Metadata Services for Distributed Event Stream Processing Agents

Mahesh B. Chaudhari
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
Tempe, AZ 85287-8809, USA
mahesh.chaudhari@asu.edu

Suzanne W. Dietrich
Division of Mathematical and Natural Sciences
Arizona State University
Phoenix, AZ 85069-7100, USA
dietrich@asu.edu

Abstract

Enterprise-level applications are becoming complex with the need for event and stream processing, multiple query processing and data analysis over heterogeneous data sources such as relational databases and XML data. Such applications require access to the metadata information for these different data sources. This paper discusses the design and implementation of a service-based dynamic metadata repository over heterogeneous data sources in a distributed event stream processing environment. The metadata repository is dynamic such that data resources can be registered or unregistered at run time. The design of such a metadata database is the first step in researching multiple query optimization over various query expressions to detect and materialize common subexpressions over relational and XML structured data sources.

Keywords: metadata, event stream processing, common subexpressions, materialized views, multiple query optimization, WCF services.

1 INTRODUCTION

Database applications have evolved from passive, stand-alone applications to active applications accessed through the web or using a Service-Oriented Architecture (SOA). These enterprise-level applications rely heavily on data from multiple sources for multiple query processing, data presentation and data analysis. These multiple sources can be either relational, XML or object databases, which are generally referred to as persistent data sources. Recently, enterprise applications such as stock-market analyzers and criminal justice applications require processing of real-time data arriving on data streams by means of continuous queries. It is also important to monitor the events generated in such systems. In the past, research was individually focused on continuous queries over streams [3, 9] and distributed event processing [16]. Stream-processing applications receive data continuously, which is time-stamped as it arrives [6]. Continuous queries are defined in the system to process such a high-rate streaming data. These continuous queries have filtering capabilities with access to the persistent data sources to remove irrelevant data. In contrast to streams, a primitive event is defined as an atomic and instantaneous occurrence of interest at a given time [7]. Composite events detect complex and meaningful relationships among the multiple occurrences of primitive events. Many enterprise-level dynamic applications have the need to define events over the streaming data with access to the heterogeneous data sources. Once such events are defined, they can be detected and consumed or sent over event streams to other applications that can consume them later on. Thus, recent research has focused on integrating events and stream processing [4, 10].

The first step in designing such enterprise applications is to collect metadata for the different sources. The proposed enterprise-level application framework for integrating events and stream processing with access to distributed heterogeneous data sources is based on the concept of co-existence of data sources, similar to a dataspace support platform [8]. This approach differs from a typical data integration framework that semantically integrates the data sources to provide a global reconciled schema over the data sources [11]. The proposed metadata design coordinates the metadata-level information from the co-existing data sources using services, which provide a protocol for interactions between loosely coupled systems.

The objective of this paper is to design a service-based metadata repository for the heterogeneous data sources registered in a distributed event stream processing framework. These heterogeneous sources include relational databases, XML data, event and data streams.

Appeared in SEDE2010, San Francisco, June 16-18, 2010, pp. 307-312.

Section 2 provides an overview of a prototype environment being built using the Coral8 Event Stream Processor, SQL Server 2008, Oracle 11g server, XML documents and the C# programming language with .NET framework 3.5. Section 3 describes the metadata repository design and the different components that contribute to the metadata repository. Section 4 describes the implementation of the metadata repository and the services exposed for accessing it through a service-oriented architecture. The paper concludes with a brief discussion of using this metadata repository in future research directions in the loosely coupled distributed event stream processing framework.

2 DISTIBUTED EVENT STREAM PROCESSING

The concept of Distributed Event Stream Processing Agents (DEPAs) was initially envisioned in [20] for future research directions over various aspects of the integration of events and stream processing with access to heterogeneous structured data sources. A DEPA can subscribe and consume event and data streams from different data sources, including application event generators, the output from continuous queries over sensor data, and streams of incremental changes from databases, such as the Oracle Streams [19] or Change Data Capture feature for SQL Server 2008 [15] for monitoring database log files. The event or data streams can be either in relational format or in a structured XML format. Processing these events or data streams often requires access to persistent data sources such as relational databases and XML data.

