SINCRONIZANDO BASES DE DADOS EM AMBIENTES DISTRIBUÍDOS PARTE II: MAPEAMENTOS AVANÇADOS

Márcio Rodrigo Melo MARTINS; Ricardo Johannsen M. C. PEREIRA; Rafael Fernandes LOPES; Paulo Cesar Viana VIEIRA; Omar A. C. CORTES

Instituto Federal de Educação, Ciência e Tecnologia do Maranhão, Av. Getulio Vargas, 04, Monte Castelo – São Luis – MA
marcio.martins@ifma.edu.br, ricardo.pereira@ifma.edu.br, rafaelf@ifma.edu.br, paulo.vieira@ifma.edu.br, omar@ifma.edu.br

RESUMO

O Sincronizador é um software *stand-alone* cujo objetivo é manter atualizadas bases de dados distribuídas. Este trabalho apresenta as inovações implementadas no desenvolvimento do Sincronizador. Especificamente, este artigo trata de mapeamentos avançados, os quais têm por objetivo manter as bases de dados consistentes após o processo de sincronização de dados, ou seja, após a transferência de uma base de dados para outra. Essas novas formas de mapeamento surgiram de necessidades identificadas durante o processo de sincronização entre bases de dados reais. Nesse contexto, além do mapeamento clássico entre as bases de dados, são apresentadas 3 formas de mapeamento: transformação de dados, correspondência de dados e correspondência de dados SQL. A primeira forma de mapeamento trata da transformação entre tipos de dados. Já a segunda e a terceira forma de mapeamento são mais complexas e tratam do mapeamento de dados entre bases que apresentam relacionamento, ou seja, faz-se o mapeamento entre tabelas relacionadas de um lado com tabelas relacionadas do outro lado. Em qualquer um dos casos de mapeamento as bases podem estar geograficamente distribuídas.

Palavras-chave: banco de dados distribuídos, mapeamento, sincronização, SIEP, integração de dados.

1 INTRODUÇÃO

No artigo anterior apresentaram-se a arquitetura e o desenvolvimento de uma ferramenta denominada Sincronizador (da Silva ET. AL., 2008), cujo objetivo é manter atualizadas as diversas bases de dados de "sistemas-alvos" (i.e. os diversos sistemas computacionais já existentes no MEC que necessitam de atualizações de dados sobre as EPTs no Brasil).

O Sincronizador é parte integrante do projeto SIEP Gerencial onde é fornecida toda uma infra-estrutura de suporte a transferência e atualização de bases de dados distribuídas conforme mostra a Figura 1, onde são apresentados os cinco subsistemas de informação que compõem o SIEP Gerencial: o Atualizador, o Extrator, o Relator, o Seletor e o Sincronizador. Detalhes sobre o funcionamento de cada módulo podem ser vistos nos trabalhos de Aleixo (ET AL 2007) e da Silva (ET. AL 2008).

Especificamente, no artigo anterior foi apresentada a arquitetura do sincronizador e suas camadas de comunicação. Apresentou-se desde sua interface gráfica até a comunicação com os bancos de dados envolvidos. No trabalho também se ilustrou o diagrama de domínio e como cada entidade interage entre si.

Dentre as entidades consideradas, a responsável pelo mapeamento entre bases de dados é a que sofreu mais alterações, pois novos recursos foram identificados como essenciais durante os testes realizados com bases de dados reais. Muito embora a interface gráfica também tenha sido modificada para atender as exigências do MEC essa modificação não é o foco deste artigo, sendo o mesmo o processo de transferência de dados entre diferentes bases de dados (locais ou distribuídas).

Transferir dados de uma base para outro é também chamado de integração de dados, que pode chegar a consumir 40% dos recursos da tecnologia da informação. Além disso, a integração de dados representa um dos grandes desafios que a tecnologia da informação tem enfrentado (Brenstein & Haas, 2008). Nessa linha

de pesquisa, o trabalho de Potinger (Potinger & Breinstein, 2008) visa criar um esquema intermediário de banco de dados para realizar a integração entre dados. O trabalho de Zhang (Et AL, 2008), cria uma nova estrutura de dados baseado em XML para realizar a integração de dados. Já o trabalho de Fengguang (et al, 2009) desenvolve um framework baseado em XML para integrar dados. Embora a utilização de XML seja interessante na integração de dados, este trabalho não é baseado nela, muito embora os dados e as formas de integração possam ser exportadas para esse formato caso o usuário deseje.

