ISSN 2277-3061

Twig Pattern Minimization Based on XML Schema Constraints

Yuan Gao, Husheng Liao, Hongyu Gao
Beijing University if technology, China , Beijing,Chaoyang district 100124
s201307022@emails.bjut.edu.cn
Beijing University if technology, China , Beijing,Chaoyang district 100124
liaohs@bjut.edu.cn
Beijing University if technology, China , Beijing,Chaoyang district 100124
hygao@bjut.edu.cn

ABSTRACT

Twig pattern is one of the core components of XQuery. Twig usually includes redundancy nodes which can be optimized. Schema feature is used to judge whether the node of Twig pattern is redundancy. In this paper, we propose sufficient Schema constraints and specific rules. We have designed more determination conditions to optimize, then we will get the most efficient results. By a large number of test case, we finally get the practical limits of minimization.

Keywords

Twig pattern, Minimization, XML Schema constraints.

Academic Discipline And Sub-Disciplines

Computer Science and technology

SUBJECT CLASSIFICATION

Computer Subject Classification

TYPE (METHOD/APPROACH)

Analysis /Quasi-Experimental

Council for Innovative Research

Peer Review Research Publishing System

Journal: INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY

Vol. 15, No. 3

www.ijctonline.com, editor@ijctonline.com

ISSN 2277-3061

INTRODUCTION

XML (Extensible Markup Language) has become the standard description of network information and information exchange. XML data store in document, be searched by information retrieval method, such as the keyword query. Some existing commercial database systems extend capabilities of XML data processing, transform XML queries to database query expressions. XML query results obtained after execute it. This approach satisfy the requirements of complex query.

XQuery is a standard developed by W3C, used to extract information from an XML document. XQuery to XML is equivalent to the SQL to database. XQuery is built on XPath expressions to query XML data. A path expression as the core statement of the query. Core part of XPath and XQuery are generally extracted as Twig pattern, which can also be expressed as a tree structure. An XQuery that user writes often complicated, includes redundant parts. Twig minimization is a hot research topic and it is by deleting unnecessary portions of the query to improve query efficiency.

Our work as described in this paper makes the following contributions.

1)We propose a relatively complete set of Schema constraints. Include basic constraints and path constraints, provide more opportunities for minimize.

2)We propose Twig pattern minimization rules. Optimization both for leaf nodes and middle nodes, so that the query can be optimized faster and correct.

BACKGROUND

XML Schema

XML Schema provide a means for defining the structure, content and semantics of XML documents. One of the most important ability of Schema is to support data type, so that can easily describe the permissible document content, can easily verify the accuracy of the data.

The indicator node in the XML Schema document, as show in Figure 1, specifies the order and occurrence times of the elements. In XML Schema, judging elements' tag and indicators, according to their nest relation, we can determine whether they have inevitable relationship.

```
<xsd:schema targetNamespace="t" xmlns="t">
        <xsd:element name="book">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element ref="author"/>
                                <xsd:element ref="description"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name="author">
                <xsd:complexType>
                        <xsd:all>
                                <xsd:element name="age" type="xsd:string"/>
                                <xsd:element name="name" type="xsd:string" minOccurs="1" maxOccurs="3"/>
                        </xsd:all>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name="description" type="xsd:string"/>
        <xsd:attribute name = "isbn" type = "xsd:string"/>
</xsd:schema>
```

Fig 1: XML Schema document

Twig Minimization

Twig pattern minimization usually fall into two categories. The one is only by Twig pattern analysis; another is using constraints.

There is a Twig query in Figure 2 a). Search the book elements satisfied the conditions in the XML. The first condition is that book element must have child element with tag author, at the same time it must have child element with tag name. The second condition is that book element must have descendent element with tag name. Obviously, if the first condition be satisfied, the second condition must be satisfied. Namely, the second condition included in the first condition is redundancy which can be deleted. Result as show in Figure 2 b). This method is only based on Twig, without reference of other conditions.

