

An Optimizing Query Processor with an Efficient Caching Mechanism for Distributed Databases

Selvaraj Prabha¹, Arputharaj Kannan², and Palaniappan AnandhaKumar²

¹Department of Information, RMK Engineering College, India

²Department of Computer Science and Engineering, Anna University, India

Abstract: *This paper provides an efficient way of querying among many distributed and heterogeneous data sources. We describe a database optimization framework that supports data and computation reuse, query scheduling and caching mechanism to speed up the evaluation of multiquery workload. The Caching query result is stored as an eXtensible Markup Language (XML) document. An XML oriented common data model and an XML Parser (XP) accept the SQL statement, which consists of selection constructs to impose constraints, project construct and also to update the database distributed. It also maintains the historical database to store the historical data. Using this optimized query processor and XML cache could simply fasten the query processing. The experimental results show that execution time is optimized and query cost is reduced.*

Keywords: *Temporal, caching, Xparser, query optimizer.*

Received May 24, 2005; accepted December 20, 2006

1. Introduction

Time is an important aspect of all real-world phenomena. Conventional databases model in an enterprise change dynamically by a snapshot at a particular point in time. As information is updated in a conventional database, the old is discarded forever. For many applications such as banking, accounting, medical record bookkeeping, etc., the changes made on their databases over time are a valuable source of information which can direct their future operation. Due to the importance of time varying data, efforts have been made to design temporal databases, which support aspects of time.

The problem of query optimization in relational database systems has received lot of attention. The efficient management and analysis of large data sets is important in many fields. As data sets sizes continue to grow, it is increasingly important and challenging to efficiently execute. In many cases, data analysis is employed in a collaborative environment, where multiple clients access the same data sets and perform similar processing on the data. In this case, a data server needs to process multiple queries simultaneously to minimize latency to the clients. The recently emerged eXtensible Markup Language (XML) has been widely accepted as the standard, and many projects are developed based on XML to facilitate data interchange and information sharing. The issue of XML data management and query processing is therefore very important and attracts a lot of attentions. The query is processed and the result is stored as an XML document. Finally, XML documents benefit from their hierarchical structure. Hierarchical

document are, in general, faster to access because you can drill down to the part you need, like stepping through the table of contents. They are also easy to rearrange, because each piece is delimited.

The main objective is to design a query processing system, which handles temporal database, and to optimize the multiple data analysis query workload in distributed environment for both temporal databases. One critical aspect in optimizing multi query loads lies in identifying data and computational reuse opportunities in the query data processing, so that same data is not retrieved multiple times and common processing is not performed repeatedly. We have designed and implemented database support for these applications by devising a data and computational reuse framework that enables the execution of queries in the result, where the client application intends to hold the results for a significantly long time interval.

This paper focuses on the design of databases with optimization but not applications. In network, the clients that do not have resources to connect to databases. In such a scenario, the complete process of retrieving the data and populating is performed on the server and the populated row set is passed on to the client using suitable architecture like Remote Method Invocation (RMI) or Enterprise Java Beans (EJB). The client would be able to perform all the operations like retrieving, scrolling, inserting, updating, and deleting on the row set without any connection to the database. Whenever data is committed to the database, a method is called which synchronizes the data in the rowset to that in the database.

The network communication cost is reduced and the response time is greatly improved, as only the required data is sent from the query optimizer to the distributed database. The optimization framework will be very effective in achieving data and computational reuse, increasing system throughput and decreasing average query execution time.

2. System Architecture

The architecture of the system developed in this work is shown in Figure 1. Caching is very attractive for use in distributed systems due to the reduced network traffic and improved response time. The idea of caching is that the maintenance of the results of previously executed queries. Since cached items are stored in XML data cache as XML document, query optimizer checks whether a query is fully answerable, partial or not. Through reasoning among cached results, the exact work to be done at the server side can be determined.

Instead of page replacement technique, we are going for query based cache replacement for better performance. Since distributed environment care should be taken to maintain the basic properties of the database systems. Whenever the query processor recognizes an updating statement, it updates the historical database and delete the XML document for the corresponding table. If no updating statement was received and the cache size increases, we use aging technique to replace the cache.

