Uma Abordagem para o Armazenamento de Documentos XML Ativos

Cláudio Ananias Ferraz[1]
Orientação: Vanessa de Paula Braganholo[2], Marta Mattoso[1]
[1]Programa de Engenharia de Sistemas e Computação - COPPE/UFRJ
[2]Departamento de Ciência da Computação – IM/UFRJ
e-mail: {cferraz, marta }@cos.ufrj.br, braganholo@dcc.ufrj.br

1 Introdução

A busca por soluções que possibilitem tanto a integração de aplicações de forma simples quanto o intercâmbio de informações de maneira padronizada fez com que surgissem padrões para publicação e acesso a informações e serviços na Web, tais como XML e Serviços Web. O sucesso destas tecnologias proporcionou o desenvolvimento de uma nova classe de documentos XML, os documentos XML Ativos (AXML) [1]. Documentos XML Ativos possuem chamadas de Serviços Web embutidas em seu conteúdo. A partir da execução destas chamadas de serviços, o resultado das mesmas é materializado dentro do documento XML, enriquecendo este documento com o novo conteúdo dinâmico de informação.

A persistência de documentos AXML apresenta particularidades referentes ao aspecto ativo desta classe de documento XML. A detecção das chamadas de serviços, suas ativações e a gerência dos resultados obtidos a partir dessas chamadas são tarefas que devem ocorrer de forma transparente ao usuário. A manutenção da integridade dos dados também se torna mais complexa, devido à dependência de agentes externos ao SGBD (provedores de serviços e seus canais de comunicação). Essas particularidades devem ser consideradas para um sistema efetivo de armazenamento desses documentos.

A adoção de SGBDs XML nativos pode não ser a melhor alternativa para o armazenamento de documentos AXML quando se busca a integração com sistemas e dados legados, além de tais SGBDs não possuírem recursos explícitos para a definição das partes ativas do documento XML. Da mesma forma, os SGBDRs não possuem esses recursos. Uma analogia pode ser feita ao mecanismo de *triggers,* contudo, este não apresenta o comportamento e a granularidade necessária para os requisitos e comportamentos desejados, o que inviabiliza tal alternativa. Outra alternativa foi usada em [5], contudo o armazenamento consistia em uma abordagem rudimentar, onde os documentos AXML eram armazenados em um diretório no sistema de arquivos, definido pela aplicação implementada. Tal abordagem não fornece controle de concorrência e de transação, não possui controle de acesso e segurança detalhado, nenhum tipo de indexação e compactação de dados, contando apenas com os serviços de gerência de arquivos disponibilizados pelo sistema operacional.

O objetivo deste trabalho é apresentar uma solução para o armazenamento de documentos AXML sob a perspectiva do modelo objeto-relacional, bem como oferecer um mecanismo de consulta eficiente, além de analisar outras possíveis alternativas e trabalhos relacionados a este problema. O trabalho está organizado da forma a seguir. A seção 2 apresenta um cenário real que motiva o desenvolvimento de uma solução para a o problema de armazenamento de dados AXML. A seção 3 abrange trabalhos correlatos. A seção 4 aborda as dificuldades do problema e propõe uma arquitetura para a solução apresentada. A seção 5 apresenta passos futuros do trabalho e algumas considerações.

2 Cenário de Aplicação

Um exemplo de uso de documentos XML ativos é descrito em [2], e trata de um programa governamental de empréstimos. O governo brasileiro possui o PRONAF, um programa de amparo à agricultura familiar. Através deste programa, são realizados empréstimos a famílias que exercem atividades agrárias. O organismo que regulamenta o

crédito é o Banco Central do Brasil (BACEN) e o organismo financiador é o Banco Nacional de Desenvolvimento Social (BNDES). Agricultores espalhados por todo território nacional, individualmente, recorrem ao empréstimo do PRONAF. Os empréstimos são realizados através de agentes locais de crédito, tipicamente bancos regionais.

