UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE SISTEMAS DE INFORMAÇÃO

PROPOSIÇÃO DE UMA ARQUITETURA DE SINCRONIZAÇÃO DE FONTES HETEROGÊNEAS DE DADOS UTILIZANDO FRAMEWORKS DE MAPEAMENTO DE DADOS

MICHAEL SOARES MOLINA

FLORIANÓPOLIS JUNHO DE 2008

UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE SISTEMAS DE INFORMAÇÃO

PROPOSIÇÃO DE UMA ARQUITETURA DE SINCRONIZAÇÃO DE FONTES HETEROGÊNEAS DE DADOS UTILIZANDO FRAMEWORKS DE MAPEAMENTO DE DADOS

BANCA EXAMINADORA:

_____

Prof. Dr. José Leomar Todesco

Orientador

_____ _____

Prof. Dr. Roberto Pacheco **Mestre Marcelo Domingos**

MICHAEL SOARES MOLINA

FLORIANÓPOLIS JUNHO DE 2008

RESUMO

O compartilhamento e manutenção de informações heterogêneas no contexto empresarial demandam por mecanismos de integração de dados que suportem a velocidade de propagação e aumento do volume da informação. Para atender a essa demanda este artigo propõe o uso de uma lógica de sincronização de dados baseada em descrições semânticas e frameworks de mapeamento de dados. O resultado é uma arquitetura que simplifica a integração de dados de fontes heterogêneas e minimiza o custo da sincronização. Isto visa garantir maior interoperabilidade entre aplicações e diminuir o tempo e complexidade de desenvolvimento de soluções de sincronização de dados.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização do problema

A evolução da internet juntamente com o aumento do acesso à informação tem diminuído a distância entre os povos, proporcionado maior comunicação entre diferentes culturas, países e sistemas. Padrões de interação humano-máquina emergem da utilização do computador em diferentes áreas e contextos para diferentes objetivos. Usuários cada vez mais exigentes e globalizados necessitam de sistemas que se comuniquem entre si de forma conveniente, padronizada e inteligente. Os sistemas distribuídos vêm crescendo consideravelmente nos últimos anos com o intuito de assegurar esses e outros objetivos, e estão sendo implantados em diferentes lugares, com diferentes tecnologias. Mas essa conveniência exige uma infra-estrutura que suporte essa heterogeneidade tecnológica, de localização e de preferências. Os dados dos usuários necessitam estar acessíveis em qualquer lugar do mundo, a qualquer hora, de maneira segura, consistente e de fácil acesso.

O compartilhamento de informações entre aplicações, cada vez mais freqüente na economia atual, tem enfatizado a necessidade de soluções de integração de dados (Alon, Anand *et al.*, 2006) .

Esta integração desempenha papel fundamental no processo de disponibilização do acesso à informação. Sistemas de sincronização de dados são mecanismos fundamentais, pois envolvem diretamente as unidades mais importantes do mundo atual, dados e informação.

Já no contexto empresarial, a manutenção da informação gera demanda por soluções tecnológicas que garantam a integridade e significado dos dados de maneira rápida, precisa e transparente ao usuário.

Um mar de informação não diferenciada se forma a partir de dados coletados por pessoas, organizações e governos, para diferentes propósitos, em diferentes tempos, e usando diferentes metodologias. A heterogeneidade massiva de dados resultante requer métodos automáticos de alinhamento, correspondência e unificação destes dados (Patrick, Andrew *et al.*, 2006).

Neste contexto, o grande número de sistemas legados, diferentes abstrações sobre um mesmo conceito e necessidades específicas de cada setor empresarial, contribuem para a proliferação de fontes heterogêneas de dados. A troca de informação entre essas fontes tornou-se um desafio a ser enfrentado por soluções de integração devido ao maior tempo e complexidade de desenvolvimento (Alon, Anand *et al*., 2006).

Em (Wilhelm, 2000), é ressaltada a importância da coexistência entre aplicações novas e legadas, uma vez que geralmente não há tempo nem justificativa para substituição desses sistemas. Ferramentas de migração de informações, em um contexto dinâmico de mudanças tecnológicas e de negócio, visam à proteção de investimentos existentes e possibilitam mudanças de requisitos. Porém, o desenvolvimento deste tipo de ferramenta é geralmente complexo, o que atualmente tem gerado um grande esforço na área de mapeamento de dados para facilitar e automatizar o processo de acesso e persistência de informações. Frameworks como JPA, Hibernate, JAXB e JIBX estão sendo amplamente utilizados pelo mercado com o intuito de mapear as informações de negócio em estruturas de fácil manipulação diminuindo o custo do desenvolvimento.

Nesta linha, este trabalho apresenta uma lógica de sincronização de dados, baseada em descrições semânticas e frameworks de mapeamento de dados, que potencializa uma arquitetura de software para simplificar a integração de dados de fontes heterogêneas e minimizar o custo de sincronização de informações. Esta arquitetura apresenta-se como uma alternativa para suprir as necessidades de migração de dados entre fontes heterogêneas levantadas anteriormente.

1.2 Motivação

A tendência é de que em um futuro muito próximo teremos aplicações de diferentes naturezas, executando em diferentes dispositivos, comunicando-se e trocando dados de maneira transparente para o usuário.

Considerando a heterogeneidade dessas aplicações, diferentes ambientes de execução e distribuições de bancos de dados, e pelo fato das aplicações web estarem diretamente inseridas nesse contexto, na medida em que atendem a um público alvo variado, é necessário um processo de sincronização de dados como mecanismo para garantir

sistemas inter-operáveis que possibilitem o acesso à informação de maneira rápida, precisa, segura e confiável.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo geral é apresentar uma lógica de sincronização de dados, baseada em descrições semânticas e frameworks de mapeamento de dados.

1.3.2 Objetivo Específico

Os objetivos específicos deste trabalho são:

- Minimizar o tempo de sincronização;
- Aumentar a precisão da sincronização;
- Demonstrar a aplicação da lógica de sincronização em estudos de caso reais onde os problemas de heterogeneidade de fontes de dados ocorrem com freqüência.

1.4 Justificativa

Este estudo justifica-se em função das novas tendências e paradigmas de programação em prol da interoperabilidade entre sistemas distribuídos, juntamente com o fato do crescimento do acesso à informação, e da importância de um sistema de sincronização para suportar estes processos.

Esta pesquisa justifica-se também para elaboração de uma solução tecnológica capaz de lidar com requisitos de compartilhamento de informações comumente encontrados em ambientes heterogêneos como os dois estudos de caso apresentados neste estudo. Particularmente o sistema de o sistema de sincronização dos Currículos Lattes com a base de dados do Portal Inovação é um sistema com grande volume de informação e taxa de atualização diária de aproximadamente 7 mil currículos.

O texto destaca primeiro alguns conceitos relacionados ao problema da sincronização de dados como a heterogeneidade, a entropia da informação, os algoritmos de diferenciação e os frameworks de mapeamento. Logo após, são apresentadas a lógica de sincronização e o processo de mapeamento das fontes de informação para um modelo compreensível ao algoritmo de sincronização. Por fim, são discutidas as implicações da arquitetura gerada, os possíveis cenários para os quais a mesma é recomendada e trabalhos futuros a serem desenvolvidos.

2 INTEGRAÇÃO DE DADOS

2.1 Introdução

O crescimento de organizações invariavelmente leva à criação de múltiplas fontes de dados isoladas, que são totalmente desconectadas umas das outras. Isto leva a menor eficiência e falta de conhecimento completo sobre a empresa e seus clientes durante a tomada de decisões críticas de negócio. Este é o clássico problema de Integração de Informação que se tornou o maior ponto de dor para as empresas atuais (Mukesh e Manish, 2008).

A Integração de Informação refere-se à categoria de middleware que permite aplicações acessarem dados como se estivessem em um único banco de dados. Ela permite a integração de dados e conteúdo e, a transformação e compartilhamento de dados para análise de negócios. Integrar dados é combinar dados residindo em fontes diferentes e prover ao usuário uma visão unificada dos mesmos. O problema da integração surge principalmente devido aos ambientes complexos e heterogêneos nas empresas.

Existe uma tendência crescente em considerar os dados de uma organização como um recurso autônomo, independente das funções atualmente em uso na organização. Assim como, existe a necessidade de capturar-se o significado dos dados para toda a organização, a fim de geri-la de forma eficaz. Devido a esta consciência, a integração de dados tornou-se uma área de crescente interesse nos últimos anos, e evoluiu principalmente devido a necessidade de integração de bancos de dados. Um dos princípios fundamentais da abordagem de banco de dados é que os bancos permitem uma representação unificada e não redundante de todos os dados geridos em uma organização. Isto é válido somente quando estão disponíveis metodologias para suportar a integração nos limites organizacional e entre aplicações (Batini, Lenzerini *et al.*, 1986).

A integração de banco de dados é um problema relativamente recente, que tem surgido no contexto das bases de dados distribuídas. Um banco de dados distribuído é uma coleção de dados que pertencem logicamente ao mesmo sistema, mas estão espalhados ao longo de sites de uma rede de computadores. Bancos de dados distribuídos e sistemas gerenciadores de bancos de dados distribuídos podem ser classificados em

duas categorias principais: homogêneos, lidando com bancos de dados locais que possuem o mesmo modelo de dados e SGBDs idênticos, e heterogêneos, com diversidade de modelo de dados e SGBDs (Batini, Lenzerini *et al.*, 1986).

Outro fator que contribuiu muito para a evolução da integração de dados foi o papel do XML como forma de compartilhamento de dados, ou seja, o XML impulsionou o desejo pela integração de dados, porque ofereceu um formato sintático comum para o compartilhamento entre as fontes de dados. No entanto, ele não fez nada para resolver as questões semânticas da integração - fontes ainda podiam compartilhar arquivos XML cujas tags não possuíam nenhum sentido fora da aplicação. Porém, uma vez que ele surgiu como alternativa ao compartilhamento de informações, o impulso para a integração tornou-se muito mais significativo. A partir da perspectiva técnica, vários sistemas de integração foram desenvolvidos usando XML como modelo de dados e linguagens de consulta XML como forma de acesso a esses dados. Para suportar esses sistemas, cada aspecto dos sistemas de integração de dados precisou ser estendido para lidar com XML. Os principais desafios foram tipicamente lidar com aspecto hierárquico do XML e o fato que era semi-estruturado (Alon, Anand *et al.*, 2006).

Atualmente novas tecnologias de rede, bancos de dados distribuídos, sistemas baseados em conhecimento, e sistemas de escritório tendem a difundir o uso compartilhado de dados em termos de número de usuários, diversidade de aplicações, e sofisticação dos conceitos. Espera-se que metodologias para integrar dados em suas diversas formas conceituais e físicas cresçam substancialmente nos próximos anos. Com isso aumentam as pesquisas em torno do processo de integração de dados, nos mais variados ambientes, com o objetivo de detalhamento das etapas do processo. As pesquisas visam obter um modelo comum, ou padronizado, a partir do qual possam evoluir novas formas e mecanismos de integração que sejam capazes de suportar o crescimento e diferenças de representação de fontes de dados organizacionais. A seguir é apresentado o processo de preparação do schema integrado que é um modelo comum de representação entre as fontes de dados e serve como base para a arquitetura a ser utilizada na integração dos dados.

2.2 Processo de preparação do schema integrado

O processo de definição do schema integrado objetiva conceber um modelo comum de representação das informações a serem integradas, relativas ao negócio da organização. Este processo divide-se em etapas comumente encontradas em processos deste gênero, que podem ser analisadas, ou mesmo executadas por diferentes pessoas e ferramentas, ou mesmo por diferentes setores dentro de uma organização. As etapas são as seguintes:

2.2.1 Pré-integração

Uma análise dos schemas é realizada antes da integração para decidir sobre certa política de integração. Esta política é usada, por exemplo, para relacionar nomes, estabelecer que um objeto de um schema é o resultado de alguma operação sobre objetos de outro schema, etc. Ela rege a escolha dos schemas a serem integrados, a ordem de integração, e de uma eventual atribuição de preferências para todo ou parte dos schemas (Batini, Lenzerini *et al*., 1986).

Para todas as metodologias, sendo a etapa de pré-integração mencionada explicitamente ou não, devem ser considerados a seqüência e agrupamento dos schemas para o processo de integração.

2.2.2 Comparação dos schemas

A atividade fundamental nesta etapa é a analise e comparação das representações dos objetos em diferentes schemas para determinar as correspondências entre os conceitos e detectar possíveis conflitos. As metodologias distinguem amplamente dois tipos de conflitos: de nome e estruturais (Batini, Lenzerini *et al*., 1986).

Conflitos de nome

Schemas em modelos de dados incorporam nomes para os vários objetos representados. Pessoas de diferentes áreas de aplicação da mesma organização referem-se aos mesmos dados usando sua própria terminologia e nomes. Isso resulta em uma proliferação de nomes, assim como uma possível inconsistência entre os nomes nos schemas componentes. As relações problemáticas entre nomes são de dois tipos:

- Homônimos: Quando o mesmo nome é usado para dois conceitos diferentes, dando origem a inconsistências menos detectadas.

- Sinônimos: Quando o mesmo conceito é descrito por dois ou mais nomes. A menos que diferentes nomes melhorem a compreensão de diferentes usuários, eles não são justificados.

Dicionários de dados têm sido defendidos entre as metodologias como uma ferramenta útil para melhor gestão dos nomes.