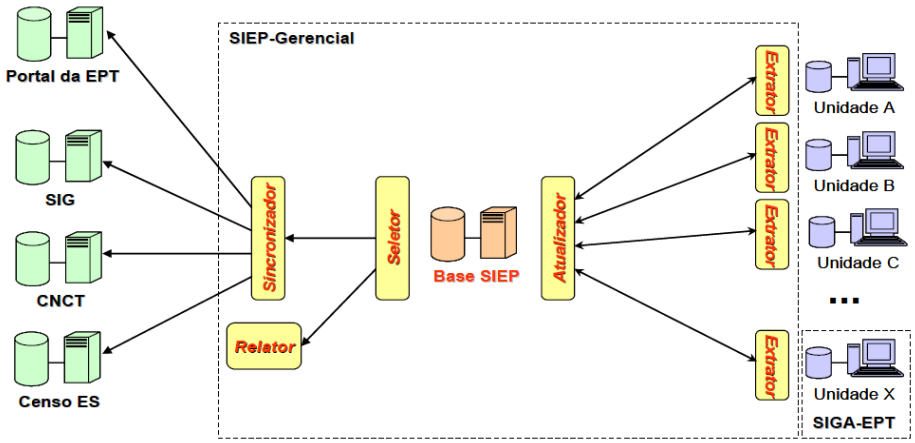

Figura 1 – Arquitetura do SIEP Gerencial

Três novas formas de mapeamento para prover mecanismos de integração de dados foram implementadas: transformação de dados, correspondência de dados e correspondência de dados SQL. Os mapeamentos tem como objetivo manter a consistência de dados após sua sincronização, isto é, manter a consistência após a transferência de dados entre as bases consideradas.

A primeira forma de mapeamento trata da transformação entre tipos de dados, nesse caso, é possível transferir dados do tipo *integer* (da base SIEP) para tabelas que usam dados do tipo *varchar* (da base do sistema alvo), por exemplo. Já a segunda e a terceira forma de mapeamento são mais complexas e tratam do mapeamento de dados entre bases que apresentam relacionamento, ou seja, faz-se o mapeamento entre tabelas relacionadas de um lado (base SIEP) com tabelas relacionadas do outro lado (base do sistema alvo).

Para cumprir seu objetivo, este trabalho esta dividido da seguinte forma: a Seção 2 indica as tecnologias utilizadas neste trabalho; A Seção 3 apresenta o Sincronizador e as modificações estruturais que nele foram realizadas; A Seção 4 trata dos mapeamentos implementados; finalmente, a Seção 5 mostra as conclusões deste trabalho.

2 TECNOLOGIAS

Para alcançar os objetivos desejados fez-se necessária uma plataforma de desenvolvimento confiável, robusta e escalável. Dentre as possibilidades, foi selecionada a plataforma *Java Enterprise Edition* – JEE – por se enquadrar melhor nas necessidades do sistema.

2.1 Java

Java é uma linguagem madura, idealizada pela Sun Microsystems. As suas principais características são: orientação a objetos, portabilidade, robustez, segurança e alta performance (DEITEL, 2007). A orientação a objetos é um paradigma de programação que traz diversas vantagens de projeto e implementação de programas modularizados, facilitando significativamente quase todas as etapas do ciclo de desenvolvimento.