One critical aspect in optimizing multi query loads lies in identifying data and computational reuse opportunities in the query data processing, so that same data is not retrieved multiple times and common processing is not performed repeatedly. We have designed and implemented database support for these applications by devising a data and computational reuse framework that enables the execution of queries. The network communication cost is reduced and the response time is greatly improved, as only the required data is sent from the query optimizer to the distributed database. The optimization framework will be very effective in achieving data and computational reuse, increasing system throughput and decreasing average query execution time.

In applications, executing queries with possibly large number of rows in the result, where the client application intends to hold the results for a significantly long time interval. The clients who do not have resources to connect to databases can do querying and updating the database.

The query distribution manager maintains the database server table along with the tables for which the user has access. The query cache manager which receives the query from the user through optimizer and checks for the availability in the data cache. If not found then it messages the mismatch information. If

found it checks whether the related data is already queried and retrieves the result.

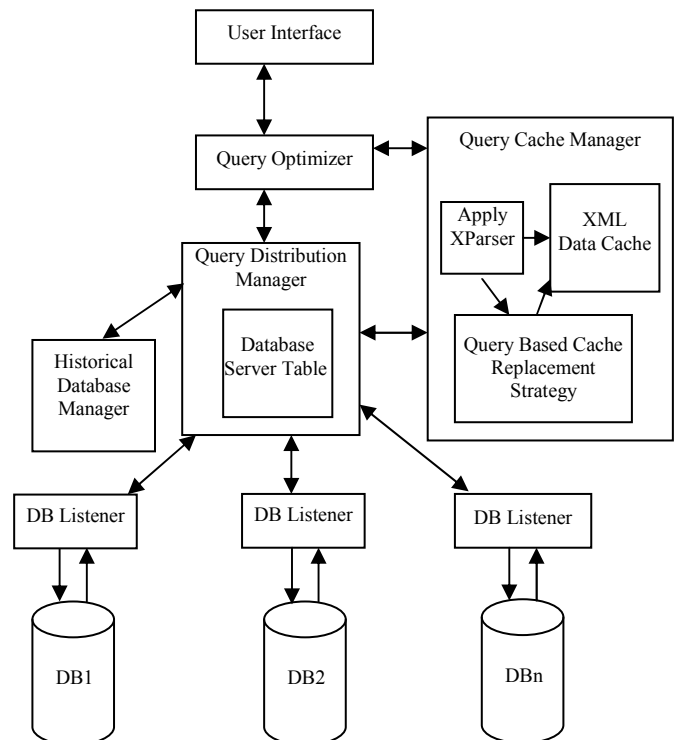


Figure 1. Optimized query processor with efficient caching mechanism.

The experiments were conducted with MS Access, Oracle and SQL Server. XML representation of the data from heterogeneous database gives a common orientation. We used MS Access, Oracle and SQL Server as back ends. The proposed scheme can be utilized to enable efficient sharing between systems. Which caters for a more structured handling of the exchanged information.

3. Related Work

3.1. Temporal Data Model

Transitioning temporal support in TSQL2 to SQL3 [6] summarizes the additions of table with transaction and valid time support. An Authorization Model for temporal XML Documents. This efficiently explains how to move conventional system to one encompassing temporal support.

Semantics of time varying attributes and their use for temporal database design [7] presents an efficient study of important aspects of the temporal semantics of attributes.

Temporal statement modifiers [9], launch the notion of statement modifiers that provide a wealth of systematically adding temporal support to an existing query language. A middle ware approach to temporal query processing [15] is well discussed. They also offer a way to methodically and temporal support to an existing implementation such that the result is a

temporal query language syntax, semantics and implementation.

3.2. Historical Database

An aggregation computation over complex objects [8], explains the transaction time model for maintaining historical database. The Vagabond approach to logging and recover in recovery in transaction-time temporal object database systems [10] concentrates on log-only-updation by keeping the old versions of objects which have to be copied to a new place before updating.

Efficient execution of multiple query work loads [3] and the utilization of intermediate results [1] and reuse of intermediate results [5] has been explained which improves the response time.

3.3. Optimizing Execution Time

Optimizing the execution of multiple query analysis on parallel and distributed environments [2] explains the analysis of data and computational reuse to improve the response time and reduce the latency [4] by caching the computed results for future use. An efficient caching mechanism for XML content adaptation [14] improves the hit ratio and space efficiency.