Conflitos estruturais

O termo conflitos estruturais é utilizado para incluir conflitos que surgem como resultado de uma escolha diferente de modelagem de estruturas e restrições de integridade (Batini, Lenzerini *et al.*, 1986).
Abaixo é apresentada uma classificação dos conflitos estruturais independente das várias terminologias e de características específicas dos diferentes modelos de dados nas metodologias adotadas.

- Conflitos de tipo. Estes surgem quando o mesmo conceito é representado por diferentes estruturas em diferentes schemas. Este é o caso quando, por exemplo, uma classe de objetos é representada como uma entidade em um schema e como um atributo em outro.

- Conflitos de dependência. Estes surgem quando um grupo de conceitos estão relacionados entre si mas com diferentes dependências em diferentes schemas.

- Conflitos de chave. Diferentes chaves são atribuídas ao mesmo conceito em diferentes schemas.

- Conflitos de comportamento. Estes surgem quando diferentes políticas de inserção/exclusão estão associadas com a mesma classe de objetos em schemas distintos. Por exemplo, em um schema um departamento pode estar autorizado a existir sem empregados, enquanto em outro, eliminar o último empregado associado com um departamento leva à exclusão do próprio departamento.

2.2.3 Ajuste dos schemas

Depois que os conflitos são detectados, um esforço é feito para resolvê-los de modo que a fusão de diversos schemas seja possível. Para isso transformações no schema são necessárias. A resolução automática dos conflitos, geralmente não é viável; por isso é necessária estreita interação com projetistas e usuários para melhor entendimento semântico por parte do projetista antes que objetivos possam ser alcançados em qualquer atividade de integração na vida real (Batini, Lenzerini *et al.*, 1986).

2.2.4 Fusão e reestruturação

Agora, os schemas estão prontos para serem sobrepostos, dando origem a algum schema integrado intermediário. Os resultados intermediários são analisados e, se necessário, reestruturados a fim de atingir várias qualidades desejáveis. Um schema conceitual global pode ser testado contra os seguintes critérios qualitativos (Batini, Lenzerini *et al.*, 1986):

- Completude e corretude. O schema integrado deve conter de maneira correta todos os conceitos presentes em qualquer schema componente. O schema integrado deve ser uma representação da união dos domínios da aplicação associados com os schemas.

- Minimalização. Se o mesmo conceito é representado em mais de um schema componente, ele deve ser representado apenas uma vez no schema integrado.

- Compreensibilidade. O schema integrado deve ser de fácil compreensão para o projetista e para o usuário final. Isto implica que entre as várias possíveis representações dos resultados da integração permitidas por um modelo de dados, a que for qualitativamente mais compreensível deve ser escolhida.

De posse do schema integrado e da validação das qualidades desejáveis do mesmo, pode-se partir para a estruturação da arquitetura de integração a ser utilizada durante um processo de integração de dados organizacionais. Porém, essa arquitetura deve ser capaz

de abstrair inúmeros problemas de heterogeneidade comumente encontrados em fontes de dados organizacionais. Devido a sua alta complexidade de resolução, é um problema que recebe tratamento especial dentro do universo da integração de dados. A seguir este problema é abordado em mais detalhes.

2.3 O problema da heterogeneidade

A heterogeneidade advém da concepção de diferentes representações de objetos do mundo real durante a especificação dos dados. Quatro motivos levam a esta diferença de representação (Stefano, Christine *et al.*, 1992):

1. Os dois projetistas não percebem exatamente o mesmo conjunto de objetos do mundo real, mas, ao invés visualizam a sobreposição de conjuntos (inclusão ou interseção de conjuntos). Por exemplo, um objeto *Gerente* pode aparecer em um schema, enquanto que um objeto *GerenteVendas*, mais restritivo, aparece em um outro schema. Esta é a primeira espécie de conflitos, chamada conflitos semânticos. O conceito de generalização tem sido amplamente utilizado como uma solução em casos de conflitos semânticos. Por exemplo, a classe *GerenteVendas* é integrada como uma subclasse de *Gerente*.

2. Ao descrever conjuntos relacionados de objetos do mundo real, dois projetistas não perceber exatamente o mesmo conjunto de propriedades. Por exemplo, vamos supor dois schemas relacionais, S1 e S2, descrevendo o mesmo conjunto de modelos de automóveis caros:

   S1: Carro_Caro (nome_do_modelo, fabricante, velocidade_maxima, preco)
   S2: Carro (nome, cavalos, consumo_de_combustivel, preco)
   Projetistas de S1 e S2 gravaram itens diferentes, devido ao seu interesse em diferentes itens disponíveis a respeito de modelos de automóveis no mundo real (um projetista pode ter de manter dados de anúncios em uma revista de artes plásticas, enquanto que o outro está preocupado com anúncios em uma revista técnica, por exemplo). Esta segunda espécie de conflitos é chamada de conflitos descritivos. Conflitos descritivos incluem homônimos e sinônimos, domínio de atributo, escala, restrições, operações, etc.

3. Os projetistas usam diferentes modelos de dados, por exemplo, um relacional e um orientado a objetos. Isto se chama conflito de heterogeneidade.

4. Por último, mesmo se eles usam o mesmo modelo de dados, eles podem escolher diferentes estruturas para representar objetos do mundo real. Por exemplo, nos modelos orientados a objetos quando um projetista descreve um componente de um tipo de objeto O, ele tem de escolher entre a criação de um novo tipo de objeto ou adicionando um atributo em O. Esse tipo de conflito é chamado de conflito estrutural. A medida em que podem ocorrer conflitos estruturais está relacionada com o relativismo semântico do modelo de dados em uso, ou seja, a sua capacidade de suportar diferentes, embora equivalentes, representações de uma mesma realidade. Modelos semânticos e orientado a objetos têm mais relativismo semântico do que o modelo relacional. Por isso, a capacidade de resolver conflitos estruturais é de importância cada vez maior para metodologias de integração.

Dentre os motivos citados, freqüentemente a heterogeneidade semântica aparece com maior destaque na literatura. Isso porque é um problema que requer completo conhecimento do domínio do negócio a ser integrado e exige a interação humana na resolução dos conflitos.

A heterogeneidade semântica refere-se a diferenças ou semelhanças com o significado de dados locais. Por exemplo, dois elementos do schema em duas fontes de dados locais podem ter o mesmo significado pretendido, mas nomes diferentes. Assim, durante a integração, deve-se perceber que esses dois elementos se referem ao mesmo conceito. Alternativamente, dois elementos do schema em duas fontes de dados poderiam ser chamados de maneira idêntica, enquanto que seus significados pretendidos são incompatíveis. Assim, estes elementos devem ser tratados como coisas diferentes durante a integração (Farshad e Andreas, 2001).

Um segundo aspecto essencial da heterogeneidade semântica é reconciliar os dados a nível de instância. Em qualquer aplicação de integração de dados vemos casos em que o mesmo objeto é referenciado no mundo de maneiras diferentes (por exemplo, as pessoas, endereços, nomes de empresa, genes). O problema da reconciliação de

referência é detectar automaticamente as referências ao mesmo objeto e unificá-las. Ao contrário de reconciliar a heterogeneidade de schema, na reconciliação de instâncias as quantidades de dados normalmente são muito maiores. Por conseqüência, os sistemas necessitam recorrer a métodos que são na sua maioria automáticos (Alon, Anand *et al.*, 2006).

Apoiar-se no senso comum também é uma fonte crítica de heterogeneidade semântica e a definição explícita dos termos utilizados na definição do schema é uma solução para este problema. A definição explícita e formal da semântica dos termos guiou muitos pesquisadores a aplicar ontologias como uma potencial solução para a heterogeneidade semântica.

Como conseqüência destes aspectos, uma integração de dados adequada e significativa é baseada na detecção de discrepâncias e semelhanças entre elementos do negócio. Assim, a semântica dos dados deve ser levada em conta durante a integração. Semântica é a interpretação que pessoas atribuem aos dados de acordo com a sua compreensão do mundo. Diferentes interpretações dos dados causam heterogeneidade semântica (Farshad e Andreas, 2001).

Uma vez que fontes são, em geral, autônomas, em muitas aplicações do mundo real existe o problema de fontes de dados inconsistentes entre si. Na prática, este problema é geralmente tratado por meio de técnicas de transformação e limpeza aplicadas aos dados extraídos das fontes.

Outro problema comumente encontrado em ambientes heterogêneos é a definição de um modelo comum de abstração entre as fontes de dados participantes do processo de integração e o mapeamento deste modelo para cada fonte específica. Este problema é discutido em detalhes a seguir.

2.4 Mapeamento entre fontes e o schema

Um dos aspectos mais importantes na concepção de um sistema de integração de dados é a especificação da correspondência entre os dados das fontes e os do schema global. Essa correspondência é modelada através do conceito de mapeamento. É exatamente

essa correspondência, que irá determinar a forma como as buscas colocadas para o sistema serão respondidas.

Escrever tais mapeamentos e mantê-los requer conhecimento em bancos de dados, para expressá-los de maneira formal, e do negócio, para compreensão do significado do schema sendo mapeado. Assim, uma significativa parte da comunidade de pesquisa focou-se na geração semi-automática dos mapeamentos do schema. Em geral, o mapeamento automático do schema é um problema de inteligência artificial completo, por isso o objetivo desses esforços foi a criação de ferramentas que aceleram a criação dos mapeamentos e reduzem a quantidade de esforço humano envolvido (Alon, Anand *et al.*, 2006).

Em primeiro lugar, a pesquisa explorou técnicas para realizar o mapeamento entre schemas com base em pistas que podem ser obtidas a partir dos próprios schemas, como semelhanças lingüísticas entre elementos do schema e sobreposições nos valores de dados ou tipos de dados de colunas. Em segundo lugar, com base na observação de que nenhuma das técnicas acima é infalível, o próximo desenvolvimento envolveu sistemas que combinavam um conjunto de técnicas individuais para criar mapeamentos. Por último, uma das principais observações foi que o mapeamento de schemas é uma tarefa geralmente repetitiva. Por exemplo, na integração de dados nós mapeamos múltiplos schemas do mesmo domínio para o mesmo schema mediador. Por isso, podemos usar técnicas de aprendizagem de máquinas que considerem os mapeamentos de schema criados manualmente como dados de treinamento, e generalize a partir deles para deduzir mapeamentos desconhecidos entre schemas (Alon, Anand *et al.*, 2006).

Apresentadas a motivação e os problemas da heterogeneidade e do mapeamento entre fontes de dados e o schema global, este trabalho explora agora as técnicas e abordagens propostas para solucionar estes problemas e as tendências no ramo da integração de dados.

2.5 Técnicas e abordagens para o problema da integração

A integração de dados tem recebido grande atenção desde os primeiros dias dos bancos de dados. Nos últimos anos, houve vários projetos com foco em integração de

informações heterogêneas. A maior parte deles é baseada em uma arquitetura mediadora comum. Nesta arquitetura, os mediadores provêm para o usuário uma interface uniforme na qual ele pode executar consultas sobre visões estruturadas em fontes de dados heterogêneas. As consultas executadas sobre conceitos globais são distribuídas pelos mediadores em sub-consultas sobre as fontes de dados (Fahad, Gray *et al.*, 2006).

A maioria das publicações de integração de visões tenta estabelecer uma técnica semi-automática para geração de um schema integrado a partir de um conjunto de afirmações, relacionando objetos correspondentes entre as visões.

Pelo contrário, as metodologias de integração de bancos de dados, objetivam proporcionar uma ferramenta ao DBA que possibilite a construção do schema integrado como uma visão sobre os schemas iniciais (Stefano, Christine *et al.*, 1992).

Duas arquiteturas básicas foram propostas para fazer integração de informações - virtualização e materialização. Na abordagem de virtualização os dados residem em fontes de dados individuais. Uma camada de virtualização é definida como um schema virtual que tem os atributos de todas as fontes de dados. Quando uma consulta definida por usuário sobre o schema virtual é recebida pelo sistema, ele determina as fontes relevantes para serem consultadas e, em seguida, quebra a consulta em sub-consultas para as diferentes fontes. As sub-consultas são realizadas sobre as fontes de dados apropriadas e as respostas recebidas a partir das fontes de dados são combinadas adequadamente e devolvidas de volta para o usuário (Mukesh e Manish, 2008).

A vantagem desta abordagem é que os dados retornados de volta para o usuário são sempre atuais. No entanto, o grande desafio é definir um mapeamento entre a camada de virtualização e cada fonte de dados. Existem duas abordagens para a definição do schema global: Global as View (GAV) e Local as View (LAV). Na abordagem GAV, o schema global é definido em termos das fontes, enquanto que na abordagem LAV, as fontes são definidas em termos do schema global. A segunda abordagem para fazer integração de informações é a materialização, na qual os dados são materializados a nível global. Esta abordagem é geralmente utilizada em datawarehouses. Esta abordagem geralmente não tem qualquer informação não estruturada. O desafio nesta abordagem é a de determinar o conjunto de views que serão materializadas. Outro

problema nesta abordagem é a manutenção das views. Quando as fontes de dados subjacentes mudam, há a necessidade de manter-se a view materializada de forma eficiente.

É de se salientar que nenhuma das abordagens acima mencionadas leva em consideração critérios de preferência ao tentar resolver inconsistências entre fontes de dados. Por outro lado, ao especificar um sistema de integração de dados, o projetista pode muitas vezes incluir na especificação informações sobre a qualidade das fontes, como confiabilidade, e disponibilidade (Giuseppe De, Domenico *et al.*, 2004).

Ultimamente tem-se observado novas tendências na integração de informação, com enfoque em diferentes aspectos do processo de integração. Algumas dessas tendências são apresentadas a seguir.

