Integrate: Infra-Estrutura para Integração de Fontes de Dados Heterogêneas

Rogério Arantes Gaioso[1], **Fábio Nogueira de Lucena**[1], **João Carlos da Silva**[1]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Caixa Postal 131 – 74001-970 – Goiânia – GO – Brasil

roggaioso@yahoo.com.br, {fabio,jcs}@inf.ufg.br

Abstract. *An application generally is designed to use a data source, establishing strong dependence between both, in which changes in one reflect in the other. This work defines an infrastructure, in the form of a framework, to integrate information registered in heterogeneous data sources, from existing applications, without demanding modifications neither in the applications or in the data sources.*

Resumo. *Uma aplicação geralmente é produzida para usar uma fonte de dados, estabelecendo forte dependência entre ambas, na qual mudanças em uma repercutem na outra. Este trabalho propõe uma infra-estrutura, na forma de framework, para integrar informações mantidas em fontes de dados heterogêneas, a partir de aplicações existentes, sem exigir modificações, quer sejam nas aplicações ou nas fontes de dados.*

1. Introdução

O crescimento da internet não tem somente viabilizado o acesso às bases de informação existentes, mas também motivado a criação de novas fontes. Este crescimento tornou mais evidente as limitações atuais para encontrar, extrair e integrar informações distribuídas em fontes de dados heterogêneas. Uma classificação dos diferentes tipos de problemas decorrentes de tais limitações estabelece os níveis de sistema, sintático, estrutural e semântico [Sheth 1998]. Segundo [Sheth 1998], o crescimento na adoção de padrões da internet, CORBA, JDBC, SQL, entre outros, resultou em progresso na solução de interoperabilidade nestes diferentes níveis. O principal desafio da integração de dados, contudo, está localizado no nível semântico. A expectativa é que programas auxiliem não apenas na perspectiva de dados de tais fontes, mas também nas informações nelas contidas, contribuindo com o aumento do conhecimento.

Uma possível solução, adotada em passado recente, seria mover os dados para um novo sistema integrado, criado para prover uma visão unificada de dados de diferentes tipos. A desvantagem desta solução é que estes dados já são adequadamente manipulados pelos sistemas legados, e algumas aplicações necessitariam ser reescritas para trabalhar com este novo sistema integrado, o que não é considerado prático [Roth and Schwarz 1997]. Esta proposta, ainda envolve riscos, custos e tempo que podem ser proibitivos [Barbosa 2001].

Diversas propostas de solução do problema de integração são normalmente específicas, atendendo a uma única aplicação, ou genéricas, para atender diversas situações

de integração [Barbosa 2001]. Novos sistemas de integração são constantemente propostos e resultam de trabalhos isolados, onde não se identifica uma abordagem padronizada que, por exemplo, facilite a interação entre grupos distintos com trabalhos similares e, especialmente, a reutilização. O presente trabalho identifica similaridades entre abordagens geralmente empregadas, propõe uma infra-estrutura que facilite a implementação delas e, dessa forma, potencializa a reutilização. No contexto da integração de fontes de dados heterogêneas, este trabalho concentra-se em cenários nos quais se deseja manter intactas as aplicações e as fontes de dados existentes.

1.1. Problema

Em um sistema de informação típico, as informações pertinentes a um determinado domínio são manipuladas por uma aplicação e registradas em uma fonte de dados em um determinado formato, geralmente relacional. Para manipular a fonte de dados, sem perda de generalidade, a aplicação faz uso de JDBC, que fornece serviços de transporte das requisições e dos resultados correspondentes. Em conseqüência, a aplicação depende de JDBC e do esquema da fonte de dados. O esquema e a fonte de dados são interdependentes, ou seja, qualquer alteração em um provoca alteração no outro e vice-versa. Como não é possível eliminar a dependência por algum mecanismo de transporte de requisições, a dependência da aplicação para JDBC é considerada "aceitável", neste trabalho. Já a dependência da aplicação para o esquema dos dados é mais "nociva", pois impede que aplicações existentes, sem alterações, empreguem outras fontes de dados mesmo quando ambas pertencem a um domínio particular.