Based on the analysis of XML Schema, if all the author element have descendent element with tag name, and all the book element have child element with tag name. The name element can be deleted, and result as show in Figure 2c). This method is based on the constraint conditions for check whether the node is redundant part in Twig. Obviously, twig c) can further reduce middle results of the query and detections of the XML, its efficiency is higher than twig b). So this paper based on Schema feature to minimize Twig pattern, finding highest efficiency minimization result.

ISSN 2277-3061

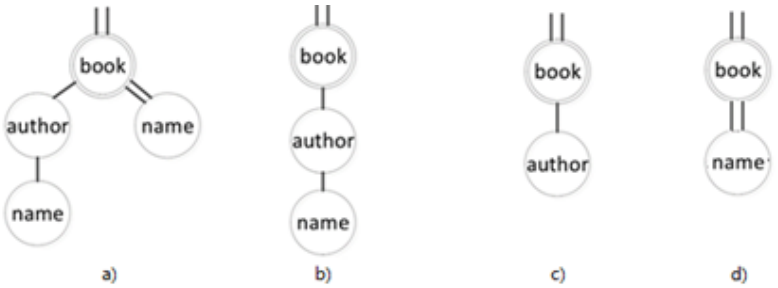

Fig 2: Twig model. Child edges are drawn as single lines, ancestor edges as double lines and returned node as double ring.

XML SCHEMA CONSTRAINT AND OPTIMIZATION RULES

XML Schema constraint

XML Schema can strictly define the structure of XML documents. The certain relationship, can extracted from XML Schema, between XML elements are called structural constraints (or feature relationship). This paper is mainly based on two kinds of constraints to minimize twig.

The basic constraints comprising: inevitable child, inevitable descendant, inevitable parent, inevitable ancestor; extended constraints comprising: Path inevitable child, Path inevitable descendant, Path inevitable parent, Path inevitable ancestor. For XML document *X*, which compliance with a given XML Schema *S*, the definitions of basic constraint are as follows:

Definition 3.1. If all the *a* elements in *S* have child element with tag *b*, then called *X* satisfy inevitable child constraint, recorded as *RPC (a, b)*.

Definition 3.2. If all the *a* elements in *S* have descendent element with tag *b*, then called *X* satisfy inevitable descendant constraint, recorded as *RAD (a, b)*.

Definition 3.3. If all the *a* elements in *S* have parent element with tag *b*, then called *X* satisfy inevitable parent constraint, recorded as *RCP (a, b)*.

Definition 3.4. If all the *a* elements in *S* have ancestor element with tag *b*, then called *X* satisfy inevitable ancestor constraint, recorded as *RDA (a, b)*.

Definition 3.5. If the *a* elements in *S* have descendent element with tag *b*, then called *X* satisfy descendant constraint, recorded as *MAD(a, b)*.

Assume that *p* is a XPath without predicate, the definitions of extended constraint are as follows:

Definition 3.6. If all the *a* element, reached from the final elements matched *p* in *S* through several element, must have child element with tag *b*, then called *X* satisfy path inevitable children constraint, recorded as *PPC(p, a, b)*.

Definition 3.7. If all the *a* element, reached from the final elements matched *p* in *S* through several elements, must have descendent element with tag *b*, then called *X* satisfy path inevitable descendent constraint, recorded as *PAD(p, a, b)*.

Definition 3.8. If all the *a* element, reached from the final elements matched *p* in *S* through several elements, must have parent element with tag *b*, then called *X* satisfy path inevitable parent constraint, recorded as *PCP (p, a, b)*.

Definition 3.9. If all the *a* element, reached from the final elements matched *p* in *S* through several elements, must have ancestor element with tag *b*, then called *X* satisfy path inevitable ancestor constraint, recorded as *PDA (p, a, b)*.

Definitions 3.10. If the a_1, a_2,…, a_n elements in *S* have a parent element with tag *b*, then called *X* satisfy inevitable common parent constraint, recorded as *RCF* $(b,a_1,a_2,...,a_n)$.

