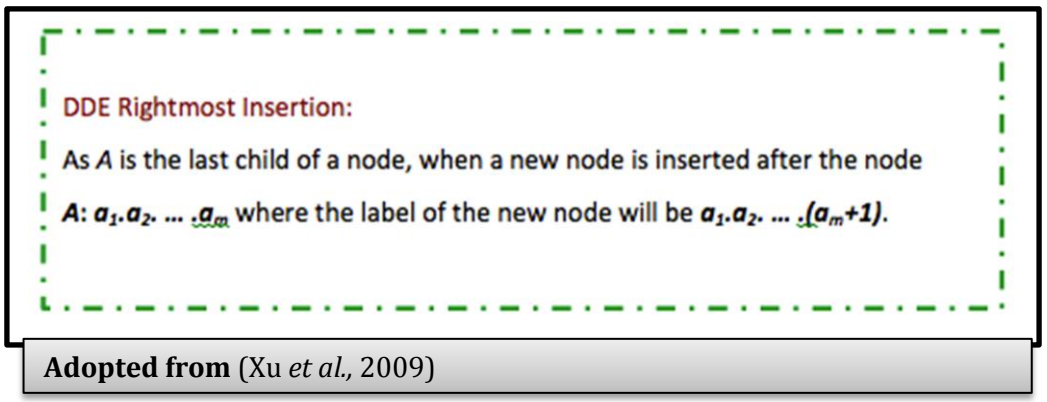


Figure 4.3: Handling Insertions (Rightmost Insertion)

Case 1: If *n* is a child of the document root, the procedure outlined for Case 1 in the leftmost insertion applies in terms of creating a new group where n_x is the root, n_x local label is '1.0' and the second part of its global label is ***null***. *But the first part of* n_x global label is set based on DDE's rightmost insertion mechanism, as follows:



Adopted from (Xu *et al.,* 2009)

e.g. node 'R' in Fig. 4.3.

Case 2: In the case of inserting n_x within a group where *n* is the last child, the node's local label is calculated by incrementing the first component of the local of *n* by one. However, the node's global label is set as follows:

- *FG:* is the same as for the first part of node *n*.
- *SG:* is calculated by incrementing the second part of *n* by one:

e.g. node 'X' in Fig. 4.3.

- *Insertion between two siblings:* Insert a new node, n_x, between n_1 *and* n_2

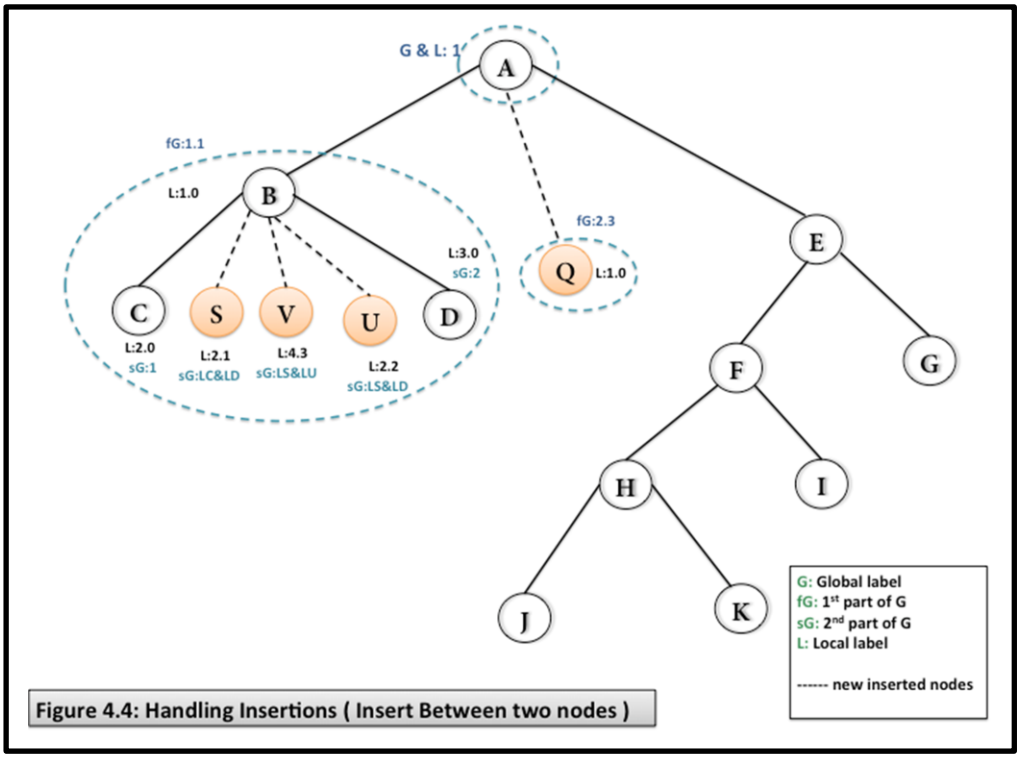


Figure 4.4: Handling Insertions (Insert Between two nodes)

Case 1:

If the document root is the parent node of n_1 and n_2, the procedure noted in Case 1 for the leftmost insertion applies in terms of creating a new group where n_x is the root, n_x local label is '1.0' and the second part of its global label is ***null***. *But the first part of* n_x global label is set based on the DDE insert-between mechanism as follows:

DDE Insert-Between:

- When a new node is inserted between two consecutive siblings A and B, the label of the new node will be ***A+B (i.e. is the result of adding each component in A to its correspondence in B)***.

Adopted from (Xu *et al.*, 2009)

Thus, n_x FG is the result of adding each component in n_1 FG to its correspondence in n_2 FG.

e.g. node 'Q' in Fig. 4.4.

Case 2:

In the case of inserting n_x within a group, and if n_1, n_2 were labelled during the initial labelling (i.e. they exist in the original document before any insertion), or n_1 is newly inserted and n_2 is not, the procedure used in Case 2 for the rightmost insertion applies for the local label and the first part of the global label. However, the second part of the global label holds references of both n_1 and n_2 locals.

e.g. node 'S' between 'C' & 'D' , then node 'U' between 'S' & 'D' in Fig. 4.4.

Case 3:

In the case of inserting n_x within a group, and when n_1, n_2 are themselves newly inserted, the node's local label is calculated by adding each component of the local of n_1 to its corresponding n_2 local. On the other hand, the node's global label is set as in Case 2:

e.g. node 'V' between 'S' & 'U' in Fig. 4.4.

- *Insertion Below a Leaf Node:* Insert new node, n_x below *n*.

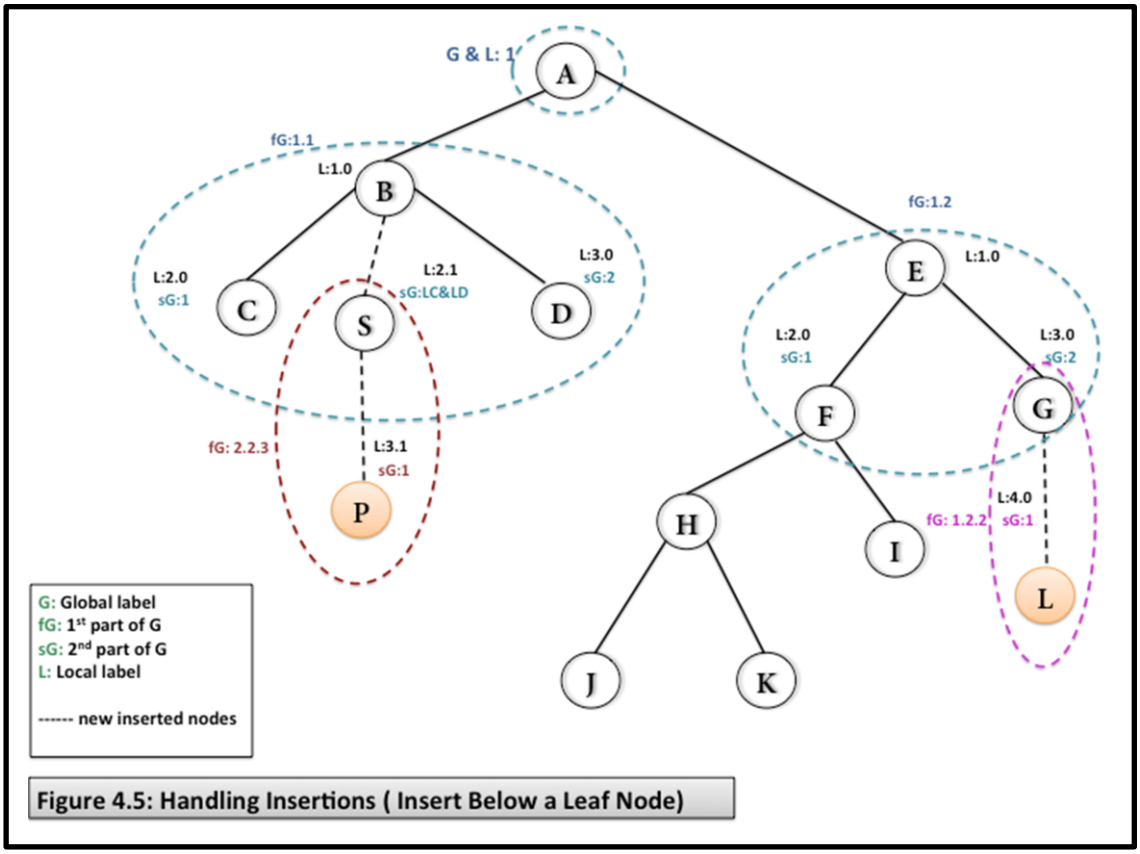


Figure 4.5: Handling Insertions (Insert Below a Leaf Node)

Case 1: if the leaf node initially exists or it is a child of the document root:

A new group is created and given a global label, which is a concatenation between the first and second parts of *n's* global label. However, the node's local label is calculated by adding one to *n's* local label and its SG is set to '1':*e.g.* node 'L' in Fig. 4.5.

Case 2: if the leaf node is inserted between two nodes within a group:

This is similar to **Case 1** in terms of SG and the local label. But the FG is set by adding the FG of the left node's child node to the FG of the right node's child node; these FGs can be extrapolated if the nodes are leaf nodes.

e.g. node 'P in Fig. 4.5; the child(s) nodes of 'C' and 'D' have '1.1.1' and '1.1.2' as their FGs respectively. Thus, the FG of node 'P' is (1+1 . 1+1 . 1+2 = 2.2.3). This also indicates that 'S' and 'P' ∈ **same group** as FG of 'P' was extrapolated from 'S' global labels where the SG of 'S' holds references.

Figures 4.6.1 and 4.6.2 show the XML tree in Figures 4.1.1 and 4.1.2 respectively after all types of insertion and Table 4.2 shows the labels after insertions.

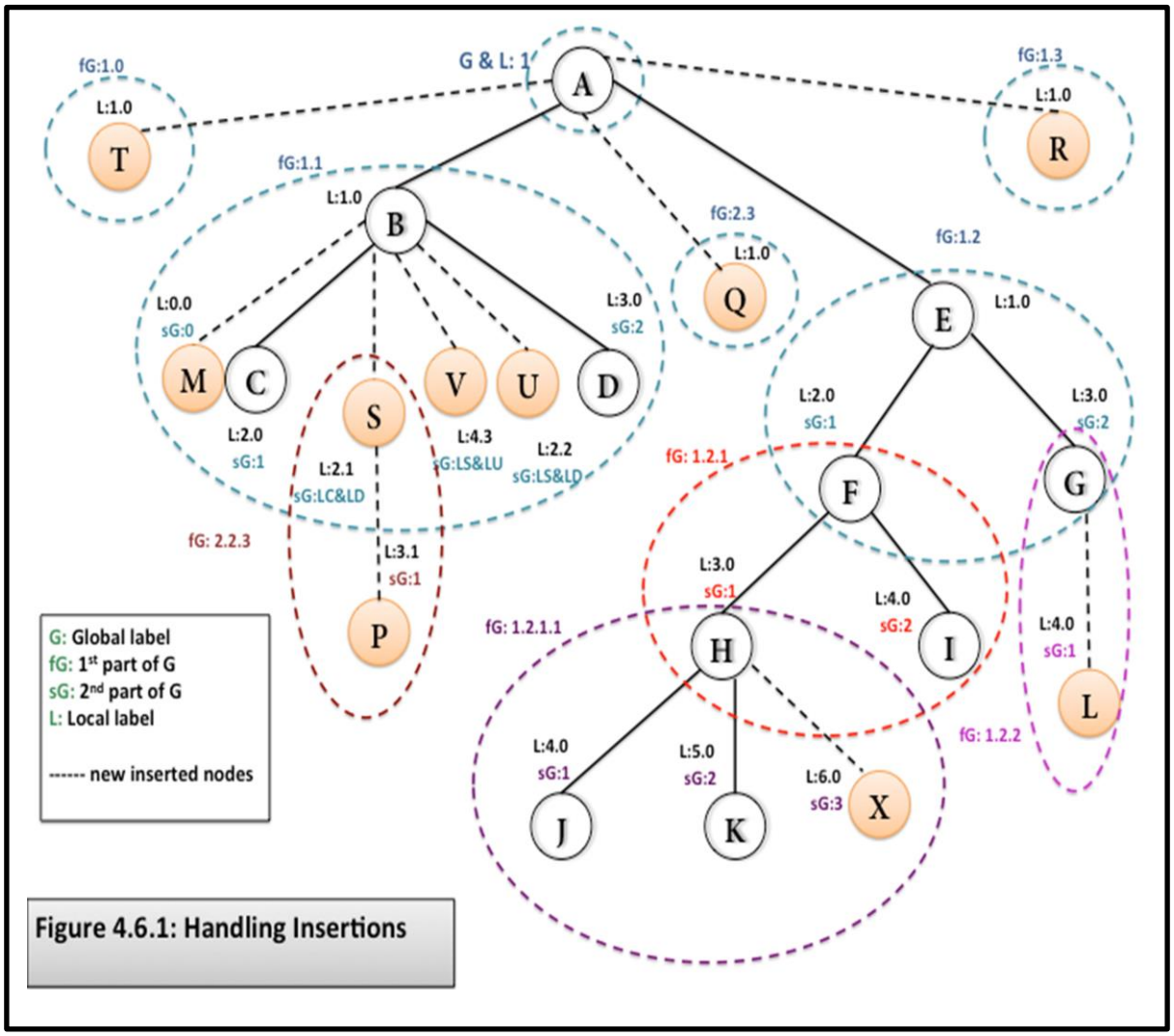


Figure 4.6.1: Handling Insertions

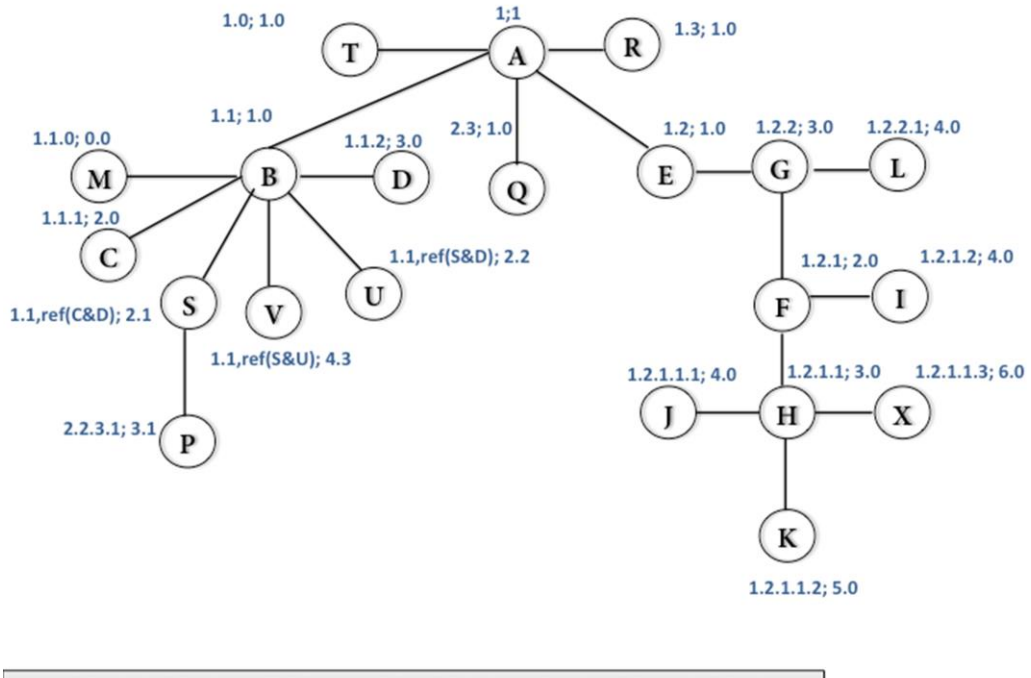


Figure 4.6.2: GroupBased Scheme Full Labels after Insertions

Node Name	1st part of the Global Label (Mesiti *et al.*)	2nd part of the Global Label (SG)	Complete Global Label	Local Label (L)
A	1	null	1	1
T	1.0	null	1.0	1.0
R	1.3	null	1.3	1.0
Q	2.3	null	2.3	1.0
B	1.1	null	1.1	1.0
C		1	1.1 , 1	2.0
D		2	1.1 , 2	3.0
M		0	1.1 , 0	0.0
S		Ref(C&D)	1.1, Ref(C&D)	2.1
U		Ref(S&D)	1.1 , Ref(S&D)	2.2
V		Ref(S&U)	1.1 , Ref(S&U)	4.3
E	1.2	null	1.2	1.0
F		1	1.2.1 , 1	2.0
G		2	1.2, 2	3.0

Node Name	1st part of the Global Label (Mesiti *et al.*)	2nd part of the Global Label (SG)	Complete Global Label	Local Label (L)
H	1.2.1	1	1.2.1 , 1	3.0
I		2	1.2.1, 2	4.0
J	1.2.1.1	1	1.2.1.1 , 1	4.0
K		2	1.2.1.1 , 2	5.0
X		3	1..2.1.1 , 3	6.0
L	1.2.2	1	1.2.2, 1	4.0
P	2.2.3	1	2.2.3, 1	3.1

Table 4.2: The GroupBased Scheme Labels after Insertions

4.4.1 The Scheme's Properties after Insertions:

When n_1 and n_2 are inserted between two nodes, the SGs of their global label are (i_1, i_2) and (j_1, j_2).

- *Node Level:*
  The same as before insertions.

  e.g. As shown in Fig. 4.6.1 and Table 4.2, the level of node 'U' is (Kasim *et al.*), whereas the level of node 'P' is (4) based on their global labels, as shown in Table 4.2.

- *Label Order:*

Case 1: label order between nodes within the same group. In this case, the order is based on either the nodes' local labels or the second part of their global labels as follows:

Definition 2.1:

n_1 (*is before*) n_2 ,if and only if, one of the four following conditions holds:

C_1: $level_1 < level_2$

C_2: $level_1 = level_2$ and $La_1 . La_2 < Lb_1 . Lb_2$: n_1 & n_2 are initially labelled ***OR*** n_1Local & n_2Local are not resulted from locals addition.

C_3: $level_1 = level_2$ and n_1 & n_2 are inserted (not initially labelled) and at least one of their locals is resulted from locals addition → $La_2 \times Lb_1 < La_1 \times Lb_2$

C_4: $level_1 = level_2$ and n_2 local is resulted from locals addition and n_1 is initially labelled; $aSG \times Lb_2 < La_1 \times Lb_1$

C_5: $level_1 = level_2$ and n_1 local is resulted from locals addition and n_2 is initially labelled; $La_2 \times bSG <= La_1 \times Lb_1$

Note: SG is the second part of n_2 global label.

e.g. from Fig 4.6.1 and Table 4.2, node 'C' is before node 'D' as 2.0<=3.0 and node 'S' is before node 'U' based on ***C_2***; node 'S' is before node 'V' based on ***C_3*** where their local labels are 2.1 and 4.3 respectively and the following equation is true: $1 \times 4 < 2 \times 3 \rightarrow 4<6$. Based on ***$C_4$*** node 'C' is before node 'V' where their locals are 2.0 and 4.3 and the SG of 'C' is 1: $1 \times 3 < 2 \times 4 \rightarrow 3 < 8$.
Based on ***C_5*** node 'V' is before node 'D' where their locals are 4.3 and 3.0 and the SG of 'D' is 2: $3 \times 2 < 4 \times 3 \rightarrow 6 < 12$.

Case 2: label order between nodes within a different group. In this case, as same as before insertions, the order is determined using the DDE pre-order definition (Xu *et al.,* 2009).

e.g . from Fig. 4.6.1 and Table 4.2, node 'P' $<_{DDE}$ node 'L' because :
FG for node 'P' → 2.2.3 & FG for node 'L' → 1.2.2, thus,
a_1=2, a_k=3 , b_1=1, b_k=2 → $3 \times 1 < 2 \times 2 \rightarrow 3 < 4$

- *Ancestor/Descendant (AD) Relationship:*
  The same definition applies as same as before insertions.
  e.g. from Fig 4.6.1 and Table 4.2, 'B' is an ancestor of 'P' such that, global label of 'B' is 1.1(as 'B' doesn't have SG) and the global label of 'P' is 2.2.3.1. Thus, $\frac{a_1}{b_1} = \frac{a_2}{b_2} = \cdots = \frac{a_m}{b_m} \rightarrow \frac{1}{2} = \frac{1}{2}$

- *Parent/Child (PC) Relationship:*
  The same definition applies as same as before insertions.

  e.g. from Fig. 4.6.1 and Table 4.2, node 'B' is the parent of node 'V' as 'B' is an ancestor of 'V' based on C_1 of *Definition 3*.

- *Sibling Relationships:*
  The same definition applies as same as before insertions.

  e.g. from Fig. 4.6.1 and Table 4.2, node 'K' & node 'X' are siblings as they belong to the same group and their levels are equal.

- *Lowest Common Ancestor (LCA):*
  The same definition applies as same as before insertions.

  e.g. from Fig. 4.6.1 and Table 4.2, the LCA between node 'P' and node 'D' is node 'B' where their global labels (FG *concatenate SG (if SG exists)* are *2.2.3.1 , 1.1.2 and 1.1* respectively, and 'B' is the ancestor of both 'P' and 'D' from

Definition 3. C_1 applies such that, the number of components in 'P' and 'D' global labels is 4 and 3 respectively → the minimum (4,3) = 3; and the following is true $\frac{1}{1} = \frac{1}{1}$.

4.5 Validating the Scheme's Properties

Given three nodes, n_1, n_2, n_3, with l_1 , l_2 , l_3 as their levels and with labels A, B and C and where their global labels are $\{a_1.a_2...a_m , (i^{th} \vee (i_1, i_2))\}$, $\{b_1.b_2...b_n , (j^{th} \vee (j_1, j_2))\}$ $\{c_1.c_2...c_r , (k^{th} \vee (k_1, k_2))\}$ and local labels are *aL* , *bL and cL respectively*:

From the notation given to the global label, it would be deduced that the global label consists of two part namely 'First Global' (Mesiti *et al.*) and 'Second Global' (SG). The global label notation can be explained by example as follow:
Label A has global label $\{a_1.a_2...a_m , (i^{th}$ ***or*** $(i_1, i_2))\}$:
Where $a_1.a_2...a_m$ → FG and $(i^{th}$ ***or*** $(i_1, i_2))$ → SG as SG can be a number that represent the child node number (e.g: first, second ,… , i^{th}) child and is assigned during the initial labelling. Or SG can hold references of two nodes locals if the node was inserted between two nodes.

- <u>*Label Order:*</u>

  Case 1:

  ∵$level_1 < level_2$ & $n_1, n_2 \in$ *same group*

  $\therefore \frac{a_1}{b_1} = \frac{a_2}{b_2} = \cdots = \frac{a_m}{b_m} \rightarrow n_1 \; ancestor \; of \; n_2$

  $\therefore n_1 \; precedes \; n_2$ → *this verifies* C_1

  C_2 , C_3 , C_4 *and* C_5 can be verified as follows:

  ∵$level_1 = level_2$ & $n_1, n_2 \in$ *same group*

  $\therefore from \; def.4 \rightarrow n_1 \; \& \; n_2 \; are \; siblings$

  $\therefore from \; def.1.1, n_1 \; precedes \; n_2$: n_1 & n_2 not inserted OR their Locals are not resulted

from locals additionif and only if $La < Lb$

$where\ <\ is\ based\ on\ the\ numbers\ natural\ order$; this verifies C_2

if ń$_1$ $is\ inserted\ between\ n_1$ & $n_3\ and$ ń$_2$ $is\ inserted\ between\ n_2$& $n_3\ and$ ń$_1$ $local\ or$ ń$_2$ $local\ resulted\ from\ local\ s\ addition \rightarrow$

$from\ def.1.1$, ń$_1$ $precedes$ ń$_2$ if and only if $Lá_1.Lá_2 < Lb'_1.Lb'_2$

$\Rightarrow \frac{Lá_1}{Lb'_1} < \frac{Lá_2}{Lb'_2} \Rightarrow Lá_2 \times Lb'_1 < Lá_1 \times Lb'_2$ This verifies C_3.

$If\ Lá_2 \times Lb'_1 < Lá_1 \times Lb'_2 \Rightarrow 0 < 0$

$\Longrightarrow Lá_1.0 < Lb'_1.0 \Rightarrow Lá < Lb' \Rightarrow This\ is\ also\ verifies\ C_3$

$if\ \ n'_2\ local\ is\ resulted\ from\ adding\ n_2\ local\ to\ n_3\ local$:

$n'_2\ is\ inserted\ between\ n_2$ & n_3 and $n_1\ is\ initially\ labelled$

$\therefore from\ def.1.1,\ n_1\ precedes\ n'_2$ if and only if $n_1SG.La_1 < Lb'_1.Lb'_2$

$\Longrightarrow \frac{n_1SG}{Lb'_1} < \frac{La_1}{Lb'_2} \Rightarrow n_1SG \times Lb'_2 < La_1 \times Lb'_1$; $This\ verifies\ C_4$

$if\ \ n'_2\ local\ is\ resulted\ from\ adding\ n_1\ local\ to\ n_2\ local$:

$n'_2\ is\ inserted\ between\ n_1$ & n_2 and $n_3\ is\ initially\ labelled$

$\therefore from\ def.1.1, n'_2\ precedes\ n_3$ if and only if $Lb'_1.Lb'_2 \leq n_2SG.Lb_1$

$\Longrightarrow \frac{Lb'_1}{n_2SG} \leq \frac{Lb'_2}{Lb_1} \Rightarrow Lb'_2 \times n_2SG \leq Lb'_1 \times Lb_1$; $This\ verifies\ C_5$

$\because\ n_1 is\ before\ n_2\ \ and\ \ n'_2\ is\ before\ n_2$

$\Longrightarrow La_2 \times Lb_1 \leq La_1 \times Lb_2\ \ and\ similarly\ Lb'_2 \times Lb_1 \leq Lb'_1 \times Lb_2$

$\Longrightarrow \frac{La_2}{La_1} \leq \frac{Lb_2}{Lb_1}\ \ and\ \ \frac{Lb'_2}{Lb'_1} \leq \frac{Lb_2}{Lb_1}$

$\Longrightarrow \frac{La_2}{La_1} \leq \frac{Lb'_2}{Lb'_1}$

$\Longrightarrow La_2 \times Lb'_1 \leq La_1 \times Lb'_2$

$\therefore\ n_1\ preceeds\ n'_2$; $this\ verifies\ the\ Transitivity\ of\ label\ order.$

Case 2: The DDE label order has already been verified in Xu *et al.* (2009)

- *AD Relationship:*

  If n_1 *is an* ancestor of n_2 and n_3 is a sibling of n_2:

  ∴ n_1 ancestor of n_3 : either n_3 is inserted or originally exists.

  To verify:

  from def.2: if n_1 is an ancestor of $n_2 \Rightarrow m < n$ & $\frac{a_1}{b_1} = \frac{a_2}{b_2} = \cdots = \frac{a_m}{b_m}$

  from def.4: if n_2 & n_3 are siblings ⇒$n=r$ & $b_1.b_2...b_n = c_1.c_2...c_r$

  $$\therefore \frac{a_1}{b_1} = \frac{a_2}{b_2} = \cdots = \frac{a_m}{b_m} \equiv \frac{a_1}{c_1} = \frac{a_2}{c_2} = \cdots = \frac{a_m}{c_m}$$

  ∴ n_1 ancestor of n_3

- *Sibling Relationships:*

  (Symmetry of sibling relationships):if n_1 is a sibling of n_2, *then* n_2 is a sibling of n_1.

  from def.4: if n_1 & n_2 are siblings ⇒$m=n$ & $a_1.a_2...a_m = b_1.b_2...b_n$

  $\Rightarrow \frac{a_1}{b_1} = \frac{a_2}{b_2} = \cdots = \frac{a_m}{b_n} = 1 = \beta$ Equivalently, $\frac{b_1}{a_1} = \frac{b_2}{a_2} = \cdots = \frac{b_n}{a_m} = \frac{1}{\beta} = \frac{1}{1} = 1$

  ∴ n_2 is a sibling of n_1

  (The transitivity of sibling relationships):if n_1 is a sibling of n_2 and n_2 is a sibling of n_3, *then* n_1 is a sibling of n_3

  To verify:

  from def.4: if n_1 & n_2 are siblings ⇒$m=n$ & $a_1.a_2...a_m = b_1.b_2...b_n$

  $\Rightarrow \frac{a_1}{b_1} = \frac{a_2}{b_2} = \cdots = \frac{a_m}{b_n} = 1 = \beta$, Similarly, n=r & $b_1.b_2...b_n = c_1.c_2...c_r$

  $\Rightarrow \frac{b_1}{c_1} = \frac{b_2}{c_2} = \cdots = \frac{b_n}{c_r} = 1 = \gamma \Rightarrow \frac{a_1}{c_1} = \frac{a_1}{b_1} \times \frac{b_1}{c_1} = \beta \times \gamma = 1 = \frac{a_2}{c_2} = \cdots = \frac{a_m}{c_r}$

  ∴ n_1 is a sibling of n_3

- *PC Relationship & LCA:*

  These are valid if, and only if, the AD relationship is valid which has already been verified.

4.6 Conclusion

This chapter described the dynamic labelling scheme presented in this thesis. It illustrated how labelling works theoretically by showing the mechanism of the initial labelling and how different relationships are determined. Then, it demonstrated how the scheme handles different types of insertion and how the relationships are preserved after insertions; simple examples were provided. Finally, a correctness of the scheme's properties was given using simple algebra. Describing the scheme from a design and implementation point of view is discussed in the next chapter.

Chapter 5: Design and Implementation

5.1 Introduction

After describing the mechanism of the labelling scheme theoretically (in Ch.4), this chapter discusses the scheme from a design and implementation point of view. As shown in the previous chapter, the labelling scheme presented in this thesis is based on the Dynamic Dewey labelling scheme (DDE), so the main comparisons are made between these two. Thus, in addition to implementing the new scheme, a DDE scheme was also implemented. This was necessary in order to cover the more experimental aspects as no published open sources were available. This is shown in Chapter 6 which describes the experimental design.

This chapter explains the design and implementation of both schemes in parallel, starting with a general explanation of the design and implementation of both schemes in Section 5.2. Then, the initial labelling mechanisms of both techniques are described in Section 5.3 while performing the search through the labelled nodes is shown in Section 5.4. The way each scheme handles different types of insertion is discussed in Section 5.5 and the implementation of different relationships is discussed in Section 5.6. Finally, the chapter concludes in Section 5.7.

5.2 Overview

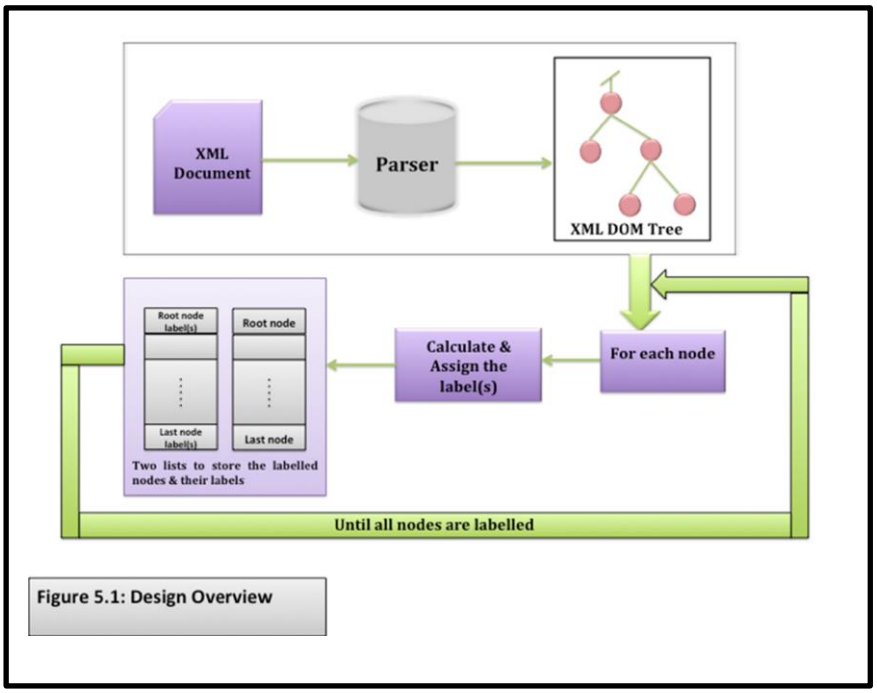


Figure 5.1: Design Overview

Generally, both schemes share the same external design but differ in the implementation of their inner methods, which are defined based on the labelling characteristics of each scheme. NetBeans IDE 8.0 and Java JDK 1.8 were used in the implementation phase of both schemes. Figure 5.1 shows the design overview.

As shown in Fig. 5.1, each scheme starts by parsing the XML file using one of the XML parsers (Ch.2). The choice of the most suitable parser between a streaming API parser, such as SAX, and a tree-based API parser, such as DOM, was based on the program's characteristics. If the size of the XML documents exceeds the available memory, the only possibility is a streaming API parser. In addition, this parser can be used if it is possible to process the document as small sequential input sections without a requirement that the whole document be available prior to processing a certain part, or else if the processing of the document can be undertaken in a series of separate operations (Brownell and Megginson, 2002, Project, 2013a).

However, there are several circumstances when a tree-based API is a more suitable choice, including when the program requires simultaneous access to different document parts, when the complexity of internal data structures matches that of the actual document, or when the program has constantly to adjust the document (Berglund *et al.*, 2010a, Frank *et al.*, 2003). Given that the main objective of this study is to assess the capacity of the scheme to manage insertions, thus involving frequent alterations of the document, a DOM parser was considered to be the most appropriate choice. This type of parser is not only well-known, but also easy to apply due to its 'pull model', which enables the client program to employ different methods to obtain the desired information from a document. By contrast, in the 'push model' of SAX, the parser specifies what and when it reads, regardless of whether or not the information is required (Harold, 2002). However, this decision has consequences and they are discussed in Chapter 8.

After parsing the document, two lists are created for each scheme. The first one holds the nodes which are being labelled and the other one holds the label of each node in its correspondence index within the first list. *'ArrayList'* was chosen from the java collections for the implementation for reasons discussed below.

Some of the tasks that java developers implement entail the storage and retrieval of objects in collections. Java offers a number of collection classes that have unique and overlapping characteristics. Possibly, the most used collection implementation classes are '*ArrayList'*, '*Vector'*, and '*LinkedList'*. It can be difficult to deal with these collection classes particularly within a multithreading setting since majority of these do not offer default-synchronised access. Even though '*Vector'* provides default-synchronised access, '*ArrayList'* compensates for this through synchronization methods (Sanghera, 2006). The structure required in the implementation of both indexing methods must have multiple threads that insert, remove, and iterate through elements of the collection.

As the name implies, the '*ArrayList*' List interface involves defining an object array and increasing the size of the array as needed to support elements contained in the collection (Naftalin and Wadler, 2006). The appealing characteristics of '*ArrayList*' include its capability to contain duplicate elements and null values. Even though it is not a naturally thread-safe class when an instance needs to be used by several threads, '*ArrayList*' provides methods for synchronising modifications made to the list. In this application, thread safety is not necessary. Because, creating and populating the '*ArrayList*' occurs in a single thread, which makes it safe for multiple threads to retrieve values from the '*ArrayList*'. Another useful feature associated with '*ArrayList*' is that it does not compel the developer to set or even update its capacity since the capacity increases automatically (Drozdek, 2004, Matha, 2011, Spell, 2005).

Although the implementation of the '*LinkedList*' interface does not provides behaviour that is visibly different from '*ArrayList*', it is different in the way the list is maintained. '*LinkedList*' class utilises double linked list to handle the collection of objects. This implies that every node within the list has pointers to nodes that precede and follow it, which allows a list to be navigated in either direction. Although in theory '*LinkedList*' ought to offer performance advantages compared to '*ArrayList*' when inserting or removing an element, in practice the performance advantage is insignificant and '*LinkedList*' is slower compared to '*ArrayList*' when inserting an element to the end of the list. The explanation lies in operations performed in the middle of '*LinkedList*' because nodes must be traversed to get to their location in the list; therefore, '*LinkedList*' execute more slowly than '*ArrayList*' because accessing an element in the middle of an '*ArrayList*' is not slower or faster than accessing one in any other location (Spell, 2005).

Apart from offering better performance, '*ArrayList*' has an extra advantage over '*LinkedList*' because it uses less memory. '*LinkedList*' need to create a node object for every element inserted in the '*LinkedList*', while '*ArrayList*' only needs to maintain a single object array and the only instance it creates a new object is when

the array needs to increase. The process of creating an object not only uses more memory but also is time consuming making '*LinkedList*' slower than '*ArrayList*' (Spell, 2005). In addition, random access is faster in '*ArrayList*' compared to '*LinkedList*' (Sanghera, 2006).

'ArrayList' is an essential class implementation for Java's Collection framework. '*ArrayList'* implements "Serializable, Coneable, Iterable<E>, Collection<E>, List<E>, and RandomAccess" (Lewis and Chase, 2010). The defining quality of '*ArrayList'* is its capability to grow or shrink in response to the needs of a program (Dale *et al.*, 2012, Flanagan, 2005, Weiss, 1992). Therefore using this approach, the developer does not have to worry about bounded stacks. Although, it is possible to implement *BoundedStackInterface*, the developer is also able to implement *UnboundedStackInterface* making sure the constructor does not need to establish the stack size (Dale *et al.*, 2012).

Similar to the '*Vector*' class, the '*ArrayList*' implementation is resizable, which indicates that addition of a new element would cause overflow in the '*ArrayList*' that in return causes the underlying array to resize automatically (Sikora, 2003). In managing the array size, '*ArrayList'* contains two extra operations: *ensureCapacity* that increases array size to the precise size if the array is not that large or even larger and *trimToSize* that trims the array to fit the current list size. Moreover, '*ArrayList'* class inherits a considerable number of extra methods from its super classes (Lewis and Chase, 2010).

Although Vector provides the thread-safe feature, which is desirable, thread safety is not necessary in many circumstances; besides, synchronisation is a time-consuming process. Since the implementation of the schemes does not need a synchronised collection class, the use of '*ArrayList'* enables constant time access to any element due to fast random read access (Documentation, 2014, Horstmann and Cornell, 2002, Schildt, 2006) and eliminates the possibility of using synchronisation that can slow down the application (Spell, 2005). Therefore in this thesis, '*ArrayList'* seems to be the optimum implementation since other classes

would not fully address the needs in the manner '*ArrayList*' does. However, The consequences of this decision are discussed in Chapter 8.

The next step in the implementation begins by examining each node and assigning a label to it. Figure 5.2 shows the pseudo code that represents the implementation process generally.

```
1) Define listOfNodes as ArrayList
2) Define listOfLabels as ArrayList

3) Read XML document
4) Parse the XML document using DOM

5) Get the NodeList of all DOM nodes

6) while (the next node exists) {
        7)Define String newLabel
         8)  if (the type of the current node == element){
                  9) Calculate the Label(s)
                  10) newLabel = the calculated label
                  11)   Add the newLabel to listOfLabels
                  12)   Add the current node to listOfNodes
         13) } //End if
14) }//End while
```

Figure 5.2: General Pseudo code

5.3 Initial Labelling

5.3.1 The GroupBased Labelling Scheme:

First, to simplify the demonstration, the following abbreviations are used.

- First part of the global label → FG
- Second part of the global label → SG
- Local label → L
- *Note: if the global label consists of two parts, the first part is referred to as FG and the second as SG.*

As mentioned in the previous section, two lists are used to store the nodes and their labels; however, the data type of the first list, which stores the nodes, is obviously 'Node'. The second list type is '*NodeInfo*, where '*NodeInfo*' is a Java class that has the global and local labels of its members; thus, the second list stores both labels together. Even though defining the node as a member of the '*NodeInfo*' class leads to using only one list in the program (as will be illustrated in Section 5.4), it makes the search process slow. Figures 5.3.1, 5.3.2 and 5.3.3 show the flow chart and pseudo code of the initial labelling process of the new scheme.

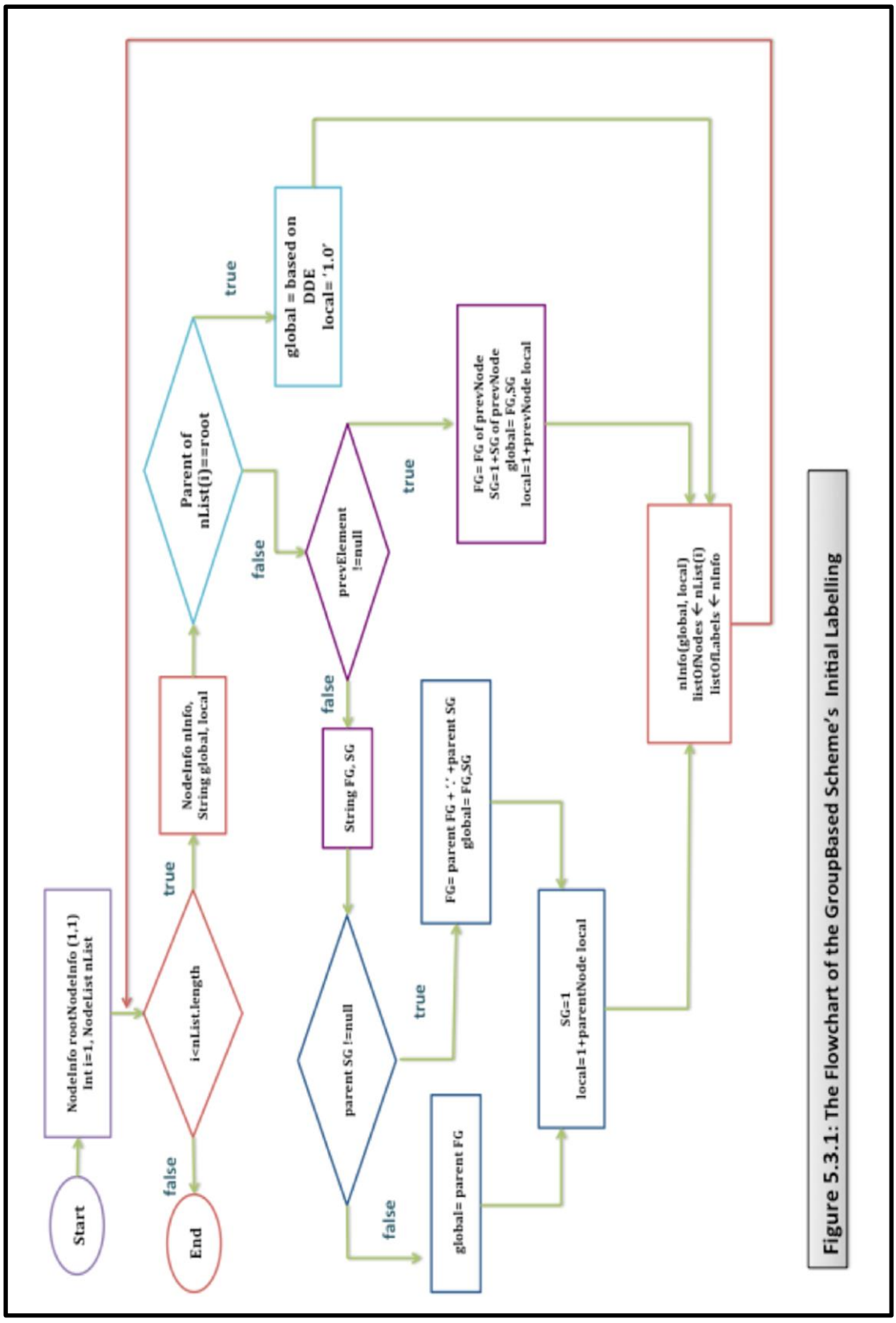


Figure 5.3.1: The Flowchart of the GroupBased Scheme's Initial Labelling

```
1) local = "1.0"
2) if (previous Element not Exists){

        3) global= parent global +"."+"1"
   }//End if

4) else {

          5) prevG=get previous node global
          6) global=Add one to the last component in prevG

   }//End else
```

Figure 5.3.2: Pseudo code: calculating the labels when the parent node is the root

```
1)  String FG, SG //first and second parts of the global

2) if (previous Element not Exists){
        3) SG="1"
        4) if(parent SG !=null){
             5) FG= parent FG +"."+parent SG
   }//End if (4)
        6)else{
             7) global= parent global
           }//End else
         8) local=(double) parent Local +1
    }//End if(2)
 9) else {

          10) FG=previous node FG
          11) local=(double) previous node Local +1

   }//End else
 12) global= FG+","+SG
```

Figure 5.3.3: Pseudo code: calculating the labels when the parent node is not the root

As seen in Fig. 5.3, and as demonstrated in the previous chapter, an instance of '*NodeInfo*' is created for the document root and both its global and local labels are set to '1'. Then, the root node and this instance are added to the first and second lists respectively. Then, all the document's nodes are obtained and examined starting from index (Kasim *et al.*); the root node is excluded as it is already labelled. After this, each node is tested to see whether it is a child of the document root or not. If it is, the local label is set to '1.0' and the global label is calculated based on the DDE labelling scheme.

If the examined node is not a child of the root, the node's global label will consist of two parts, as described in the previous chapter. It starts by checking whether a previous sibling exists and, if so, the node's FG is set to be equal to the previous sibling FG while its SG and L are calculated by adding one to their corresponding values in the previous sibling's labels. Thus, if the examined node is the first child, its SG is set to '1' and its L is calculated by adding one to the parent node's L; and the nodes FG is set based on the parent node's SG such that, if it exists, the examined node's FG is formed by concatenating the parent node's FG and SG; otherwise, the FG is set to be equal to the parent node's FG.

After determining the node's label, an instance of type '*NodeInfo*' is created to contain the global and local labels; this is then added to the second list while the labelled node is added to the first list. These processes continue until all the nodes have been labelled.

5.3.2 DDE Labelling Scheme

As with the new scheme, two lists are used and the first one is of type 'Node' to store the labelled nodes; the second one is of type 'String' as there is only one label so there is no need to define a class. From the description available in Xu *et al.* (2009) on how DDE labelling is calculated, Figure 5.4 shows the flow chart and the pseudo code of the initial labelling process of the DDE scheme.

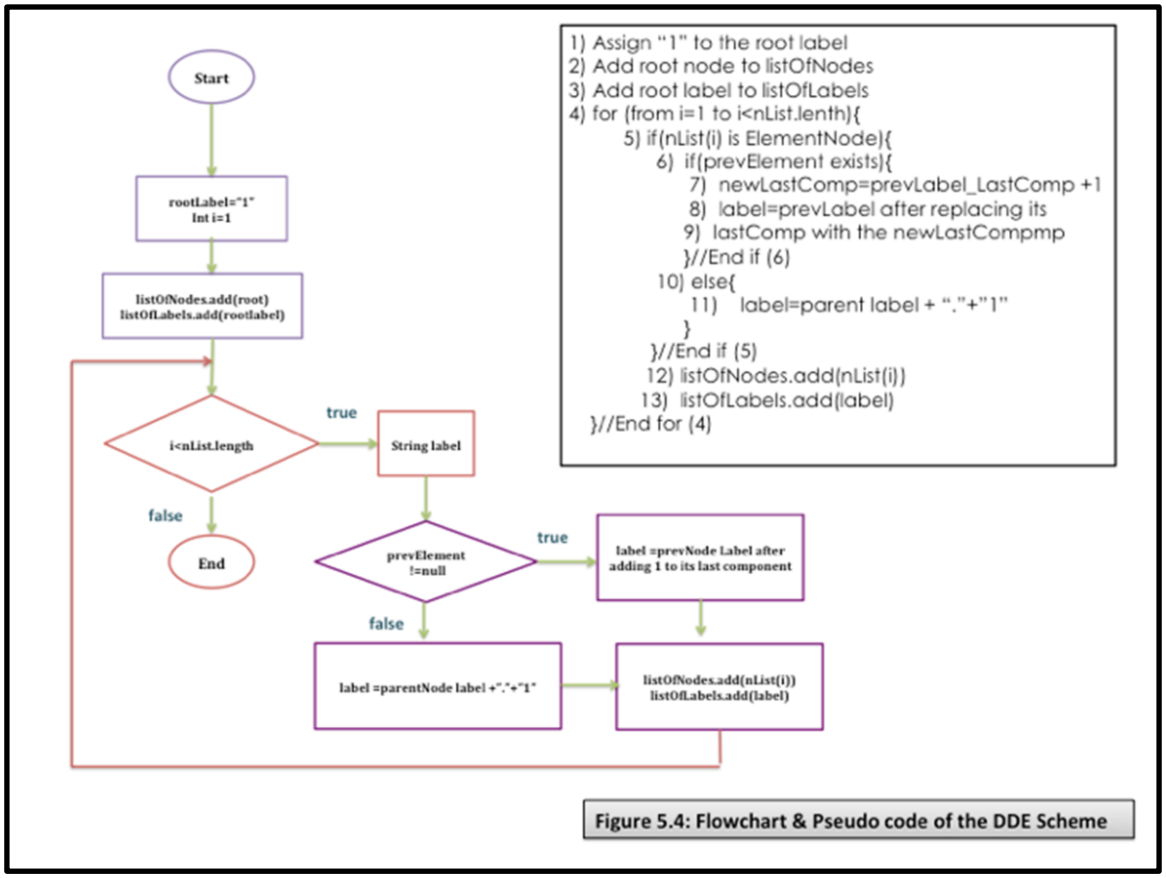


Figure 5.4: Flowchart & Pseudo code of the DDE Scheme

The labelling starts by assigning '1' to the document root, and the root node and its label are added to the lists. Then, a list of all document nodes is obtained and examined, ignoring the root node. If the node is a first child, the label is calculated from its parent label; otherwise, the previous sibling label is used in the calculation. Every time, the node and its label are added to the nodes' and labels' lists respectively.

5.4 Search Mechanism

As stated in the previous section, storing the nodes in a separate list facilitates the search for a certain node and its label(s). For example, if the node is defined as a class member besides its label(s), obtaining a specific node's label means traversing the list until the matching node is found; the object is returned and is used to get the label(s) which is a time-consuming process. However, because the nodes are in a different list, finding the index of a specific node and using that

index to access the label is much faster and more efficient as each label is stored under the same index in the second list. Figure 5.5 shows the flow chart and the pseudo code of the search mechanism.

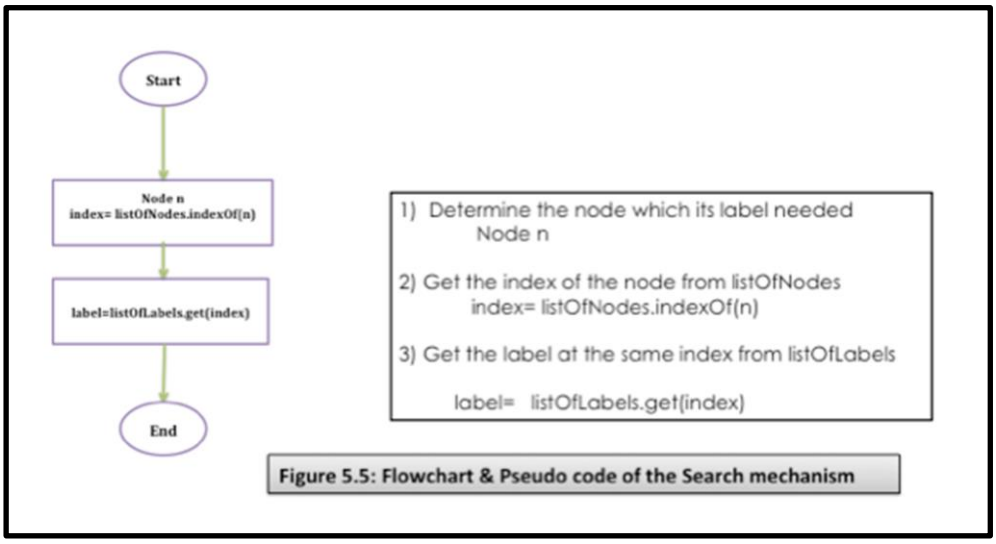


Figure 5.5: Flowchart & Pseudo code of the Search mechanism

5.5 Performing Insertions

This section describes how each labelling scheme performs insertions in practice.

