2.2 Twig Pattern Query

Based on the containment labeling scheme, prior work decomposes a twig pattern into a set of binary relationships, which can be either ancestor –descendant relationships or parent–child relationships. Then, each relationship in binary is processed using structural join techniques and the final match results are obtained by "merging" individual binary join results together. The query processing of a core subset of XML query languages [11], XML twig queries. An XML twig query [2], represented as a small query tree is essentially a complex selection on the structure of an XML document.

The twig pattern query [19] is a core operation in XML query processing and popularly used as it can represent complex search conditions [12]. Fig. 2 shows two example twig pattern queries Q2 and Q3 with their tree structure. Cities located in the provinces of a country found by Q2 that has a child node "name" whose text content is "Belgium," and Q3 is to find cities in provinces located at "Middle" Matching a twig query [19] means finding all the instances of the query tree embedded in the XML data tree [17].

Finding all the occurrences of a twig pattern specified by a selection predicate on multiple elements in an XML document is a core operation for efficient evaluation of XML queries. A typical approach decompose the pattern into a set of binary structural relationships (parent- child or ancestor-descendant) between pairs of nodes then match each of the binary structural relationships against the XML database and finally stitch together the results from those basic matches.

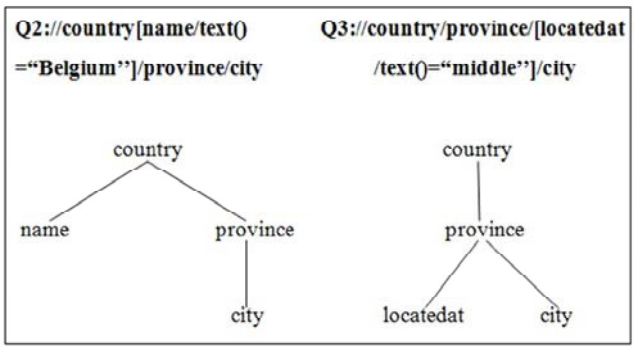

Figure 3: Example twig pattern queries and their tree representation.

2.3. The Structure Index

In this section, we introduce the Structure Index 13], which is document structure derived from the input XML documents. It can be used to preprocess the structure navigation part of XPath queries. At runtime, the Structure Index can be used to efficiently find queries that match a given XML document [10] by traversing its structure, and perform additional computation for predicate evaluation. Structure Index [20] is constructed by representing each element or attribute node as a node in a Structure Index called *Index Node*. The relationships between Index Nodes are the same as parent-child relationship in nodes of document structure. This index will be used (as an XML document tree) to preprocess the structural navigation part of XPath queries. It will attempt to find all possible ways that the structural parts of the queries can be matched with this XML document tree. Subsequently, each result from structure matching is stored at the Index Node that it matches. In XML indexing [20] techniques [14], XML indexes can be regarded as summary of XML source documents and thus has much smaller sizes compared to the original source.

3. Wireless XML Data Broadcasting

Broadcast is one of the basic ways of information access via wireless technologies. The server broadcasts information to all mobile devices in a wireless data transmission system within its transmission range via a downlink broadcast channel. Clients "listen" to the downlink channel and access information of their interest directly when related information arrives. Broadcast [3] is bandwidth efficient because all clients can share the same downlink channel and retrieve data from it simultaneously. Broadcast stream [15], [1], [3] is also energy efficient at the client ends because downloading data costs much less energy than sending data. In this work we focus on the periodic broadcast mode since it has many benefits such as saving uplink bandwidth and power at the client ends by avoiding uplink transmissions and effectively delivering information to an unlimited number of clients simultaneously.