2.5.1 Integração de informação orientada ao contexto

As abordagens de integração de informação que têm sido exploradas na literatura assumem que o usuário está ciente do schema das diferentes fontes de dados. No entanto, hoje isso nem sempre é possível nas empresas, devido à natureza de mudanças do negócio.
Isso levou à criação de novos aplicativos que podem inteligentemente descobrir informações relevantes de outros repositórios através da derivação do contexto das consultas de usuário. Estas aplicações não exigem que o usuário saiba sobre os outros repositórios e, conseqüentemente, diminui a complexidade das outras fontes de dados que podem ser terceirizadas para diferentes fornecedores (Mukesh e Manish, 2008).

2.5.2 Integração de informação baseada em evento

No entanto, existem muitas aplicações que exigem uma abordagem mais proativa na qual a integração de informações precisa ser disparada através de eventos. Por exemplo, se as vendas de um brinquedo nas diferentes regiões são menores que 100 unidades por mês, dar desconto de 10%. Essa integração de informações proativa é absolutamente útil para a tomada de decisões de negócio. O sucesso da integração de dados baseia-se na

forma como as fontes de dados podem propagar os eventos que ocorreram de forma eficiente. Em uma empresa, geralmente as aplicações são heterogêneos em termos de plataforma em que executam, no tipo de SGBD que utilizam, etc. Na maioria das vezes, as fontes de dados têm a capacidade de definir eventos que podem ser tanto publicados ou distribuídos para um grupo de ouvintes do evento. Algumas bases de dados em que estas aplicações estão sendo executadas poderiam não ter a capacidade de definir triggers ou publicar eventos. Em tais casos, o código da aplicação deve ser alterado para detectar eventos ou para acompanhar acontecimentos exteriores a fonte de dados que são de interesse para a integração das aplicações. Note que estes eventos podem eventos de banco de dados, eventos temporais definidos pelo usuário e eventos externos (Mukesh e Manish, 2008).

2.5.3 Integração de informação de fluxos

Neste tipo de configuração, é importante que se detecte eventos interessantes que poderiam ser repartidos para diversos fluxos, em tempo real, e tomar as ações necessárias.

Detectar eventos definidos em fluxos poderá exigir o armazenamento do estado da informação entre vários fluxos de dados. Os desafios em tal tipo de integração incluem: armazenar quantidade mínima de dados, a fim de detectar eventos; encontrar eventos em tempo real analisando uma quantidade mínima de dados (Mukesh e Manish, 2008).

2.5.4 Integração de informação centralizada no usuário

Muitas abordagens de integração, exploradas na literatura, assumem que a integração é feita por um usuário que tem permissão total de acesso às fontes de dados. Porém a integração de informações é, muitas vezes, executada sobre a camada de aplicações e não sobre a camada de dados.

A integração de informação centralizada no usuário é uma abordagem de integração que leva em consideração os direitos de acesso do usuário durante o processo de integração (Mukesh e Manish, 2008).

Como se pôde observar, o ramo da integração de dados continua em constante aperfeiçoamento e evolução. A prova disso é o surgimento do conceito de Enterprise Information Integration, apresentado em seguida.

2.6 Enterprise Information Integration

Com início no final dos anos 90, a integração de dados migrou do laboratório para a área comercial. Hoje, esta indústria é conhecida como Enterprise Information Integration (EII). A visão subjacente a esta indústria é fornecer ferramentas para integração de dados de múltiplas fontes, sem ter que primeiro carregar todos os dados em um repositório central como exigido por soluções anteriores.

Tal como acontece com qualquer nova indústria, EII tem enfrentado muitos desafios, alguns dos quais continuam a impedir o seu crescimento hoje. A seguir, são apresentados os desafios mais representativos (Alon, Anand *et al*., 2006).

- Escalabilidade e desempenho: O desafio inicial era o de convencer os clientes de que a idéia iria funcionar. As ferramentas EII freqüentemente enfrentam competição com ferramentas relativamente maduras de data warehousing. Para complicar as coisas, as ferramentas de datawarehouse começaram a enfatizar suas capacidades de tempo real, supostamente removendo uma das principais vantagens do EII sobre o datawarehouse. O desafio era o de explicar aos potenciais clientes que as implicações entre o custo de construção de um datawarehouse, o custo de uma consulta e o custo de acesso a dados obsoletos.

- Crescimento horizontal versus crescimento vertical: De uma perspectiva comercial, uma empresa de EII teve que decidir entre construir uma plataforma horizontal que pode ser usada em qualquer aplicação ou ferramentas especiais para a construção de plataformas verticais. O argumento para a abordagem vertical era de que os clientes preocupavam-se em resolver seu problema como um todo, em vez de pagar por mais uma parte da solução e de ter que se preocupar com integração entre as partes. O argumento para a abordagem horizontal é a generalidade do sistema e, muitas vezes, a incapacidade de decidir, em tempo hábil, qual foco vertical deve ser seguido.

- Integração ferramentas EAI e outros middlewares: De maneira simples, o contexto de gerenciamento de dados em produtos de middleware é um contexto bastante complicado. Diferentes companhias abordam problemas relacionados utilizando diferentes perspectivas e muitas vezes é difícil de ver exatamente que parte do problema uma ferramenta está resolvendo. O surgimento de ferramentas EII apenas tornaram o problema mais complicado. Apesar destes desafios, a concorrência feroz e o ambiente empresarial extremamente difícil após o estouro da Internet, a indústria de EII sobreviveu e está agora a emergindo como uma tecnologia indispensável para a empresa. Produtos de integração de dados são oferecidos pela maioria dos principais vendedores de SGBDs, e estão desempenhando um papel importante em produtos de análise de negócio.

Até aqui a integração de dados foi apresentada como um processo que permite o usuário visualizar os dados, residentes em diferentes fontes de dados, de maneira unificada. Foram apresentados também os principais problemas comumente encontrados neste processo e as técnicas e abordagens para a resolução dos mesmos. Logo em seguida, foi apresentada a evolução do ramo de integração para a área comercial, assumindo o nome de Enterprise Information Integration. Agora o foco do trabalho será o detalhamento de um sub-processo da integração de dados muito importante para o contexto geral e que é o objetivo primário deste trabalho. A sincronização de dados.

2.7 Sincronização de dados

A sincronização de dados é o processo de estabelecer consistência entre diferentes fontes de dados e a manter essa consistência ao longo do tempo. É fundamental em organizações que possuem replicação de informação em fontes de dados distintas, devido a diferenças de representação na concepção dos sistemas.

Às vezes, dados terceirizados necessitam ser armazenados em fontes próprias da organização para utilização por outros sistemas, caracterizando a necessidade de sincronização entre os dados terceirizados e os dados mantidos na organização.

Bancos de dados distribuídos são perfeitos exemplos de ambientes onde um processo de sincronização é necessário, pois em muitos destes sistemas é necessária a replicação imediata dos dados durante uma transação. A replicação é útil na melhoria da disponibilidade dos dados. O caso mais extremo é a replicação do banco de dados inteiro em todos os sites de um sistema distribuído, criando assim, um banco de dados completamente replicado. Isso pode melhorar notavelmente a disponibilidade porque o sistema pode continuar operando desde que pelo menos um site esteja em operação. A desvantagem da replicação completa é que ela pode reduzir drasticamente a velocidade das operações de atualização, uma vez que uma única atualização lógica deve ser executada em todas as cópias do banco de dados para manter as cópias consistentes. Para realizar esse procedimento de maneira eficiente devem estar disponíveis mecanismos de sincronização que considerem apenas as reais modificações sobre os dados e propaguem o mínimo de informação necessária para as cópias do banco.

Algumas ferramentas de sincronização foram desenvolvidas nos últimos anos. O Microsoft Sync Framework é um framework de sincronização de dados independente de tecnologia de armazenamento de dados e de protocolos de transporte de dados. Suporta qualquer tipo de fonte de dados através do conceito de provedores de fonte de dados específicos para cada domínio de informação. É amplamente utilizado no processo de sincronização remota de data sets através da plataforma ADO.NET, sincronização de sistemas de arquivo e feeds.

Outra solução disponível é o framework OpenSync. Este framework foi originalmente desenvolvido para dar suporte a sincronização de handhelds, celulares e informações pessoais como calendários, contatos e outros. Utiliza o mesmo conceito do framework da Microsoft através de plugins que fornecem a interface com as fontes de dados.

Ambos os frameworks tiveram influência na concepção da arquitetura de sincronização proposta. O objetivo desta arquitetura é suprir as necessidades levantadas anteriormente e auxiliar o processo de integração de dados organizacionais. Essa arquitetura é apresentada a seguir.

3 A LÓGICA PROPOSTA

A lógica de sincronização proposta utiliza-se de uma abstração de representação da informação comum a ambas as fontes de dados participantes do processo de sincronização e está baseada nas seguintes premissas:

- Reaproveitar a facilidade de transformação dos dados e geração automática dos deltas, conseguida através de frameworks de mapeamento;
- Utilizar-se do conceito de entropia da informação para acelerar o tempo de processamento do algoritmo de sincronização, dado que uma informação possa ser identificada por um conjunto mínimo de requisitos;
- Possibilitar a integração de fontes heterogêneas de dados uma vez que as fontes estão abstraídas em um conceito comum de representação da informação;
- Os dados sendo sincronizados devem pertencer ao mesmo contexto conceitual, ou seja, uma unidade de informação de projeto pode estar representada de maneira diferente entre as fontes de dados, mas o conceito de projeto em ambas as fontes deve ser o mesmo.

Esta lógica está representada na figura 1.

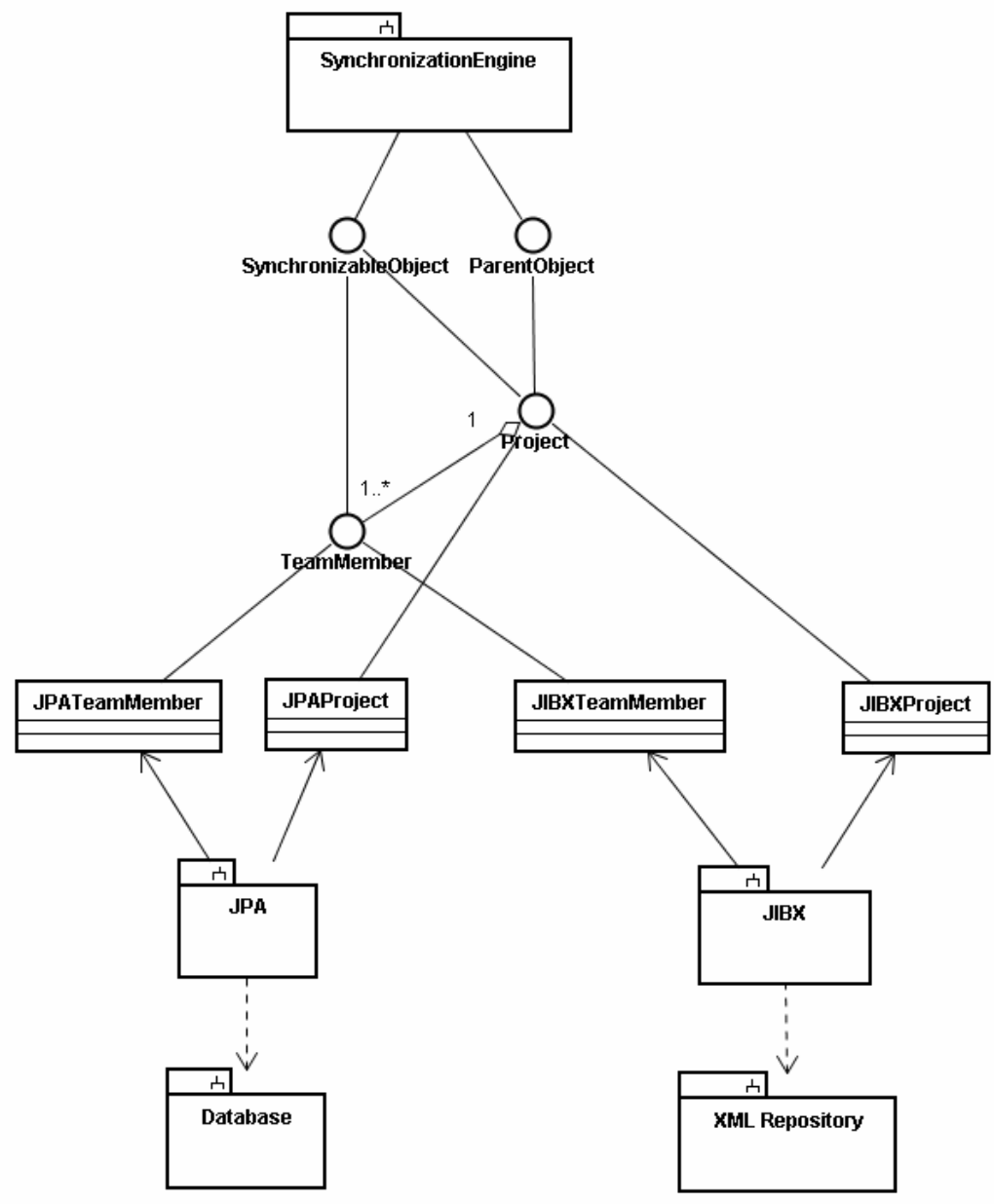

Figura 1 – A lógica de sincronização proposta.

A figura 1 apresenta uma série de camadas de abstração que visam suprir os seguintes requisitos essenciais de uma solução de sincronização de fontes heterogêneas de dados:

3.1 Abstração de representação dos dados

Um algoritmo de sincronização de fontes heterogêneas de dados deve ser capaz de abstrair as diferenças entre as representações de um mesmo conceito. O algoritmo proposto utiliza um modelo flexível de trabalho ao qual as fontes de dados devem adequar-se para que se garanta a abstração da heterogeneidade das fontes.

