Query Processing for XML Data Stored in a Relational Database

A Masters Thesis Proposal
Bhaskar R. Rathod
with Prof. Paul Helman
Computer Science Department, The University of New Mexico
(bhaskar@cs.unm.edu)
June 27, 2004

Abstract

Semi-structured data is gaining a wide acceptance in many application domains due to its standardization into XML. Relational database systems have a rigorous theory and mature implementation practices behind them. Storing XML in relational systems may allow improved query processing performance due to mature relational practices of indexing, storage techniques and query optimization primitives. With the advent of several XML to relational transformation techniques, inherently hierarchical XML data can be stored into relational systems. XML Schema and XML documents can be queried using XML Query (XQuery) language and XML Path (XPath) expressions. The problem of transforming XML query expressions into Structured Query Language (SQL) expressions is an active area of research. We want to study query processing and optimization techniques for XML data stored in a relational storage. Specifically, we want to study the impact of knowledge of relational schema and mapping information for query optimization of such stored XML data in relational systems.

1 Introduction

Due to the advent of the web, there is an explosion of information availability. This information can be automatically generated by applications or can be human-coded. Web-enabled applications can consume, process, transform and generate information that can be human-understandable as well as that can be used for data exchange among applications. Such information is called semi-structured data (ssd) due to a varying degree of structure in such information. In the domain of semi-structured data, eXtensible Markup Language (XML) is a major data representation as well as data exchange format. XML is a W3C specification [Wor04] that allows creation, transformation and validation of a semi-structured document conforming to meta data of the document. Such metadata can either be in the form of Document Type Definition (DTD) or XML schema. The DTD is a BNF-like grammar, while an XML schema document is the description of metadata in XML format. The metadata information of XML documents enables automated processing, generation, transformation and consumption of semi structured information by applications.

With a widespread availability and an extensive research base, Relational Database Management Systems (RDBMS) are a very popular choice for data storage, transformation and retrieval. Semi-structured data, for
which we will consider XML as a representative technology, provides very convenient data interchange and presentation techniques through various W3C specifications. Thus, an amalgam of these two technologies has the potential to provide many benefits and hence is a topic of extensive research.

1.1 XML and Relational Database Systems

While Relational Database systems are backed by a rigorous formalism and relational data present a high level of structure and regularity, XML data may not have an a-priori structure; it may often be self-describing, irregular, hierarchical, and at times, may have a recursive structure.

Relational schemas have a relation as a basic building block. A relation is a flat structure, possibly derived using normalization. An XML schema has an XML document as a basic building block. An XML document is derived using natural, hierarchical and often recursive relationships among the objects of a domain. XML schema captures relationships among data in the ancestor-descendant like edge structure, which can be represented as a directed graph with multiplicity information on the edges. Relational schemas use key constraints and relationship tables to capture relationship among data. XML Schema allows user defined types for XML elements in the instance, whereas RDBMS mainly allow pre-defined basic types and do not greatly extend relational primitives for user defined types.

Naturally, these are fundamentally different data models. XML storage in relational systems pose significant challenges in schema mapping and query processing.

Many XML to relational schema translation techniques have been proposed in the recent past that claim lossless schema and data transformation. Many query translation algorithms claim XML Query [Wor03] to Structured Query Language translation. Noticeable among them are [KCKN04] and [DTCO03]. [KCKN04] first carries out XML to relational schema translation using schema-based shredding and then performs query translation based on the mapped relational schema. [KCKN04] also assumes implementation of WITH clause in SQL.

SQL’s WITH clause is a subquery factoring clause and is part of SQL-99 standard. An SQL query using WITH clause starts with the keyword “WITH” such that WITH is followed by the subquery definition and a label for the resulting cursor. Aparently, this cursor label can be queried upon allowing sophisticated querying.

[DTCO03] performs interval encoding on XML schema while evaluating input XQuery and generating XML fragments.