5.5.1 Leftmost Insertion (new node (n_2) is inserted before node (n_1)).

The leftmost insertion refers to insert a node before the first child node of any parent node. A method called '*leftmost_Insertion'* handles this type of insertion in both schemes, '*void'* is the method's return type and n_2, n_1 are the method's argument. The process starts by calling the search, as described in the previous section, to obtain the label(s) of n_1 (the existing node) ; so, n_2 (the new node) label(s) can be calculated based on the scheme's characteristics as follows:

- *The GroupBased Labelling Scheme:*

When performing a leftmost insertion, the inserted node's n_2 label is calculated based on the n_1 parent node, as described in Chapter 4. If the document root is the parent, the n_2 global label is formed based on the DDE scheme; then '1.0' is assigned as the n_2 local label. But, if the n_1 parent is not the root, the FG of n_2 equals n_1 FG and the SG of n_2 is calculated by decrementing the n_2 SG by one. The n_2 local label is formed by subtracting one from the first component of the n_1 local label until the new local label is less than the parent's local label. Then, a new instance of '*NodeInfo*' is created using the calculated labels. Finally, in order to keep the document order, n_2 is added to the labelled nodes' list at index(n_1-1). Similarly, the '*NodeInfo*' instance is added at this index in the second list and n_2 is added to the XML tree using the DOM method *'insertBefore'*. Figures 5.6.1 and 5.6.2 show the flowchart and the pseudo code of this method.

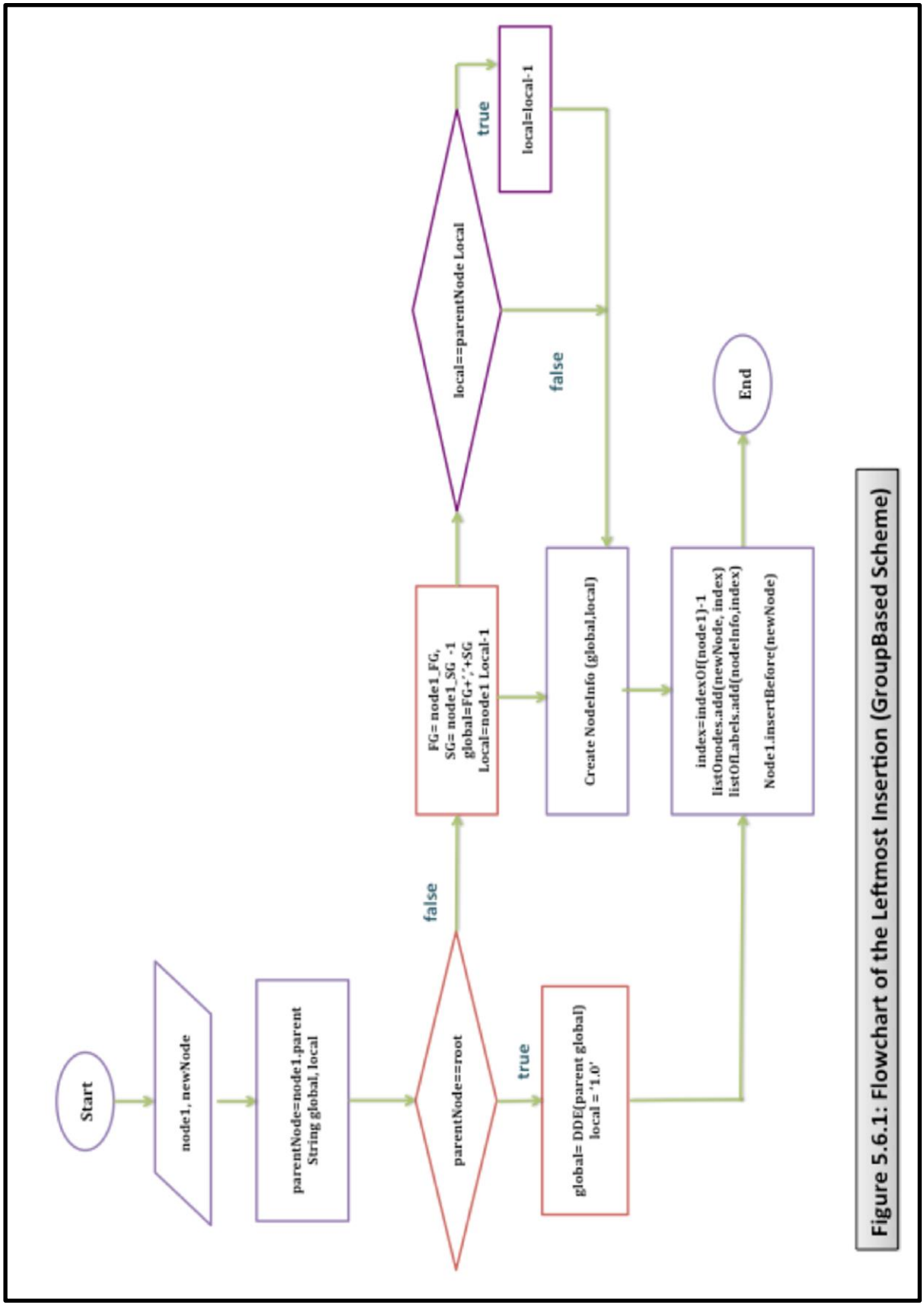


Figure 5.6.1: Flowchart of the Leftmost Insertion (GroupBased Scheme)

```
1)  Get the method's arguments (node1 , newNode)

2) Get node1 parent node
3) Define String variables newGlobal and newLocal
4) if(parent node == root){

      5) get node1_global
      6) newGlobal = node1_global without the last component
      7) last_component=  node1_global (last component )-1
      8) newGlobal = global (concatenation) ) last_component
      9) newLocal = '1.0'
   }//End if (4)

10) else{
      11) String newFG , newSG // 1st and 2nd parts of the global
      12) newFG =node1_FG
      13) newSG=node1_SG -1
      14) newGlobal=newFG + "," + newSG
      15) newLocal = node1_local -1
      16) if(newLocal == parentLocal)
            17) newLocal-=1
   }//End else (10)
18)NodeInfo  newNodeInfo=new NodeInfo(newGlobal,
newLocal)
19) index= listOfNodes.indexOf(node1) -1
20) listOfLabels.add(newNodeInfo, index)
21) listOfNodes.add(newNode, index)
22) node1.insertBefore(newNode)
```

Figure 5.6.2: Pseudo code of the Leftmost Insertion (GroupBased Scheme)

- *DDE Labelling Scheme:*

Calculating the new DDE label is less complicated as it is sufficient to use the search mechanism outlined in Section 5.4 (where the n_2 label is formed by reducing the last component of the n_1 label by one) to obtain the n_1 label. Finally, n_2 and its label are added to the first and second lists at index(n_1-1) and n_2 is added to the XML tree using the DOM method *'insertBefore'*. Figure 5.7 shows the flowchart and the pseudo code of this method.

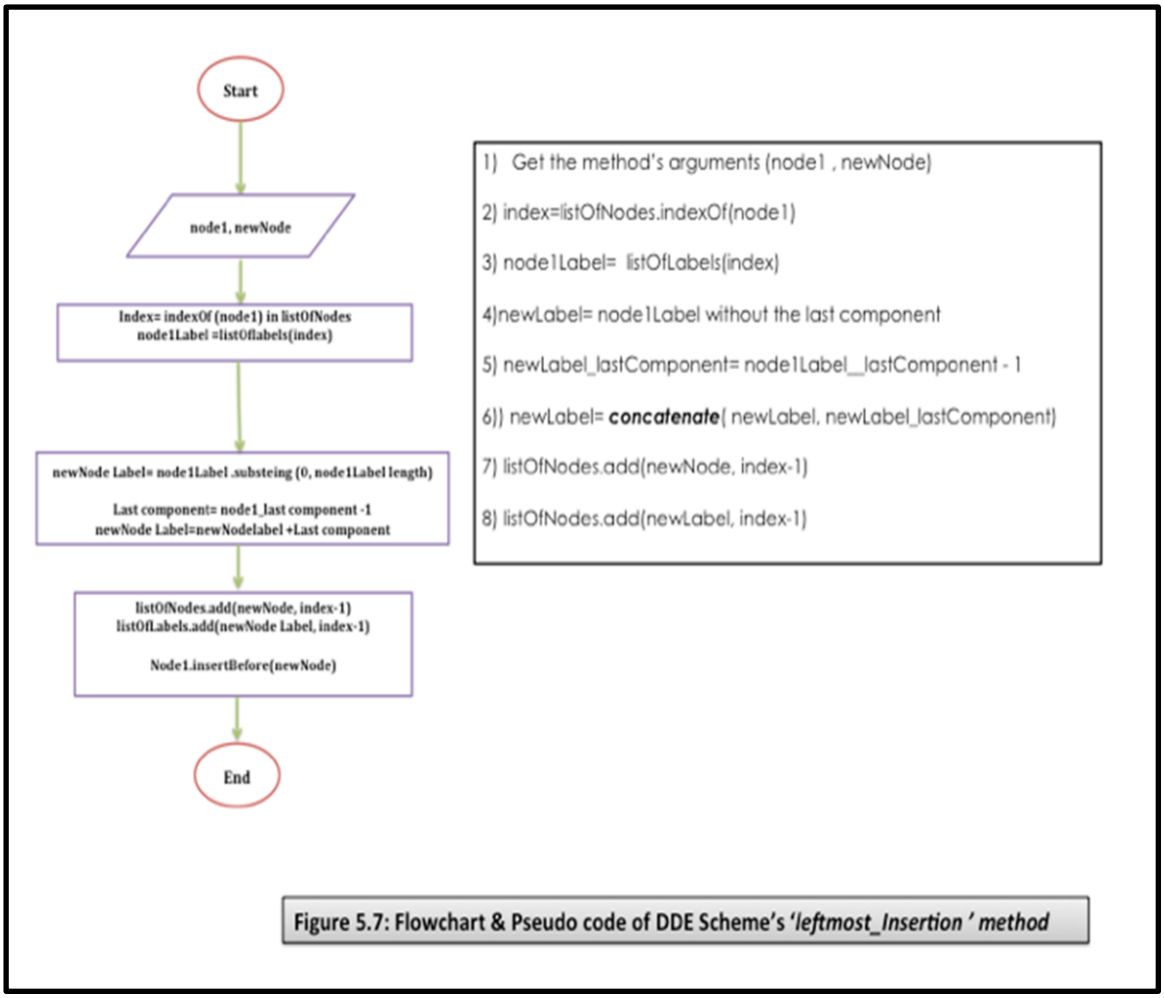


Figure 5.7: Flowchart & Pseudo code of DDE Scheme's *'leftmost_Insertion ' method*

5.5.2 Rightmost Insertion (new node (n_2) is inserted after node (n_1)).

The rightmost insertion refers to insert a node after the last child node of any parent node. Similar to the leftmost insertion, a method called 'rightmost_Insertion' is responsible for processing this type of insertion; it returns 'void' and takes n_2, n_1 as its argument. The method starts by obtaining the n_1 label, as described in Section 5.4. Then the new label is calculated as follows:

- *The GroupBased Labelling Scheme:*

  The new label is calculated as in the leftmost insertion method expect when the root document is not the parent of n_1. In this case, the new SG is computed by adding one to the n_1 SG while the n_2 local label is computed by adding one to the last component of the n_1 local label. Then, a '*NodeInfo*' instance is created and added to the labels' list at index(*lastDescendant*(n_1)+1); the node is also added at the same index. Finally, n_2 is added to the XML tree using '*appendChild*' in the DOM method. Figures 5.8.1 and 5.8.2 show the flowchart and the pseudo code of the rightmost insertion.

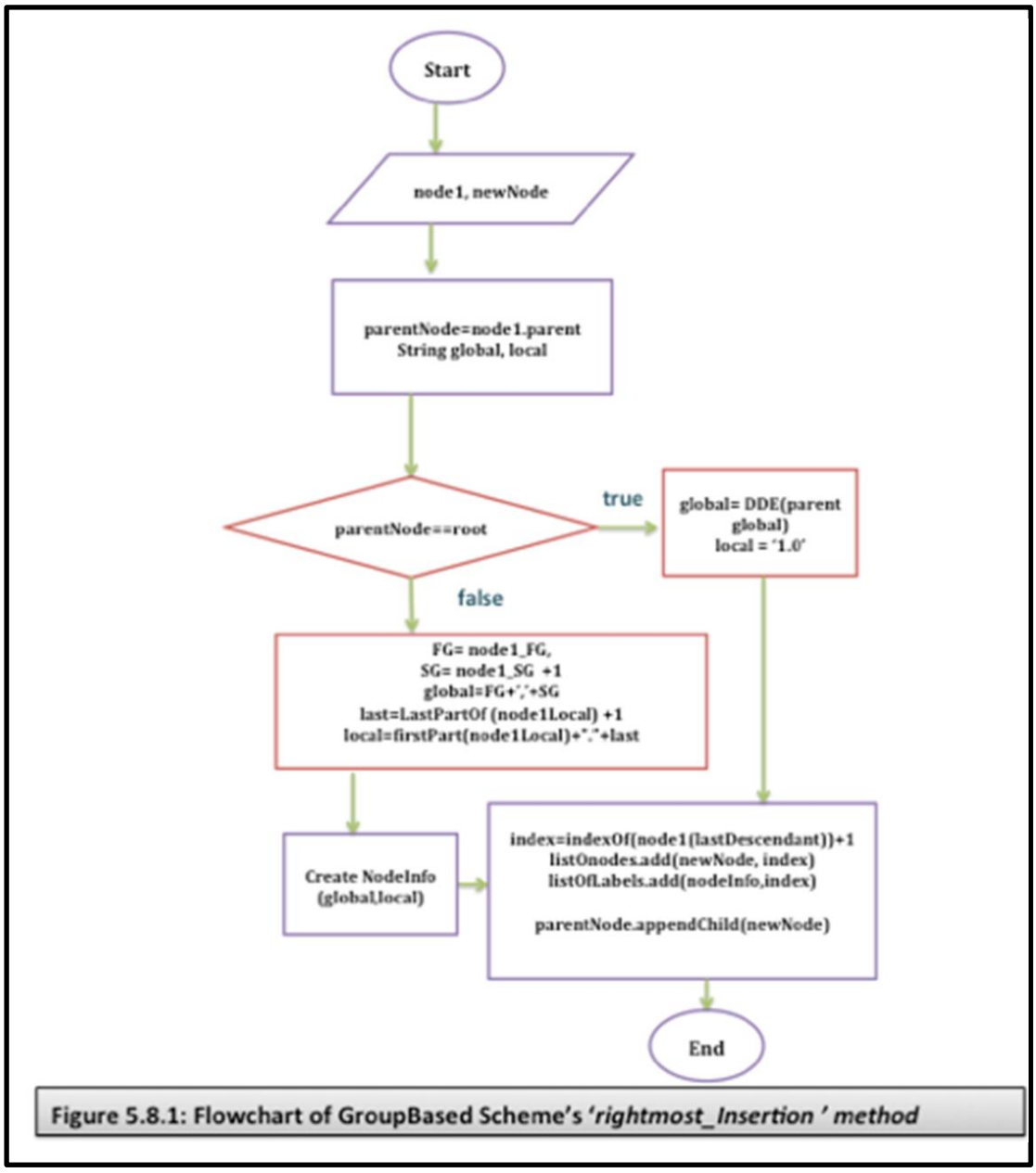


Figure 5.8.1: Flowchart of GroupBased Scheme's '*rightmost_Insertion ' method*

```
1)  Get the method's arguments (node1 , newNode)

2) Get node1 parent node
3) Define String variables newGlobal and newLocal
4) if(parent node == root){

     5) get node1_global
     6) newGlobal = node1_global without the last component
     7) last_component=  node1_global (last component )+1
     8) newGlobal = global (concatenation) ) last_component
     9) newLocal = '1.0'
   }//End if (4)

10) else{
     11) String newFG , newSG // 1st and 2nd parts of the global
     12) newFG =node1_FG
     13) newSG=node1_SG +1
     14) newGlobal=newFG + "," + newSG
     15) newLocal_lastComponent=node1Local_lastComponent+1

     16) newLocal = firstComponent(node1_local)
+","+newLocal_lastComponent

}//End else (10)

17)NodeInfo  newNodeInfo=new NodeInfo(newGlobal, newLocal)
18) index= listOfNodes.indexOf(lastDescendant(node1)) +1
19) listOfLabels.add(newNodeInfo, index)
20) listOfNodes.add(newNode, index)
21) parentNode.appendChild(newNode)
```

Figure 5.8.2: Pseudo Code of GroupBased Scheme's *'rightmost_Insertion ' method*

- *DDE Labelling Scheme:*

The only difference between this insertion and the leftmost insertion is that the new DDE label is calculated by adding one to the last component of the n_1 label. Then, n_2 and its label are added to the first and second lists at index(*lastDescendant*(n_1)+1) and n_2 is added to the XML tree using '*appendChild*' in the DOM method. Figure 5.9 shows the flowchart and the pseudo code of this method.

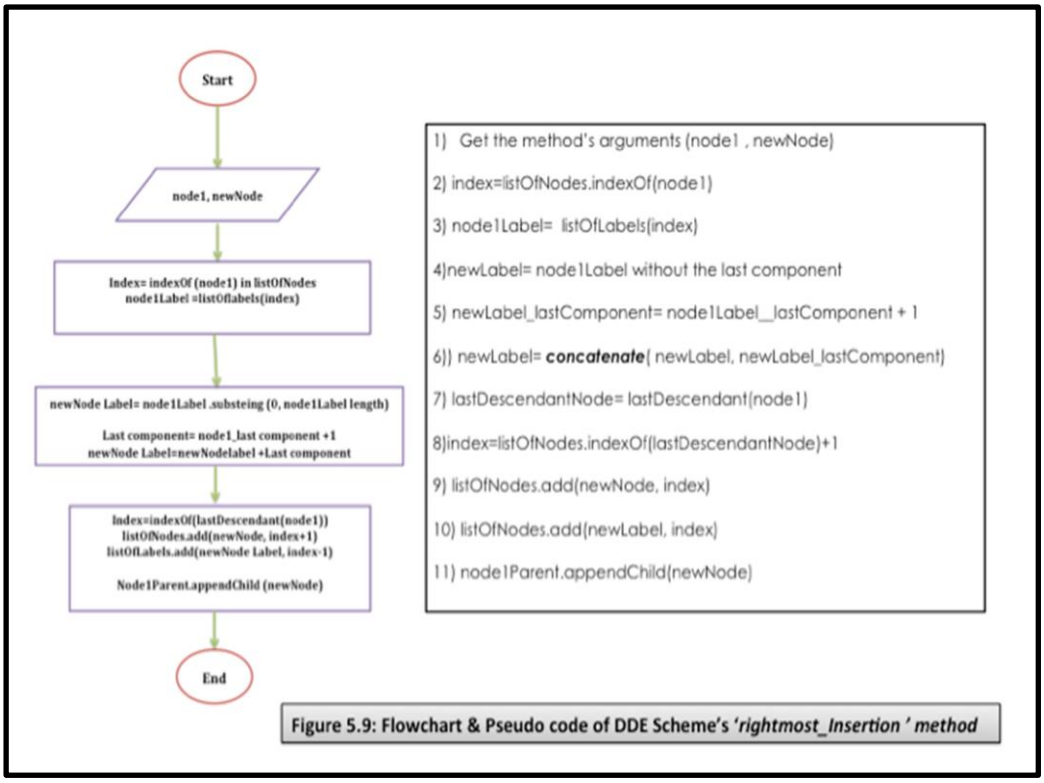


Figure 5.9: Flowchart & Pseudo code of DDE Scheme's *'rightmost_Insertion ' method*

However, the '*lastDescendant*' method is used to extrapolate the index of the last descendant node of n_1 in order to add the new node and its labels at the correct position based on the DOM parser. The method starts by examining the last child node of n_1 and checking whether or not it is a leaf node. If it is a leaf node, its index is returned; otherwise this node's last child is examined and so on until the last descendant node is reached. Figures 5.10.1 and 5.10.2 show how the '*lastDescendant*' method works.

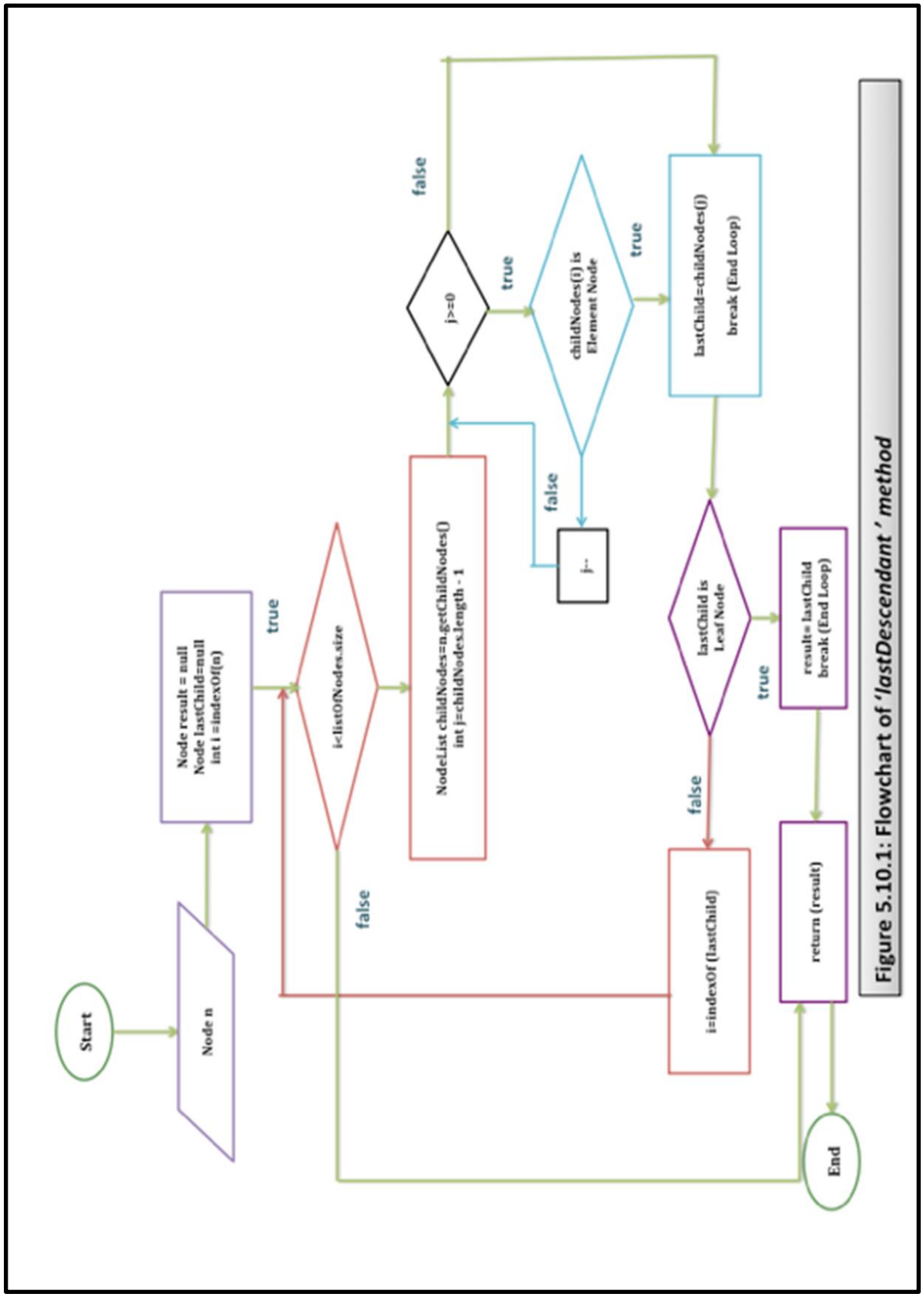


Figure 5.10.1: Flowchart of *'lastDescendant ' method*

```
1)  Get the method's arguments (Node n)

2) Declare Node result & lastChild
3) int i= listOfNodes.indexOf(n)
4) while(i<listOfNodes.size){

      5) get all childNodes of n
      6) initialise j=childNodes.size -1 //start from the last child of n
      7)for loop from j to j>=0  after each iteration (j--)
            8)if(childNodes(j) == Element){

                  9) lastChild=childNodes(j)
                  10) break from for loop(7)
             }//End if (8)
       }//End for (7)

      11) if(lastCihild is LeafNode){

            result= lastChild
          }//End if (11)
        12) else{

                  i=listOfNodes.indexOf(lastChild)
            }//End else (12)
}//End while (4)

13) return (result)
```

Figure 5.10.2: Pseudo code of *'lastDescendant ' method*

5.5.3 Inserting Between Two Consecutive Nodes (Inserting new node (n_3)) between (n_1, n_2).

The method's name is '*InsertBetween*'; it returns void and takes n_3, n_1, n_2 as its arguments. Using the search mechanism described in Section 5.4, the label(s) of n_1, n_2 are obtained and each scheme forms the new label(s) as follows:

- *The GroupBased Labelling Scheme:*

  Like the previous two types of insertion, the new local label is '1.0' and the new global label is based on the DDE scheme where the parent of n_1 is the document root. Otherwise, as described in the previous chapter, two cases are available and, in both of them, the FG of n_3, the new node, is equal to the FG of n_1 & n_2, the nodes that it is to be inserted between, while n_3 SG holds references to the local labels of n_1, n_2. However, the n_3 local label is calculated differently, as follows:

  1. When n_1, n_2 have not been inserted after the initial labelling or only n_1 has been inserted: the n_3 local label is calculated by adding one to the last component of the n_1 local label.
  2. When n_1, n_2 were both inserted: the n_3 local label is calculated by performing an addition between n_1, n_2 local labels, as described in the previous chapter.

  Then, a '*NodeInfo*' instance is created and added to the labels' list at index(n_2); the node is also added at the same index. Finally, n_3 is added to the XML tree using DOM's '*insertBefore*' method on n_2. Figures 5.11.1 and 5.11.2 show the flowchart and the pseudo code of the *insertBetween* method.

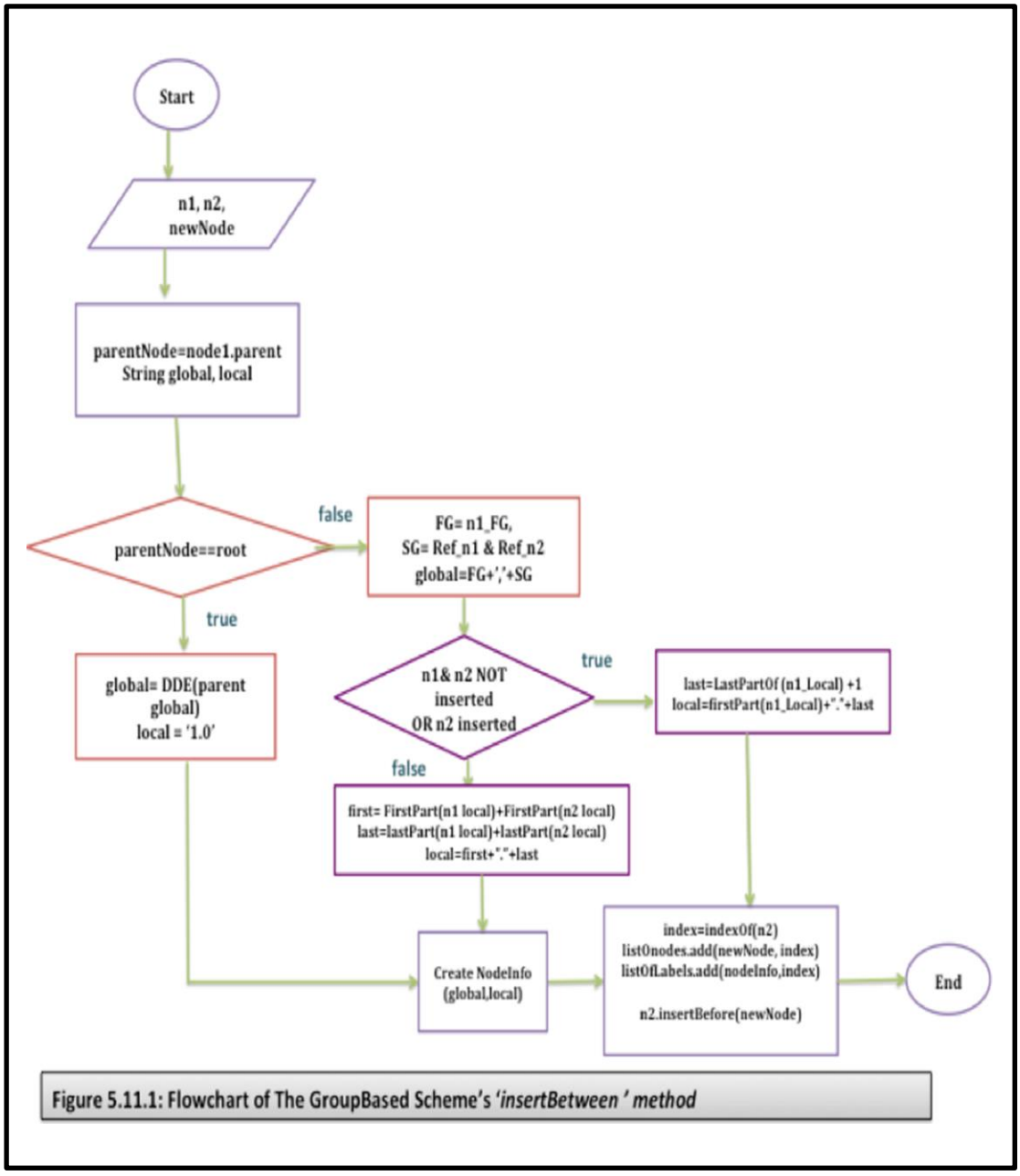


Figure 5.11.1: Flowchart of The GroupBased Scheme's *'insertBetween ' method*

```
1)  Get the method's arguments (n1 ,n2,  newNode)

2) Get n1 parent node
3) Define String variables newGlobal and newLocal
4) if(parent node == root){

     5) get n1_global & n2_global
     6) newGlobal = DDE_Addition(n1_global, n2_global )
        //DDE_Addition (Adding each component in (n1_global)
        // to its correspondence in (n2_global)
     7) newLocal = '1.0'
   }//End if (4)

8) else{
     9) String newFG , newSG // 1st and 2nd parts of the global
     10) newFG =node1_FG
     11) newSG=n1LocalRef & n2LocalRef
     12) newGlobal=newFG + "," + newSG
     13) if((n1 & n2 NOT inserted) || (only n2 is inserted)){
           14) newLocal = n1_local after adding 1 to its last part
         }//End if (13)
     15) else{
           16) newLocal= Adding each component in n1_local
               to its correspondence in n2_local
         }//End else (15)
}//End else(8)
16)NodeInfo  newNodeInfo=new NodeInfo(newGlobal, newLocal)
17) index= listOfNodes.indexOf(n2)
18) listOfLabels.add(newNodeInfo, index)
19) listOfNodes.add(newNode, index)
20) n2.insertBefore(newNode)
```

Figure 5.11.2: Pseudo code of The GroupBased Scheme's *'insertBetween ' method*

- *DDE Labelling Scheme:*

  The new label is calculated by adding each component of the n_1 label to its correspondence in the n_2 label. Then, n_3 and its label are added to the lists at index(n_2) and n_3 is added to the XML tree using DOM's '*insertBefore*' method on n_2. Figures 5.12.1 and 5.12.2 show the flowchart and the pseudo code of this method.

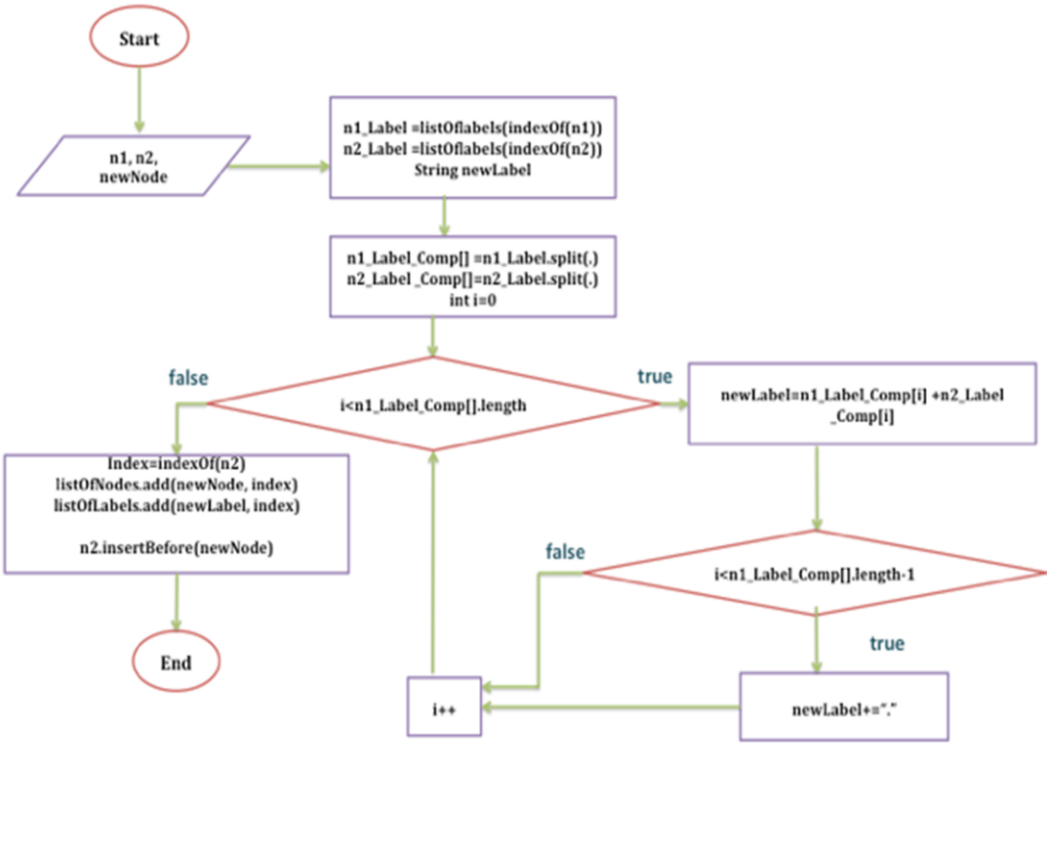


Figure 5.12.1: Flowchart of DDE Scheme's '*insertBetween* ' *method*

```
1)  Get the method's arguments (n1, n2 , newNode)

2) index1=listOfNodes.indexOf(n1)
3) index2=listOfNodes.indexOf(n2)

4) Label1=  listOfLabels(index1)
5) Label2=  listOfLabels(index2)

6) Initialise i =0
7) get labe1 & label2 components in array1 & array2

8)for (i <array1.length ; i++){

    9)  newLabel=array1[i]+array2[i]

    10) if( i<array1.length -1)
            11) newLabel+="."

}//End for(8)


12)index=listOfNodes.indexOf(n2)

13) listOfNodes.add(newNode, index)

14) listOfNodes.add(newLabel, index)

15) n2.insertBefore(newNode)
```

Figure 5.12.2: Pseudo code of DDE Scheme's *'insertBetween ' method*

5.5.4 Inserting below a Leaf Node (new node (n_2) is inserted below node (n_1)). It starts by obtaining n_1 label(s), then the new label(s) is calculated as follows:

- *The GroupBased Labelling Scheme:* Two cases are available and in both of them n_2 SG is set to '1' and the n_2 local label is calculated by adding one to the n_1 local label. However, FG is calculated differently, as follows:
    1. When n_1 parent is the document root: n_2 FG equals the n_1 global label.
    2. Otherwise: n_2 FG is calculated based on n_1SG as follows:
        - If n_1SG contains a number; this means that n_1 has not been inserted after initial labelling. In this case, n_2 FG is formed by concatenating n_1FG and n_1SG, with '.' is between them.
        - n_1SG contains references to other nodes; this means that n_1 is inserted between two nodes. Thus, the process to form the new FG starts by using *'isSimplified'* and *'Simplify'* methods in order to extrapolate the FGs of the first and second referenced nodes. Then, these two FGs are added to each other where each component of the first FG is added to its correspondence component in the second FG; the result is n_2FG.

Figures 5.13.1 and 5.13.2 show the flowchart and the pseudo code of this type of insertion.

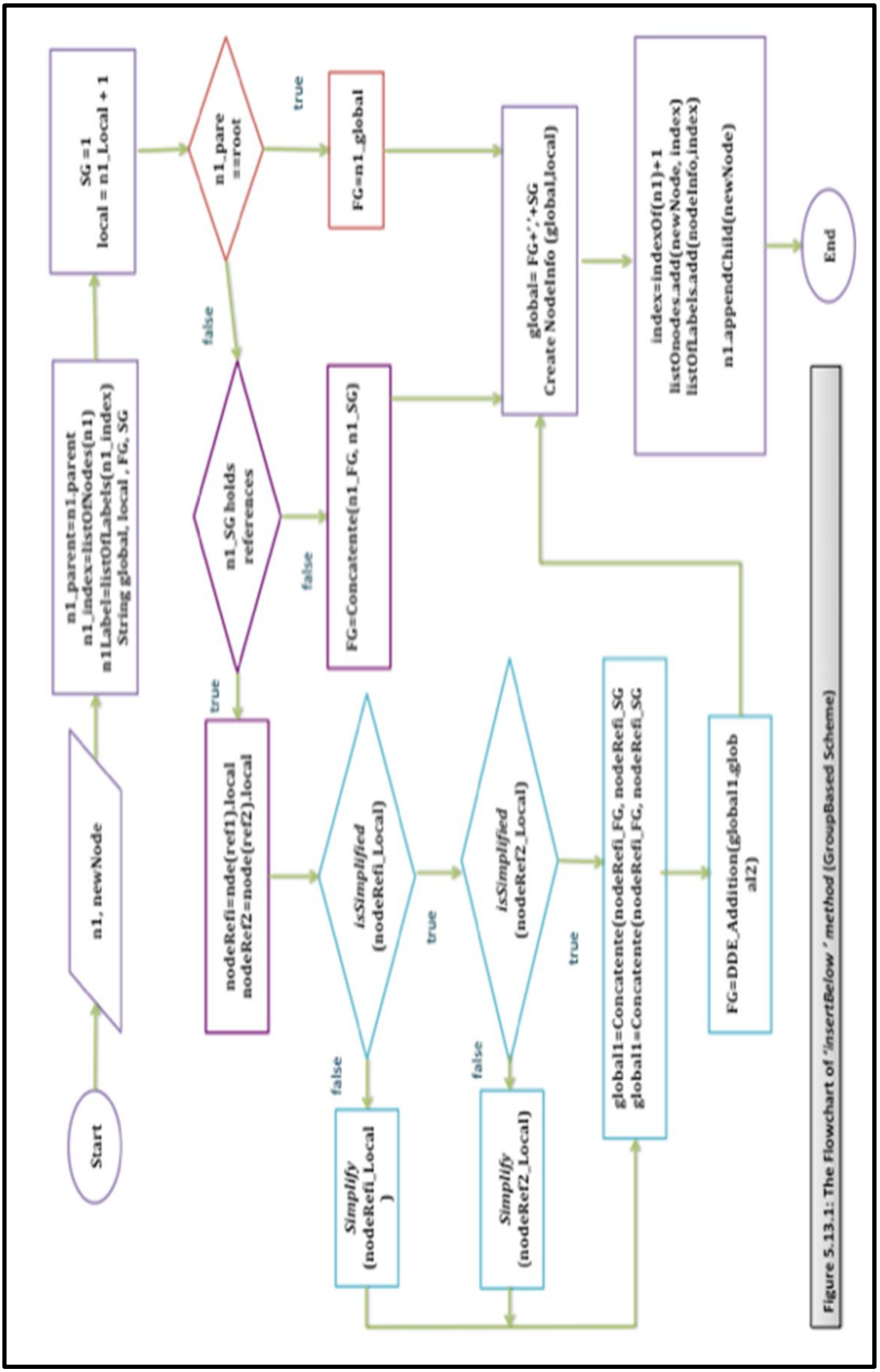


Figure 5.13.1: The Flowchart of "*insertBelow* " *method* (GroupBased Scheme)

```
1)  Get the method's arguments (n1 , newNode)
2) Get n1 parent node
3) Define String variables newGlobal , newLocal, FG, SG
4) index= listOfNodes.indexOf(n1), n1_Label=listOfLabels(index)
5) SG='1', newLocal = n1_Local+1
6) if(parent node == root){
     7) FG= n1_global
}//End if (6)
8) else{
    9) if(n1_SG.contains (references)){
          11)  ref1_local= local label of the referred node by ref1
          12)  ref2_local= local label of the referred node by ref2
           13) if(isSimplified(ref1_local))
                    14) global1=concatenate(FG_Ref1, SG_Ref1)
           15) else{
                    16) Simplify(ref1_local)
                     17) global1=concatenate(FG_Ref1, SG_Ref1)
                }//End else (15)
           18) if(isSimplified(ref2_local))
                    19) global2=concatenate(FG_Ref2, SG_Ref2)
           20) else{
                    21) Simplify(ref2_local)
                     22) global2=concatenate(FG_Ref2, SG_Ref2)
                }//End else (20)
}//End else(8)
12) FG= DDE_Addition(global1,global2)
13) newGlobal=concatenate(FG,',',SG)
14)NodeInfo  newNodeInfo=new NodeInfo(newGlobal, newLocal)
17) index= listOfNodes.indexOf(n1) +1
18) listOfLabels.add(newNodeInfo, index)
19) listOfNodes.add(newNode, index)
20) n1.appendChild(newNode)
```

Figure 5.13.2: The Pseudo Code of *'insertBelow ' method* (GroupBased Scheme)

Explanation of 'isSimplified' and 'Simplify' methods:

1. *'isSimplified':*

   This method takes the referenced node as an argument and returns true if, and only if, the referenced node originally existed within the document which means that the second component of its local label is '0'.

2. *'Simplify':*

   This method is used to calculate the referenced node FG, when the *'isSimplified'* method returns false, which means this node was inserted between two nodes. The process performs an addition between the FGs of the child nodes of the right and the left nodes surrounding the referenced node. This addition may be performed recursively if the surrounding nodes were also inserted until an original node is reached (i.e. initially labelled).

Figure 5.14 shows an output example of these two methods.

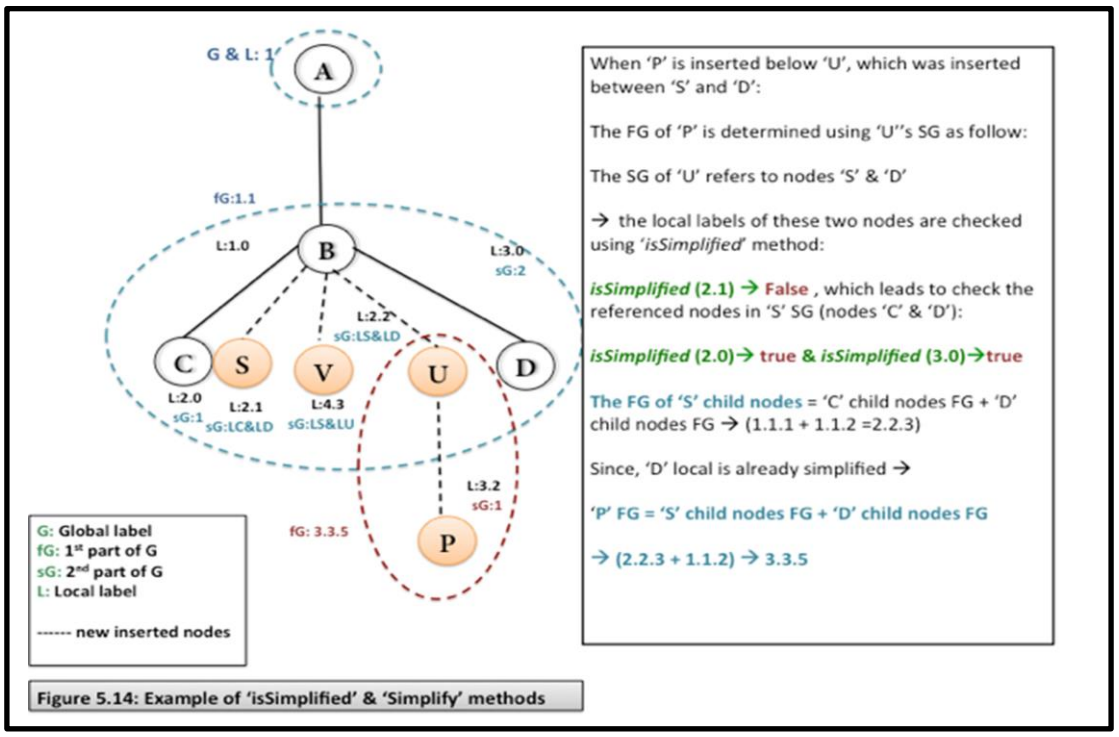


Figure 5.14: Example of 'isSimplified' & 'Simplify' methods

- *DDE Labelling Scheme:*

  Based on the DDE scheme, when adding below a leaf node, the new DDE label is formed by adding '1' as a last component of the n_1 label. Figure 5.15 shows how this is implemented.

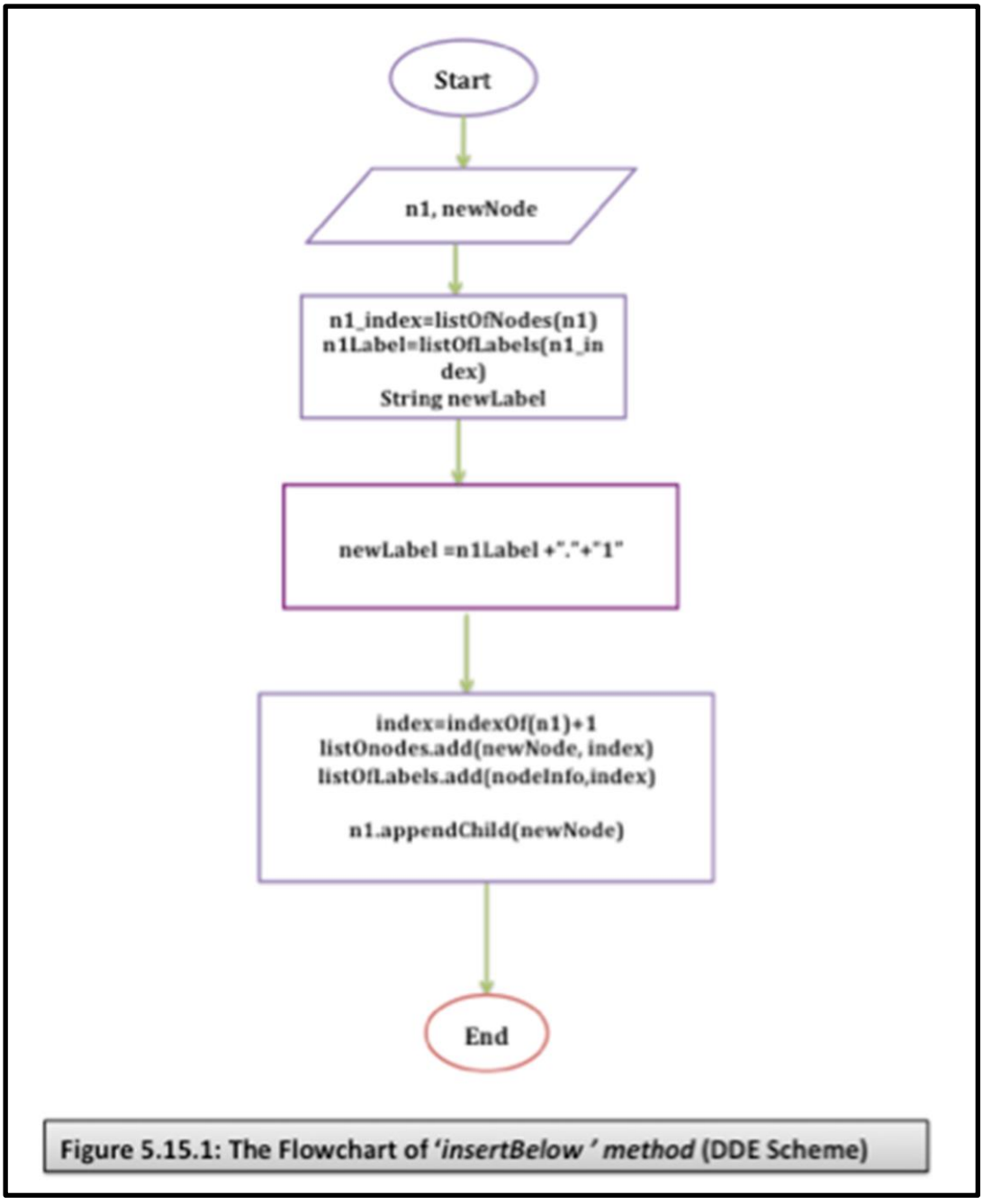


Figure 5.15.1: The Flowchart of '*insertBelow ' method* (DDE Scheme)

```
1)  Get the method's arguments (n1 , newNode)

2) Define String newLabel

3) index= listOfNodes.indexOf(n1), n1_Label=listOfLabels(index)

4) newLabel=n1_Label + "."+"1"

5) index= listOfNodes.indexOf(n1) +1
6) listOfLabels.add(newNodeInfo, index)
7) listOfNodes.add(newNode, index)
8) n1.appendChild(newNode)
```

Figure 5.15.2: The Pseudo Code of *'insertBelow ' method* (DDE Scheme)

5.6 Determining Different Relationships

Five methods have been developed to determine the different relationships. Generally, the implementation phase of these methods was straightforward in both schemes and based on the relationship definitions provided in the previous chapter for the GroupBased scheme and in Xu et al. (Xu *et al.*) for the DDE scheme.

5.6.1 Level

- *The GroupBased Labelling Scheme:*

  For each node the level is calculated by counting the number of components in the first part of the global label. This number is the level unless the second part of the global label does not equal null; in this case the number is incremented by one.

- *DDE Labelling Scheme:*

  The number of the level is the number of components within the DDE label.

5.6.2 Label Order

- *The GroupBased Labelling Scheme:*

  The order between two nodes, n_1 and n_2, is determined by checking their levels first; if the n_1 level is less than the n_2 level, n_1 is before n_2 in the document order and vice versa. However, if they are in the same level, two conditions are checked:

  1. If they belong to the same group, the order is based on a comparison of their local labels or SGs, as described in the previous chapter.
  2. Otherwise, the order is based on the DDE order.

- *DDE Labelling Scheme:*

  Similar to the GroupBased scheme, the order between two DDE labels is determined from their levels. However if the nodes are in the same level, the order is determined as follows:

  n1 precedes n_2 if the result of multiplying the last component in the n_1 label by the 1^{st} component in the n_2 label is less than the result of multiplying the last component in the n_2 label by the 1^{st} component in the n_1 label; and vice versa.

5.6.3 Ancestor/Descendant Relationship (AD)

- *The GroupBased Labelling Scheme:*

  n_1 is an ancestor of n_2 *if* the level of n_1 is less than the n_2 level and one of the following is true:

  1. Their FGs are equal or the FG of n_2 is formed from the n_1 global label, which means they belong to the same group.

  2. The results of dividing each component in n_1*GlobalLabel by* its correspondence in n_2 *GlobalLabel are* equal.

- *DDE Labelling Scheme:*

  In the DDE scheme, the AD relationship is determined as it is in the GroupBased scheme, such that, n_1 is an ancestor of n_2 *if* the level of n_1 is less than the n_2 level and the results of dividing each component in the n_1 label by its corresponding component in the n_2 label *are* equal.

5.6.4 Parent/Child Relationship (PC)

- *The GroupBased Labelling Scheme:*

  n_1 is a parent of n_2 *if* the level of n_2- n_1 level = 1 and they belong to the same group or if the n_1*FG* = 1 and n_2 level =2.

- *DDE Labelling Scheme:*

  n_1 is a parent of n_2 *if* the level of n_2- n_1 level = 1 and the results of dividing each component in the n_1 label by its corresponding component in the n_2 label *are* equal.

5.6.5 Computing the Lowest Common Ancestor (LCA)

In both schemes, the computation of LCA is based on the AD relationship computation.

5.7 Conclusion

To conclude, this chapter described how the new labelling scheme was designed and implemented based on the theoretical discussion presented in the previous chapter. In addition to the new scheme's implementation, the DDE labelling scheme was also implemented because of its role in forming the new scheme. Both schemes were discussed in parallel from a practical point of view; thus, each aspect of the implementation was described in terms of both schemes in order to show the differences between the two. Flowcharts and pseudo codes were presented to offer more clarification and to give visual guidance.

Chapter 6: Experimental Framework

6.1 Introduction

The proposed labelling scheme was explained in detail in Chapters 4 and 5. Testing the scheme's performance and scalability is a key to evaluating the scheme accurately and so a set of experiments was carried out. This chapter describes the experiments and data used in the evaluation process.