A atividade de empréstimo é feita pelo BNDES para os agentes locais de crédito. Por sua vez, cada agente está envolvido em várias outras sub-atividades, garantindo o empréstimo aos agricultores. Os agentes locais então fazem pagamentos periódicos ao BNDES. O BNDES mantém o rastro apenas dos empréstimos feitos aos agentes locais. São estes agentes que gerenciam os detalhes dos empréstimos feitos aos agricultores. Atividades de pagamento envolvem computação para contabilizar o montante a ser reembolsado e juros a serem pagos, dependendo do tipo do empréstimo, taxas, data de pagamento, etc. O BACEM deve ser capaz de auditar todo o processo de empréstimo. Contudo, o BACEM somente interage com o BNDES, só o BNDES conhece os agentes de empréstimo e alguns agentes delegam atividades de empréstimo a outros agentes. A Figura 1~~Figura 1~~ apresenta um exemplo de parte de um documento AXML que caracteriza um empréstimo neste cenário e sua evolução a partir da chamada ao Serviço Web que recupera a taxa de juros de um determinado dia.

<doc_emprestimos><#emprestimo>110-334S-H1/2002</#emprestimo>
<taxa> <sc>**BACEN.getTaxaJuro(<dia>14/5/04</dia>)**</sc></taxa>
</doc_emprestimos>

<doc_emprestimos><#emprestimo>110-334S-H1/2002</#emprestimo>
<taxa> <sc>BACEN.getTaxaJuro(<dia>14/5/04</dia>)</sc>**<valor>2.455 %</valor>**</taxa>
</doc_emprestimos>

Figura 1 Exemplo de Documento AXML [2](adaptada)

Esse cenário evidencia a necessidade do armazenamento de um volume grande de dados AXML. Para que estes dados sejam passíveis de auditoria, também é necessário um mecanismo de consulta simples e eficiente, caracterizando a motivação deste trabalho.

3 Trabalhos Relacionados

Na implementação desenvolvida no projeto GEMO [3], o armazenamento de documentos AXML consiste em um diretório de arquivos definido pela aplicação, onde se encontram os documentos AXML, não havendo nenhum armazenamento mais sofisticado destes documentos. Desse modo, pode-se antecipar problemas com o gerenciamento de documentos ativos, o que pode interferir diretamente na escalabilidade da implementação.

Para contornar este problema, algumas extensões foram propostas, como o caso do Xyleme-AXML, uma implementação do Peer AXML integrada com o Xyleme Server [4] (um repositório de dados XML nativo). Na implementação, apesar de carregado sistematicamente como um objeto DOM na memória principal, o documento AXML é armazenado de forma persistente no Xyleme[5]. Contudo, o uso de armazenamento nativo não é a melhor alternativa para aplicações empresariais cujos dados estão fundamentados em tecnologia relacional e objeto-relacional. Tais dados de um modo ou outro se relacionarão com os dados AXML armazenados. Neste caso, o ideal seria usar os SBGDs já existentes. Pesquisas sobre o armazenamento de dados XML mostram que a alternativa objeto-relacional é eficiente para tal propósito[6].

O mapeamento de documentos XML para sistemas relacionais já foi amplamente discutido pela comunidade científica, como em [7] e já foram propostas várias soluções para este problema. Todavia, o tratamento do comportamento ativo implica em novos aspectos envolvendo as chamadas de Serviços Web, as quais são tratadas nesse trabalho. Em sistemas relacionais pode-se fazer uma analogia à ativação de chamadas de serviços com o uso de gatilhos (triggers) [5]. Contudo, uma possível integração de um sistema AXML com sistemas de banco de dados relacionais deve considerar uma classe de *triggers* que não se encontra

implementada na grande maioria dos produtos SGBDs relacionais comerciais. Trata-se de *triggers* que envolvem os eventos de seleção de entidades, ou seja, gatilhos disparados por expressões SQL da forma "*select* <colunas> *from* <tabelas> *where* <predicados>". Esta classe de gatilhos se faz necessária para a ativação automática das chamadas de serviço, uma vez que a existência dessas chamadas é transparente ao usuário. Deve-se considerar também que a granularidade dos gatilhos, que é definida sobre relações, não atende ao requisito desejado.