1.2 Research Opportunities

For XML data stored in relational systems, the focus in database research has shifted to the fields of [KKN03]:

1. Bi-directional data and query translation between relational data and XML.
2. Data transportation using XML.

The former provides many exciting research opportunities. Some of them can be summarized as follows:

1. Deriving equivalent relational model interpretations for possibly recursive XML data.
2. Transforming an XML query [Wor03] over an XML schema into appropriate query for a relational query engine over its mapped relational schema.
3. Translating relational results into schema-conforming XML fragments. This is interesting because it may involve type coercion, regeneration of hierarchies conforming to XML metadata and valid XML syntax of the results.

We are interested in general translation of XQuery expressions into equivalent SQL expressions and later in this proposal we discuss a specific problem of interest in the section “Problem Discussion”.

2 Background

To enable a better understanding of the proposed research, we very briefly describe various technologies of interest in this section.

2.1 XML

Extensible Markup Language (XML) is defined by [Wor04] as:

XML is an application profile or restricted form of SGML, the Standard Generalized Markup Language. By construction, XML documents are conforming SGML documents.

An XML document describes some structured information about a relatively higher level object in terms of the described object’s attributes and objects it contains. This information is built up of entities called Elements. An element may contain attributes and other elements in a hierarchical manner. Some elements in this hierarchy may create recursions by direct or indirect references to themselves.

An XML document conforms to its metadata. We will concentrate on XML schema documents as the metadata format. An XML schema can specify hierarchy, order, type, and attributes for all the elements of an XML document. XML schema specification[Wor01] allows defining custom types and building types recursively using them. Each document mentions the name and location of one or more XML schemas it conforms to. When an XML document conforms to multiple schemas, element names are resolved using the concept of namespace. A namespace is a unique alias for a schema in the context of an XML document, eg. namespace : element distinctly resolves the element names. Listing 1 shows a fragment of an XML Schema document. Note the custom type definition called SimpleLiteral and its use in type specification of other elements. Line numbers are not part of XML Schema, and have been used here for easy referral.

1. <?xml version="1.0" encoding="UTF-8"?>
2. <xs:schema
3.  targetNamespace="http://purl.org/dc/elements/1.1/
4.  xmlns:xs=http://www.w3.org/2001/XMLSchema
5.  version="1.00.002">
6.  
7.  <xs:complexType name="SimpleLiteral" mixed="true">
8.  <xs:simpleContent>
9.    <xs:extension base="xs:anySimpleType">
10.       <xs:attribute ref="xml:lang" use="optional"/>
Listing 1: An Example XML Schema

An XML element has a start tag and an end tag. The content of the element is in between. As it may be seen in the example, metadata of an XML document is scattered in the structure of the document as well as in tag attributes. This structural property, along with recursion and hierarchy in structure, may contribute to the difficulty of the schema and query translation problems.

Listing 2 is an example of an XML document. The document describes a fragment of XML for an archived image at the UNM General Library. The document conforms to a standardized schema called Dublin Core v.1. Note that this document contains a namespace alias called dc. This schema is mapped to a physical schema file location that contains the Dublin Core v.1 schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<qualifieddc xmlns:dc="http://purl.org/dc/elements/1.1/"
    SchemaLocation="qualifieddc.xsd">

    <dc:title>
        Hauling machines to the U.S. Treasure Mine
    </dc:title>

    <dc:creator>
        Schmidt, Henry A.
    </dc:creator>

    <dc:subject>
        Mining machinery
    </dc:subject>

    <dc:description>
        Horse-drawn wagons loaded with machinery bound for the U.S. Treasury mine; men and boys, some quite young children, in and around wagons; buildings visible on either side of the dirt roadway.
    </dc:description>

