Chapter III
Active XML Transactions

Debmalya Biswas
SAP Research, Germany

Il-Gon Kim
Korea Information Security Agency (KISA), Korea

ABSTRACT

Active XML (AXML) provides an elegant platform to integrate the power of XML, Web services and Peer to Peer (P2P) paradigms by allowing (active) Web service calls to be embedded within XML documents. In this chapter, the authors present some interesting aspects encountered while investigating a transactional framework for AXML systems. They present an integrated locking protocol for the scenario where the structure of both data and transactions are nested. They show how to construct the undo operations dynamically, and outline an algorithm to compute a correct optimum undo order in the presence of nesting and parallelism. Finally, to overcome the inherent problem of peer disconnection, the authors propose an innovative solution based on “chaining” the active peers for early detection and recovery from peer disconnection.

1. INTRODUCTION

Active XML (AXML) (Abiteboul et. al., 2003) systems provide an elegant way to combine the power of XML, Web services and Peer to Peer (P2P) paradigms by allowing (active) Web service calls to be embedded within XML documents. An AXML system consists of the following main components:

- **AXML documents**: XML documents with embedded AXML service calls (defined below). For example, the AXML snippet in Fig. 1 is an AXML document with the embed-
ded service call ‘getGrandSlamsWon’. The function of the service getGrandSlamsWon is basically to retrieve the Grand Slams won by a tennis player, and the abbreviations A, F, W and U correspond to the Australian, French, Wimbledon and US Grand Slams respectively.

- **AXML Services:** AXML services are basically Web services, defined as queries/updates over local AXML documents. Note that while AXML services can be invoked remotely, the operations they encapsulate are local, that is, defined on AXML documents hosted on the same peer.

- Peers where both the AXML documents and services are hosted. AXML peers also provide a user interface to query/update the stored AXML documents locally.

An embedded service call may need to be materialized: 1) in response to a query on the AXML document (the materialization results are required to evaluate the query) or 2) periodically as specified by the ‘frequency’ attribute of the AXML service call tag <axml:sc>. We illustrate materialization with the following example: Let the AXML document D corresponding to Fig. 1 be hosted on peer AP1, and the service getGrandSlamsWon hosted on another peer AP2. Now, assuming the embedded service call getGrandSlamsWon needs to be materialized, the following sequence of steps takes place:

1. Fig. 2(a). AP1 invokes the service getGrandSlamsWon of AP2 with the parameter value children nodes of the service call getGrandSlamsWon node of D. Note that a service call’s parameters may themselves be defined as service calls. Given such a scenario, AP1 needs to first materialize the parameter service calls, and then invoke the service getGrandSlamsWon.

2. Fig. 2(b). On receiving the invocation results (an XML subtree), AP1 does one of the following based on the ‘mode’ of the embedded service call getGrandSlamsWon. A service call can have the following modes: a) replace: the previous results are replaced by the current materialization results, or b) merge: the current results are appended as siblings of the previous results. The resulting AXML document, after a materialization of getGrandSlamsWon of D with parameter value “$year = 2004”, is shown in Fig. 3. Analogous to parameter inputs, the invocation results may also be static XML or another service call. If the invocation results contain another service call, then AP1 needs to materialize them first before inserting the results in D.

Transactions are a useful abstraction to provide fault-tolerance, reliability and robustness to distributed systems. A transaction (Bernstein et. al., 1987) can be considered as a group of operations encapsulated by the operations Begin and Commit/Abort having the following properties A (Atomicity), C (Consistency), I (Isolation), D (Durability). We assume prior knowledge of the basic transactional concepts, especially, lock based concurrency control protocols, undo/redo recovery and nested transactions. In this work, we study a transactional framework for AXML systems. Characteristics of an AXML system, important from a transactional point of view, are as follows:

- **Distributed:** The distributed aspect follows from: 1) the capability to invoke services hosted on remote peers, and 2) distributed storage of parts of an AXML document across multiple peers (Abiteboul et. al. 2003). In case of distributed storage, if a query Q on peer AP requires part of an AXML document stored on peer AP2, then there are two options: a) the query Q is decomposed and the relevant sub-query sent to peer AP2 for evaluation, or b) AP1 acquires
Related Content

Participation in Social Networks as Feral Information Systems
www.igi-global.com/chapter/participation-in-social-networks-as-feral-information-systems/94683?camid=4v1a

An Overview of Enterprise Resource Planning for Intelligent Enterprises
www.igi-global.com/chapter/overview-enterprise-resource-planning-intelligent/44064?camid=4v1a

Ontological Analysis of Business Systems Analysis Techniques: Experiences and Proposals for an Enhanced Methodology
www.igi-global.com/chapter/ontological-analysis-business-systems-analysis/6117?camid=4v1a

Introducing the Elasticity of Spatial Data
www.igi-global.com/chapter/introducing-elasticity-spatial-data/44110?camid=4v1a