A portabilidade é fundamental devido ao fato de o projeto ter de rodar em uma grande quantidade de ambientes diferentes, envolvendo inclusive diferença de sistemas operacionais. No entanto, o Java executa sobre uma máquina virtual – a *Java Virtual Machine* (JVM). Isto torna o Java independe do sistema operacional, precisando apenas da uma máquina virtual. Existem verificações de código em tempo de compilação e em tempo de execução, um sistema adequado de tratamento de exceções, um modelo de

gerenciamento automático de memória, entre outros fatores (LEWIS, 2007). A segurança é implementada no núcleo da plataforma, de maneira a evitar ataques de códigos maliciosos em ambientes isolados ou distribuídos. A alta performance provém de diversos fatores, que vão desde os algoritmos eficientes de execução do interpretador e da liberação automática de memória não utilizada, até a possibilidade de que determinados pontos críticos do código sejam compilados em linguagem nativa (SAVITCH, 2007).

2.2 Java Enterprise Edition 5

O JEE é a plataforma Java especifica para aplicações corporativas distribuídas (MCLAUGHLIN, 2002). O JEE é um superconjunto do JSE (*Java Standard Edition*), que possui os componentes básicos da linguagem Java. Os principais componentes da plataforma são os EJBs (*Enterprise JavaBeans*), JSP (*Java Server Pages*) /*Servlets* e JSF (*Java Server Faces*) (SAMPAIO, 2007). Neste trabalho será descrita apenas a tecnologia EJB.

Os EJBs são componentes Java que executam em um servidor de aplicação específico e permitem a criação rápida de sistemas distribuídos com suporte a transações, segurança, portabilidade, e todas as outras vantagens da linguagem (CRAWFORD, 2005). Um componente EJB nada mais é do que uma classe Java com anotações (*annotation*). O conceito de anotações surgiu com a versão 5 do Java com o objetivo de simplificar a construção de componentes. Uma anotação é equivalente a um comentário colocado no código-fonte gerado pelo programador. Mas, diferentemente de um comentário, que não tem efeito nenhum sob o código-fonte, a anotação implica na injeção de código de forma automática pelos compiladores e préprocessadores da linguagem.

Os EJBs executam em servidores de aplicação e podem ser acessados através da rede através do protocolo RMI-IIOP (MARK, 2001). Um servidor de aplicação é responsável por controlar o ciclo-de-vida dos EJBs e oferecer serviços, tais como: localização de recursos através de nomes e diretórios, autenticação e autorização de clientes, controle de transações, controle de balanceamento de carga, troca de mensagens assíncronas, entre outros (SULLINS, 2004).

2.3 Java Persistence API

Além dos componentes JEE, outro recurso do Java foi selecionado para ser utilizado no projeto: o JPA, ou Java Persistence API. JPA é um framework de mapeamento objeto-relacional introduzido oficialmente no mundo Java para tentar eliminar as diversas maneiras diferentes de se realizar tal mapeamento, definindo assim um padrão (KEITH, 2006). Funciona de maneira muito simples, bastando que o desenvolvedor defina classes devidamente anotadas que representam as tabelas do banco de dados relacional e um único e bastante simples arquivo XML de configuração. Definidas as entidades JPA, o desenvolvedor poderá utilizar a API do JPA para persistir estas classes de maneira orientada a objetos, tornando transparente todo o trabalho de conversão objeto-relacional por baixo disto (KEITH, 2006).

O projeto SIEP Gerencial buscou utilizar tais tecnologias por elas serem amplamente aceitas pelos desenvolvedores em todo o mundo, e por amparar-se em uma plataforma altamente confiável, como a plataforma Java.

2.4 SwingX

De acordo com o site do SwingX no Java.net3, o SwingX contém extensões para o *toolkit GUI Swing*, incluindo novos e aprimorados componentes que fornecem funcionalidades comumente requisitadas por aplicativos que usem clientes ricos. Destaques incluem": ordenamento, filtragem, destaques (*highlighting*) para tabelas, árvores (*trees*) e listas *Find/Search*, Auto-complemento, Framework para autenticação/login, Componente *TreeTable*, *CollapsiblePanel* (painel dobrável), Componente *DatePicker* (seletor de datas), Componente *Tip-of-the-Day* (dica-do-dia), *SwingX* é um subprojeto do *SwingLabs*, que conta com suporte da Sun Microsystems e com a participação de alguns dos mais importantes engenheiros da Sun responsáveis pelo Swing.