Essa abstração é conseguida pela definição de um modelo comum de sincronização, representado na figura 1 pelas interfaces *SynchronizableObject* e *ParentObject*. A primeira define os métodos necessários para realização do processo de sincronização enquanto que a segunda trata de estruturas de dados hierárquicas. Abaixo temos o exemplo de definição dessas interfaces.

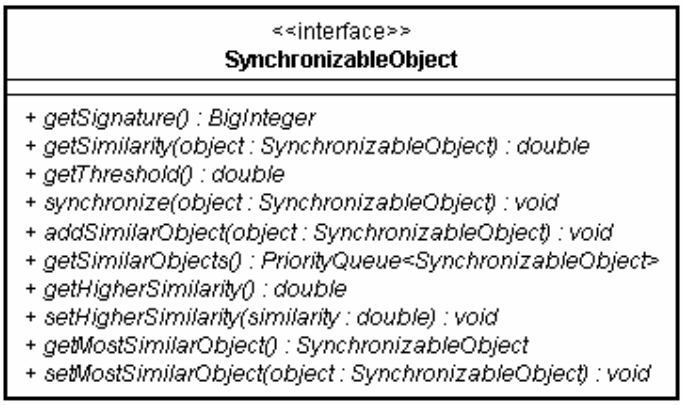

Figura 2 - A interface SynchronizableObject.

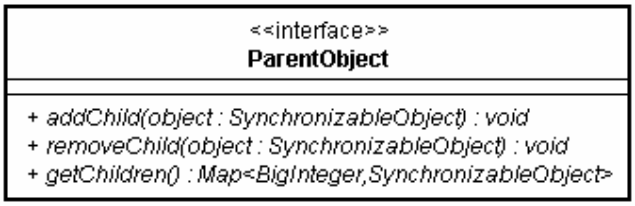

Figura 3 - A interface ParentObject.

3.2 Entropia da informação

O conceito de entropia da informação foi definido em (Shannon, 2001), como uma medida da quantidade de informação proveniente em um processo, ou seja, a

informação provida por um processo sobre si mesmo e sobre outros processos relacionados (Robert, 1990). Também pode ser definida como uma medida de incerteza associada a uma variável aleatória ou ao comprimento mínimo necessário que uma mensagem deve ter para transmitir informação.

Em (William, 2000), é abordado o problema da integração de dados em bancos de dados distribuídos e heterogêneos, como os bancos existentes na World Wide Web, que não possuem identificadores comuns entre os dados e portanto não podem ser considerados durante o processo de integração.

A lógica de sincronização desconsidera qualquer identificador associado a um dado, pois em muitas aplicações tais identificadores não existem, como parágrafos em arquivos de texto. Ainda, dados com o mesmo conteúdo podem estar representados com identificadores diferentes (Sudarshan e Hector, 1997).

Para abstrair os identificadores associados a um determinado dado, o algoritmo de sincronização proposto utiliza uma chave hexadecimal, gerada pelo algoritmo MD5, a partir dos atributos que identificam unicamente o dado em relação aos demais, aplicando uma função de n-grama para eliminar eventuais diferenças textuais. Essa chave é obtida através do método *getSignature()*.

Um *n*-grama é uma seqüência de *n* letras ou palavras, onde *n* geralmente é 1, 2 ou 3, respectivamente monograma, bigrama e trigrama. N-gramas são utilizados em várias áreas de processamento estatístico de linguagens naturais e análise de seqüenciamento genético.

Como exemplo, considere a figura 4.

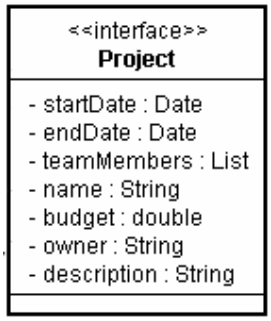

Figura 4 - A interface Project em detalhe.

A chave hexadecimal que identifica unicamente os dados poderia ser obtida através da concatenação dos atributos *name*, *owner* e *description*, aplicados a função de n-grama de tal maneira que se dois projetos possuírem as mesmas informações, mas com diferenças textuais irrelevantes, o algoritmo consideraria os projetos como a mesma unidade de informação.

É nesta etapa de escolha dos atributos identificadores que apresenta-se o conceito de entropia da informação, pois cada atributo deve ser analisado e os atributos que contiverem a maior entropia, ou seja, a maior quantidade de informação que represente o dado como um todo, deve ser escolhido para compor a chave. Esta escolha é feita geralmente pelo analista de negócio devido ao conhecimento do domínio.

Este conceito de chave foi baseado no algoritmo de diferenciação *rsync* que trabalha sobre blocos de texto. Este algoritmo utiliza duas assinaturas por bloco de texto, uma assinatura de computação rápida e menos precisa e outra de computação mais cara com nível de colisão mínimo (Tridgell, 2000). Segundo o mesmo, a principal propriedade que o algoritmo deve possuir é que se dois blocos de texto são diferentes deve-se ter uma probabilidade muito pequena de terem a mesma assinatura, sendo o algoritmo *message digest* o mais utilizado para esse tipo de requisito. A mesma idéia de geração de uma chave identificadora é aplicada no algoritmo proposto, o que elimina a necessidade de qualquer identificador previamente presente nos dados.

3.3 Similaridade entre objetos

O algoritmo de similaridade apresentado está baseado em diversos outros algoritmos publicados na literatura. Destacam-se as publicações (Sudarshan e Hector, 1997), (Raihan, Archana *et al.*, 2005), (Ralescu, 2007), (Ana, Filippo *et al.*, 2005), (Joyce e Altigran, 2003), (Sven, 2007), (Jin-Mao Wei; Shu-Qin Wang; Wei Zheng; Jing Wang; Jun-Ping You; Jie Zhang; Liu, 2006) nos quais é abordada a similaridade entre estruturas hierárquicas de objetos, ou árvores.

Em (Vipul e Amit, 1996), é discutida a similaridade semântica entre objetos de negócio considerando aspectos de contexto e domínio dos mesmos em relação ao mundo real.

Já (Richard, 1964), demonstra claramente a aplicação da entropia da informação na identificação de similaridade entre subconjuntos de um conjunto de objetos considerando-se as propriedades existentes em cada objeto pertencente ao subconjunto.

Com base nos trabalhos citados, a lógica de sincronização define um conjunto de métodos responsáveis pelo gerenciamento de similaridade entre as unidades de negócio.

O método *getThreshold()* define o nível de similaridade mínimo aceitável para que dois objetos sincronizáveis sejam considerados semelhantes. Quanto mais alto o valor menor a probabilidade de erro na comparação textual, mas menor a chance de semelhança entre textos com diferenças irrelevantes. Este valor deve ser ajustado conforme natureza do negócio e através de testes de comparação.

Cada objeto elegível para sincronização deve guardar uma lista de objetos similares a ser utilizada pelo algoritmo durante o processamento. Para isso a interface SynchronizableObject define os métodos *addSimilarObject(SynchronizableObject object)* e *getSimilarObjects()*.

Outras informações que devem ser disponibilizadas ao algoritmo de sincronização são a maior similaridade e o objeto mais similar computados. Para isso são definidos os métodos *getHigherSimilarity()* e *setHigherSimilarity(double similarity)* e *getMostSimilarObject()* e *setMostSimilarObject(SynchronizableObject object)* respectivamente.

O método *synchronize(SynchronizableObject object)* verifica se dois objetos estão sincronizados de acordo com seus atributos e, se necessário, sincroniza as informações entre os objetos.

O método mais importante para a eficiência do algoritmo é o método *getSimilarity(SynchronizableObject object).* Este método é responsável por comparar os atributos mais relevantes entre dois objetos sincronizáveis e retornar um valor de similaridade entre os objetos. Aqui o conceito de entropia da informação minimiza o custo computacional do algoritmo e garante que informações semelhantes possam ser consideradas iguais.

3.4 Mapeamento e aspecto semântico

Em (Alon, Anand *et al.*, 2006), é ressaltada a dificuldade de configuração de aplicações de integração devido à necessidade de criação de descritores das fontes de dados e ao mapeamento semântico entre as fontes e o domínio do negócio. Já (Youn, 1992), complementa que se os domínios das fontes de dados tiverem interpretação comum dos valores e o relacionamento entre esses dados for bem definido, o processo de mapeamento é relativamente direto. Porém se os domínios são diferentes e os relacionamentos não estão bem definidos, um processo de transformação é necessário.

Considere novamente a figura 4. No modelo relacional a data de início de um projeto poderia estar definida em uma única coluna de uma tabela, como DTA_HORA_INICIO_PROJETO = 2008-11-23 17:23:00.000 enquanto que no repositório de arquivos XML a data poderia estar definida em dois atributos com formatação diferente, DATA_INICIO = 23/11/2008 e DATA_FIM = 17h23min. Neste caso, não é possível um mapeamento direto entre as duas abstrações e um processo de transformação é necessário.

3.5 O algoritmo de sincronização

O algoritmo de sincronização proposto está baseado nas mesmas premissas definidas para o algoritmo *rsync* e divide-se em 13 passos:

1. Primeiramente criam-se dois mapeamentos temporários, um para a fonte e outro para o destino, com todas as unidades de informação candidatas a sincronização, ou seja, cada objeto que implementa a interface *SynchronizableObject,* é adicionado a um mapeamento ou hash utilizando a sua assinatura como chave e o seu conteúdo como valor. Se os dados possuem uma estrutura hierárquica, apenas os pais são adicionados ao mapeamento. A assinatura de um objeto é obtida através do método *getSignature().* Este passo está ilustrado na figura 5.

Mapeamento fonte

Projeto A	54ADB8 9BBA21
Projeto B	456ABC D392EE
Projeto E	32987D EAB3BA
Projeto F	3347DB C3AB6C
Projeto C	0D7B6C 4A3C11
Projeto D	23BCAA A34CC2

Mapeamento destino

0D7B6C 4A3C11	Projeto C
286EAB C35B63	Projeto G
8B267C BA73DB	Projeto M
54ADB8 9BBA21	Projeto A
3347DB C3AB6C	Projeto F
EBAB36 BC7AB2	Projeto R

Figura 5 - O mapeamento entre fonte e destino.

2. No segundo passo o algoritmo compara as chaves dos mapeamentos temporários de fonte e destino, que foram geradas a partir dos atributos mais relevantes do objeto, buscando similaridade entre as assinaturas. Todos os objetos que tiveram similaridade de assinatura são enviados para o passo 7. Os demais são enviados para o passo 3.

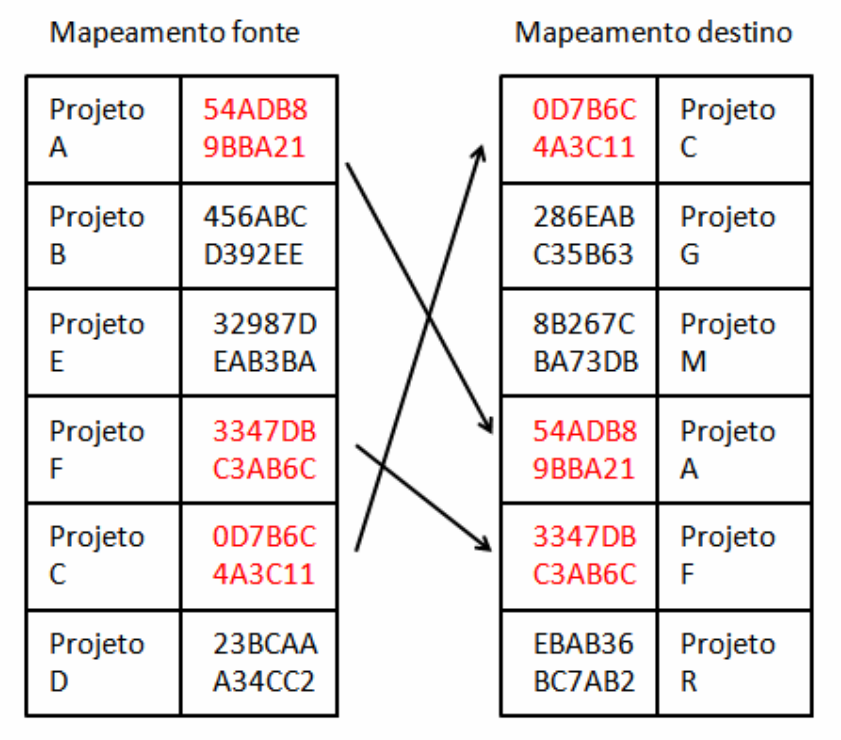

Figura 6 - A comparação das chaves geradas a partir dos atributos mais relevantes.

3. Os objetos que não obtiveram similaridade de assinatura são enviados para comparação de força bruta. Nesse processo todos são comparados entre si mapeando as maiores similaridades entre eles.

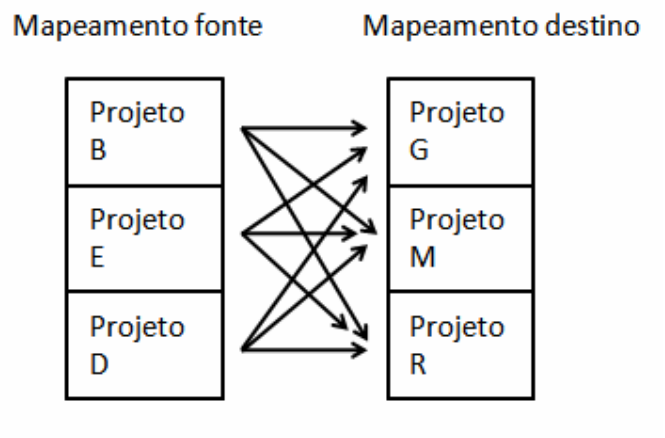

Figura 7 - A comparação de força bruta.

