Different types of XML compressor

(Group-6, Clicker Optimization)

Under guidance of Prof D B Pathak

IIT bombay

CSE department

Anoop Dobhal

31, july,13

Introduction

There are different types of compressor are available for compressing XML document. XML compression is needed because

1. XML is medium of exchange of information and storage, especially over the internet.
2. XML store schema for every record in the document, so size of document increased greatly.
3. In recent years XML document size increased by 400%.

Compressor used for XML compression are

1. Gzip
2. XMill
3. XGRIND
4. Delta
5. NATIVE
**XMill**

It groups semantically related data items into different containers. Then separately compress each container with a compressor ideal for them. For example, `<name>` data item is in one container and `<phone number>` in one container. XMill provides better performance than gzip. XMill does not support queries or updates on compressed documents. So for querying or updating on the document, it first decompressed the document then apply operations.

So the idea to support query-friendly compressor is XGRIND.

XMill operates as

1. First metadata in the form of tags and attributes is compressed separately from data values.
2. Semantically related data items are compressed separately.
3. Then different container compressed separately by using different compressor.

**XGRIND**

It supports direct prefix-match, exact-match queries on compressed documents. But range queries need a document to be decompressed before applying the query to it. Only that part of the document is decompressed which is between range-match, not all the document.

Compression scheme used by XGRIND is “Huffman coding”. The most advantage of XGRIND compressor is

1. It retains structure of the original document in the compressed format also.
2. Compressed document can be parsed in same way as original document.
3. Indexing is also possible on compressed document.
4. Update are possible on compressed document.
5. Validity of compressed document can be checked against the compressed DTD (Document type declaration).

XGRIND use different compressing technique for meta data, enumerated-type attribute values and element/attribute values.

Compression example

The student document contain five elements: STUDENT, NAME, YEAR, PROG, DEPT. STUDENT has an “rollno” attribute and DEPT have “name” attribute of enumerated type. Nahuff(s) denotes the huffman-compressor for an input data value s, while enum(s) denotes the output of the enum-encoder for the input data value.

<!- student.xml -->
<STUDENT rollno = 604100418>
    <NAME>rajesh kumar</NAME>
    <YEAR>2000</YEAR>
    <PROG>Master of Engg</PROG>
    <DEPT name = “Computer Science”>
</STUDENT>

(Fragment of Student DB)

<!- DTD for the student database -->
<!ELEMENT STUDENT (NAME, YEAR, PROG, DEPT)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT YEAR (#PCDATA)>
Query Processing

The compressed-domain query processing engine consists of a lexical analyzer that emits tokens for encoded tags, attributes, and values, and a parser that does the dumping and matching of records. For **exact-match** or **prefix-match** queries, the query path and the query predicate are converted to the compressed-domain equivalent. During parsing of the compressed XML document, when the parser detects that the current path matches the query path, and that the compressed data value matches the compressed query predicate, it outputs the matched XML fragment.

For **range** or **partial-match** queries, only the query path is compressed.
While parsing the compressed XML document, when the parser detects that the current path matches the query path, the associated data value is decompressed and used for evaluating the match. This decompression is required since the compression scheme we use is not “order preserving”.

Query example in XGRID

```
CONSTRUCT <student rollno = $r> {
    WHERE
    <student rollno = 123050062>
    <name>$n</name>
    <year>$y</year>
    <dept name = $d>
    </student> IN “student.xml”
    CONSTRUCT <name>$n</name>
} </student>
```