This research is exploring the incremental maintenance of materialized views over heterogeneous structured data sources for multiple query optimization in a distributed event stream processing environment as shown in Figure 1. In order to define materialized views, the DEPA maintains its own metadata repository for persistent data sources such as relational and structured XML databases, different query expressions such as data and event streams, continuous queries with composite event definition and detection over these streams, and queries defined over the persistent databases. The vision is that each DEPA will have a specific responsibility within the distributed system, which will increase the probability of common subexpressions on which to define materialized views for improving performance. Thus, each DEPA will subscribe to its own set of resources required to perform the specific responsibility. However,

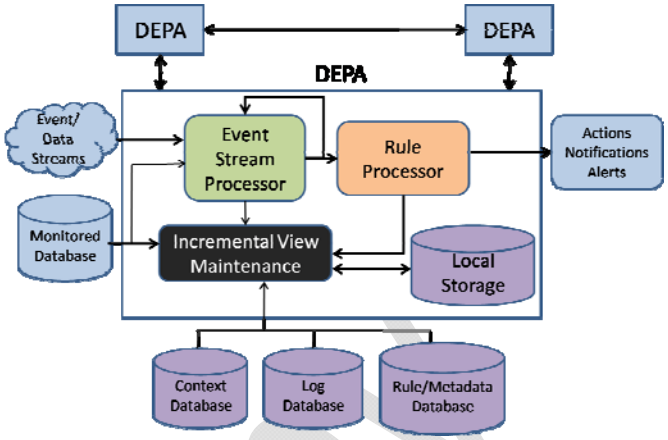

Figure 1: DEPA Architecture

DEPAs can communicate with each other to exchange information.

The prototype for the DEPA environment is currently under development to facilitate the exploration of these research issues. The persistent data sources are SQL Server 2008 databases, Oracle 11g Server databases, and XML documents. SQL Server and Oracle Server allow access to database log files through the Change Data Capture feature and Oracle Streams, respectively. Flat XML files are used at present to store XML information, although an open-source native XML database is being considered for future development. For event and stream processing, DEPAs are using the Coral8 Event Stream Processor [1]. Coral8 has its own Continuous Computation Language (CCL) that has SQL-like syntax for defining different data streams and event streams. CCL also has the capability of defining continuous queries and primitive/composite events with temporal and windowing capabilities over the streams with access to persistent data sources.

This research project is exploring the use of the Language INtegrated Query (LINQ) [5, 13] as a materialized view definition language within this framework. LINQ provides a unifying paradigm for querying heterogeneous data sources declaratively. An advantage of LINQ is that the same language can query a collection of: objects in memory or an object-oriented database, tuples in a relational table, or elements in an XML document. This scenario is likely for a DEPA, since the event and stream processing often requires access to heterogeneous structured data sources at the same time. The coordination of accessing all these sources, processing events and streams and defining materialized

views requires DEPA-level metadata information of the registered data sources.

3 METADATA DESIGN

The DEPAs are essential to the vision of developing enterprise-level applications by orchestrating events and streams over heterogeneous data sources that are registered with the DEPA. The "application integrator", typically a knowledgeable programmer, can define events of interest, filter streams and pose queries to coordinate these applications.

The goal of this research is to examine multiple query optimization within this context to improve the performance of the system. Specifically, the research will use the metadata to identify common subexpressions from the various queries to identify views over potentially distributed sources that can be materialized at the DEPA level to improve performance. Ultimately, the research will eventually explore incremental view maintenance within this context.

In order to extract common subexpressions from these different query expressions, it is important for the DEPA to have access to metadata-level information on the different persistent data sources such as relational and XML databases and parsing-level access to various streams, events, continuous queries, SQL and LINQ queries. The conceptual metadata repository is shown in Figure 2.