4. Todos os objetos sincronizáveis possuem uma fila de objetos similares ordenada em sentido decrescente, estando o objeto mais similar no início da fila. Se

durante o processo de comparação um objeto destino (Projeto M) possuir similaridade maior que o *threshold* de um objeto fonte (Projeto B), ele entra na fila de objetos similares do objeto fonte (Projeto B) de acordo com o valor da similaridade encontrada. O método utilizado nesse procedimento é *addSimilarObject(SynchronizableObject object).*

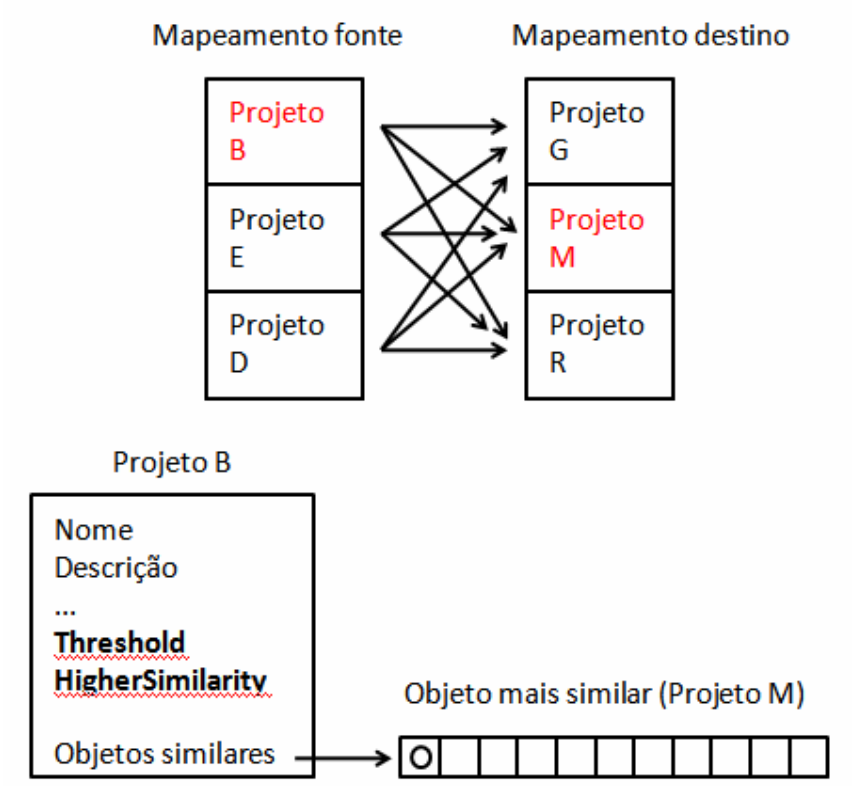

Figura 8 - A fila de objetos que possuem similaridade superior ao threshold da unidade de informação.

5. Caso a similaridade seja maior que a maior similaridade computada até o momento em relação ao objeto fonte (Projeto B), o valor da maior similaridade computada é atualizado para esse novo valor de similaridade e o objeto fonte (Projeto B) é armazenado como objeto mais similar no objeto destino (Projeto M). Nesse processo são utilizados os métodos *getHigherSimilarity(),*

setHigherSimilarity(Double similarity) e *setMostSimilarObject(SynchronizableObject object)*.

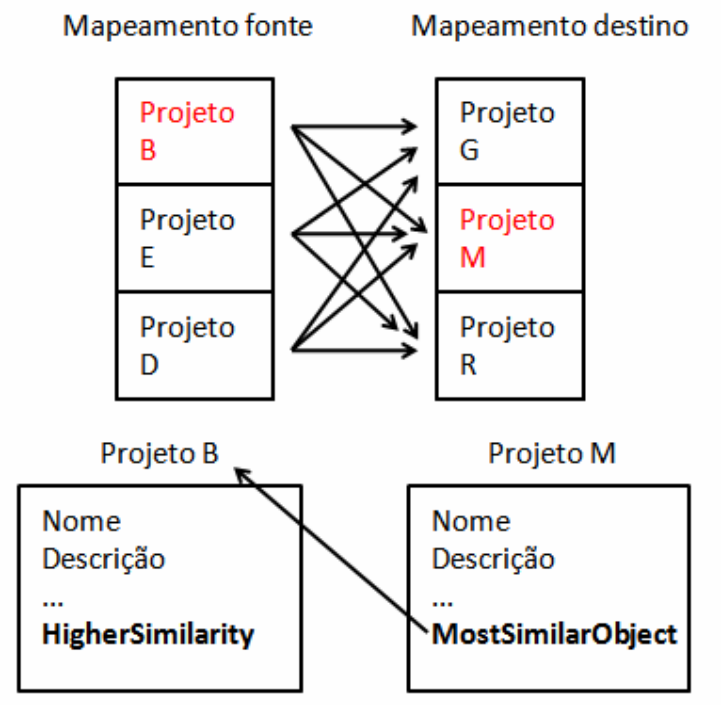

Figura 9 - O armazenamento da maior similaridade.

6. Ao final do processo, todos os objetos foram comparados entre si e as filas de similaridade foram associadas a cada objeto fonte. Agora o algoritmo itera sobre todos os objetos fonte buscando o objeto destino mais similar na fila de similaridades. Caso encontre, o algoritmo verifica se nesse objeto destino encontrado está armazenado como objeto mais similar o próprio objeto fonte. Se sim, significa que o objeto fonte é o objeto mais similar ao objeto destino e, portanto, a similaridade entre os objetos é legítima. Os métodos *getSimilarObjects()* e *getMostSimilarObject()* são invocados durante a iteração.

7. Caso uma similaridade seja encontrada o algoritmo verifica se o conteúdo dos objetos é idêntico, e se não for, sincroniza os atributos através do método *synchronize(SynchronizableObject object)*.

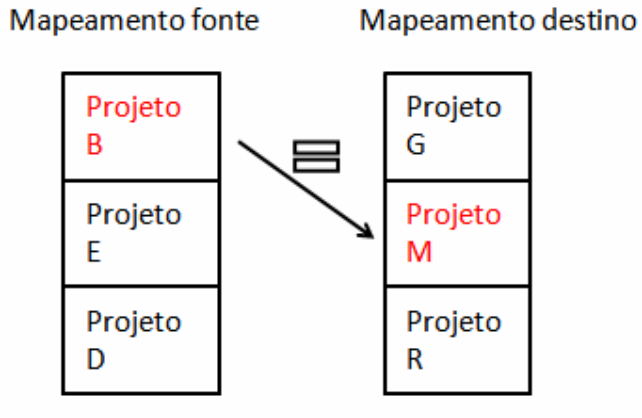

Figura 10 - A sincronização dos projetos B e M.

8. Após a sincronização dos atributos os objetos similares são removidos dos mapeamentos temporários.

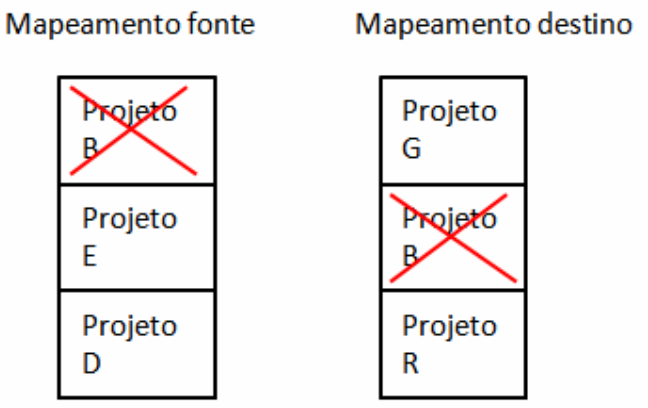

Figura 11- Remoção dos objetos sincronizados.

9. Durante o processo de sincronização dos atributos os deltas são gerados automaticamente pelos frameworks de mapeamento e ficam armazenados para posterior processamento.

10. Após a sincronização dos atributos, caso o objetos implementem a interface *ParentObject*, os filhos dos objetos, obtidos através do método *getChildren()*, são enviados para o processo de sincronização, de maneira recursiva.

11. Após a sincronização dos filhos, os objetos fonte sem par similar, ou seja, que restaram no mapeamento temporário fonte, são adicionados ao mapeamento destino com o método *addChild(SynchronizableObject object)*.

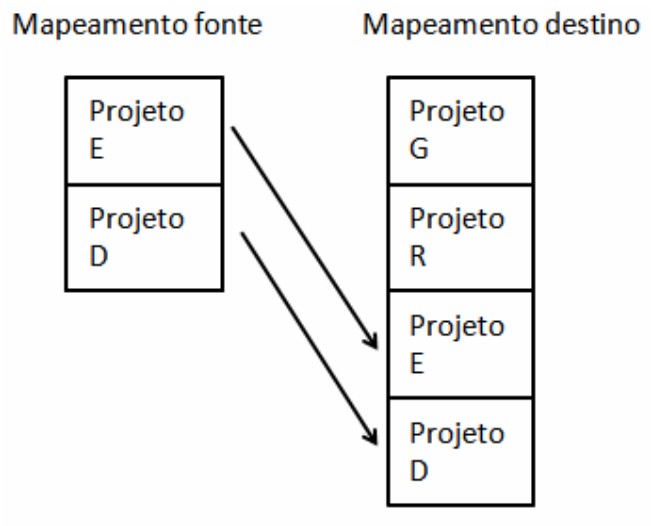

Figura 12 - Inserção dos objetos fonte restantes no mapeamento destino.

12. Logo em seguida os objetos destino sem par similar, ou seja, que restaram no mapeamento temporário destino, são removidos do mesmo com o método *removeChild(SynchronizableObject object)*.

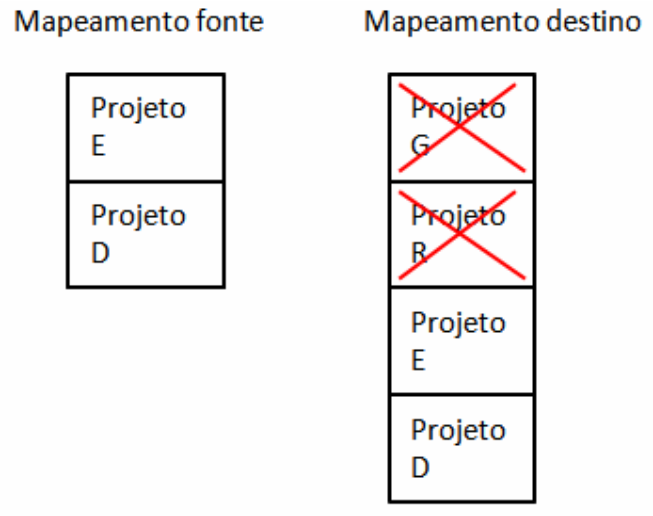

Figura 13 - Remoção dos objetos destino restantes do mapeamento destino.

13. Finalizado o processo de sincronização pelo algoritmo, os objetos encontram-se sincronizados e os deltas gerados pelos frameworks de mapeamento são persistidos nas fontes de dados.

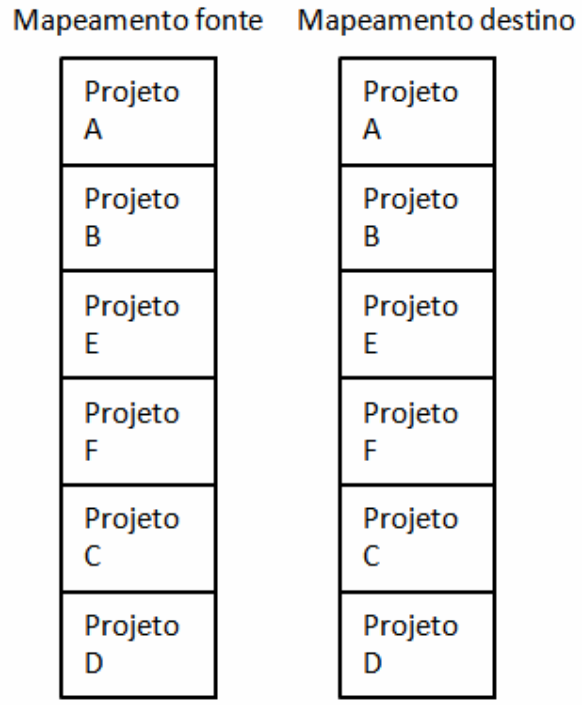

Figura 14 - Os mapeamentos após a sincronização.

Após a sincronização, os mapeamentos modificados devem ser traduzidos em ações na fonte destino, de maneira a torná-la idêntica a fonte de origem. Para isso são utilizados frameworks de mapeamento de dados.

3.6 Frameworks de mapeamento de dados

Segundo (Fakhar e Muhammad Ahmad, 2007), os paradigmas orientado a objetos e relacional representam a informação de maneira diferente um do outro, resultando em modelos distintos de representação da mesma informação, com isso aumentando o esforço de concepção e manutenção de sistemas.

