OPTIMISATION TECHNIQUES FOR STORING AND QUERYING XML DATA IN RELATIONAL DATABASE SYSTEMS

A thesis submitted in fulfilment of the requirements of
the degree of Doctor of Philosophy

Mo'ad Maghaydah

December 2010

Master of Engineering Science (University of New South Wales) 2001

Bachelor of Engineering (Jordan University of Science and Technology) 1995

Department of Computing

Faculty of Science

Macquarie University, NSW 2109, Australia

© 2010 Moad Maghaydah

Acknowledgements

I am grateful to my supervisor, Professor Mehmet A. Orgun, for his supervision, encouragement, unbounded enthusiasm, and generous support from the preliminary to the concluding level. He led me to the correct direction at every stage of the research. Without his care, supervision and friendship, this thesis would not have been possible.

I am also grateful to my co-supervisor, Dr. Abhaya Nayak, for his suggestions and support.

I am deeply indebted to my wife Hadeel and my two beautiful children Tamir and Talah who brought a great deal of love and inspiration to my life, and have always been by my side to encourage and motivate me.

I would like to express my appreciation to the managers and my colleagues in the IT Department at Star Track Express Pty Limited for their understanding and support during my time there as an IT Systems Specialist (2004–2010), while I was also carrying out this research.

I would like to acknowledge that this thesis was edited by Dr Lisa Lines director and head editor of Elite Editing and Tutoring, the editorial intervention was restricted to Standards D and E of the Australian Standards for Editing Practice.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the thesis.

Declaration

I hereby certify that the work embodied in this thesis is the result of original research. This work has not been submitted for a higher degree to any other university or institution.

Signed: ______________________________

Date : ______________________________

Abstract

Extensible Markup Language (XML) has recently emerged as a standard for electronic information interchange, due to its flexibility and portability. For instance, in service-oriented applications, XML messages are commonly used for inter-company interactions. However, those XML messages may be used for purposes other than transactions and data interchange (for example, purchase orders and invoice statements) and they must often be retained for later use and analysis. This requires scalable technology to effectively store and query XML data.

Due to their widespread availability and robustness, relational database management systems (RDBMS) still offer the most affordable technology to develop XML database systems. However, the XML data model presents new challenges such as maintaining the document order and supporting complex structural-join queries, which require tree-aware processing mechanisms. While the state-of-the-art approaches to support XML data in relational systems require new algorithms and indexing techniques that make them powerful, it has been observed that some of those changes may not be directly applicable to relational database systems and/or they may present a trade-off between performance and storage usage. Further, the modification of the relational system's kernel is hardly an option for many RDBMS vendors. There are still considerable benefits in developing solutions that do not involve changes to the RDBMS's kernel, thereby reducing the cost of re-engineering relational database systems.

In order to improve the process of storing and querying XML data in relational systems, in this thesis, we propose a new compact Dewey-based labelling scheme to support it. The new label structure, composed of two components (parent, child) in the Dewey format, would significantly improve the performance of those XML queries that are based on parent-child and sibling relationships. Moreover, we propose advanced query optimisation techniques based on certain features that exist in Dewey labels and based on a better utilisation of the document schema summary of XML documents. Our techniques are portable and can be applied to any Dewey-based labeling technique proposed for storing and querying XML data. Through extensive experimental studies, we show that these techniques make off-the-shelf relational systems more tree-aware, and significantly improve their capabilities to support XML data.

Contents

List of Figures XIII
List of Tables........ XVII
Chapter 1: Introduction 1
1.1 Introduction 1
1.2 XML Database Systems 2
1.2.1 Relational-based XML Database Systems........ 3
1.2.2 Native XML Database Systems........ 4
1.3 Motivation 5
1.4 Problem Definitions 6
1.5 Research Contributions 8
1.6 Publications 9
Chapter 2: XML Overview 13
2.1 Introduction 13
2.1.1 XML Application Domains 14
2.2 XML Data Model 15
2.2.1 Features of the XML Tree Data Model 16
2.2.2 XML Schema........ 17
2.3 XML Documents........ 19
2.3.1 XML Syntax 20
2.3.2 A Well-formed XML Document 21
2.3.3 A Valid XML Document........ 22
2.4 Querying XML Data 22
2.4.1 XPath Path Expressions Overview 24
2.4.2 XQuery Overview........ 25
2.5 Conclusion........ 26
Chapter 3: Prefixing on Demand Labelling Approach 29
3.1 Introduction 29
3.1.1 Motivation........ 30
3.2 Background 31
3.2.1 Labelling and Indexing Techniques in XML Database Systems........ 32
3.2.1.1 Range and Intervals Encoding Techniques........ 33
3.2.1.2 Prefix-free String and Dewey Labels........ 35
3.2.1.3 Other Labelling Approaches........ 37
3.3 Prefixing on Demand (PoD)........ 38
3.3.1 Basic Labelling Unit (BLU) 38
3.3.2 Supporting the Insertion of New Nodes 40
3.3.3 Features of the PoD Labelling Scheme 42
3.3.4 Two-component Dewey Labels........ 44
3.3.5 New Nodes Insertion Is Still Supported in Pod-S Mode........ 46
3.3.6 PoD Space Requirement Analysis........ 48
3.3.7 User-defined Functions Used in PoD 49
3.4 Evaluation Experiments 50
3.4.1 Study Overview 50
3.4.2 Evaluating Different Configurations of PoD........ 52
3.4.2.1 Test Setup........ 52