Four experiments were performed to test different aspects of the proposed GroupBased labelling scheme. Each experiment was carried out on both the GroupBased scheme and the DDE scheme; thus, comparisons between the two schemes were possible. These experiments were designed to evaluate the scheme's ability when the XML documents were static with no insertions, and when the XML documents were dynamic. All the experiments evaluated the scheme's performance in terms of time and the size of labels.

The remainder of this chapter is divided as follows: Section 6.2 describes the experimental setup and the platform used in the experiments. Then, the objective of each experiment is explained in Section 6.3. The evaluation criteria are outlined in Section 6.4 while some of the available XML datasets are reviewed and the experimental data chosen is specified in Section 6.5. The aim of each query used in the experiments is described in Section 6.6. Finally, the chapter is concluded in Section 6.7.

6.2 The Experimental Setup and the Implementation Platform

For the comparison between different dynamic labelling schemes, DDE was chosen as it plays a role in forming the proposed labelling scheme, as mentioned in Chapter 4. In addition, in choosing between the comparable schemes, supporting

the efficient computation of different relationships was considered as a primary factor, as well as the ability to handle insertions without the need for re-labelling. These factors are available in the DDE scheme. Then, the published results from other schemes, which were compared to the DDE scheme, were also used in the comparison.

The data sets used in the experiment and their characteristics are described in Section 6.4. All experiments were conducted on a laptop with 2.7 GHz Intel Core i7 CPU, 4 GB for main memory and with an OS X 10.9.2 operating system. NetBeans IDE 8.0 and Java JDK 1.8 were used in the implementation of both the proposed labelling scheme and the DDE scheme.

6.3 An overview of the experimental framework

The main objective of running the experiments described in this chapter was to evaluate the proposed labelling scheme as accurately as possible in order to assess the hypothesis stated in Chapter 1. The setup of these experiments was designed based on the details described in Chapters 4 and 5 in order to determine whether the scheme's design and implementation met the objectives of the scheme mentioned in Chapter 1. Four experiments were carried out to evaluate the scheme's performance, scalability and efficiency. However, the lack of a universal platform in which all XML labelling schemes can be experimented on in the same test-bed in order to prove their effectiveness, led to a challenge in verifying the scheme's credibility. Hence, the DDE labelling scheme was also implemented from scratch, so that the characteristics of the proposed scheme could be tested against the characteristics of the DDE scheme. Then, transitivity logic was used to compare the results of the experiments on the proposed scheme with other published results where the DDE scheme was compared to other schemes. These experiments served the overall scientific approach of this kind of work where a more specialised approach requires intensive experiments involving software engineering tests. Thus, no rigorous hypotheses can be developed from these experiments.

By implication, even though there was a hypothesis from the first chapter of the study, the experimental framework provided that the study was conducted with focus on an inductive and deductive research approaches. As explained by O'Leary (2006), a deductive approach is generally suitable for scientific research of this nature. This is because in thus scientific research, a deductive approach is used by developing a hypothesis which is tentatively tested and examined to establish a theory (Creswell, 2007 , Hardy and Bryman, 2004, Ridley, 2012, Saunders *et al.*, 2011).

It was not possible to use deductive approach alone because no rigorous hypothesis was developed based on the experiment. As has been explained earlier, the use of deductive approach alone would have required a common test-bed based on which the performance, scalability and efficiency of labelling schemes can be assessed and none exists. Meanwhile, Sapsford & Jupp (2006) indicated that for a rigorous hypothesis to be set, on which deductive research could be carried out, it is important that there is an easily substantiated framework or platform on which the hypothesis can be tested. In the absence of such a framework or platform, the hypothesis cannot consider a rigorous hypothesis but only a guide hypothesis that specifies what needs to be achieved by the study.

Also writing on research approaches, Riley *et al.* (2000) suggested that in such a scientific research as this where cannot be a rigorous hypothesis due to lack of a test-bed based on which the hypothesis can be justified, it is important that a combined approach that involves an inductive approach is used. It was based on this that the experimental framework used a combined approach comprising both a deductive and inductive approach. Yin (2009) explained an inductive research approach is one which provides the researcher with greater flexibility and opportunity to modify the research emphasis depending on the accumulated findings throughout the research process. As a result of this, instead of exclusively basing the work on the hypothesis defined in the first chapter, part of the research approach was inductive, where the researcher's main basis for drawing

conclusions on the performance, scalability and efficiency of the proposed labelling scheme was taken from the accumulated findings throughout the research process.

It is important to emphasise here that the research process as used in this case was experiment. As a result of the inductive approach, the researcher was afforded the opportunity of using transitivity logic in which a qualitative approach to analysis, together with quantitative analysis, where the differences in readings between the DDE and GroupBased experiments were interpreted to draw conclusion on the performance, scalability and efficiency of the proposed scheme. Having said this, it must be acknowledged that such software engineering testing strategies as unit testing, integration testing and system testing could all be used in developing a rigorous hypothesis from experiments.

The experiments can be grouped based on the type of XML document: either static or dynamic. All the experiments were run both on the proposed scheme and the DDE scheme. The first three experiments are applicable for both types of document while the last experiment was designed to run on dynamic XML documents. These experiments were as follows:

- The initial labelling
- Determining different relationships
- Query performance
- Handling insertions

The following section describes these experiments in detail, along with their objectives

6.3.1 Objectives of the Experiments

- ***The Initial Labelling:***

  This experiment aimed to evaluate the initial labelling process by measuring two factors: the time required to label the document and the growth of the label's size, and how these factors are affected by the document size. Then, the results of each scheme are compared. It would be logical to expect the labelling time to increases as the document size increases but the rate of increase in the two schemes is of interest as a comparison of memory required. The proposed scheme labels are shorter in complex documents and so could be expected to take less storage but the level of complexity at which this occurs is not initially obtained.

- ***Determining Different Relationship:***

  This experiment was run on both static and dynamic XML documents. The aim of this experiment is to find out how fast the five relationships mentioned in Chapter 4 can be determined from the labels before and after insertions. The experiment consisted of five mini experiments where each one was run to test a specific relationship. These relationships are:

  - Finding the order between two nodes
  - Finding the node's level
  - Finding the ancestor/descendant relationship
  - Finding the parent/child relationship
  - Finding the lower common ancestor between two nodes

  These experiments were run on both schemes before and after document modification.

- ***Query Performance:***

  This experiment aimed to test the performance of different types of query on the labelled document before and after insertions. Nineteen types of query were executed; these were varied in their purpose and complexity. Section 5.6 describes these queries and their purposes. The expected result was that the queries' response times in the proposed scheme would be less than in the DDE scheme, especially when the document is dynamic.

- ***Handling Insertions:***

  This experiment was only applicable to the dynamic document. It tested the scheme's scalability in handling different types of insertion: ordered skewed insertions and random skewed insertions. Ordered skewed insertions refer to those which repeatedly perform leftmost and rightmost insertions on a particular node, whereas random skewed insertions refer to nodes which are repeatedly inserted between two consecutive nodes in random order. This experiment measured two factors: the size of the labels after the insertions and the time required for each type of insertion. Then, how the number of insertions affected the time was tested. The expected result was that the DDE scheme would show a slightly better performance in terms of time and memory allocation for the labels.

6.4 The Experimental Evaluation Criteria

To achieve the aims of the experiments the following criteria must be specified:

- The experimental environment in terms of hardware and software are identified (Sec. 6.2)
- The datasets used in the experiments are specified, based on the experiments' aims and objectives (Sec. 6.5)
- The boundaries of each experiment, as well as the measurement's unit, are stated (Sec. 6.3)

- The expected results from each experiment are outlined (Sec. 6.3)
- Finally, the experimental results are analysed and the scheme is evaluated, as are described in Chapter 7 and Chapter 8.

6.5 A Review of Existing XML Datasets

In this section, some of the most widely used XML datasets are briefly reviewed and the datasets used in the experiments are specified. XML datasets can be divided into two types: XML benchmarks, which are used to generate synthetic datasets in XML format where the size of the generated XML document can be specified as required; and XML datasets, based on real public data where each dataset is a validated XML document (this type of dataset is referred to as a *real-life* dataset). However, both types of dataset were commonly used to assess the performance and the functionality of XML schemes or systems. In addition to evaluating the characteristics of the XML schemes, XML benchmarks allow the query performance to be evaluated by providing a set of range queries that simulate real-world scenarios in order to assess the XML database when applying the new scheme; this facilitates comparisons to be made between different XML schemes(Schmidt *et al.*, 2001).

6.5.1 XML Benchmarks

Designed for the storage of data and the processing of queries (Schmidt *et al.*, 2001), XML benchmarks are divided into application benchmarks and micro benchmarks. The purpose of application benchmarks is to assess how the XML database performs overall, in terms of data as well as queries. On the other hand, micro benchmarks are geared towards the assessment of features of a particular system component, such as query processing (Barbosa *et al.*, 2002, Mlýnková, 2008, Runapongsa *et al.*, 2006b, Yao *et al.*, 2004). In the following, the most commonly used XML benchmarks are presented.

- ***XOO7 Benchmark:***

  Initially formulated by Carey *et al*. (1994), the Object Oriented RDBMS benchmark (OO7) was applied by Li *et al.* (2001) to the XML environment. In order to be employed in the XML version of the benchmark (XOO7), the OO7 data and query sets were subjected to modifications. In addition, XOO7 produces an XML data set as a separate XML file in three different sizes: small, medium and large. However, the assessment of scalability is limited by the size restrictions on the data sets. This data set has a constant depth of five levels, regardless of size. The query set consists of twenty-three queries that target search processes without update (Li, 2003). This benchmark is available for free from the XOO7 benchmark website (Li, 2003).

- ***XMark Benchmark:***

  Designed by Schmidt *et al.* (2002), the XMark benchmark is frequently employed in the assessment of XML Applications (Arion *et al.*, 2004, Chen *et al.*, 2006, Davis *et al.*, 2003, Lawrence, 2004, Lee *et al.*, 2010, Li *et al.*, 2007, Lu *et al.*, 2005, Wang *et al.*, 2003, Wang *et al.*, 2005). It can reproduce an XML database in a range of sizes, while the query set encompasses the majority of query-related aspects. The dataset is produced by XMark as a single XML file incorporating simulated data pertaining to an auction website. Understanding an XMark dataset is straightforward. The XMark dataset generator can be downloaded free of charge from the XMark project website (Schmidt, 2003). A scaling factor regulates the database size, therefore enabling developers to produce data sets that suit their requirements. Moreover, the assessment of system performance can be effectively conducted with the use of XMark data sets, particularly with regard to scalability. Similar to the XOO7, irrespective of the XML file size, the XML tree or depth has a constant number of twelve levels; it presents a repetitive structure with a considerable number of recursions (Chen *et al.*, 2005, Zhang *et al.*, 2005) and it includes a query set intended for the

evaluation of a number of features of databases. However, it does not include update transactions; its twenty queries address only searching transactions (Schmidt, 2003).

- ***XBench Benchmark:***

  A template-based benchmark, XBench produces a broad range of XML files, including data centric XML files (DC) and text centric XML files (TC). The database can take the form of either a single XML document (SD) or a multiple one (MD). The *toXgen* tool can be used to generate four types of XML database: namely, DC/SD, DC/MD, TC/SD, and TC/MD. The sizes of the XML databases supported by this benchmark are four: small (10 MB), normal (100 MB), large (1 GB) and extra large (10 GB) (Yao *et al.*, 2003, Yao *et al.*, 2004). Similar to XOO7, the database sizes are constant. However, this benchmark differs from XMark and XOO7 in that it enables a restricted choice of the number of levels established by parameter. The benchmark comprises twenty queries designed to search without update.

- ***XMach-1 Benchmark:***

  Böhme and Rahm (2003) first created XMach-1 as a multi-user benchmark. As such, it is underpinned by a web-based application scenario and is composed of four parts: namely, the XML database, server, loader and client. The structure of the XMach-1 benchmark is presented in Figure 6.1. The data set encompasses a great number of small XML files. According to the number of XML files, the data set displays four versions. All XML files range in size from 2 KB to 100 KB. The maximum number of levels is six, while the query set consists of eleven queries. Of these queries, eight focus on search processes, whilst the remaining three are concerned with update transactions (Böhme and Rahm, 2003).

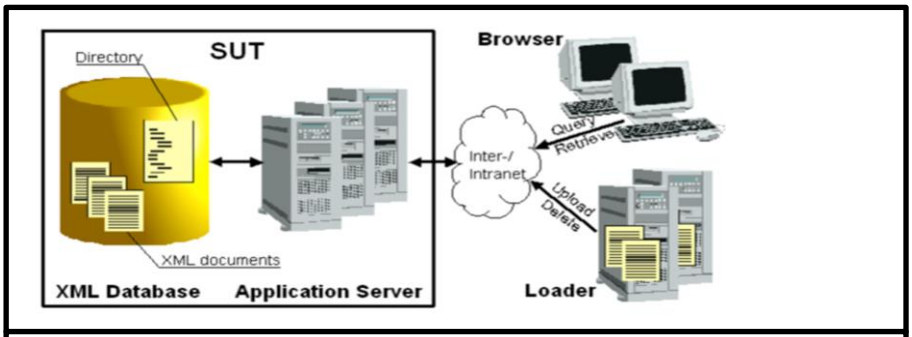


Figure 6.1: XMach-1 Structure of the Benchmark(Böhme and Rahm, 2003)

- ***The Michigan Benchmark:***

  The Michigan Benchmark was designed by Runapongsa *et al.* (2006a), who labelled it as a micro benchmark intended for the assessment of particular system features (Runapongsa *et al.*, 2006a, Yao *et al.*, 2004). The data set takes the form of a single XML file consisting of at least 728,000 nodes and can be ten times the size. The data set has a depth of sixteen levels and an adjustable breadth. The latter is set by a fan-out parameter, with a minimum and maximum value of, respectively, two and thirteen nodes at each level. The thirty-one queries included in the query set focus on the assessment of a number of dimensions of databases, including update processes (Runapongsa *et al.*, 2006c). The benchmark is available on the project website (Runapongsa *et al.*, 2006c).

- ***TPoX benchmark:***

  Transaction Processing over XML (TPoX) is an application benchmark intended for the assessment of the entire system. Templates influence the production of XML file. The size of the XML files is regulated by an XML Schema, which establishes database depth and breadth. The database takes the form of numerous small XML files, ranging from 2 KB to 20 KB (Nicola *et al.*, 2007). The query set comprises seventeen queries which, in contrast to other benchmarks that focus more on search processes, are concerned with updating the XML database.

  The benchmark is available on the project website (Nicola *et al.*, 2007).

Table 6.1, shows certain features of certain XML benchmarks.

	XOO7	XMark	XBench	XMach-1	The Michigan	TPoX
Type	Application-Level	Application-Level	Application-Level	Application-Level	Micro	Application-Level
No. Users	Single	Single	Single	Multiple	Single	Multiple
No. Applications	1	1	4 (TC/SD, TC/MD, DC/SD, DC/MD)	1	1	1 (Complex)
Document in Data Set	Single	Single	Single/ Multiple	Multiple	Single	Multiple
Data Generator	✓	✓	✓	✓	✓	✓
Key Parameters	Depth, fan-out, size of textual data	Size	Size	Number of documents / elements / words in a sentence, probability of phrases and links	Size	Size + number of users
The size	3 sizes (small,	Differs from tiny	Small (10MB) ,	The size starts	Single document	The size starts

	medium, large) with pre-defined parameters	(KB) to huge (Ogbuji)	normal (100MB), large (1GB) ,huge (10GB) document	from 2 to 100 KB per document	with 728 000 nodes as a min and 10 times more as a max	from 2 to 25 KB per document
Schema of Document	DTD derived from 007 relational schema	DTD of an internet auction database	DTD/XSD	DTD of an document having chapters, paragraphs and sections	DTD/XSD of the recursive element	XSD
Average / Max Depth	5/7	6/12	Limited	3/6	5/16	Controlled by template
No. Queries	**23**	**20**	**20**	**11**	**31**	**17**
No. Updates	**0**	**0**	**0**	**3**	**3**	**10**

6.5.2 *Real-Life* XML Datasets

Compared to synthetic benchmark data sets, realistic data and structures encompassed in these datasets facilitate the assessment process. In the following sections, an overview of the available *real-life* datasets employed in XML assessments is provided. Each dataset can be accessed and downloaded free of charge from the XML Data Repository website (Suciu, 2002).

- ***Protein Sequence Database:***

  Created by Georgetown University, the protein sequence database provides information about integrated bioinformatics, including protein sequences. Similar to the DBLP, this dataset is an XML file of 683 MB in size with a simple, broad and regular structure (Wong *et al.*, 2007) while its depth expands over seven levels. Among the applications that use it for the assessment are of experiments on XML storage (Wong *et al.*, 2007), XML stream processing (Green *et al.*, 2003, Jittrawong and Wong, 2007), and filtering (Silvasti *et al.*, 2009, Suciu, 2002).

- ***SWISS-PROT:***

  The size of the XML file of the swiss-port dataset is 109 MB (Suciu, 2002). It provides a high quality, annotated, protein sequence database that manages to maintain a minimal redundancy level. It also efficiently supports integration with other databases. (Suciu, 2002, UniPort, 2014). This dataset is employed in the assessment of a variety of XML technologies such as query processing (Gulhane and Ali, 2013, Rao and Moon, 2004).

- ***Auction Data:***

  The auction dataset represents auction data from web sources such as EBay, Yahoo and UBid. Besides the lack of attributes in this dataset, the XML files of these auctions are very small since the largest file is only 34 KB (Suciu, 2002); this may limit its usage in an evaluation of XML technologies.

- ***DBLP Computer Science Bibliography:***

  The Digital Bibliography Library Project (DBLP) database is a large XML file that contains authentic bibliographic information related to computer science publications, including important conferences (e.g. VLDB, ICDE), journals (e.g. TODS), series (e.g. LNCS/LNAI), as well as books (Suciu, 2002, DBLP, 2013). This dataset is employed by a wide range of XML database applications (Al-Badawi, 2010, Chen *et al.*, 2006, Lawrence, 2004, Li *et al.*, 2007, Liefke and Suciu, 2000, Lu *et al.*, 2005, Wang and Liu, 2003, Xu and Papakonstantinou, 2005) for assessment experiments. The structure of the dataset is straightforward and wide (Chen *et al.*, 2006, Lee *et al.*, 2010, Lu *et al.*, 2005). It is possible to download the original version of the dataset from the DBLP website (DBLP, 2013). However, the dataset is very large, being, as of March 14th 2013, approximately 1.1 GB (DBLP, 2013).

- ***University Courses:***

  This database includes information related to the courses provided by three different academic institutions. It presents three small versions of 277 KB, 1 MB and 2MB, respectively. The first and second versions each have four levels, while the third version has two levels. As noted by Suciu (2002), despite the low number of versions, the data set enables, to some degree, the performance of scalability tests due to its varying sizes.

- ***Treebank:***

  Designed by the Computer and Information Science Department at the University of Pennsylvania, the Treebank Database comprises English sentences explained for linguistic structures. To safeguard copyright for

text nodes, the database has partial encryption; this has no impact whatsoever on the XML structure. Moreover, the dataset exhibits a deep recursive structure, which makes it relevant to assessment experiments (Chen *et al.*, 2006, Chen *et al.*, 2005, Lu *et al.*, 2005, Onizuka, 2003, Wong *et al.*, 2007). Deemed to be a complex XML database, the tree encompasses a large number of nested structures (386,614) (Onizuka, 2003). The most frequent use of this data set is in the assessment of various dimensions of different XML applications (Chen *et al.*, 2006, Chen *et al.*, 2005, Green *et al.*, 2003, Li *et al.*, 2007, Liefke and Suciu, 2000, Lu *et al.*, 2005, Onizuka, 2003, Steedman *et al.*, 2003, Wong *et al.*, 2007). The XML file is 82 MB in size (Suciu, 2002, Treebank, 1999).

- ***NASA:***

  The NASA database consists of authentic astronomical data, having been developed from a flat file format during the GSFC/NASA XML Project. The XML file is 23 MB in size (Suciu, 2002, Nasa, 2001). In comparison to Treebank, this dataset has a shallow structure, displaying only 18 recursive elements (Onizuka, 2003). Its primary use is in the evaluation of various XML applications intended for XPath and XML query processing (Green *et al.*, 2003, Jittrawong and Wong, 2007, Onizuka, 2003, Wong *et al.*, 2007, Zhang *et al.*, 2005), indexing methods (He and Yang, 2004), labelling (Wu *et al.*, 2004), filtering (Silvasti *et al.*, 2009), and searching (Lee *et al.*, 2010).

- ***SIGMOD Record:***

  With an XML file size of about 0.5 MB (Merialdo, 1999, Suciu, 2002) sigmod record is usually used in the performance evaluation of small XML databases (Lawrence, 2004, Lee *et al.*, 2010, Li *et al.*, 2007, Li and Moon, 2001, Rafiei *et al.*, 2006, Wu *et al.*, 2004), this database contains real data pertaining to certain articles circulated by the ACM SIGMOD website(Suciu, 2002).

- ***TPC-H Relational Database Benchmark:***

  TPC-H dataset is a well-known relational benchmark and it has been widely used in a relational context (Duan *et al.*, 2011). However, this benchmark is converted to XML as a representation of transactional processes (Suciu, 2002). It has been used in assessments of XML technologies (Baralis *et al.*, 2007, Shah *et al.*, 2009) but not as widely as in relational database evaluations.

- ***Mondial:***

  Mondial dataset provides statistical geographic information about the world's countries (Suciu, 2002). It has been used in evaluating the performance of XML applications such as query related technologies (Atique and Raut, 2012, Senellart and Souihli, 2010), XML comparisons techniques (Sakr, 2009) and XML search analysis (Balmin *et al.*, 2009).

Table 6.2: Some features of the existing real-life dataset

Dataset Name	Protein Sequence	SwissProt	Auction Data				DBLP	University Courses		
Size	683 MB	109 MB	23 KB	34 KB	19 KB	24 KB	172 MB	277 KB	2 M	1M
No. Nodes	21,305,818	2,977,031	311	156	342	342	3,332,130	10,546	66729	74557
No. Attributes	1,290,647	2,189,859	0				404276	0	6	0
Avg. Depth	5.15147	3.55671	3.7				2.90228	3.19979	3.95243	3.15787
Max Depth	7	5	5				6	4	5	4

Dataset Name	Nasa	SIGMOD	TPC-H								Treebank	Mondinal
Size	32 MB	467 KB	603 KB	30 MB	2 MB	28 KB	5 MB	4 KB	787 K	503 K	82 MB	1 MB
No. Nodes	476,646	11,526	20,001	1,022,976	48001	801	150001	126	21	13501	2,437,666	22423
No. Attributes	56317	3,737	1								1	47423
Avg. Depth	5.58314	5.14107	2.8999	2.94117	2.8333	2.87266	2.89999	2.78571	2.66667	2.88875	7.87279	3.59274
Max Depth	8	6	3								36	5

6.5.3 The Experimental Datasets

Both real and artificial datasets were chosen. From the XML benchmarks (Sec. 6.4.1), XMark was chosen, along with its queries set, as a baseline dataset for all the experiments described in Section 6.3. This benchmark was used in DDE scheme experiments and in other comparable schemes and beside has all the features and characteristics needed in evaluating the proposed labelling scheme as discussed in Section 6.5.4. From the *real-life* datasets (Sec. 6.4.2), 'Nasa' and the 'TPC-H' were used, for the initial labelling experiment; in order to test the scalability of the proposed scheme in terms of the type of the tree (i.e. wide and deep tree structure). Table 6.3 shows the chosen datasets along with their sizes.

6.5.4 The XMark Benchmark

Data is a critical part of any experiment, and all precautions have to be taken in ensuring that the data covers all experimental aspects such as being retrievable whenever it is needed. However in the relational database management systems (DBMS), the challenge to store data in well-arranged tables, which is not the case for Extensible Markup Language (XML). XMark, a benchmark specifically for XML (Al-Khalifa *et al.*, 2002, Franceschet, 2005, Yao *et al.*, 2004) was invented to solve this problem. The XMark suite assists the users and developers to gain insights into the behaviour of their XML storehouses (Wang and Meng, 2005. This section first discusses the various features of the XMark benchmark, which make it a useful tool for many developers and users. The range of queries of the XMark benchmark is examined, showing how each of them makes this dataset a good choice in the evaluation of the XML labelling schemes. In an XML tree, there is a node for each document's element, attribute and value (O'Neil, 2004).

The XMark benchmark is able to generate various sizes of XML files by making use of a data generator called XMLGen, which enables it to create synthetic XML documents according to a fixed number of DTD (Document Type Definitions) of an internet auction database (Kochmer and Frandsen, 2002). This benchmark features a tool kit for evaluation of the retrieval performance of XML stores and query processors. The benchmark is scalable and allows a comprehensive set of queries designed to feature natural and intuitive semantics (see Section 6.6). To the facilitate the analysis and interpretation, each specific query is meant to utilise a primitive of the query language; this generally challenges the query processor.

Since XMark benchmark is platform independent, any user interested in running it can download the binary and generate the same document regardless of the hardware or operating system the developer is using; thus, making experiments reproducible. It is also accurately scalable and therefore can be restricted by the system's capacity. It is both time and resource efficient, and therefore elapsed time will scale linearly when the resource allocation is constant, regardless of the size of the generated document (Yoshikawa *et al.*, 2010). The ability of the XMark benchmark to meet the above demands by making use of the XMLGen makes it desirable in evaluating the proposed scheme. There are alternative to XMark (see Section 6.5.1) but none of them offer this linear scalability.

6.6 The Objectives of the Experimental Queries

As mentioned in the previous section, the XMark queries set was chosen to test the query performance. Nineteen out of twenty queries were implemented and were grouped based on their objectives, as described in the following table:

Table 6.3: The experimental queries set and their description

No. Query	Purpose	Description
Q1	Exact match	*Return the name of the person with ID 'person0'.*
Q2	Ordered access	*Return the initial increases of all open auctions*
Q3		*Return the first and current increases of all open auctions whose current increase is at least twice as high as the initial increase*
Q4		*List the reserves of those open auctions where a certain person issued a bid before another person.*
Q5	Casting	*How many sold items cost more than 40?*
Q6	Regular Path Expression	*How many items are listed on all continents?*
Q7		*How many pieces of prose are in our database?*
Q8	Chasing references	*List the names of persons and the number of items they bought. (joins person, closed auction)*
Q9		*List the names of persons and the names of the items they bought in Europe. (joins person, closed auction, item)*

No. Query	Purpose	Description
Q10	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.*
Q11		*For each person with an income of more than 50000, list the number of items currently on sale whose price does not exceed 0.02% of the person's income.*
Q12	Reconstruct portions of the original XML document.	*List the names of items registered in Australia along with their descriptions.*
Q13	Full text	*Return the names of all items whose description contains the word 'gold'.*
Q14	Path traversals	*Print the keywords in emphasis in annotations of closed auctions.*
Q15		*Return the IDs of the sellers of those auctions that have one or more keywords in emphasis.*
Q16	Finding missing elements	*Which persons don't have a homepage?*

No. Query	Purpose	Description
Q17	Function Application	*Convert the currency of the reserves of all open auctions to another currency.*
Q18	Sorting	*Give an alphabetically ordered list of all items along with their location.*
Q19	Aggregation	*Group customers by their income and output the cardinality of each group.*

Table 6.3: shows the experimental queries and their description

Only query 10 in the XMark queries (Schmidt *et al.*, 2002) was ignored as it tests the database's ability to translate the constructed results into another language to avoid the simple copying of the original database; this is not related to the proposed scheme's characteristics (i.e. it will not provide any pros or cons regard to labelling scheme) and so was irrelevant.

6.7 Conclusion

In order to evaluate the performance and scalability of the proposed scheme, four sets of experiments were performed on static and dynamic XML documents. This chapter outlined these experiments along with their objectives. Then, the experimental setup was discussed and the datasets used were determined after briefly examining some *real-life* datasets and the existing XML benchmarks. The expected result from each experiment was outlined and the aims of the experimental queries were described.

The following chapter presents the results from these experiments along with their analysis.

Chapter 7: Results and Analysis

7.1 Introduction

In this chapter, the results of the experiments are described and analysed. Chapter 6 described the experiments used in evaluating the GroupBased labelling scheme. Four experiments were designed to evaluate the scheme's functionality and performance in both static and dynamic XML documents. The first experiment evaluated the time needed for the initial labelling process along with the labels' size. The second experiment focused on assessing the time needed to determine the different relationships. The third experiment evaluated the queries' response times before and after insertions. Finally, the fourth experiment evaluated the scheme's ability to handle different types of insertion. As mentioned in the previous chapter, these experiments were also run on the Dynamic Dewey labelling scheme (DDE) to permit comparable evaluations to be made between both schemes under the same circumstances.

In similar research by Fennell (2013) to test the performance and functionality of labelling schemes on XML documents, Fennell (2013) noted that outcomes of test the same could fluctuate between repeated experiments. This means that relying on only a few tests for each experiment in the design of the new scheme could damage the credibility of outcome Murata, Kohn and Lilley (2009). Based on this, for each of these four different experiments, tests were repeated. This was done to find the most consistent line of results to use in the study's results and analysis. The results that are presented in this chapter of the study therefore represent the outcome of the average of 20 different test runs performed on each variable that was tested under each of the four experiments. Once this was done, statistical analysis was performed on the outcomes to validate the results

As far as the presentation of results and analysis is concerned, it is important to stress that this chapter has two major types of approaches to the presentation of findings. The first has to do with the use of graphs, which give a pictorial outcome of the experiments. The results from the graphs are more or less descriptive in nature as they describe the performance or behaviour of the two major schemes, GroupBased labelling scheme and DDE scheme. The second type of presentation of finding takes a more statistical approach to the results that were gathered as it gives the outcome of the significance of the results using the Wilcoxon rank-sum test. Thus the rest of the chapter is organised as follow:

Section 7.2 discusses how the statistical significance of the results is computed. Section 7.3 presents the different sizes of the XML files used in the experiments. Section 7.4 discusses the graphical outcomes of the static document experiments along with their statistical interpretation. Similarly, the outcomes of the dynamic document experiments are discussed in Section 7.5. The chapter concludes in Section 7.6.

7.2 Statistical significance of the results

The graphical presentation of results is very important in providing a descriptive overview of the differences that exists between the GroupBased labelling scheme and the DDE scheme. However, Harold (2004) argued that differences recorded between the performance and functionality of two schemes for XML documents may not necessarily imply that the two schemes cannot be used interchangeably to achieve the same outcomes. To get the real import of the differences therefore, Benjamini (2008) recommended finding the statistical significance of the results obtained between the two schemes. This is because finding the statistical significance deals with the introduction of a null hypothesis, which seeks to equalise the viability and usability of the two schemes until the equalisation is proven otherwise with an alternative hypothesis. For this study, the statistical significance was focused on the time-related experiments. The study featured the use of time-related performance and size-related performance. Only the time-

related experiments were further tested for statistical significance as the size-related experiments gave almost the same line of results in both schemes, making it easier to draw conclusions that there was no significance difference between the results in terms of size. To find the statistical significance of the time-related results, two important statistical procedures or tests were used, which are box plot technique and Wilcoxon rank-sum test. These two were used in an interrelated manner but were relevant for separate purposes. The reason for using each and how the two contributed to the determination of statistical significance of the results is outlined below.

7.2.1 Overview of Statistical Significance Tests

One of the statistical significance techniques used was the box plot method. Box plots have been found to be ideal for graphically presenting groups of numerical data through the use of quartiles. Even though the box plot method also makes use of graphical presentation and could be said to be a type of descriptive statistics, it was described under the statistical significance because of the need to use the outcome with the quartiles to find p-values using Wilcoxon rank-sum test. By implication, the box plot was not used totally in isolation from the Wilcoxon rank-sum test. The major rationale for selecting the box plots is that there are non-parametric in nature. What this means is that they make use of statistics that are not based on parameterised families of probability distributions. Meanwhile, the outcomes with the two samples namely GroupBased labelled scheme and DDE scheme were not decided based on parameters necessary for achieving relevant specification of the XML document as done with a typical parameterised family of probability distribution (Cunningham, 2006). The box plots therefore made it possible to obtain information about the probability distribution in terms of how the two samples impacted on the dependent variable of time. Visually, this was done as the box plots portrayed extreme values by showing differences between distributions.

The Wilcoxon rank-sum test was used as a non-parametric statistical hypothesis test to compare the two related samples, which are GroupBased labelled scheme and DDE scheme. The reason for doing this was in order to discover if the two related samples have population mean ranks that differ. This makes the Wilcoxon rank-sum test a paired difference test given the fact that its statistical numeration of outcomes with results among the two related variables was done in pairs as showed with the box plot instead of individually. This further brings out reason the statistical significance test is regarded as non-parametric as the outcomes with the two schemes were not based on parameterised families of probability distribution. Because the two independent samples were non-parametric, they could not meet the requirements for the t-test. In cases like this, Rousseeuw, Ruts and Tukey (1999) noted that the Wilcoxon rank-sum test becomes useful in drawing statistical significance based on the p-value.

In line with the above position to use the p-value instead of the t-test, the box plot was used to perform r-statistic that calculated the significance of the results using the Wilcoxon rank-sum test. This is why it was said earlier that the two tests were used interchangeably. When used with the p-value, the statistical significance of the two samples, which in this case are the GroupBased labelled scheme and DDE scheme are determined by setting a null hypothesis that neutralises their significance. In this context, the null hypothesis (H_0) states that at significance level of 0.05, scheme has no effect on time. Based on the box plot, the null hypothesis will be accepted upon the probability that the populations for each sample have the same medians. In the next sub-section therefore, the results of p-values as found for ten different parameters within the time-related experiments are presented and interpreted for significance.

7.2.2 Significance interpretation of results

There is a very simple interpretation given to the figures produced by way of the box plot and the Wilcoxon rank-sum test that was performed. The interpretation is regarded as simple because it emphasises the use of the p-values in determining the significance between the tests. The significance interpretation of results is based on ten experimental measures, under each of which the null hypothesis will be tested based on the p-values obtained. The 109 figures of box plots were shown in Appendix A.

7.3 Experimental Data

As mentioned in Chapter 6, XMark benchmark, Nasa and TPC-H in various sizes were the datasets chosen for the experiments. As shown in Table 7.1, the XML file (xml2) was used in all experiments because of its small size (1 MB) which simplifies the tracking the changes in the document when performing insertions; it also contains all the necessary information to answer the 20 different queries described in the previous chapter. Additionally, the XMark file sizes differed by 5 MB, starting from *xml1* to *xml12*, where *xml13* was generated with a size of 32 MB so that it could be compared with the *lineitem* file which is a wide tree of the same size (32 MB). Similarly, the *xml14* was generated to be compared with the *nasa* file, which represents deeper tree structure than XMark data.

Dataset Name	File Name	File Size (MB)	Experiment
XMark Benchmark	xml1	0.5	Initial labelling (time, size)
	xml2	1	• Initial labelling (time, size) • Determining relationships • Query performance • Handling insertions
	xml3	5	Initial labelling (time, size)
	xml4	10	Initial labelling (time, size)
	xml5	15	Initial labelling (time, size)
	xml6	20	Initial labelling (time, size)
	xml7	25	Initial labelling (time, size)
	xml8	30	Initial labelling (time, size)
	xml9	35	Initial labelling (time, size)
	xml10	40	Initial labelling (time, size)
	xml11	45	Initial labelling (time, size)
	xml12	50	Initial labelling (time, size)
	xml13	32	Initial labelling (time, size)
	xml14	23	Initial labelling (time, size)
Nasa	nasa	23	Initial labelling (time, size)
TPC-H	lineitem	32	Initial labelling (time, size)

Table 7.1: XML Files used in the experiments

7.4 Static Document Experiments

7.4.1 Initial Labelling Experiment

The initial labelling experiment, as explained in Chapter 6, focused on evaluating the initial labelling process in terms of time and size. Thus, the experiment was divided into two sub experiments as follows:

- *Initial Labelling Time:* this measure was the time spent calculating and assigning the labels to each node. The labels were calculated and assigned based on the rules and the implementation methods described in Chapters 4 and 5 respectively.

- *Label Sizes:* this experiment evaluated the growth in the labels' size in terms of memory allocation.

However, these two experiments were carried out using different file sizes (Table 7.1) in order to evaluate how increasing the file's size affected the labels' calculation time and size.

7.4.1.1 Results' Analysis

This section discusses the results of the initial labelling experiments in the GroupBased labelling scheme and in the DDE scheme.

- *Initial Labelling Time*:

  Figure 7.1.1 shows the line chart that represents the time needed to label the whole XML document when using the GroupBased scheme. As shown in the chart, there is an exponential correlation between time and file size where the time increases rapidly when the file's size increases by only 5 MB. The average percentage of this increase is 53%.

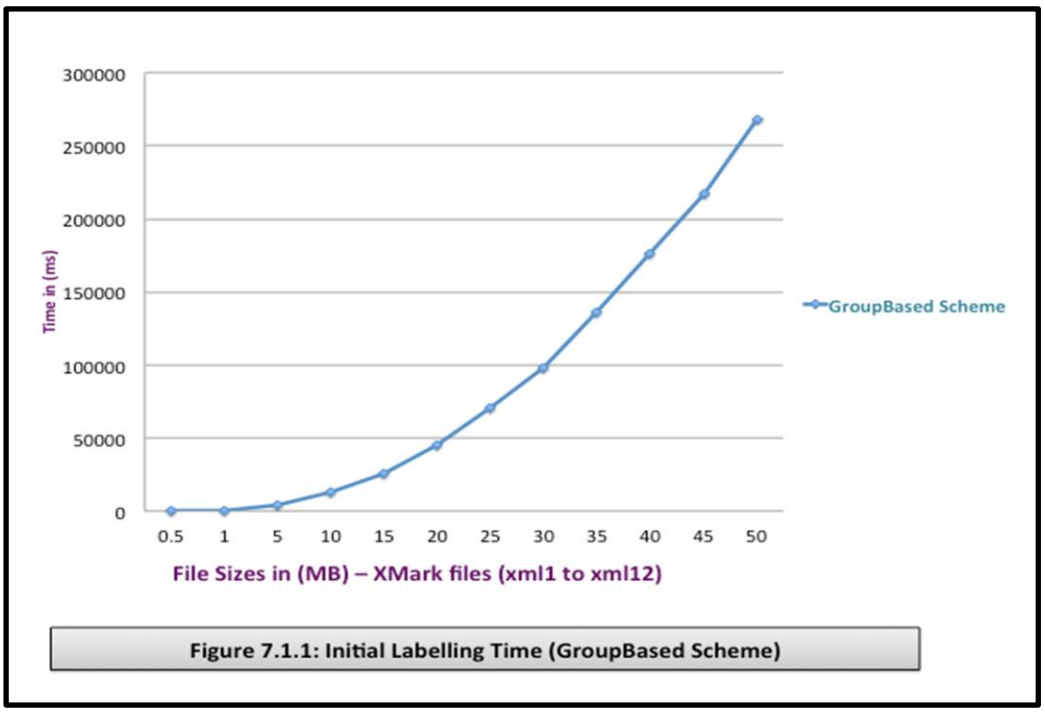


Figure 7.1.1: Initial Labelling Time (GroupBased Scheme)

Performing the same experiment using the DDE scheme gave the same results with regard to the exponential correlation between the file's size and the average time increase, as shown in Figure 7.1.2.

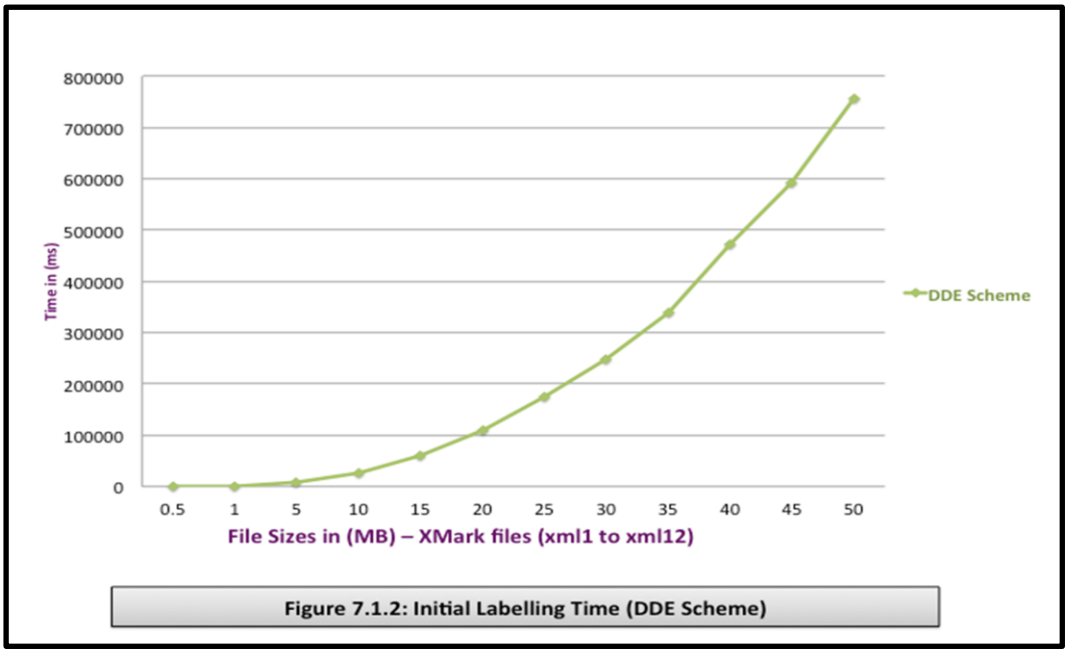


Figure 7.1.2: Initial Labelling Time (DDE Scheme)

Although both schemes show significant time growth, by comparing the results from both schemes (as shown in Figure 7.1.3), the time taken for the initial labelling using the DDE scheme was 40% higher than the initial labelling time in

the GroupBased scheme when the file size was only 0.5 MB; additionally, this time increased by 180% when the file was 50 MB.

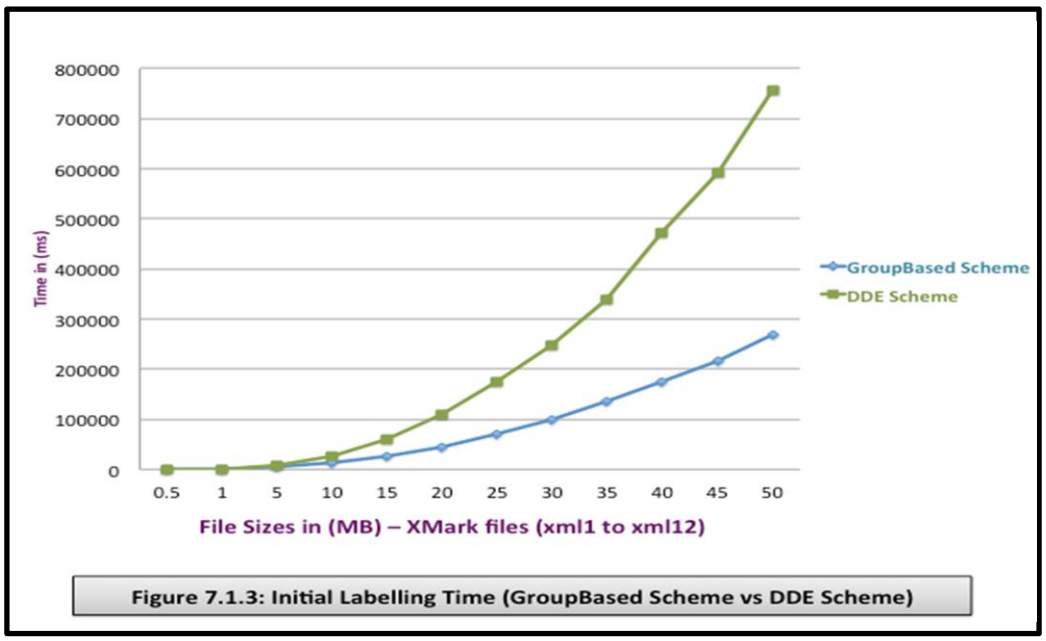


Figure 7.1.3: Initial Labelling Time (GroupBased Scheme vs DDE Scheme)

- *Label Size*:

  Monitoring the growth of the labels' sizes during the initial labelling showed that, in both schemes, starting from a file size of 5 MB and then increasing the size by 5MB increments to 50 MB, each increase in the labels' size was steady at 0.056 MB. This is shown in Figure 7.2.1.

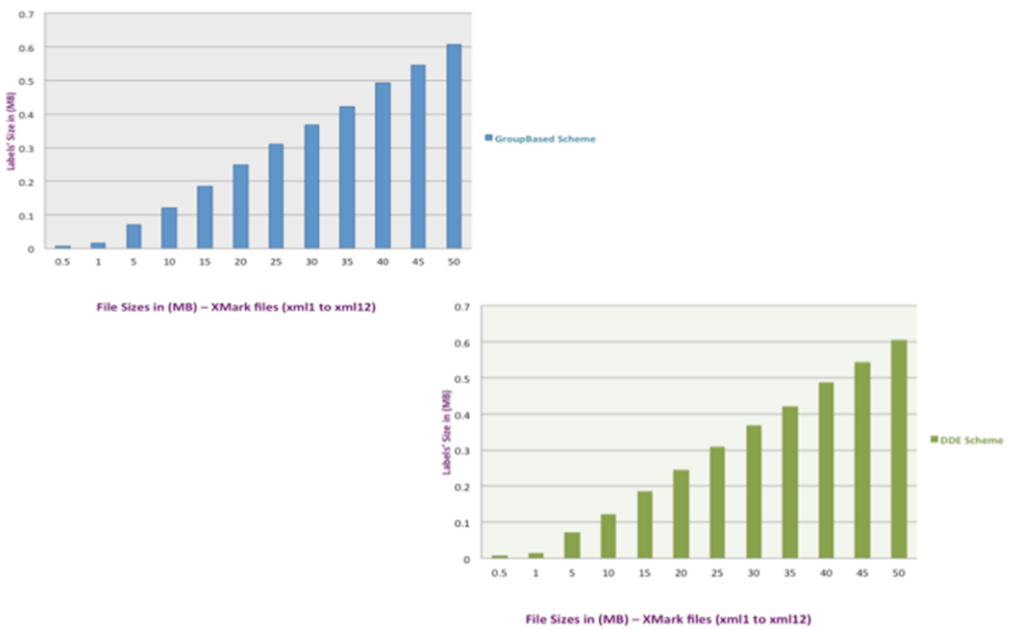


Figure 7.2.1: Initial Labelling Size (GroupBasedScheme & DDE Scheme)

However, the labels' sizes when using the GroupBased scheme were surprisingly only slightly higher (0.002 %) than those using the DDE scheme because the GroupBased scheme's label actually consists of two labels, as illustrated in previous chapters (Ch.4 & Ch.5) while a DDE label consists of one which might lead to an expectation that the difference between the two would be higher. Figure 7.2.2 shows this difference.

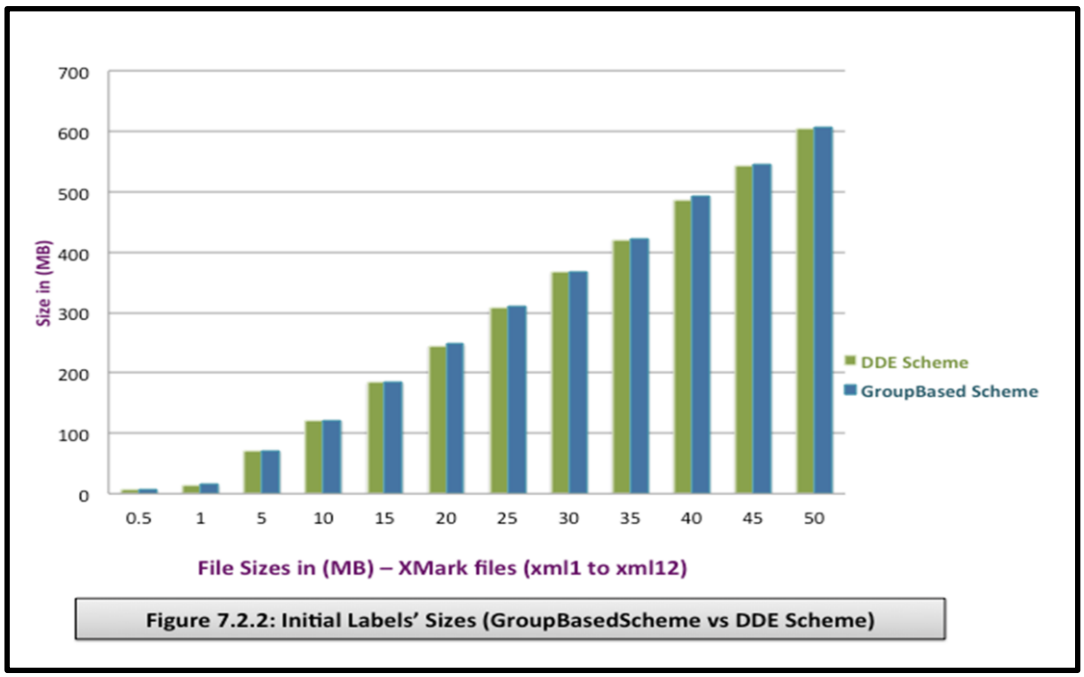


Figure 7.2.2: Initial Labels' Sizes (GroupBasedScheme vs DDE Scheme)

In order to test the GroupBased scheme's ability using a wider tree structure, the same experiment was performed on *'linitem.xml'*, *'xml13.xml'* and '*nasa.xml'* files where the first two were of the same size (Table 7.1). Their tree structures were of wide and medium depth respectively while the '*nasa.xml'* file had more depth than the *'xml13.xml'*, which is an XMark file (Ch.6). The results obtained show that both schemes are more effecient with a deep tree structure in terms of time and label size. Using the GroupBased and DDE schemes with the '*linitem.xm*' file, the initial labelling time was five times higher; both schemes showed a double increase in the initial labelling time in the *'nasa.xml'* file compared to the *'xml14.xml'* file and with a 42% increase in terms of the labels' size. Figures 7.3.1 and 7.3.2 present these results.

Despite the similarity between the results of both schemes, the GroupBased scheme offers better performance in terms of time than the DDE scheme using wide (*linitem.xml)* and deeper tree structures, as the DDE labelling time was 165% higher than with the GroupBased scheme, although the DDE scheme provided slightly more concise labels, as shown in Figures 7.3.3 and 7.3.4.