Frameworks de mapeamento de dados são estruturas de software que visam facilitar a transformação de dados entre diferentes formas de representação de uma mesma abstração, por exemplo, transformar um dado representado em XML para sua correspondente representação em uma tabela de um banco de dados relacional. Alguns frameworks de mercado amplamente conhecidos são o JAXB e JIBX, que são frameworks de mapeamento objeto-XML, e JPA e Hibernate que são frameworks de mapeamento objeto-relacional. O mapeamento objeto-relacional é a técnica de conversão de registros de um banco relacional em instâncias de objetos pertencentes a um ambiente de programação orientado a objetos (Patrick Connor e Marc, 2007).

Um dos grandes benefícios dos frameworks de mapeamento de dados é a geração automática dos deltas quando os dados originais sofrem algum tipo de modificação.

3.7 Geração automática dos deltas

Algoritmos de diferenciação, ou delta, são algoritmos que computam diferenças entre dois arquivos ou literais e têm grande utilidade quando múltiplas versões de dados devem ser armazenadas, transmitidas ou processadas (James, Kiem-Phong *et al.*, 1998).

Inúmeros algoritmos foram desenvolvidos ao longo dos últimos anos, principalmente para diferenciar fontes textuais. Com a evolução dos repositórios de dados, objetos binários começaram a ser armazenados com maior freqüência e demandaram por mecanismos de sincronização que consideravam esse tipo de dado. Hoje existem muitas fontes de dados em diferentes formatos binários, como áudio e imagens, que necessitam de gerenciamento eficiente por parte dos algoritmos (James, Kiem-Phong *et al*., 1998).

Os algoritmos de diferenciação tiveram também que se adaptar para considerar o contexto da informação durante o processamento dos dados devido à troca massiva de informações pela rede através do XML. Algoritmos de diferenciação de fontes textuais não entendiam a informação contida em fontes de dados estruturadas hierarquicamente (Raihan, Archana *et al.*, 2005).

O grande benefício da integração do algoritmo de sincronização proposto anteriormente com frameworks de mapeamento é a geração e persistência automática do delta, ou seja,

das diferenças entre os dados originais e modificados. Este benefício diminui a complexidade e o tempo de desenvolvimento de soluções de integração de dados e reaproveita soluções difundidas amplamente pelo mercado.

3.8 A arquitetura resultante

O resultado da aplicação desta lógica de sincronização é uma arquitetura de software que minimiza os custos do processo de integração, pois abstrai a heterogeneidade das fontes de dados, devido ao modelo comum de mapeamento da informação; e acelera o processo de desenvolvimento, através da automatização da geração e processamento dos deltas com os frameworks de mapeamento. Os custos de persistência das informações também são reduzidos, pois a propagação de scripts de atualização é limitada ao número real de modificações nas fontes de dados. Esta arquitetura está representada na figura 15.

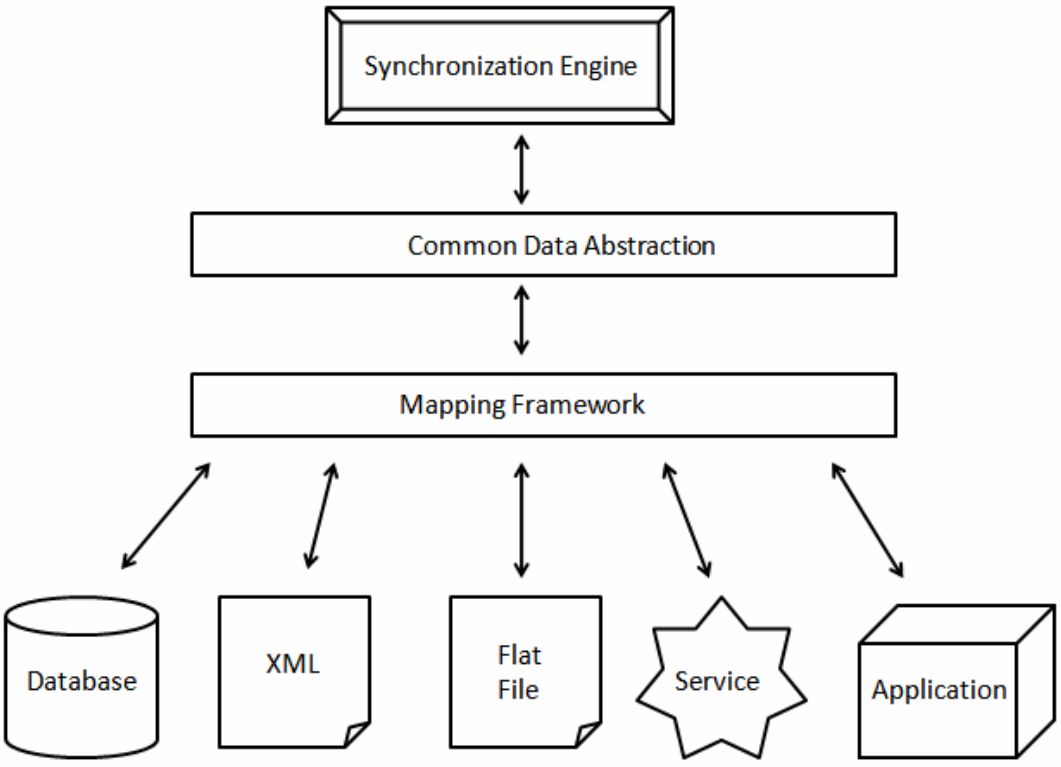

Figura 15 - A arquitetura resultante.

A concepção dessa arquitetura visa possibilitar a interoperabilidade entre aplicações. As informações corporativas encontram-se armazenadas em distintas fontes de dados e uma visão sistêmica dessa informação é necessária ao processo de tomada de decisão. Essa arquitetura facilita a integração dessas informações.

A reutilização de frameworks de mapeamento existentes no mercado possibilita a utilização de diferentes tecnologias de armazenamento, o que traz agilidade ao processo de desenvolvimento em um contexto onde as soluções são específicas para cada domínio de negócio.

Em fim, a arquitetura surge como uma alternativa à preservação, armazenamento, migração, integração e disponibilização da informação no contexto corporativo.

4 ESTUDO DE CASOS

4.1 Concepção da arquitetura proposta no projeto FioLattes

4.1.1 Histórico

O Sistema Fiolattes é um sistema definido pela FIOCRUZ para obter informações adicionais, não solicitadas no Sistema de Currículos Lattes, de seus afiliados. As informações deste sistema são informadas junto com os dados do currículo Lattes. O conjunto formado pelas informações do currículo Lattes e do Sistema Fiolattes formam o currículo FIOLATTES, que é enviado à base de currículos da FIOCRUZ.

Neste sistema estão os detalhamentos, solicitados pela FIOCRUZ, para algumas informações preenchidas no Sistema de Currículos Lattes, além de um conjunto de informações adicionais, específicas da instituição, que o usuário preenche.

O sistema verifica o correto preenchimento das informações da FIOCRUZ, semelhante ao que faz o Sistema de Currículo Lattes com as informações curriculares, antes de permitir que o usuário envie seus dados. As regras de consistência, definidas pela FIOCRUZ, garantem a qualidade da informação recebida na instituição.

4.1.2 O sistema de sincronização FioLattes

Para transferir as informações fornecidas no Sistema de Currículos Lattes para o Sistema Fiolattes era necessário um sistema de back-end capaz de analisar as duas fontes de dados e persistir somente as diferenças encontradas entre as bases de dados. Este sistema deveria ser capaz de trabalhar com bases de dados que possuíam diferentes modelos de dados e algumas informações representadas de maneira distinta. Outro requisito fundamental do sistema era a capacidade de associar dados que possuíam pequenas diferenças de descrição, mas que representavam a mesma informação, com o intuito de manter a integridade dos dados. Por exemplo, um projeto chamado “Pesquisa em inovação tecnológica” deveria ser associado a um projeto chamado “Pesquisa na área de inovação tecnológica” se ambos possuíssem o mesmo significado.

A figura abaixo apresenta o cenário de sincronização do sistema FioLattes.

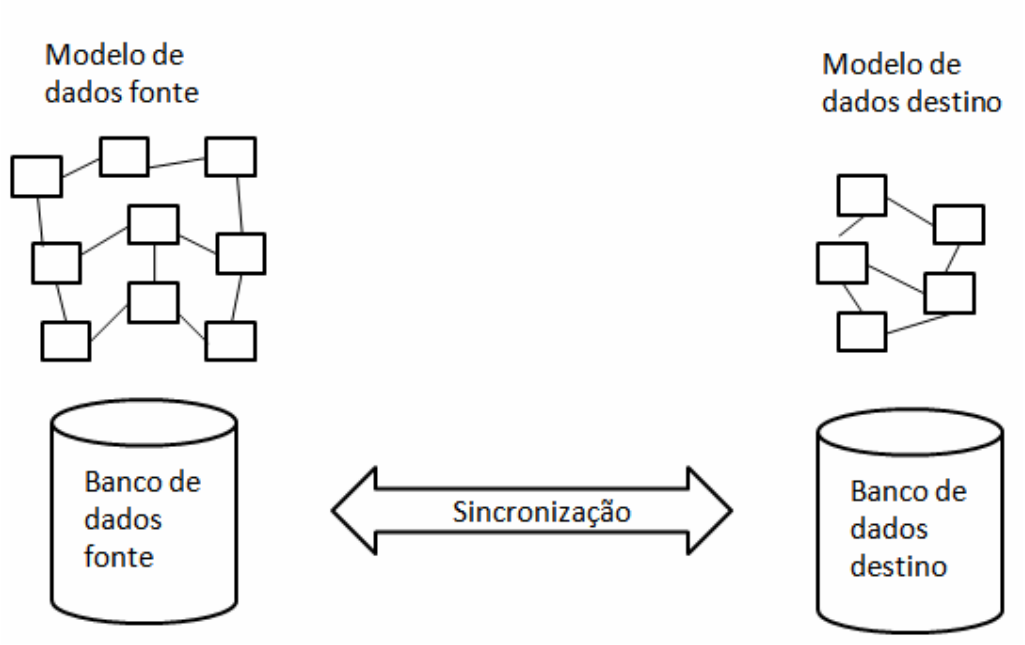

Figura 16 - O cenário de sincronização do sistema FioLattes.

4.1.3 A solução tecnológica

O primeiro requisito a ser considerado na elaboração da solução tecnológica capaz de suprir os requisitos levantados anteriormente foi a capacidade do sistema de trabalhar com diferentes modelos de dados. Para isso pensou-se na abstração descrita anteriormente representada pela interface *SynchronizableObject*.

Em relação ao requisito de associação de informações iguais com diferentes representações foram utilizados os conceitos de n-grama e entropia da informação. No exemplo do projeto, cada atributo que o descrevia recebeu um peso correspondente a sua importância na identificação do projeto, e a cada um desses atributos foi aplicada a técnica de transformação dos textos em sua respectiva descrição fonética, para eliminar diferenças de escrita.

Em seguida foi considerado o requisito de persistência apenas das diferenças entre as bases de dados, de maneira a garantir o desempenho do sistema, executando o menor número de operações necessárias para a correta sincronização entre as bases. Para isso

houve pesquisa nas áreas de algoritmos de diferenciação até a concepção do algoritmo apresentado anteriormente que utiliza uma assinatura de computação rápida e menos precisa e outra de computação mais cara com nível de colisão mínimo. Com esse algoritmo e os conceitos de entropia da informação e n-grama foi possível reduzir em muito a quantidade de operações necessárias para sincronização entre as bases.

4.1.4 Resultados obtidos

O s resultados obtidos neste projeto foram:

- A minimização das operações em banco. As operações de atualização foram limitadas ao número real de alterações na fonte de dados de origem;
- Desempenho acima do esperado. No começo do projeto as estimativas de sincronização estavam em 300ms por currículo, mas na prática o tempo foi de 90ms por currículo;
- Alta taxa de correção dos dados. Todas as redundâncias na fonte de dados destino foram eliminadas com os conceitos de similaridade e entropia da informação utilizados pelo algoritmo.

4.2 Aplicação da arquitetura proposta no projeto Portal Inovação

4.2.1 Histórico

O Portal Inovação consiste em serviço de governo eletrônico que visa promover inovação no Brasil por meio de um espaço de cooperação entre os diferentes atores do Sistema Nacional de Inovação. Trata-se de iniciativa do Ministério da Ciência e Tecnologia (MCT) no contexto das ações do governo federal no âmbito da Lei de Inovação. O Portal é resultado de ação coordenada pelo Centro de Gestão e Estudos Estratégicos (CGEE) e de execução do Instituto Stela, tanto em sua fase piloto (2004) como no desenvolvimento e no lançamento da primeira versão (2005). Além das entidades citadas, o Portal contou com estreita participação dos representantes dos setores acadêmico, tecnológico, empresarial e governamental.

A Fase III do Portal Inovação apresenta três frentes de pesquisa, desenvolvimento e inovação nas áreas de governo eletrônico, sistemas de conhecimento e inovação. Para o fortalecimento do Portal como instrumento de apoio à cooperação tecnológica, prevêem-se ações que visem:

- A ampliação de seus serviços e recursos atuais (bem como a inclusão de mais atores da rede de inovação);
- A conexão com fontes de informação correlatas e disponíveis em projetos públicos ou de associações empresariais; e
- A viabilização de visões setoriais ou temáticas da inovação.

Essas linhas de ação formam as três famílias de pesquisas e desenvolvimento necessárias à Fase III do Portal Inovação, denominadas, respectivamente, de (a) ampliação e consolidação de seus instrumentos; (b) interoperabilidade com outras fontes de informação; e (c) recortes temáticos ou setoriais em áreas específicas de interesse estratégico à inovação.