3.4.2.2 Results and Discussion *53*
3.4.2.3 Brief Discussion *56*
3.4.3 Evaluating PoD Space Efficiency Using Double Parsing 56
3.4.3.1 Test Setup *57*
3.4.3.2 Results and Discussion *57*
3.4.4 Comparing PoD-4 with Other Dewey Labelling Approaches 58
3.4.4.1 Test Setup *58*
3.4.4.2 Results and Discussion *58*
3.4.5 Evaluating Space Requirements for PoD-Split 59
3.4.5.1 Test Setup *60*
3.4.5.2 Results and Discussion *60*
3.4.6 Evaluating the Effect of Inserting New Nodes 61
3.4.6.1 Test Setup *61*
3.4.6.2 Results and Discussion *61*
3.4.7 Evaluating the Effect of Label Size on the Query Performance 62
3.4.7.1 Test Setup *63*
3.4.7.2 Results and Discussion *63*
3.5 Conclusion 64
Chapter 4: Navigating the XML Tree Using the PoD Approach 67
4.1 Introduction 67
4.2 XML Query Optimisation Objectives 68
4.3 XML Query in Relational Database Systems 69
4.3.1 Labelling Techniques Effect 70
4.3.2 XML Schema Information Effect 71
4.4 XML Schema Summary in the PoD System 72
4.4.1 Capturing XML Document Schema Summary 72
4.4.1.1 In-memory Schema Summary *74*
4.4.1.2 XML Schema Summary in Table Format *75*
4.5 PoD Optimisation Approach 76
4.5.1 Supporting Search on Path Expression 77
4.5.2 Optimising Child and Descendants Axis Steps 78
4.5.3 Evaluating Ancestor at any Level of the Tree 79
4.5.4 Evaluating Document Structure and Statistics Queries 81
4.6 Evaluating and Optimising the XPath Axis Steps 81
4.6.1 XPath Axis Steps Overview 82
4.6.2 The Child Axis Step 83
4.6.3 The Descendant Axis Step 85
4.6.4 The Descendant-self Step 85
4.6.5 The Attribute Axis Step 86
4.6.6 The Following Axis Step 86
4.6.7 The Following-sibling Axis Step 87
4.6.8 The Parent Axis Step 87
4.6.9 The Ancestor Axis Step 88
4.6.10 The Ancestor-self Axis Step 90
4.6.11 The Preceding Axis Step 90
4.6.12 The Preceding-sibling Step 90
4.7 XPath Axis Steps Experimental Evaluation 91
4.7.1 Study Overview 91
4.7.2 Experiment Setup 91
4.7.3 Child Axis Step 92

