

Fast Similarity Search in XML Data

Dissertation

am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

eingereicht von

Torsten Schlieder

am 9. Dezember 2002

Betreuer:

Prof. Dr. Heinz Schweppe
Prof. Dr. Myra Spiliopoulou

Datum der Disputation: 14. April 2003

Abstract

The eXtensible Markup Language (XML) is a widely accepted standard for the representation of data. The more data is stored in XML documents, the more important become methods for effective and efficient searching. An important characteristics of XML documents is their self-describing structure. Queries that specify selection conditions for the structure promise to greatly improve the precision of the search. However, the use of the structure can also be problematic, because it is hard for users to learn all of the details of the often complex and heterogeneous structure required to phrase a query, and because structural selection conditions often lead to overspecified queries that miss relevant results.

In this thesis, we propose an innovative method for searching in XML data, which uses the descriptive structure as a guide to locate the requested information. A user needs only partial knowledge of the structure to formulate queries that specify conditions on both the content and structure of documents. A query is interpreted in such a way that it retrieves not only exact matches, but also results considered to be similar to the query. To find the similar results, sequences of transformations are applied to the query so that its structure is adapted to the structure of each document in the collection. Each transformation within a sequence has a cost; the total cost of a sequence measures the similarity between the original query and a document matched by the transformed query. This total cost is assigned to the document and determines its position in the list of results, which is sorted by decreasing similarity. By adjusting the costs, the interpretation of queries can be tailored to the needs of different users, and also to the varied characteristics of XML documents.

We present all necessary algorithms and data structures to implement a query processor that answers a query in polynomial—typically sublinear—time with respect to the size of the database. For a given query, the query processor creates a compact query-execution plan that represents all possible query transformations. It evaluates the plan by executing operators that successively calculate the transformation costs for each document in the collection. We present techniques to effectively optimize the evaluation of query-execution plans by exploiting equivalences between operators. To reduce the query-evaluation times even more, we propose a method to retrieve the best n results, without computing similarity scores for all documents in the collection. This method uses a structural summary of the data to estimate the best k transformed queries, which are successively evaluated until the best n results are found.

The theoretical concepts are validated by a prototypical implementation. We describe the architecture of the prototype, and discuss the results of systematic tests carried out to analyze the evaluation times for a representative set of queries with respect to various collections of real and synthetic XML documents.

Acknowledgements

Many people have accompanied my long journey to this dissertation. To all of them I give my deepest thanks, especially

My first advisor, Prof. Heinz F. Schweppe, for his continuous guidance, support, and his belief in my work.

My second advisor, Prof. Myra Spiliopoulou, for her critical yet constructive remarks, which substantially improved this dissertation.

The members of the Database Group at the Freie Universität Berlin for many controversial discussions, helpful feedback, and for the great time we had together.

Among them, in particular Annika Hinze, for working as my closest colleague, sharing literally all my joys and sorrows throughout this effort.

The students and professors of the Graduate School in Distributed Information Systems for their useful suggestions and all the great discussions at our meetings.

Among them, in particular Ulf Leser and Felix Naumann, for all the help and support they gave, leading the path I was to follow.

Prof. Klaus U. Schulz from the Universität München, for enabling a very close and successful cooperation with his group.

Holger Meuss from the Universität München, for the very enjoyable, interesting, and fruitful work on our joint research project.

My family, for their patience and understanding.

Last but not least, Ella, for her encouragement and belief in me, for her warmth and love.

This research was supported by the German Research Society, Berlin-Brandenburg Graduate School in Distributed Information Systems (DFG grant no. GRK 316).

Contents

1	Introduction	1
1.1	Searching in XML Data: Why Traditional Approaches Fail	1
1.2	Thesis Objective	4
1.3	Solutions Presented in the Thesis: An Overview	4
1.4	Thesis Outline	6
2	State of the Art	9
2.1	XML and Semistructured Data	9
2.2	Query Languages for XML	11
2.2.1	The Basic Query Components	12
2.2.2	Language Aspects Related to Semistructured Data	13
2.3	Flexible Query Mappings for Trees and Graphs	14
2.3.1	Flexible Query Mappings without Valuation	14
2.3.2	Flexible Query Mappings with Valuation	15
2.4	XML and Information Retrieval	16
2.4.1	Query Languages and Algebras for Structured Text Databases	17
2.4.2	Focused Document Retrieval	18
2.4.3	Combining Structured Queries and Relevance Ranking	19
2.5	Distance Measures for Labeled Trees	22
2.6	Similarity Search in Tree-Structured Data	23
2.7	A Taxonomy of XML Query Languages and Retrieval Models	24
2.8	What is Missing?	26
3	The approXQL Query Language	31
3.1	The Core Syntax	31
3.1.1	Structural Selectors	32
3.1.2	Data Selectors and Predicates	33
3.1.3	Containment Expressions	34

