NavigationPlanTool: Uma Ferramenta para o Controle de Processos no Modelo de Dados Relacional

Kelly Rosa Braghetto[1], Osvaldo K. Takai[1], João Eduardo Ferreira[1], Calton Pu[2]

[1]Instituto de Matemática e Estatística – Universidade de São Paulo (USP)
Rua do Matão, 1010 – Cidade Universitária – 05508-090 – São Paulo – SP – Brasil

[2]College of Computing – Georgia Institute of Technology
801 Atlantic Drive, Atlanta GA 30332-0280 – USA

{kellyrb, takai, jef}@ime.usp.br, calton@cc.gatech.edu

Abstract. *The need to expand the business cycle processing and to represent it in a relational database system can be noticed mainly in several efforts of transactional control in initiatives as web services composition and control of business processes. The main challenge is in enlarging the capacity of semantic representation of triggers for situations in which the processing of pre and post conditions needs explicit representation for the control of the referred complex relationships. This work presents NavigationPlanTool, an implementation of the Navigational Plan's Definition Language as extension of SQL language and provides mechanisms for the creation and manipulation of data structure for processes representation in relational data model.*

Resumo. *A necessidade de expandir o processamento de ciclo de negócios e representá-lo em um banco de dados relacional pode ser vista no esforço de garantir controle transacional em iniciativas como composição de serviços web e controle de processos de negócio. O desafio maior é enriquecer a capacidade semântica de representação de triggers quando o processamento de pré e pós-condições necessite de representação explícita para o controle de relacionamentos complexos. Este trabalho apresenta a ferramenta NavigationPlanTool, que implementa a Linguagem de Definição de Plano Navegacional como uma extensão da SQL, fornecendo meios para a criação e manipulação de estruturas de dados para a representação de processos.*

1. Introdução

A agilidade com a qual os processos de validação e controle de dados são criados e disponibilizados nas aplicações afetam diretamente a vantagem competitiva das instituições. Em especial, a agilidade no controle de processos em ambientes cooperativos é um aspecto fundamental para o sucesso das aplicações desenvolvidas na plataforma de serviços *web*.

Sistemas de informação geralmente são compostos por dados e mecanismos de controle a estes dados (processos*)*. A abordagem escolhida para o presente trabalho baseia-se no conceito de Plano Navegacional da arquitetura *RiverFish* [Ferreira, Takai e Pu 2005]. A arquitetura *RiverFish* está fundamentada na Álgebra de Processos [Fokkink 2000]. Tal arquitetura define uma linguagem de manipulação de processos para o

modelo relacional, cujos comandos se baseiam nos principais operadores da Álgebra de Processos.

A linguagem de manipulação de processos como uma extensão SQL viabiliza a representação e controle de fluxos, podendo, nestes casos, substituir o uso de *triggers* em cascata. Nesta abordagem, não existe restrições à implementação das regras como as impostas pelo conceito de confluência [Zaniolo, Ceri, Faloutsos, Snodgrass, Subrahmanian e Zicari 1997].

A ferramenta *NavigationPlanTool* implementa a linguagem de manipulação de processos da Arquitetura *RiverFish* e pode ser aplicada na composição de serviços básicos, como os especificados pelo paradigma SOC (*Service-Oriented Computing*) [Georgakopoulos 2003]. Esta abordagem é particularmente interessante em um ambiente *web*, no qual os serviços que compõem um processo de validação ou controle de dados podem ser heterogêneos e autônomos. Para o processo de validação é necessário um mecanismo de controle transacional para o tratamento apropriado de exceções como uma falha ou a indisponibilidade de um serviço durante a execução do fluxo. A representação de um processo e suas instâncias dentro de um gerenciador de banco de dados possibilita este controle transacional de forma mais segura e genérica.