3.1 G-Node

This paper proposes a wireless XML stream by integrating information of elements of the same path. In this project G node [9] is used for streaming XML data in the wireless environment. That is the XML data stream consists of the sequence of group nodes called G-node. A streaming unit of a wireless XML stream, called G-node. The G-node structure remove structural overheads of XML documents and enables clients to skip downloading of irrelevant data during query processing. Fig.3 illustrates the G-node structure that integrates elements of the path "/mondial/country/province". The group descriptor (GD) is a collection of indices for selective access of a wireless XML stream. Node name provide the tag name of integrated datas, and Location path of integrated elements from the root node to the element node in the XML document tree structure provided by the Xpath expression. Child Index (CI) is a set of addresses that direct to the starting positions of child G-nodes in the wireless XML stream. Attribute Index (AI) contains the Couples of attribute name and address to the starting position of the values of the attribute that are stored contiguously in attribute value list. Text Index (TI) is an address directing to the starting position of Text List. In this scheme, an address means a point in time when the relevant data is broadcast on the air. The components of the GD are used to process XML queries in the client efficiently. Generally, node name and location path are used to identify G-nodes. Indices such as CI, AI and TI are used to selectively download the next G nodes, attribute values, and text. Attribute Value List (AVL) store attribute values and Text List (TL) store text contents of the elements by the G-node. Attribute values and text contents are collected in document order of elements.

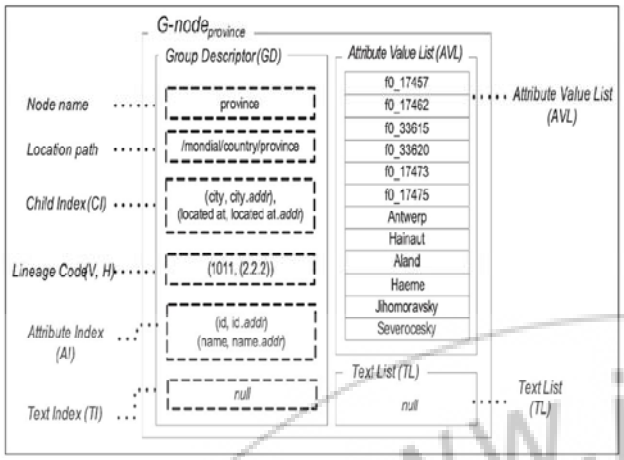

Figure 4: Structure of an example G-node.

3.2 Attribute Summarization

Attribute Summarization [16] technique is used to reduce the size of a wireless XML stream and that eliminates repetitive attribute names in a set of elements when generating a stream of G-nodes. In this project we exploit the benefits of the Attribute Summarization technique [11] which is used to reduce the size of a wireless XML element stream. An element in XML document may have multiple attributes, each attributes consists of a name and value pair. There is a structural characteristic that element with the same tag name and location path often contain the attributes of the same name. During the generation of G-nodes stream Attribute Summarization eliminates repetitive attribute names in a set of elements.

4. Lineage Encoding

This work introduces a novel encoding technique, called Lineage Encoding, to permit queries involving predicates and twig pattern matching. In the proposed system, two kinds of lineage codes, i.e., vertical code denoted by Lineage Code (V) and horizontal code denoted by Lineage Code (H) are used to represent parent-child relationships among XML elements in two G-nodes.

4.1 Lineage Code

This paper provide a light-weight encoding scheme, called Lineage Encoding [9] to represent parent-child relationships among XML elements in the G-nodes. We propose applicable functions and operators that apply bit-wise operations on the Lineage codes. Our scheme is the first wireless XML streaming approach that completely supports twig pattern query processing in the wireless broadcast environment. The Lineage Code scheme encodes parent-child relationships between two sets of elements in two G-nodes based on light-weight and efficient bit string representation. Fig. 4 demonstrates an example of Lineage Codes in G node country, G-node province, and G-node city. Note that Lineage Code (V) of G-node province is defined by 1,011 since the elements collected in G-node province are mapped to only the first, third, and fourth elements in G-node country. Lineage Code (H) of G-node province is (2, 2, 2) where each value represent the number of child elements in G-node province mapped to the same parent element in G-node country in document order.

4.2 Wireless XML Stream Generation

In wireless XML stream generation [14] we explain how to produce wireless XML element stream [21]. A server gets an XML document to be broadcasted from the XML repository and it generates wireless XML steam by using SAX (Simple API for XML) [17], which is an event-driven API. During the parsing of an XML document SAX invokes content handlers. Then streaming of XML data streamed XML data are disseminated via a broadcast channel.

