Rodrigo Alves Costa

Inteligência Artificial na Integração de Banco de Dados Heterogêneos

Atena Editora

Ano 2019

Rodrigo Alves Costa

Inteligência Artificial na Integração de Banco de Dados Heterogêneos

Atena Editora
2019

2019 *by Atena Editora*

Editora Chefe: Profª Drª Antonella Carvalho de Oliveira
Diagramação e Edição de Arte: Lorena Prestes
Revisão: Os autores

Dados Internacionais de Catalogação na Publicação (CIP)
(eDOC BRASIL, Belo Horizonte/MG)

Costa, Rodrigo Alves
C837i Inteligência artificial na integração de banco de dados heterogêneos [recurso eletrônico] / Rodrigo Alves Costa. – Ponta Grossa (PR): Atena Editora, 2019.

Formato: PDF
Requisitos de sistema: Adobe Acrobat Reader.
Modo de acesso: World Wide Web.
Inclui bibliografia
ISBN 978-85-7247-149-7
DOI 10.22533/at.ed.497192602

1. Banco de dados. 2. Inteligência artificial. 3. Sistemas de computação. I. Título.

CDD 001.535

Elaborado por Maurício Amormino Júnior – CRB6/2422

www.atenaeditora.com.br

A Deus
A meus pais
A Karine

AGRADECIMENTOS

Ao Senhor Deus Pai, o Todo Poderoso: a Ele toda a honra, toda a glória, todo o louvor e toda a adoração. Ao Senhor Jesus, Aquele que é, que era e que há de ser: meu Mediador amado, perfeito e santo – o único santo. Ao Consolador Espírito Santo, por Sua doçura, amizade e poder, e por ter enchido por diversas vezes o meu espírito com uma unção sem limites. Maravilhoso é fazer parte desta família e poder ser chamado filho de Deus.

Ao meu pai, Hianto Costa *(in memoriam)*, sem o qual eu não entenderia o valor de família e a importância da educação em minha. Graças a ele, ao seu apoio, à sua inteligência e visão é que tudo isto se tornou possível.

À minha mãe amada, Maria Núbia, pois não haveria palavras suficientes para descrever a sua importância em minha vida e o amor que ela me dispensou.

À minha noiva, Karine, pela enorme compreensão e apoio incondicional. Por ser a companheira que Deus escolheu a dedo para o resto dos meus dias.

Às minhas irmãs, sempre presentes em minhas orações.

Ao professor, orientador deste trabalho e quem, acima de tudo, sempre considerarei um grande amigo, Fernando Fonseca de Souza, pelo apoio nestes cinco anos e pelos eternos ensinamentos.

Aos meus amigos, sempre companheiros com os quais contei em toda a caminhada.

A todos que contribuíram, direta ou indiretamente, para a realização deste trabalho e sonho, meus agradecimentos.

APRESENTAÇÃO

Integração de dados é o processo de combinar dados de diferentes fontes em uma visão unificada: a partir da ingestão, limpeza, mapeamento e transformação para um destino final, e finalmente transformação para tornar os dados mais “acionáveis” e valiosos para aqueles que os acessam. As organizações nos dias de hoje estabelecem iniciativas de integração de dados para analisar e agir de forma mais eficaz em seus dados, especialmente com a explosão da informação e novas tecnologias de nuvem e big data. A integração de dados é uma obrigação para as empresas modernas melhorarem a tomada de decisões estratégicas e aumentarem sua vantagem competitiva.

Não há uma abordagem universal para integração de dados. No entanto, as soluções de integração de dados geralmente envolvem alguns elementos comuns, incluindo uma rede de fontes de dados, um servidor mestre e clientes acessando dados nesse servidor mestre.

Em um processo típico de integração de dados, o cliente envia uma solicitação ao servidor mestre para dados. O servidor mestre, em seguida, inunda os dados necessários de fontes internas e externas. Os dados são extraídos das fontes e depois combinados em uma forma coesa e unificada. Isso é retornado ao cliente em uma forma utilizável e coesa.

Mesmo quando uma empresa está recebendo todos os dados necessários, esses dados geralmente residem em várias fontes de dados separadas ou de formato distintos. Por exemplo, para um caso de uso típico, os dados que podem ser combinados poderiam incluir dados de seus sistemas CRM, tráfego da Web, software de operações de marketing, aplicativos voltados ao cliente, sistemas de vendas e sucesso do cliente e até dados de parceiros para nomear alguns. As informações de todas essas fontes diferentes geralmente precisam ser reunidas para necessidades analíticas ou ações operacionais, e reunir todos eles não deve ser uma tarefa simples para engenheiros de dados ou desenvolvedores.

Em um caso de uso analítico típico, sem dados unificados, um único relatório geralmente envolve o registro em várias contas, em vários sites, com acesso a dados em aplicativos nativos, cópia dos dados, reformatação e a limpeza – tudo isso antes que a análise possa acontecer.