O SwingLabs serve para implementação e teste de idéias relacionadas à tecnologia voltada para o desenvolvimento de GUI's (*Graphical User Interface*) para clientes ricos. Essa biblioteca foi utilizada para o desenvolvimento da interface gráfica, com intuito de utilizar os painéis (*JXImagePanel*) que permitissem adicionar imagens. Foi utilizados também o componente *JXDatePicker* (calendário) para evitar que sejam entradas informações invalidas com validações na data inicial e final do agendamento.

2.5 Quartz

Para o desenvolvimento do módulo Agendador foi utilizado um framework denominado *Quartz*, que pode ser definido como um Agendador de tarefas. Portanto, o *Quartz* é um framework que tem como responsabilidade notificar outros componentes de software quando um pré-determinado horário ocorre. Nesse contexto, torna-se possível executar tarefas de forma agendada, sejam essas tarefas periódicas ou não.

2.6 SVN e Maven

Com o objetivo de tornar o desenvolvimento colaborativo mais eficiente lançou-se mão de duas tecnologias: o SVN e o *Maven*. SVN ou *SubVersion* é o nome de um sistema para permitir que muitas pessoas colaborem escrevendo textos (normalmente código de linguagens de programação). Os textos ficam armazenados em um repositório central, e os colaboradores podem fazer cópias de trabalho locais. Uma vez encerrada uma sessão de trabalho, um colaborador pode enviar sua cópia de trabalho de volta para o repositório. Se outras pessoas editaram os mesmos arquivos durante a sessão de trabalho, o SVN ajuda o colaborador a identificar e resolver eventuais conflitos. Já o *Maven* permite que sejam utilizadas diferentes IDEs no desenvolvimento da mesma aplicação, ou seja, o programador pode usar ou o Eclipse ou o *Netbeans*, dependendo da familiaridade que o programador possua com ela(s).

3 O SINCRONIZADOR

O Sincronizador foi desenvolvido em linguagem Java (ARNOLD et. al, 2000) na sua versão JEE (Java Enterprise Edition), que compreende um vasto conjunto de tecnologias que abrangem uma série de especificações voltadas principalmente para aplicações corporativas, contendo quase todos os pontos vitais para o desenvolvimento de aplicações de alta disponibilidade (MCLAUGHLIN, 2002;NICHOLAS, 2000; SAMPAIO, 2007). A Figura 2 mostra como funciona o Sincronizador.

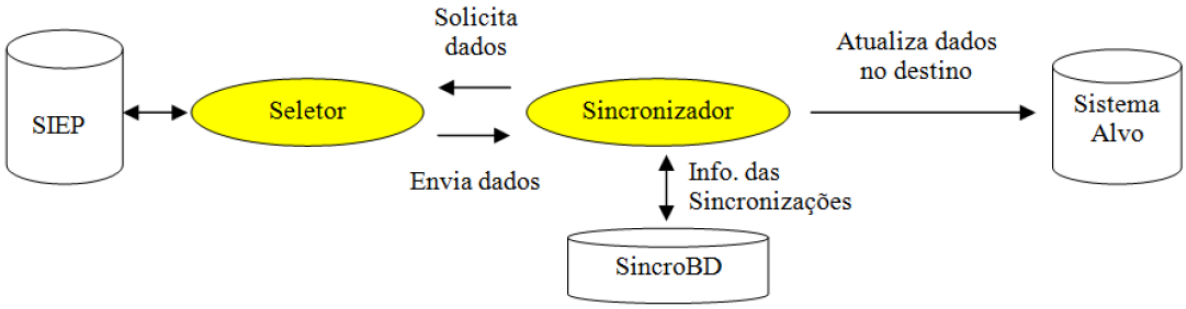

Figura 2 – Funcionamento do Sincronizador

A camada de acesso ao Seletor envia solicitações contendo consultas no formato da linguagem SQL ou XQL para a base central. A partir destas solicitações o Seletor responderá (de forma síncrona ou assíncrona) contendo as informações requisitadas em formato XML. A manipulação das informações coletadas a partir do Seletor será realizada através do *framework* Seletor (Aleixo ET. AL. 2007). Todo este ciclo de comunicação será controlado pelo *core* (núcleo) do sistema. É esta camada que controla os fluxos de informações entre a base SIEP e as bases dos Sistemas-alvo, implementando as regras de negócio do processo de sincronização. Este processo é configurado a partir de uma interface gráfica intuitiva, através da qual é possível gerar os mapeamentos, agendar e executar as sincronizações. Todos os dados necessários à execução das Sincronizações ficam armazenados como configurações de sincronização no banco denominado SicroDB.

