DISTRIBUTED TRANSACTIONS FOR ODMG FEDERATED DATABASES

Damir Becarevic and Mark Roantree
Interoperable Systems Group
Dublin City University, Ireland
damirb@compapp.dcu.ie

Abstract Global transactions are still an issue for federated database systems. In the IOMPAR project, one of the goals is to develop a transaction protocol for ODMG databases which act as wrappers to information systems in a federation. In this short paper we describe an architecture for transporting secure data in a federated database system. Our on-going work involves providing the transaction service which can guarantee global transactions within our architecture.

1. INTRODUCTION

Federated database systems are comprised of heterogeneous information systems and use an integrated schema to provide a global interface for users and applications. Global transactions submitted at the federated layer are divided into set of subtransactions to be executed at local databases. This research is focused on the development of a secure transaction architecture for object databases operating in a federated database architecture. Research is conducted under the IOMPAR (Internet Object Management: Privacy ARchitecture) project in Dublin City University [8], The IOMPAR project is focused on the provision of an architecture and services for the execution of complex transactions in a secure environment. The strategy is to reuse existing (standard) technologies where possible and only supply those layers that are required: security, a flexible transaction manager, and multimedia handling.

IOMPAR uses the OASIS federated architecture described in [7], which uses the ODMG model [4] as its common model. OASIS provided an architecture and services to construct federated schemas for heterogeneous systems [5]. This architecture did not provide security, support for behaviour in views, or multimedia support. Since the majority of database systems are not ODMG databases, a wrapper language for mapping local schemas to ODMG schemas was developed and implemented. Please refer to [9] for a description of federated architectures and issues, and to [2] for a description of research work using object-based federated systems.

The motivation for this research is as follows: a global transaction service needs to extract data from local databases, compare sub-results and compute the final result for each global transaction. This means that data is moved from one database to another in the form of objects. Transaction consistency for all modifying operations must be guaranteed at both the local and global levels. There are many problems and issues which arise with global transactions across multiple heterogeneous systems. This research attempts to solve some of these problems as we construct a new global transaction manager for a federation of databases which uses an ODMG canonical model.

2. RESEARCH DETAILS

In this Section we introduce the concept of the Secure Communication Architecture (SCA) for ODMG databases. The architecture provides an object transport infrastructure and distributed transaction control system for object transfer.

2.1. Secure Communication Architecture Overview

The Secure Communication Architecture (SCA) extends the Object Transport Architecture presented in [1] for the OASIS project. The SCA provides transaction control and common protocol for transfer of data between systems which employ an ODMG wrapper. Each database and its corresponding interconnecting layers form one node in the federated system. At a high level, the SCA consists of three modules: The Transport Service, Local Transaction Service and Global Transaction Service as illustrated in Figure 1.

The Transport Service manages the transport of objects and commands between nodes in the distributed system. The protocol used for the encapsulation of metadata and commands is XML. XML is convenient as a data interchange format for incompatible data sources, because data encoded using XML can be easily interpreted by different types of systems. The object data is not encoded in XML, but using a protocol that support binary data encoding. Protocol encoded data is transported over a CORBA transport channel, as CORBA [6] provides the low level transport infrastructure needed for the physical transfer of data. The separation of data and metadata transport protocols is to improve performance since XML has a large overhead caused by markup information.

The Local Transaction Service manipulates a local database and ensures local transaction consistency. It performs read and write operations on both data and metadata stored in the database. The execution of a series of modifying

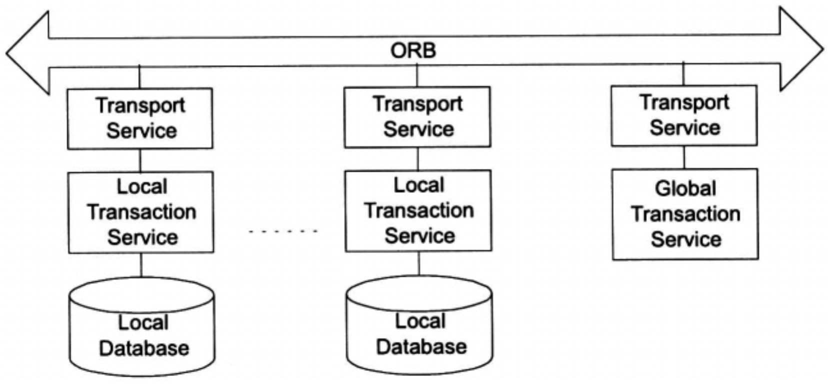

Figure 1. The Secure Communication Architecture

operations in a single global transaction is guaranteed by the local transaction manager.

The Global Transaction Service is positioned at the federated layer and controls execution of distributed transactions. It initiates global transactions and ensures transaction consistency and global serialization using a new protocol suitable for global transactions. This protocol forms the main part of this research task within the IOMPAR project. The GTS generates a set of commands and submits them to local databases for execution.

2.1.1 The Transport Service. The Transport Service performs protocol coding/decoding and transport for objects and commands. The Protocol Coder module transforms metadata and transaction commands into an XML representation. The format for XML encoding of ODMG objects is OIFML introduced in [3]. Data types, attribute names and relations in OIFML are defined by XML tags. Metadata values are placed between attribute start tag and the attribute end tag, while complex data structures are represented by tag hierarchies. This XML transport protocol is capable of encoding ODMG metadata, OQL commands, and messages. Object data is transported in some protocol that supports binary data encoding. Encoding format of transport protocol is called protocol metadata and is defined in XML.