5. Experiments and Results

5.1 Experimentation

An experiment is conducted to measure energy and latency efficiency for the entered query by using novel unit structure called G-node. It includes advantages like structure indexing and attribute summarization that can combine relevant XML elements into a group. It provides a way for selective access of their attribute values and text content.

Table 5.1: Test X-Path Queries on the catalog Data Set.

#	Test Query
Q1	/catalog/book/Authors
Q2	/catalog/book/Authors[text()=”yashwanth”]
Q3	/catalog/book[name]/ISBN/Price
Q4	/catalog/book[Title]/price[@38]/Publisher
Q5	//price
Q6	//catalog//Title

Work also proposes a lightweight and effective encoding scheme, Lineage Encoding, to support analysis of predicates and twig pattern queries over the stream. The Lineage Encoding scheme represents the parent-child relationships among XML elements as a sequence of bit-strings, called Lineage Code(V, H).Experiment by providing totally different x-path expression in order to measure the tuning time and access time.

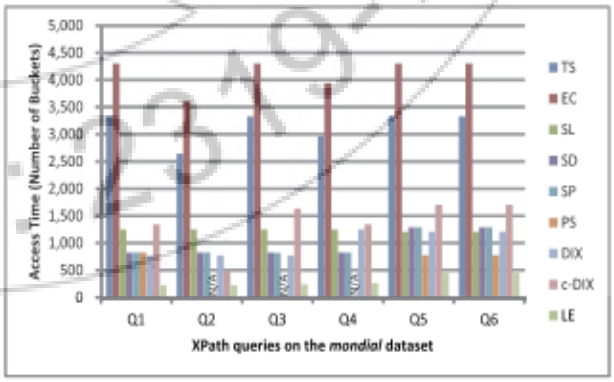

Figure 5.2: shows access time evaluation results on the real XML data set.

The access time is decided by two factors: 1) the size of data stream and 2) a correct prediction ensuring early termination of query processing. As shown within the table, LE exhibits

the best performance because it generates the smallest data stream by eliminating redundant tag names and attribute names, and terminates query processing quickly whereas S-node and DIX approaches explore the complete stream to seek out desired data dispersed over the stream. Particularly, the access times of TS and EC are considerably larger than the others, because the sizes of indices are considerably larger.

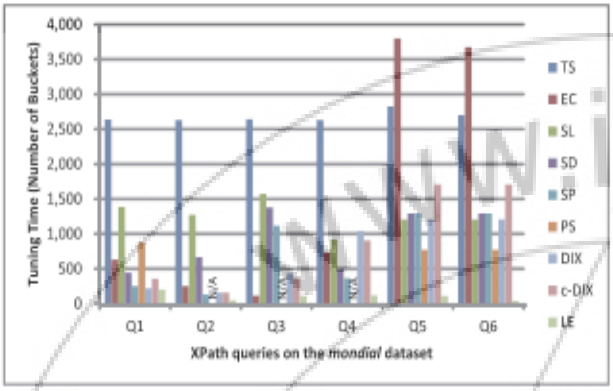

Figure 5.3: shows tuning time evaluation results on the real XML data set.

For all queries, LE exhibits the best performance because it avoids tuning of unnecessary data in the stream. Specifically, LE selectively accesses only relevant attribute values and texts needed for query processing, whereas the others tune all attributes and texts. For queries Q1-Q4, EC shows higher performance than TS, because it skips irrelevant parts of inverted lists using extent chaining. In contrast, TS tunes all indices to identify nodes needed for query processing.