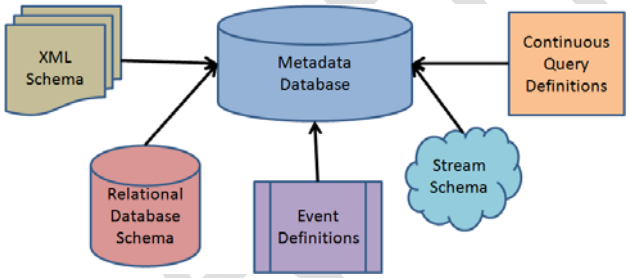

Figure 2: Conceptual Metadata Repository

DEPAs are autonomic agents that can register/unregister resources at run time, which requires the design of the metadata repository to be able to handle dynamic updates. The metadata repository consists of a persistent component that stores the access information for the data sources and a corresponding runtime component that provides access to the metadata through services.

The persistent metadata consists of query expressions (QE) and structured data sources (DS). QE is a 5-tuple of the different query expressions defined in the DEPA as follows:

$$QE = (mv, lq, sq, cq, ed)$$

where each component is a set of query expressions:

- *mv* - materialized views
- *lq* - LINQ queries
- *sq* - SQL queries
- *cq* - continuous queries
- *ed* - event definitions

DS is a 4-tuple of different data sources registered with the DEPA of the form:

$$DS = (rel, xml, evt, str)$$

where each element is a set of resources:

- *rel* - relational databases
- *xml* - XML documents
- *evt* - primitive or composite events
- *str* - streams

Figure 3 shows the class diagram of the metadata repository. The persistent metadata section stores only the information necessary to rebuild the metadata required at run time. Since one component of this research is to explore LINQ as the materialized view definition language, the metadata for relational databases is accessed using the Entity Framework through the *EntityMetaData* clas and its associated classes [14]. Metadata for the XML data, which is XML schema, is exposed through the *XMLSchemaMetaData* handler class and its associated classes.

The metadata for the Coral8 server is accessed using the API calls provided by the Coral8 .NET development kit [1]. These calls are wrapped in the *Coral8ServerMetadata* class and its associated classes. The query expressions registered with the DEPA access the persistent data sources for the required data. In order to find out which data sources are required for a query expression, it is necessary to parse the expression and extract the relevant information. For the LINQ queries, the Entity Framework provides an expression tree visitor that allows access to different parts of the LINQ expression. Thus, for LINQ-based query expressions, the parsing information is accessed through the