3.4. XML Representation

Converting relational database into XML document [12] explains the methodology of translating relational database into XML schema. Query translation from eXtensible Style Sheet Language Transformations (XSLT) to Sequential Query Language (SQL) [9] explains connection between the data that user sees and the data in the database. More specifically in translating XSLT queries to SQL queries. XML storing and processing techniques [10] explains the XML data organization methods, query evaluation model for Xquery and physical optimization of Xpath and Xquery queries.

4. Translate the Data to the XML Document

Designing data organization, we present a possible XML document to store the information about the data retrieved from the RDBMS for further processing to be efficient for both queries and updates. We can transform a database table into an XML document with a Document Type Definition (DTD) as shown below:

```
<metadata>
<column-definition>
<column-definition>
</metadata>
<data>
<currentRow>
```

```
<field-name1>value</field-name1>
<field-name2>value</field-name2>
</currentRow>
<currentRow>...
</currentRow>
</data>
<?xml version="1.0"?>
_ <properties>
- <XPRowset>
  <command />
  <concurrency>1008</concurrency>
  <datasource />
  <escape-processing>true
</escape-processing>
  <fetch-irection>0</fetch-direction>
  <fetch-size>0</fetch-size>
  <isolation-level>2</isolation-level>
  <key-columns />
  <map />
  <max-field-size>0</max-field-size>
  <max-rows>0</max-rows>
  <query-timeout>0</query-timeout>
  <read-only>true</read-only>
  <rowset-type>
ResultSet.TYPE_SCROLL_INSENSITIVE</rowset-
type>
  <show-deleted>>false</show-deleted>
  <table-name />
  <url />
_ <sync-provider>
</properties>
_ <metadata>
<column-count>9</column-count>
_ <column-definition>
  <column-index>1</column-index>
  <auto-increment>>false
</auto-increment>
  <case-sensitive>>false</case-sensitive>
  <currency>>false</currency>
  <nullable>0</nullable>
  <signed>>false</signed>
  <searchable>true</searchable>
  <column-display-size>11
</column-display-size>
  <column-label>au_id</column-label>
  <column-name>au_id</column-name>
  <schema-name />
  <column-precision>11
</column-precision>
  <column-scale>0</column-scale>
  <table-name />
  <catalog-name />
  <column-type>12</column-type>
  <column-type-name>varchar
</column-type-name>
  </column-definition>
_ <column-definition>
```

```

    ± <column-definition>
    ± <column-definition>
    ± <column-definition>
    ± <column-definition>
    ± <column-definition>
    ± <column-definition>
    ± <column-definition>
  </metadata>
</data>
<currentRow>
<au_id>341-22-1782</au_id>
<au_lname>Smith</au_lname>
<au_fname>Meander</au_fname>
<phone>913 843-0462</phone>
<address>10 Mississippi Dr.</address>
<city>Lawrence</city>
<state>KS</state>
<zip>66044</zip>
<contract>>false</contract>
</currentRow>
± <currentRow>
    ± <currentRow>
    ± <currentRow>
    ± <currentRow>
    ± <currentRow>
    ± <currentRow>
<au_id>998-72-3567</au_id>
<au_lname>Ringer</au_lname>
<au_fname>Albert</au_fname>
<phone>801 826-0752</phone>
<address>67 Seventh Av.</address>
<city>Salt Lake City</city>
<state>UT</state>
<zip>84152</zip>
<contract>>true</contract>
</currentRow>
</data>
</XPRowSet>

```

The common structure of node with left-sibling, parent and right-sibling pointers is straightforward. The data organization has two advantages. First, simply evaluating this query and get access to the blocks with data we need. Second, direct pointer allows us passing from one node to its neighbors almost for free.

5. The Temporal XML Data Model

In our XML document model, we associate two time dimensions: Valid time and transaction time (bi-temporal). The valid time of data denotes the time(s) when the data is true for the modeled entity. In the valid time dimension, which is now commonly used to indicate that a fact is currently valid. For instance, valid time interval states that a fact has been true. The transaction time represents the evolution of data and the time instants at which data is in the system. While a

valid time interval is generally supplied by the user, a transaction time interval is supplied by the system and the end time associated with the data in the system until it is subjected to any updating.

6. Xparser

Algorithm

Input: Query.

Output: Resultset.

For each query perform the following steps.