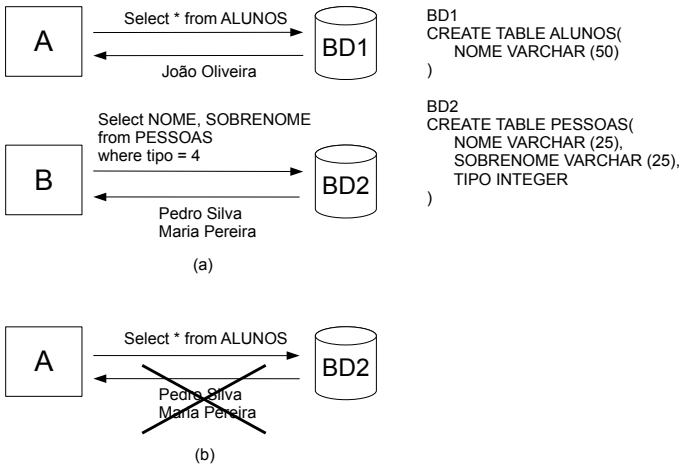

Figura 1. (a) Cenário real. (b) Cenário desejado.

Observe o cenário ilustrado na figura 1, onde as bases de dados *BD1* e *BD2* possuem esquemas diferentes. As aplicações *A* e *B*, construídas de maneira dependente dos esquemas de *BD1* e *BD2*, respectivamente, para interagir com estas bases, necessitam executar as consultas exibidas na figura 1-a para obter a relação de alunos em cada uma delas. Para contornar a dependência que a aplicação *A* tem do esquema de *BD1* para que possa acessar as informações de *BD2*, as maneiras possíveis seriam (i) alterar a aplicação *A* ou (ii) migrar os dados de *BD2* com as devidas transformações para um esquema correspondente a *BD1*. Ambas as alternativas são, em geral, impraticáveis pelos custos decorrentes, e, neste trabalho, consideradas insatisfatórias, principalmente quando considerado

um cenário mais complexo, onde várias aplicações legadas dependem de um esquema de dados. Ou seja, a aplicação *A*, sem sofrer alterações, não tem como acessar *BD2* naturalmente, conforme ilustrado na figura 1-b.

Ao contrário destas alternativas, a proposta aqui apresentada visa facilitar a implementação de uma solução de um problema de integração de dados onde uma aplicação *A* possa fazer uso de uma base de dados *BD2*, diferente em vários níveis da base *BD1* para a qual foi construída, sem que esta precise ser modificada, ou seja, não é necessário acesso ao código-fonte de *A* nem a migração dos dados de *BD2* para um formato semelhante ao de *BD1*.

A solução proposta, após devidamente configurada, permite que requisições de uma aplicação, sem que esta seja alterada, consigam manipular outra fonte de dados, com esquema diferente daquele originalmente manipulado. Estas requisições, que no cenário original são enviadas diretamente à fonte de dados, são interceptadas e convertidas em outras compatíveis com as fontes de dados heterogêneas desejadas. O resultado parcial de cada fonte de dados é então integrado e convertido para o formato conhecido pela aplicação requisitante. Este processo ocorre de forma transparente para a aplicação.

Este texto encontra-se organizado em seções. A seção 1.2 apresenta fundamentos pertinentes à integração de dados. A seção 2 apresenta o Integrate, e inclui o modelo de solução adotado. A seção 3 descreve as decisões de projeto de software. A seção 4 resume o que foi realizado e ressalta as contribuições obtidas.

1.2. Fundamentos

Este trabalho tem como principal produto o *framework* denominado Integrate. *Framework* é uma arquitetura semi-completa que pode ser instanciada para produzir aplicações personalizadas, permitindo a reutilização de análise, de projeto e de código [Barbosa 2001], auxiliando a construção de aplicações inseridas em um domínio particular. Pode ser visto como uma composição de classes concretas e abstratas, cuja instanciação consiste da composição de novas classes e a extensão de classes abstratas predefinidas [Buschmann et al. 1996]. Sua utilização minimiza o esforço no desenvolvimento de aplicações ao permitir que o desenvolvedor se abstraia de preocupações na definição da arquitetura do sistema.