</qualifieddc>
```
Listing 2: An Example XML Document

2.2 XQuery

XQuery [Wor03] is a query language for XML data. XQuery enables FLWOR (For, Let, Where, Order By, Return) expression queries on an XML document. A general model of FLWOR expression in XQuery is minimally defined as [Wor03]:

\[
\text{FLWORExpr} ::= (\text{ForClause} \mid \text{LetClause})
\]

+ \text{WhereClause}?
+ \text{OrderByClause}?
+ “return” \text{ExprSingle}

where For, Let, Where, OrderBy and return clauses above have their intuitive and usual meanings and \text{ExprSingle} is defined in core XQuery Grammar as:

\[
\text{ExprSingle} ::= \text{FLWORExpr}
\]

+ | \text{TypeswitchExpr}
+ | \text{IfExpr}
+ | \text{OrExpr}

where If and Or expressions have their intuitive and usual meanings. The \text{Typeswitch} expressions can be thought of as type coercion expressions. An expression is a path in a schema graph of the XML document being queried.

As may be seen, FLWOR expressions make XQuery an extensive and comprehensive querying technique. This property of XQuery makes it a challenge to devise efficient, equivalent non-procedural SQL expressions, as a FLWOR expression can often result in a set of complex and separate SQL expressions instead of a single declarative SQL query. Also, the resultant SQL query may need to use multiple joins to achieve the same results as a query in XQuery. Listing 3 describes example XQuery and its SQL translation over an imaginary schema.
SELECT NAME 
FROM EMPLOYEE;

A Simple XQuery Expression and its SQL equivalent

<manager name = "JOE">
  FOR $myNode in document("employees.xml")//employee 
  WHERE $myNode/manager="JOE" 
  ORDER BY $myNode/desig 
  RETURN 
  <employee>
    <name>$myNode/name</name>
  </employee>
</manager>

SELECT E1.NAME 
FROM EMPLOYEE E1, EMPLOYEE E2 
WHERE E1.MANAGER = E2.EMP_ID 
AND E2.NAME = "JOE"

A FLWOR expression query and its SQL equivalent

Listing 3: Example XML Queries

2.3 Query Processing in Relational Systems

Query Processing in relational systems is essentially an optimization problem of finding a least cost query execution plan that can generate correct results when executed against a relational schema and its instance. Query Processing in Relational Systems can be decomposed in various levels:

- Parsing an SQL query into an equivalent relational algebra expression containing relational algebra operators, query parameters and references to relational schema.

- Transforming a relational algebra expression into an equivalent expression that is less costly in the context of underlying relational schema and instance compared to other equivalent expressions.

- Transforming a relational algebra expression into a query execution plan that incorporates physical implementation operators for relational algebra operators in the expression.

In the context of XML data stored in relational systems, query processing may provide an interesting exercise in transformation at various levels. An XML schema can be queried using XQuery as mentioned above. Many algorithms claim to achieve XQuery-to-SQL translation. Such a translation may usually yield
an SQL query with a high number of joins. As [KCKN04] implies, such a translation may not be independent of schema transformation scheme used. In the light of this discussion, a mapping-aware query optimization may be an interesting idea to pursue.

Another interesting idea is to study the impact of storage structure on query processing for XML data stored in relational systems. It is duly noted here that [Kri04] has attempted a mapping-aware XQuery-to-SQL query translation and our belief is that their research may be categorized at a logical relational operator-level optimization. Our interest is in studying physical implementation of operators and storage structure for query optimization of XML data stored in relational systems.

3 Problem Discussion

It is our conjecture that the knowledge of XML summarization, relational summarization, query statistics and schema mapping information of XML data stored in a relational system may provide a useful heuristic for pruning the search space of query processing on such data. In this section we propose a query complexity metric for SQL queries, we propose a mapping between XML summarization and relational summarization of an instance based on independent research on XML and relational synopsis of data. Finally we propose a heuristic for determining cost of a query plan based on these factors.

3.1 Proposing an SQL Query categorization Metric

Extending query processors for stored XML data can benefit from categorization of queries. Such categorization may help determine a heuristic for path determination in query plans space. In [Rat04], we have outlined a scheme for such a metric. The following is a brief discussion on the proposed scheme.