O novo modelo de domínio é apresentado na Figura 3. Como já mencionado as principais modificações foram efetuadas no módulo de mapeamento. Entretanto, a descrição de cada módulo é feita para um melhor entendimento sobre o funcionamento do Sincronizador.

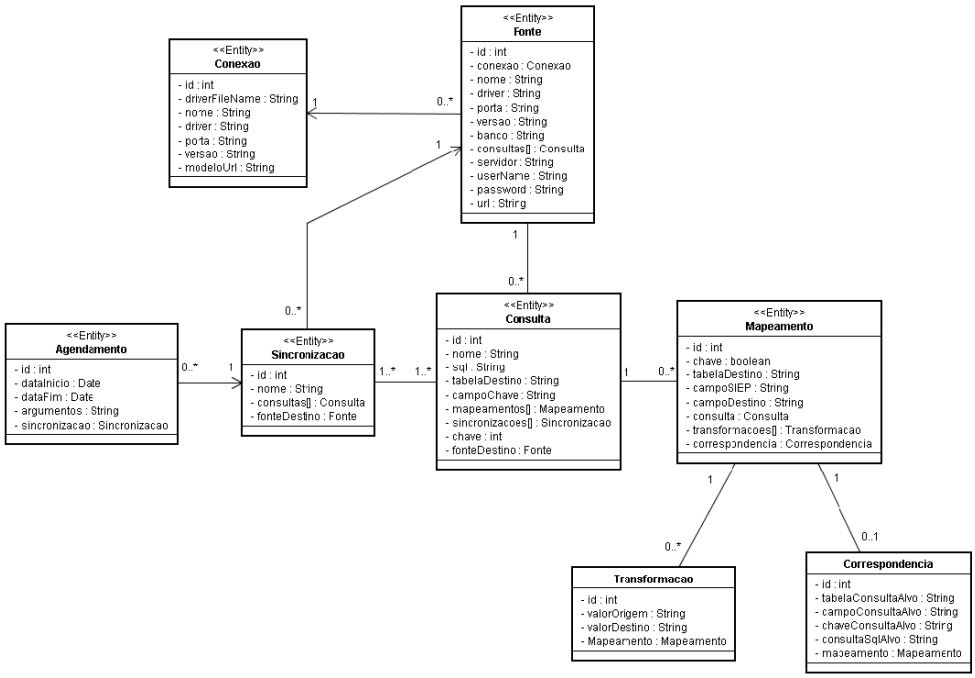

Figura 3 – Novo modelo de domínio

O módulo de Conexões (Conexão) permite que o usuário cadastre e gerencie diversos SGBDs que a partir de então poderão ser acessados pelo sistema, para isso é necessário apenas especificar um arquivo com as informações da conexão e o Driver JDBC para o SGBD. O módulo Fonte controla os sistemas alvo, permitindo que o usuário cadastre e gerencie os sistemas-alvo que poderão ser acessados pelo sincronizador, selecionando o SGBD que ele acessa e informando dados sobre a conexão. Os módulos que precisam de persistência utilizam a tecnologia JPA (KEITH, 2006; BURKE, 2006) para trabalhar com bancos de dados relacioanais. O ponto forte para utilização do JPA é que essa tecnologia gera as chamadas SQL e libera o desenvolvedor do trabalho manual da conversão dos dados resultante, mantendo o programa portável para quaisquer bancos de dados que utilize a linguagem SQL, sendo que no Sincronizador as informações persistentes são armazenadas, em um banco de dados PostgreSQL

O módulo de sincronizações (Sincronização) possui o cadastro e gerenciamento de sincronizações, que permite que o usuário adicione sincronizações para um sistema-alvo selecionado. Uma sincronização consiste de uma lista ordenada de mapeamentos, que são selecionados dentre os mapeamentos cadastrados e identificados por um nome (que é o nome da sincronização).