1. Check whether table is available as XML document.
2. If available in data cache then do
 - Lexical analysis ()*
 - Syntax analysis ()*
 - Get the metadata value and retrieve the corresponding values from the XML document by applying anyone of the operation complete match, projection, overlap and mismatch.*
 - Go to step 5.*
3. If not available in data cache then
 - DatabaseServerTable()*
 - Query the corresponding database*
 - Return the resultset.*
4. Translate the data to XML document if not already exists or append the data to the existing document, care should be taken to avoid redundancy.
5. Return the resultset to the user.

7. XML Caching

The optimized query processor acts as a middleware to transfer data between the user and the database. XP upon receiving the query checks for the availability of the document corresponding to the table in the SQL query statement. In this approach, there is no need to convert the SQL query to XQuery to query the XML document. The XP takes care of all and it directly maps the SQL query to the XML document.

The middleware components are query optimizer, query distribution manager, and query cache manager.

- *Query Optimizer*: Which connects and co-ordinates the user interface, query cache manager and query distribution manager.
- *Query Cache Manager*: Which checks for the availability. If available then it checks for the operation complete match, projection, overlapping using the Xparser by querying the XML document. Upon mismatch transfers the control to the database through query optimizer. If any updating is done, historical database is updated and the XML document is removed the data cache since we are using query based cache replacement strategy. If cache overflows then we use aging as second replacement strategy.

- *Query Distribution Manager*: Contains the various databases it interacts with and also the tables in the various databases. When the user queries it, checks with this table to find the availability of table in the database. If it finds the table name in this table it sends the query to the corresponding database server. The table is shown below:

```

<config>
<data>
<database>
<table_name>value1</table_name>
<table_name>value2</table_name>
<server>servername1</server>
<class>driver class </class>
<conn>location </conn>
</database>
</data>
</config>
<?xml version="1.0"?>
_ <config>
_ <data>_ <database>
<table_name>authors</table_name>
<table_name>master</tablename>
<table_name>Northwind</table_name>
<server>Sql Server</server>
<class>net.sourceforge.jtds.jdbc.Driver
</class>
<conn>jdbc:jtds:sqlserver://localhost/pubs;user=sa;
</conn>
</database>

```

8. Performance Evaluation

In this section, we present the experiment results to show the performance.

The time taken for the different queries to access data from distributed database and the retrieval of data from XML cache is observed. The performance of XML cache vs database was assessed and the results analysed. The time taken to process a query from XML cache decreases as the number of records increases is shown in Figure 2.

The following are the advantages of using XML in the implementation:

- Standard format for multiple applications.
- Data transformation across application.
- Disconnected database access.
- Reducing the database size according to the client side requirement.
- Reduced retrieval time.

9. Conclusion and Future Work

Providing an efficient way to query distributed data source has become an important issue. There has been lot of researches on processing database in distributed environment. In this paper, we presented an XML

database system and an Xparser suitable for the implementation of data analysis applications dealing with distributed heterogeneous data source and multiquery workloads. It uses a common optimization query processing model whose ability is to take advantage of data and computation reuse opportunities in the presence of application specific operators. A step is taken for developing temporal XML. We conducted experiments to show its performance in terms of execution time.

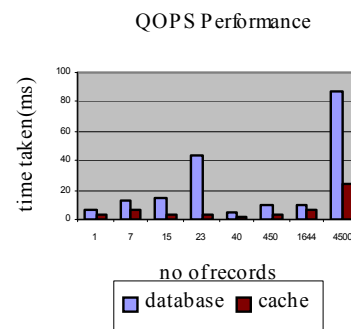


Figure 2. Time taken to process a query from XML cache decreases.

Further work in this direction will be the inclusion of forecasting the knowledge discovery methods for effective temporal reasoning and the provision of learning capabilities to improve the intelligence of the system. Certainly, more sophisticated compiler optimisation techniques can be employed to automatically identify and describe the reuse. We will test the performance of our caching mechanism with real world data to prove its efficiency.

References

- [1] Andrade H., Kurc T., Sussman A., and Saltz J., "Efficient Execution of Multiple Query Workloads in Data Analysis Applications," in *Proceedings of 2001 ACM/IEEE Super Computing Conference*, November 2001.
- [2] Andrade H., Kurc T., Sussman A., and Saltz J., "Exploiting Functional Decomposition for Efficient Parallel Processing of Multiple Data Analysis Queries," in *Proceedings of 2002 IEEE International Parallel and Distributed Processing Symposium*, April 2002.
- [3] Andrade H., Kurc T., Sussman A., and Saltz J., "Optimizing the Execution of Multiple Analysis Queries on Parallel and Distributed Environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 6, pp. 520-532, June 2004.
- [4] Andrade H., Kurc T., Sussman A., Borovikov E., and Saltz J., "On Cache Replacement Policies for Sensing Mixed Data Intensive Query Workloads," in *Proceedings of the 2nd Workshop Caching Coherence and Consistency*, June 2002.