Definitions 3.11. If the a_1, a_2,…, a_n elements in *S* have an ancestor element with tag *b*, then called *X* satisfy inevitable common ancestor constraint, recorded as *RCA* $(b,a_1,a_2,...,a_n)$.

Optimization rules

Node in Twig are fall into query node, matching XML elements, and logical node, represents the logical relationship between various conditions. In this paper, we introduce query node optimization rules based on Schema features. Twig optimization not only upon leaf node but also middle node.

Leaf node optimization rules: suppose that *x* is a leaf node and *y* is a query node in Twig, *p* is an XPath expression without predicate, if *x* is not returned node and satisfying one conditions of the four groups on Twig structural and XML Schema features in Table 1, the node *x* may be deleted from the twig.

ISSN 2277-3061

Table 1. Leaf node optimization rules

number	Twig structural	XML Schema features
1	PC(y,x)	RPC(tag(y),tag(x))
2	AD(y,x)	RAD(tag(y),tag(x))
3	PC(y,x)	PPC(p,tag(y),tag(x))
4	AD(y,x)	PAD(p,tag(y),tag(x))

For example, if the constraint RPC (author, name) can be obtained from XML Schema. When optimizing the node name of twig in Figure 2 b), both it and its parent author are query node; it is a leaf node and not returned. Processing the Twig according to the rules in Table 1. We can find that first line have been satisfied, the node name can be deleted, result Twig in Figure 2 c).

The middle node optimization rules:

(1) suppose that *y* is a middle node, *x* is a query node in Twig, *p* is an XPath expression without predicate and *z* is the only descendent node of *y*. If *y* is not returned node and satisfying one conditions of the eight groups on Twig structural and XML Schema features in Table 2, the node *y* may be deleted from the twig and the node *z* may connect to the node *x* using double lines.

(2) suppose that *y* is a middle node, *x* is a query node in Twig, *p* is an XPath expression without predicate and z_1, z_2,…, z_n are descendent nodes of *y*. If *y* is the only child node of *x* and not returned node, satisfying one conditions of the four groups on Twig structural and XML Schema features in Table 3, the node *y* may be deleted from the twig and the nodes z_1, z_2,…, z_n may connect to the node *x* using double lines.

Table 2 Middle node optimization rules

No.	Twig structural	XML Schema features
1	AD(x,y) AD(y,z)	RAD(tag(x),tag(y)) !MAD(tag(y),tag(x)) RDA(tag(z),tag(y))
2	PC(x,y) AD(y,z)	RCP(tag(y),tag(x)) !MAD(tag(x),tag(x)) RDA(tag(z),tag(y))
3	AD(x,y) PC(y,z)	RAD(tag(x),tag(y)) RCP(tag(z),tag(y))
4	PC(x,y) PC(y,z)	RCP(tag(y),tag(x)) !MAD(tag(x),tag(x)) RCP(tag(z),tag(y))
5	AD(x,y) AD(y,z)	PAD(p,tag(x),tag(y)) !MAD(tag(y),tag(x)) PDA(p,tag(z),tag(y))
6	PC(x,y) AD(y,z)	PCP(p,tag(y),tag(x)) !MAD(tag(x),tag(x)) PDA(p,tag(z),tag(y))
7	AD(x,y) PC(y,z)	PAD(p,tag(x),tag(y)) PCP(p,tag(z),tag(y))
8	PC(x,y) PC(y,z)	PCP(p,tag(y),tag(x)) !MAD(tag(x),tag(x)) PCP(p,tag(z),tag(y))

For example, if the constraint RPC(book, author), RCP (name, author) and !MAD(author, author) can be obtained from XML Schema. When optimizing the node author of twig in Figure 2 b), both it and its parent author are query node; it is a middle node and not returned; it have single child node name. Processing the Twig according to the rules in Table 2. We can find that fourth line have been satisfied, the node author can be deleted, and the node name can connect to the node book using double lines, result Twig in Figure 2d).