A execução das sincronizações consiste em executar o script SQL de cada mapeamento na base SIEP por meio do serviço seletor. Em seguida, inserem-se cada campo do resultado na base do sistema-alvo de acordo com as associações de tabela de destino e campo de destino, observando os valores dos campos definidos como chaves dos mapeamentos, executando em primeiro lugar um comando UPDATE. Caso o número de linhas afetadas seja igual a zero é executado então um comando INSERT, garantindo assim que os registros que já existem sejam atualizados e os que não existem sejam inseridos. É importante ressaltar que a ordem em que os dados são transferidos é a mesma em que os mapeamentos se encontram dentro da sincronização. Essa ordem é importante para que as tabelas que possuem chaves estrangeiras sejam sincronizadas somente após a sincronização das tabelas que elas referenciam, mantendo assim a integridade referencial.

O Agendamento permite que o processo de sincronização das bases de dados seja mais ágil, pois permite o agendamento de sincronizações executadas periodicamente. Este módulo é especificamente um cadastro de agendamentos, que torna o processo de sincronizações automatizadas, pois as informações são salvas na base de dados do próprio sincronizador. Este módulo utiliza a Quartz que permite a implementação e execução dos agendamentos sem complexidade.

Como já mencionado, o módulo de mapeamentos sofreu alterações consideráveis. Este módulo consiste basicamente no processamento de um script SQL de seleção (a ser executado na base do SIEP pelo serviço seletor) e um conjunto de associações entre campos das tabelas da base SIEP e campos das tabelas da base do sistema-alvo selecionado. Cada associação consiste no nome de um campo de uma tabela SIEP (campo de

origem), o nome de uma tabela da base do sistema-alvo (tabela de destino) e o nome de um campo dessa tabela (campo de destino). Para realizar o cadastro o usuário seleciona o sistema-alvo que deseja sincronizar, e em seguida, por meio de um script SQL, seleciona os dados que deseja extrair da base do SIEP. Os resultados então são visualizados e depois se associam os campos selecionados aos campos de alguma tabela do sistema alvo. Essa associação é feita graficamente, observando que para cada tabela do sistema alvo envolvida no mapeamento é necessário definir uma associação que exercerá o papel de chave primária do mapeamento naquela tabela.

Inicialmente, o sistema impedia que fosse realizada uma associação entre dois campos com tipos de dados incompatíveis. Com a nova estrutura esse problema foi resolvido, sendo possível mapear campos, por exemplo, inteiros para caractere e vice-versa. O mapeamento clássico e as novas formas avançadas de mapeamento são apresentadas na próxima seção.

4 MAPEAMENTOS

O mapeamento clássico, por assim dizer, faz o mapeamento entre a base de dados SIEP e um sistema alvo. Para criar um mapeamento é necessário criar inicialmente um consulta. Essa consulta nada mais é do que uma consulta SQL à base SIEP que irá retornar os dados que deverão ser inseridos ou atualizados no sistema alvo, sendo que em caso de erro o sincronizador avisa sobre o erro detectado.

Caso haja sucesso na consulta à base SIEP, o mapeamento entre a base SIEP e o sistema alvo deve ser feito associando-se os respectivos campos. Em seguida escolhe-se a chave primária do mapeamento. Esta chave é importante, pois a partir dela o Sincronizador identifica se um determinado registro deve ser atualizado ou inserido, ou seja, se no banco do sistema alvo já houver um registro de algum campo disponível na base SIEP,este campo sofrerá atualização dos dados, caso contrário este campo sofrerá inserção.

Com o mapeamento definido é possível realizar a sincronização entre as bases de dados. Destaca-se neste processo que o DBA deve conhecer bem o banco do sistema alvo e também definir a ordem com que as consultas serão sincronizadas. O ordenamento das consultas é importante para evitar problemas de quebra de integridade referencial tendo como exemplo em um relacionamento do tipo *1:n* ,onde o lado*1* deve ser sincronizado antes que o lado *n* para evitar que o registro do lado *n* seja inserido referenciando um registro no lado *1* que ainda não existe.