4.2.2 O sistema de sincronização dos Currículos Lattes com a base de dados do Portal Inovação

O sistema de sincronização de Currículos Lattes é um sistema back-end responsável por atualizar as informações dos currículos, que são modificadas pelo site da Plataforma Lattes sob responsabilidade do CNPq, com as informações do Portal Inovação. Este sistema faz parte da linha de ação “interoperabilidade com outras fontes de informação” citada anteriormente.

Os currículos provenientes do CNPq são distribuídos na forma de arquivos XML enquanto que as informações do Portal Inovação encontram-se armazenadas em bases de dados relacionais.

Há grande diferença de representação das unidades de informação entre as duas fontes de dados, por isso é necessária a transformação e validação das informações durante o processo de sincronização. Outro requisito fundamental é que as informações do

currículo possuem caráter fortemente hierárquico, então a solução tecnológica deve ser capaz de trabalhar de maneira eficiente com este tipo de dado.

O principal requisito deste sistema é o desempenho, uma vez que o número de currículos já supera um milhão, o número de informações disponibilizadas é considerável e a taxa de atualização dos currículos é de aproximadamente 7 mil currículos diários.

A figura abaixo apresenta o cenário de sincronização do sistema de sincronização dos Currículos Lattes.

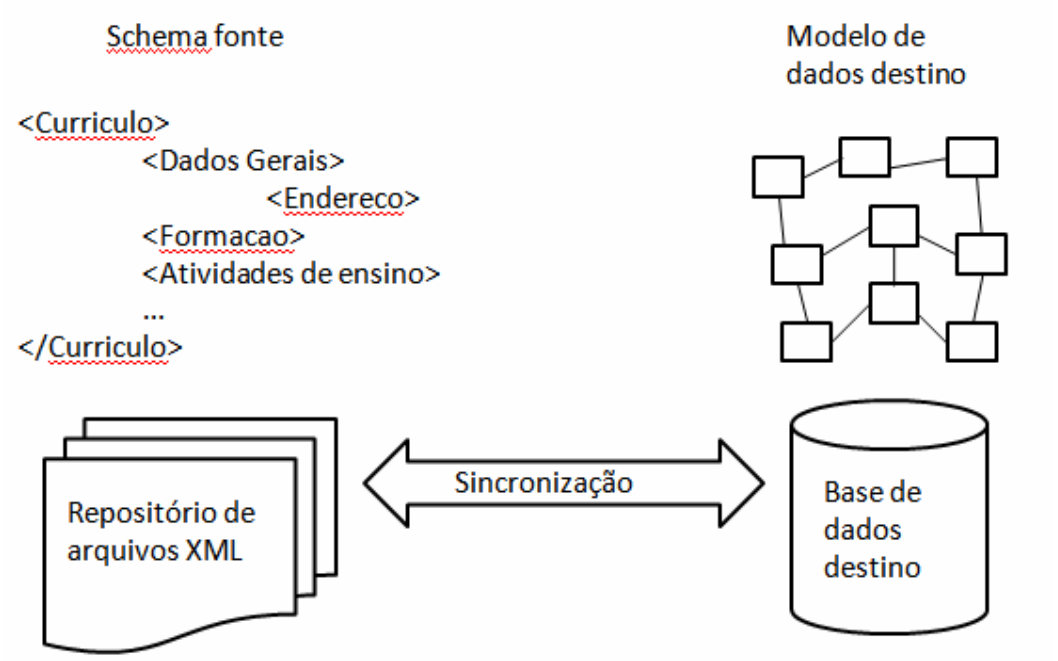

Figura 17 - O cenário do sistema de sincronização dos Currículos Lattes.

4.2.3 A solução tecnológica

A solução tecnológica foi dividida em três etapas:

1. Leitura das informações dos currículos a partir da fonte XML;
2. Validação e transformação das informações de acordo com as regras de negócio especificadas;

3. Leitura das informações dos currículos a partir da base de dados relacional destino;
4. Mapeamento das unidades de negócio para a abstração comum do algoritmo de sincronização;
5. Persistência das diferenças entre as fontes na base relacional destino;
6. Sincronização das fontes de dados.

4.2.4 Leitura das informações dos currículos a partir da fonte XML

Para resolução do problema de leitura das informações dos currículos a partir de arquivos XML foram pesquisadas diversas soluções comumente utilizadas pelo mercado, como JAXB, JIBX, XStream e outros. Através de testes comparativos de desempenho, optou-se pela utilização da solução JIBX.

O framework JIBX é um framework de mapeamento objeto-XML que necessita de um arquivo descrevendo o mapeamento das informações nas unidades de negócio para os elementos descritos no XML Schema que descrevem a mesma unidade. Este arquivo é chamado de *binding*, e é utilizado pelo compilador JIBX para embutir código no código compilado das unidades de negócio, no caso, os *.class* da aplicação.

Nesta etapa o XML Schema que descrevia as unidades de negócio do Currículo Lattes foi atualizado para ser utilizado na geração automática das classes de negócio, através de uma ferramenta contida na distribuição do JIBX chamada Xsd2Jibx.

Após a geração das classes de negócio, todas as informações contidas nas classes foram mapeadas no arquivo de *binding* para os respectivos elementos do XML Schema do Currículo.

Este mapeamento permite ao JIBX carregar as informações do XML e popular as classes de negócio automaticamente, de maneira que ao final do processo o Currículo Lattes estivesse totalmente carregado em memória.

4.2.5 Validação e transformação dos dados de acordo com as regras de negócio especificadas

A segunda etapa no processo de carga das informações dos Currículos Lattes para o Portal Inovação foi a validação e transformação das informações carregadas em memória na etapa anterior. Havia grande diferença de representação dos dados entre as fontes, tanto na formatação dos campos, quanto ao número de campos que armazenavam determinada informação. Foram construídos algoritmos de conversão de dados, como hora, classificações e campos de *lookup*. Como exemplo, no XML a representação da data de nascimento de uma pessoa estava no formato DATA_NASCIMENTO = 12051985 e HORA_NASCIMENTO = 1523. Já na base relacional a mesma informação estava representada no formato DTA_NASCIMENTO = 1985-05-12 15:23:00.000. O sexo estava representado no XML como SEXO = Masculino, já na base como COD_SEXO = M. Em relação a campos de lookup, diversas regras de negócio foram encontradas mas a título de exemplo, enquanto que no XML a representação de estado e cidade de nascimento estava no formato ESTADO_NASCIMENTO = Santa Catarina e CIDADE_NASCIMENTO = Florianópolis, na base relacional os mesmos eram representados por UF_NASCIMENTO = SC e COD_MUNICIPIO = 4906. Ambos os dados da base eram provenientes de tabelas de *lookup*, as chamadas “tabelas amarelas”.

Algumas regras de validação dependiam de *lookups* em tabelas amarelas como instituições, cursos e áreas de formação.

Ao final do processo as informações em memória encontravam-se validadas e transformadas de acordo com as regras de negócio.

4.2.6 Leitura das informações dos currículos a partir da base de dados do Portal Inovação

O próximo passo envolvia carregar as informações da base de dados relacional destino para a memória, na mesma estrutura de objetos gerada automaticamente pelo JIBX. Para isso novamente pesquisou-se frameworks de mapeamento objeto-relacional utilizados pelo mercado como Hibernate, JPA, JDO e outros.

(Pieter Van, Derrick *et al.*, 2006) coloca que ferramentas objeto-relacional acarretam overhead de tradução para todas as operações de persistência e são proporcionalmente mais lentas que soluções diretamente ligadas ao modelo relacional. Porém, essas ferramentas permitem aos desenvolvedores utilizar técnicas de modelagem de domínio orientadas a objetos, deixando a cargo da ferramenta a tradução para o modelo relacional. Ainda assim, especificar as regras de tradução acrescenta tempo e energia para o processo de desenvolvimento.

Após a realização de testes de desempenho, facilidade de mapeamento, geração de consultas e flexibilidade das ferramentas, optou-se por desenvolver uma solução própria, devido principalmente a requisitos de desempenho e número de consultas geradas pelos frameworks. Como esses frameworks têm o objetivo de cobrir grande parte das operações necessárias à persistência de objetos em bases de dados relacionais, da forma mais automatizada possível, acabaram tornando-se pesados para persistência de informações em ambientes que exigem grau de desempenho elevado, além de gerarem consultas não otimizadas no estágio atual de desenvolvimento. Outros inconvenientes encontrados, que pesaram muito na decisão de utilização de uma solução própria foram a dificuldade de mapeamento das unidades de negócio quando já existe um modelo de dados pré-definido e complexo, como é o caso do modelo relacional do Lattes e os erros encontrados nos frameworks de mapeamento de dados testados.

Considerando-se os pontos levantados, foi desenvolvida uma solução de carga das informações da base relacional própria, utilizando a extração da informação em blocos de dados. Esta solução foi embasada em pontos importantes constatados durante o processo de desenvolvimento:

- O maior tempo do processo total de sincronização das fontes estava na extração das informações do banco de dados;
- A extração em blocos reduziria o número de consultas à base de dados acarretando aumento de desempenho;
- Seria possível paralelizar o processo de carga, dado que enquanto um bloco de dados estivesse sendo carregado do banco, o bloco anterior pudesse estar sendo processado pelo algoritmo de sincronização;

- A qualidade das consultas seria maior que as consultas geradas automaticamente por um framework de mapeamento objeto-relacional.

Com isso, obteve-se redução no tempo de carga das informações e número de consultas à base de dados, diminuindo o tempo total de extração das informações em aproximadamente 500% para cada 1000 currículos.

4.2.7 Mapeamento das unidades de negócio para a abstração comum do algoritmo de sincronização

O quarto passo do processo foi a adaptação das unidades de negócio para a abstração do algoritmo de sincronização, ou seja, fazer com que as unidades de negócio se tornassem unidades sincronizáveis.

Neste ponto, o algoritmo de sincronização desenvolvido anteriormente para o projeto FioLattes teve que ser atualizado para suportar a entrada de dados hierárquicos de forma eficiente, dado que um Currículo Lattes possui estrutura totalmente hierárquica. Desta atualização nasceu a interface *ParentObject* que define os métodos necessários para o processamento hierárquico pelo algoritmo de sincronização.

Após a atualização do algoritmo, todas as unidades de negócio do currículo foram adaptadas para implementar as interfaces de sincronização e prover as informações necessárias ao processo.

4.2.8 Persistência das informações na base do Portal

Como se optou pela não utilização de um framework objeto-relacional para o gerenciamento entre as unidades de negócio e o banco de dados relacional, foi necessário simular a geração do delta que é automaticamente disponibilizada por esse tipo de framework.

Para isso embutiu-se nas unidades de negócio um controle de mudança de estado similar ao controle encontrado em frameworks de mapeamento objeto-relacional, porém

pregando-se a especificidade da solução para o projeto, com o intuito de redução do cálculo da diferença e geração otimizada dos scripts de banco de dados.

4.2.9 Sincronização das fontes de dados

O processo de sincronização das fontes de dados é apresentado na figura 18.

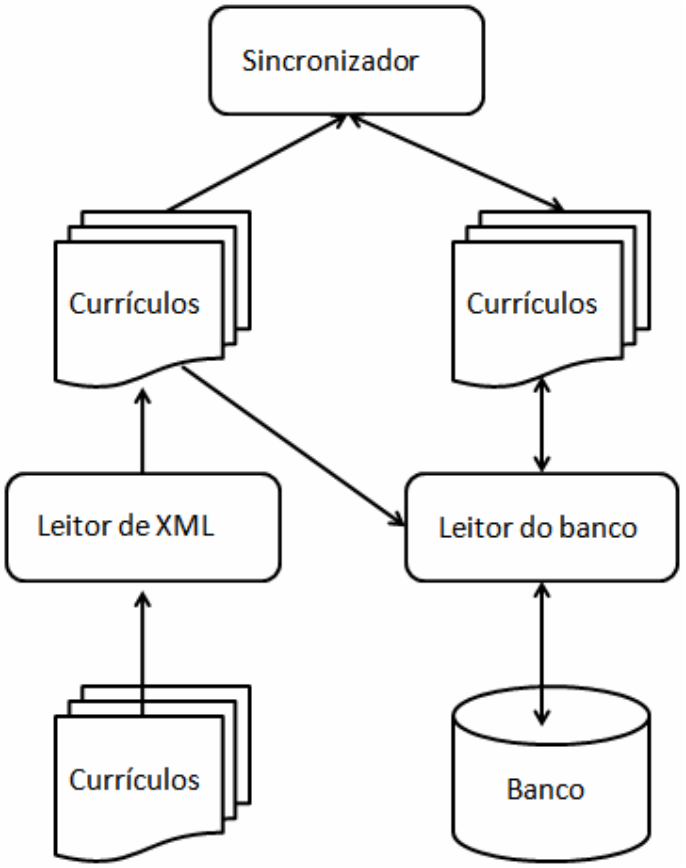

Figura 18 - O processo de sincronização dos currículos.

O processo de sincronização foi dividido de maneira a tirar proveito da diferença de tempo de execução das tarefas de leitura dos arquivos XML, leitura das informações em banco e sincronização dos dados. Como os tempos de execução são consideravelmente diferentes, o processo de sincronização ocupa menos de 1% do tempo total, utilizou-se o conceito de programação concorrente para distribuir cada tarefa para uma *thread* de execução em paralelo.