A principal função de um sistema de integração de dados é disponibilizar uma interface capaz de atender requisições que normalmente requerem extração e combinação de dados originários de múltiplas fontes distintas e heterogêneas [Wiederhold 1992]. Uma das abordagens disponíveis é a virtual [Busse et al. 1999].

A arquitetura mediador/tradutor [Wiederhold 1992], que implementa esta abordagem, conceitualmente é dividida em três camadas: aplicações clientes, mediador e tradutores. Nesta arquitetura, uma das funções do mediador é definir um esquema global que integra os esquemas de todas as fontes de dados [Shvaiko and Euzenat 2005], e uma maneira de eliminar os conflitos semânticos durante a modelagem é definir o domínio através de ontologias [Uschold and Grüninger 1996]. As aplicações requisitam o acesso aos dados através do mediador, que gera as subconsultas pertinentes e as repassa aos tradutores correspondentes em um formato comum. Cada tradutor converte a consulta deste formato comum para o da fonte de dados por ele manipulada e a acessa para obter os dados. Os dados obtidos são então convertidos pelo tradutor para o formato comum e devolvidos ao

mediador. Este providencia a integração dos dados obtidos e os entrega à aplicação que fez a requisição.

Diversas propostas de sistemas de integração que empregam estas técnicas foram levantados e serviram de insumo para este trabalho, como Integra [Lóscio 2003], MOMIS [Università di Modena 2004] e LORIS [Moura et al. 2005].

2. Modelo conceitual

Integrate [Gaioso 2007] é uma proposta na forma de *framework* a ser empregada por desenvolvedores de código de sistemas de integração de dados heterogêneos, e visa oferecer serviços que facilitem a geração deste tipo de sistemas. A infra-estrutura fornecida pode ser estendida para contemplar as especificidades do caso de integração em questão, e se baseia na arquitetura de mediadores e tradutores. Além de contribuir com as pesquisas na área, contemplando uma abordagem recorrente e de fácil utilização, um dos principais objetivos do *framework* é possibilitar a integração sem necessariamente exigir alterações nas aplicações clientes e nas fontes de dados originais, diferente da maioria das propostas pesquisadas, que fornecem seus serviços através de interfaces próprias.

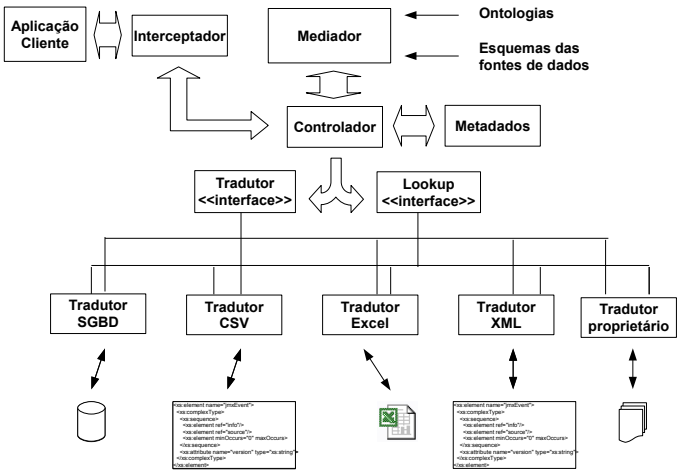

Figura 2. Modelo de solução estendido

2.1. Modelo de solução

Semelhante à maioria dos trabalhos pesquisados, o Integrate adota a arquitetura mediador/tradutores tradicional. Para permitir a integração de dados onde as fontes de dados e as aplicações não sejam alteradas, o modelo de solução inclui um interceptador, que age entre a aplicação cliente e o mediador, conforme a figura 2.