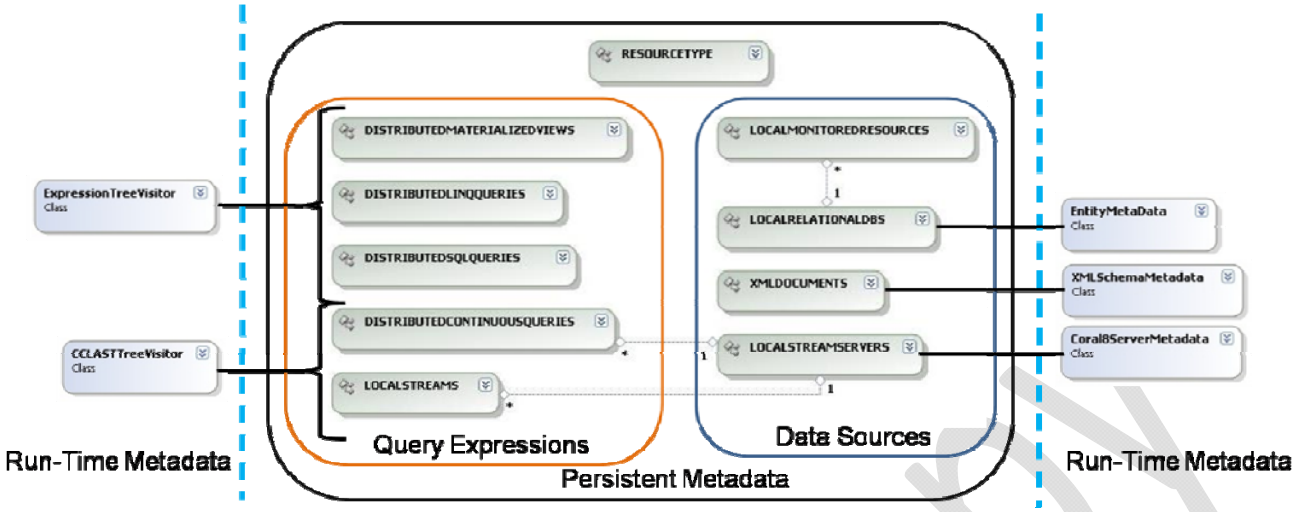

Figure 3: Metadata Respository

ExpressionTreeVisitor class and its associated classes. Similarly, for the Coral8 event stream definitions and continuous queries, the parsing information is accessed through the *CCLASTTreeVisitor* class and its associated classes. The Coral8 parser has been written for C# using ANTLR tool [2] and the modified ANTLR grammar provided by Aleri Inc.

4 IMPLEMENTATION OF METADATA SERVICES

The DEPAs communicate peer-to-peer in a distributed environment. The DEPAs exchange information about the subscribed resources, as well as the event and data streams they are exposing to other DEPAs. The DEPAs also communicate about the changes occurring in the persistent data sources. Thus, in order to achieve these key functionalities, a Service-Oriented Architecture (SOA) serves as an ideal technology to develop the DEPA framework. Since the research focuses on using LINQ as the view definition language and LINQ is part of the .NET framework, the .NET technology called Windows Communication Foundation (WCF) is a unified framework to build SOA-based applications [12].

WCF is a SDK provided by Microsoft to develop and deploy services, which can take advantage of the underlying .NET 3.5 framework. Some of the key features of WCF are service instance management, asynchronous calls, transaction management, disconnected message queuing and security. These features distinguish WCF services from traditional web services such that WCF services are stateful and they can retain the client call-back handlers to send messages or data back to the client in an asynchronous way. This is also helpful in achieving server-pushing so that if any metadata or data has changed, then that DEPA service will notify other subscribing DEPAs about the changes. Hence, it is not essential for all the DEPAs to keep polling for the changes occurring at other DEPAs.

Like any typical service, a WCF service has endpoints through which the clients can communicate with the service. The WCF service can be hosted on either an IIS server or Windows Activation Service (WAS) or in a standalone windows process. The same service can have multiple endpoints providing different subsets of services to different clients based on their subscriptions. Each endpoint is characterized by 3 parameters: a URL address of the endpoint; a binding mechanism that tells how the endpoint can be accessed; and a contract name that indicates which service contract is exposed at that endpoint. The binding mechanism is the most important criteria through which the client can communicate with the endpoint either synchronously or asynchronously, through TCP, HTTP, MSMQ or Named Pipes protocol.

In the DEPA environment, the metadata repository can be accessed at different levels through different WCF services. Figure 4 shows two important WCF services that allow access to the metadata repository. The Subscription service is accessible to the clients to register/unregister the persistent resources and different query expressions. The Metadata provider service provides actual access to the metadata for the resources registered.