4 Armazenamento de Documentos XML Ativos

Uma possibilidade mais adequada de integração da tecnologia AXML com sistemas de bancos de dados relacionais é a utilização de sistemas objeto-relacionais e seus recursos de gerência de tipos de dados complexos definidos pelo usuário e métodos associados a estes tipos. Tais métodos serviriam para ativar as chamadas aos Serviços Web. Consideramos que a analogia entre métodos e serviços é bastante próxima, e esta é a abordagem utilizada neste trabalho.

4.1 Arquitetura proposta

A partir de trabalhos propostos na literatura sobre mapeamento de documentos XML para bases de dados relacionais [7-9], e sobre tradução de consultas XML para consultas SQL, estabelecemos um ponto inicial para o mapeamento de documentos XML Ativos para sistemas objeto-relacionais, possibilitando um sistema sofisticado de armazenamento e consulta destes documentos AXML. De acordo com comportamentos e operações desejados em tal sistema, definimos uma arquitetura básica para a solução do problema de mapeamento de documentos AXML para o modelo objeto-relacional.

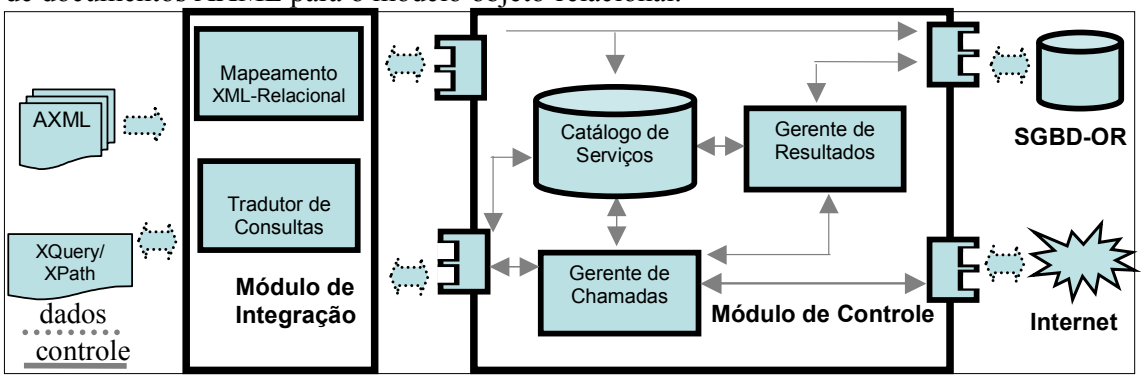

Figura 2-Arquitetura proposta

De acordo com a arquitetura definida na Figura 2~~Figura 2~~, podemos definir uma aplicação onde o documento XML é submetido ao módulo responsável pelo mapeamento dos dados XML para um esquema objeto-relacional. Este mapeamento inclui, além das tarefas normais de tradução de esquema, uma tarefa de identificação de chamadas de serviços. A partir desta etapa, são identificadas as informações necessárias ao catálogo de serviços. Quando o usuário realiza uma consulta sobre o documento XML, o Tradutor de Consultas, baseado nas regras de tradução definidas em [7], traduz a consulta XQuery/XPath para SQL. Definida a consulta em SQL, realiza-se uma consulta ao catálogo de serviços onde é verificada a necessidade de aplicação de chamadas de serviços sobre a consulta e, de acordo com o resultado obtido, a consulta então é reescrita incluindo a chamada de função que é responsável pela execução da chamada ao Serviço Web.

A consulta em SQL traduzida é então submetida ao SGBD objeto-relacional, que encarrega o executor de chamadas de efetivamente fazer a requisição ao Serviço Web. O Gerente de resultados fica responsável por manipular os resultados obtidos e materializar os resultados de acordo com as diretrizes impostas pelo Catálogo de Serviços.

A arquitetura proposta possui como vantagem a integração natural com dados e sistemas legados, além de utilizar técnicas de mapeamento XML-Relacional comprovadamente eficientes [9]. Além disso, pode ser implementada em grande parte dos SGBDs comerciais e livres, como por exemplo, o PostGreSQL, utilizado nesta pesquisa.