Os passos a seguir ilustram o processo de consulta no modelo estendido. Uma aplicação cliente envia uma requisição no formato original para a qual foi construída. O Interceptador intercepta esta requisição e a repassa ao Controlador, que a redireciona ao Mediador. O Mediador retorna as subconsultas referentes às fontes de dados envolvidas nos seus respectivos formatos. As subconsultas são repassadas às fontes de dados correspondentes através dos Tradutores, e os resultados parciais de cada fonte de dados retornam ao Controlador. O Controlador repassa os resultados parciais ao Mediador, que executa a integração dos resultados parciais no formato conhecido pela aplicação cliente

e retorna o resultado integrado ao Controlador. O resultado integrado é então entregue à aplicação pelo Interceptador. Assim, a consulta a fontes de dados diferentes daquela para a qual a aplicação cliente foi produzida ocorre de forma transparente.

Este tipo de conversão de requisição (solicitada ao Mediador) é o trabalho típico realizado por algumas ferramentas levantadas [Alexiev et al. 2005, Hernández et al. 2001, Lóscio 2003]. Segundo [Halevy 2001], reescrever estas requisições não introduz dificuldades, pois é possível criar visões sobre o esquema mediado que espelham precisamente o esquema original. Porém, [Lóscio 2003] cita esta reescrita de sentenças como um problema crítico para sistemas de integração, principalmente em cenários onde há alterações dos esquemas após a integração dos mesmos.

2.2. Modelo de dados e troca de mensagens

A interação entre os módulos é feita através do modelo relacional, mas é possível o uso de XML como formato das mensagens trocadas entre o Controlador e o Mediador, visto que este formato oferece flexibilidade para representar dados estruturados e semi-estruturados e facilidade de conversão de/para dados relacionais [Wiederhold 1992]. Embora o uso de XML não elimine os problemas de heterogeneidade semântica [Cui et al. 2002], sua aplicação tem sido aceita como uma forma de fornecer uma sintaxe comum para a troca de informações entre fontes heterogêneas, além de ser um formato inteligível por seres humanos.

2.3. Escopo

Na solução aqui proposta, há ênfase nas funções atribuídas aos Tradutores e na interação entre estes e o Mediador. Em conseqüência, mediadores propriamente ditos, ontologias e outras questões correlatas estão além do escopo do Integrate. Noutras palavras, o Integrate fornece recursos para apoiar integrações que se baseiem no modelo de solução apoiado.

Os serviços disponibilizados pelo Integrate permitem a criação de mediadores sem que estes conheçam necessariamente os detalhes de acesso às fontes de dados a serem integradas. Cabe ao Mediador as questões semânticas e a interação com as aplicações clientes (em cenários tradicionais). No caso do cenário estendido (figura 2), cabe ao Integrate intermediar as conversões entre o Mediador e as fontes de dados, além da interação com as aplicações clientes. Desta maneira, o escopo do Integrate engloba a definição e a implementação do *Interceptador*, do *Controlador*, do *Tradutor*, do *Lookup* e do acesso aos metadados.

3. Projeto e implementação

O Integrate adota um modelo de solução inspirado na arquitetura mediador/tradutor, e oferece serviços por meio de um *framework*. A arquitetura de software é descrita por várias perspectivas, similar ao sugerido em [Kruchten 1995], registrado essencialmente por meio de diagramas UML.

O *framework* foi implementado em Java. O download e uma descrição mais detalhada dos módulos pode ser encontrada no portal do projeto [Gaioso 2007].

3.1. Metadados

Uma das principais tarefas executadas antes do uso do Integrate é o devido preenchimento dos seus arquivos de configuração. Os principais arquivos de configuração do sistema são:

- **integrate-datasources.xml** - Principal arquivo de configuração do sistema, onde são definidas informações referentes a cada fonte de dados que se deseja integrar. Cada fonte de dados contém um identificador único.
- **integration.xml** - Define as integrações desejadas. Cada integração possui um identificador, que corresponde ao que é utilizado no URL de conexão do *driver* JDBC implementado pelo Interceptador. Neste arquivo definem-se as fonte de dados de origem a de destino que se deseja integrar, sendo uma única de origem (a fonte de dados original para o qual a aplicação cliente foi inicialmente desenvolvida) e uma ou mais de destino. Para isso, utilizam-se os identificadores definidos no arquivo *integrate-datasources.xml*. Logo, há uma dependência entre eles.