6. Conclusions and Future Work

Twig pattern queries containing complex conditions are well-established and critical in XML query processing. This project proposed an efficient wireless XML streaming method supporting twig pattern queries. The previous work on wireless XML streaming solely addressed processing of simple path queries. Thus, they are inefficient for twig pattern queries. In contrast, our scheme provides an energy and latency efficient way to evaluate predicates and twig pattern matching. Specifically, project scheme reduces the size of the XML stream, exploiting the advantages of the structure indexing and attributes summarization. In addition, our scheme reduces the tuning time as it provides an effective way for selective access of XML elements as well as their attribute values and texts. The work proposed Lineage Encoding to support queries involving predicates and twig pattern matching. Work also defined the relevant operators and functions to efficiently process twig pattern matching. The mobile client will retrieve the specified data satisfying the given twig pattern by performing bit-wise operations on the Lineage Codes within the relevant G-nodes. Thus, project work can support twig pattern query processing while providing both energy and latency efficiencies.

We evaluated the performance of our scheme in the experiments, compared not only to the previous wireless XML streaming methods but to conventional XML query processing methods supporting twig pattern matching. Experiment can be done by using a real XML data set and a synthetic data set for the correctness of experiments. Demonstrated project work is effective and efficient in terms of the access time and tuning time. Work also showed conventional XML query processing strategies are inefficient within the wireless mobile environment due to their vast indices. In the future, work plan to analyze the problems that were not fully addressed in this paper. First, depth-first traversal of components increases the access time for specific queries. Second, as communication is not stable in wireless broadcasting atmosphere, the indexing mechanism should consider network failures such as tail drops and packet losses.

References

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, “Broadcast Disks: Data Management for Asymmetric Communication Envir- onments,” Proc. ACM SIGMOD Int’l Conf. Management of Data Conf., pp. 199-210, Mar. 1995.

[2] S. Al-Khalifa, H.V. Jagadish, N. Koudas, J.M. Patel, D. Srivastava, and Y. Wu, “Structural Joins: A Primitive for Efficient XML Query Pattern Matching,” Proc. Int’l Conf. Data Eng. (ICDE), pp. 141-152, Feb. 2002.

[3] M. Altinel and M. Franklin, “Efficient Filtering of XML Docu- ments for Selective Dissemination of Information,” Proc. Int’l Conf. Very Large Data Bases (VLDB), pp. 53-64, 2000.

[4] S. Amer-Yahia, S. Cho, L.V.S. Lakshmanan, and D. Srivastava, “Minimization of Tree Pattern Queries,” Proc. ACM SIGMOD Int’l Conf. Management of Data Conf., pp. 497-508, 2001.

[5] A. Berglund, S. Boag, D. Chamberlin, M.F. Fernandez, M. Kay, J. Robie, and J. Simeon, “XML Path Language (XPath) 2.0,” Technical Report W3C, 2002.

[6] N. Bruno, D. Srivastava, and N. Koudas, “Holistic Twig Joins: Optimal XML Pattern Matching,” Proc. ACM SIGMOD Int’l Conf. Management of Data Conf., pp. 310-321, 2002.

[7] S. Chen, H. Li, J. Tatemura, W. Hsiung, D. Agrawal, and K.S. Candan, “Scalable Filtering of Multiple Generalizaed-Tree-Pattern Queries over XML Streams,” IEEE Trans. Knowledge and Data Eng., vol. 20, no. 12, pp. 1627-1640, Dec. 2008.

[8] C. Chung, J. Min, and K. Shim, “APEX: An Adaptive Path Index for XML Data,” Proc. ACM SIGMOD Int’l Conf. Management of Data Conf., June 2002.

[9] Y.D. Chung, S. Yoo, and M.H. Kim, “Energy- and Latency- Efficient Processing of Full- Text Searches on a Wireless Broadcast Stream,” IEEE Trans. Knowledge and Data Eng., vol. 22, no. 2, pp. 207-218, Feb. 2010.

[10] B. Cooper, N. Sample, M.J. Franklin, G.R. Hjaltason, and M. Shadmon, “A Fast Index for Semistructured Data,” Proc. Int’l Conf. Very Large Data Bases (VLDB), Jan. 2001.