2. A Ferramenta Desenvolvida

A *NavigationPlanTool* disponibiliza uma biblioteca de funções que implementa uma extensão da linguagem SQL, com adição de operadores da Álgebra de Processos que auxiliam a criação e manutenção da representação de um processo dentro de um banco de dados relacional. A ferramenta, além da execução de comandos do SQL padrão, possibilita a execução de comandos para a definição de fluxos - como os de criação, remoção e alteração de processos e ações - bem como comandos que definem a associação e desassociação de ações a processos. A sintaxe dos comandos que realizam estas operações é a definida pela *Navigation Plan Definition Language* (NPDL). Uma síntese dessa linguagem será apresentada na seção 3 deste texto. A biblioteca provê também mecanismos para o controle da instanciação de processos no banco, através de operações como criação/remoção de instâncias e serviços para o monitoramento da execução do plano navegacional. Estes serviços são responsáveis pela manutenção dos *logs* de execução de planos no banco de dados e recuperação de execuções que tenham sido interrompidas antes de serem finalizadas.

A linguagem de programação escolhida para o desenvolvimento da *NavigationPlanTool* foi Java (*Java 2 Platform Standard Edition* - J2SE 5.0), por ser portável e por contar com a JDBC API - J*ava DataBase Conectivity Application Programming Interface*. A JDBC possibilita que programas Java executem comandos SQL e interajam com bancos de dados compatíveis com SQL, o que torna a implementação independente do sistema gerenciador de banco de dados.

Por ter sido desenvolvida na forma de biblioteca de funções, a *NavigationPlanTool* pode ser integrada de forma fácil a outras aplicações Java.

3. A linguagem NPDL

Nesta seção, apresentamos uma síntese da linguagem. Em especial, na tabela 1 apresentamos detalhes da sintaxe da linguagem no formato BNF (Bachus-Nour Form).

Tabela 1. Sintaxe da NPDL no formato BNF

```
<S> ::= (<command>? '\n')+
<command> ::= <new-process>
        | <new-action>
        | <add-process-service-description>
        | <add-action-execution-call>
        | <process-expression>
        | <delete-process>
        | <delete-action>
        | <list-processes>
        | <list-actions>
<new-process> ::= CREATE PROCESS <process-description>
                                          [<service-description>]
<new-action>  ::= CREATE ACTION <action-description>
                                          [<execution-call>]
<add-process-service-description> ::= ADD <process-description>
                        SERVICE DESCRIPTION <service-description>
<add-action-execution-call> ::= ADD <action-description>
                                          EXECUTION CALL <execution-call>
<attribution-operator-sign> ::= =
<process-expression> ::= SET <process-description>
                                  <attribution-operator-sign> <process>
<delete-process> ::= DROP PROCESS <process-description>
<delete-action>  ::= DROP ACTION <action-description>
<list-processes> ::= SELECT PROCESSES
<list-actions >  ::= SELECT ACTIONS
<left-parentheses>  ::= (
<right-parentheses> ::= )
<alternative-operator-sign> ::= *
<sequential-operator-sign>  ::= +
<process> ::= <term>
        | <process> <alternative-operator-sign> <term>
<term>    ::= <factor>
        | <term> <sequential-operator-sign> <factor>
<factor>  ::= <action-description>
        | <process-description>
        | <left-parentheses> <process> <right-parentheses>
<action-description>  ::= <identifier>[1]
<process-description> ::= <identifier>
<service-description> ::= <character_string_literal>2
<execution-call>      ::= <character_string_literal>
```

Exemplos de definição de processos e ações através de comandos da NPDL:

```
   CREATE ACTION A1 ‘VerificarDadosDoPedido’;
   CREATE ACTION A2 ‘PriorizarItensDoPedido’;
   CREATE ACTION A3 ‘IniciarProcessoDeCompra’;
   CREATE ACTION A4 ‘CancelarPedido’;
   CREATE PROCESS P1 ‘Processo de Aquisicao’;
   SET P1 = A1 . ( A4 + A2 . (A3 + A4 ) );
```

[1] <**identifier**> está definido na especificação BNF do ANSI SQL92.

[2] <**character_string_literal>** está definido na especificação BNF do ANSI SQL92.

A representação dos processos por meio de expressões algébricas baseadas em Álgebra de Processos possibilita a criação das árvores de expressão que são utilizadas pelos algoritmos que determinam a ordem de execução das ações que compõem o processo. A Álgebra de Processos constitui um arcabouço para o raciocínio formal sobre processos e dados; ela pode ser usada para detectar propriedades indesejáveis e formalmente derivar propriedades desejáveis da especificação do sistema.