4.1 Mapeamentos Avançados

A primeira forma de mapeamento avançado é denominada ***transformação de dados*** e permite que dados de diferentes tipos sejam transferidos da base SIEP para o sistema alvo. Por exemplo, considere que na base de dados SIEP exista uma tabela chamada *alunos*; e essa tabela tenha um campo denominado *sexo*; e que este campo esteja definido como *Integer*, onde *1* representa o sexo masculino e *0* represente o sexo feminino. Além disso, considere que no sistema alvo exista uma tabela chamada *tb_aluno*; e que esta tabela também possua um campo chamado *sexo*, porém o campo está definido como *char* onde 'M' representa o sexo masculino e 'F' representa o sexo feminino. Essa transformação agora é possível através do módulo Transformação, onde todos os valores são convertidos em *String* e transformados no tipo correspondente no momento da inserção/atualização no sistema alvo. A Figura 4a mostra a interface gráfica onde a transformação de dados é realizada. Nela pode se observar: a consulta realizada na base SIEP, os campos de origem e destino e os tipos nas respectivas bases de dados.

A segunda forma de mapeamento avançado é mais complexa do que a primeira e envolve a utilização de chaves estrangeiras, sendo chamado de ***correspondência de dados***. O problema ocorre quando se faz necessário levar dados relacionados da base SIEP para um sistema alvo que também possui informações relacionadas. Esse processo cria um novo problema ao processo de sincronização: a manutenção dos relacionamentos no sistema alvo destino depende do conhecimento prévio do valor da chave primária do relacionamento (que é requerida pelo campo de chave estrangeira). No entanto, durante o processamento da sincronização, o valor da chave primária é desconhecido, já que cada registro é recriado no sistema alvo com novos valores de chave. Isso cria a demanda por um novo mecanismo, por meio do qual o sincronizador necessita consultar a própria base alvo local.

(a) Transformação de Dados (b) Correspondência de Dados

Figura 4 – Transformação de dados

Por exemplo, considerando as tabelas *Instituição(id_instituicao, CNPJ, nome_instituicao)* e *Campus(id_campus, fk_id_instituicao, nome_campus)* na base SIEP, onde ambas se relacionam através dos campos *id_instituicao* e *fk_id_instituicao*. É possível perceber que no sistema alvo, para manutenção desse relacionamento, duas prerrogativas necessitam ser consideradas: (a) os registros da tabela *Instituição* precisam ser inseridos ANTES dos registros da tabela *Campus*; e (b) ao inserir os registros dos campi na base alvo, é necessário conhecer o valor do campo *id_instituicao* gerado pelo próprio banco local e manter seu relacionamento consistente. Nesse caso, um campo que pode ser utilizado para identificação única das instituições é o seu CNPJ. Logo, considerando que as instituições já foram inseridas (prerrogativa "a"), basta-se realizar uma consulta a esta tabela (local) com base no relacionamento obtido a partir dos registros das tabelas SIEP (remotos) (prerrogativa "b"). O sincronizador então precisa executar a seguinte consulta:

```
SELECT ID_INSTITUICAO FROM INSTITUICAO

WHERE CNPJ = <CNPJ MAPEADO AO BANCO SIEP REMOTO>
```

A partir dessa consulta é possível obter o campo *id_instituicao* e então armazená-lo no campo de destino (no exemplo, *fk_id_instituicao*). A Figura 4b ilustra uma tela com um outro exemplo de mapeamento desse tipo sendo realizado. A montagem da consulta (de acordo com os campos mapeados na figura) é da seguinte forma:

```
SELECT CAMPO_ORIGEM FROM TABELA_ORIGEM

WHERE CHAVE_ORIGEM = <CAMPO_MAPEADO_NA_ORIGEM>
```

A partir do valor do campo de origem é possível realizar a inserção do registro relacionado, com o valor do seu campo `TABELA_DESTINO.CAMPO_DESTINO` igual ao valor obtido no `CAMPO_ORIGEM` (no exemplo, *fk_id_instituicao* = *id_instituicao*).