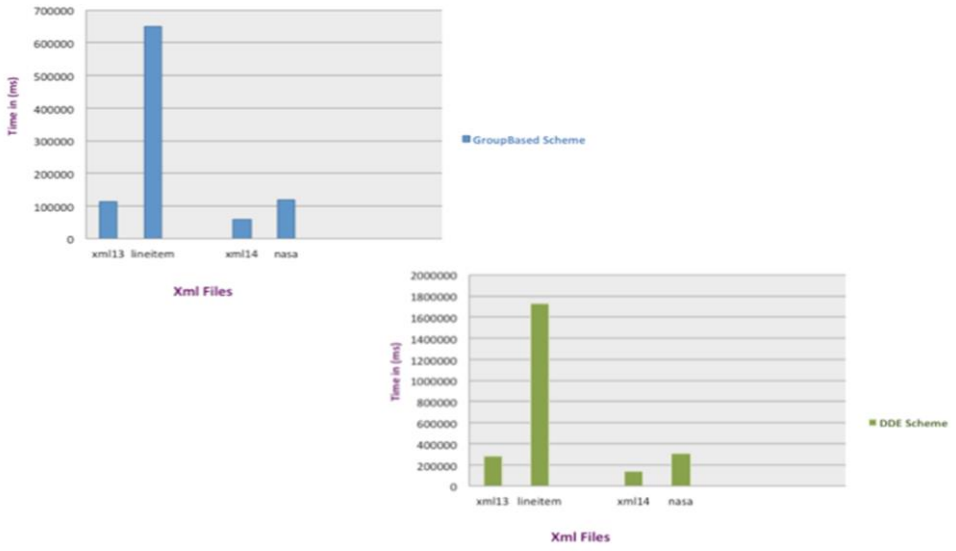


Figure 7.3.1: Initial Labels' Time (wide vs deep) Tree Structure

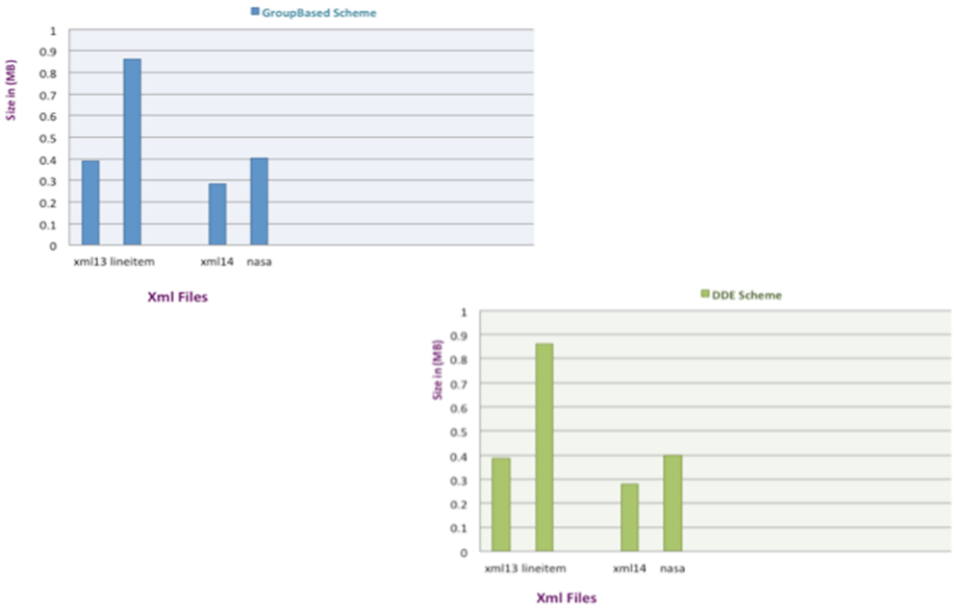


Figure 7.3.2: Initial Labels' Sizes (wide vs deep) Tree Structure

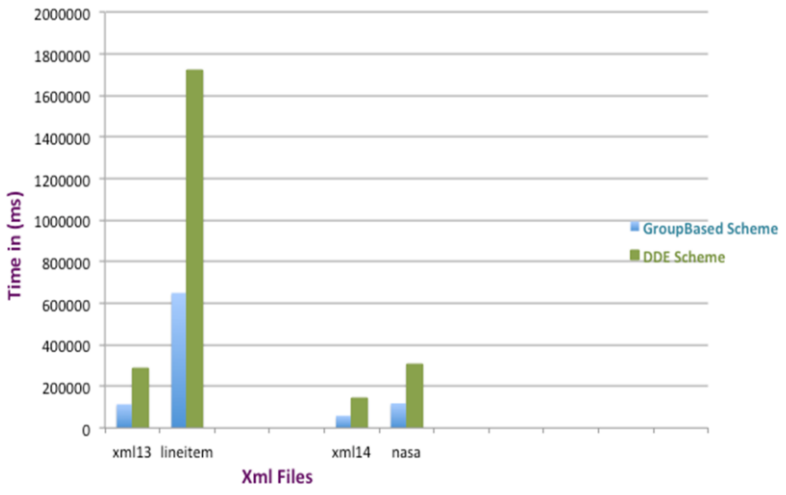


Figure 7.3.3: Initial Labels' Time (wide vs deep) Tree Structure , (GroupBased vs DDE)

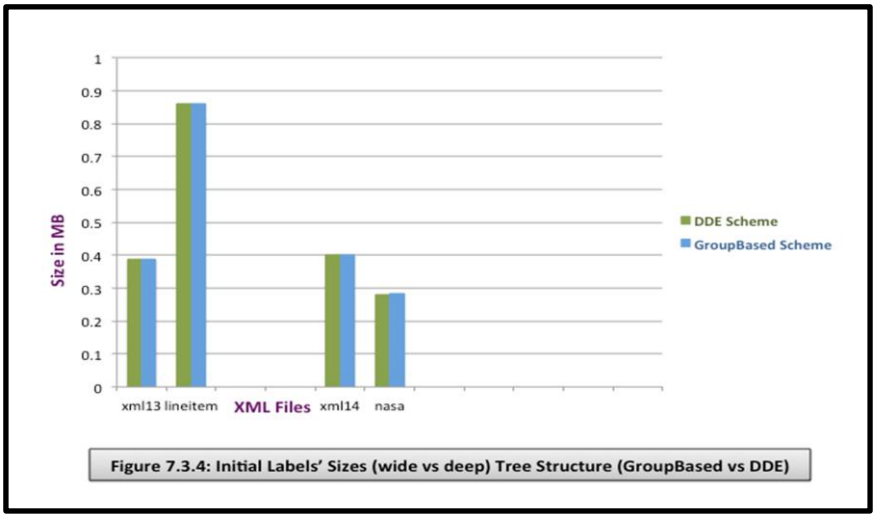


Figure 7.3.4: Initial Labels' Sizes (wide vs deep) Tree Structure (GroupBased vs DDE)

In addition to these findings, a review of literature shows that the GroupBased scheme provides better performance in terms of time in the initial labelling experiments than the *QED-based* labelling schemes (Li and Ling, 2005b), *ORDPATH* (O'Neil *et al.*, 2004) and the *vector-order based* labelling scheme (Xu *et al.*, 2012)(Ch.3) because the published results in Xu *et al.*(2009), Xu *et al.* (2012), and Qin *et al.*(2012) confirm that the DDE scheme is better than these schemes. Moreover, based on the results published in Qin *et al.*(2012) and Liu *et al.*(Liu *et al.*), the GroupBased scheme shows that less labelling time is needed than with the *Dynamic float-point Dewey* scheme (Liu *et al.*, 2013) or the *Dynamic Common Prefix* scheme (Qin *et al.*, 2012) as they both consume either the same initial labelling time or more than the DDE scheme's labelling time.

7.4.1.2 Statistical Interpretation of the Results

Difference in initial labelling time was found between the GroupBased scheme and DDE scheme as a measure of time spent calculating and assigning the labels to each node. Between the two samples, the initial labelling time was measured using populations of document sizes from 0.5 MB to 50MB with each population having a gradual increase in size by 5 MB with the exception of the first and second, which were increased only by 0.5 MB. The dataset *lineitem*, *xml13* and *nasa* were also included in the populations, giving a total of 15 populations for this variable of initial labelling time. Importantly, it was noted that in all 15 populations, the box

plot showed distributions that were shifted towards GroupBased scheme as against DDE scheme, giving the indication that the former provides a better labelling time than the latter as showed in Figure 7.4.1.

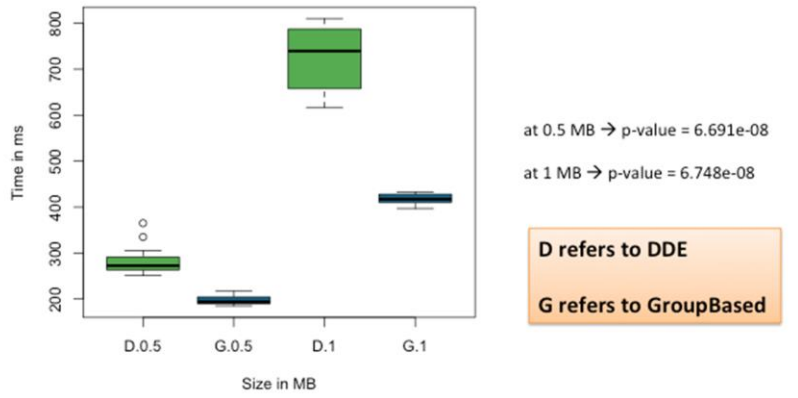


Figure 7.4.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes using XML files of size **0.5 and 1 MB** for the initial labelling experiment

In the figure above, the p-value obtained at a size of 0.5MB was $6.691x10^{-8}$ whilst that obtained for 1MB of document size was $6.748x10^{-8}$. There were 11 other similar results gathered -using different size of documents- with the use of the box plot which have been made available in Appendix a.1.

As far as the p-value which gives significance using the Wilcoxon rank-sum test is concerned, it was noted that in all cases, extremely low values were obtained. The p-value ranged from as low as $6.691x10^{-8}$ to $1.451x10^{-11}$. This means that with this variable, the test supports the alternative hypothesis that GroupBased scheme had an effect on time and that it is faster than compared to the use of DDE scheme. Meanwhile, Bosak and Bray (1999) stated that one important parameter for which XML has been formulated is to ensure that there is efficient initial labelling time. This advantage with initial labelling time is achieved with the GroupBased scheme.

As explained earlier, the dataset *lineitem* and *nasa* were used to test the schemes ability in labelling wide and deep XML trees. These dataset also exhibited very low p-values of 1.451x10-11 each. This means that, the alternative hypothesis for the

GroupBased scheme will also be accepted. Figure 7.5.1, shows the box plots of the datasets *lineitem* and *xml13* whereas the box plots of *nasa* and *xml14* datasets can be found in There were two sets used for the *nasa* and *xml14*, one of which can be found in Appendix a.1

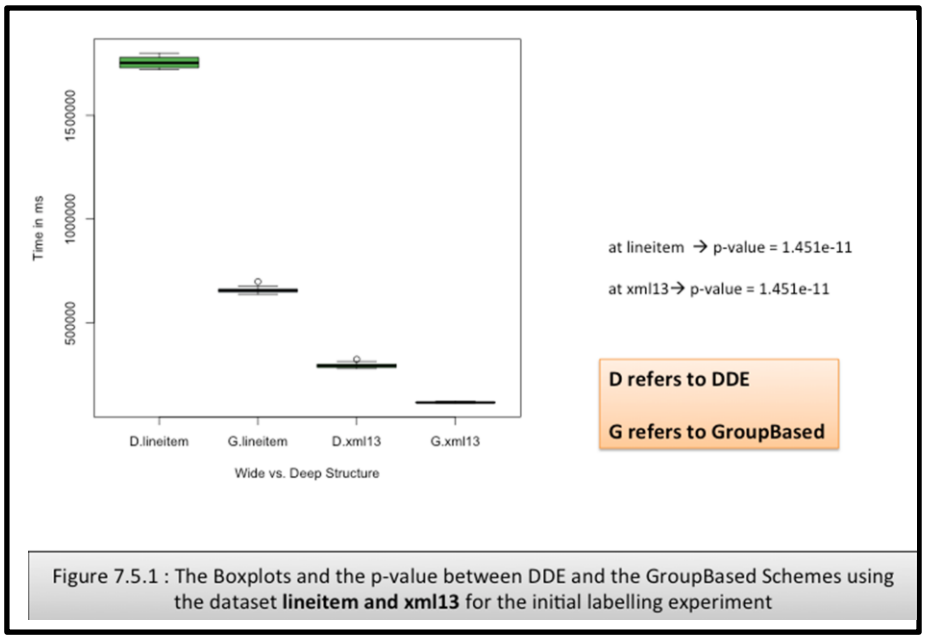


Figure 7.5.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes using the dataset **lineitem and xml13** for the initial labelling experiment

7.4.2 Determining Different Relationships

This experiment measured the time needed to calculate the six relationships between two nodes using their labels (Ch.4 & Ch.6). These relationships are:

- *Order:* this determines which node is first in the XML document.
- *Level:* this determines the node's level within the xml tree where the document root level is one.
- *Ancestor/Descendant (AD), Parent/Child (PC) and Lowest Common Ancestor (LCA):* these relationships are determined based on the rules defined in Chapter 4 where the parent/child and the lowest common ancestor are established based on the AD relationship.
- *Sibling:* this determines whether two nodes share the same parent node.

7.4.2.1 Results' Analysis

Figure 7.6.1 shows the time spent when determining the relationships using the GroupBased scheme's labels. Calculating the node's level represents the minimum time consumption, where PC and LCA relationships represent the higher consumption, equaling a 20% time increase in the level calculation. The second smallest time is represented by the order determination with only a 4% time increase over the level calculation. The results for the AD and the sibling relationships are placed in the middle with increases of 7% and 9% more than the level calculation.

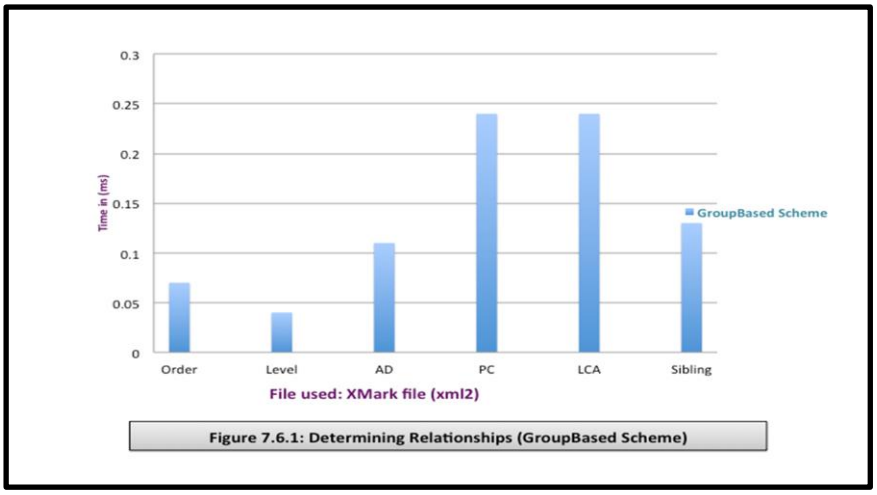


Figure 7.6.1: Determining Relationships (GroupBased Scheme)

Using DDE labels, computing the level is also faster while the longest time is represented by the LCA calculation which required 43% more time, as shown in Figure 7.6.2.

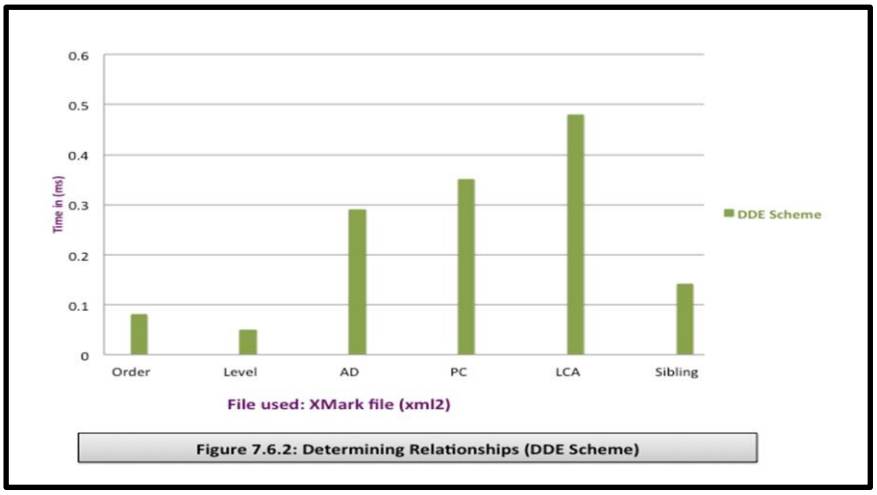


Figure 7.6.2: Determining Relationships (DDE Scheme)

Figure 7.6.3 compares the results of both schemes and from this, it emerged that, determining different relationships using DDE labels gave quite similar results to the GroupBased scheme labels in terms of level, order and sibling calculations with only 1.5% increases. However, AD, PC and LCA showed a higher time consumption of 18%.

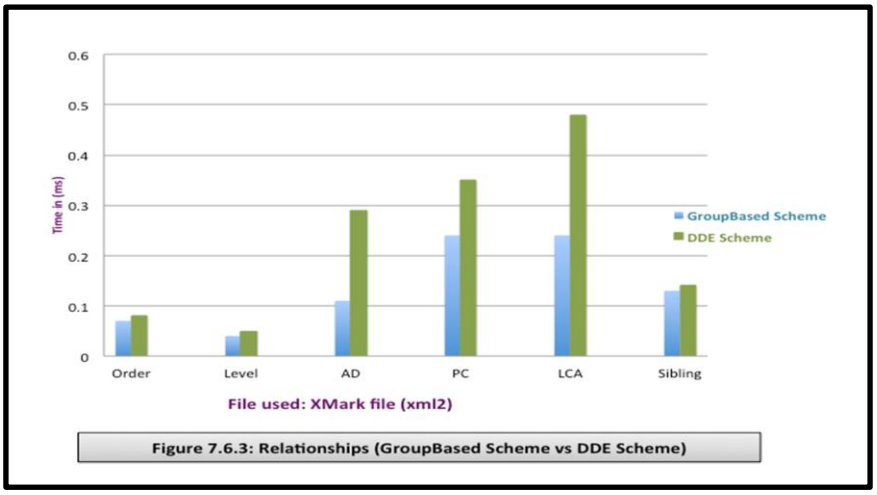


Figure 7.6.3: Relationships (GroupBased Scheme vs DDE Scheme)

As with the initial labelling experiment, the GroupBased scheme shows better performance in determining different relationships when the document is static compared to the DDE scheme. Thus, this finding can be extended to state that the GroupBased scheme determines relationships faster than *QED-based* labelling schemes (Li and Ling, 2005b) and the *ORDPATH* scheme (O'Neil *et al.*, 2004), as mentioned in Xu *et al.,* (2009), Xu *et al.,* (2012) and Liu *et al.*,(2013).

7.4.2.2 Statistical Interpretation of the Results

Different relationships in static XML were also examined to the time needed to calculate six different relationships between two given nodes using their labels. The relationships have already been outlined and explained in detail under 7.4.2. From the statistical techniques, six populations tested under this variable. These were order between nodes, nodes level, AD relationship, PC relationship, LCA relationship, and sibling relationships. Using outcomes from the box plot, it was seen that in all six populations, there was a shift in distribution towards the GroupBased scheme as against the DDE scheme. The indication that this gives in a

holistic perspective is that GroupBased sceheme provides faster calculation of the six relationships than the DDE scheme. To discover how significant the difference in time of calculation was, the p-value was found as the outcome of Wilcoxon rank-sum test for each population. Six box plots were created for this purpose, one of which has been shown in Figure 7.7.1 below. The remaining box plot results have been given in Appendix a.2

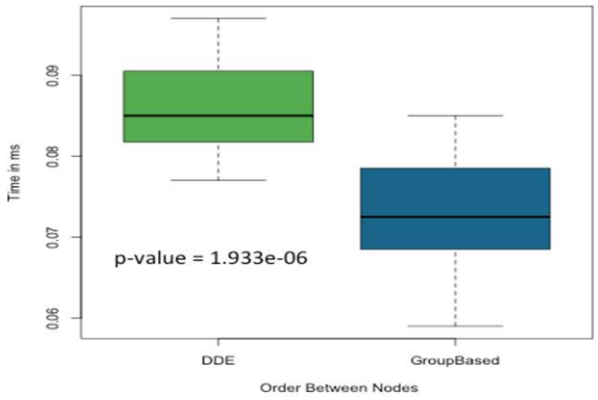


Figure 7.7.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying the **Order** between two nodes in a static XML document

7.4.3 Query Performance

As explained in the previous chapter, this experiment evaluates the query response time on the labelled XML document using 19 different queries that vary in their complexity and objectivity (Ch.6: Sec. 6.6). As mentioned earlier, all queries were evaluated on *'xml2.xml'* file an XMark file of 1 MB (Table 7.1).

7.4.3.1 Results' Analysis

Figures 7.8.1 (a, b, c and d) show the response times of the tested queries when using the GroupBased scheme labels. Q13, the full text search, shows the shortest time among all the queries at 7 milliseconds. However, Q12, which evaluates the scheme's ability to extract and reconstruct a portion of the original XML document, represents the longest response time at 895 milliseconds; this may be because of the nested nature of the XML document. The second longest time is for Q8, which evaluates a complex case of chasing references by traversing the XML tree

horizontally; the response time of this query is 57% less than Q12. The ordered access queries (Q2, Q3 and Q4) and values' joining queries (Q10 and Q11), which evaluate the scheme's ability to handle large intermediate results where join operations are performed on the basis of values, consume about 108-228 milliseconds but they require multiple join, an operation which can be expected to be slow. The response times of the rest of the queries were within (10-122) milliseconds.

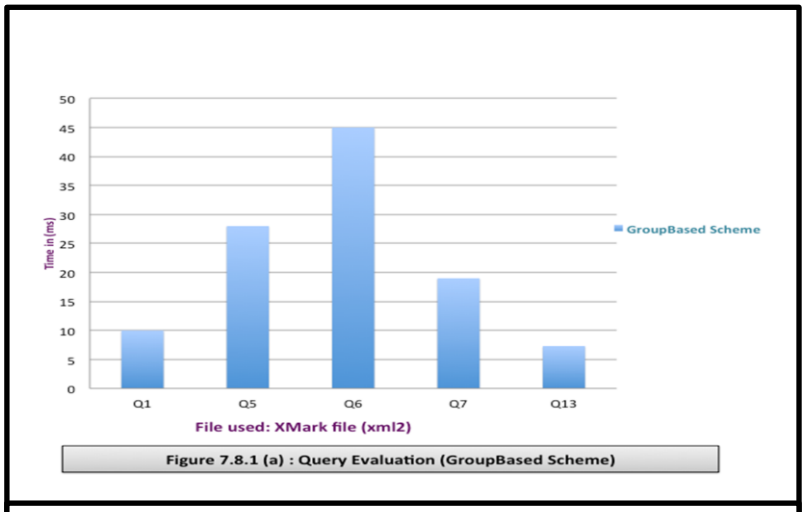


Figure 7.8.1 (a) : Query Evaluation (GroupBased Scheme)

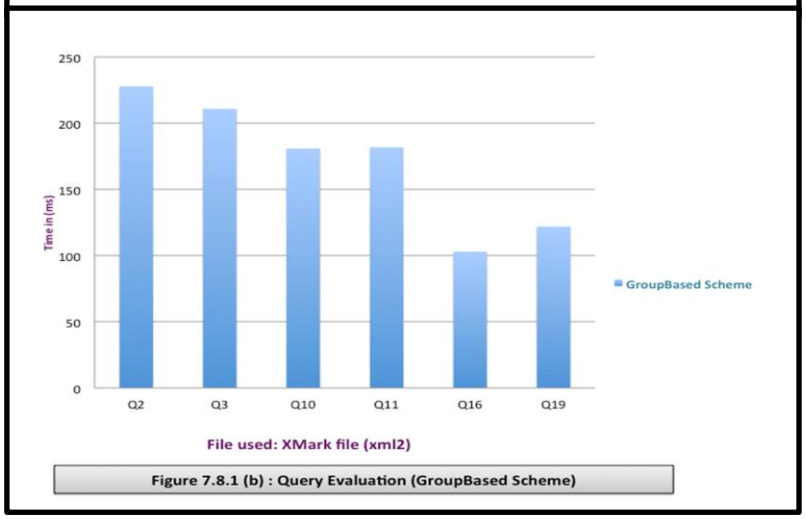


Figure 7.8.1 (b) : Query Evaluation (GroupBased Scheme)

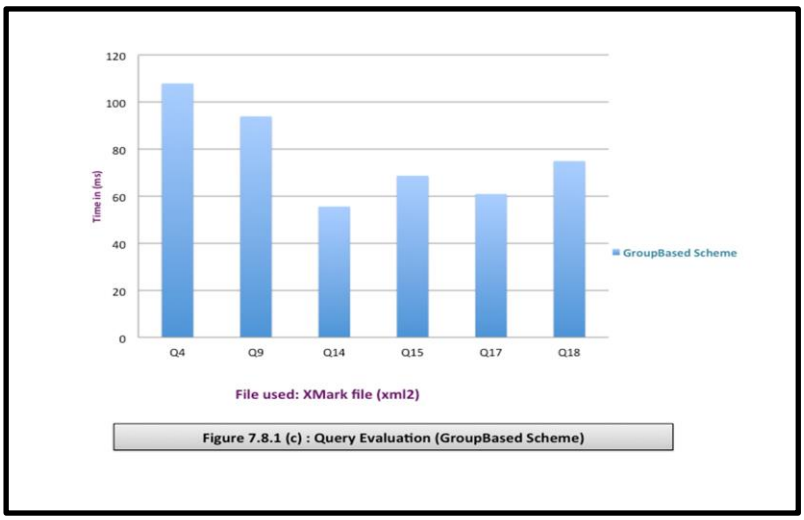


Figure 7.8.1 (c) : Query Evaluation (GroupBased Scheme)

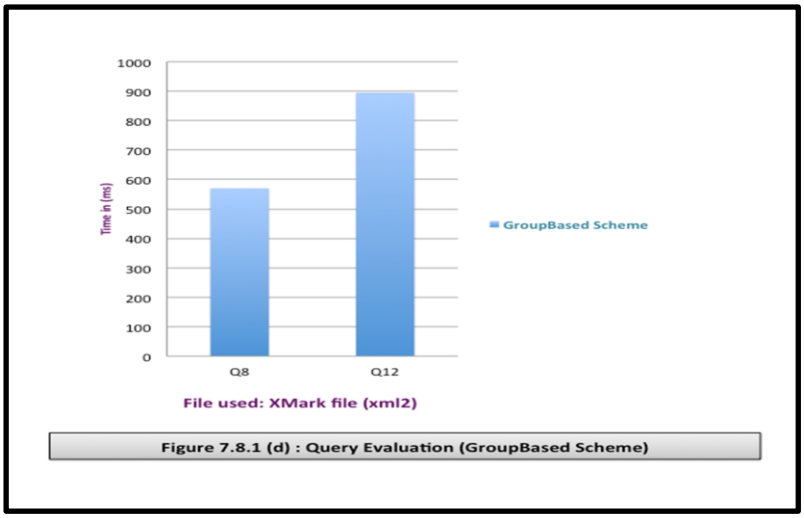


Figure 7.8.1 (d) : Query Evaluation (GroupBased Scheme)

The results of evaluating the same queries using DDE labels are shown below in Figure 7.8.2 (a, b and c).

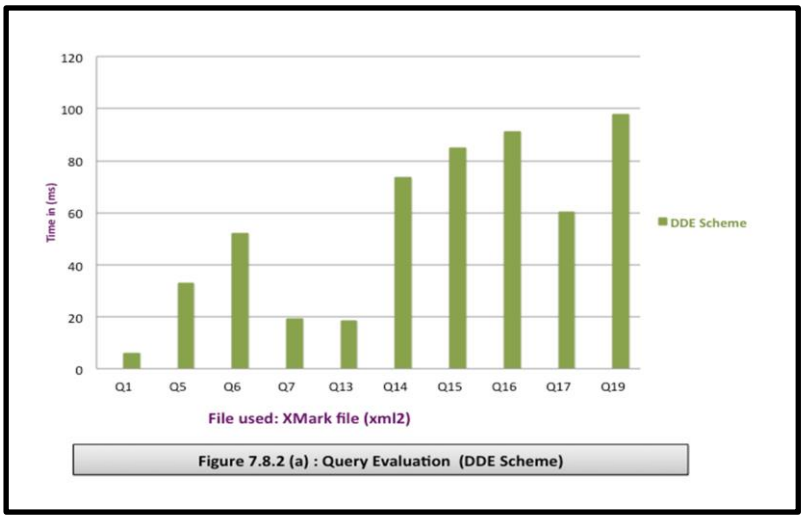


Figure 7.8.2 (a) : Query Evaluation (DDE Scheme)

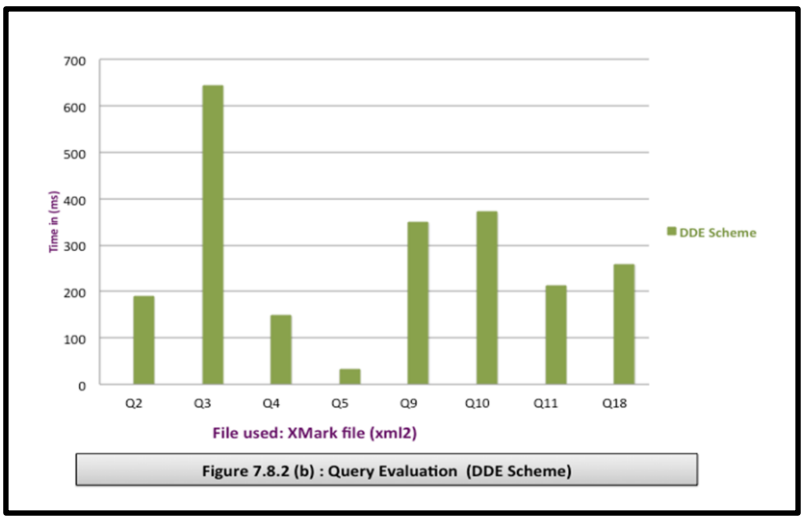


Figure 7.8.2 (b) : Query Evaluation (DDE Scheme)

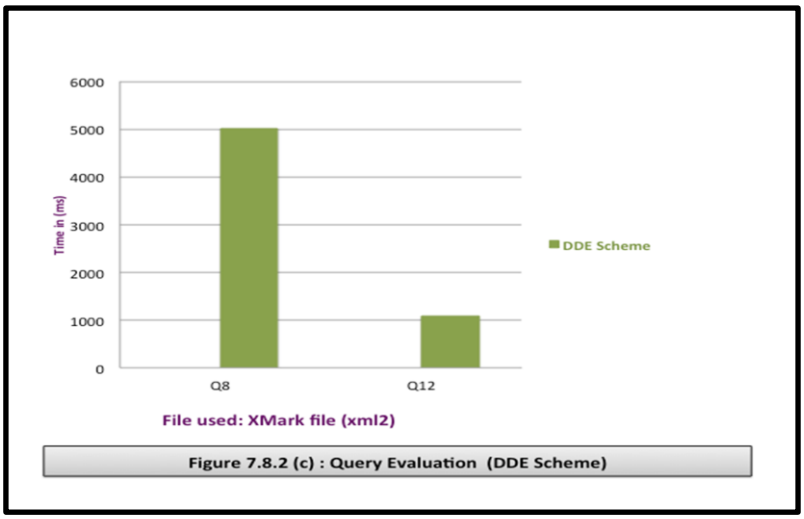


Figure 7.8.2 (c) : Query Evaluation (DDE Scheme)

Similar to the GroupBased scheme's queries' evaluation but the other way around, Q8 and Q12 achieved the longest time with (5024) and (1079) milliseconds respectively. The third longest time was for Q3 with 67% less than Q12 while the execution time of the other queries was within (6-371) milliseconds.

However, a comparison of the queries' performance of both schemes shows that the GroupBased scheme offered a better response time in thirteen queries out of nineteen. Q7 showed equal performance in both schemes and the other five queries (Q1, Q2, Q16, Q17 and Q19) showed better performance using the DDE scheme as shown in Figures 7.8.3 (a, b and c).

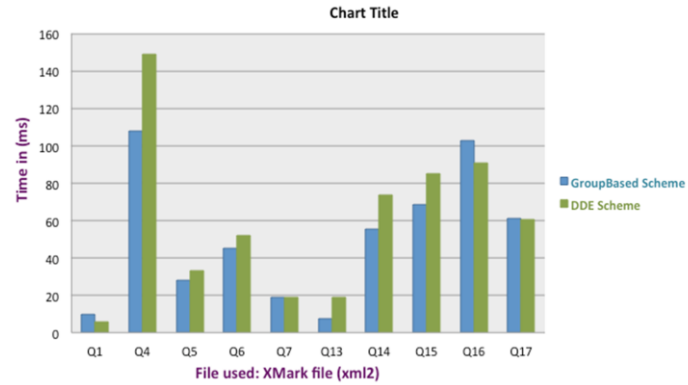


Figure 7.8.3 (a) : Query Evaluation (GroupBased Scheme vs DDE Scheme)

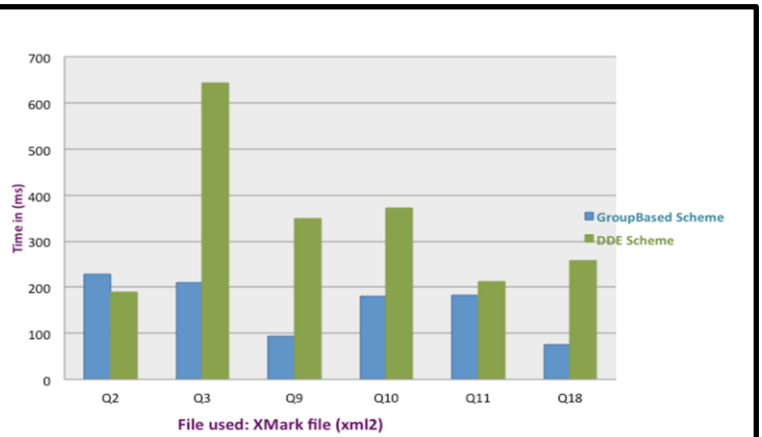


Figure 7.8.3 (b) : Query Evaluation (GroupBased Scheme vs DDE Scheme)

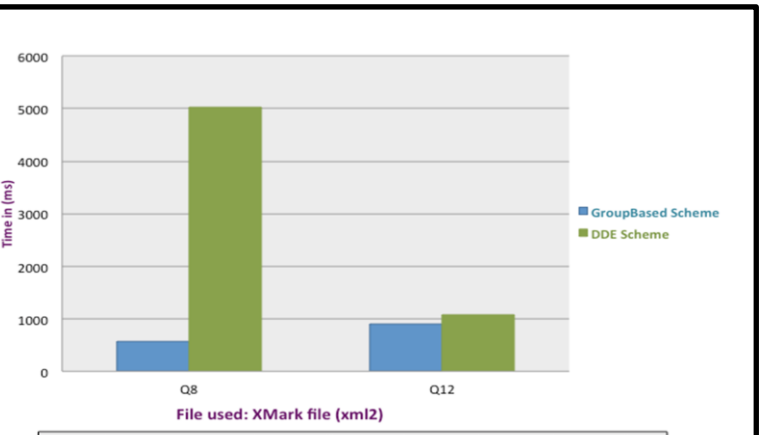


Figure 7.8.3 (c): Query Evaluation (GroupBased Scheme vs DDE Scheme)

7.4.3.2 Statistical Interpretation of the Results

As stated earlier, nineteen different queries were employed to evaluate the query response time on the labelled XML document that vary in complexity and objectivity. For each of these, box plot readings were made, the outcome of which has been produced as Appendix a.3. In the literature, this query performance is important to establish the relationship between performance and time, as some have argued that efficiency with time always compromises performance (Frigge *et al.*, 2009). The box plots and Wilcoxon rank-sum tests were therefore used to discover if the GroupBased scheme or DDE scheme could overcome this limitation. Using the nineteen queries which have already been explained in 7.4.3.1, very different behaviours with queries were seen between the GroupBased scheme and DDE scheme. First, using the box plots, it was seen that in queries 1, 2, 16, and 19, the DDE scheme clearly provides better performance than the GroupBased scheme.

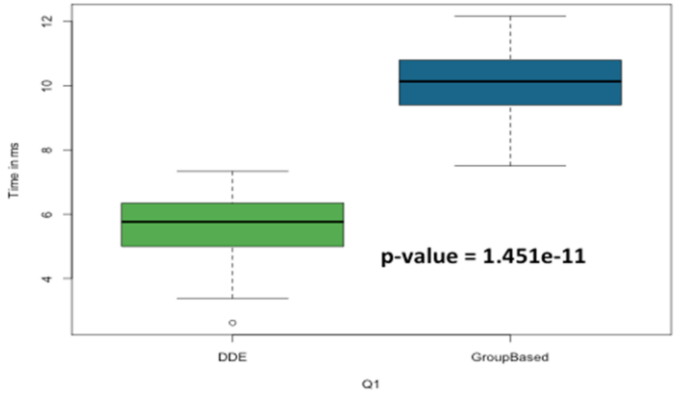


Figure 7.9.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q1)**

As Figure 7.9.1 shows with query 1, the Wilcoxon rank-sum test was used to test for significance. In this a p-values of $1.451x10^{-11}$ was recorded in all cases. This means that the named queries, were faster.

When it came to query 7, a very interesting line of results were obtained. This is because even though the box plots showed that the DDE gave better performance, the Wilcoxon rank-sum test showed that this did not amount to a significant difference as the p-value was 0.4777 when the significance level was 0.05. For all the remaining queries namely 3, 4, 5, 6, 8-15, 17, and 18, both the box plots and Wilcoxon rank-sum test favoured the GroupBased scheme, meaning that there was better performance. This line of data confirms why Bosak and Bray (1999) admonished thorough consideration with the selection of schemes when dealing with questions on static XML documents. This is because as shown in this study, different queries could show different behaviour on performance between different schemes.

7.5 Dynamic Document Experiments

As mentioned in the previous chapters, handling XML insertions without re-labelling the existing labels is one of the most important features provided by the GroupBased scheme. This section presents the results and analyses of experiments that evaluated the scheme's ability to handle different types of insertion, as well as its performance after the insertions. These experiments are categorised as:

- Handling insertions
- Determining different relationships
- Query performance

7.5.1 Handling Insertions

The handling insertions experiment evaluates the time and the label sizes when performing different types of insertion: namely, uniform insertions, ordered skewed insertions and random skewed insertions.

7.5.1.1.1 Uniform Insertions

The uniform insertions refer to the insertion of a new node between two consecutive nodes. The time spent in executing the insertions and calculating the new labels is measured, as well as the labels' sizes after the insertions, which are then compared to the sizes before the insertions. This experiment was run on 12 XMark files as in the initial labelling experiment. Figure 7.10.1 shows the results of performing the uniform insertions on different sizes of XML file using the GroupBased scheme. This experiment shows that the execution time of insertions increased when the file's size increased; this is because the number of insertions increased when the file size increased.

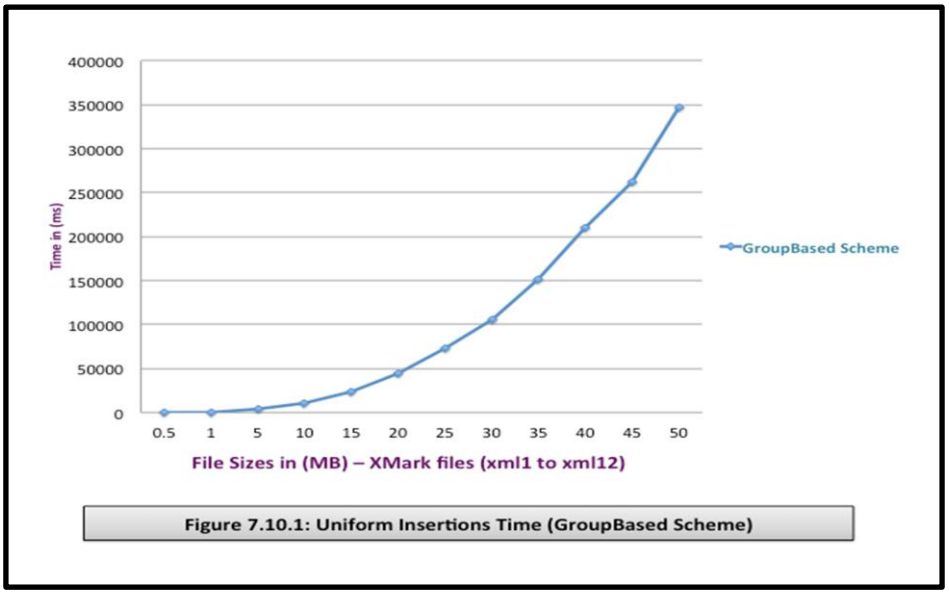


Figure 7.10.1: Uniform Insertions Time (GroupBased Scheme)

The uniform insertions were also performed using the DDE scheme with very similar results. Nevertheless, using the GroupBased scheme led to a slightly faster execution time (faster by 6%), as shown in Figures 7.10.2 (a, b, c and d).

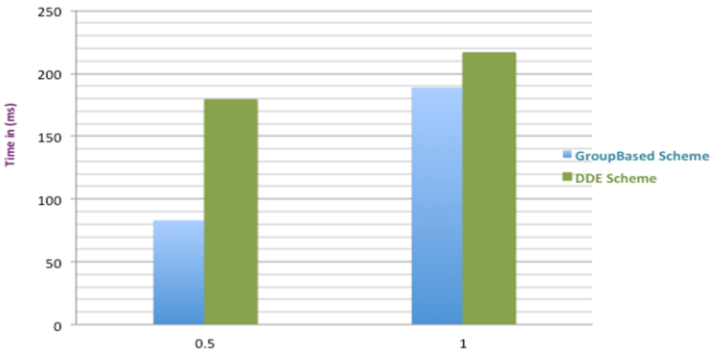


Figure 7.10.2 (a) : Uniform Insertions Time (GroupBased Scheme vs DDE Scheme)

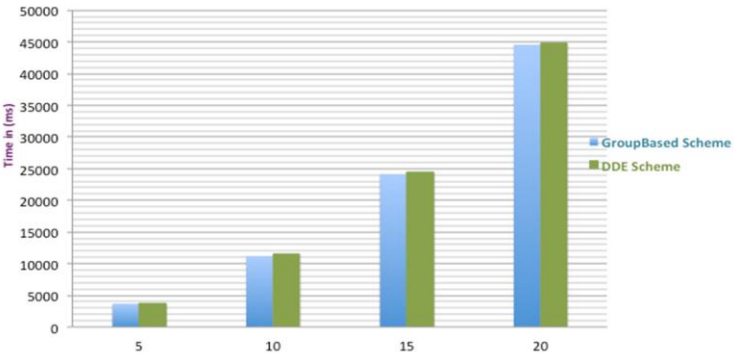


Figure 7.10.2 (b) : Uniform Insertions Time (GroupBased Scheme vs DDE Scheme)

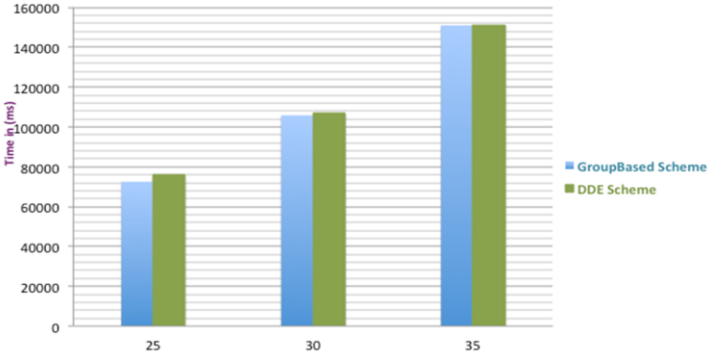


Figure 7.10.2 (c) : Uniform Insertions Time (GroupBased Scheme vs DDE Scheme)

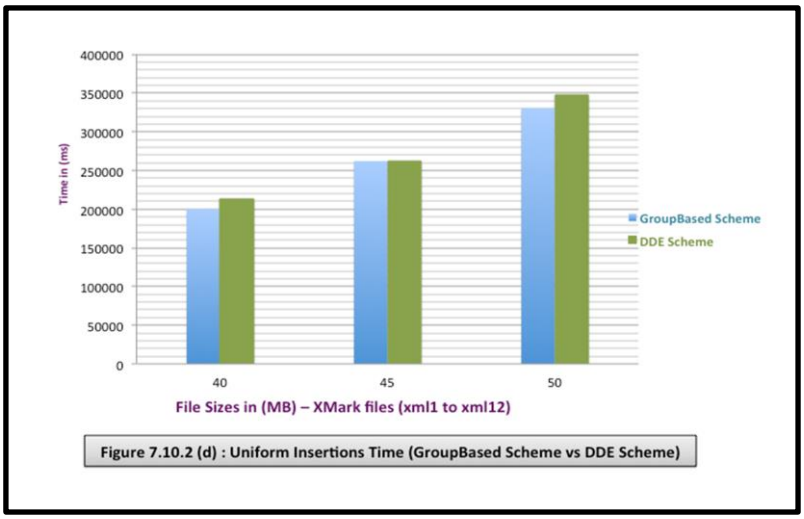


Figure 7.10.2 (d) : Uniform Insertions Time (GroupBased Scheme vs DDE Scheme)

The size of the labels after performing the uniform insertions was almost identical using both schemes, with a negligible improvement of 0.03% when using the DDE scheme. This is shown in Figure 7.10.3.

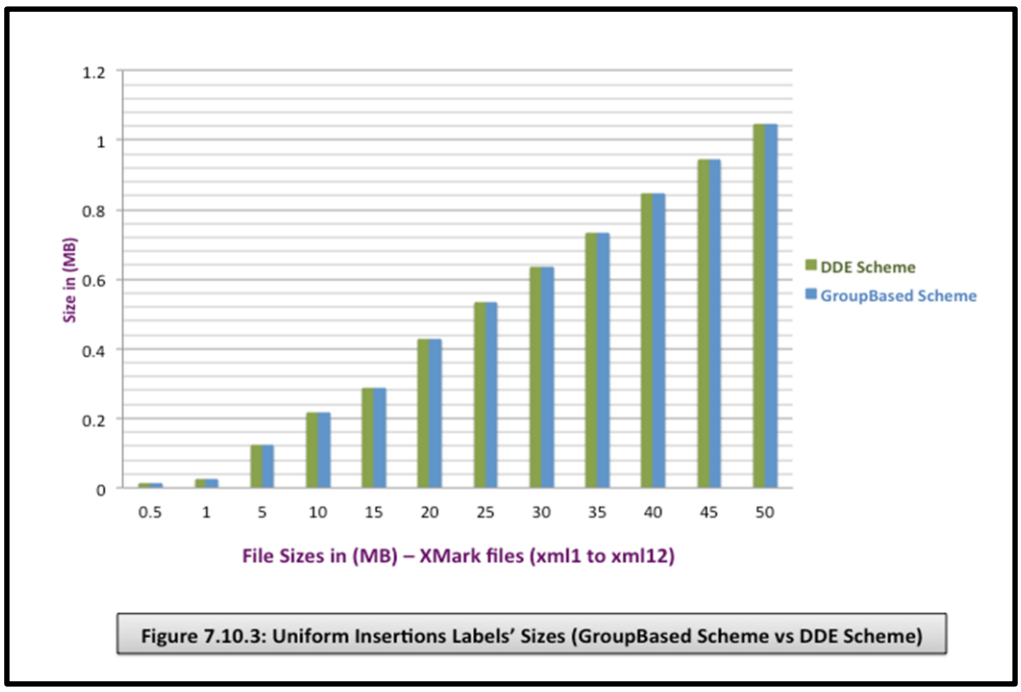


Figure 7.10.3: Uniform Insertions Labels' Sizes (GroupBased Scheme vs DDE Scheme)

7.5.1.1.2 Statistical Interpretation of the Results

When working on XML documents, 'Uniform-insertion' means the insertion of new nodes between two consecutive nodes. Because the consecutive nodes in-between which the insertion is made may have their own characteristics, some experts have argued that uniform insertions could introduce efficiency challenges (McGill *et al.*, 1978). It was for this reason that the GroupBased scheme and DDE were both

tested for their effect on time when performing uniform insertions. In this case, the uniform insertion was performed based on size of file whereby XML documents with sizes from 0.5 MB to 50 MB were used as populations. There were a total of twelve populations for each sample because after the 0.5MB, the next file size was 1MB, and then 5MB before a steady increase of 5MB was performed for each subsequent XML document till 50MB was reached. Per the box plots produced, it is seen that in all cases between the GroupBased scheme and DDE scheme, labelling was faster in the former than the latter.

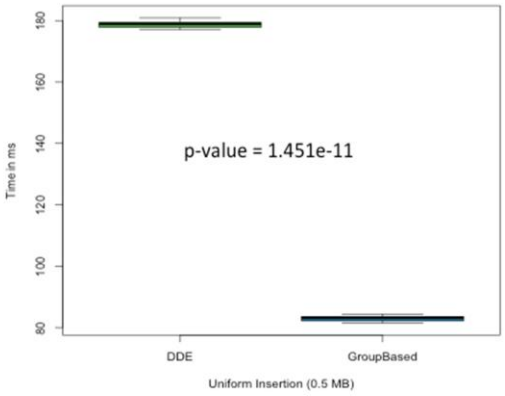


Figure 7.11.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Uniform Insertion using XML file of size 0.5 MB**

This made it necessary to find the statistical significance of the difference in performance. With the null hypothesis that scheme does not have any effect on time, the p-value measured in all cases were far lower than 0.05. As showed with the uniform insertion at 0.5MB in Figure 7.11.1, the range of p-value recorded were 6.771×10^{-8} to 1.451×10^{-11}. This shows that the alternative hypothesis will be accepted that difference in labelling performance between GroupBased scheme and DDE scheme has a direct impact on time. The remaining outcomes for the box plots have been displayed in Appendix a.4 where the p-value and performance for 12 other figures are given.

7.5.1.2.1 Ordered Skewed Insertions

The Order Skewed of insertion refers to the process of inserting before or after a particular node repeatedly. An *'xml2.xml'* file was used in this experiment but the change factor was the number of insertions. The results, which are presented in Figure 7.12.1, show that the insertions' execution time when using the DDE scheme was faster, and with a non-steady increase, than when using the GroupBased scheme.

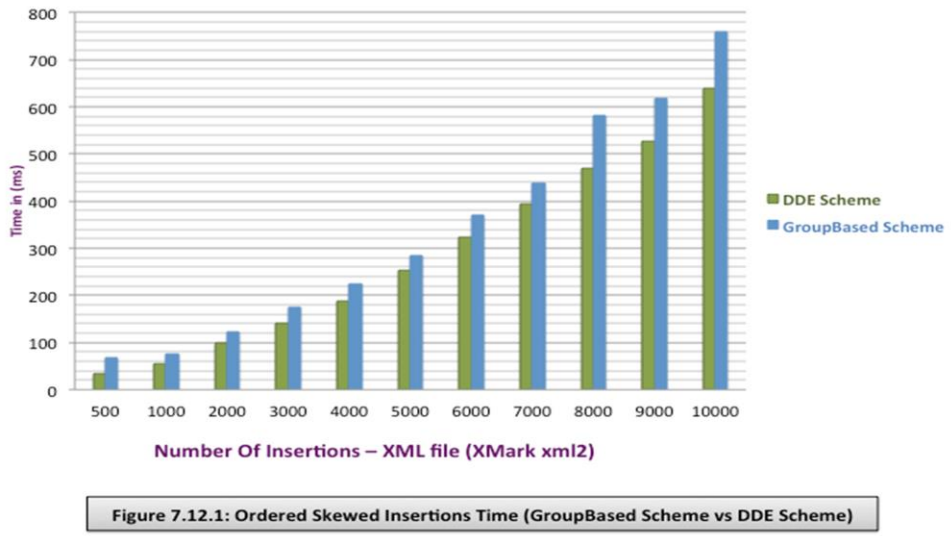


Figure 7.12.1: Ordered Skewed Insertions Time (GroupBased Scheme vs DDE Scheme)

Similar to the labels' size after uniform insertions, both schemes showed almost identical label sizes, with only 0.2% more concise labels when using the DDE scheme, as shown in Figure 7.12.2.

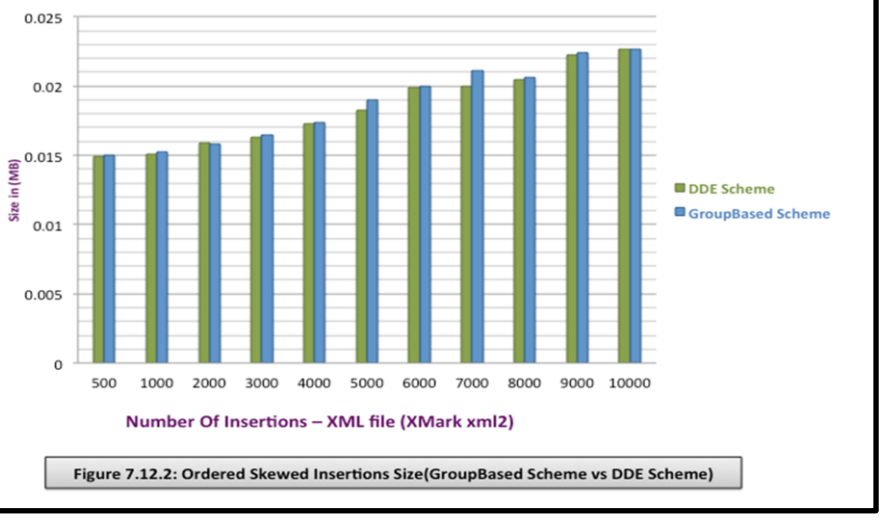


Figure 7.12.2: Ordered Skewed Insertions Size(GroupBased Scheme vs DDE Scheme)

7.5.1.2.2 Statistical Interpretation of the Results

As has been outlined in 7.5.1.2.1, ordered skewed insertion is performed when there is an insertion made before or after a particular node in a repeated manner. Because the insertion made before or after an existing node is done in a repeated manner, the emphasis with the comparison between GroupBased scheme and DDE was done based on the increases made in the number of nodes added. The initial node was 500, after which this was increased to 1000 nodes. Thereafter, there was systematic increase of 1000 nodes till 10000 nodes were reached. This means that the number of populations used in finding the Wilcoxon rank-sum test were eleven in each case. The outcome with these eleven tests has been showed as box plots in Appendix a.5.

The plot box depicted a very strong advantage with the use of DDE scheme over the GroupBased scheme in all eleven populations because the distributions were shifted in favour of the DDE scheme as seen in Figure 7.13.1 for ordered skewed insertion at 500 nodes.

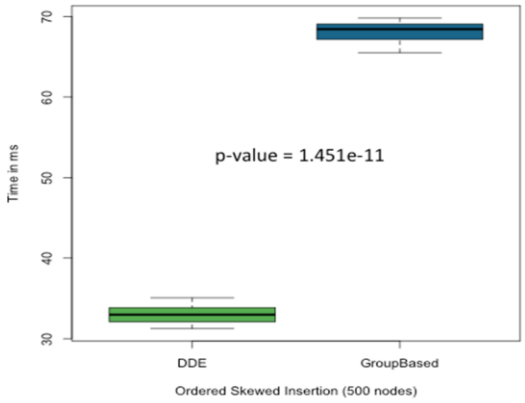


Figure 7.13.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Ordered Skewed Insertion (500 nodes were inserted)**

When Wilcoxon rank-sum test was performed to see if the performance was significance, it was found that the null hypothesis was rejected in all cases. This is because for all eleven populations, the same p-value was produced between the

GroupBased scheme and DDE scheme. As in Figure 7.13.1, the p-value produced was 1.451×10^{-11}, which is far less than the significance level of 0.05. Because of this the alternative hypothesis was accepted that the difference was significant and that it had effect on time.