4.2 Tratamento de aspectos ativos dos documentos

Ao tratar o aspecto "ativo" de documentos XML dentro da arquitetura proposta (Figura 2~~Figura 2~~), podemos identificar três componentes principais:

a) Identificação de chamadas de Serviços Web: é o momento onde o Módulo de Integração faz a identificação de quais elementos dentro do documento AXML representam uma chamada de serviço e o armazenamento desta informação. Para tal identificação, durante o processo de análise do documento para seu mapeamento para o esquema objeto-relacional, deve-se identificar as marcações que representam chamadas de serviços (**<sc>).** As informações que se encontram dentro desse ramo da árvore do documento são parâmetros utilizados para a realização da chamada ao Serviço Web, e também são tratados. Uma vez identificadas, devem ser armazenadas no Catálogo de Serviços que também deve ser definido no esquema objeto-relacional.

b) A execução da chamada ao Serviço Web: Uma vez traduzida a consulta XML para uma consulta no padrão SQL, deve-se verificar automaticamente a necessidade de realizar chamadas de serviços, tomando como base as informações a serem retornadas ao usuário. O módulo de Controle faz a verificação de acordo com o Catálogo de Serviços e com a própria consulta XPath.

Propomos para a execução de chamadas a Serviços Web, a utilização de funções e procedimentos em linguagens de alto nível definidas e incorporadas internamente ao SGBD objeto-relacional. Dessa forma, nesta etapa, há a necessidade de uma nova reescrita da consulta, onde são incorporadas as funções responsáveis pelas chamadas de serviços. Esta incorporação é feita com base nas informações armazenadas no Catálogo de Serviços e com os critérios de seleção impostos pela consulta. As modificações realizadas na consulta não interferem na estrutura do resultado. Desse modo, a construção do resultado em formato XML não difere do executado no mapeamento XML-Relacional já discutidos pela comunidade. A Figura 3~~Figura 3~~ ilustra um exemplo do processo de tradução e reescrita de consultas.

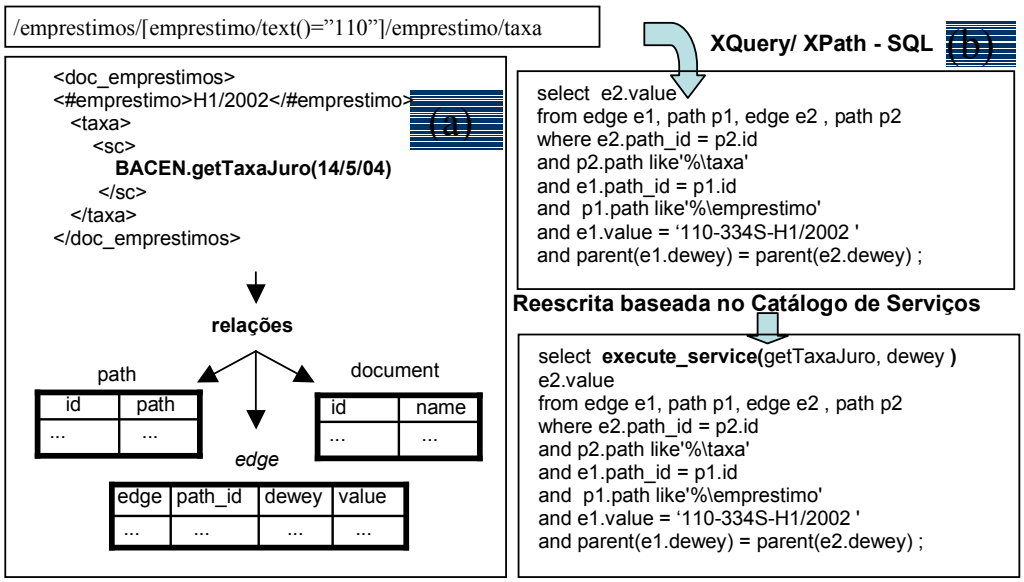

Figura 3 Tradução e reescrita de consultas e mapeamento XML-Objeto-Relacional

Em (a) é ilustrado o modelo de mapeamento XML-Relacional utilizado, em (b) é exemplificado o processo de tradução de consultas. O retângulo superior mostra a tradução da consulta XPath para SQL, e o retângulo inferior mostra a consulta SQL já com a chamada da função responsável pela ativação do Serviço Web.