4.7.4 Descendant Axis 95
4.7.5 Following Axis 97
4.7.6 Following-sibling Axis 98
4.7.7 Parent Axis 100
4.7.8 Ancestor Axis 101
4.7.9 Preceding Axis 104
4.7.10 Preceding-sibling Axis 105
4.8 Conclusion 106
Chapter 5: Optimising XML Structural-join Queries 107
5.1 Introduction 107
5.2 XML Query Optimisation Techniques 108
5.2.1 Optimisation Based on Query Rewriting 109
5.2.2 Implementation of New Join Operators 110
5.2.3 New Labelling and Indexing Techniques 111
5.2.4 Effect of Indexing Data Values 112
5.2.5 XML Schema to Relational Schema Mapping Techniques 113
5.2.5.1 Model-Mapping (Fixed Relational Schema) 113
5.2.5.2 Structure-Mapping (DTD Dependents) 115
5.2.5.3 XML Data Type 116
5.3 PoD Relational Schema 116
5.3.1 The Edge Approach 117
5.3.2 The Node Type Approach 118
5.4 Finding Optimal Join Order in the PoD System 119
5.5 PoD Approach for Optimising Twig Queries 122
5.5.1 Sibling-based Twig Queries 123
5.5.2 Ancestor-descendent Twig Queries 124
5.6 Minimising XML Queries 126
5.6.1 Building the XPath Matrix 127
5.6.2 Minimising the XPath Matrix 128
5.7 Putting It All Together 131
5.8 Experimental Evaluations 136
5.8.1 Experiment Setup 137
5.8.2 Join Order Effect 137
5.8.2.1 Experiment Setup 138
5.8.2.2 Results and Analyses 139
5.8.3 Twig Query Optimisation 141
5.8.3.1 Test Setup 141
5.8.3.2 Result and Analysis 142
5.8.4 Efficiency of Off-the-shelf Relational Systems 143
5.8.4.1 Test Setup 143
5.8.4.2 Results and Discussion 144
5.9 Conclusion 149
Chapter 6: Conclusions and Future Work 151
6.1 Conclusions 151
6.2 Future Work 154
Appendix A: XML Benchmarks 155
Bibliography 177

List of Figures

Figure 2-1: Sample XML tree representing the XML data model. 15

Figure 2-2: Sample XML schema using XSD language. 19

Figure 2-3: Sample bookstore XML document. 21

Figure 2-4: Sample XQuery/XPath expressions. 23

Figure 2-5: Sample XML tree fragment from sample bookstore XML document. 26

Figure 2-6: Sample XQuery and equivalent SQL query. 26

Figure 3-1: Region encoding representation (pre, size, level) for a sample XML tree. 34

Figure 3-2: Dewey label representation for a sample XML tree. 35

Figure 3-3: ORDPATH labelling scheme, which eliminates relabelling. 37

Figure 3-4: PoD labelling for a sample people database. 43

Figure 3-5: Space requirements comparison between BLU_D and BLU_4. 57

Figure 3-6: Maximum label length comparison between PoD and two other Dewey-based labelling techniques (Ordpath and QED). 59

Figure 3-7: Total label size comparison in logarithmic scale between PoD and two other Dewey-based labelling techniques (Ordpath and QED) using documents from Table 3-5. 59

Figure 3-8: Total label size comparison in logarithmic scale between PoD and PoD-S (Split mode) using the documents from Table 3-6. 60

Figure 3-9: Queries' run time in logarithmic scale for 16 queries from the Michigan benchmark evaluated using different Dewey-based labelling schemes. 64

Figure 3-10: Queries' run time in logarithmic scale for 16 queries from the Michigan benchmark evaluated using different Dewey-based labelling schemes. 64

Figure 4-1: Block diagram for XML-schema summary in the PoD System. 73

Figure 4-2: A sample XML document. 76

Figure 4-3: Relational schema for major relational-based approaches. 77

Figure 4-3: Query run-times in logarithmic scale for queries Q1, Q2, and Q3 in Table 4-2. 93

Figure 4-4: Query run-times in logarithmic scale for queries Q4, Q5, and QR2 in Table 4-2. 93

Figure 4-5: Query run-times in logarithmic scale for queries Q2, Q3 in Table 4-2. 94

Figure 4-6: Query run-times in logarithmic scale for queries Q4, Q5 in Table 4-2. 94

Figure 4-7: Query run times in logarithmic scale for queries Q1 and Q2 in Table 4-3. 95

Figure 4-8: Query run times in logarithmic scale for queries Q3, Q4 and QR3 in Table 4-3. 96

Figure 4-9: Query run times in logarithmic scale for queries Q1 and Q2 in Table 4-3. 96

Figure 4-10: Query run times in logarithmic scale for queries Q3 and Q4 in Table 4-3. 97

Figure 4-11: Query run times in logarithmic scale for queries Q1 and Q2 in Table 4-4. 98

Figure 4-12: Query run times in logarithmic scale for queries Q3 and Q4 in Table 4-4. 98

Figure 4-13: Query run times in logarithmic scale for queries Q1 and Q2 in Table 4-5. 100

Figure 4-14: Query run times in logarithmic scale for queries Q3 and Q4 in Table 4-5. 100

Figure 4-15: Query run-times in logarithmic scale for Q1 and Q2 in Table 4-5. 101

Figure 4-16: Query run times in logarithmic scale for Q3 and Q4 in Table 4-5. 101

Figure 4-17: Query run times in logarithmic scale for Q1 and Q2 in Table 4-5 using the ancestor axis. 102

Figure 4-18: Query run times in logarithmic scale for Q3 and Q4 in Table 4-5 using the ancestor axis. 102