At the target node the metadata transfer is decoded and verified against the local database schema using the Local Transaction Service. Protocol metadata is used for decoding of data protocol, so the receiving side does not need to have previous knowledge of object data encoding format. Constructed memory structures representing transported objects are delivered to the Local Transaction Service. The Transfer Agent is a CORBA object that can function both as

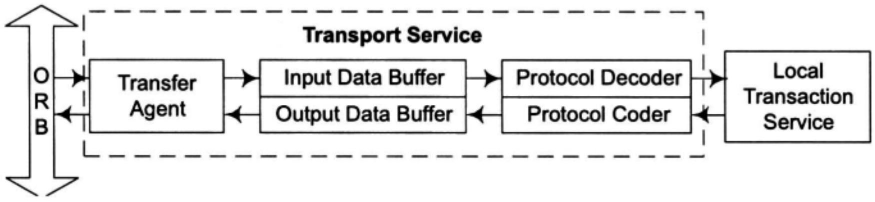

Figure 2. The Transport Service

a sender and as a receiver for data transport. The transport packages are transferred between nodes as binary data streams. Received packages are placed in the Input Data Buffer for processing by the data decoder.

2.1.2 The Local Transaction Service. The Local Transaction Service is the operational centre for each local database node. It consists of six modules as illustrated in Figure 3.

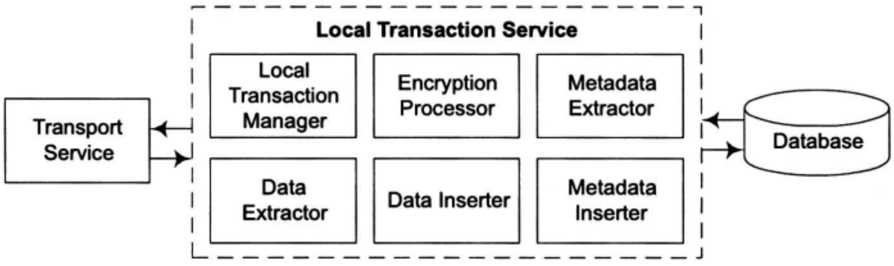

Figure 3. The Local Transaction Service

The Local Transaction Manager receives decoded commands and data from the Transport Service. It analyzes commands and activates other Local Transaction Service modules. Commands can be classified as: transaction commands such as `begin transaction`, `commit`, `rollback`; metadata manipulation commands such as create class definition and edit class definition; and data manipulation commands such as create modify objects.

The Encryption Processor performs encryption and decryption of data. Not all data or commands need be encrypted, so the Encryption Processor can selectively encrypt only specified property values.

The Metadata Extractor reads metadata information from the schema repository of the local database. Extracted metadata is passed to the Transport Service for XML encoding and transported to the target system.

The Data Extractor reads data values from the database. It can extract specified property values if required. Prior to transfer, the Encryption Processor is called.

The Metadata Inserter creates new class definitions and delete existing ones. For deleted class definition, all objects belonging to this class are also deleted.

The Data Inserter module is responsible for all data manipulations on physical data. All operations are performed within a transaction session started by the Local Transaction Manager.

2.1.3 The Global Transaction Service. The Global Transaction Service (GTS) is positioned at the federated level of the architecture. The GTS analyses global transactions and divides them into a set of subtransactions that are submitted to local database nodes for execution. The priority level[1] is specified for each subtransaction by the client who submitted the global transaction. If a high priority is required, then the atomicity of the corresponding subtransaction must be guaranteed by the two-phase-commit protocol. Otherwise, the subtransaction is considered a low priority, and it is submitted to the LTS but does not require completion (for the global transaction to succeed). For the global transaction to succeed, all high priority subtransactions must successfully commit. The Local Transaction Service determines if a global subtransaction is executable based on information of transaction capabilities of the local database system. If the LTS cannot support the required transactional protocol, the global transaction is aborted.

3. CURRENT RESEARCH

In this short paper we provided an overview of the framework in which our proposed global transaction service will operate.

Our current focus is on examining current research in the field of distributed transaction management for heterogeneous database systems. This information will provide input into the development of our own ODMG-based transaction manager. We are also focused on providing a detailed specification for transport of data (as binary streams) and metadata and decoding information (as XML data). Binary protocols should support binary data encoding and efficient transport of large volumes of multimedia.

[1] Priority levels are part of our work on a transaction taxonomy and are not covered here.

Work is already underway on a prototype developed in C++ using the Versant 6.0 ODMG databases and Oracle 8i object-relational databases operating on both Windows and Linux platforms. The middleware system is Iona Orbix 2000, a CORBA development environment for Windows and Linux.

References

[1] Byrne R, Roantree M., *An Object Architecture for ODMG Databases*, Proceedings of the 34th Hawaii International Conference on System Sciences.

[2] Bukhres O. and Elmagarmid A. (eds.), *Object-Oriented Multidatabase Systems*, Prentice Hall, 1996.

[3] G. M. Bierman, *Using XML as an Object Interchange Format*, ODMG web url: www.odmg.org, May 2000.

[4] Cattel R. et. al. (eds.) (2000). *The Object Data Standard: ODMG 3.0*, Morgan Kaufmann.

[5] Oasis Project, *www.compapp.dcu.ie/˜oasis,* 2000.

[6] Orfali R. and Harkey D. (1998) *Client/Server Programming with Java and CORBA*, Wiley.

[7] Roantree M., Murphy J. and Hasselbring W. *The OASIS Multidatabase Prototype*. ACM Sigmod Record 28:1, March 1999

[8] Roantree M., *The Iompar Project: Secure Transport of Complex Objects,* Technical Report IOM-001, November 2000

[9] Sheth A., Larson J. *Federated Database systemsfor Managing Distributed Heterogeneous and Autonomous Databases*, ACM Computing Surveys, 22:3, pp 183-236, ACM Press, 1990