- [5] Dumas ET AL, "TEMPOS: A Platform for Developing Temporal Applications on Top of Object DBMS," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 3, pp. 354-574, March 2004.
- [6] Fomichev A., "XML Storing and Processing Techniques," in *Proceedings of the Spring Young Researcher's Colloquium on Database and Information Systems (SYRCODIS)*, Russia, 2004.
- [7] Fong J. and Bloor C., "Converting Relational Database into XML Document," *Technical Report*, Francis Pang Department of Computer Science, City University Hong Kong, Hong Kong, 2001.
- [8] Kinno A., Yukitomo H., and Nakayama T., "An Efficient Caching Mechanism for XML Content Adaptation" in *Proceedings of the 10th International Multimedia Modelling Conference (MMM'04)*, IEEE, 2004.
- [9] Liu J. and Millist V., "Query Translation from XSLT to SQL," in *Proceedings of the 7th International Database Engineering and Applications Symposium (IDEAS'03)*, IEEE, 2003.
- [10] Nervag K., "The Vagabond Approach to Logging and Recover Recovery in Transaction-Time Temporal Object Database Systems" *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 4, pp. 504-519, April 2004.
- [11] Slivinskas G., "A MiddleWare Approach to Temporal Query Processing," *PhD Dissertation*, Faculty of Engineering and Science, Aalborg University Denmark, 2001.
- [12] Snodgrass R. T. "Semantics of Time Varying Attributes and Their Use for Temporal Database Design," *IEEE Transactions on Knowledge and Data Engineering*, pp. 366-377, 1997.
- [13] Snodgrass R. T., "Temporal Statement Modifiers," *ACM Transactions on Database Systems*, vol. 25, no. 4, pp. 407-456, December 2000.
- [14] Snodgrass R. T., "Transitioning Temporal Support in TSQL2 to SQL3," *IEEE Transactions on Knowledge and Data Engineering in Temporal Databases: Research and Practice*, vol. 26, no. 2, pp. 615-663, 1998.
- [15] Zhang D., "An Aggregation Computation Over Complex Objects" *PhD Dissertation*, University of California Riverside, August 2002.

KS Rangasamy College of Engineering, Tiruchengode till 2003. She has been working as a senior lecturer in the Department of Information Technology in RMK Engineering College Kavarapettai, Gummidipundi. Her research interests include database systems, system modelling and compiler design.



Arputharaj Kannan has completed his BS degree in mathematics from Madurai Kamaraj University in 1979, his Master degree in mathematics from Annamalai University in 1986, his ME degree in computer science and engineering from College of Engineering Guindy, Anna University in 1991, and his PhD in intelligent temporal databases from the Faculty of Electrical Engineering, Anna University in September 2000. He worked as a computer programmer in Bhabha Atomic Research Centre (BARC), Bombay from 1981 to 1989. He worked as a lecturer in the School of Computer Science and Engineering, Anna University from 1991 to 2000. Since 2000, he worked as an assistant professor in the Department of Computer Science and Engineering, Anna University, Chennai. He published 11 research papers in national and international journals, and also published more than 30 research papers in national and international conferences. His research interests include artificial intelligence, database systems, software engineering, intelligent agents and pattern recognition.



Palaniappan Anandha Kumar has completed his BE in electronics and communication engineering from University of Madras, Chennai and ME in computer science and engineering from Bharathiar University, Coimbatore in 1994 and 1997, respectively. Since 1998, He has been working as a lecturer in the Department of Computer science and Engineering, College of Engineering, Guindy, Anna University, Chennai. Currently, he is pursuing his PhD in computer science and engineering. His research interests include image processing, computer graphics, artificial intelligence, telemedicine, artificial neural networks, and robotics.



Selvaraj Prabha obtained her BE in computer science and engineering from Kongu Engineering College, Perundurai in 1998, and her ME in computer science and engineering from Anna University, Chennai in 2005. She has worked as a lecturer in