3.2. Controlador

Inspirado no CoDIMS[1] [Barbosa 2001], este módulo intermedeia as trocas de mensagens com o Mediador e também entre os demais módulos do Integrate.

3.3. Tradutores

Como o modelo de dados do Integrate é o relacional, os tradutores que manipulam fontes de dados relacionais podem utilizar os *drivers* JDBC específicos para as fontes de dados, funcionando apenas como um *proxy*, repassando a sentença SQL à fonte de dados manipulada e devolvendo ao Controlador o resultado obtido. Tradutores de outros formatos devem fazer as conversões necessárias, através de classes que implementam uma interface específica.

3.4. Lookup

O Integrate auxilia o Mediador na definição do esquema global, ao disponibilizar serviços que facilitam a obtenção dos esquemas das fontes a serem integradas.

3.5. Interceptador

Um conjunto de classes implementa um *driver* JDBC que intercepta as requisições feitas por aplicações clientes.

Para a utilização do Interceptador, a aplicação cliente deve fazer uso do URL de conexão específico do Integrate, no formato **jdbc:integrate:id**, onde o **id** é o identificador da integração desejada, configurada no arquivo *integration.xml* (ver seção 3.1), determinando univocamente o cenário desejado pelo cliente.

A requisição interceptada é repassada ao Controlador, junto com o identificador da integração desejada, para que seja feita a integração dos resultados. Quando o *ResultSet* integrado é retornado ao Interceptador, este o devolve à aplicação cliente.

3.6. Mediador

Para utilizar o Integrate no modelo estendido, o desenvolvedor do mediador deve implementar a interface *Mediator*.

Inicialmente o Integrate comunica-se com o Mediador através de JDBC puro, o que exige deste o conhecimento de SQL e de JDBC. Uma das propostas futuras do Integrate é definir um protocolo de mais alto nível, que permita esta troca de informação através de mensagens que eximam do Mediador este conhecimento.

[1] Embora o referido sistema não implemente a arquitetura de mediadores.

Sugere-se o emprego de ontologias para resolver as heterogeneidades semânticas das fontes de dados, mas esta decisão fica a cargo do desenvolvedor do mediador que utilizar o Integrate. Nenhum suporte ao emprego de ontologias é oferecido na versão corrente do *framework*.

4. Conclusão

A literatura especializada disponibiliza diversas propostas de solução para o problema de integração de fontes de dados heterogêneas. Em geral, a interação com o usuário ocorre por meio de uma interface definida pela proposta de integração em questão. Estes casos são incompatíveis com o cenário onde as fontes de dados e a aplicação cliente, que faz uso destas fontes, não podem passar por modificações. Os motivos são vários, por exemplo, custos e até mesmo a indisponibilidade de código fonte.

A proposta apresentada neste trabalho oferece serviços por meio de um *framework*, denominado Integrate, baseado na arquitetura mediador/tradutores, para auxiliar desenvolvedores na construção de sistemas de integração de dados heterogêneos. Além de oferecer serviços para a confecção de sistemas tradicionais que pretendam integrar dados heterogêneos, ao contrário de várias outras abordagens, o Integrate é adequado onde não se espera alteração no esquema das fontes de dados e/ou acesso ao código-fonte das aplicações que as empregam. O Integrate é particularmente útil nestes casos, funcionando como uma camada intermediária entre o novo esquema e estas aplicações, executando as devidas transformações enquanto as aplicações não são alteradas.

O *framework* estabelece uma arquitetura de software para os sistemas que o adotam. Isto inclui componentes como mediador e tradutores bem como as comunicações entre estes componentes. O objetivo é facilitar a substituição independente destes módulos por outros sem comprometer a comunicação entre eles.