Query cost in a relational query processor is calculated using the cost of disk access and I/O operations as a unit. As discussed, XQuery FLWOR expressions, when transformed into SQL, result in costly SQL queries. Many of these queries involve multiple columns together, while others involve multiple rows together in order to achieve final results. The statistical distribution of the degree of involvement of rows and columns in a representative set of queries may provide important insights in query optimization. Based on the underlying storage structure and this distribution, the I/O cost estimation of a query may be ranked. Based on such a rank, the query processor can take many important decisions like selection of physical implementation operators and caching, and derive useful selectivity statistics on relations and attributes.

Such a metric can be created by analyzing relational operators and their cost information of available physical implementation operators. Such a metric can be useful for cost estimation in relational query processing also. Based on practical experience, it is our conjecture that XML data stored in a relational system tends to scatter over many relations; which may not always be the case with relationally derived schemas. It is our belief that a metric that can take this into account might provide a better cost estimation for XML data stored in a relational system.

We have outlined an approach to analyzing relational operators in [Rat04]. We formally define the terms used in [Rat04] here:

- **Relation**
  A relation \( r \) is a base relation or a derived relation in a given relational schema \( R \). The relation \( r \) can also be an intermediate result relation during query processing over the given relational schema \( R \).
• **General Operation**
  A general operation is a primitive relational algebra operation over a relation \( r \).

• **Dense Tuple Operation**
  If a relation \( r \) contains \( k \) attributes, then a general operation \( o \) over \( r \) retrieves or changes at least \( c \) attributes in the schema of \( r \) either in intermediate results, or in final results or in the both, such that
  \[
  \frac{c}{k} \geq \delta
  \]
  where \( \delta \) is called **tuple density constant**, whose value can be fixed by considering query cost.

• **Sparse Tuple Operation**
  A general operation that is not a Dense Tuple operation.

• **Dense Attribute Operation**
  If a relation \( r \) contains \( l \) tuples, then a general operation \( p \) over \( r \) retrieves or changes at least \( d \) attributes in the schema of \( r \) either in intermediate results, or in final results or in the both, such that
  \[
  \frac{d}{l} \geq \eta
  \]
  where \( \eta \) is called **attribute density constant**, whose value can be fixed by considering query cost.

• **Sparse Attribute Operation**
  A general operation that is not a Dense Attribute operation.

Based on above definitions, queries can be classified on the distribution of dense and sparse operations. The following table describes categories for a general operation:

<table>
<thead>
<tr>
<th>Operations under ( \delta ) and ( \eta )</th>
<th>Dense Attribute ((^D))</th>
<th>Sparse Attribute ((^S))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dense Tuple ((^D))</td>
<td>DD</td>
<td>DS</td>
</tr>
<tr>
<td>Sparse Tuple ((^S))</td>
<td>SD</td>
<td>SS</td>
</tr>
</tbody>
</table>

*Table 1: Categories for a General Expression*

A query may be analyzed for the distribution of general operations as listed above. Based on this analysis, the query may be categorized. We want to formalize and experiment with the metric for relational operators as described in this section. We want to use this metric to experiment with XQuery expressions translated into SQL for XML data stored in a relational system. We elaborate more on this in the proposed thesis work section.

### 3.2 Synopsis

A Summary or a Synopsis of a database instance is a collection of structures that captures important statistical insights of the database instance. Synopses can provide information like selectivity estimation and approximation of results to a query processor. Such information may facilitate improvements in query plan generation. For relational systems, synopses can be based on ranges of values. As it may be seen intuitively, a relational database instance can be summarized as median or mode values of attributes, range representations etc and are captured in structures like histograms, wavelets, etc.
For XML documents, XSKETCH [PG02] creates histograms of distribution summaries. [PG02] observes that building XSKETCH summaries is an NP Hard problem and proposed a greedy, forward selection heuristic for building XSKETCH synopses of XML data.

The study of synopses of relational and XML data is another interesting area. Such a study can involve the creation, maintenance and possible inter-relationships between relational and XML synopses of XML data stored in a relational database system, and their impact on query processing of XQuery expressions transformed into SQL against the relational data store.

4 Proposed Thesis Work

In this section we outline thesis goals, major tasks and a thesis plan based on these goals and tasks.

4.1 Goals

The intended goal of this thesis can be enumerated as follows:

1. To study in depth the relational query processing mechanism.
2. To implement and experiment with XML-to-relational schema transformation algorithms and XQuery-to-SQL transformation algorithms.
3. To develop and experiment with query categorization metric as outlined in previous sections.