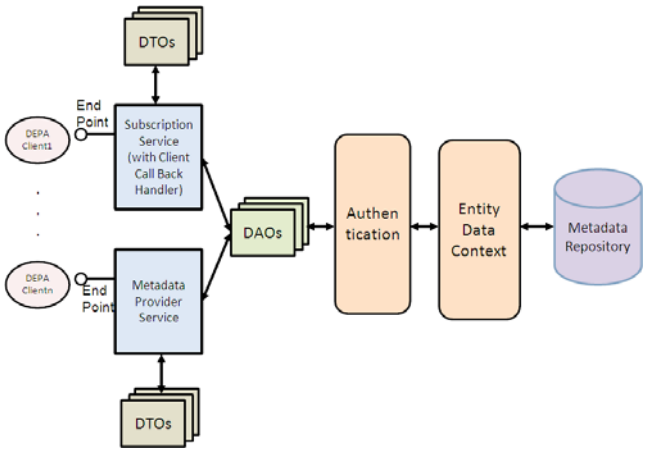

Figure 4: Exposing Metadata through SOA Services

One of the main issues regarding providing data access over services is marshalling and unmarshalling of the data between the client and the service. In order to avoid heavy objects transfer between the client and service, the metadata services use the enterprise-level design patterns of Data Access Objects (DAO) [17] and Data Transfer Objects (DTOs) [18]. DAOs are the abstraction-layer objects through which the metadata repository is accessible. DTOs are the light-weight replica objects of actual entities retrieved from the repository. These DTOs are the actual objects that are marshalled and unmarshalled. The mapping between DTOs and the actual entities is handled by DAOs.

The metadata prototype for the DEPA environment is developed using WCF services and C# with the metadata stored in an Oracle 11g database. The service administrator can subscribe/unsubscribe resources at run time. For registering a relational data source, the *EntityMetaData* handler class takes the access information and creates the Entity Context class for that relational database, converts this class into a Dynamic Linked Library (DLL) and loads the DLL into the WCF service, thus updating the metadata repository at run time. For unregistering a relational data source, the same handler class unloads the DLL file, deletes the file and then removes access information from the metadata repository, thus keeping the repository updated. A similar process is followed for the XML documents. The event stream and continuous query definitions are first registered or unregistered with the Coral8 Streaming server and then the *Coral8ServerMetadata* and the *CCLASTTreeVisitor* classes can access the metadata.

5 DISCUSSION

This paper has discussed the design and implementation of a SOA-based metadata repository for processing events and data streams in a loosely coupled distributed environment in coordination with heterogeneous structured data sources. One aspect of this research is to explore the feature of providing materialized views local to the DEPAs for processing events and data streams. In order to define materialized views, it is important to extract common subexpressions from different query expressions, such as continuous queries, primitive and composite event definitions, SQL queries and existing view definitions. Designing the metadata repository to provide access to metadata-level information of the distributed sources is the first step in the journey of exploring multiple query optimization for a variety of query expressions to detect common subexpressions and using LINQ as a materialized view definition language over heterogeneous structured data sources while respecting the native format of the data. This metadata repository is the common backbone for the following future research challenges:

1. *Dependency analysis across different filtering queries to identify common subexpressions as potential candidates for materialized partial joins.*

Using an analysis of the various rules, conditions, and queries, materialized views representing the partial joins of common subexpressions can provide the foundation for the efficiency of the incremental evaluation by exploring multiple query optimization within the framework. While identifying these common subqueries, the metadata repository is essential for query unfolding and to get metadata information about the data sources on which the queries are defined. The metadata database will also provide schema-level information for the candidate partial joins across relational and XML data sources.

2. *Techniques for selectively materializing the partial joins over relational as well as XML data sources.*

After identifying the potential candidates for the materialized views, one research challenge is to design a heuristic algorithm that will selectively choose partial joins over the heterogeneous data sources that will be beneficial to materialize. Once the materialized views are defined in the system, the original query expressions have to be rewritten to use the materialized views instead of the

underlying data sources. The metadata repository will play an important role in assisting this process.

3. *Incremental Evaluation and Materialized Views for Integrating Streams, Events, and Persistent Data.*

Capturing of deltas or changes to the original data sources is important in incremental view maintenance of the materialized partial joins. Using the metadata repository and the materialized views, the native deltas arriving into the system must be analyzed and used to incrementally update the views.