7.5.1.3.1 Random Skewed Insertions

The Random Skewed of insertion refers to randomly inserting between two nodes; '*xml2.xml*' was used in this experiment. Using the GroupBased scheme, this type of insertion was performed in 176 milliseconds on average and with a 22% average increase, while the average execution time when using the DDE scheme was 188 milliseconds with a 19% increase rate. Comparing both schemes' results shows that, with this type of insertion, the GroupBased scheme was 6% faster than the DDE scheme. This is shown in Figure 7.14.1.

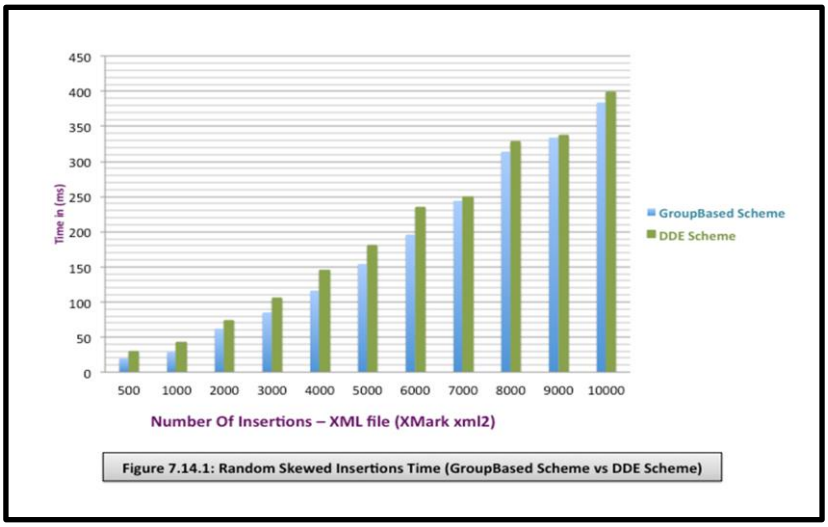


Figure 7.14.1: Random Skewed Insertions Time (GroupBased Scheme vs DDE Scheme)

Regarding the labels' sizes, both schemes offered very similar results with only 0.06% better performance when using the DDE scheme, as shown in Figure 7.14.2.

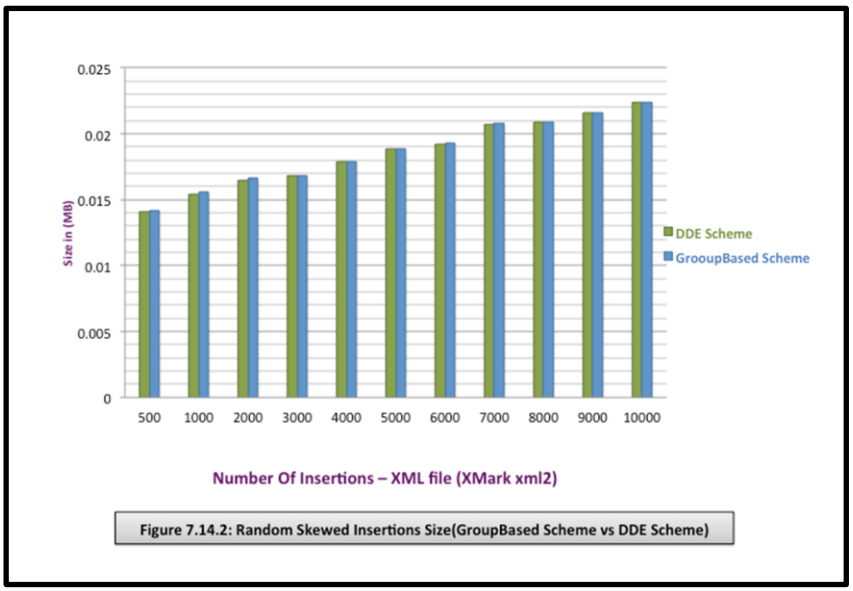


Figure 7.14.2: Random Skewed Insertions Size(GroupBased Scheme vs DDE Scheme)

7.5.1.3.2 Statistical Interpretation of the Results

The third type of insertion performed was the random skewed insertions. As the name implies, this type of insertions was performed randomly between two nodes. Bosak and Bray (1999) had argued that due to the random nature of the insertions, their effect on performance and time are hardly felt with the introduction of new schemes. This was the rationale that informed the testing with GroupBased scheme and DDE scheme. As with the ordered skewed insertion, the populations used were based on number of nodes instead of file size. This gave rise to eleven populations just as was done with the ordered skewed insertion. The outcomes of all these eleven populations have been given in Appendix a.6.

From the box plots that were performed, it was seen that the labelling performance distribution was shifted towards the GroupBased scheme as seen in Figure 7.15.1.

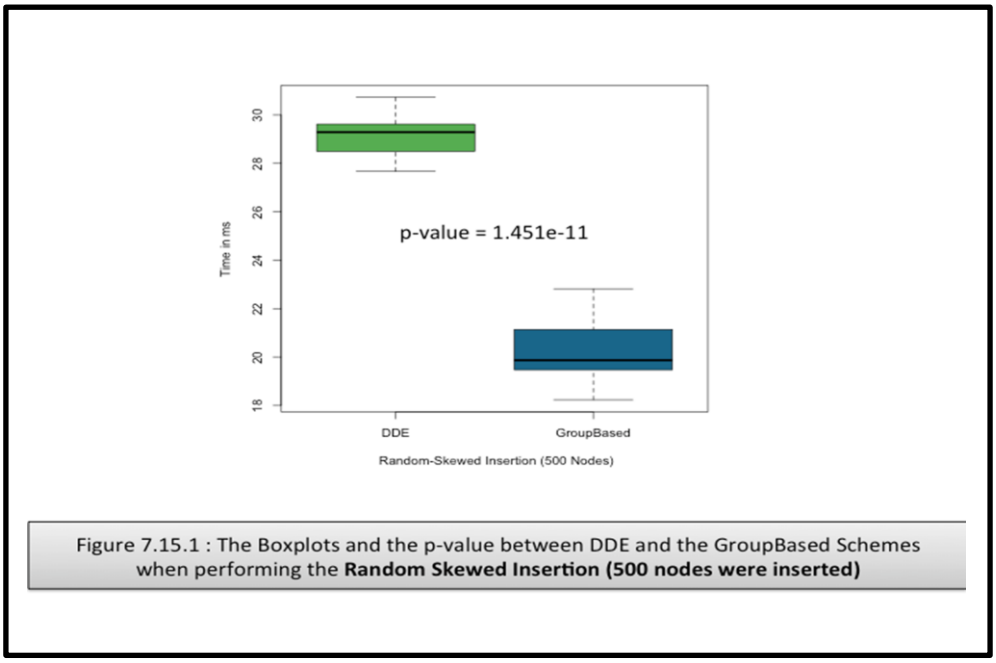


Figure 7.15.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Random Skewed Insertion (500 nodes were inserted)**

The shift towards the GroupBased scheme indicates that the GroupBased scheme provides better labelling performance than the DDE scheme. When the Wilcoxon rank-sum test was performed, the labelled performance differences were noted to be significant at 0.05. This is because with the exception of the 9000 nodes which had a p-value of 5.804×10^{-11}, all the others, including the random-skewed insertion at 500 nodes as in the Figure 7.15.1 above produced p-value of 1.451×10^{11}. In either case however, the values produced showed that the alternative hypothesis was true as the p-value was far less than its significant point. This line of result shows that unlike ordered-insertion, the GroupBased scheme has the potential to improve performance with random skewed insertion.

7.5.2 Determining Different Relationships

As mentioned in Section 7.4.2, this experiment evaluated the time spent in determining different relationships but the experiment here was run after the insertions had been made (Sec. 7.4.2).

7.5.2.1 Results' Analysis

Figures 7.16.1, 7.16.2 and 7.16.3 present the results from this experiment after each type of insertion. Determining the order and the level took the same time in both schemes after the three types of insertion with (0.08 ms) and (0.05 ms) respectively. The GroupBased scheme shows the same calculation time for AD, LCA and sibling relationships after the uniform and ordered-skewed insertions, and 53%, 57%, 172% and 1% more time for the PC, LCA, AD and sibling relationships after the random-skewed insertions; this is in fact less than 0.20 millisecond for them all. On the other hand, the DDE scheme shows the same calculation for the AD, LCA and sibling relationships after all types of insertion while the PC relationship calculation time was the same after the uniform and the random-skewed insertions and with 3 times more time after the ordered-skewed insertions. However, after all types of insertion the GroupBased scheme was 86%, 40%, 18% and 10% faster than the DDE scheme in AD, PC, LCA and sibling calculations respectively.

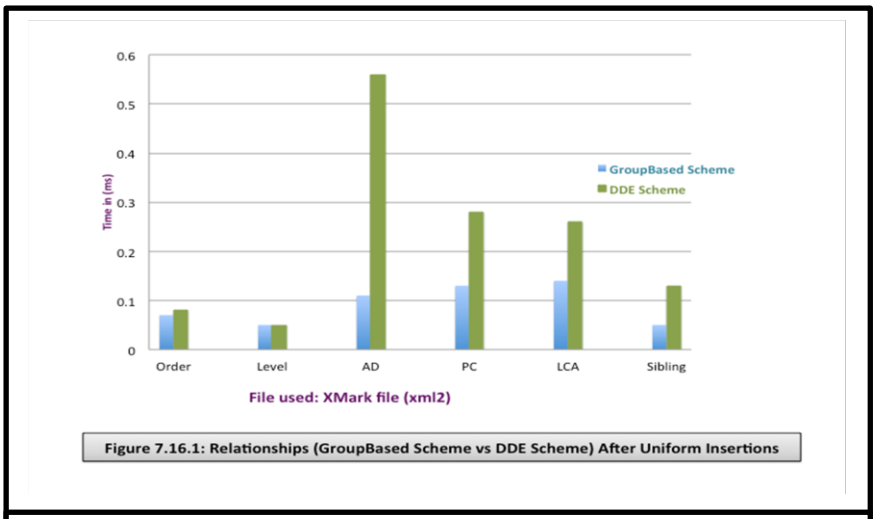


Figure 7.16.1: Relationships (GroupBased Scheme vs DDE Scheme) After Uniform Insertions

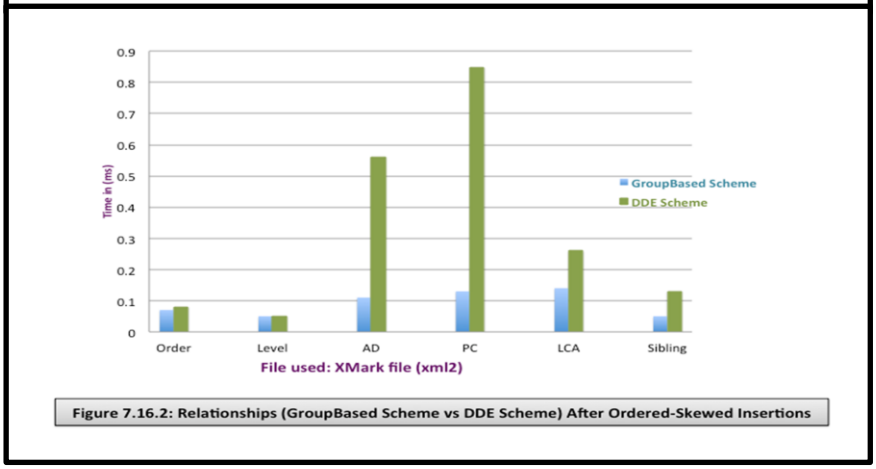


Figure 7.16.2: Relationships (GroupBased Scheme vs DDE Scheme) After Ordered-Skewed Insertions

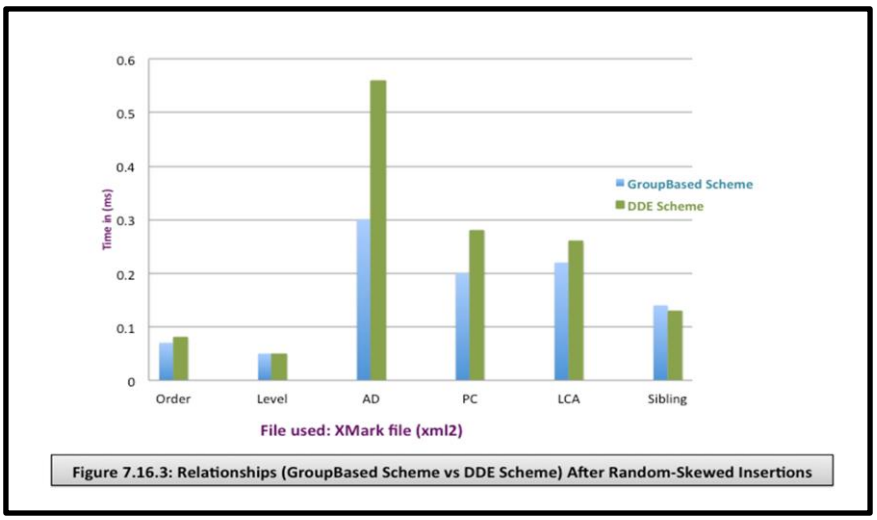


Figure 7.16.3: Relationships (GroupBased Scheme vs DDE Scheme) After Random-Skewed Insertions

Since the GroupBased scheme shows better performance than the DDE scheme in handling insertions and in determining different relationships when the document is dynamic, it is fair to state that it also provides better performance than *QED-based* labelling schemes (Li and Ling, 2005a) and the *ORDPATH* scheme (O'Neil *et al.*, 2004), as mentioned in Xu *et al.* (2009), Xu *et al.*(2012); and Liu *et al.* (2013).

7.5.2.2 Statistical Interpretation of the Results

7.5.2.2.1 Different relationships after Uniform insertion

Based on the insertions that were performed, the different relationships that were established were also tested for their time related performance. The first focused on different relationships after the uniform insertion. Here, the population used for each sample was six, based on the six relations already outlined in Section 7.4.2. The outcome of the experiment has been displayed with the use of box plots for all six samples given in Appendix a.7. As can be seen in Figure 7.17.1 for order between nodes after the uniform insertion, in all six samples, the time spent in determining different relationships with DDE scheme was better than that of GroupBased scheme.

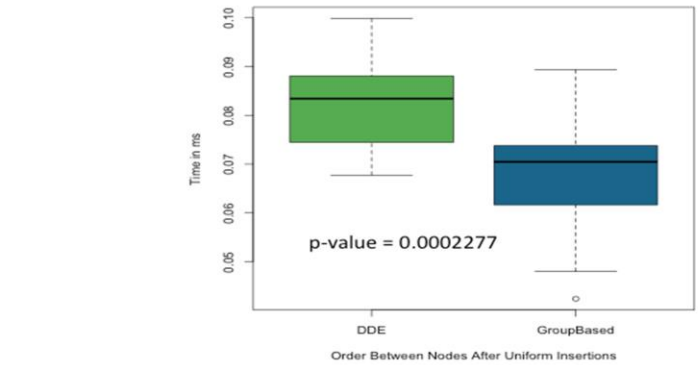


Figure 7.17.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying the **Order** between two nodes after the uniform insertion

Based on the outcome explained above the Wilcoxon rank-sum test which was used to determine the significance of the performance. It was here that interesting outcomes were manifested. This is because at significance level of 0.05, most of the relationship performances were showed to be significant but this was not the case with all relationships. For example, the p-value for nodes level after uniform insertions was 0.3408, which showed that there was weak evidence against the null hypothesis, meaning that the performance did not have effect on time. Apart from this, all the others including the one in Figure 7.17.1 showed p-values that were far less than 0.05 with the closest to that value being 0.0002277 recorded in the order between nodes after uniform insertions. This means that with the exception of nodes after uniform insertions, the performance difference was significant in all other relationships.

7.5.2.2.2 Different relationships after Ordered insertion

After the ordered insertions had also been performed, the relationships that exist with order between nodes after the skewed insertions, nodes after ordered skewed insertion, AD relationship, PC relationship, LCA relationship, and sibling relationship were all tested for performance efficiency in the GroupBased scheme as against the DDE scheme. The total number of samples under this experiment were therefore six. The results from each of these have been given in the form of

box plots in Appendix a.8. Using the box plots, it was found that in this aspect, the distributions were shifted towards the GroupBased scheme.

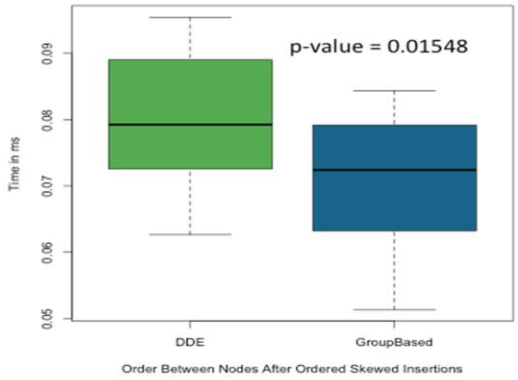


Figure 7.18.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying the **Order** between two nodes after ordered skewed insertion

The shift towards the GroupBased scheme as in Figure 7.18.1 implies that the GroupBased scheme was faster than that of the DDE scheme. When it came to the Wilcoxon rank-sum test, the outcome with results was very similar to what was determined in the uniform skewed insertion where the DDE scheme was noted to be faster. This is because there was one and the same population, which were nodes levels after ordered skewed insertions that there was weak evidence to reject the null hypothesis. This is because the p-value recorded for this was 0.6783, which was far higher than the significance level of 0.05. All the other populations such as the one in Figure 7.18.1 gave strong evidence to reject the null hypothesis and thus justified the significance of the relationship. This is because the p-value for these was 0.01548, which was less than the significance level.

7.5.2.2.3 Different relationships after Random insertion

A similar experiment was performed for random skewed insertion to find the different relationships as had been done with the uniform and random. This means that there were six populations for each of the samples as had been the case before. Using the box plots, it was found that the GroupBased scheme was faster in

all six populations as against the DDE scheme. The outcome for these six populations has been given in Appendix a.9. Figure 7.19.1 however gives the results for the order skewed nodes after random skewed insertion.

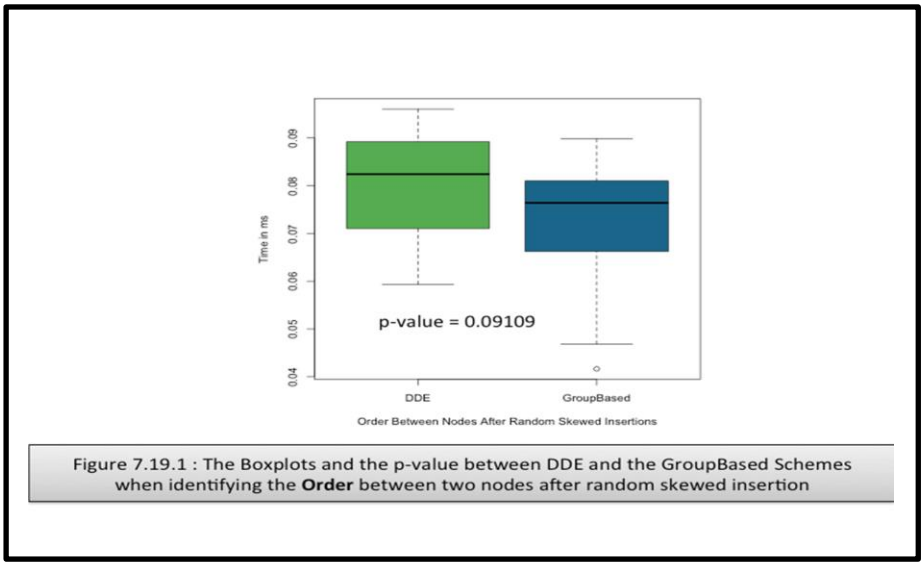


Figure 7.19.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying the **Order** between two nodes after random skewed insertion

In terms of significance however, the case was somewhat different. This is because with significance level of 0.05, it was found that two of the six populations did not show strong evidence to reject the null hypothesis that the scheme did not have effect on time. These two were order between nodes after random skewed insertions and nodes after random skewed insertions (Appendix a.9). The p-value recorded in the two cases was 0.09109 and 0.9042 respectively, including the one showed in Figure 7.19.1. The implication here is that for these populations, GroupBased scheme may be selected for performance related advantages but when it comes to time related advantages, either GroupBased scheme or DDE scheme could be used to achieve the same purpose. With the four other relationships, there was strong evidence to reject the null hypothesis because the p-values were far below the significance level. The range of p-value for the four was 2.898×10^{-05} to1.451×10^{-11}.

7.5.3 Query Performance

In this section, the nineteen queries that were used to evaluate the query performance in the static document (Sec. 7.4.3) were run again after the insertions.

7.5.3.1 Results' Analysis

Using the GroupBased and DDE schemes, the queries' performance on dynamic XML documents showed a significant increase in the queries' response time compared to the results using static documents. As mentioned in Section 7.4.3, the DDE scheme offers a shorter execution time for (Q1, Q2, Q16, Q17 and Q19) when the document was static but, when the document was dynamic, the performance of the GroupBased scheme was better for dynamic documents. Thus, all the queries tested showed better performance using the GroupBased scheme when the document was dynamic. Figures 7.20.1 (a, b and c), 7.20.2 (a, b and c) and 7.20.3 (a and b) present the evaluations of the queries.

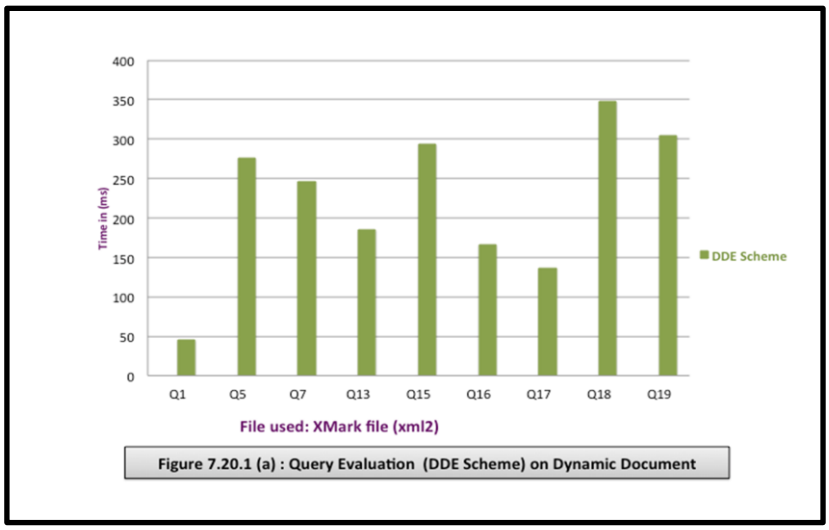


Figure 7.20.1 (a) : Query Evaluation (DDE Scheme) on Dynamic Document

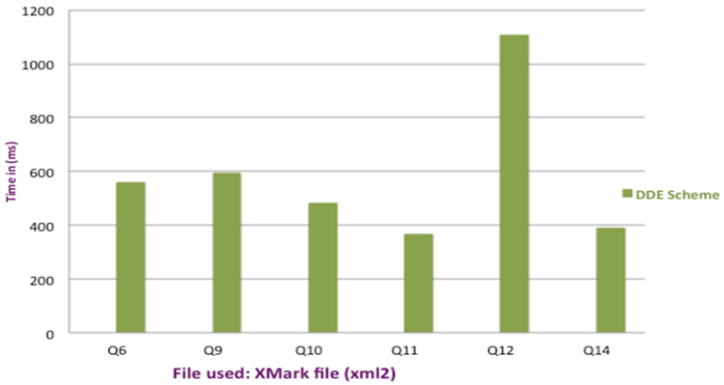


Figure 7.20.1 (b) : Query Evaluation (DDE Scheme) on Dynamic Document

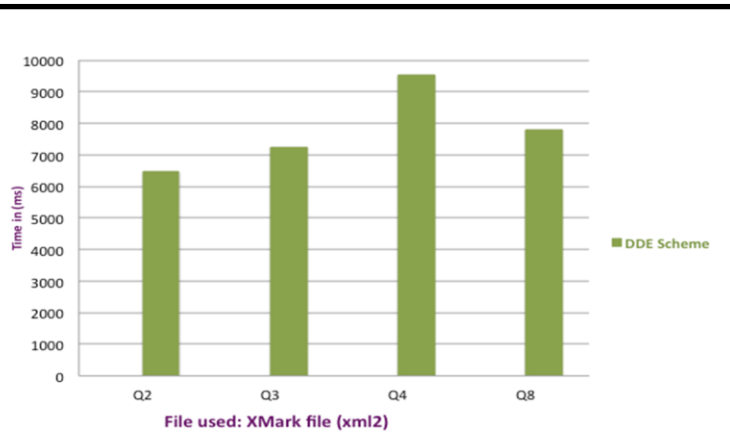


Figure 7.20.1 (c) : Query Evaluation (DDE Scheme) on Dynamic Document

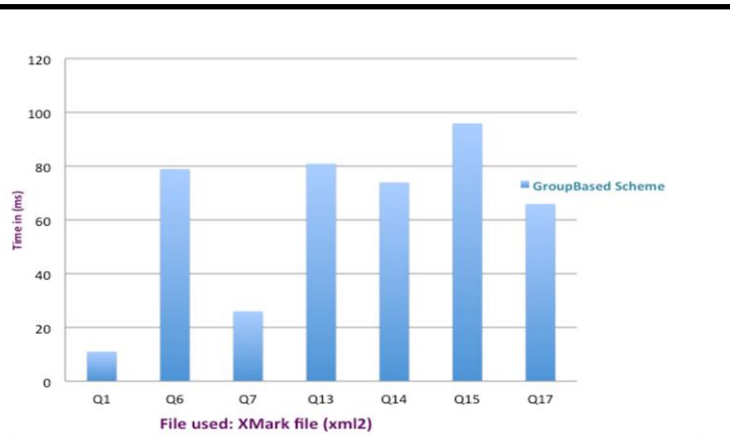


Figure 7.20.2 (a) : Query Evaluation (GroupBased Scheme) on Dynamic Document

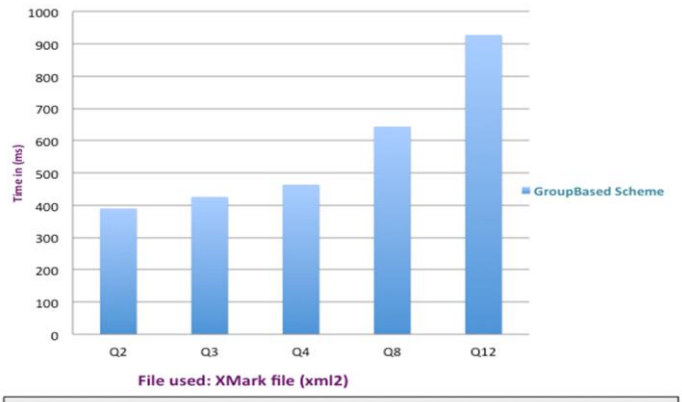


Figure 7.20.2 (c) : Query Evaluation (GroupBased Scheme) on Dynamic Document

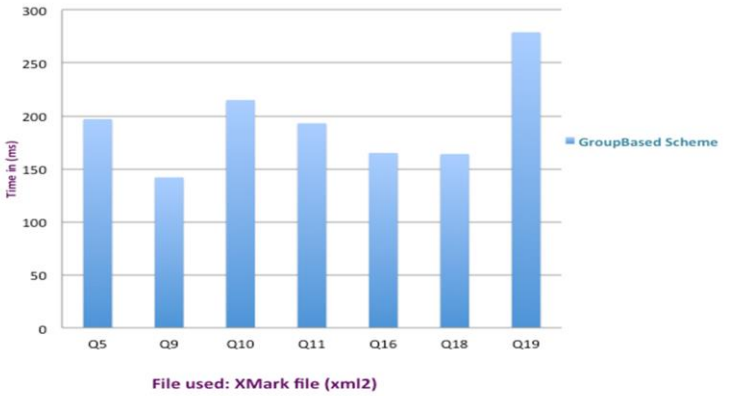


Figure 7.20.2 (b) : Query Evaluation (GroupBased Scheme) on Dynamic Document

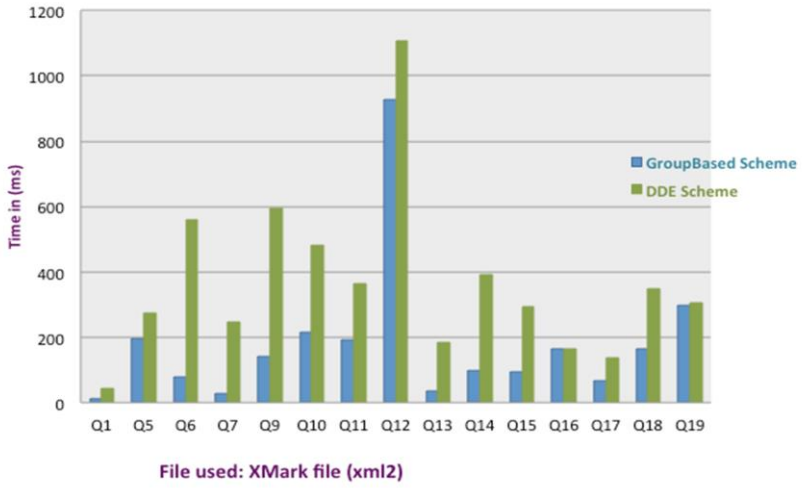


Figure 7.20.3 (a) : Query Evaluation (GroupBased Scheme vs DDE Scheme) on Dynamic Document

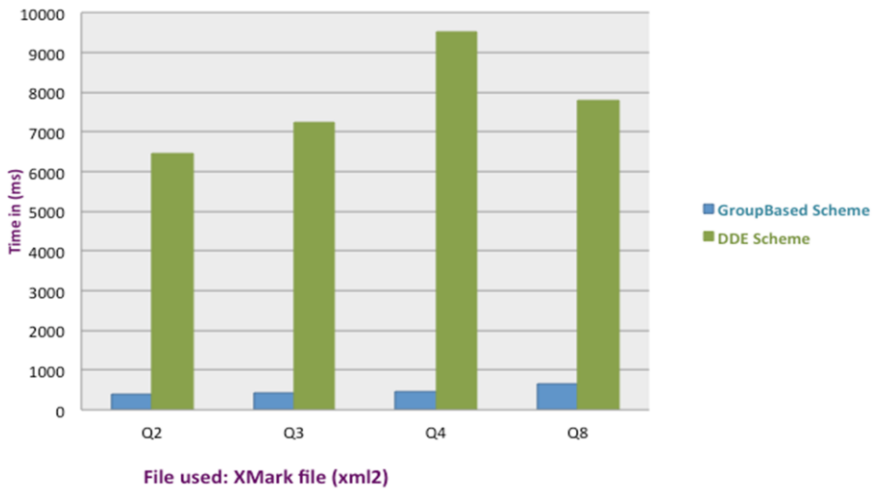


Figure 7.20.3 (b) : Query Evaluation (GroupBased Scheme vs DDE Scheme) on Dynamic Document

7.5.3.2 Statistical Interpretation of the Results

After insertion into dynamic XML was performed also, queries were undertaken by the use of the same nineteen queries already used in this study. It would be recalled that when the queries were performed ahead of the insertions, there were instances where the DDE scheme proved to be more effective while in other instances, the GroupBased scheme showed more efficiency. Almost the same range of results was obtained after the insertion. This is because in queries 1, 2, 16, 17, and 19, the DDE scheme showed to be executed in shorter time under the box plots. All the others however favoured the GroupBased scheme. This necessitated the need to test for significance. The individual outcomes for these 19 populations have been given in Appendix a.10. Figure 7.21.1 shows the outcome for the query 1 after insertion.

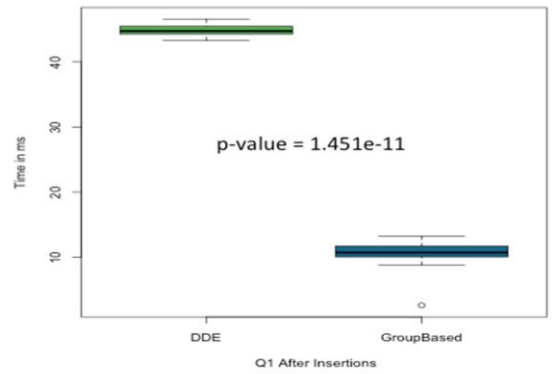


Figure 7.21.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on dynamic XML document **(Q1)**

For the DDE scheme dominated performance, p-values produced showed that the null hypothesis could be rejected in all cases. This is because the p-values produced ranged from 0.02272 in query 16 to 1.451×10^{-11} for all the others. In terms of the GroupBased scheme, the pattern of significance was not different with what was obtained earlier. This is because in all cases, there was very strong evidence to reject the null hypothesis, meaning that for the GroupBased scheme was faster.

7.6 Conclusion

In this chapter, the experimental results were presented and analysed based on the document type: static or dynamic. For each type, three experiments were performed. The static document experiments show, firstly, how the GroupBased scheme is an improvement on the DDE scheme's initial labelling process in terms of time and label sizes; it also shows how these factors are affected by the size of the XML document. Then, the time needed to determine different relationships using the labels and the performance of nineteen different queries were evaluated. The dynamic document experiments started by testing the scheme's ability to handle different types of insertion by avoiding the re-labelling process and how fast the new label was constructed compared to the DDE scheme, as well as the growth rate in the labels' size. Then, the relationships were evaluated again to see

how the insertions affected their calculations. Finally, an evaluation of all queries was performed again to assess the queries' response times.

After obtaining the descriptive results by the use of graphs, a statistical significance test was performed where the box plots and Wilcoxon rank-sum test were used. From these two, p-values were produced, based on which it was possible to find the significance in differences of results obtained between the use of GroupBased labelled scheme and DDE scheme for XML documents. Out of this also, there was an strong statistical endorsement that not only were differences obtained when it comes to time-related experiments but that the differences are significant. Because of the statistical differences obtained, future experimenters and users of XML documents may want to use the GroupBased scheme over the DDE scheme in order to avoid the limitation with the DDE when it comes to time-related performance and functionality.

Generally, the results with a few exceptions from GroupBased scheme were better than those from DDE regardless of their limitations. A discussion of each experiment is provided in the next chapter, which also evaluates the findings based on the research hypotheses in order to outline the research's limitations and offer suggestions for future work.

Chapter 8: Evaluation

8.1 Introduction

The process of evaluation refers to the activities undertaken to determine if a given technique best suits the intended purpose. The aim of the evaluation process is to ascertain if the proposed scheme is working as expected and also whether the scheme fits the intended purpose: i.e., if it has fulfilled the research's hypothesis. There are a number of methods can be used for evaluation. All of these methods test a range of criteria which include robustness, reliability, efficiency, maintainability, functionality and portability (April and Abran, 2012). The evaluation task results in several outcomes that can be utilised in the planning of further outcomes. Evaluation techniques can be classified into two major categories: predictive and descriptive techniques (Perlis *et al.*, 1981). The process of evaluation is goal oriented and the goals towards which an evaluation process is addressed define the importance of such a process. One of the goals for this research is to determine if the proposed scheme is better than existing schemes. This process entails comparing the proposed scheme with other existing schemes, as described in Chapter 7, with the aim of determining the value of the proposed scheme (Clements *et al.*, 2003). This is significant since it helps in assessing the proposed scheme with respect to schemes which already exist and thus to determine its viability.

Another goal of the evaluation process is to find out how effective the proposed scheme is. This process is important since it facilitates the assessment of the proposed scheme to determine if it has the qualities it was intended to possess. Furthermore, the process of evaluation is also aimed at determining the weaknesses of the proposed scheme if any. This attempts to detect any weaknesses in the scheme; this is important since, by using these weaknesses, suggestions for further development can be generated (Farooq and Quadri, 2011).

Thus, this chapter discusses the proposed scheme from an evaluative point of view. As mentioned in the previous chapters (Ch.6 & Ch.7) a number of experiments were performed in order to evaluate the proposed scheme. These experiments tested different aspects of the proposed labelling scheme; their results and analyses were presented in Chapter 7. These experiments are evaluated in this chapter so that the weaknesses and limitations of the proposed scheme, as well as the future trend of this work, can be highlighted.

The chapter starts by describing the potential threats to the experiments and the precautions that were taken to minimise theses threats in Section 8.2. Then, Each experiment is evaluated in Section 8.3. Then, a self-comparison of the proposed and the DDE schemes is provided in Section 8.4. Section 8.5 provides a general evaluation of the proposed scheme, along with the experiments' main findings. The consequences of some implementation decisions (Ch.5) are discussed in Section 8.6 while the experiments' limitations are outlined in Section 8.7. Finally, the chapter concludes in Section 8.8.

8.2 Threats to the experiments

Hakim (2000) noted that in the performance of any scientific experiment such as this one, there are a number of things that can reduce the impact of the results on science, especially things that can be controlled. These things that may impact on the results of the study and its contribution to science are referred to as threats (Bell, 2006, Sapsford and Jupp, 2006). If these threats are not well controlled, they affect the study's validity, reliability and authenticity (Creswell and Clark, 2007, Robson, 2011). A number of such threats were identified in the current study, all of which were addressed with to ensure that the study's findings could be justified as being valid rather than the outcome of chance. These threats are generally referred to as noise.

Noise represent extraneous events that affected the timing of the outcomes of the various experiments undertaken by the two major schemes which were the control and experimental schemes. The experimental scheme use in the study was the GroupBased labelling scheme proposed by the researcher while the experimental scheme was the DDE labelling scheme, which was used to test the effectiveness of the proposed scheme. A total of four experiments were designed to evaluate the proposed scheme's functionality and performance in both static and dynamic XML documents. Each of the four forms of experiments involved recording time with the use of the wall-clock. The sections below addresses noise was controlled in each of the four major experiments where wall-clock time measurements were used.

8.2.1 Presenting equal computer tasks to pairs of experiments

Time was a very important exercise in the whole experiment. In all four experiments, the researcher needed to record the time used by the GroupBased scheme and DDE scheme to undertake different activities. For example in the first experiment, the researcher needed to record the time that the two schemes used to undertake their initial labelling processes. It is important to stress that all these experiments were performed with the use of the computer while taking reading from a wall clock. The rationale for using a wall clock was in the guarantee it gave over the use of the computer's own clock. For example the computer could suddenly go off or get frozen and this might have affected the timing measurement. Whiles using the wall clock to undertake the readings, one of the first things the researcher did to ensure credibility with the readings for both sets of experiments was to present the same tasks from the computers. To ensure this, the researcher used the task manager to display all programs running in the background of the computer.

The task manager also revealed the apps that were running on the computer as well as windows processes. While some of the background processes and windows processes were needed to keep the computer running smoothly, most of the apps could be done away with. The researcher therefore closed all apps that were not needed as part of the experiment. All background and windows processes that could also be closed without any impact on the computer's function were also closed. The total number of background and windows processes was observed for each pair of experiment to ensure that they were always the same. Once this was done, the researcher did not have to worry about CPU, memory, disk, and network consumption of the computers and how these affected the results. This is because given the same number of apps, background processes and windows processes the consumption was almost the same in all cases. Providing equal computer tasks to each pair of experiments for all four experiments ensured that results gathered were hardly influenced by other computer tasks that were running on the computer used. This way, credibility of results was enhanced because there was fair basis given for the experiments (Cooper, 2008, Remenyi, 1998).

8.2.2 Test-retest reliability

The second approach used to minimise or deal with the threat of noise was test-retest reliability testing. In scientific experiment, reliability is said to be attained when the results are more than one-off findings but inherently repeatable (Collis *et al.*, 2003, Saunders *et al.*, 2011). What this implies is that when the researcher repeats the experiment in any other research setting where the variables remain the same, the results must be relatively same (Gill and Johnson, 2010). There are several ways in which reliability can be guaranteed, including the use of test-retest (Adams and Schvaneveldt, 2011). More particularly, the researcher selected the use of test-retest as it afforded the opportunity to determine if there was any hidden noise that affected a single experiment. A very simple approach was taken to test-retest reliability. This was done by ensuring that for each of the four experiments, a minimum of twenty tests were used to measure the same outcome. For example in the second experiment the researcher assessed the time needed to

determine the different relationships that existed between different nodes in the GroupBased scheme on one side and the DDE scheme on the other side. In order to ensure test-retest reliability, the assessment of time for the GroupBased scheme was performed on twenty different occasions for the same experiment.

While doing the above, the researcher ensured that the earlier provision of providing the same computer tasks for each pair of assessment was in (Ghauri and Grønhaug, 2005). When the test-retest was done, it was revealed that the timing or readings made for each set of assessment were relatively close. Where there were any differences, they varied by less than 0.05%. Nevertheless such differences could have had an impact on the reliability of the study. In effect, the mean or average reading made for the three sets of assessments were taken and are presented in the final outcome of the study in Chapter 7. It is also important to note that one other way in which the test-retest was done was by various different computers, the researcher's personal computer and the university's lab-computer. Even though it was difficult controlling the activities running in the background of the university's lab-computer, it was seen that the results collected from the lab-computer were not significantly different from those collected from the personal computer. This helps in concluding that computer did not have any major impact as a noise threat to the study.

8.3 Evaluation of the Experiments

In this section, the design and the results of the experiments performed are evaluated (Ch.6 & Ch.7).

8.3.1 Evaluation of the Initial Labelling Experiment

As explained in Chapters 6 & 7, the initial labelling experiment aimed to assess two factors: the labels' size and time needed for assigning each label, and how these factors are affected by the size and the structure of the XML tree. In this section, this experiment is evaluated.

Generally, the results met the aim and expectation of the experiment (Ch.6) as they proved the exponential correlation between the initial labelling time and the size of the XML file, as well as the depth of the XML tree. Additionally, better performance was noticed in the XML tree with a deep rather than a wide structure.

With regards to the comparison between the proposed scheme and the *'Dynamic Dewey'* scheme (DDE), the former showed an exponential improvement in the initial labelling time. This improvement can be justified based on how each scheme calculated and assigned the labels (Ch.5). As explained in Chapter 4, calculating the proposed scheme's labels was achieved by using a simple addition operation while calculating the DDE label involved string matching and concatenation which was more time-consuming. As shown in the previous chapter, the DDE initial label sizes were slightly smaller than the proposed scheme's label sizes; this is to be expected as the proposed scheme's label consists of two labels: global and local (Ch.4). Based on the results obtained, the proposed scheme was shown to outperform other labelling schemes (Li and Ling, 2005a, Liu *et al.*, 2013, O'Neil *et al.*, 2004, Qin *et al.*, 2012, Xu *et al.*, 2012) as regards time only.

Generally, the findings answered the research questions of this experiment. However, the experimental results could be extended to test the scheme's ability with even larger XML files which might lead to a more reliable scalability evaluation.

8.3.2 Evaluation of Relationships Experiment

The evaluation of relationships experiment was designed to test how fast each relationship was determined (Ch.4 & Ch.6). This experiment was evaluated using static and dynamic documents employing both the proposed and the DDE labelling schemes. The experimental design worked as intended and the results were better than expected.

The results obtained when the experiment was run on static and dynamic XML documents (Ch.7) met the research's expectations by determining that on static documents the different relationships were identified faster using the GroupBased scheme than the DDE scheme. At the same time, this experiment on dynamic files also exceeded expectations since the parent-child, ancestor-descendant and lowest-common ancestor relationships were determined much faster than the DDE scheme, especially after *'uniform'* and *'ordered'* types of insertion. This time improvement related to the simple relationship calculation (Ch.4) for both types of document in the proposed scheme unlike the scheme relationships calculations in the DDE (Ch.3 & Ch.4).

Although the experiment could be considered to be limited as it was run on only one dataset, its results were compared to the *QED-based* labelling schemes (Li and Ling, 2005a) and the *ORDPATH* scheme (O'Neil *et al.*, 2004) as the same dataset was used, as mentioned in Xu *et al.* (2009), Xu *et al.* (2012) and Liu *et al.* (2013). The comparison supported the case that the proposed scheme was more effective than these schemes mentioned above in determining different relationships.

8.3.3 Evaluation of the Queries Experiment

As illustrated in Chapter 6, nineteen out of twenty XMark queries were evaluated using the proposed scheme and the DDE scheme. The experiments were performed twice to test the schemes under both static and dynamic circumstances. The experimental results (Ch.7) showed the benefit of using the proposed scheme instead of the DDE scheme with both static and dynamic documents, as described below:

- ***Static XML document:***

  The experimental results (Ch.7) on the static document were as expected (Ch.6) for most of the queries tested, especially for complex queries where several join operations were required. However, five queries were below expectation as using the DDE scheme showed slightly better response times. These queries (Ch.6) were based on a simple parent-child relationship and because the document was static, this relationship was determined based on the 'Dewey' labelling scheme instead of the DDE scheme. In the Dewey scheme, the child node's label equals the parent node's label plus '1' as the last component of the child's label; this is faster than both the proposed and the DDE schemes.

- ***Dynamic XML document:***

  The results of repeating the same experiments on a dynamic document met expectations (Ch.6) as using the proposed scheme provided an exponential improvement in all query response times. This was due to the fast relationship determination, as explained in Section 8.2.2.

The experimental design met its objectives and the results were promising. Nevertheless, the experimental boundaries could be expanded to evaluate the scheme on more complex queries, using other XML benchmarks that consider update queries (Ch.6). This is because the XMark queries set lacks this type of

query and the ability to perform comparative tests between different benchmarks; this would definitely facilitate more accurate and reliable results in terms of query performance.

8.3.4 Evaluation of Handling Insertions Experiment

The design of the evaluation of handling insertions experiment (Ch.6) aimed to test the proposed scheme's ability to handle different types of insertion by measuring the insertion time and the labels' sizes after insertions. To recap, these types of insertion are: *'uniform insertions'* which refers to the insertion of a new node between each consecutive node; *'ordered skewed insertions'* which refers to an insertion before and after a specific node repeatedly; and *'random skewed insertions'* which refers to randomly inserting nodes between two consecutive nodes. The experiment was run on both schemes and the experiment's framework served its purposes; the results (Ch.7) were partially what were expected but others were opposite to expectation, as illustrated below:

- ***Insertion Times:***

  The results obtained by measuring the time required to perform the *'uniform'* and the *'random skewed'* types of insertion was better than expected and the proposed scheme was shown to be quicker than the DDE scheme, whereas the DDE scheme showed slightly better performance in the *'ordered skewed'* insertions. The expected result was that the DDE scheme would offer slightly better performance in all types of insertion; this expectation was based on the simplicity of the scheme's implementation. However, as explained in the initial labelling evaluation (Sec. 8.2.1), calculating the proposed scheme's labels required a simple addition operation, especially when inserting between two consecutive nodes; this could explain the time improvement in the '*uniform'* and the *'random skewed'* insertions. This did not apply to the *'ordered skewed'* insertions since the experiment was run on a worst-case scenario where the insertions occurred before and after a node that was a child of the document-root.

This meant that, for every new inserted node, two labels were calculated and assigned (Ch.4 & Ch.5).

- ***Label Sizes***

  In term of the labels' sizes, the labels of the DDE scheme were allocated less memory after all types of insertion, as shown in Chapter 7; this is reasonable as the difference between both schemes was insignificant and the proposed scheme's label consisted of two labels (Ch.4).

8.4 The Schemes' Self-Comparisons

In general, the process of evaluation is aimed at enhancing the usability of any given technique while this process of enhancement is aimed at improving users' experiences, detecting flaws in the technique, addressing concerns, and removing unwanted features from the technique. This process is vital since it plays a significant role in the development of the technique. Moreover, the formative aspect of the evaluation process facilitates the detection of usability problems associated with the technique (Vlahavas *et al.*, 1999).

From this point of view, evaluating the scheme against itself was undertaken and the results presented in this section. This evaluation intended to add more clarity to the comparison as well as to determine the scheme's limitations and offer recommendations to improve it. Each scheme was evaluated using the experimental results presented in Chapter 7 by comparing the schemes' abilities in static and dynamic documents in all experimental aspects (Ch.6). However, to facilitate this evaluation, an XMark file (xml2.xml) was used (Ch.7) which consisted of 17,132 nodes. Moreover, 12,503 nodes were inserted to assess the scheme after the insertions; this number of nodes was determined based on the *'uniform insertions'* process and so the same number of nodes was used in all types of insertion.

8.4.1 The GroupBased Schemes' Self-Comparisons

In this section, the initial label sizes and the initial labelling times were compared to their correspondences after different types of insertion. The query response times and the time needed to determine different relationships before and after insertions were also compared.

- ***Label Sizes:***

  The size of the labels after the initial labelling process was 0.015 MB. This size was about five times more after the *'uniform'* insertions whereas, it increased by 70% and 34% respectively above the initial size after the *'random'* and *'ordered'* insertions. This is shown in Figure 8.1.

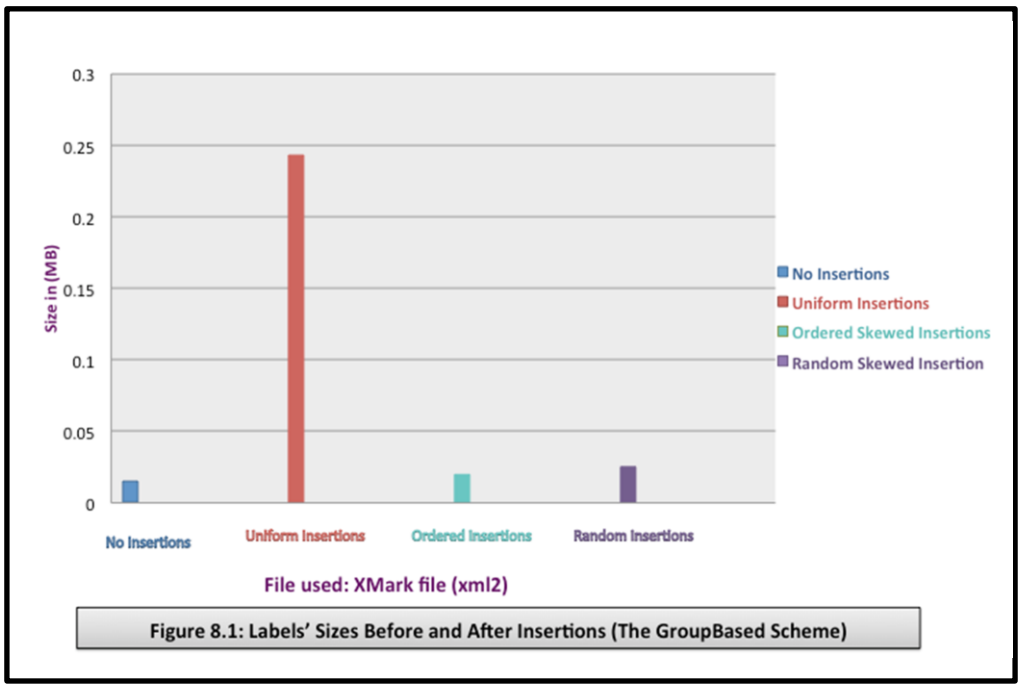


Figure 8.1: Labels' Sizes Before and After Insertions (The GroupBased Scheme)

- ***Labelling Time:***

  Unlike the label sizes, *'ordered skewed'* insertions indicated the longest label construction time which was more than double the initial labelling time of 415 milliseconds. The *'uniform'* and '*random skewed*' insertions, on the other hand, consumed 8% and 20% less time respectively than the initial labelling, as shown in Figure 8.2.

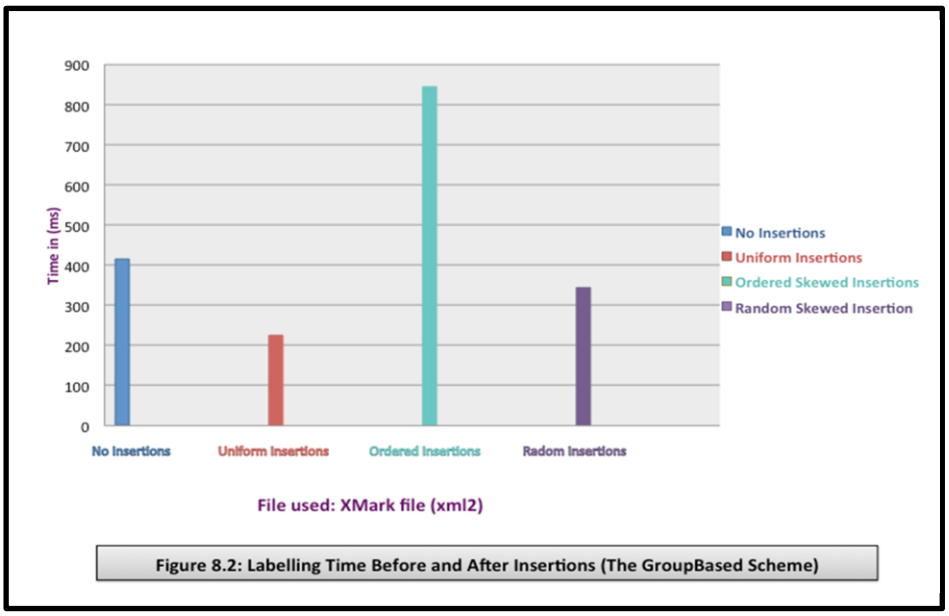


Figure 8.2: Labelling Time Before and After Insertions (The GroupBased Scheme)

- ***Determining Different Relationships:***

  Measuring the proposed scheme's stability in determining different relationships demonstrated its constancy in calculating most of the relationships because, in calculating the order, sibling and parent-child relationships, it consumed the same time before and after the three types of insertion. Also, the level's calculation time was consistent after the first insertion but it increased by 25% compared to the times before any insertions had been made. The calculation time for the ancestor-descendant and lowest-common ancestor relationships was consistent before and after the *'uniform'* and *'ordered-skewed'* insertions while it was increased by 200% and 57% after the *'random-skewed'* insertions. Figure 8.3 shows a comparison of the relationships in time before and after insertions.