Figure 4-19: Query run times in logarithmic scale for Q1 and Q2 in Table 4-5 for the ancestor axis using XGP function. 103

Figure 4-20: Query run times in logarithmic scale for Q3 and Q4 in Table 4-5 for the ancestor axis using XGP function. 103

Figure 4-21: Query run times in logarithmic scale for Q1& Q2 in Table 4-5 for preceding axis. 104

Figure 4-22: Query run times in logarithmic scale for Q3& Q4 in Table 4-5 for preceding axis. 104

Figure 4-23: Query run times in logarithmic scale for Q1 and Q2 for preceding-sibling axis. 105

Figure 4-24: Query run times in logarithmic scale for Q3 and Q4 for preceding-sibling axis. 105

Figure 5-1: Edge relational schema in PoD and PoD-S. 118

Figure 5-2: Node type relational schema in PoD. 118

Figure 5-3: Join order that might result in a slower execution plan. 119

Figure 5-4: A more efficient Join order for XML queries. 120

Figure 5-5: Sample XML twig queries that search for the occurrence of the shaded node patterns. 123

Figure 5-6: Initial XQuery and XPath query path matrix. 127

Figure 5-7: Initial XPath matrix for query in Example 5-2. 128

Figure 5-8: An example of cyclic structural relationship joins. 129

Figure 5-9: Sample Join_list based on the query in Example 5-2. 130

Figure 5-10: Algorithm outline to minimise XPath matrix. 131

Figure 5-11: Algorithm outline to generate optimised join list....................................132

Figure 5-12: Algorithm outline to generate optimised SQL query. 133

Figure 5-13: Initial XPath Matrix for XML query in Example 5-3. 135

Figure 5-14: Minimised XPath Matrix for XML query in Example 5-3. 135

Figure 5-15: The join_list for XML query in Example 5-3. 135

Figure 5-16: Query run times in logarithmic scale for 11 queries from XMark benchmark for three Dewey-based labelling schemes showing the difference between forced join order (SJ) and the join order generated by the system built-in optimiser. 139

Figure 5-17: Query run times in logarithmic scale for 11 queries from XMark benchmark for three Dewey-based labelling schemes showing the difference between forced join order (SJ) and the join order generated by the system built-in optimiser. 140

Figure 5-18: Query run times in logarithmic scale for XMark benchmark queries against a medium XML document for three different approaches in XML management systems. 145

Figure 5-19: Query run times in logarithmic scale for XMark benchmark queries against a large XML document for three different approaches in XML management systems. 146

Figure 5-20: Query run times in logarithmic scale for three different XML management systems using the Michigan benchmark with a 50MB document. 147

Figure 5-21: Query run times in logarithmic scale for three different XML management systems using the Michigan benchmark with 500MB document. 148

List of Tables

Table 3-1: Dewey and PoD representations for some sample labels in Example 3-1. ...40
Table 3-2: Some suggested configurations for BLU of 4 bits. 44
Table 3-3: PoD and PoD-S representations for the labels in Example 3-3.....................46
Table 3-4: PoD and PoD-Split representation for labels: N_1, Nx, and N_2 in Example 3-4. 47
Table 3-5: The features of some XML documents from different XML benchmarks....51
Table 3-6: BLUs of different sizes and configurations, which are used in space evaluation test..52
Table 3-7: The maximum label length for small BLU configurations using the evaluation documents from Table 3-5..53
Table 3-8: The total label size for small BLU configurations using the evaluation documents from Table 3-5. 53
Table 3-9: The maximum label length for medium BLU configurations using the evaluation documents from Table 3-5...54
Table 3-10: The total label size for medium BLU configurations using the evaluation documents from Table 3-5. 54
Table 3-11: The maximum label length for large BLU configurations using evaluation documents from Table 3-5. 55
Table 3-12: The total label sizes for large BLU configurations using evaluation documents from Table 3-5. 55
Table 3-13: Details of some update queries from the Michigan benchmark. 61
Table 3-14: The number of nodes that were added and deleted after executing each update query in Table 3-13..62
Table 3-15: Label size changes after executing update queries in Table 3-13. 62
Table 4.1: XPath axis steps. 83
Table 4-2: Queries used to evaluate the child axis step. 92
Table 4-3: Queries used to evaluate the descendant axis step. 95
Table 4-4: Queries used to evaluate the following axis step...97
Table 4-5: Queries used to evaluate the following-sibling axis step. 99
Table 5-1: Join relationship priority...129
Table 5-2: Run time results for selected twig queries from XMark benchmark...........142