Outra possibilidade é a utilização da chamada ***correspondência de dados SQL***, onde ao invés de se indicar os dados de origem como feito no exemplo anterior, o DBA pode digitar um comando SQL diretamente em uma caixa de texto indicando os relacionamentos, eliminando assim o processo de escolha através de *comboboxes*. Na verdade o comando SQL será um Inner Join com as tabelas relacionadas.

5 CONCLUSÕES

Este trabalho apresentou a novas formas de mapeamento de dados que permitem efetuar sincronizações entre bases de dados, mantendo a consistência e a integridade entre as informações. Além do mapeamento clássico forma apresentadas três formas de mapeamento avançado: transformação de dados, correspondência de dados e correspondência de dados SQL. A implementação dos mapeamentos considerados não foi uma tarefa

trivial, especialmente dos mapeamentos por correspondências, pois os mesmos envolvem chaves primárias e chaves estrangeiras

A utilização do sincronizador em ambientes reais tem mostrado que os mapeamentos implementados têm permitido a inclusão e/ou atualização de dados nos sistemas alvos sem gerar problemas de inconsistência de dados, mostrando assim sua aplicabilidade prática. Por outro lado, este trabalho apresenta uma contribuição tecnológica na área de banco de dados dsitribuídos.

REFERÊNCIAS

Bernstein, P.; Haas, L. M., Information Itegration in the Enterprise, Communications of the ACM, v. 56, n.9, 2008

ALEIXO, Fellipe Araújo; AZEVEDO da Silva, Gilbert; dos SANTOS, Adriano Bezerra; RODRIGUES, Thyago Barbosa; do CARMO, Michelle Furtado Pinheiro, **SIEP GERENCIAL: SISTEMA DE CENTRALIZAÇÃO DAS INFORMAÇÕES DA EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA**, II Congresso de Pesquisa e Inovação da Rede Norte Nordeste de Educação Tecnológica, João Pessoa – PB, 2007

Da SILVA, J. A.; MARTINS, M. R. M.; LOPES, R. F.; CORTES, O. A. C.; PINHEIRO, D. N., **Sincronizando Bases de Dados em Ambientes Distribuídos: Um Estudo de Caso no Projeto Siep Gerencial**, III Congresso de Pesquisa e Inovação da Rede Norte Nordeste de Educação Tecnológica, Fortaleza-CE, 2008.

Fengguang, X.; Xie, H.; Liqun, K., Research and implementation of heterogeneous data integration based on XML, In: 9th International Conference on Electronic Measurement & Instruments, 2009.

SAMPAIO, Cleuton. **Guia do Java Enterprise Edition** 5. Brasport, 2007

MCLAUGHLIN, Brett. **Building Java Enterprise Applications**. O'Reilly & Associates, 2002.

KEITH, Michael; HALEY, Jason; SCHINCARIOL, Merrick. **PRO EJB 3 – Java Persistence API**. Apress, 2006.

DEITEL, Harvey M. **Java – Como Programar**. Prentice Hall Brasil, 2007.

LEWIS John; CHASE, Joe; DEPASQUALE, Peter. **Java Foundations**. Addison Wesley, 2007.

Pottinger, R.; Bernstein, P. A., Schema merging and mapping creation for relational sources. EDBT, p 73-84, 2008.

SAVITCH, Walter; MOCK, Kenrick. **Absolute Java**. Addison Wesley, 2007.

CRAWFORD, William; FARLEY, Jim. Java Enterprise in a Nutshell. O'Reilly & Associates, 2005.

MARK, Grand. **Java Enterprise Design Patterns**. John Wiley Consumer, 2001.

SULLINS, Benjamin G.; WHIPPLE, Mark B. **EJB – Livro de Receitas**. Ciência Moderna, 2004.

Zhang, Z.; Song, M.; Liu, D.; Wei, Z.; Wang, H.; Ni, J., Data-Structure-Model for Data Integration in Distributed Systems, In: International Multisymposiums on Computer and Computational Sciences, 2008