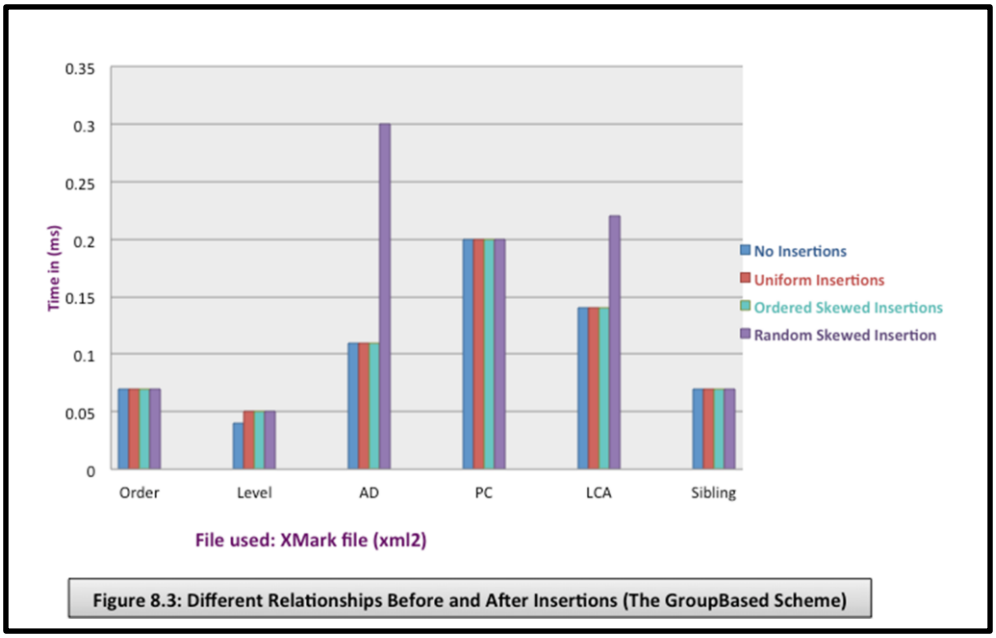


Figure 8.3: Different Relationships Before and After Insertions (The GroupBased Scheme)

- ***Queries' Performance:***

Due to the nature of the XML tree structure and the complexity of the queries evaluated (Ch.6), most of them required multiple join operations either on values or references and aggregations, their response times might be considered high before any insertions, even though it was faster than the DDE scheme (Ch.7) for most of them.

Generally, the query response time was affected by the complexity of the query and the number of nodes examined. Thus, after the insertions, this time showed a significant increase in six out of nineteen queries by more than twice the time taken before any insertions. However, it is reasonable to state that the time taken to answer some of these queries can be considered small even after this increase; i.e. it was less than one second. Figures 8.4(a, b, c) show these comparisons.

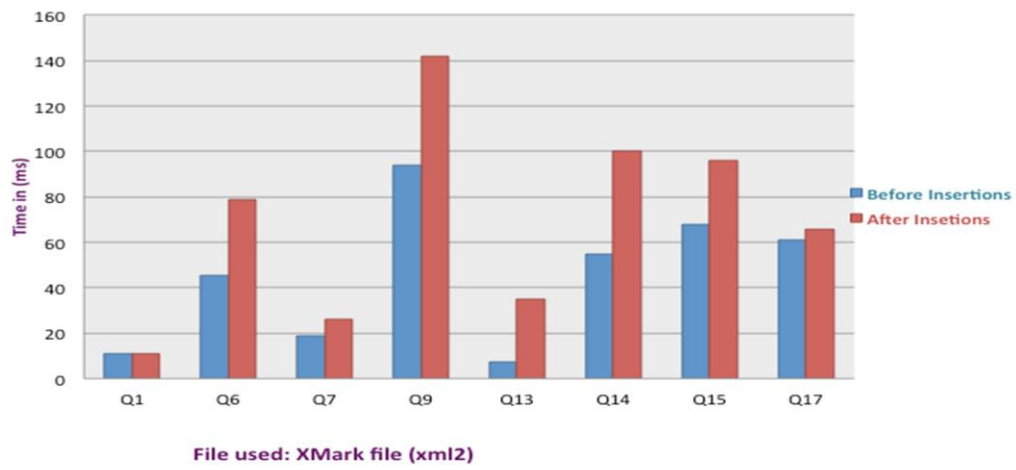


Figure 8.4(a): Query Evaluation Before and After Insertions(The GroupBased Scheme)

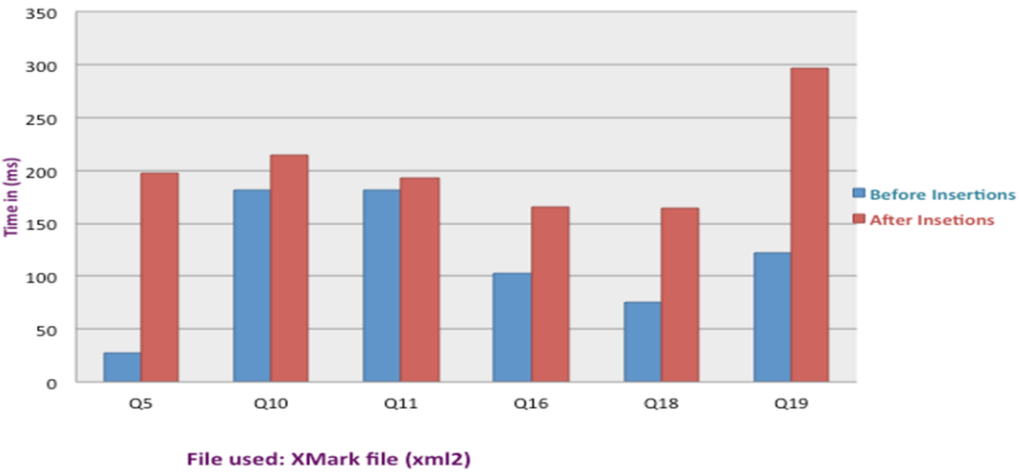


Figure 8.4 (b): Query Evaluation Before and After Insertions(The GroupBased Scheme)

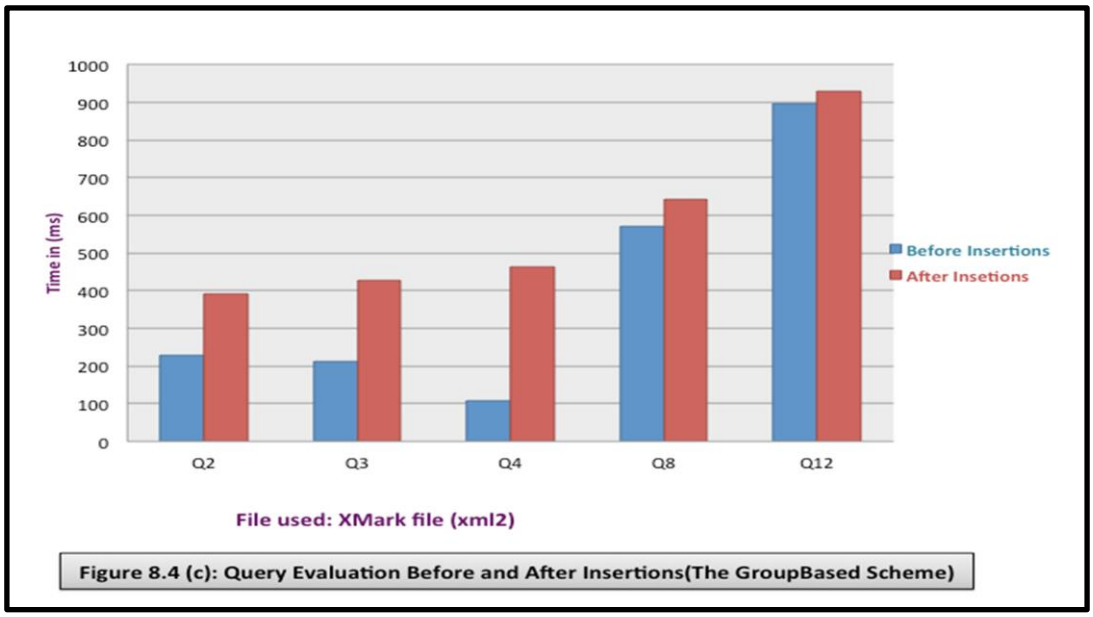


Figure 8.4 (c): Query Evaluation Before and After Insertions(The GroupBased Scheme)

8.4.2 DDE Scheme's Self-Comparisons

Similar to the proposed scheme, in this section, the DDE scheme's capabilities before any insertions are compared against itself after the insertions.

- ***Label Sizes:***

  The size of the labels after the initial labelling process was (14 MB). The *'random skewed'* insertions hit the highest size increase (76%) whereas the *'uniform'* and *ordered skewed'* insertions showed 67% and 39% increases respectively, as shown in Figure 8.5. This indicates that the DDE scheme provides less memory allocation when performing *'uniform'* insertions.

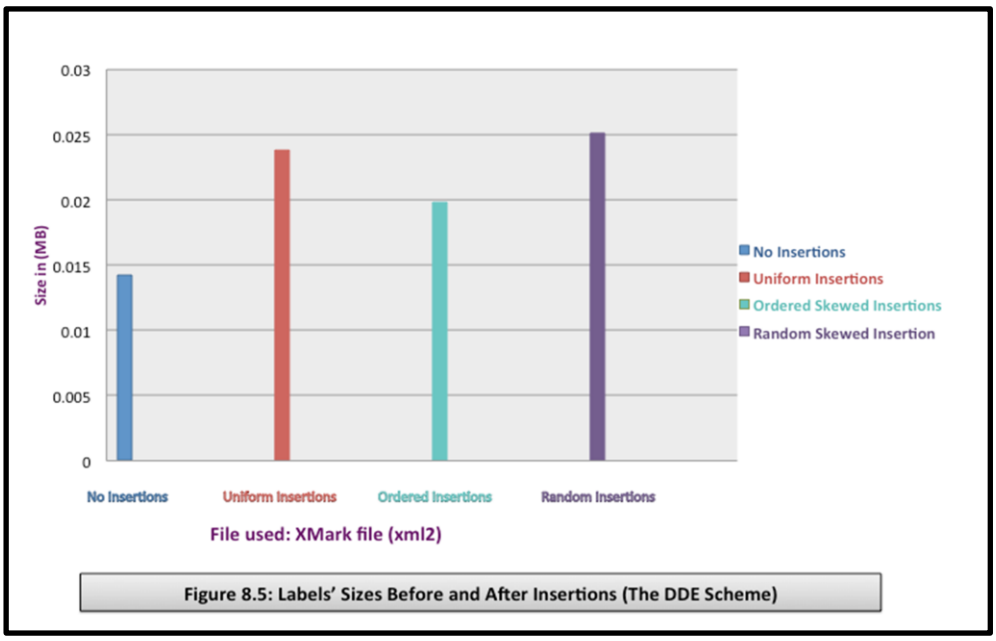


Figure 8.5: Labels' Sizes Before and After Insertions (The DDE Scheme)

- ***Labelling Time:***

  Unlike the proposed scheme, none of the insertion types exceeded the time consumed during the initial labelling, which was 56% more than the GroupBased scheme's initial labelling time. However, among the three types of insertion, the *'uniform'* insertions showed the better performance, at 75% less than the initial time, whereas the *'ordered'* and *'random'* skewed insertions consumed only 19% and 14% less respectively, as shown in Figure 8.6.

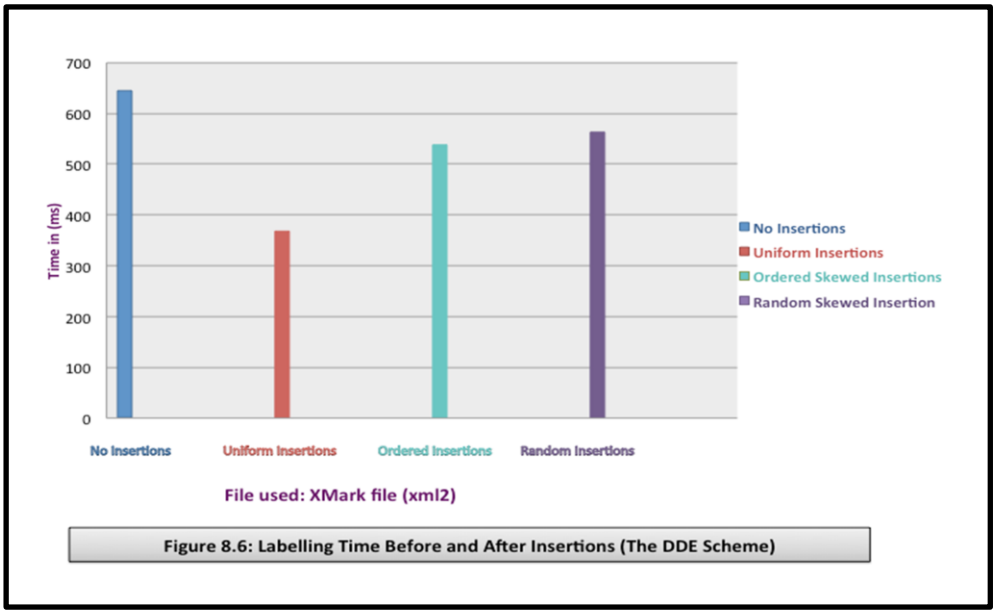


Figure 8.6: Labelling Time Before and After Insertions (The DDE Scheme)

- ***Determining Different Relationships:***

  Similar to the proposed scheme, the DDE scheme showed partial stability in computing different relationships. The level and order relationships were consistent in terms of calculation time before and after insertions. However, the parent-child relationship showed a significant time increase after the *'ordered skewed'* insertions with more than twice the time before insertions, and by 34% and 42% respectively after the *'uniform'* and *'random'* skewed insertions. The calculation time for the ancestor-descendant and lowest-common ancestor relationships was the same after insertions, which was 86% and 25% higher than their time before insertions. The sibling relationship gave the same calculation time before and after the *'uniform'* and *'ordered-skewed'* insertions and increased by 28% after *'random-skewed'* insertions. Figure 8.7 shows the comparison of the relationships' time before and after insertions.

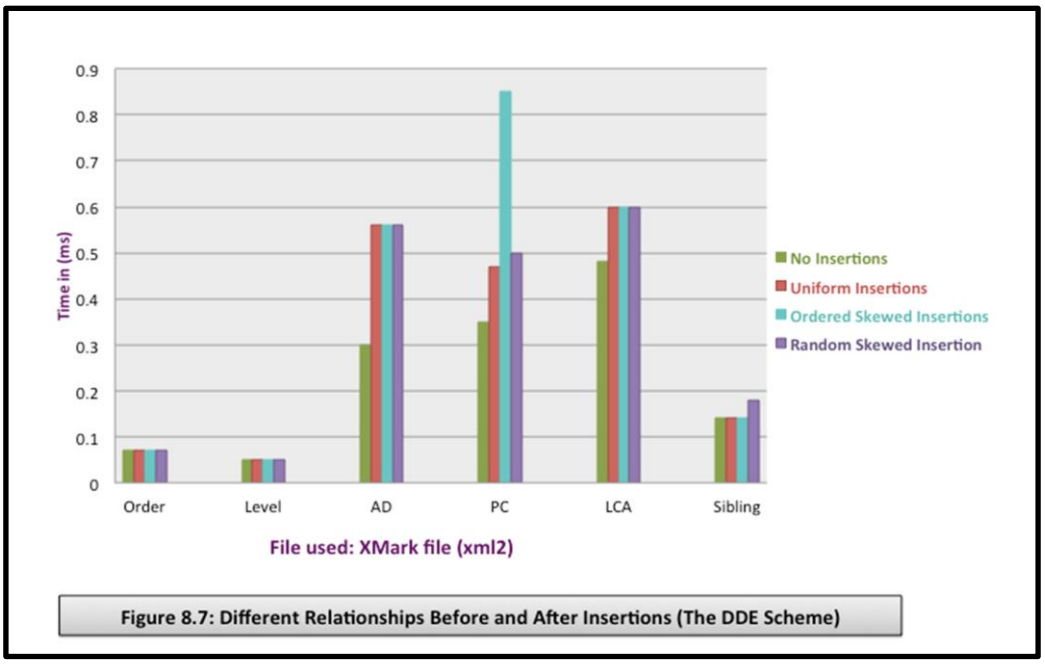


Figure 8.7: Different Relationships Before and After Insertions (The DDE Scheme)

- ***Queries' Performance:***

Despite the high response time, the response times for 6 queries increased by more than six times their time before insertions, even for the simplest query such as Q1 which is a simple exact match query. Figure 8.8 shows the queries' response times before and after insertions.

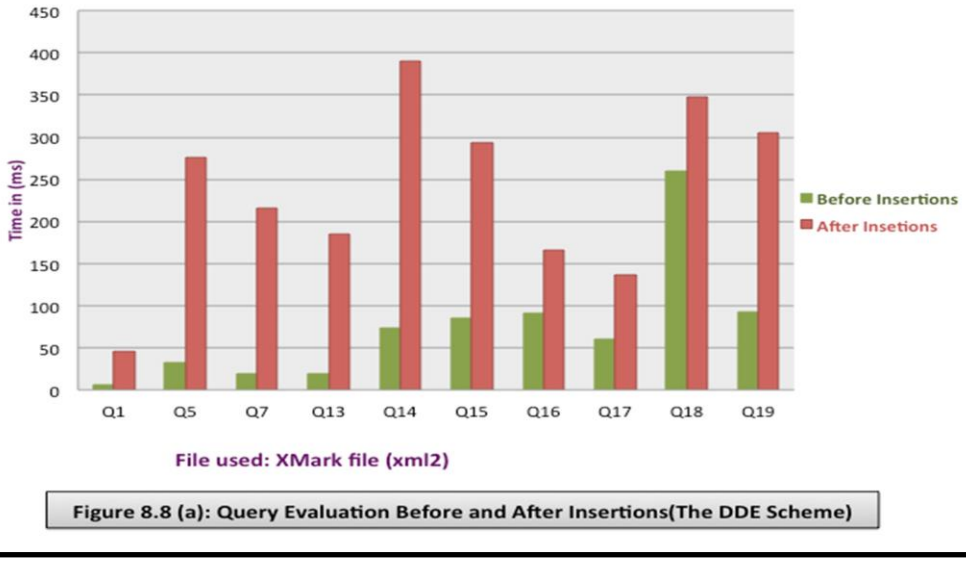


Figure 8.8 (a): Query Evaluation Before and After Insertions(The DDE Scheme)

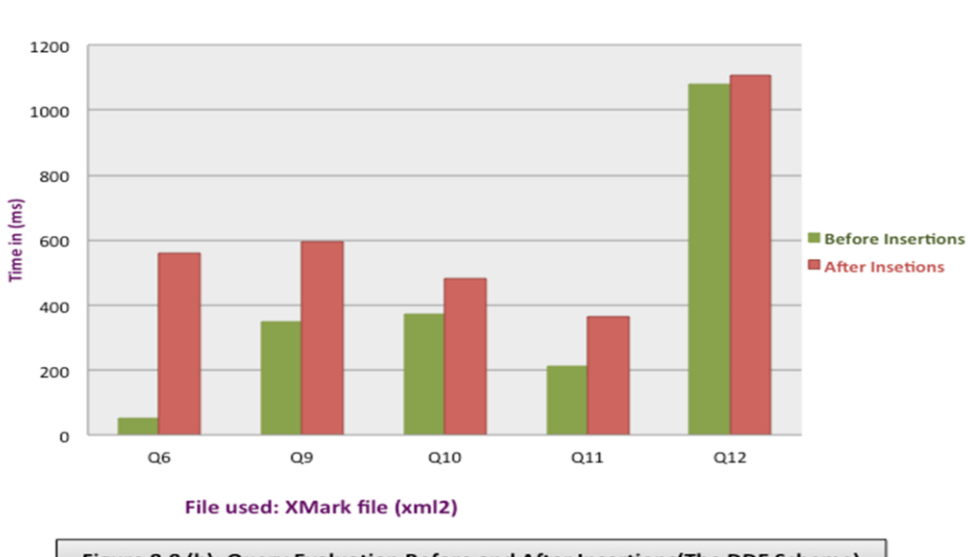


Figure 8.8 (b): Query Evaluation Before and After Insertions(The DDE Scheme)

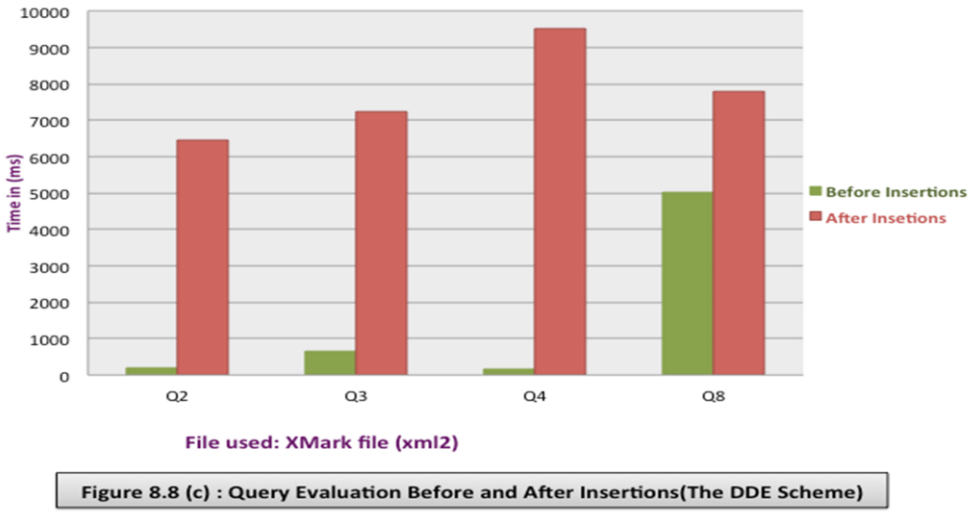


Figure 8.8 (c) : Query Evaluation Before and After Insertions(The DDE Scheme)

8.5 The Proposed Scheme: General Evaluation

This section provides an overall evaluation of the proposed scheme based on the research hypothesis stated in Chapter 1, i.e.:

"Applying a second layer of labels and grouping the nodes based on the parent-child relationship may facilitate node insertions in dynamic XML data in an efficient way, offering inexpensive labels without excessive label size growth rate in which it is easy to maintain structural relationships, as well as improved query performance."

Then, the main experimental findings are highlighted.

The GroupBased scheme was designed to improve the XML labelling by providing a scheme that deals with insertions without the need for re-labelling and without sacrificing the performance, construction-time and memory usage of queries. The overall rationale for this was that there are core qualities of all XML labelling schemes which must be maintained, but it should be possible to improve on them. These qualities were deemed essential because of the advantages they offer to data

interchange programming. As mentioned earlier, some of these qualities were query performance, construction time and memory usage (Fennell, 2013). To get a realistic measure of the GroupBased scheme's performance or merits, it was important to include something that would enable comparative measurement. This necessitated the introduction of an implementation of the Dynamic Dewey labelling scheme (DDE) on which the experiments were carried out to allow comparable evaluation.

To test the research hypothesis, the scheme was implemented based on the rules and characteristics defined above (Ch.4). The design and implementation specifications were provided in detail in Chapter 5. As explained in the earlier chapters, the DDE scheme was implemented as it contributed to the formation of the proposed scheme and is currently the state of the art scheme. In order to evaluate the performance of the proposed scheme, four main experiments were performed to test whether the scheme fulfilled its intentions. The experimental framework of these experiments and an analysis of their results were discussed in Chapter 6 and Chapter 7.

Generally, it is fair to state that the research hypothesis was partly supported by the results; some of the results obtained were fully supportive of the hypothesis. For example, the hypothesis tested three major outcomes as far as performance is concerned. These were the need for the scheme to facilitate node insertions efficiently way, the need to provide inexpensive labels, and the need to achieve improved query performance.

The result of node insertions in dynamic XML data are given in Chapters 6 and 7. The GroupBased scheme gave better performance in determining different relationships in static form as against the DDE scheme, with up to 1.5% of time saved. For efficiency to be attained with the node insertions, it is important that much insertion be done within a very short time frame (Murata *et al.*, 2001). In this regard, the first component of the hypothesis was slightly supported, with the

GroupBased scheme outperforming the DDE scheme in terms of determining relationships. Nevertheless, the performance in calculating levels was variable.

The second component of the hypothesis was expected to offer an inexpensive label with an adequate label size growth rate where the structural relationships are easily maintained. To measure this, an experiment evaluated the growth of label size in terms of memory allocation. The findings showed that there was only a slight change in label size between the GroupBased scheme and the DDE scheme. This is a weak indication that the hypothesis can be accepted in this context, because the rate of difference was merely 0.002%. This was lower than expected, given that the GroupBased scheme's label was made up of two labels whereas that of the DDE scheme had only one.

The last aspect of the hypothesis focused on improvement of query performance. This was a very important aspect of the whole experiment, given the role played by query performance in query response time. Again, the emphasis was on a comparative analysis designed to measure how effective the proposed GroupBased scheme was over the DDE scheme. Of the 19 queries that were used, the results showed that different queries achieved different times. Overall however, there were better responses with the GroupBased query performances, as shown in Chapter 7. There were actually only 5 queries for which the DDE scheme outperformed the GroupBased scheme. This means that the hypothesis can be accepted on the grounds of improved query performance as well.

Accordingly, it can be stated that the scheme's implementation worked as intended, proving its superiority to other similar labelling schemes in many comparable time-related aspects. Nevertheless, the scheme was found to be inferior in some respects, such as label sizes and some other time-related features.

As a consequence of the evaluation process, some changes could be made to improve the scheme's efficiency. From a design standpoint, the proposed scheme could be redesigned using another dynamic labelling scheme instead of the DDE

scheme in order to enhance the growth of the label. However, the DDE scheme was chosen in the first place for its simple implementation and its feature of extracting the different relationships from the label. This was one contribution of the thesis.

From an implementation perspective, the scheme was implemented as efficiently as possible based on its design. However, the implementation was a complex process due to the two labels that formed the proposed label. Thus, as each label required a number of processes, this could be considered as a drawback. Also, although synthetic and real datasets were used in the experiments to cover as many scenarios as possible, lack of resources and time restrictions limited the scalability experiments. The results gathered were sufficient, however, to analyse and evaluate the hypothesis. More datasets could be used to provide more analytical results.

8.5.1 The Main Experimental Findings

The most important finding confirms the hypothesis as the GroupBased scheme showed better time performance than did the DDE scheme and other similar labelling schemes. The flexible and fast calculation of different relationships led to faster answering of queries. Despite the scheme's complex implementation, calculating the labels was fast, which resulted in handling *'uniform'* and *'random skewed'* insertions efficiently. Even though label sizes grew slightly faster than the DDE labels, and the proposed scheme was slower when performing *'ordered skewed'* insertions, it delivered better scalability by providing more consistent results both before and after insertions. Moreover, it has been established the proposed scheme showed better performance on a deep-tree structure rather than a wide-tree structure. The whole research findings are presented in the next chapter.

The main experimental findings above can be further evaluated with respect to the two main objectives of the experiments, which were to check for initial labelling time and label size. In data interchange on the web and on any other platform,

Rusty (2004) noted that the activities of debugging programs, storing small and large amounts of data, and providing scalability for configuration files is very important. It is for this reason that XML would mainly be relied upon to execute all of these functions. Yet Cunningham (2006) emphasised the need to accept that in XML, the initial labelling time and label size played crucial roles when talking about efficiency and effectiveness respectively. It was for this reason that the place of the GroupBased scheme in attaining such goals was measured as against the conventional DDE schemes.

The experimental findings on initial labelling time showed three major outcomes. The first is that both the GroupBased scheme and DDE scheme showed significant growth of time consumption as the size of the file increased. This means that the XML labelling as a collective entity does not necessarily guarantee lower turnaround times. The second outcome was that regardless of the limitation of not guaranteeing lower turnaround times during initial labelling time, the GroupBased scheme was more efficient than the DDE scheme. This is because the time taken to undertake the initial labelling via the DDE scheme was 40% greater at the initial stage and then rose to much as 180% greater in latter stages. The implication here is that even if the GroupBased scheme does not reduce the time for initial labelling, using DDE would be worse. The third outcome was that the size of file plays an important role in efficiency and initial labelling time. This is because the difference in initial labelling time between the GroupBaseed scheme and the DDE scheme kept increasing with increasing file size so the advantage of using GroupBased increased with file size. The implication here is that for best initial labelling times, the GroupBased scheme must be used.

With regard to label size, the experimental findings showed that file size was an important determinant in the growth of label sizes. For both schemes, it was only when there was an increase in the file size by a margin of 5MB that there was a corresponding increase in growth of the label size by 0.056 MB. Indeed, the increase in growth of the label size, when compared to the file size, can be said to be marginal. Between the two schemes however, the growth of the label size was

higher for the proposed GroupBased scheme, though the difference was only 0.002%. It is important to emphasise that an increase in label growth size for the GroupBased scheme was expected, but not at the rate at which it was recorded. The increase in GroupBased scheme was expected because, as shown in Chapters 4 and 5, two labels are used in the GroupBased scheme compared to only one in the DDE scheme. For this reason the hypothesis was not rejected, despite the fact that the proposed GroupBased scheme brought about an increase in growth of the label size.

8.6 The Consequences of Some Practical Decisions

In evaluating the main experimental findings, a very strong case can be made for the relationship between time, size and query performance. Sean (2006) had argued that there are several advantages over conventional data exchanging programmes that make it preferred over HTML. One of the qualities of XML for which it may be preferred includes the fact that it allows multiple functionality based on its plasticity to dynamic add-ons and changes. The experimental findings obtained however suggests the need to give this attribute a second look. This is because even though XML labelling may cause overflow of space available, overloading the programming outcome of the XML by the use of schemes may lead to label sizes growing even faster and becoming slower (Bosak and Bray, 1999). Future implementers may therefore have to make a case between having a full XML labelling programme that produces multiple series of functionality with increased label size and time-consuming pace, or one that is focused on fewer functions in order to guarantee efficiency.

With reference to the tree structures on which the two schemes operated, there are some practical implications for implementers. In particular, there are areas or aspects in which the proposed GroupBased scheme has advantages over the DDE scheme, but there are other times that the opposite is true. For example, the findings showed that performance on a deep-tree structure gives essentially the same value between the two schemes. Here, selection of scheme to operate on

deep-tree structures could go either way. But even on deep trees when it comes to time used in performance, the proposed GroupBased scheme can be said to be superior. This is because it offers better performance with respect to time whether taken from a wide-tree structure or a deep-tree structure. The difference in performance time between the two schemes was as much as 165%. The problem arising, which requires careful consideration in making selection, has to do with the more concise labels generated by the DDE scheme than the GroupBased scheme. Thus, implementers have to be certain of their ultimate goal before making a selection.

A DOM parser was applied for the implementation of the proposed scheme, as discussed in Chapter 5. The features of the parser resulted in this decision being made as it ensured that any section of the document could be easily accessed, thereby allowing the XML tree to be effectively modified. Additionally, the functionality of this parser further simplifies the access and retrieval processes that occur. However, one of the major disadvantages of DOM is that it is highly inefficient with regard to memory usage. It creates a tree of nodes that are stored within the memory, and is reflective of the size of the documents, which can be especially problematic for large document. The parsing and labelling processes can consequently become slower, resulting in an 'out of heap' memory during the tree loading process, and subsequently reducing the effectiveness of the operation.

The *ArrayList* suffers from the same disadvantage. If the array is completely full, then any additional elements require further memory, often at a significant cost (approximately 1.5 times the original array size). These elements are copied over from the old source into the new source, which results in *O (n)*. This issue is especially problematic as the label number cannot be easily extrapolated before the process takes place, as there is variation between different documents. Additionally, the use of ArrayList results in a distortion of the element positions; the latter elements having to be shifted to make new spaces, in spite of the efficiency of the addition and removal process. This operation is highly slow, especially when insertions are occurring.

8.7 Experimental Limitations

Despite the fact that all the experiments worked and served their purposes, limitations were detected. These arose as a consequence of the simple experimental design, which was selected to validate the scheme's capabilities before extending it to a more complex level. Generally, all the experiments might be extended by using more datasets, more complex and varying queries, and different comparable schemes in order to obtain more elaborate results. The document size restrictions could be temporarily improved by using a more efficient platform; however, this will always be an issue as the data increases. Some calculation and storage approaches could also be improved to achieve better performance.

In order to adhere to the hypothesis that was set from the beginning of the study and restated in this chapter, it was important that the experiments be focused and limited in design to testing the hypothesis. However, this requirement was itself found to create a form of limitation since the study could not be extended to XML document parsing and storage mechanisms. To this point, it is not certain whether the proposed GroupBased scheme has any effect on how XML documents are parsed and how the labels and data associated with them are stored. What is more, even though the approach to the experiment was to avoid re-labelling for inserting new nodes, this could not be entirely followed to the end. This is because re-labelling was found to be required in cases where the structure of the XML document was changed. (Sean, 2006) confirms the rapid speed at which XML documents change in the real world.) In this respect, the proposed scheme did not fully consider re-labelling.

Even though re-labelling was not fully considered in the research, re-labelling could be required in certain complex situations. A typical example of such a finding, backed by literature from Bosak and Bray (1999), is the high exponential growth that was recorded in labelling time and label size. Notwithstanding the exponential growth in labelling time in the DDE scheme was much higher, than in

the proposed GroupBased scheme. Having said this, it must be reiterated that the general scheme evaluation which focused on testing the hypothesis, makes it possible to conclude that the study has tested the hypothesis and covered all the experimental aspects, particularly in terms of labelling time and label size.

8.8 Conclusion

The process of evaluation is paramount in every software development process, although it is overlooked in some instances. It is important to evaluate closely any process or technique in order to determine the limitations associated with it, and to highlight future work to enhance usability. An evaluation of the experiments and their results was presented in this chapter, demonstrating the proposed scheme's efficiency and scalability as compared to the DDE scheme. An overall evaluation of the scheme was provided, along with the main findings of the experiments, while taking into account the simple experimental frameworks and the limited datasets used. This was intentional in order to ensure that the scheme worked properly before extending it to further, more complex development. Some suggestions of importance to the experiment were also briefly mentioned.

Chapter 9: Conclusion

9.1 Introduction

This thesis has highlighted the difficulties associated with employing a dynamic labelling scheme for XML documents. The difficulties include the complexity of the process, inefficient querying or labelling, and large storage needs. The fully dynamic labelling scheme, GroupBased, was proposed to resolve these issues. The goals of this scheme were discussed in the earlier chapters with reference to the evaluation of perspectives, results, experiments, implementation and design. Section 9.2 summarises the data and the work completed in order to conclude the thesis. Section 9.3 discusses the main contributions of the research and Section 9.4 details how the results support the initial hypothesis. Section 9.5 addresses ways in which the scheme's development could be enhanced and suggests future directions for investigation.

9.2 Thesis Summary

This study assessed the Groupbased labelling scheme which was created to handle dynamic XML documents by responding quickly to queries and creating labels after insertions while avoiding the need to re-label. Chapter 1 outlines the aims and objectives of this study. XML has become a standard of information exchange and representation on the web. Labelling schemes, such as ancestor-descendent, are frequently employed to define the connections between two element nodes in order to query the XML data efficiently. In static XML documents, queries are processed efficiently by existing labelling schemes such as the containment scheme (Zhang *et al.*, 2001), the Dewey scheme (Tatarinov *et al.*, 2002) or the prime scheme (Wu *et al.*, 2004) when the XML is static. However, XML databases suffer from the bottleneck effect when the XML data are dynamic and a large number of nodes require expensive re-labelling. Creating dynamic labelling schemes that circumvent the re-labelling of a current node is a vital research problem.

Various labelling schemes (Li *et al.*, 2008, Li *et al.*, 2005, O'Neil *et al.*, 2004, Xu *et al.*, 2007, Xu *et al.*, 2010) have been suggested to support dynamic XML documents. An effective labelling scheme needs to label and respond to queries efficiently, create unique labels continuously, be concise, avoid the re-labelling of nodes and be capable of identifying structural relationships immediately. Finally, a successful labelling scheme needs to be easy to comprehend and execute. Designing a labelling scheme that possesses all of these properties is the purpose of this investigation.

In general, labelling schemes that create small labels are not dynamic or they fail to give enough information to classify all of the structural relationships between nodes (Dietz, 1982, Li and Moon, 2001, Yun and Chung, 2008. However, dynamic labelling schemes require additional storage space, are less efficient when evaluating queries (Cohen, 2010, Duong and Zhang, 2005, Duong and Zhang, 2008, Gabillon and Fansi, 2005, O'Neil *et al.*, 2004, Tatarinov *et al.*, 2002) or are unable to create unique labels continuously (Duong and Zhang, 2005, Duong and Zhang, 2008).

The GroupBased labelling scheme delivers enhanced performance regarding labelling time, avoids re-labelling, identifies structural data, and responds to queries for both static and dynamic XML documents.

The research motivations and objectives, as well as the hypothesis, are described in Chapter 1; the structural organisation of the thesis is also outlined in this chapter. An overview of XML database technology, along with its main topics, such as the basic component of XML documents, query languages and parsing methods, are described in Chapter 2. Chapter 3 discusses different approaches and schemes that are used in labelling XML documents; the advantages and disadvantages of each scheme are also highlighted.

Chapter 4 defines the underlying structure of the GroupBased scheme emphasising the necessity to form the GroupBased scheme's labels using two labels: namely, global and local labels. The global label is assigned to each group of nodes where

the nodes are grouped together based on the parent-child relationship (i.e. a node and its child nodes belong to the same group) whereas the local label is assigned based on the position of the node within a group. The chapter provided definitions and rules on how the scheme works in terms of its initial labelling process, its handling of insertions and its determination of different structural information before and after insertions.

The GroupBased scheme's practical design and implementation process are described in Chapter 5, along with some justification of various implementation choices. The 'Dynamic Dewey' labelling scheme (DDE) was chosen for use in forming the proposed scheme's labels due to its better performance compared to other similar schemes, as well as its simple implementation. It was necessary to implement this scheme, as no source code was available; it provided accurate and fair comparisons.

To test the scheme's design and implementation, four experiments were performed to test different aspects of the scheme's capability using different datasets. In addition, in order to evaluate the experiments, the GroupBased scheme and the DDE scheme were tested on both static and dynamic documents. The experimental framework is described in Chapter 6, along with the chosen datasets, while Chapter 7 describes the experimental results and their analysis.

The experiments and their results are evaluated in Chapter 8. The evaluation outcomes can be summarised by saying that the GroupBased scheme outperformed other schemes with similar objectives, especially the DDE scheme, in terms of efficient labelling time and the performance of insertions, offering faster and stable relationship determinations under static and dynamic circumstances, and faster query response times. Although the GroupBased scheme did not provide better memory allocation, which is justifiable due to the usage of two labels rather than one, it provided more stability under dynamic circumstances. The research contributions are discussed in the following section.

9.3 Research's Main Contributions

The present study aimed to discuss the problems associated with dynamic XML labelling schemes and proposed a new GroupBased scheme as a solution to those problems, specifically re-labelling and scheme performance in terms of initial labelling time, storage and query time. The following are the contributions that this study has made to the existing literature:

- A new perspective in labelling XML documents, based on grouping the nodes and using two labels, was proposed which suits both static and dynamic documents.
- The GroupBased scheme provides greater capability and stability in handling insertions by avoiding re-labelling.
- The GroupBased scheme provides faster labelling construction times in both types of document.
- The GroupBased scheme identifies all structural relationships faster.
- The GroupBased scheme provides better query performance.

Empirical evidence supported all these contributions.

9.4 How the Hypothesis is supported by the Outcomes

The research hypothesis (Ch.1) stated that the GroupBased labelling scheme could be applied with both dynamic and static XML documents without the need for re-labelling and with better performance in terms of time, labels' growth, identifying structural relationships, and with different classes of queries. As can be seen from the experimental results and the evaluation, the proposed scheme was implemented successfully and worked as intended. Four experiments were used to test the hypothesis using various sizes and structures of XML documents. The hypothesis was generally supported by reasonable results. The first experiment showed the positive relationships between the size of the document and the labels' construction time and size; this experiment also proved the scheme offered better performance with a deep tree structure rather than a wide structure. The second

experiment illustrated the efficiency and stability of the scheme in calculating all structural relationships before and after insertions. The third experiment evaluated the query performance before and after insertions using different classes of query. The fourth experiment showed the scheme's capability in handling different types of insertion by measuring the labelling time and the growth in the labels' size. The results demonestrated that the GroupBased scheme outperformed other similar schemes in terms of time which positively supported the hypothesis but which provided less support for the hypothesis in terms of size.

9.5 Further Research Developments and Future Directions

Although the GroupBased scheme outperforms similar existing schemes in many respects, it is not concise in terms of size, which indicates the need for further investigation. The novel idea in this thesis is that the GroupBased scheme is group-based and uses two labels instead of one. This allows the possibility of improvement, but results in a more complicated scheme that is difficult to implement compared to the simplicity of the DDE scheme. It is possible that some enhancement could be applied in terms of implementation and experimental aspects to achieve better performance.

From an implementation perspective, as argued in the previous chapter, using *'ArrayList'* to store the labels and *'DOM'* as the parser resulted in inefficient memory usage. Therefore, using other approaches might improve the performance of the current development.

From an experimental perspective, other datasets could be used in the experiments, as well as more complex queries, in order to obtain more comprehensive results. This would be useful in the evaluation process and would help in highlighting future work that could be carried out with regard to the technique, as well as in pinpointing limitations. Then, the identified limitations could be used to highlight what needs to be added to the technique in the future in order to reduce the limitations and enhance usability (Vlahavas *et al.*, 1999). Also,

obtaining more results would facilitate comparisons with other existing techniques.

Re-designing the GroupBased scheme using a different labelling scheme instead of the DDE scheme could improve efficiency or lead to new theory. Moreover, storing only one of the two labels and extrapolating the other when needed could result in improving the memory usage and the time required to calculate the label.

Investigating XML compression methods is the next direction to follow after this research, either to find a suitable compression technique that could be smoothly applied or to build a more suitable one that would preserve the scheme's characteristics and provide better performance.

As discussed in this thesis, finding a labelling scheme that resolves all of the issues is still a very challenging task and needs further investigation.

Lastly, as revealed by the limitations, it will be expected that future research directions will focus on ways in which the proposed scheme can address the issue of structural changes to XML documents. It is hoped that with such focus, the problem of re-labelling will be well addressed.

9.6 Finally

This research focused on dynamic XML labelling mechanisms. It developed a dynamic labelling scheme called the GroupBased labelling scheme. This new scheme provided efficient performance and proved itself to be efficient with regard to labelling and querying time; it was consistent in assuring unique labels and was dynamic in that it avoided the re-labelling of nodes in an updated intensive environment; it was also able to identify directly and stably all structural relationships. Finally, carrying out this research raises other questions and reveals other experiments worth investigating, as described in the previous section. These will need to be addressed in the future.

References

A., Gabillon. & M., Fansi. A Persistrent Labeling Scheme for XML and tree Database. In Proc. of ACI, , 2006.

ABD EL-AZIZ, A. & KANNAN, A. 2012. Storing Xml Document and Xml Policies in Relational Databases. *The International Conference on Computer Communication and Informatics (ICCCI) Coimbatore.* India: IEEE.

ABITEBOUL, S., BUNEMAN, P. & SUCIU, D. 2000. *Data on the Web: From relational to Semistructured Data and XML,* USA, Morgan Kaufmann.

ABITEBOUL, S., KAPLAN, H. & MILO, T. Compact labeling schemes for ancestor queries. Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms, 2001. Society for Industrial and Applied Mathematics, 547-556.

ADAMS, G. & SCHVANEVELDT, J. 2011. *Understanding Research Methods.,* New York, Longman.

AL-BADAWI, M. 2010. *A Performance Evaluation of a New Bitmap-Based Xml Processing Approach.* PhD, University of Sheffield.

AL-KHALIFA, S., JAGADISH, H., KOUDAS, N., PATEL, J. M., SRIVASTAVA, D. & WU, Y. Structural joins: A primitive for efficient XML query pattern matching. Data Engineering, 2002. Proceedings. 18th International Conference on, 2002. IEEE, 141-152.

ALSTRUP, S. & RAUHE, T. Improved labeling scheme for ancestor queries. Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms, 2002. Society for Industrial and Applied Mathematics, 947-953.

AMAGASA, T., YOSHIKAWA, M. & UEMURA, S. QRS: A robust numbering scheme for XML documents. Data Engineering, 2003. Proceedings. 19th International Conference on, 2003. IEEE, 705-707.

AMATO, G., DEBOLE, F., ZEZULA, P. & RABITTI, F. 2003. YAPI: Yet another path index for XML searching. *Research and Advanced Technology for Digital Libraries.* Springer.

ANDERSON, T. 2008. *Use an Xml Database in Php and Java Applications* [Online]. http://harisetiaji.files.wordpress.com/2009/05/db2-application-with-php-and-java.pdf: IBM. [Accessed 18-12-2013 2013].

APRIL, A. & ABRAN, A. 2012. *Software maintenance management: evaluation and continuous improvement*, John Wiley & Sons.

ARION, A., BONIFATI, A., COSTA, G., D'AGUANNO, S., MANOLESCU, I. & PUGLIESE, A. 2004. Efficient query evaluation over compressed XML data. *Advances in Database Technology-EDBT 2004.* Springer.

ARION, A., BONIFATI, A., MANOLESCU, I. & PUGLIESE, A. 2007. XQueC: A query-conscious compressed XML database. *ACM Transactions on Internet Technology (TOIT),* 7**,** 10.

ASSEFA, B. G. & ERGENC, B. 2012. OrderBased Labeling Scheme for Dynamic XML Query Processing. *Multidisciplinary Research and Practice for Information Systems.* Springer.

ATIQUE, M. & RAUT, A. 2012. A non redundant compact XML storage for efficient indexing and querying of XML documents. *Global Trends in Computing and Communication Systems.* Springer.

BALMIN, A., COLBY, L., CURTMOLA, E., LI, Q. & OZCAN, F. 2009. Search driven analysis of heterogenous XML data. *arXiv preprint arXiv:0909.1773*.

BARALIS, E., GARZA, P., QUINTARELLI, E. & TANCA, L. 2007. Answering XML queries by means of data summaries. *ACM Transactions on Information Systems (TOIS),* 25**,** 10.

BARBOSA, D. & BONIFATI, A. 2007. *Database and XML Technologies: 5th International XML Database Symposium, XSym 2007, Vienna, Austria, September 23-24, 2007, Proceedings*, Springer.

BARBOSA, D., MENDELZON, A., KEENLEYSIDE, J. & LYONS, K. ToXgene: a template-based data generator for XML. Proceedings of the 2002 ACM SIGMOD international conference on Management of data, 2002. ACM, 616-616.

BEHRENDS, E. 2007. *Evaluation of queries on linked distributed XML data.* University of Göttingen.

BELL, J. 2006. Doing Your Research Project: A Guide for First-time Researchers in Education, Health and Social Science-4/E.

BENJAMINI, Y. 2008 Opening the Box of a Boxplot. *The American Statistician,* 42 257–262.

BERGLUND, A., BOAG, S., CHAMBERLIN, D., FERNÁNDEZ, M., KAY, M., ROBIE, J. & SIMÉON, J. 2010a. *XML path language (XPath) 2.0* [Online]. http://www.w3.org/TR/xpath20/ W3C. [Accessed 01-04-2011 2011].

BERGLUND, A., BOAG, S., CHAMBERLIN, D., FERNÁNDEZ, M., KAY, M., ROBIE, J. & SIMÉON, J. 2010b. *Xml Path Language (Xpath) 2.0* [Online]. http://www.w3.org/TR/xpath20/: W3C. [Accessed 04-12-2013 2013].

BOAG, S., BERGLUND, A., CHAMBERLIN, D., SIMEON, J., KAY, M., ROBIE, J. & FERNANDEZ, M. 2007. XML path language (XPath) 2.0. *W3C, W3C Recommendation, Jan*.

BOAG, S., CHAMBERLIN, D., FERNANDEZ, M. F., FLORESCU, D., ROBIE, J. & SIMACON, J. 2011. *Xquery 1.0: An Xml Query Language* [Online]. http://www.w3.org/TR/xquery/: W3C. [Accessed 22-10-2013 2013].

BOAG, S., CHAMBERLIN, D., FERNÁNDEZ, M. F., FLORESCU, D., ROBIE, J., SIMÉON, J. & CONSORTIUM, W. C. W. W. W. 2003. XQuery 1.0: An XML Query Language, November 2003. *W3C Working Draft*.

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.* Springer.

BÖHME, T. & RAHM, E. 2003. Multi-User Evalaution of Xml Data Management Systems with Xmach~1. *VLDB 2002 Workshop EEXTT and CAiSE.* London, UK.

BOSAK, J. & BRAY, T. 1999. XML and the second-generation Web. *Scientific American,* 280**,** 89-93.

BOUGANIM, L., NGOC, F. D. & PUCHERAL, P. Client-based access control management for XML documents. Proceedings of the Thirtieth

international conference on Very large data bases-Volume 30, 2004. VLDB Endowment, 84-95.

BOURRET, R. 2005. *Xml and Databases* [Online]. http://www.rpbourret.com/xml/XMLAndDatabases.htm. [Accessed 22-03-2011 2011].

BRAY, T., PAOLI, J., SPERBERG-MCQUEEN, C., MALER, E. & YERGEAU, F. 2008. Extensible Markup Language (XML) 1.0 , W3C 2008. *URL* http://www. *w3. org/TR/xml*.

BRENES, S., WU, Y., VAN GUCHT, D. & SANTA CRUZ, P. Trie Indexes for Efficient XML Query Evaluation. WebDB, 2008. Citeseer.

BROWNELL, D. & MEGGINSON, D. 2002. SAX: Simple API for XML. SAX Project Organization.

BRUNO, N., KOUDAS, N. & SRIVASTAVA, D. Holistic twig joins: optimal XML pattern matching. Proceedings of the 2002 ACM SIGMOD international conference on Management of data, 2002. ACM, 310-321.

CAMERON, D. 2008. *How Xquery Extends Xpath* [Online]. http://www.ibm.com/developerworks/xml/library/x-queryxpath/index.html: IBM. [Accessed 26-10-2013 2013].

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**,** 414-426.

CATANIA, B., OOI, B. C., WANG, W. & WANG, X. Lazy XML updates: laziness as a virtue, of update and structural join efficiency. Proceedings of the 2005 ACM SIGMOD international conference on Management of data, 2005. ACM, 515-526.

CHAMPION, M. 2001. Storing XML in databases. *eAI journal,* 10**,** 53-55.

CHANG, Y.-H., WU, C.-Y. & LO, C.-C. 2012. Processing xml queries with structural and full-text constraints. *Journal of Information Science and Engineering,* 28**,** 221-242.

CHASE, N. 2003. *Validating Xml* [Online]. http://www.ibm.com/developerworks/xml/tutorials/x-valid/section4.html: IBM. [Accessed 12-12-2012 2012].

CHEN, S., LI, H.-G., TATEMURA, J., HSIUNG, W.-P., AGRAWAL, D. & CANDAN, K. S. Twig 2 Stack: bottom-up processing of generalized-tree-pattern queries over XML documents. Proceedings of the 32nd international conference on Very large data bases, 2006. VLDB Endowment, 283-294.

CHEN, T., LU, J. & LING, T. W. On boosting holism in XML twig pattern matching using structural indexing techniques. Proceedings of the 2005 ACM SIGMOD international conference on Management of data, 2005. ACM, 455-466.

CHOI, H., LEE, K.-H. & LEE, Y.-J. 2014. Parallel labeling of massive XML data with MapReduce. *The Journal of Supercomputing,* 67**,** 408-437.

CHOI, R. H. & WONG, R. K. 2014. VXQ: A visual query language for XML data. *Information Systems Frontiers***,** 1-21.