Enterprise-level software systems are becoming increasingly complex, requiring data integration of heterogeneous structured data sources in a loosely coupled distributed environment with support for handling events and streaming data. The research objective of this project is to explore the use of multiple query optimization over various query expressions to identify common subexpressions as the potential candidates for materialized views over distributed heterogeneous data sources. A unique aspect of this research is the use of LINQ as the materialized view definition language, which supports querying the data sources and their deltas in their native formats. The proposed metadata repository and its SOA-based services will provide the necessary information during all these research activities. The development of incremental evaluation techniques for the materialized views over heterogeneous data sources is expected to improve the performance of these complex applications incorporating events and streams.

6 ACKNOWLEDGEMENTS

This research is partially supported by the National Science Foundation (CSR 0915325).

7 REFERENCES

[1] Aleri Inc., http://www.aleri.com/products/aleri-cep/coral8-engine, accessed on February 11, 2010.

[2] Antlr v3, http://www.antlr.org/, accessed on February 08, 2010.

[3] S. Babu and J. Widom, "Continuous queries over data streams," *SIGMOD Rec.*, vol. 30, pp. 109-120, 2001.

[4] R. S. Barga, J. Goldstein, M. H. Ali and M. Hong, "Consistent streaming through time: A vision for event stream processing," in *CIDR*, pp. 363-374, 2007.

[5] C. Calvert and D. Kulkarni, *Essential LINQ*. Boston, MA: Addison-Wesley, 2009.

[6] D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul and S. Zdonik, "Monitoring streams: A new class of data management applications," in *Proc of VLDB*, pp. 215-226, 2002.

[7] S. Chakravarthy and D. Mishra, "Snoop: an expressive event specification language for active databases," *Data Knowl. Eng.*, vol. 14, pp. 1-26, 1994.

[8] M. Franklin, A. Halevy, and D. Maier, "From databases to dataspaces: a new abstraction for information management," *SIGMOD Rec.* vol. 34, 4, pp. 27-33, 2005.

[9] L. Golab and M. T. Özsu, "Issues in data stream management," *SIGMOD Rec.*, vol. 32, pp. 5-14, 2003.

[10] Q. Jiang, R. Adaikkalavan and S. Chakravarthy, "MavEStream: Synergistic integration of stream and event processing," in *Proc of ICDT '07*, pp. 29, 2007.

[11] M. Lenzerini, "Data integration: a theoretical perspective," in *Proc of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (Madison, Wisconsin, June 03 - 05, 2002), PODS '02, ACM, New York, NY, 233-246, 2002.

[12] J. Löwy, *Programming WCF Services*, Sebastopol, CA: O'Reilly Media Inc., 2009.

[13] Microsoft Corporation, LINQ: .NET Language-Integrated Query, http://msdn.microsoft.com/en-us/library/bb308959.aspx, accessed on January 19, 2010.

[14] Microsoft Corporation. The ADO.NET Entity Framework Overview, http://msdn.microsoft.com/en-us/library/aa697427(VS.80).aspx, accessed on January 19, 2010.

[15] Microsoft Corporation. Tracking Data Changes: SQL Server 2008 Books Online, http://msdn.microsoft.com/en-us/library/bb933994.aspx, accessed on June 20, 2009.

[16] G. Mühl, L. Fiege and P. Pietzuch, *Distributed Event-Based Systems*. Berlin; New York: Springer-Verlag, pp. 384, 2006.

[17] Oracle Corporation, Core J2EE Patterns - Data Access Object, http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html, accessed on January 05, 2010.

[18] Oracle Corporation, Core J2EE Patterns - Transfer Object http://java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html, accessed on January 05, 2010.

[19] Oracle Corporation. Oracle Streams - Features Overview. http://oracle.com/technology/products/dataint/htdocs/streams_fo.html, accessed on June 20, 2009.

[20] S. Urban, S. W. Dietrich and Y. Chen, "An XML framework for integrating continuous queries, composite event detection, and database condition monitoring for multiple data streams," in *Proc of Daghstul Seminar on Event Processing*, pp. 1-5, 2007.