Os comandos da NPDL são convertidos pela ferramenta *NavigationPlanTool* em comandos SQL padrões; estes comandos inserem, alteram, removem e listam dados em tabelas especialmente criadas para representar o controle de processos da arquitetura *RiverFish* em um banco de dados relacional.

4. Estudo de Caso da Biblioteca "Carlos Benjamin de Lyra" do IME, USP

Dentro de uma biblioteca há vários setores; os mais comuns são os de Acervo, Controle de Usuários, Empréstimos e Controle de Aquisição de Acervo. Como resultado de um projeto de informatização da biblioteca do instituto, os alunos do curso de graduação em Ciência da Computação do Instituto de Matemática e Estatística da USP, sob a supervisão do professor responsável pela área de Banco de Dados e com a ajuda de uma equipe de alunos da pós-graduação, trabalharam na modelagem de um banco de dados modular e flexível [Ferreira e Finger, 2000], que pudesse também atender às necessidades de outras bibliotecas.

Foi utilizado o arcabouço *Naked Objects* (NO) [Pawson e Matthews 2002][3] para o desenvolvimento de protótipos e possibilitar a interação entre usuários e as principais entidades do sistema, com o objetivo de validar os modelos criados e detectar falhas ainda na fase de modelagem. O arcabouço será apresentado na subseção 4.1.

A detecção de módulos com características de fluxos de atividades na modelagem dos dados da biblioteca e as limitações dos *Naked Objects* motivaram o desenvolvimento de uma ferramenta capaz de controlar a manutenção, instanciação e execução de processos – a *NavigationPlanTool* – e que pudesse ser acoplada de forma fácil a outras aplicações (não necessariamente desenvolvidas com o arcabouço NO).

O sistema para o Controle de Aquisição de Acervo da biblioteca do IME alia o arcabouço NO à *NavigationPlanTool* e funciona como uma boa demonstração de uso da ferramenta. A descrição do caso será feita na subseção 4.2 .

4.1. O Arcabouço *Naked Objects*

Os *Naked Objects* são objetos comportamentalmente completos, assim denominados por Pawson e Matthews [2002]. Um objeto comportamentalmente completo não apenas conhece os atributos da entidade do domínio a qual representa, mas também sabe como modelar o comportamento dessa entidade. Isso faz com que *Naked Objects* sejam flexíveis o bastante para lidar com as mudanças inesperadas nos requisitos do sistema.

A partir desta motivação, foi criado o arcabouço *Naked Objects* - um pacote de software *open-source* escrito em Java que facilita a construção completa de aplicações

[3] Uma tradução para o português do livro de referência pode ser obtida no endereço: http://www.ime.usp.br/~jef/NOportugues.pdf.

de negócio a partir de *Naked Objects*. O arcabouço integra um ambiente de execução que inclui um mecanismo de visualização que cria, em tempo real, uma representação manipulável de qualquer *naked object* que o usuário precise acessar. Entretanto, o arcabouço NO, ao mesmo tempo em que apresenta soluções inovadoras para identificação de objetos e seus relacionamentos, expõe um dos seus maiores limites: a impossibilidade de tratar funções transversais aos objetos. O arcabouço NO ajuda a manter a completeza comportamental dos objetos, mas impossibilita a definição dos processos que fazem parte da especificação de um sistema porque somente é capaz de lidar com os processos intra-objetos, não provendo mecanismos para o tratamento de processos comuns a mais de um objeto.

O processamento de regras de negócio é um aspecto transversal aos objetos de um sistema e requer, portanto, um tratamento especial. O uso da ferramenta *NavigationPlanTool* é capaz de atender tal necessidade.

4.2. O processo de Aquisição de Acervo e a Ferramenta *NavigationPlanTool*

Durante o processo de modelagem do banco de dados da biblioteca, os alunos se depararam com módulos com características difíceis de serem representadas dentro do modelo relacional. Isto ocorre porque existem módulos que são mais bem caracterizados por fluxos de processos que por relacionamentos entre entidades. Um exemplo disso aparece no setor de Aquisição de Acervo de uma biblioteca: um item de acervo passa por vários passos entre o momento em que é solicitado e o momento em que é colocado na estante e disponibilizado para empréstimo. Cada passo pode mudar o estado do item de acervo; alguns passos podem ser repetidos dentro do processo ou então interrompidos e depois retomados. As principais preocupações relacionadas ao fluxo são os requisitos mínimos (*consistências*) para que se possa passar de um estado para outro. No módulo de Aquisição, o controle da ordem em que são executadas as consistências e passos que compõem o processo é tão importante quanto o armazenamento das informações envolvidas no processo todo; cada novo pedido de compra é uma instância deste processo. Detalhes do processo de aquisição podem ser encontrados na página da internet http://malariadb.ime.usp.br/mac439/projetobiblioteca.