4.1. Contribuições

Uma das vantagens desta proposta está na definição dos serviços oferecidos, através de uma abordagem recorrente entre as propostas conhecidas (baseada em mediadores), o que proporciona uma maior aplicabilidade e a reutilização de código e de projeto. Estes serviços disponibilizam uma API que permite a evolução independente tanto de quem a usa quanto de quem a implementa.

Por trabalhar internamente com o modelo relacional, o Integrate oferece seus serviços através de JDBC. Para facilitar a sua utilização, o *framework* também fornece seus serviços que retornam mensagens no formato XML, formato comumente utilizado pelas comunidades envolvidas com questões semânticas.

Outra contribuição da presente proposta está na disponibilização do código-fonte produzido, o que a difere da maioria das propostas levantadas. O presente trabalho, a versão completa da dissertação, bem como o código-fonte e sua documentação, estão disponíveis em [Gaioso 2007].

Referências

Alexiev, V., Breu, M., de Bruijn, J., Fensel, D., Lara, R., and Lausen, H. (2005). *Information Integration with Ontologies: Ontology based Information Integration in an Industrial Setting*. John Wiley & Sons.

Barbosa, A. C. P. (2001). *Middleware para Integração de Dados Heterogêneos Baseado em Composição de Frameworks*. PhD thesis, PUC-Rio de Janeiro, Brasil.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. (1996). *Pattern-oriented Software Architecture: a System of Patterns*. John Wiley & Sons, Inc., New York, NY, USA.

Busse, S., Kutsche, R.-D., Leser, U., and Weber, H. (1999). Federated Information Systems: Concepts, Terminology and Architectures. Technical Report Forschungsberichte des Fachbereichs Informatik 99-9, Technische Universität Berlin.

Cui, Z., Jones, D. M., and O'Brien, P. (2002). Semantic B2B Integration: Issues in Ontology-Based Applications. *SIGMOD Record*, 31(1):43–48.

Gaioso, R. A. (2007). Integrate. Disponível em http://integrate.sourceforge.net/, acessado em 21/01/2008.

Halevy, A. Y. (2001). Answering Queries Using Views: A Survey. *The VLDB Journal*, 10(4):270–294.

Hernández, M. A., Miller, R. J., and Haas, L. M. (2001). Clio: a Semi-Automatic Tool for Schema Mapping. In *SIGMOD '01: Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, page 607, New York, NY, USA. ACM Press.

Kruchten, P. (1995). The 4+1 View Model of Architecture. *IEEE Software*, 12(6):42–50.

Lóscio, B. F. (2003). *Managing the Evolution of XML-based Mediation Queries*. PhD thesis, Universidade Federal de Pernambuco, Recife, Brasil.

Moura, S. L., da Silva, F. J. C., Siqueira, S. W. M., and Melo, R. N. (2005). LORIS: Integrating Distributed and Heterogeneous Metadata Repositories of Learning Objects. In *3rd PGL International Conference - Consolidating eLearning Experiences, São Paulo*.

Roth, M. T. and Schwarz, P. (1997). A Wrapper Architecture for Legacy Data Sources. Proc. VLDB Conference.

Sheth, A. P. (1998). Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. In *Interoperating Geographic Information Systems. Kluwer. 5–30*.

Shvaiko, P. and Euzenat, J. (2005). A Survey of Schema-Based Matching Approaches. In *J. Data Semantics IV*, pages 146–171.

Università di Modena (2004). The MOMIS Project. Disponível em http://www.dbgroup.unimo.it/Momis/, acessado em 30/10/2007.

Uschold, M. and Grüninger, M. (1996). Ontologies: Principles, Methods, And Applications. *Knowledge Engineering Review*, 11(2):93–155.

Wiederhold, G. (1992). Mediators in the Architecture of Future Information Systems. volume 25, pages 38–49, Los Alamitos, CA, USA. IEEE Computer Society Press.