[11] Y. Diao, M. Altinel, M. Franklin, H. Zhang, and P.M. Fischer, “Path Sharing and Predicate Evaluation for

High-Performance XML Filtering," ACM Trans. Database Systems, vol. 28, no. 4, pp. 467-516, 2003.
[12] D. Florescu and D. Kossmann, "Storing and Querying XML Data Using an RDBMS," IEEE Data Eng. Bull., vol. 22, no. 3, pp. 27-34, Mar. 1999.
[13] M. Francechet, "XPathMark: An XPath Benchmark for XMark Generated Data," Proc. Third Int'l Conf. Database and XML Technologies (XSYM), 2005.
[14] R. Goldman and J. Widom, "DataGuides: Enable Query Formula- tion and Optimization in Semistructured Databases," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 436-445, 1997.
[15] T.J. Green, A. Gupta, G. Miklau, M. Onizuka, and D. Suciu, "Processing XML Streams with Deterministic Automata and Stream Index," ACM Trans. Database Systems, vol. 29, no. 4, pp. 752-788, 2004.
[16] A. Gupta and D. Suciu, "Stream Processing of XPath Queries with Predicates," Proc. ACM SIGMOD Int'l Management of Data Conf., pp. 419-430, 2003.
[17] T. Imielinski, S. Viswanathan, and B. Badrinath, "Energy Efficient Indexing on Air," Proc. ACM SIGMOD Int'l Conf. Management of Data Conf., pp. 25-36, 1994.
[18] T. Imielinski, S. Viswanathan, and B. Badrinath, "Data on Air: Organization and Access," IEEE Trans. Knowledge and Data Eng., vol. 9, no. 3, pp. 353-372, June 1997.
[19] H. Jiang, W. Wang, H. Lu, and J. Yu, "Holistic Twig Joins on Indexed XML Documents," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 273-284, 2003.
[20] H. Jiang, H. Lu, and W. Wang, "Efficient Processing of XML Twig Queries with OR-Predicates," Proc. ACM SIGMOD Int'l Manage- ment of Data Conf., pp. 59-70, June 2004.
[21] R. Kaushik, P. Bohannon, J.F. Naughton, and H.F. Korth, "Cover- ing Indexes for Branching Path Queries," Proc. ACM SIGMOD Int'l Management of Data Conf., pp. 133-144, June 2002.
[22] R. Kaushik, R. Krishnamurthy, J.F. Naughton, and R. Ramakrish- nan, "On the Integration of Structure Indexes and Inverted Lists," Proc. ACM SIGMOD Int'l Management of Data Conf., June 2004.
[23] C.-S. Park, C.S. Kim, and Y.D. Chung, "Efficient Stream Organization for Wireless Broadcasting of Xml Data," Proc. Int'l Conf. Asian Computing Science Conf., pp. 223-235, 2005.
[24] J.P. Park, C.-S. Park, and Y.D. Chung, "Attribute Summarization: A Technique for Wireless XML Streaming," Proc. Int'l Conf. Interaction Sciences, pp. 492-496, Dec. 2009.
[25] J.P. Park, C.-S. Park, and Y.D. Chung, "Energy and Latency Efficient Access of Wireless XML Stream," J. Database Management, vol. 21, no. 1, pp. 58-79, 2010.
[27] S.H. Park, J.H. Choi, and S. Lee, "An Effective, Efficient XML Data Broadcasting Method in Mobile Wireless Network," Proc. 17th Int'l Conf. Database and Expert Systems Applications (DEXA), pp. 358-367, 2006.
[26] F. Peng and S.S. Chawathe, "XPath Queries on Streaming Data," Proc. ACM SIGMOD Int'l Management of Data Conf., pp. 431-442, June 2003.
[27] SAX (Simple API for XML), http://www.saxproject.org, 2004.
[28] I. Tatarinov, S. Viglas, K. Beyer, J. Shanmugasundaram, E. Shekita, and C. Zhang, "Storing and Querying Ordered XML Using a Relational Database System," Proc. ACM SIGMOD Conf., pp. 204- 215, 2002.