CHUNG, C.-W., MIN, J.-K. & SHIM, K. APEX: An adaptive path index for XML data. Proceedings of the 2002 ACM SIGMOD international conference on Management of data, 2002. ACM, 121-132.

CLARK, M., RILEY, M., SZIVAS, E., WILKIE, E. & WOOD, R. 2000. Researching and writing dissertations in business and management. London: Thomson Learning.

CLEMENTS, P., KAZMAN, R. & KLEIN, M. 2003. *Evaluating software architectures*, 清华大学出版社.

COHEN, E., KAPLAN, H. & MILO, T. 2010. Labeling dynamic XML trees. *SIAM Journal on Computing,* 39**,** 2048-2074.

COLLIS, J., HUSSEY, R., CROWTHER, D., LANCASTER, G., SAUNDERS, M., LEWIS, P., THORNHILL, A., BRYMAN, A., BELL, E. & GILL, J. 2003. Business research methods. Palgrave Macmillan, New York.

CONNOLLY, T. M. & BEGG, C. E. 2005. *Database systems: a practical approach to design, implementation, and management*, Pearson Education.

COOPER, B. F., SAMPLE, N., FRANKLIN, M. J., HJALTASON, G. R. & SHADMON, M. A fast index for semistructured data. VLDB, 2001. 341-350.

COOPER, H. 2008. *Synthesizing Research: A Guide for Literature Reviews.* , Harlow: Pearson Education Limited.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L. & STEIN, C. 2001. *Introduction to algorithms*, MIT press Cambridge.

CRESWELL, J. 2007 *Review of the Literature*, Thousand Oaks: Sage Publications.

CRESWELL, J. W. & CLARK, V. L. P. 2007. Designing and conducting mixed methods research.

CUNNINGHAM, L. A. 2006. Language, Deals, and Standards: The Future of XML Contracts. *Wash. UL Rev.,* 84**,** 313.

DALE, N. B., JOYCE, D. T. & WEEMS, C. 2012. *Object-oriented data structures using Java*, Sudbury, MA, Jones & Bartlett Learning.

DARUGAR, P. 2000. *Dare to Script Tree-Based Xml with Perl* [Online]. http://www.ibm.com/developerworks/xml/library/xmlperl2/index.html: IBM. 12-12-2012].

DAVIS, K. C., ZHAN, Y. & DAVIS, R. B. An XML/XPath query language and XMark performance study. Applications and the Internet, 2003. Proceedings. 2003 Symposium on, 2003. IEEE, 422-427.

DBLP. 2013. *DBLP: Computer Science Bibliography* [Online]. http://dblp.uni-trier.de/db/: DBLP. [Accessed 20-12-2013 2013].

DELLINGER, A. B. & LEECH, N. L. 2007. Toward a unified validation framework in mixed methods research. *Journal of Mixed Methods Research,* 1**,** 309-332.

DENG, Z. H., XIANG, Y. Q. & GAO, N. 2013. LAF: a new XML encoding and indexing strategy for keyword‐based XML search. *Concurrency and Computation: Practice and Experience,* 25**,** 1604-1621.

DIETZ, P. F. Maintaining order in a linked list. Proceedings of the fourteenth annual ACM symposium on Theory of computing, 1982. ACM, 122-127.

DIRIWÄCHTER, R. & VALSINER, J. Qualitative developmental research methods in their historical and epistemological contexts. Forum Qualitative Sozialforschung/Forum: Qualitative Social Research, 2006.

DOCUMENTATION, O. 2014. *ArrayList* [Online]. http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html: Oracle. [Accessed 02-01-2014 2014].

DROZDEK, A. 2004. *Data structures and algorithms in Java*, Cengage Learning.

DUAN, S., KEMENTSIETSIDIS, A., SRINIVAS, K. & UDREA, O. Apples and oranges: a comparison of RDF benchmarks and real RDF datasets. Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, 2011. ACM, 145-156.

DUONG, M. & ZHANG, Y. LSDX: a new labelling scheme for dynamically updating XML data. Proceedings of the 16th Australasian database conference-Volume 39, 2005. Australian Computer Society, Inc., 185-193.

DUONG, M. & ZHANG, Y. 2008. Dynamic Labelling Scheme for XML Data Processing. *On the Move to Meaningful Internet Systems: OTM 2008.* Springer.

EDA, T., SAKURAI, Y., AMAGASA, T., YOSHIKAWA, M., UEMURA, S. & HONISHI, T. Dynamic range labeling for XML trees. Current Trends in Database Technology-EDBT 2004 Workshops, 2005. Springer, 230-239.

ELMASRI, R. 2008. *Fundamentals of database systems*, Pearson Education India.

ERIKSEN, L. 2004. Xml Parsing with Sax and Dom: A Code Comparision. *The Australia Open Source Developers' Confrence (OSDC2004).* Melboyrne, Australia.

FALLSIDE, D. C. & WALMSLEY, P. 2004. XML schema part 0: primer second edition. *W3C recommendation*, 16.

FAROOQ, S. & QUADRI, S. 2011. Evaluating Effectiveness of Software Testing Techniques With Emphasis on Enhancing Software Reliability. *Journal of Emerging Trends in Computing and Information Sciences,* 2, 740-745.

FENNELL, P. 2013. Extremes of XML. *XML LONDON 2013*.

FIEBIG, T., HELMER, S., KANNE, C., MOERKOTTE, G., NEUMANN, J. & SCHIELE, R. 2002. Anatomy of a Native Xml Base Managment System. *VLDB*, 292-314.

FISHER, D. K., LAM, F., SHUI, M., W. & WONG, R. K. Dynamic labeling schemes for ordered XML based on type information Proceedings of the 17th Australasian Database Conference, 2006 Australia. Australian Computer Society, Inc., 7-12.

FISHER, D. K., LAM, F., SHUI, W. M. & WONG, R. K. Dynamic labeling schemes for ordered XML based on type information. Proceedings of the 17th Australasian Database Conference-Volume 49, 2006. Australian Computer Society, Inc., 59-68.

FLANAGAN, D. 2005. *Java in a Nutshell*, " O'Reilly Media, Inc.".

FRANCESCHET, M. 2005. XPathMark: an XPath benchmark for the XMark generated data. *Database and XML Technologies.* Springer.

FRANK, T., APEL, A. & SCHAEBEC, H. Web integration of gOcad using a 3d-Xml application server. Proceedings of the 23rd Gocad Meeting, Nancy, France, June, 2003. 10-11.

FRIGGE, M., HOAGLIN, D. C. & IGLEWICZ, B. 2009. Some implementations of the boxplot. *The American Statistician,* 43**,** 50-54.

GABILLON, A. & FANSI, M. A persistent labelling scheme for XML and tree databases. SITIS, 2005. 110-115.

GHAURI, P. N. & GRØNHAUG, K. 2005. *Research methods in business studies: A practical guide*, Pearson Education.

GILL, J. & JOHNSON, P. 2010. *Research methods for managers*, Sage.

GIVEN, L. M. 2008. *The Sage encyclopedia of qualitative research methods*, Sage Publications.

GOLDMAN, R. & WIDOM, J. 1997. Dataguides: Enabling query formulation and optimization in semistructured databases.

GOU, G. & CHIRKOVA, R. 2007. Efficiently querying large XML data repositories: A survey. *Knowledge and Data Engineering, IEEE Transactions on,* 19**,** 1381-1403.

GREEN, B. N., JOHNSON, C. D. & ADAMS, A. 2006. Writing narrative literature reviews for peer-reviewed journals: secrets of the trade. *Journal of Chiropractic Medicine,* 5**,** 101-117.

GREEN, T. J., MIKLAU, G., ONIZUKA, M. & SUCIU, D. 2003. Processing XML streams with deterministic automata. *Database Theory—ICDT 2003.* Springer.

GROPPE, J. 2008. *SPEEDING UP XML QUERYING.* Citeseer.

GULHANE, V. & ALI, M. Partial Query Processor For Compressed Xml. International Journal of Engineering Research and Technology, 2013. ESRSA Publications.

GUSFIELD, D. 1997. *Algorithms on strings, trees and sequences: computer science and computational biology*, Cambridge university press.

HAKIM, C. 2000. *Research design: successful designs for social and economic research*, Psychology Press.

HAMMAWA, M. & SAMPSON, G. 2011. Applying Data Mining Research Methodologies on Information Systems.

HÄRDER, T., HAUSTEIN, M., MATHIS, C. & WAGNER, M. 2007. Node labeling schemes for dynamic XML documents reconsidered. *Data & Knowledge Engineering,* 60**,** 126-149.

HÄRDER, T. & MATHIS, C. 2010. Key concepts for native XML processing. *From active data management to event-based systems and more.* Springer.

HARDY, M. A. & BRYMAN, A. 2004. *Handbook of data analysis*, Sage.

HAROLD, E. 2005. *Managing Xml Data: Native Xml Databases Theory and Reality.* [Online]. http://www.ibm.com/developerworks/xml/library/x-mxd4.html: IBM. [Accessed 13-06-2011 2011].

HAROLD, E. R. 2002. *Processing XML with Java*, Addison-Wesley Longman Publishing Co., Inc.

HAROLD, E. R. 2004. *Effective XML: 50 specific ways to improve Your XML*, Addison-Wesley Professional.

HAROLD, E. R., MEANS, W. S. & UDEMADU, K. 2004. *XML in a Nutshell*, O'reilly Sebastopol, CA.

HART, C. 2008. Searching and Reviewing the Literature and Information Skills. *The Postgraduate's Companion***,** 162.

HAUSTEIN, M. & HÄRDER, T. 2007. An efficient infrastructure for native transactional XML processing. *Data & Knowledge Engineering,* 61**,** 500-523.

HAW, S.-C. & LEE, C.-S. 2009. Extending path summary and region encoding for efficient structural query processing in native XML databases. *Journal of Systems and Software,* 82**,** 1025-1035.

HE, H. & YANG, J. Multiresolution indexing of XML for frequent queries. Data Engineering, 2004. Proceedings. 20th International Conference on, 2004. IEEE, 683-694.

HÉGARET, P., WHITMER, R. & WOOD, L. 2005. *Document Object Model (Dom)* [Online]. http://www.w3.org/DOM/ [Accessed 25-12-2013 2013].

HORSTMANN, C. S. & CORNELL, G. 2002. *Core Java 2: Volume I, Fundamentals*, Pearson Education.

HOU, J., ZHANG, Y. & KAMBAYASHI, Y. Object-oriented representation for XML data. Cooperative Database Systems for Advanced Applications, 2001. CODAS 2001. The Proceedings of the Third International Symposium on, 2001. IEEE, 40-49.

HUNTER, D., CAGLE, K. & DIX, C. 2007. Beginning XML: XML Schemas, SOAP, XSLT, DOM, and SAX 2.0. Wrox Press.

HUNTER, L. & LEAHEY, E. 2008. Collaborative research in sociology: Trends and contributing factors. *The American Sociologist,* 39**,** 290-306.

IDRIS, N. 1999. Should I use SAX or DOM.

JEBREEN, I. 2012. Using Inductive Approach as Research Strategy in Requirements Engineering. *International Journal of Computer and Information Technology,* 1**,** 162-173.

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.

JITTRAWONG, K. & WONG, R. K. Optimizing XPath queries on streaming XML data. Proceedings of the eighteenth conference on Australasian database-Volume 63, 2007. Australian Computer Society, Inc., 73-82.

JOHNSON, J., MILLER, A., KHAN, L. & THURAISINGHAM, B. Extracting semantic information structures from free text law enforcement data. Intelligence and Security Informatics (ISI), 2012 IEEE International Conference on, 2012. IEEE, 177-179.

JONGE, A. 2008. *Comparing Xml Database Approaches* [Online]. http://www.ibm.com/developerworks/library/x-comparexmldb/: IBM. [Accessed 05-11-2012 2012].

K., S. 2006 *Making Mistakes with XML,* Texas, Addison-Wesley.

KAPLAN, H., MILO, T. & SHABO, R. A comparison of labeling schemes for ancestor queries. Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms, 2002. Society for Industrial and Applied Mathematics, 954-963.

KASIM, R., ALEXANDER, K., HUDSON, J. & 2010 *A choice of research strategy for identifying community-based action skill requirements in the process of*

delivering housing market renewal., Research Institute for the Built and Human Environment, University of Salford, UK.

KAUSHIK, R., BOHANNON, P., NAUGHTON, J. F. & KORTH, H. F. Covering indexes for branching path queries. Proceedings of the 2002 ACM SIGMOD international conference on Management of data, 2002a. ACM, 133-144.

KAUSHIK, R., SHENOY, P., BOHANNON, P. & GUDES, E. Exploiting local similarity for indexing paths in graph-structured data. Data Engineering, 2002. Proceedings. 18th International Conference on, 2002b. IEEE, 129-140.

KHA, D. D., YOSHIKAWA, M. & UEMURA, S. A structural numbering scheme for XML data. XML-Based Data Management and Multimedia Engineering—EDBT 2002 Workshops, 2002. Springer, 91-108.

KHAN, W. & ULLAH, H. 2010. Scientific Reasoning: A Solution to the Problem of Induction. *International Journal of Basic & Applied Sciences,* 10.

KOCHMER, C. & FRANDSEN, E. 2002. *JSP and XML: From Web Services to XML in Your JSP Application*, Addison-Wesley Longman Publishing Co., Inc.

LAWRENCE, R. 2004. The space efficiency of XML. *Information and Software Technology,* 46**,** 753-759.

LEE, D. & CHU, W. 2000. Comparative Analysis of Six Xml Schema Languages. *ACM SIGMOD Record***,** 76-87.

LEE, K.-H., WHANG, K.-Y., HAN, W.-S. & KIM, M.-S. 2010. Structural consistency: enabling XML keyword search to eliminate spurious results consistently. *The VLDB Journal,* 19**,** 503-529.

LEWIS, J. & CHASE, J. 2010. *Java software structures: designing and using data structures,* New York, Addison-Wesley.

LI, C. & LING, T. W. An improved prefix labeling scheme: A binary string approach for dynamic ordered XML. Database Systems for Advanced Applications, 2005a. Springer, 125-137.

LI, C. & LING, T. W. QED: a novel quaternary encoding to completely avoid re-labeling in XML updates. Proceedings of the 14th ACM international conference on Information and knowledge management, 2005b. ACM, 501-508.

LI, C., LING, T. W. & HU, M. Efficient processing of updates in dynamic XML data. Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on, 2006a. IEEE, 13-13.

LI, C., LING, T. W. & HU, M. Reuse or never reuse the deleted labels in XML query processing based on labeling schemes. Database Systems for Advanced Applications, 2006b. Springer, 659-673.

LI, C., LING, T. W. & HU, M. 2008. Efficient updates in dynamic XML data: from binary string to quaternary string. *The VLDB Journal—The International Journal on Very Large Data Bases,* 17**,** 573-601.

LI, C., LING, T. W., LU, J. & YU, T. On reducing redundancy and improving efficiency of XML labeling schemes. Proceedings of the 14th ACM international conference on Information and knowledge management, 2005. ACM, 225-226.

LI, G., FENG, J., WANG, J. & ZHOU, L. Effective keyword search for valuable lcas over xml documents. Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, 2007. ACM, 31-40.

LI, M., JIANG, Q., TAN, C.-H. & WEI, K.-K. 2014. Enhancing User-Game Engagement Through Software Gaming Elements. *Journal of Management Information Systems,* 30**,** 115-150.

LI, Q. & MOON, B. Indexing and querying XML data for regular path expressions. VLDB, 2001. 361-370.

LI, Y. 2003. *TheXoo7Benchmark* [Online]. http://www.comp.nus.edu.sg/~ebh/XOO7.html. [Accessed 01-12-2013 2013].

LIEFKE, H. & SUCIU, D. XMill: an efficient compressor for XML data. ACM Sigmod Record, 2000. ACM, 153-164.

LIU, J., MA, Z. & YAN, L. Efficient processing of twig pattern matching in fuzzy XML. Proceedings of the 18th ACM conference on Information and knowledge management, 2009. ACM, 117-126.

LIU, J., MA, Z. & YAN, L. 2013. Efficient labeling scheme for dynamic XML trees. *Information Sciences,* 221**,** 338-354.

LOESER, H., NICOLA, M. & FITZGERALD, J. Index Challenges in Native XML Database Systems. BTW, 2009. Citeseer, 508-525.

LORENZ, R. C., KRÜGER, J. K., NEUMANN, B., SCHOTT, B. H., KAUFMANN, C., HEINZ, A. & WÜSTENBERG, T. 2013. Cue reactivity and its inhibition in pathological computer game players. *Addiction biology,* 18**,** 134-146.

LU, J. 2013. XML Labeling Scheme. *An Introduction to XML Query Processing and Keyword Search.* Springer.

LU, J., LING, T. W., CHAN, C.-Y. & CHEN, T. From region encoding to extended dewey: On efficient processing of XML twig pattern matching. Proceedings of the 31st international conference on Very large data bases, 2005. VLDB Endowment, 193-204.

LUO, C., JIANG, Z., HOU, W.-C., YU, F. & ZHU, Q. A sampling approach for XML query selectivity estimation. Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, 2009. ACM, 335-344.

M., M., D., K. & C., L. 2009 Internet Drafts: XML Media Types *Internet Engineering Task Force,* 2**,** 75-89.

MA, Z. & YAN, L. 2010. *Soft Computing in XML Data Management: Intelligent Systems from Decision Making to Data Mining, Web Intelligence and Computer Vision*, Springer.

MATHA, M. P. 2011. *Core Java: a comprehensive study,* New Delhi, PHI Learning.

MCCREIGHT, E. M. 1976. A space-economical suffix tree construction algorithm. *Journal of the ACM (JACM),* 23**,** 262-272.

MCGILL, R., TUKEY, J. W. & LARSEN, W. A. 1978. Variations of box plots. *The American Statistician,* 32**,** 12-16.

MEGGINSON, D. 2001. Sax 2.0: The simple api for xml. *SAX project*.

MEHEUS, J. & NICKLES, T. 2009. *Models of discovery and creativity*, Springer.

MERIALDO, P. 1999. *Acm Sigmod Record: Xml Version* [Online]. http://www.dia.uniroma3.it/Araneus/Sigmod/ [Accessed 22-02-2013 2013].

MESITI, W. L. M., TZITZIKAS, C. T. Y. & VAKALI, A. 2004. Current Trends in Database Technology–EDBT 2004 Workshops.

MEUSS, H. & STROHMAIER, C. M. Improving Index Structures for Structured Document Retrieval. BCS-IRSG Annual Colloquium on IR Research, 1999. Citeseer.

MILO, T. & SUCIU, D. 1999. Index structures for path expressions. *Database Theory—ICDT'99.* Springer.

MIN, J.-K., LEE, J. & CHUNG, C.-W. 2009. An efficient XML encoding and labeling method for query processing and updating on dynamic XML data. *Journal of Systems and Software,* 82**,** 503-515.

MIRABI, M., IBRAHIM, H., MAMAT, A., UDZIR, N. I. & FATHI, L. 2010. Controlling label size increment of efficient XML encoding and labeling scheme in dynamic XML update. *Journal of Computer Science,* 6**,** 1535.

MLÝNKOVÁ, I. 2008. Xml benchmarking: Limitations and opportunities. Technical Report, Department of Software Engineering, Charles University, Czech Republic.

MOGHADDAM, G. G. & MOBALLEGHI, M. 2008. How do we measure the use of scientific journals? A note on research methodologies. *Scientometrics,* 76**,** 125-133.

MOLINA, H., ULLMAN, J. & WIDOM, J. 2009. *Database systems: the complete book,* USA, Pearson Education.

MURATA, M., LAURENT, S. S. & KOHN, D. 2001. XML media types. *RFC3023, January*.

NAFTALIN, M. & WADLER, P. 2006. *Java generics and collections*, " O'Reilly Media, Inc.".

NASA. 2001. *Gsfc Open Source Softwere* [Online]. http://developerlife.com/tutorials/?p=28. [Accessed 22-03-2014 2014].

NICOLA, M., KOGAN, I. & SCHIEFER, B. An XML transaction processing benchmark. Proceedings of the 2007 ACM SIGMOD international conference on Management of data, 2007. ACM, 937-948.

NOAMAN, A. Y. & AL MANSOUR, A. A. 2012. A Comparative Study Between Two Types of Database Management Systems: XML-Enabled Relational and Native XML. *World Applied Sciences Journal,* 19**,** 972-985.

NOAMAN, A. Y. & ALMANSOUR, A. 2012. Towards Achieving an Optimum Performanceof XML Data into Both Types of XML Databases: XML-Enabled Databases and Native XML Databases. *Middle-East Journal of Scientific Research,* 12**,** 182-194.

NOLAN, D. & LANG, D. T. 2014. *XML and Web Technologies for Data Sciences with R*, Springer.

O'NEIL, P., O'NEIL, E., PAL, S., CSERI, I., SCHALLER, G. & WESTBURY, N. ORDPATHs: insert-friendly XML node labels. Proceedings of the 2004 ACM SIGMOD international conference on Management of data, 2004. ACM, 903-908.

O'LEARY, Z. 2006 Researching Real-World Problems – A Guide to Methods of Inquiry. T. *housand Oaks: SAGE.*

OGBUJI, U. 2004. A hands-on introduction to Schematron. IBM.

OLTEANU, D. 2005. *Evaluation of XPath queries against XML streams.* lmu.

ONIZUKA, M. Light-weight XPath processing of XML stream with deterministic automata. Proceedings of the twelfth international conference on Information and knowledge management, 2003. ACM, 342-349.

OQBUJI, U. 2004a. *A Hands-on Introduction to Schematron* [Online]. http://www.ibm.com/developerworks/xml/tutorials/x-schematron/section2.html IBM. [Accessed 22-11-2012 2012].

OQBUJI, U. 2004b. *A Survey of Xml Standards* [Online]. http://www.ibm.com/developerworks/xml/library/x-stand1/index.html: IBM. 2012].

PALANI, G. 2011. *Investigate Current Xml Tools* [Online]. http://www.ibm.com/developerworks/xml/library/x-xmltools/index.html: IBM. [Accessed 10-11-2012 2012].

PAPAMARKOS, G., ZAMBOULIS, L. & POULOVASSILIS, A. 2009. Xml Databases. London, UK: School of Computer Science and Information Systems.

PERLIS, A. J., SAYWARD, F. & SHAW, M. 1981. *Software metrics: an analysis and evaluation*, MIT Press.

POWELL, G. 2007. *Beginning XML databases*, John Wiley & Sons.

PROJECT, S. 2013a. *About SAX* [Online]. http://www.saxproject.org: SAX Project. [Accessed 23-03-2013 2013].

PROJECT, S. 2013b. *Events vs Trees* [Online]. http://www.saxproject.org/event.html: SAX Project. [Accessed 24-03-2013 2013].

QIN, Z. Y., TANG, Y. & XU, H. Z. 2012. A string approach for dynamic XML document. *Applied Mechanics and Materials,* 220**,** 2512-2519.

RADIYA, A. & DIXIT, V. 2000. *The Basics of Using Xml Schema to Define Elements* [Online]. http://www.ibm.com/developerworks/xml/library/xml-schema/index.html: IBM. [Accessed 02-12-2012 2012].

RAFIEI, D., MOISE, D. L. & SUN, D. Finding syntactic similarities between xml documents. Database and Expert Systems Applications, 2006. DEXA'06. 17th International Workshop on, 2006. IEEE, 512-516.

RAO, P. & MOON, B. PRIX: Indexing and querying XML using prufer sequences. Data Engineering, 2004. Proceedings. 20th International Conference on, 2004. IEEE, 288-299.

RAY, E. T. 2003. *learning XML*, " O'Reilly Media, Inc.".

REMENYI, D. 1998. *Doing research in business and management: an introduction to process and method*, Sage.

RIDLEY, D. 2012. *The literature review: A step-by-step guide for students*, Sage.

ROBSON, C. 2011. *Real world research: a resource for users of social research methods in applied settings*, Wiley Chichester.

ROUSSEEUW, P. J., RUTS, I. & AND TUKEY, J. W. 1999. The Bagplot: A Bivariate Boxplot *The American Statistician* 53 382–387.

RUNAPONGSA, K., PATEL, J. M., JAGADISH, H., CHEN, Y. & AL-KHALIFA, S. 2006a. The Michigan benchmark: towards XML query performance diagnostics. *Information Systems,* 31**,** 73-97.

RUNAPONGSA, K., PATEL, M., JAGADISH, H., CHEN, Y. & S., A.-K. 2006b. *Michigan Benchmark* [Online]. http://www.eecs.umich.edu/db/mbench/description.html [Accessed 22-12-2013 2013].

RUNAPONGSA, K., PATEL, M., JAGADISH, H., CHEN, Y. S. & A.-K. 2006c. *The Michigan Benchmark* [Online].

http://www.eecs.umich.edu/db/mbench/description.html. [Accessed 22-01-2014 2014].

SAKR, S. 2009. XML compression techniques: A survey and comparison. *Journal of Computer and System Sciences,* 75**,** 303-322.

SANGHERA, P. 2006. *SCJP Exam for J2SE 5: A Concise and Comprehensive Study Guide for the Sun Certified Java Programmer Exam*, Apress.

SANS, V. & LAURENT, D. 2008. Prefix based numbering schemes for XML: techniques, applications and performances. *Proceedings of the VLDB Endowment,* 1**,** 1564-1573.

SAPSFORD, R. & JUPP, V. 2006. *Data collection and analysis*, Sage.

SAUNDERS, M. N., SAUNDERS, M., LEWIS, P. & THORNHILL, A. 2011. *Research methods for business students, 5/e*, Pearson Education India.

SCHILDT, H. 2006. *Java The Complete Reference, (Osborne Complete Reference Series)*, McGraw-Hill Osborne Media.

SCHMIDT, A. 2003. *Xmark* [Online]. http://www.xml-benchmark.org/. [Accessed 11-11-2013 2013].

SCHMIDT, A., WAAS, F., KERSTEN, M., CAREY, M. J., MANOLESCU, I. & BUSSE, R. XMark: A benchmark for XML data management. Proceedings of the 28th international conference on Very Large Data Bases, 2002. VLDB Endowment, 974-985.

SCHMIDT, A., WAAS, F., KERSTEN, M., FLORESCU, D., CAREY, M. J., MANOLESCU, I. & BUSSE, R. 2001. Why and how to benchmark XML databases. *ACM SIGMOD Record,* 30**,** 27-32.

SEAN, K. 2006. *Making Mistakes with XML,* Texas USA Addison-Wesley.

SENELLART, P. & SOUIHLI, A. Un système de gestion de données XML probabilistes. Proc. BDA, 2010. Citeseer.

SHAH, B., RAO, P. R., MOON, B. & RAJAGOPALAN, M. 2009. A data parallel algorithm for XML DOM parsing. *Database and XML Technologies.* Springer.

SHEN, H. T., PEI, J., ÖZSU, M. T., ZOU, L., LU, J., LING, T. W., YU, G., ZHUANG, Y. & SHAO, J. 2010. *Web-Age Information Management. WAIM 2010 Workshops: WAIM 2010 International Workshops: IWGD 2010, WCMT 2010, XMLDM*

2010, Jiuzhaigou Valley, China, July 15-17, 2010, Revised Selected Papers, Springer.

SIKORA, M. 2003. *Java: Practical Guide for Programmers*, Morgan Kaufmann.

SILBERSTEIN, A., HE, H., YI, K. & YANG, J. BOXes: Efficient maintenance of order-based labeling for dynamic XML data. Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on, 2005. IEEE, 285-296.

SILVASTI, P., SIPPU, S. & SOISALON-SOININEN, E. Schema-conscious filtering of XML documents. Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, 2009. ACM, 970-981.

SJOBERG, D. I., ANDA, B., ARISHOLM, E., DYBA, T., JORGENSEN, M., KARAHASANOVIC, A., KOREN, E. F. & VOKÁC, M. Conducting realistic experiments in software engineering. Empirical Software Engineering, 2002. Proceedings. 2002 International Symposium n, 2002. IEEE, 17-26.

SPELL, B. 2005. *Pro Java Programming*, Springer.

STADLER, F. 2004. *Induction and Deduction in the Sciences*, Springer.

STAKEN, K. 2001. *Introduction to Native Xml Databases* [Online]. http://www.xml.com/pub/a/2001/10/31/nativexmldb.html: O'Reilly. [Accessed 22-09-2011 2011].

STEEDMAN, M., OSBORNE, M., SARKAR, A., CLARK, S., HWA, R., HOCKENMAIER, J., RUHLEN, P., BAKER, S. & CRIM, J. Bootstrapping statistical parsers from small datasets. Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1, 2003. Association for Computational Linguistics, 331-338.

STEEGMANS, B. 2004. *XML for DB2 Information Integration*, IBM Corporation, International Technical Support Organization.

SUCIU, D. 2002. *Xml Data Repository* [Online]. http://www.cs.washington.edu/research/xmldatasets/: University of Washington. [Accessed 10-01-2014 2014].

SUN, C., CHAN, C.-Y. & GOENKA, A. K. Multiway slca-based keyword search in xml data. Proceedings of the 16th international conference on World Wide Web, 2007. ACM, 1043-1052.

SUN, L. & WANG, H. 2012. A purpose-based access control in native XML databases. *Concurrency and Computation: Practice and Experience,* 24**,** 1154-1166.

TATARINOV, I., VIGLAS, S. D., BEYER, K., SHANMUGASUNDARAM, J., SHEKITA, E. & ZHANG, C. Storing and querying ordered XML using a relational database system. Proceedings of the 2002 ACM SIGMOD international conference on Management of data, 2002. ACM, 204-215.

TEOREY, T. J., LIGHTSTONE, S. S., NADEAU, T. & JAGADISH, H. 2011. *Database modeling and design: logical design*, Elsevier.

THIMMA, M., TSUI, T. K. & LUO, B. HyXAC: a hybrid approach for XML access control. Proceedings of the 18th ACM symposium on Access control models and technologies, 2013. ACM, 113-124.

THONANGI, R. A Concise Labeling Scheme for XML Data. COMAD, 2006. 4-14.

TIAN, D. & GEORGANAS, N. D. A coverage-preserving node scheduling scheme for large wireless sensor networks. Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, 2002. ACM, 32-41.

TIDWELL, D. 2002. *Introducti on to Xml* [Online]. http://www.ibm.com/developerworks/xml/tutorials/xmlintro/section2.html: IBM. [Accessed 11-11-2012 2012].

TREEBANK. 1999. *The Penn Treebank Project* [Online]. http://www.cis.upenn.edu/~treebank/. [Accessed 29-01-2014 2014].

UNIPORT. 2014. *Swiss-Port* [Online]. http://www.uniprot.org: UniPort. [Accessed 22-03-2014 2014].

VAKALI, A., CATANIA, B. & MADDALENA, A. 2005. XML data stores: emerging practices. *Internet Computing, IEEE,* 9**,** 62-69.

VLAHAVAS, I., STAMELOS, I., REFANIDIS, I. & TSOUKIÀS, A. 1999. ESSE: an expert system for software evaluation. *Knowledge-based systems,* 12**,** 183-197.

W3C. 2010. *What Is Xml?* [Online]. http://www.w3.org/standards/xml/core [Accessed 10-12-2011 2011].

W3SCHOOLS. 2013a. *DTD Tutorial* [Online]. http://www.w3schools.com/DTD/: W3schools. [Accessed 05-01-2013 2013].

W3SCHOOLS. 2013b. *Intoduction to XML Schema* [Online]. http://www.w3schools.com/schema/schema_intro.asp: W3schools.

W3SCHOOLS. 2013c. *Introduction to DTD* [Online]. http://www.w3schools.com/dtd/dtd_intro.asp: W3schools. [Accessed 03--1-2013 2013].

W3SCHOOLS. 2013d. *Introduction to Xml* [Online]. http://www.w3schools.com/xml/xml_whatis.asp. [Accessed 22-10-2011 2011].

W3SCHOOLS. 2013e. *Introduction to XQuery* [Online]. http://www.w3schools.com/xquery/xquery_intro.asp: W3schools. [Accessed 22-01-2013 2013].

W3SCHOOLS. 2013f. *XML Attributes* [Online]. http://www.w3schools.com/xml/xml_attributes.asp: W3schools. [Accessed 20-01-2013 2013].

W3SCHOOLS. 2013g. *XML DOM - Properties and Methods* [Online]. http://www.w3schools.com/dom/dom_methods.asp: W3schools. [Accessed 25-02-2013 2013].

W3SCHOOLS. 2013h. *XML DOM Nodes* [Online]. http://www.w3schools.com/dom/dom_nodes.asp: W3schools. [Accessed 20-02-2013 2013].

W3SCHOOLS. 2013i. *XML Elements* [Online]. http://www.w3schools.com/xml/xml_elements.asp: W3schools. [Accessed 22-1-2013 2013].

W3SCHOOLS. 2013j. *XML Syntax Rules* [Online]. http://www.w3schools.com/xml/xml_syntax.asp: W3schools. [Accessed 22-03-2013 2013].

W3SCHOOLS. 2013k. *Xml Tree* [Online]. http://www.w3schools.com/xml/xml_tree.asp: W3schools. [Accessed 03-01-2013 2013].

W3SCHOOLS. 2013l. *XPath Axes* [Online]. http://www.w3schools.com/xpath/xpath_axes.asp: W3schools. [Accessed 13-01-2013 2013].

W3SCHOOLS. 2013m. *XPath Introduction* [Online]. http://www.w3schools.com/xpath/xpath_intro.asp: W3schools. [Accessed 13-01-2013 2013].

W3SCHOOLS. 2013n. *XPath Nodes* [Online]. http://www.w3schools.com/xpath/xpath_nodes.asp: W3schools. [Accessed 13-01-2013 2013].

W3SCHOOLS. 2013o. *XPAth Syntax* [Online]. http://www.w3schools.com/xpath/xpath_syntax.asp: W3schools. [Accessed 15-01-2013 2013].

WALDT, D. 2010. *Six Strategies for Extending Xml Schemas in a Single Namespace* [Online]. http://www.ibm.com/developerworks/xml/library/x-xtendschema/index.html: IBM. [Accessed 12-11-2011 2011].

WALSH, N. 1998. A technical introduction to XML. *Available on the World Wide Web (accessed Oct 18, 2000): www. isgmlug. org*, 2-49.

WAN, C. & LIU, X. 2008. XML database technology. Beijing: Tsinghua University Press.

WANG, G. & LIU, M. Query processing and optimization for regular path expressions. Advanced Information Systems Engineering, 2003. Springer, 30-45.

WANG, H. & MENG, X. On the sequencing of tree structures for XML indexing. Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on, 2005. IEEE, 372-383.

WANG, H., PARK, S., FAN, W. & YU, P. S. ViST: a dynamic index method for querying XML data by tree structures. Proceedings of the 2003 ACM SIGMOD international conference on Management of data, 2003. ACM, 110-121.

WANG, W., JIANG, H., WANG, H., LIN, X., LU, H. & LI, J. Efficient processing of XML path queries using the disk-based F&B index. Proceedings of the 31st international conference on Very large data bases, 2005. VLDB Endowment, 145-156.

WEINSTEIN, A. M. 2010. Computer and video game addiction-a comparison between game users and non-game users. *The American journal of drug and alcohol abuse,* 36, 268-276.

WEISS, M. A. 1992. *Data Structures and Algorithms*, Benjamin/Cummings.

WHATLEY, K. 2009. *Xml Basics for New Users* [Online]. IBM. [Accessed 02-12-2012 2012].

WHITMER, R. 2004. Document Object Model (DOM) Level 3 XPath Specification. *W3C, http://www. w3. org/TR/DOM-Level-3-XPath*.

WONG, R. K., LAM, F. & SHUI, W. M. Querying and maintaining a compact XML storage. Proceedings of the 16th international conference on World Wide Web, 2007. ACM, 1073-1082.

WU, X., LEE, M. L. & HSU, W. A prime number labeling scheme for dynamic ordered XML trees. Data Engineering, 2004. Proceedings. 20th International Conference on, 2004. IEEE, 66-78.

XU, L., BAO, Z. & LING, T. W. A dynamic labeling scheme using vectors. Database and Expert Systems Applications, 2007. Springer, 130-140.

XU, L., LING, T. W., BAO, Z. & WU, H. Efficient label encoding for range-based dynamic XML labeling schemes. Database Systems for Advanced Applications, 2010. Springer, 262-276.

XU, L., LING, T. W. & WU, H. 2012. Labeling dynamic xml documents: an order-centric approach. *Knowledge and Data Engineering, IEEE Transactions on,* 24**,** 100-113.

XU, L., LING, T. W., WU, H. & BAO, Z. DDE: from dewey to a fully dynamic XML labeling scheme. Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, 2009. ACM, 719-730.

XU, Y. & PAPAKONSTANTINOU, Y. Efficient keyword search for smallest LCAs in XML databases. Proceedings of the 2005 ACM SIGMOD international conference on Management of data, 2005. ACM, 527-538.

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.* Springer.

YAO, B. B., OZSU, M. T. & KHANDELWAL, N. XBench benchmark and performance testing of XML DBMSs. Data Engineering, 2004. Proceedings. 20th International Conference on, 2004. IEEE, 621-632.

YIN, R. K. 2009 *Case Study Research Design and Methods,* London: SAGE., Harlow: Pearson Education Limited.

YOSHIKAWA, M., MENG, X., YUMOTO, T., MA, Q., SUN, L. & WATANABE, C. 2010. *Database Systems for Advanced Applications: 15th International Conference, DASFAA 2010, International Workshops: GDM, BenchmarX, MCIS, SNSMW, DIEW, UDM, Tsukuba, Japan, April 1-4, 2010, Revised Selected Papers*, Springer.

YU, J. X., LUO, D., MENG, X. & LU, H. 2005. Dynamically updating XML data: numbering scheme revisited. *World Wide Web,* 8**,** 5-26.

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**,** 56-70.

ZHANG, C., NAUGHTON, J., DEWITT, D., LUO, Q. & LOHMAN, G. On supporting containment queries in relational database management systems. ACM SIGMOD Record, 2001. ACM, 425-436.

ZHANG, N., HAAS, P. J., JOSIFOVSKI, V., LOHMAN, G. M. & ZHANG, C. Statistical learning techniques for costing XML queries. Proceedings of the 31st international conference on Very large data bases, 2005. VLDB Endowment, 289-300.

ZHUANG, C. & FENG, S. Full Tree-Based Encoding Technique for Dynamic XML Labeling Schemes. Database and Expert Systems Applications, 2012. Springer, 357-368.

Appendix A: Full Box Plots

a.1 Initial Labelling Experiments:

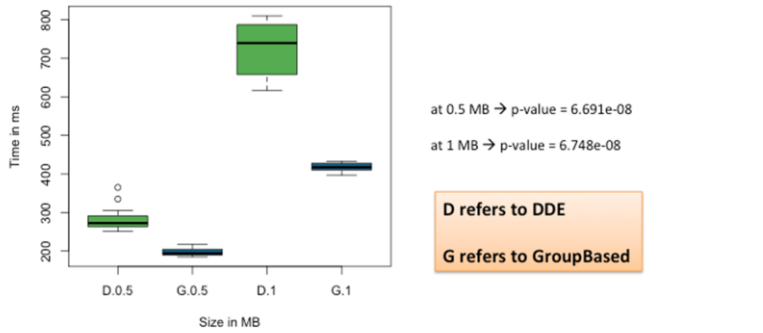


Figure 7.4.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes using XML files of size **0.5 and 1 MB** for the initial labelling experiment

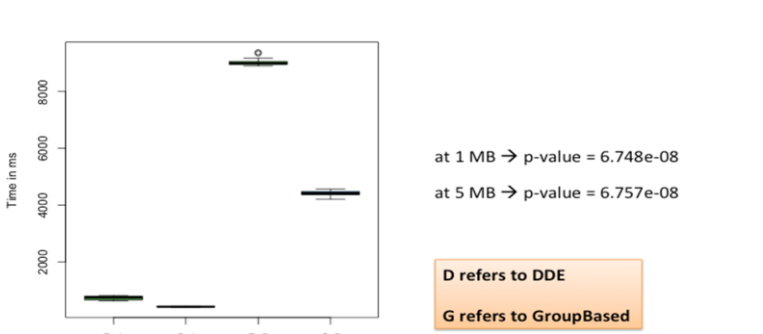


Figure 7.4.2 : The Boxplots and the p-value between DDE and the GroupBased Schemes using XML files of size **1 and 5 MB** for the initial labelling experiment

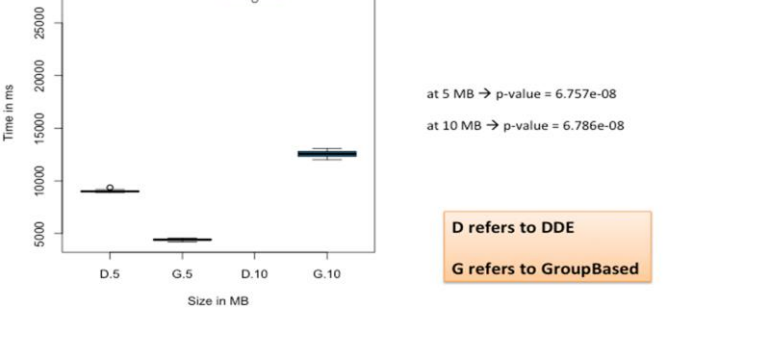
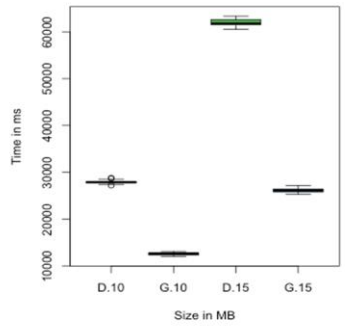


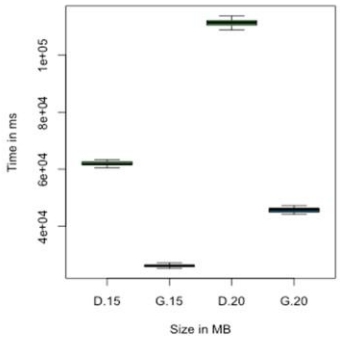
Figure 7.4.3 : The Boxplots and the p-value between DDE and the GroupBased Schemes using XML files of size **5 and 10 MB** for the initial labelling experiment



at 10 MB → p-value = 6.757e-08

at 15 MB → p-value = 6.786e-08

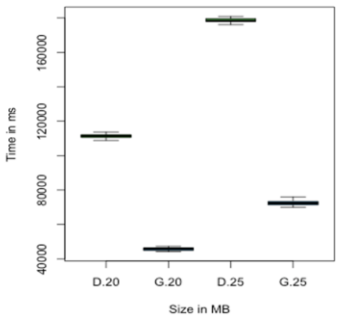
Figure 7.4.4 : The Boxplots and the p-value between DDE and the GroupBased Schemes using XML files of size **10 and 15 MB** for the initial labelling experiment



at 15 MB → p-value = 6.757e-08

at 20 MB → p-value = 1.451e-11

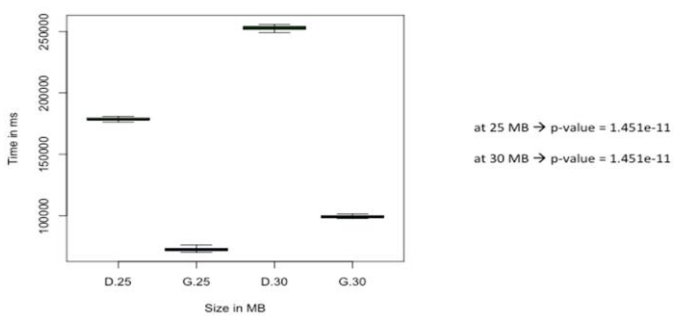
Figure 7.4.5 : The Boxplots and the p-value between DDE and the GroupBased Schemes using XML files of size **15 and 20 MB** for the initial labelling experiment



at 20 MB → p-value = 1.451e-11

at 25 MB → p-value = 1.451e-11

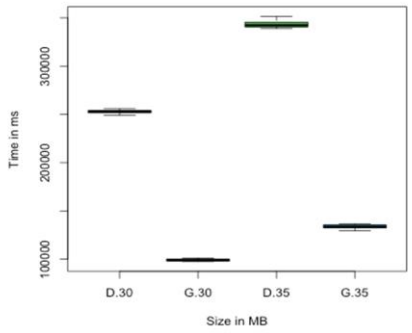
Figure 7.4.6 : The Boxplots and the p-value between DDE and the GroupBased Schemes using XML files of size **20 and 25 MB** for the initial labelling experiment



at 25 MB → p-value = 1.451e-11

at 30 MB → p-value = 1.451e-11

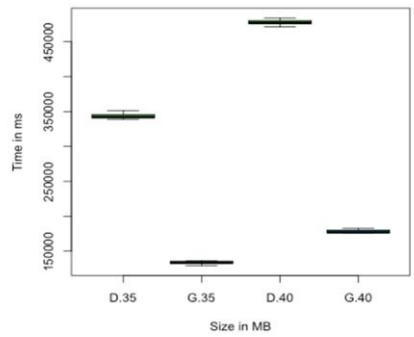
Figure 7.4.7 : The Boxplots and the p-value between DDE and the GroupBased Schemes using XML files of size **25 and 30 MB** for the initial labelling experiment



at 30 MB → p-value = 1.451e-11

at 35 MB → p-value = 1.451e-11

Figure 7.4.8 : The Boxplots and the p-value between DDE and the GroupBased Schemes using XML files of size **30 and 35 MB** for the initial labelling experiment



at 35 MB → p-value = 1.451e-11

at 40 MB → p-value = 1.451e-11

Figure 7.4.9 : The Boxplots and the p-value between DDE and the GroupBased Schemes using XML files of size **35 and 40 MB** for the initial labelling experiment

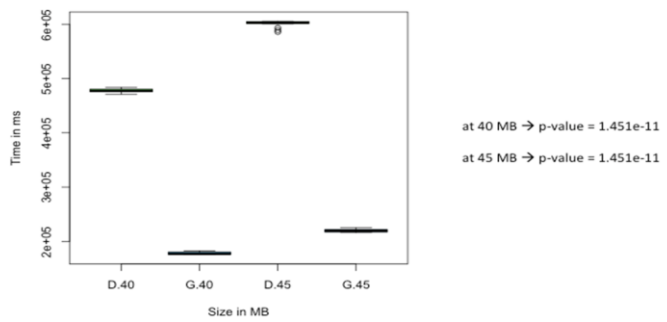


Figure 7.4.10 : The Boxplots and the p-value between DDE and the GroupBased Schemes using XML files of size **40 and 45 MB** for the initial labelling experiment

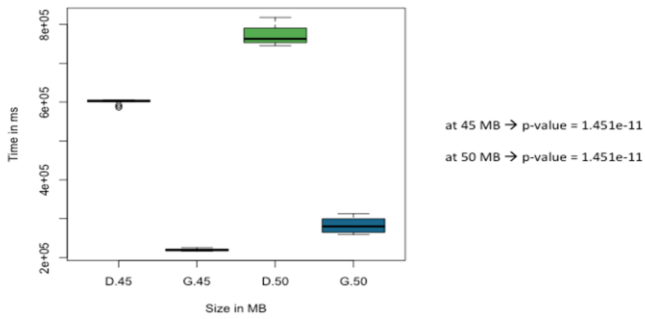


Figure 7.4.11 : The Boxplots and the p-value between DDE and the GroupBased Schemes using XML files of size **45 and 50 MB** for the initial labelling experiment

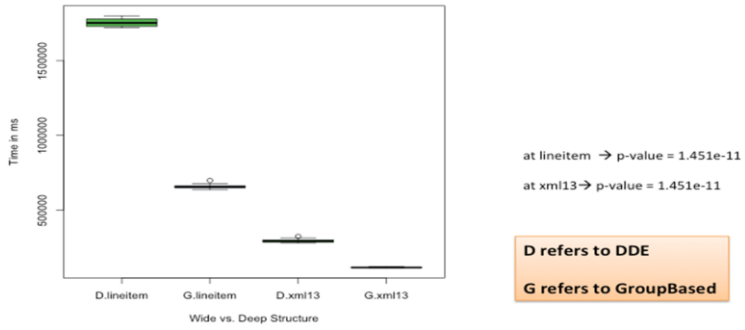


Figure 7.5.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes using the dataset **lineitem and xml13** for the initial labelling experiment

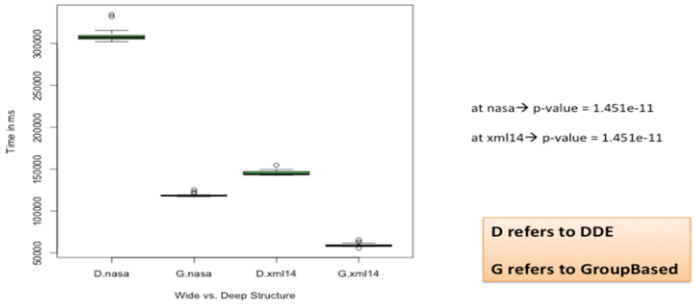


Figure 7.5.2 : The Boxplots and the p-value between DDE and the GroupBased Schemes using the dataset **nasa and xml14** for the initial labelling experiment

a.2 Determining Different Relationships on Static XML:

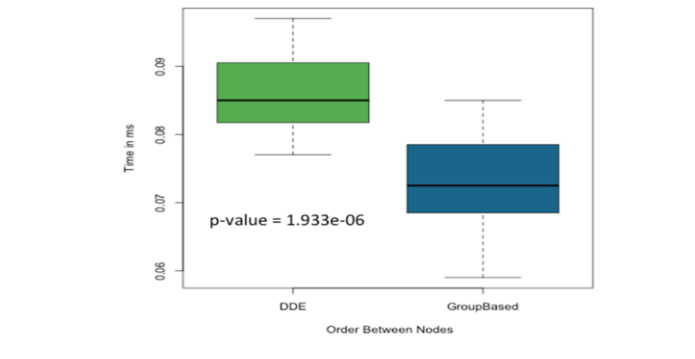


Figure 7.7.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying the **Order** between two nodes in a static XML document

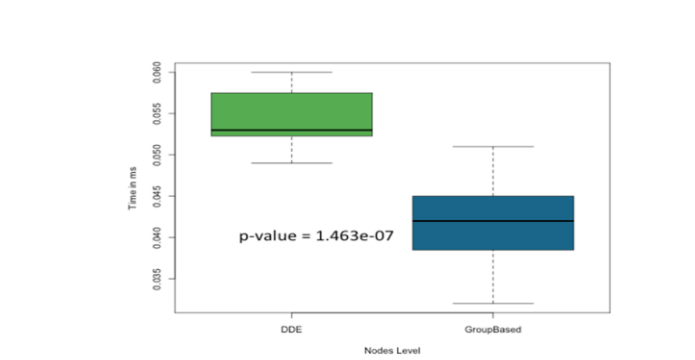


Figure 7.7.2 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying nodes's **Level** in a static XML document

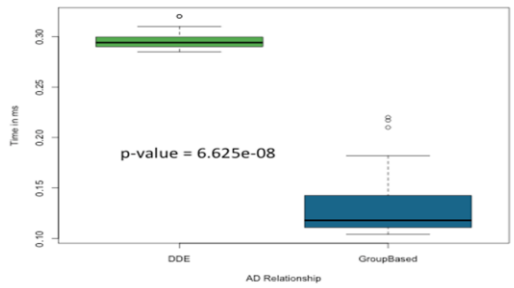


Figure 7.7.3 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying the **AD relationship** between two nodes in a static XML document

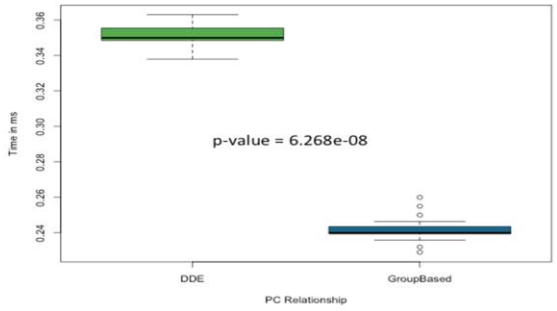


Figure 7.7.4 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying the **PC relationship** between two nodes in a static XML document

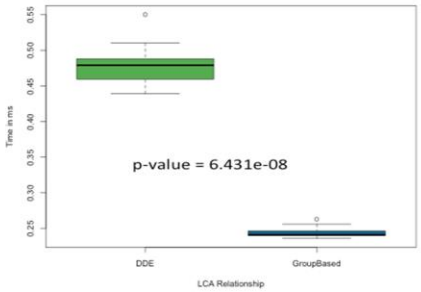


Figure 7.7.5 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying the **LCA** between two nodes in a static XML document

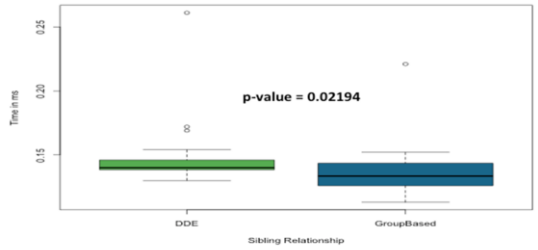


Figure 7.7.6 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying the **Sibling relationship** between two nodes in a static XML document

a.3 Queries on Static XML:

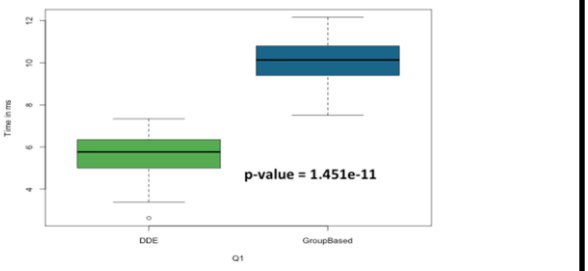


Figure 7.9.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q1)**

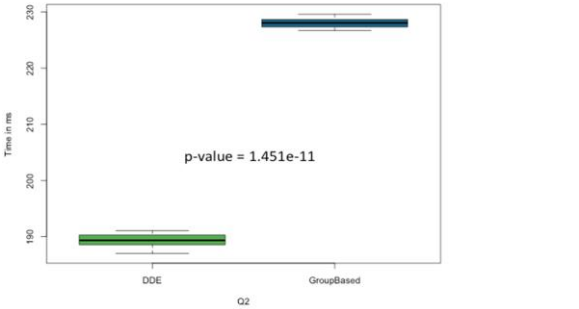


Figure 7.9.2 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q2)**

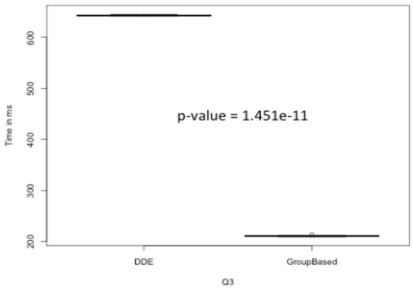


Figure 7.9.3 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q3)**

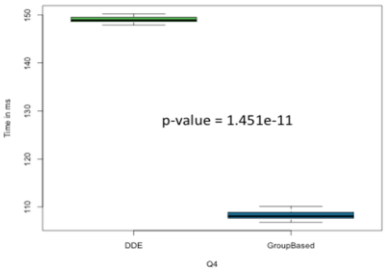


Figure 7.9.4 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q4)**

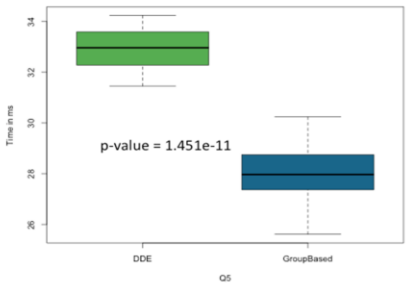


Figure 7.9.5 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q5)**

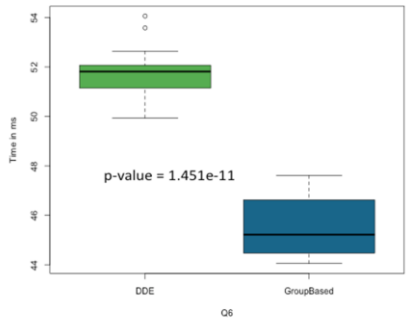


Figure 7.9.6 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q6)**

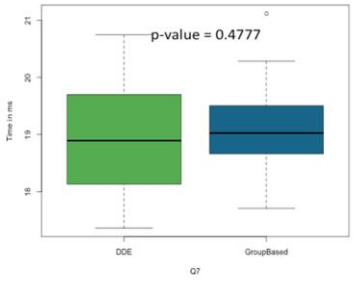


Figure 7.9.7 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q7)**

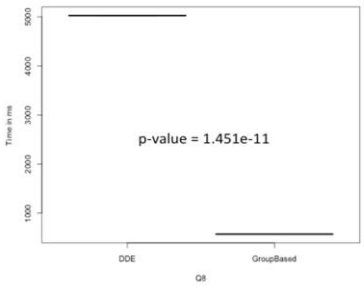


Figure 7.9.8 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q8)**

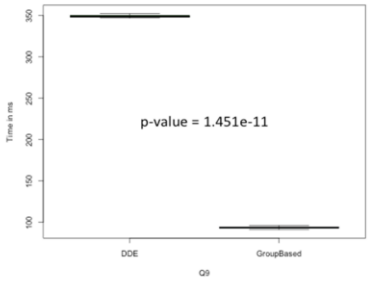


Figure 7.9.9 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q9)**

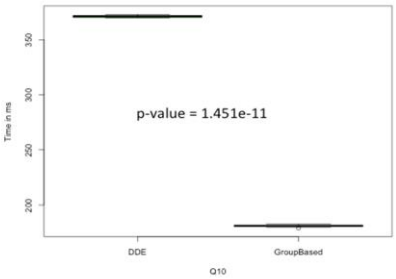


Figure 7.9.10 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q10)**

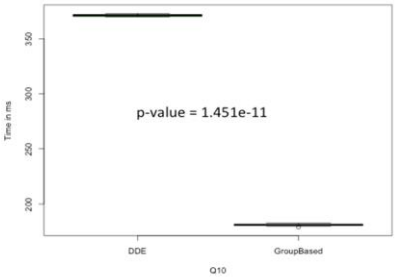


Figure 7.9.10 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q10)**

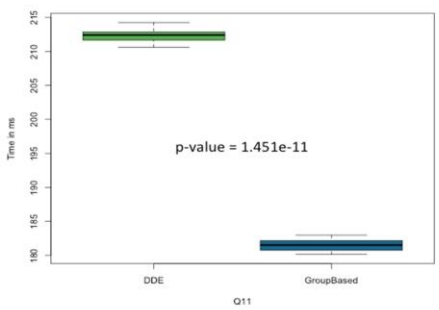


Figure 7.9.11 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q11)**

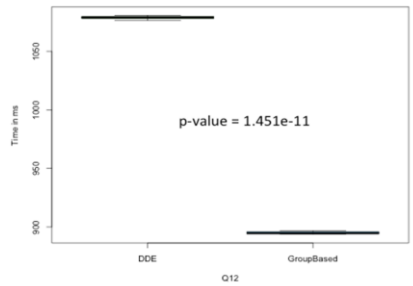


Figure 7.9.12 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q12)**

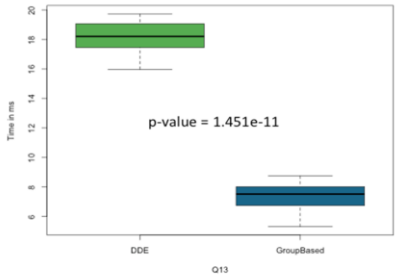


Figure 7.9.13 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q13)**

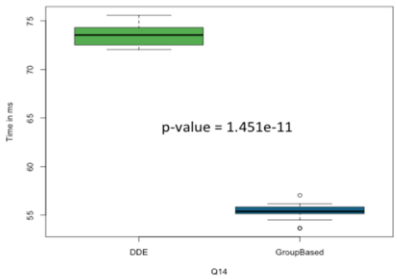


Figure 7.9.14 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q14)**

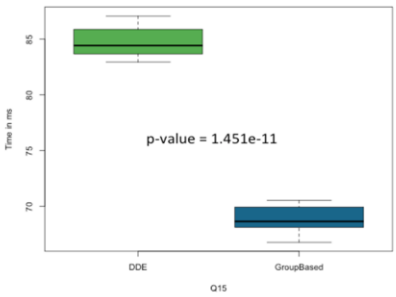


Figure 7.9.15 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q15)**

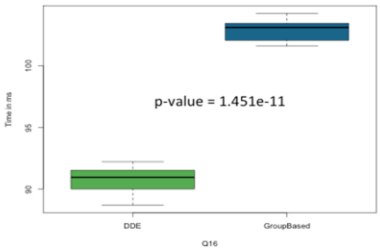


Figure 7.9.16 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q16)**

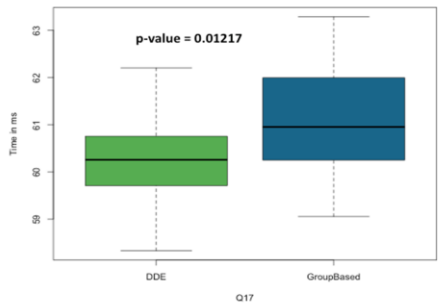


Figure 7.9.17 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q17)**

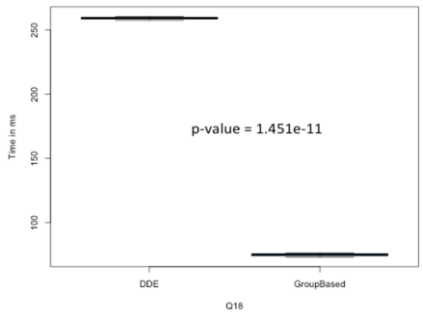


Figure 7.9.18 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q18)**

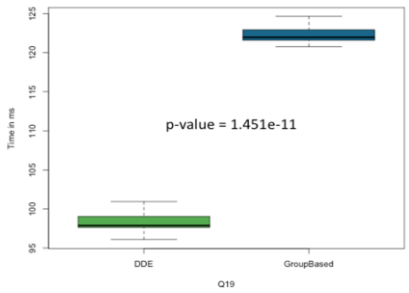


Figure 7.9.19 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on static XML document **(Q19)**

a.4 Uniform Insertions:

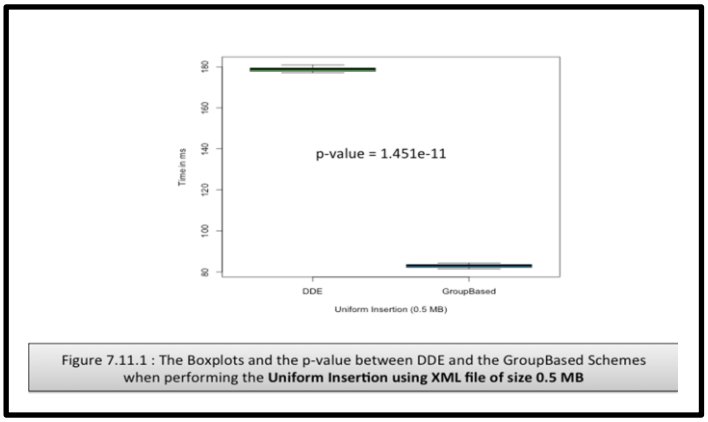


Figure 7.11.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Uniform Insertion using XML file of size 0.5 MB**

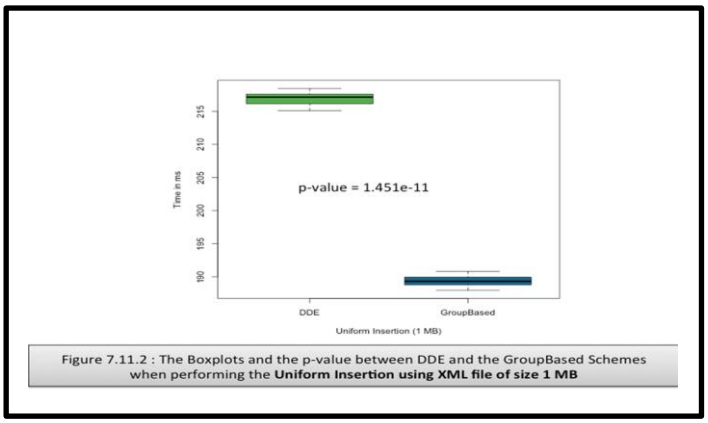


Figure 7.11.2 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Uniform Insertion using XML file of size 1 MB**

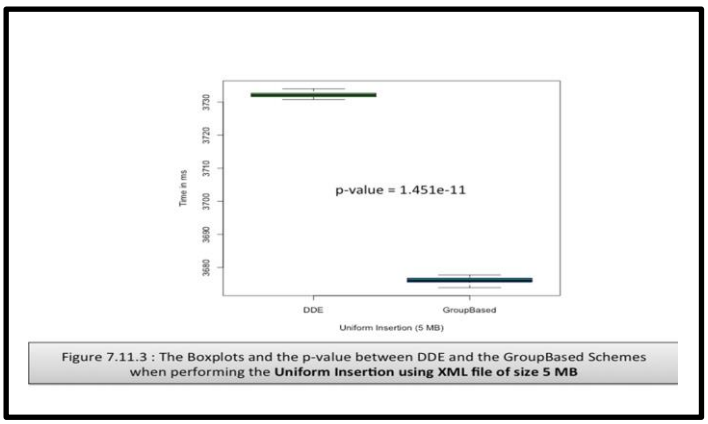


Figure 7.11.3 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Uniform Insertion using XML file of size 5 MB**

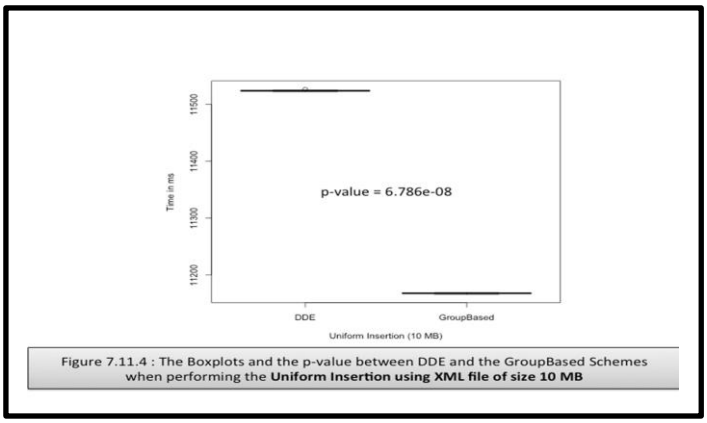


Figure 7.11.4 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Uniform Insertion using XML file of size 10 MB**

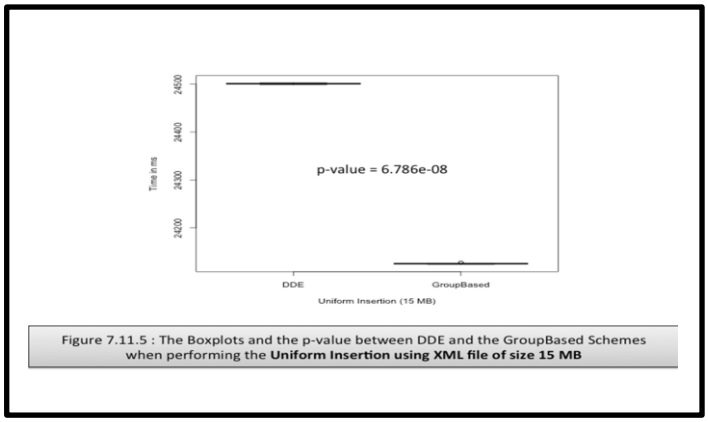


Figure 7.11.5 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Uniform Insertion using XML file of size 15 MB**

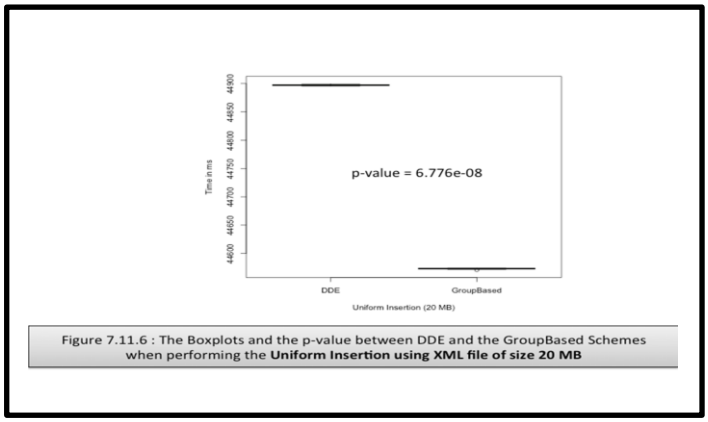


Figure 7.11.6 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Uniform Insertion using XML file of size 20 MB**

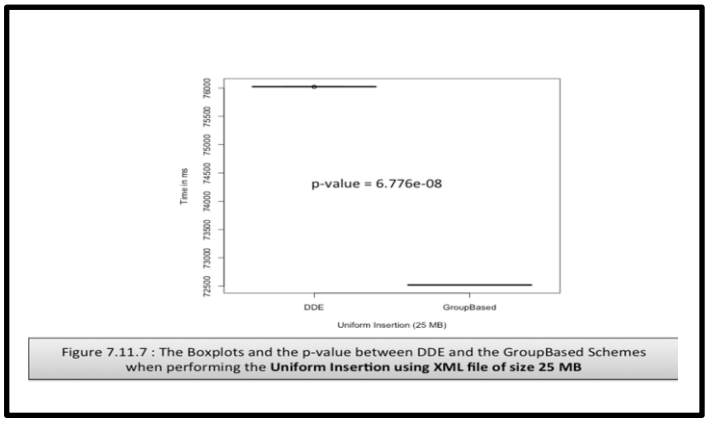


Figure 7.11.7 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Uniform Insertion using XML file of size 25 MB**

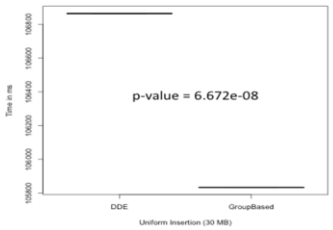


Figure 7.11.8 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Uniform Insertion using XML file of size 30 MB**

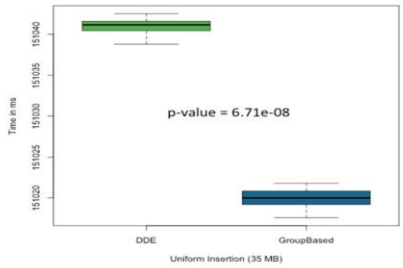


Figure 7.11.9 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Uniform Insertion using XML file of size 35 MB**

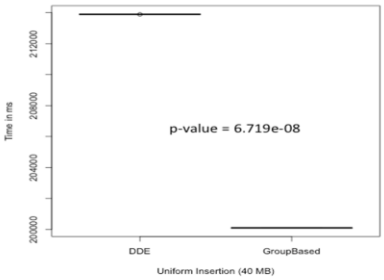


Figure 7.11.10 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Uniform Insertion using XML file of size 40 MB**

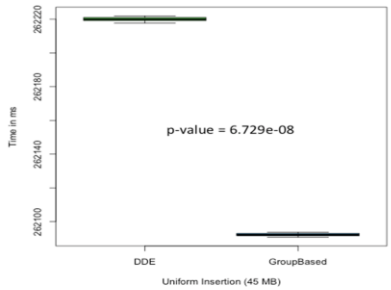


Figure 7.11.11 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Uniform Insertion using XML file of size 45 MB**

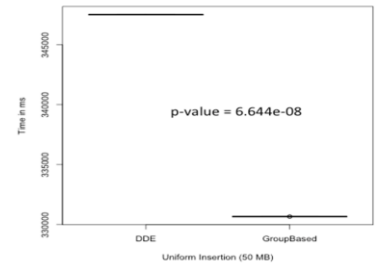


Figure 7.11.12 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Uniform Insertion using XML file of size 50 MB**

a.5 Ordered-Skewed Insertions:

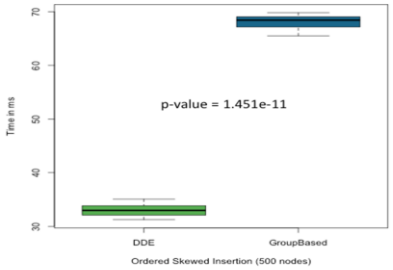


Figure 7.13.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Ordered Skewed Insertion (500 nodes were inserted)**

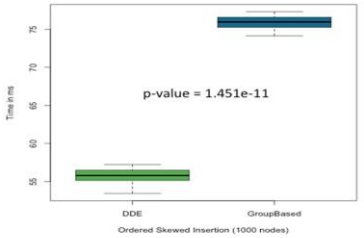


Figure 7.13.2 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Ordered Skewed Insertion (1000 nodes were inserted)**

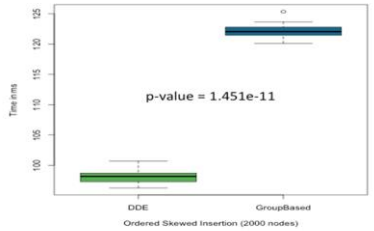


Figure 7.13.3 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Ordered Skewed Insertion (2000 nodes were inserted)**

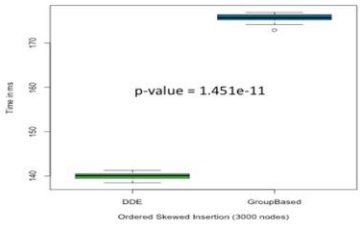


Figure 7.13.4 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Ordered Skewed Insertion (3000 nodes were inserted)**

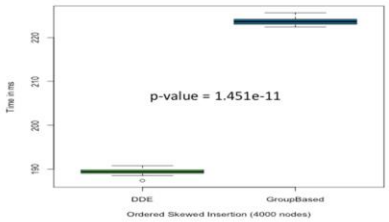


Figure 7.13.5 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Ordered Skewed Insertion (4000 nodes were inserted)**

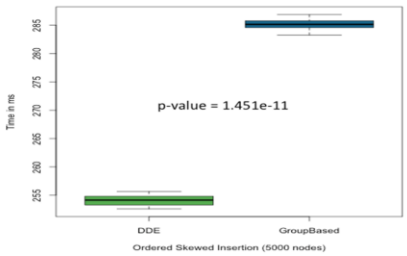


Figure 7.13.6 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Ordered Skewed Insertion (5000 nodes were inserted)**

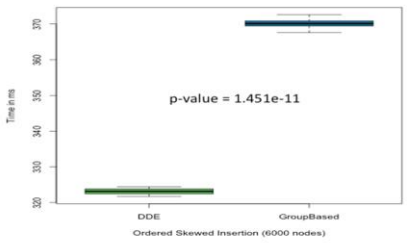


Figure 7.13.7 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Ordered Skewed Insertion (6000 nodes were inserted)**

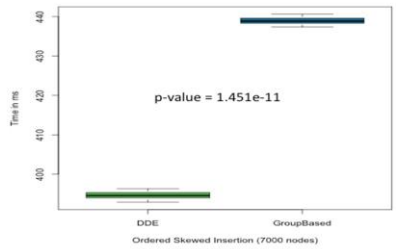


Figure 7.13.8 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Ordered Skewed Insertion (7000 nodes were inserted)**

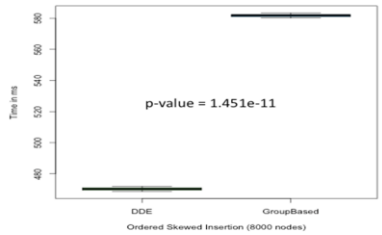


Figure 7.13.9 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Ordered Skewed Insertion (8000 nodes were inserted)**

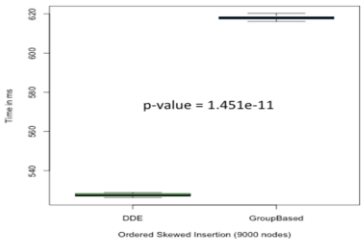


Figure 7.13.10 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Ordered Skewed Insertion (9000 nodes were inserted)**

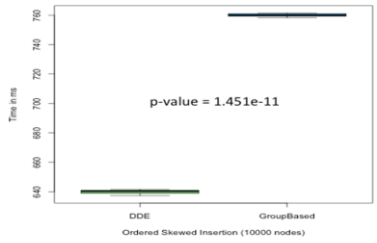


Figure 7.13.11 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Ordered Skewed Insertion (10000 nodes were inserted)**

a.6 Random-Skewed Insertions:

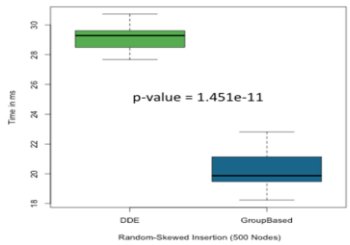


Figure 7.15.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Random Skewed Insertion (500 nodes were inserted)**

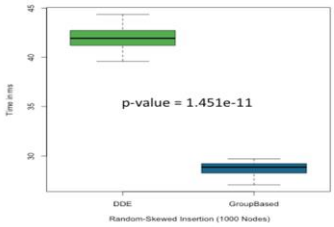


Figure 7.15.2 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Random Skewed Insertion (1000 nodes were inserted)**

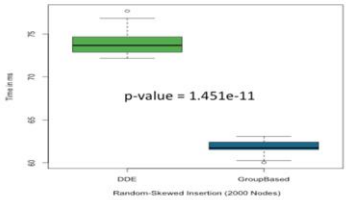


Figure 7.15.3 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Random Skewed Insertion (2000 nodes were inserted)**

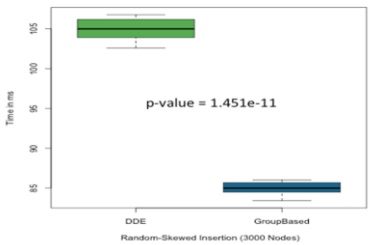


Figure 7.15.4 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Random Skewed Insertion (3000 nodes were inserted)**

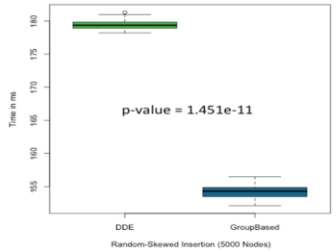


Figure 7.15.6 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Random Skewed Insertion (5000 nodes were inserted)**

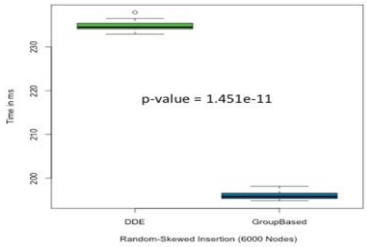


Figure 7.15.7 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Random Skewed Insertion (6000 nodes were inserted)**

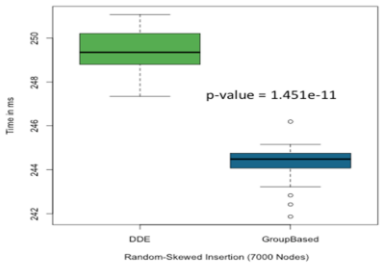


Figure 7.15.8 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Random Skewed Insertion (7000 nodes were inserted)**

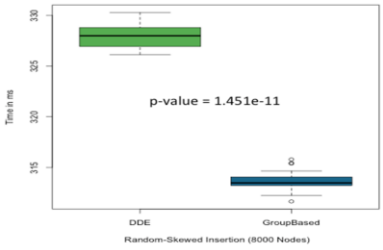


Figure 7.15.9 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Random Skewed Insertion (8000 nodes were inserted)**

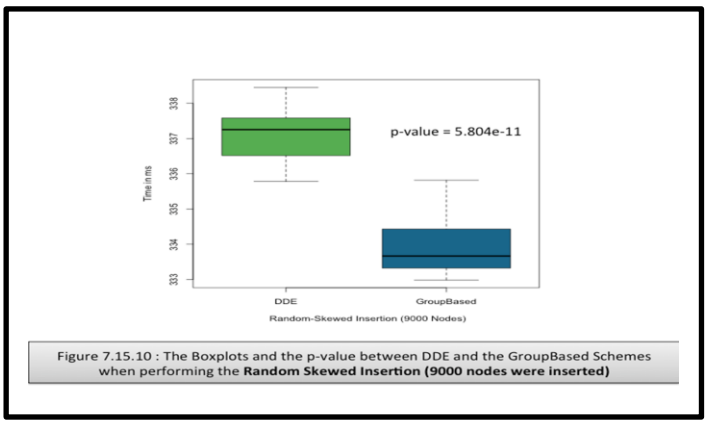


Figure 7.15.10 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Random Skewed Insertion (9000 nodes were inserted)**

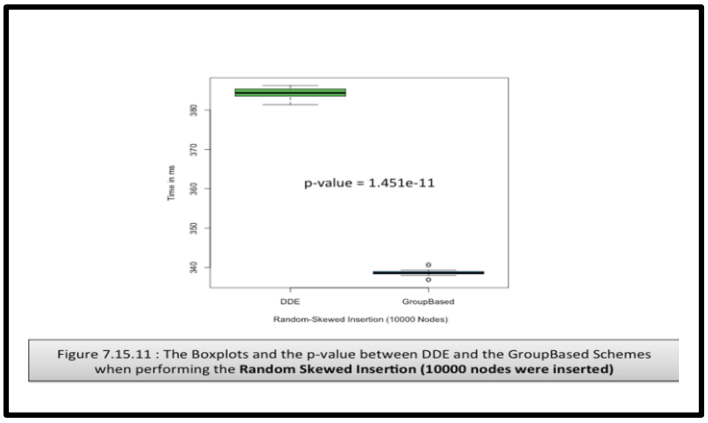


Figure 7.15.11 : The Boxplots and the p-value between DDE and the GroupBased Schemes when performing the **Random Skewed Insertion (10000 nodes were inserted)**

a.7 Relationships after Uniform-Insertions:

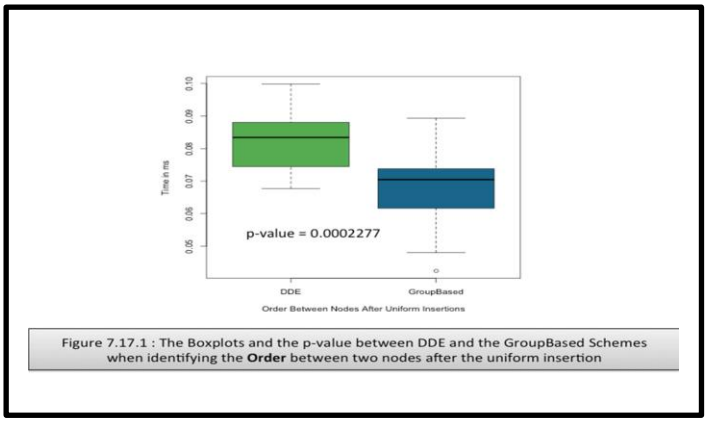


Figure 7.17.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying the **Order** between two nodes after the uniform insertion

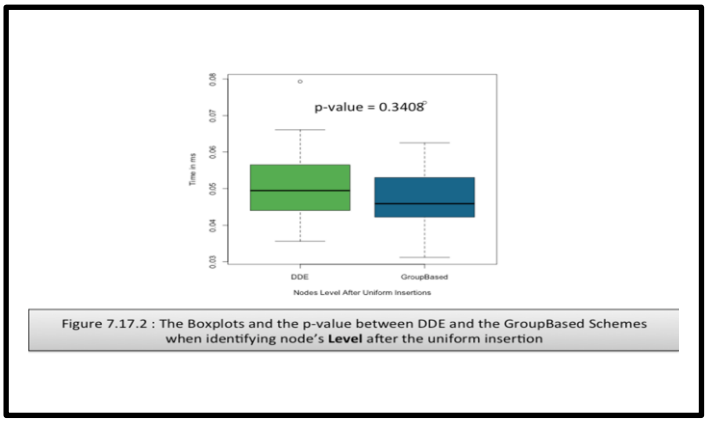


Figure 7.17.2 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying node's **Level** after the uniform insertion

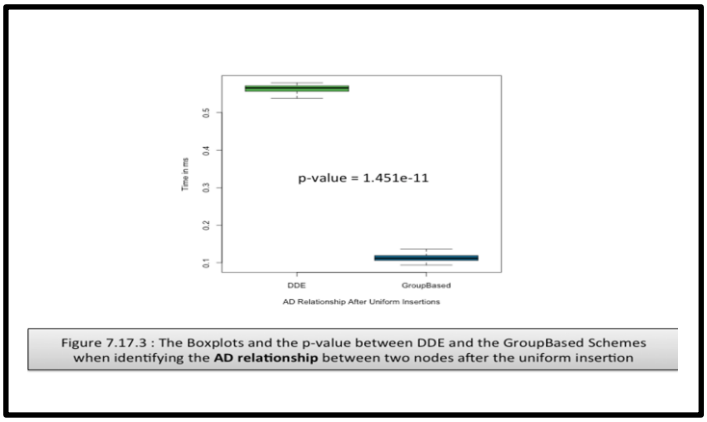


Figure 7.17.3 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying the **AD relationship** between two nodes after the uniform insertion

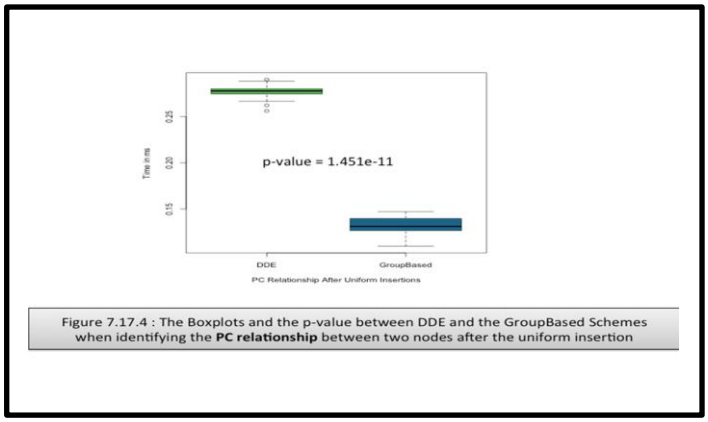


Figure 7.17.4 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying the **PC relationship** between two nodes after the uniform insertion

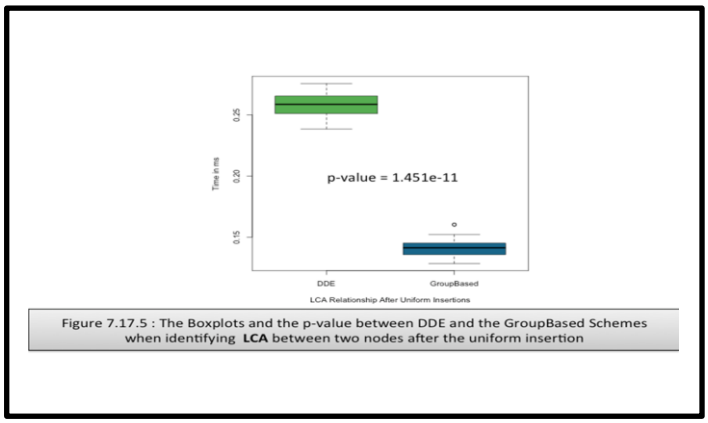


Figure 7.17.5 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying **LCA** between two nodes after the uniform insertion

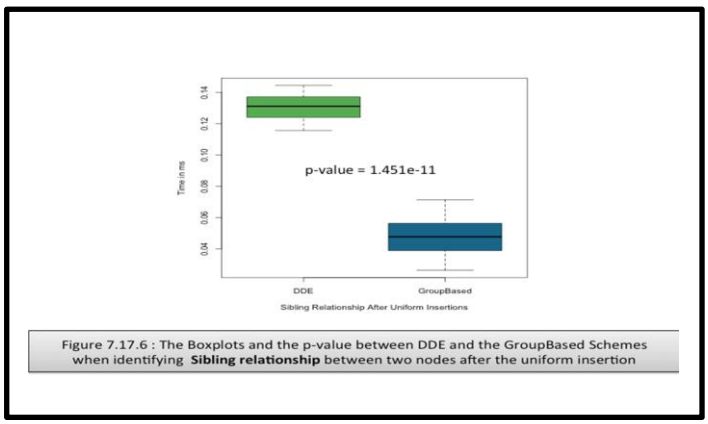


Figure 7.17.6 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying **Sibling relationship** between two nodes after the uniform insertion

a.8 Relationships after Ordered-Skewed Insertions:

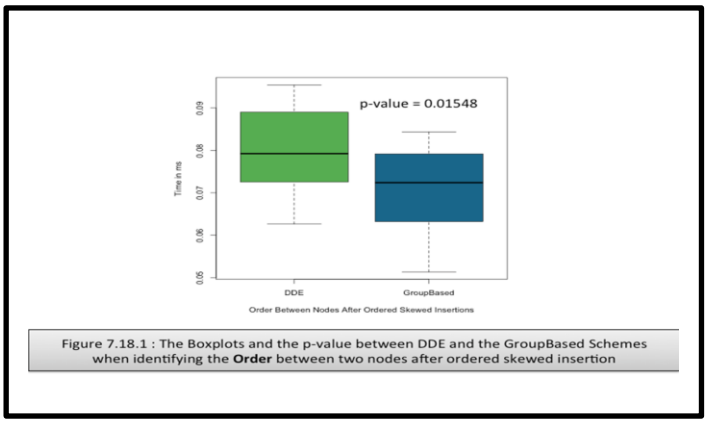


Figure 7.18.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying the **Order** between two nodes after ordered skewed insertion

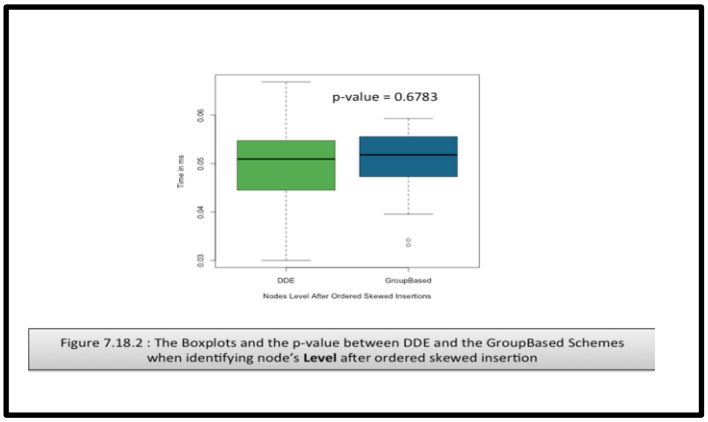


Figure 7.18.2 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying node's **Level** after ordered skewed insertion

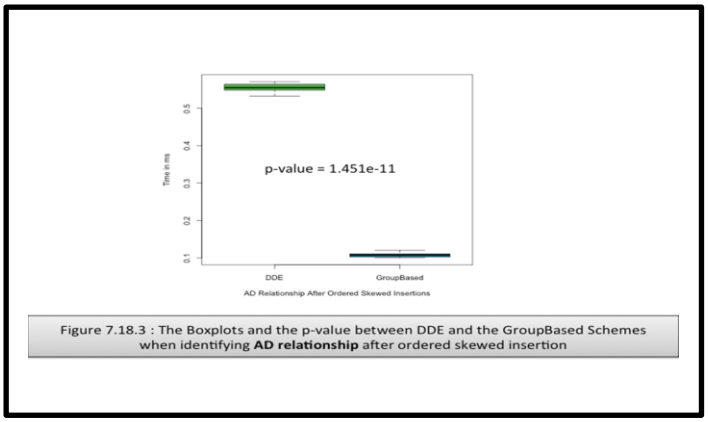


Figure 7.18.3 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying **AD relationship** after ordered skewed insertion

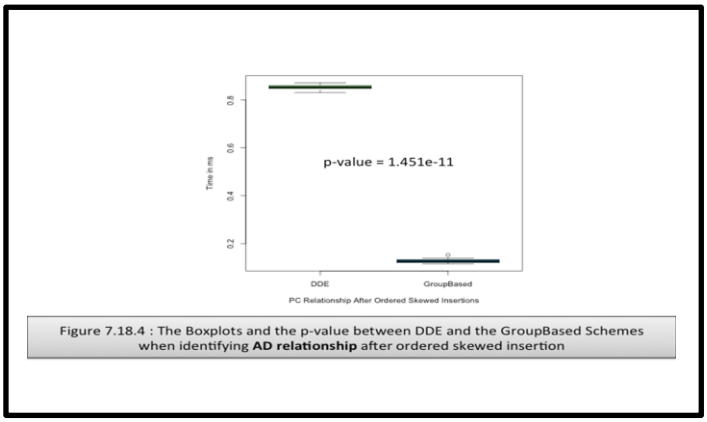


Figure 7.18.4 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying **AD relationship** after ordered skewed insertion

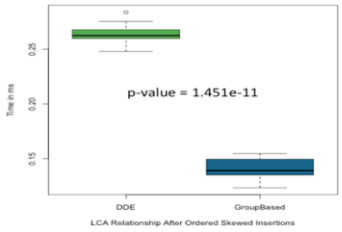


Figure 7.18.5 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying **PC relationship** after ordered skewed insertion

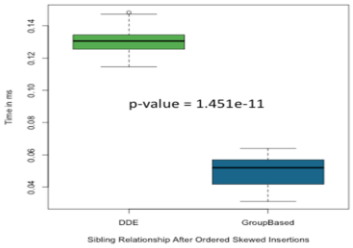


Figure 7.18.6 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying **Sibling relationship** after ordered skewed insertion

a.9 Relationships after Random-Skewed Insertions:

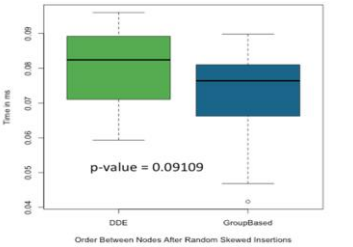


Figure 7.19.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying the **Order** between two nodes after random skewed insertion

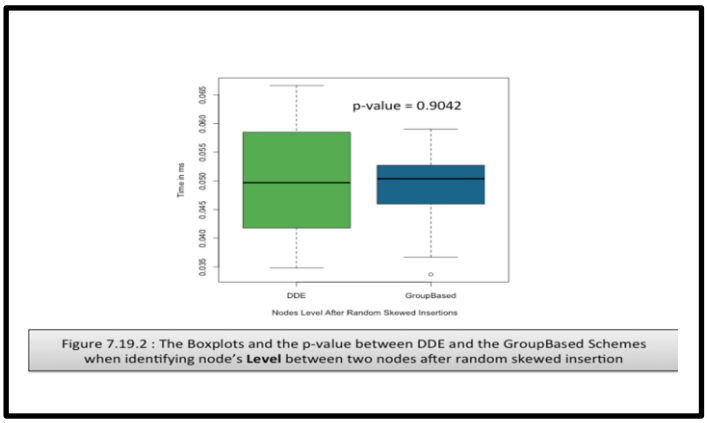


Figure 7.19.2 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying node's **Level** between two nodes after random skewed insertion

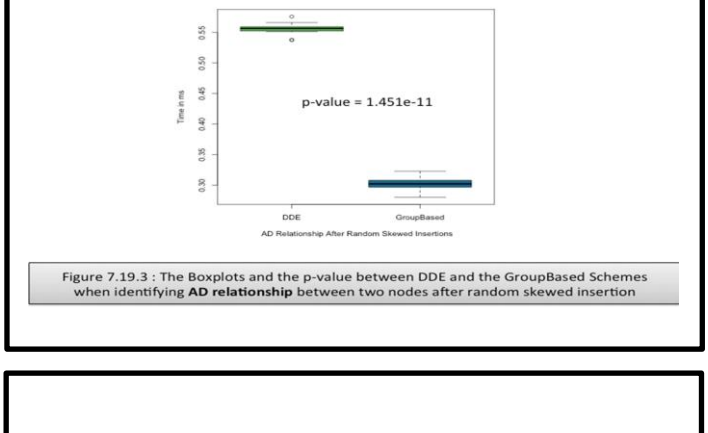


Figure 7.19.3 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying **AD relationship** between two nodes after random skewed insertion

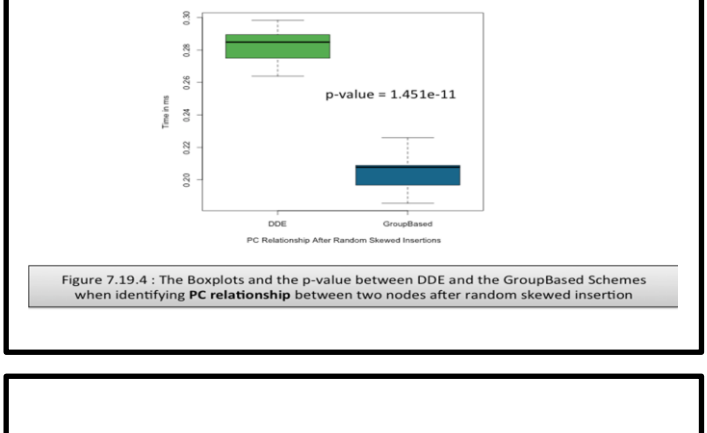


Figure 7.19.4 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying **PC relationship** between two nodes after random skewed insertion

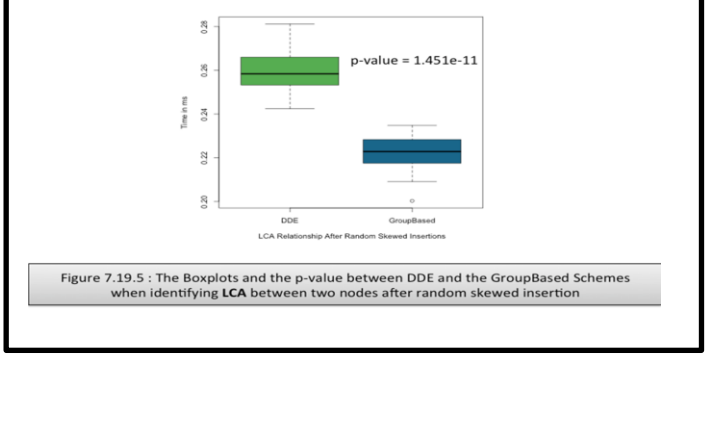


Figure 7.19.5 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying **LCA** between two nodes after random skewed insertion

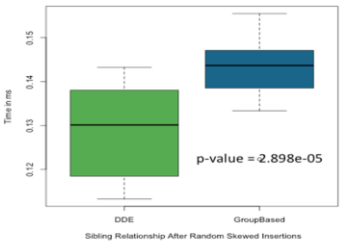


Figure 7.19.6 : The Boxplots and the p-value between DDE and the GroupBased Schemes when identifying **Sibling relationship** between two nodes after random skewed insertion

a.10 Queries on Dynamic XML:

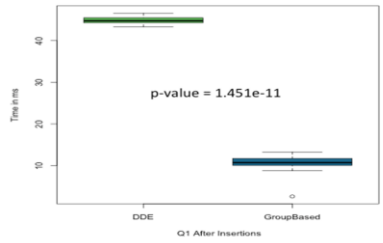


Figure 7.21.1 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on dynamic XML document **(Q1)**

Figure 7.21.2 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on dynamic XML document **(Q2)**

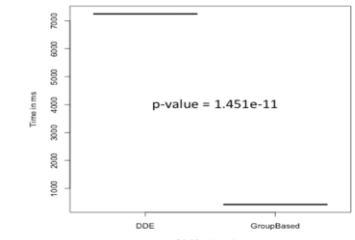


Figure 7.21.3 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on dynamic XML document **(Q3)**

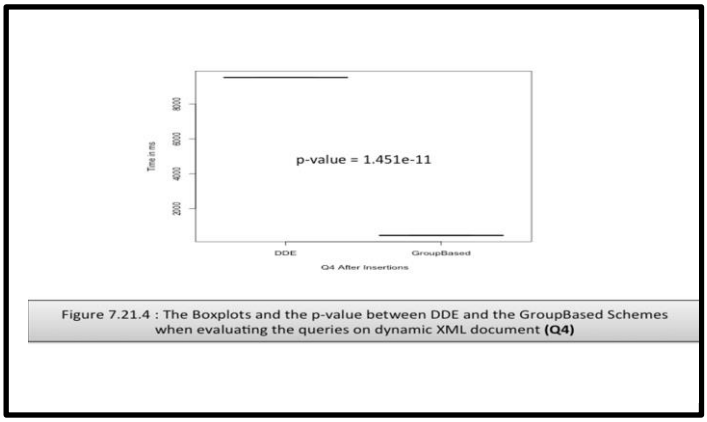


Figure 7.21.4 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on dynamic XML document **(Q4)**

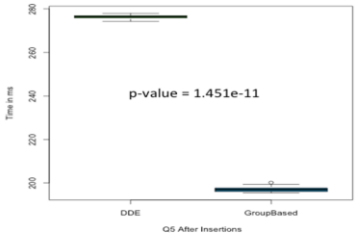


Figure 7.21.5 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on dynamic XML document **(Q5)**

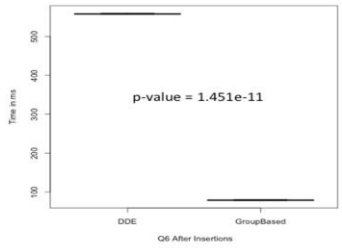


Figure 7.21.6 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on dynamic XML document **(Q6)**

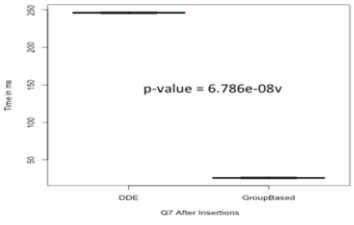


Figure 7.21.7 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on dynamic XML document **(Q7)**

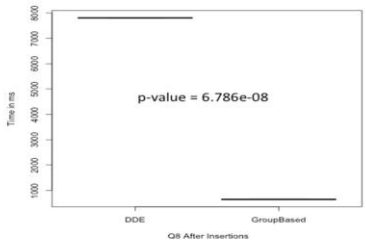


Figure 7.21.8 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on dynamic XML document **(Q8)**

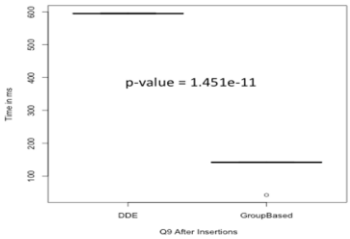


Figure 7.21.9 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on dynamic XML document **(Q9)**

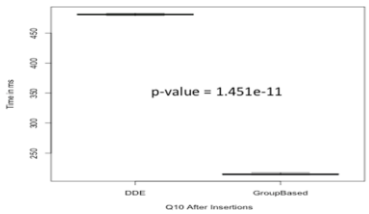


Figure 7.21.10 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on dynamic XML document **(Q10)**

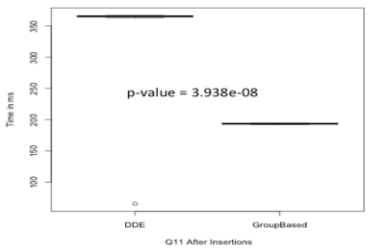


Figure 7.21.11 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on dynamic XML document **(Q11)**

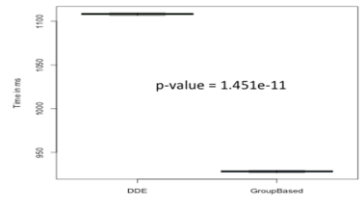


Figure 7.21.12 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on dynamic XML document **(Q12)**

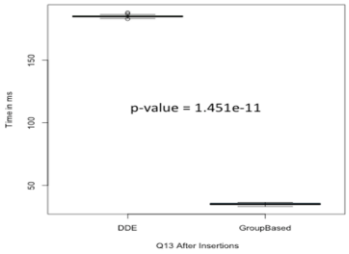


Figure 7.21.13 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on dynamic XML document **(Q13)**

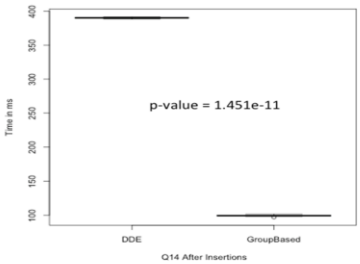


Figure 7.21.14 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on dynamic XML document **(Q14)**

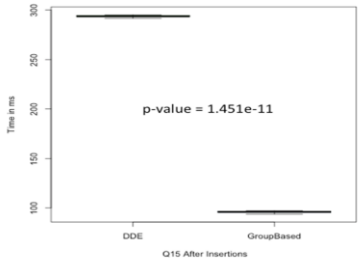


Figure 7.21.15 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on dynamic XML document **(Q15)**

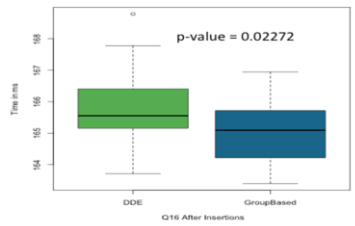


Figure 7.21.16 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on dynamic XML document **(Q16)**

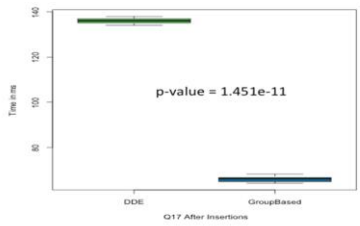


Figure 7.21.17 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on dynamic XML document **(Q17)**

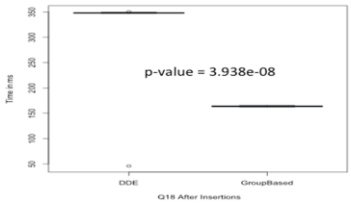


Figure 7.21.18 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on dynamic XML document **(Q18)**

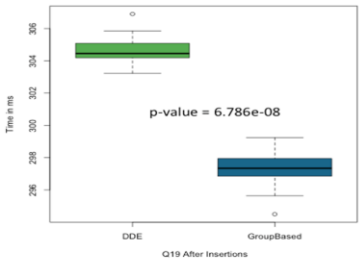


Figure 7.21.19 : The Boxplots and the p-value between DDE and the GroupBased Schemes when evaluating the queries on dynamic XML document **(Q19)**