Table 3 Middle node optimization rules.

No.	Twig structural	XML Schema features
1	AD(x,y) AD(y,z_i) (i=1,2,…,n)	RAD(tag(x),tag(y)) RCA (tag(y),tag(z_1),…,tag(z_n)) !MAD(tag(y),tag(x))
2	PC(x,y) AD(y,z_i) (i=1,2,…,n)	RCP(tag(y),tag(x)) RCA (tag(y),tag(z_1),…,tag(z_n)) !MAD(tag(x),tag(x))
3	AD(x,y) PC(y,z_i) (i=1,2,…,n)	RAD(tag(x),tag(y)) RCF (tag(y),tag(z_1),…,tag(z_n))
4	PC(x,y) PC(y,z_i) (i=1,2,…,n)	RCP(tag(y),tag(x)) RCF (tag(y),tag(z_1),…,tag(z_n)) !MAD(tag(x),tag(x))

ISSN 2277-3061

TEST CASE AND TEST RESULTS

Experiment carried out with different size XML Schema documents and different types of Twig pattern. The Twig patterns which before and after optimization were executed in the 82M XML document using TwigList algorithm. If the number of Twig nodes after optimizing reduce with large degree, the minimization results are obvious. The optimization time and query time after optimizing divide the query time without optimizing. The smaller the ratio, the higher the efficiency of the optimization. If the ratio is greater than 1, the optimization algorithm is meaningless. The test results as shown in Table 4.

Table 4. test results(number of Twig nodes recorded as TN, time of query record as QT)

No.	Schema nodes	Returned nodes	TN before optimization	TN after optimization	QT before optimization	Optimization time	QT after optimization
1	10	1	4	3	242ms	28ms	208ms
2	10	1	6	4	356ms	54ms	257ms
3	10	2	4	4	242ms	12ms	242ms
4	10	2	6	5	356ms	46ms	298ms
5	20	1	4	3	242ms	35ms	208ms
6	20	1	6	4	356ms	62ms	257ms
7	20	2	4	4	242ms	24ms	242ms
8	20	2	6	5	356ms	51ms	298ms

CONCLUSION

The two aspects problem from the data structure and the query needs led to XML query optimization. In this paper, twig minimization based on the constraints from XML Schema, which are relatively complete. On the basis of the necessary constraint, the path constraint are added, which provide more opportunities for optimization. In this paper, we also propose specific rule that is optimized for different kinds of nodes. When optimizing node according to the specific rules and the corresponding algorithm. By a large number of test case, the conclusion is drawn that the minimization cost is greater and the result is better when the Schema nodes number is far greater than the Twig nodes number; the minimization cost is smaller but the result are not obvious when the Twig contains a large number of returned nodes.

ACKNOWLEDGMENTS

Supported by “Study on Automatic Program Instantiation When XQuery Running” of Youth Fund Project of Natural Science Foundation of China (NO. 61202074).

REFERENCES

[1] G. Miklau and D. Suciu. Containment and Equivalence for a Fragment of Xpath[J].(J. ACM. January 2004. 2004, 51(1): 2-45.)

[2] C. Chan, W. Fan and Y. Zeng. Taming Xpath Queries by Minimizing Wildcard Steps[C]. (VLDB, 2004. VLDB Endowment, 2004：156-167.)

[3] http://www.w3school.com.cn/

[4] Sihem Amer-Yahia, SungRan Cho, Laks V.S.Lakshmanan, Divesh Srivastava . Minimization of Tree Pattern Queries. (ACM SIGMOD 2001 May 21-24,Santa Barbara, California, USA Copyright 2001 ACM 1-58113-332-4/01/05 ...5.00.)

Author’ biography with Photo

Yuan Gao, Master, the main research field for the XML database technology.

Husheng Liao, Ph.D., professor, main research field for the automation of software methods, compiler technology, XML data processing, program analysis and transformation.

Hongyu Gao, Master, the main research field for the automation of software methods, compiler technology.