Em implementações tradicionais de sistemas com estas características, todo o controle do fluxo de processos é feito no próprio código fonte da aplicação, o que torna a definição dos processos rígida: mudanças na definição implicam em alterações no código do sistema. Em casos especiais de processos, é possível realizar este controle através de regras de “evento-condição-ação” para sistemas de banco de dados ativos [Zaniolo, Ceri, Faloutsos, Snodgrass, Subrahmanian e Zicari 1997]. Uma das desvantagens desta última abordagem é a dificuldade na implementação de regras que não violem o conceito de terminação e confluência.

Com a *NavigationPlanTool* foi possível desenvolver uma aplicação integrada ao arcabouço NO para o controle de aquisição através da qual a definição do fluxo pode ser feita de forma fácil (usando comandos da NPDL) e em tempo de execução. A ferramenta mantém os processos, suas definições e suas instâncias armazenados em um banco de dados relacional, provendo também serviços para o monitoramento da execução das instâncias.

5. Conclusão

A abordagem tradicional para o tratamento de um módulo como o de Aquisição apresentado na seção 4.2 seria manter o controle de processos fora do banco de dados e implementá-lo como parte da aplicação computacional. As principais desvantagens desta abordagem é que ela torna o processo rígido e, muitas vezes, não possibilita o reaproveitamento de passos já definidos em outros processos. Além disso, o uso convencional e exagerado de *triggers* tem gerado ineficiência no tempo de resposta para as aplicações.

Para que as consistências e passos envolvidos no processo de Aquisição fossem levantados e validados de forma precisa pelos usuários do sistema da biblioteca, novamente usou-se o arcabouço NO para a construção de uma aplicação. Mas, como discutido na seção 4 deste documento, embora o arcabouço NO possibilite uma boa interação entre usuários e modelagem de objetos, ele não oferece recursos apropriados para o tratamento de processos, uma vez que seu foco é somente objetos e os relacionamentos entre eles.

Usando a *NavigationPlanTool* - a biblioteca de funções de extensão do SQL que implementa a NPDL, foi possível enriquecer o poder de expressividade do arcabouço NO e construir objetos que representam processos, compor as ações (passos) que os definem e, além disso, criar requisições (instâncias) destes processos. Os processos, uma vez representados na forma de *naked objects*, podem ser criados e terem sua definição alterada em tempo de execução da aplicação. Isto implica que o uso da *NavigationPlanTool*, aliada ao arcabouço NO, possibilitou a construção de uma aplicação para o controle do processo de Aquisição que permite que os passos que compõem o processo sejam alterados sem a necessidade de se alterar o código fonte da aplicação.

6. Referências

Ferreira, J.E., Finger, M. (2000) “Controle de Concorrência e Distribuição de Dados: Teoria Clássica, suas Limitações e Extensões Modernas”. In: XII Escola de Computação. p. 73-83, 161-174.

Ferreira, J.E., Takai, O.K., Pu, C. (2005) “Integration of Business Processes with Autonomous Information Systems: A Case Study in Government Services”. (artigo aceito para o IEEE CEC 2005 - http://cec05.in.tum.de/index.htm)

Fokkink, W.J., Introduction to Process Algebra - Texts in Theoretical Computer Science, Springer-Verlag, 2000.

Georgakopoulos,D. (2003) “Service Oriented Computing”, Communications of the ACM, v. 26, n. 10, p.25-28.

Pawson, R., Matthews, R., Naked Objects, John Wiley and Sons Ltd., 2002. http://www.nakedobjects.org

Zaniolo, C., Ceri, S., Faloutsos, C., Snodgrass, R., Subrahmanian, V.S., Zicari, R., Advanced Database Systems, Morgan Kaufmann Publishers, 1997.