4. To study relational and XML synopsis techniques.
5. To study the combined impact of the knowledge of synopses, schemas and schema transformation on query processing for XML data stored in a relational system.

4.2 Major Tasks

In this section, we discuss tasks for each goal mentioned in the previous section. A task is a specific outline that can be used to measure the progress towards a goal.

1. Goal: To study in depth the relational query processing mechanism.
   Tasks:
   (a) To understand schematically relational query processing, relational algebra operators and physical implementation operators.
   (b) To survey query optimization schemes.
   (c) To study query optimization implementation in PostgreSQL 7.1, MySQL 5.0.

2. Goal: To study-XML-to relational schema transformation algorithms and XQuery-to-SQL transformation algorithms.
   Tasks:
   (a) To study XML-to-Relational schema transformation technique of LegoDB [BFH+02] and Inlining [STH+99].
(b) To select a representative XML Schema and XML data for transformation into Relational schema and instance.

c) To transform selected XML schema and instance into Relational schema and instance.

d) To study XQuery-to-SQL transformation algorithms of Dynamic Interval encoding and [KCKN04].

e) To select representative XQuery expressions over selected XML schema and instance, and transform them to SQL using one of the XQuery-to-SQL transformation algorithms.

3. Goal: To study query categorization metric as outlined in previous sections
   Tasks:
   (a) To formalize the definition of the metric.
   (b) To categorize transformed queries based on the metric. Such a categorization involves deriving distribution of general operations in a query from its relational algebra representation.

4. Goal: To study relational and XML synopses techniques and their inter-relationships
   Tasks:
   (a) To study histogram-based XSKETCH XML synopsis technique in the context of our selected XML schema and instance, and generate a relational synopsis.
   (b) To study a histogram-based relational synopsis technique in the context of our transformed relational schema and instance, and generate an XSKETCH synopsis.

5. Goal: To study the combined impact of the knowledge of synopses, schemas and schema transformation on query processing for XML data stored in a relational system.
   Tasks:
   In view of the main theme of studying cost minimization of XQuery processing on a relational system, the tasks for this goal can be listed as follows:
   (a) To study appropriate knowledge representation scheme for synopses and schemas such that the query processor can benefit from their knowledge.
   (b) To propose a cost minimization algorithm in the context of above.
   (c) to experiment with XMark benchmark for XML data stored in a relational database system using XQueries and transformed SQL queries on relational and XML schema and instances.

4.3 Proposed deliverables and an Approximate Schedule

Here, we outline proposed deliverables and approximate schedule for the proposed thesis work based on the goals and tasks discussed above. On a timeline, the approximate schedule would be carried out in the order described below. It is our estimate that along with a full-time course load, the proposed work should take about 20 weeks, not counting documentation. Subject to the approval of this proposal and active work starting in Fall 2004 semester, we believe that the intended study can be finished by mid-semester in Spring 2005.
1. Analysis, design and implementation of cost estimation metric for an SQL query as outlined in this proposal. This would involve:

(a) Developing an approach for analysis of input relational schema and SQL query posed over it.
(b) Studying implementation approaches for various relational operators and integrating them in the analysis.

Given an SQL query and relational schema over which the query is posed, our implementation should categorize the query. Given a set of SQL queries and relational schema over which the queries are posed, our implementation should produce a statistical profile of each logical relational operator describing possible performance of its various physical implementations.

Approximate Schedule: 6 weeks.

2. Analysis and implementation of Xpath-to-SQL query translation algorithms proposed in [KCKN04] and [STH+99].

Approximate Schedule: 3 weeks, considering current progress.

3. Analysis and experimentation with generation of XML and relational data synopses of XMark dataset in XML format as well as its transformed relational format.

Approximate Schedule: 5 weeks.

4. Experimenting our metric with respect to the following on XMark benchmark:

(a) XML-to-relational schema translation algorithms proposed in [ABS99].
(b) XPath-to-SQL query translation algorithms proposed in [KCKN04] and [STH+99].

Approximate Schedule: 6 weeks.

This should hopefully provide us enough insights about the patterns in queries for XML data stored in relational systems and their relationships with physical implementation approaches in relational systems. These insights might be useful for relational systems, extensions to relational systems as well as native XML database systems using relational query processors.

References