3.1.4	Boolean Expressions	34
3.2	The Extended Syntax	35
3.2.1	Query Restrictions	37
3.2.2	Query Relaxations	38
3.2.3	The Type System	39
4	Modeling Documents and Queries	41
4.1	Trees and their Properties	41
4.2	Type-Value Trees	43
4.3	Tree Representation of XML Documents	45
4.4	Tree Representation of <i>approXQL</i> Queries	47
4.5	Related Work	50
5	Querying by Approximate Tree Embedding	51
5.1	The Interpretation of <i>approXQL</i> Queries: An Overview	52
5.2	The Tree-Embedding Formalism	53
5.3	Basic Transformations of Query Trees	55
5.3.1	Deletions	55
5.3.2	Permutations	56
5.3.3	Insertions	57
5.3.4	Value Changes	58
5.4	Assigning Costs to Basic Transformations	59
5.5	Approximate Query Answering	61
5.6	Example	63
5.7	Related Work	64
6	Direct Query Evaluation	67
6.1	From the Semantics of a Query to its Evaluation: An Overview	68
6.2	The Expanded Representation of a Query	69
6.2.1	Constructing the Basic Form of an Expanded Query Representation	72
6.2.2	Encoding Deletions	72
6.2.3	Encoding Permutations	73
6.2.4	Encoding Value Changes	77
6.3	Cost-Calculating Set Operations	77
6.3.1	Node-Cost Tuples	77
6.3.2	Selection	78
6.3.3	Join and Outerjoin	79
6.3.4	Union and Intersection	80

6.3.5	Operator Equivalences	81
6.4	Query-Execution Plans	84
6.5	The Equivalence of Theoretical and Practical Query Evaluation	89
6.5.1	Conjunctive Queries	89
6.5.2	Boolean Queries	94
6.5.3	Deletions, Permutations, and Value Changes	98
6.6	Optimizing Direct Query Evaluation	101
6.6.1	Compacting Query-Execution Plans	101
6.6.2	Dynamic Programming	103
6.7	Space Complexity of Query-Execution Plans	105
6.8	Related Work	106
7	Schema-Driven Query Evaluation	109
7.1	The Relationship between a Data Tree and its Path Tree	110
7.1.1	Path Trees and Node Classes	110
7.1.2	Using the Path Tree to find the Images of Data-Tree Embeddings	112
7.2	Finding Second-Level Queries in a Path Tree	115
7.2.1	Representing Embedding Images	115
7.2.2	k -Segments	116
7.2.3	Operators for Sets with Extended Node-Cost Tuples	116
7.2.4	Finding the Best k Second-Level Queries	119
7.3	Finding Results of Second-Level Queries in a Data Tree	119
7.4	An Incremental Algorithm for the Best- n -Results Problem	120
7.5	Optimizing Schema-Driven Query Evaluation	122
7.6	Related Work	126
8	Efficient Algorithms for Plan Operators	127
8.1	Compacting and Encoding a Target Tree	128
8.2	The Indexes of a Target Tree	130
8.3	Lists	131
8.4	Operations on Lists	132
8.4.1	Selection	133
8.4.2	Join and Outerjoin	133
8.4.3	Union and Intersection	138
8.5	Complexity Bounds for Operators and Query-Execution Plans	139
8.5.1	Time and Space Complexities of the Operators	140
8.5.2	Complexity Bounds for the Direct Evaluation Method	140

8.5.3	Complexity Bounds for the Schema-Driven Evaluation Methods	141
8.6	Related Work	142
9	The approXQL Query Engine	145
9.1	The Server-Side Modules	145
9.1.1	Database Kernel	146
9.1.2	Loader and Indexer	147
9.1.3	Query Parser	148
9.1.4	Query Processor	148
9.1.5	Abstract Generator	150
9.2	The Graphical Query Editor	151
10	Experimental Efficiency Analysis	153
10.1	Hypotheses	154
10.2	Experimental Setup	155
10.2.1	System Configuration	155
10.2.2	Software used for Experimental Setup	156
10.2.3	XML Collections	157
10.2.4	Query Patterns	158
10.3	Performed Experiments	159
10.4	Results of the Experiments	161
10.4.1	Hypothesis 1 (Number of Requested Results)	161
10.4.2	Hypothesis 2 (Number of Value Changes)	165
10.4.3	Hypothesis 3 (Schema Size, Varying Number of Names)	167
10.4.4	Hypothesis 4 (Schema Size, Constant Number of Names)	168
10.4.5	Hypothesis 5 (Term Selectivity)	170
10.5	Summary	172
11	Conclusion	175
11.1	Summary of Contributions	175
11.2	Future Work	178
	Bibliography	183
A	The Grammar of approXQL	199
B	Symbols used in the Thesis	201
C	Anhang gemäß Promotionsordnung	205