c) A gerência dos resultados obtidos a partir da execução da chamada: similarmente ao processo de materialização existente nos documentos AXML, a gerência dos resultados obtidos nas chamadas aos serviços Web é um fator complexo. O tratamento dos resultados obtidos ao se acessar um Serviço Web pode requerer diferentes comportamentos de acordo com o modelo AXML [5]. É necessário tratar esses diferentes comportamentos através de uma abordagem relacional generalista. Este fator interferiu na escolha no modelo de mapeamento XML–Relacional adotado, i.e. a abordagem *Edge*, utilizando esquema de ordenação *Dewey* [7], uma vez que, ao trabalhar com um esquema genérico de representação dos documentos XML, dimimui-se a complexidade de gerenciamento dos relacionamentos existentes entre as entidades, bem como as atualizações na base de dados, baseando-se na definição WSDL do serviço utilizado e no comportamento definido na chamada do serviço.

5 Considerações finais

Pretendemos, com esta dissertação, definir regras para o mapeamento de documentos AXML para sistemas de banco de dados objeto-relacionais. Para a validação destas regras, será implementado um conjunto de funcionalidades para o tratamento do conteúdo ativo dos documentos XML, seguindo a arquitetura definida.

Para a implementação da solução apresentada é necessária uma arquitetura sofisticada para gerenciar o Catálogo de Serviços, que incorpore os comportamentos referentes à execução e manipulação de resultados de chamadas de serviços Web. A definição deste catálogo é um fator de grande impacto na arquitetura proposta, e representa uma contribuição importante para alternativas relacionais análogas a este tipo de mapeamento.

Pretendemos verificar a eficiência da nossa proposta para o armazenamento de grandes massas de documentos XML Ativos através de experimentos com medida de desempenho seguindo o AXMLBenchmarks [10].

6 Referências Bibliográficas

1. ABITEBOUL, S., BENJELLOUN, O., MILO ,T. , *Active XML Primer* in *GEMO Report number* 2003, Tech Report INRIA Futurs.
2. RUBERG, N., RUBERG, G., MANOLESCU, I., *Towards cost-based optimization for data-intensive Web service computations.* Proceedings of SBBD, 2004: p. 283-297.
3. ActiveXML. *ACTIVE XML WEBSITE*. 2006 [cited 2006 05/03/2006]; Available from: http://activexml.net/.
4. Xyleme. *XYLEME WEBSITE.* 2006 [cited 2006 05/03/2006]; Available from: http://www.xyleme.com/.
5. ABITEBOUL, S., BENJELLOUN, O., MILO ,T. , *The Active XML project: an overview*. 2005, Tech Report GEMO INRIA.
6. PLOUGMAN, D., LAURITSEN, T., OLESEN J. *Shredding and Querying XML Data Using an RDBMS.* Eletronic Document Libray 2004 [cited; Available from: .http://www.cs.aau.dk/library/files/rapbibfiles1/1085738327.ps
7. TATARINOV, I., VIGLAS, S., BEYER, K., SHANMUGASUNDARAM, J., SHEKITA, E. E ZHANG, C.. *Storing and Querying Ordered XML Using a Relational Database System.* SIGMOD, 2002: p. 204-215.
8. FLORESCU, D., KOSSMAN, D., *Storing and Querying XML Data using an RDBMS.* IEEE Data Engineering Bulletin, 1999. **22**: p. 27-34.
9. FLORESCU, D., KOSSMAN, D. , *A performance evaluation of alternative mapping schemes for storing XML data in a relational database*. 1999, Tech Report INRIA: Paris, France.
10. AXMLBenchmarks. *AXMLBenchmarks*. [cited 29/05/2006]; Available from: http://projects.caixamagica.pt/projects/edos/wiki/AXMLBenchmarks.