Na figura 18, os retângulos com cantos arredondados representam diferentes *threads*, e os currículos entre o sincronizador e os leitores representam buffers temporários

utilizados para armazenar e ler currículos. O fluxo do processo começa com a leitura dos arquivos XML pela *thread* de leitura de XML. Após o processamento do arquivo, esta *thread* armazena o currículo em um buffer temporário que é consumido pela *thread* de leitura do banco, com o objetivo de obter as chaves dos currículos a serem processados no banco. Após obter essas chaves, a *thread* de leitura do banco busca pelas informações correspondentes aos currículos selecionados e os armazena em um segundo buffer de currículos. A terceira *thread*, de sincronização, consome os currículos de ambos os buffers de armazenamento, sincronizando os currículos e removendo dos buffers os currículos processados. Após a sincronização as informações resultantes são processadas no banco de dados através do framework de mapeamento desenvolvido. Esse processo ocorre em paralelo utilizando a abordagem de produtor-consumidor.

4.2.10 Resultados obtidos

O projeto de sincronização dos currículos ainda está em desenvolvimento, por isso os resultados aqui apresentados são estimativas obtidas através dos primeiros testes de execução realizados sobre o atual estágio do novo sistema.

Hoje existe um sistema legado que realiza a mesma operação de sincronização de currículos com a base de dados do Portal Inovação. Pretende-se substituir este sistema com o desenvolvimento desse novo sistema.

A seguir são apresentados alguns gráficos comparativos entre o sistema legado e estimativas obtidas com o novo sistema de sincronização.

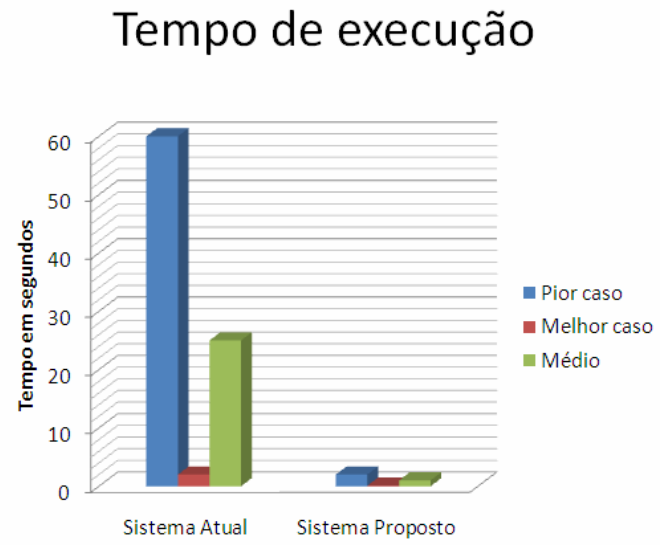

Figura 19 - Comparação do tempo de execução.

A figura 19 demonstra claramente a diferença do tempo de execução entre o sistema legado e o sistema novo. No gráfico, o pior caso é referente a um currículo cujo tamanho do arquivo XML é de 9 megabytes. O melhor caso tem um tamanho de aproximadamente 100 kbytes. A média do tamanho de um currículo é de aproximadamente 1,5 megabytes.

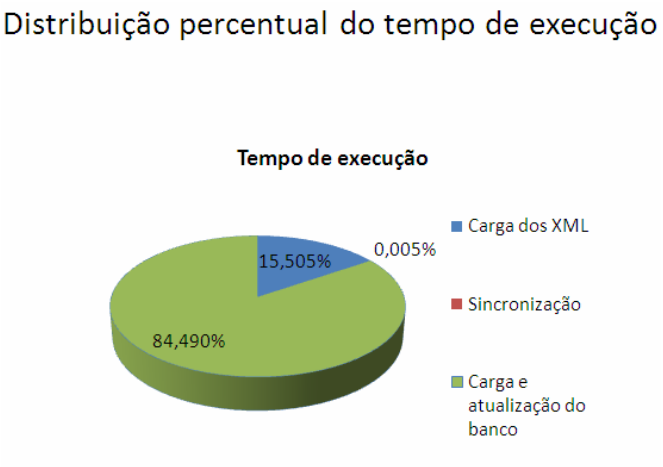

Figura 20 - Distribuição percentual do tempo de execução.

A figura 20 demonstra que o maior tempo do processo está na tarefa de carga e escrita das informações no banco de dados. Como as tarefas de sincronização e leitura

executam em paralelo e com tempos reduzidos, o tempo total do processo está diretamente relacionado ao tempo de carga e escrita das informações no banco de dados.

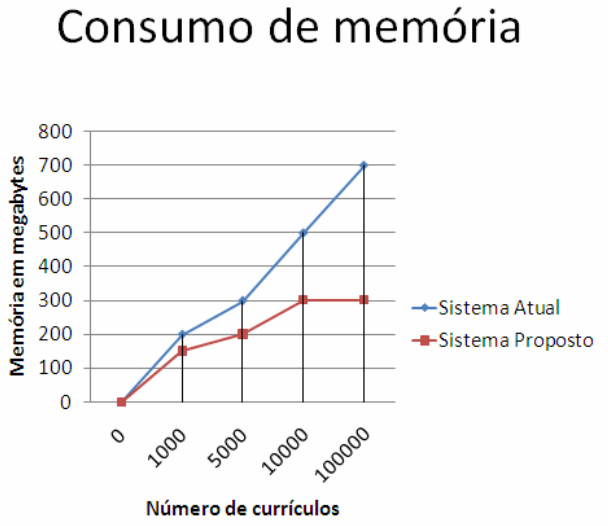

Figura 21 - Comparação do consumo de memória.

A figura 21 mostra um comparativo do consumo de memória entre os dois sistemas. Observe que durante o processo de sincronização o sistema legado tende a esgotar a memória disponível enquanto que o sistema novo tem o consumo de memória estabilizado na faixa dos 300 megabytes.

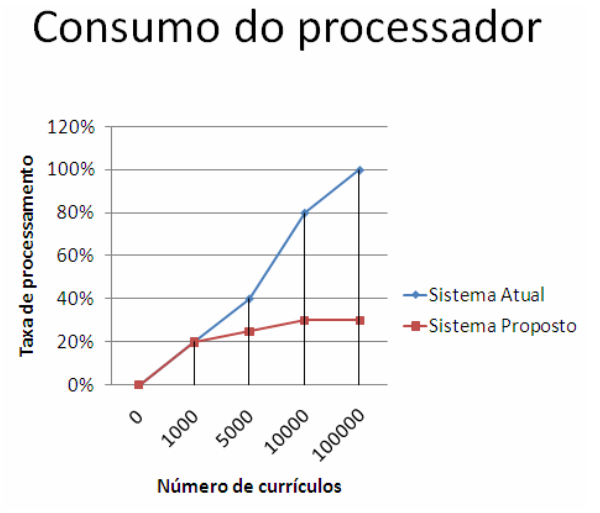

Figura 22 - Comparação do consumo do processador.

Como o maior tempo do processo de sincronização no sistema novo está atrelado a operações no banco de dados, e como o tempo de sincronização ocupa 0,005% do tempo total, o consumo médio de processamento tende a ser muito baixo quando comparado ao sistema legado.

Estas análises iniciais demonstram claramente a evolução do processo de sincronização com o desenvolvimento de um sistema segundo a arquitetura proposta neste trabalho.

5 CONCLUSÃO E TRABALHOS FUTUROS

Vários fatores fundamentais garantem que os desafios da integração de dados continuarão ocupando nossa sociedade por um longo tempo. O primeiro fator é social. A integração de dados é fundamental para motivar as pessoas a colaborar e compartilhar dados. O segundo fator é a complexidade da integração. Em muitos contextos de aplicação não é claro o que significa integrar os dados ou como conjuntos de dados combinados podem funcionar em conjunto.

A integração de informação precisa ser parte desta infra-estrutura e amadurecer ao ponto em que é considerada essencialmente garantida e desaparece em um plano de fundo como outras tecnologias ubíquas.

Para isso foi proposta uma lógica de sincronização de fontes heterogêneas de dados que reutiliza frameworks de mapeamento existentes. Desenhou-se uma arquitetura de software que facilita a integração de dados, minimiza o custo de sincronização e acelera o tempo de desenvolvimento.

A arquitetura aqui apresentada é uma alternativa a resolução de problemas de integração de informações coorporativas onde a heterogeneidade de dados é prevalente.

Como próximo passo planeja-se desenvolver ferramentas que auxiliem o processo de mapeamento das informações de negócio para a estrutura suportada pelo algoritmo de sincronização.

Também está previsto o estudo da aplicação da arquitetura com outros frameworks de mapeamento de dados e a comparação de desempenho com o framework desenvolvido no projeto do Portal Inovação.

Devido aos resultados significativos em relação ao tempo de execução do algoritmo de sincronização, pretende-se fazer uma experiência em aplicar o mesmo algoritmo na geração de deltas em aplicações web 2.0, embutindo o algoritmo no núcleo de classes destinadas ao cliente da aplicação.

Pretende-se ainda aperfeiçoar o suporte do algoritmo a estruturas de dados hierárquicas, devido ao fato que este tipo de representação é muito utilizado no contexto organizacional.

6 REFERÊNCIAS BIBLIOGRÁFICAS

Alon, H., R. Anand, *et al.* Data integration: the teenage years. Proceedings of the 32nd international conference on Very large data bases. Seoul, Korea: VLDB Endowment 2006.

Ana, G. M., M. Filippo, *et al.* Algorithmic detection of semantic similarity. Proceedings of the 14th international conference on World Wide Web. Chiba, Japan: ACM 2005.

Batini, C., M. Lenzerini, *et al.* A comparative analysis of methodologies for database schema integration. ACM Comput. Surv., v.18, n.4, p.323-364. 1986.

Fahad, M. A.-W., W. A. Gray, *et al.* Establishing an XML metadata klnowledge base to assist integration of structured and semi-structured databases. Proceedings of the 17th Australasian Database Conference - Volume 49. Hobart, Australia: Australian Computer Society, Inc. 2006.

Fakhar, L. e G. Muhammad Ahmad. Design of a simple and effective object-to-relational mapping technique. Proceedings of the 2007 ACM symposium on Applied computing. Seoul, Korea: ACM 2007.

Farshad, H. e G. Andreas. Resolving semantic heterogeneity in schema integration. Proceedings of the international conference on Formal Ontology in Information Systems - Volume 2001. Ogunquit, Maine, USA: ACM 2001.

Giuseppe De, G., L. Domenico, *et al.* Tackling inconsistencies in data integration through source preferences. Proceedings of the 2004 international workshop on Information quality in information systems. Paris, France: ACM 2004.

James, J. H., V. Kiem-Phong, *et al.* Delta algorithms: an empirical analysis. ACM Trans. Softw. Eng. Methodol., v.7, n.2, p.192-214. 1998.

Jin-Mao Wei; Shu-Qin Wang; Wei Zheng; Jing Wang; Jun-Ping You; Jie Zhang; Liu, D. On structural information similarity measurements. Granular Computing, 2006 IEEE International Conference, 10-12 May 2006, p.124-129. 2006.

Joyce, C. P. C. e S. D. S. Altigran. Finding similar identities among objects from multiple web sources. Proceedings of the 5th ACM international workshop on Web information and data management. New Orleans, Louisiana, USA: ACM 2003.

Mukesh, M. e B. Manish. New trends in information integration. Proceedings of the 2nd international conference on Ubiquitous information management and communication. Suwon, Korea: ACM 2008.

Patrick Connor, L. e P. H. Marc. An in-depth look at the architecture of an object/relational mapper. Proceedings of the 2007 ACM SIGMOD international conference on Management of data. Beijing, China: ACM 2007.

Patrick, P., P. Andrew, *et al.* Matching and integration across heterogeneous data sources. Proceedings of the 2006 international conference on Digital government research. San Diego, California: ACM 2006.

Pieter Van, Z., G. K. Derrick, *et al.* Comparing the performance of object databases and ORM tools. Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries. Somerset West, South Africa: South African Institute for Computer Scientists and Information Technologists 2006.

Raihan, A.-E., A. Archana, *et al.* diffX: an algorithm to detect changes in multi-version XML documents. Proceedings of the 2005 conference of the Centre for Advanced Studies on Collaborative research. Toranto, Ontario, Canada: IBM Press 2005.

Ralescu, A. M., M. Measuring Proximity between Heterogeneous Data. Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007. IEEE International, 23-26 July 2007. 2007.

Richard, V. S. Similarity and entropy. Proceedings of the 1964 19th ACM national conference: ACM 1964.

Robert, M. G. Entropy and information theory: Springer-Verlag New York, Inc. 1990. 332 p.

Shannon, C. E. A mathematical theory of communication. SIGMOBILE Mob. Comput. Commun. Rev., v.5, n.1, p.3-55. 2001.

Stefano, S., P. Christine, *et al.* Model independent assertions for integration of heterogeneous schemas. The VLDB Journal, v.1, n.1, p.81-126. 1992.

Sudarshan, S. C. e G.-M. Hector. Meaningful change detection in structured data. SIGMOD Rec., v.26, n.2, p.26-37. 1997.

Sven, H. Measuring the structural similarity of semistructured documents using entropy. Proceedings of the 33rd international conference on Very large data bases. Vienna, Austria: VLDB Endowment 2007.

Tridgell, A. Efficient Algorithms for Sorting and Synchronization. Australian National University, 2000.

Vipul, K. e S. Amit. Semantic and schematic similarities between database objects: a context-based approach. The VLDB Journal, v.5, n.4, p.276-304. 1996.

Wilhelm, H. Information system integration. Commun. ACM, v.43, n.6, p.32-38. 2000.

William, W. C. Data integration using similarity joins and a word-based information representation language. ACM Trans. Inf. Syst., v.18, n.3, p.288-321. 2000.

Youn, C. K., C.S. Data migration. Systems, Man and Cybernetics, 1992., IEEE International Conference, v.2, 18-21 October 1992, p.1255-1258. 1992.