Realizar todas essas operações da maneira mais eficiente possível destaca a importância da integração de dados. Também mostra os principais benefícios de uma abordagem bem pensada à integração de dados:

1 I INTEGRAÇÃO DE DADOS MELHORA A COLABORAÇÃO E A UNIFICAÇÃO DE SISTEMAS

Os funcionários de todos os departamentos - e às vezes em locais físicos

diferentes - precisam cada vez mais de acesso aos dados da empresa para projetos compartilhados e individuais. A TI precisa de uma solução segura para entregar dados via acesso de autoatendimento em todas as linhas de negócios.

Além disso, os funcionários em quase todos os departamentos estão gerando e melhorando os dados das demais áreas de negócios. A integração de dados precisa ser colaborativa e unificada para melhorar a colaboração e a unificação em toda a organização.

2 I INTEGRAÇÃO DE DADOS ECONOMIZA TEMPO

Quando uma empresa toma medidas para integrar seus dados adequadamente, ela reduz significativamente o tempo necessário para preparar e analisar esses dados. A automação de visualizações unificadas elimina a necessidade de coletar dados manualmente e os funcionários não precisam mais criar conexões a partir do zero sempre que precisarem gerar um relatório ou criar um aplicativo.

Além disso, usar as ferramentas certas, em vez de codificar manualmente a integração, retorna ainda mais tempo (e recursos gerais) para a equipe de desenvolvimento.

Todo o tempo economizado nessas tarefas pode ser colocado em outros usos melhores, com mais horas destinadas a análise e execução para tornar uma organização mais produtiva e competitiva.

3 I INTEGRAÇÃO DE DADOS REDUZ ERROS (E RETRABALHO)

Há muito o que manter em dia quando se trata dos recursos de dados de uma empresa. Para coletar dados manualmente, os funcionários precisam conhecer todos os locais e contas que precisam explorar - e ter todos os softwares necessários instalados antes de começar - para garantir que seus conjuntos de dados estejam completos e precisos. Se um repositório de dados for adicionado e esse funcionário não estiver ciente, ele terá um conjunto de dados incompleto.

Além disso, sem uma solução de integração de dados que sincronize dados, os relatórios devem ser periodicamente refeitos para levar em conta quaisquer alterações. Com as atualizações automáticas, no entanto, os relatórios podem ser executados facilmente em tempo real, sempre que forem necessários.

4 I NTEGRAÇÃO DE DADOS FORNECE DADOS MAIS VALIOSOS

Os esforços de integração de dados realmente melhoram o valor dos dados de uma empresa ao longo do tempo. À medida que os dados são integrados em um sistema centralizado, problemas de qualidade são identificados e melhorias necessárias são implementadas, o que acaba resultando em dados mais precisos – a base para a

análise de qualidade.

É importante, também, compreender que integração de dados não é uma solução única para todos; a fórmula correta pode variar de acordo com várias necessidades de negócios. Aqui estão alguns casos de uso comuns para ferramentas de integração de dados:

BIG DATA

Volumes de dados podem ser altamente complexos e maciços em volume. Empresas como o Facebook e o Google, por exemplo, processam um fluxo ininterrupto de dados de bilhões de usuários. Esse nível de consumo de informações é comumente chamado de big data. À medida que mais empresas de Big Data surgem, mais dados ficam disponíveis para as empresas alavancarem. Isso significa que a necessidade de esforços sofisticados de integração de dados torna-se central para as operações de muitas organizações.

DATA WAREHOUSES

Iniciativas de integração de dados – particularmente entre grandes empresas – costumam ser usadas para criar data warehouses, que combinam várias fontes de dados em um banco de dados relacional. Os data warehouses permitem que os usuários executem consultas, compilem relatórios, gerem análises e recuperem dados em um formato consistente.

BUSINESS INTELLIGENCE (BI)

Ao fornecer uma visão unificada de dados de várias fontes, a integração de dados simplifica os processos de análise de *business intelligence* (BI). As organizações podem visualizar facilmente e compreender rapidamente os conjuntos de dados disponíveis para obter informações sobre o estado atual dos negócios. Com a integração de dados, os analistas podem compilar mais informações para uma avaliação mais precisa, sem serem sobrecarregados por grandes volumes.

Ao contrário da análise de negócios, o BI não usa análise preditiva para fazer projeções futuras; em vez disso, concentra-se em descrever o presente e o passado para auxiliar na tomada de decisões estratégicas. Esse uso de integração de dados é bem adequado ao data warehousing, em que as informações de visão geral de alto nível em um formato facilmente consumível se alinham bem.

ETL E INTEGRAÇÃO DE DADOS

Extrair, Transformar, Carregar, comumente conhecido como ETL *(extract,*

transform and load), é um processo dentro da integração de dados em que os dados são retirados do sistema de origem e entregues no repositório de dados. Esse é o processo contínuo em que o data warehousing se compromete a transformar várias fontes de dados em informações úteis e consistentes para inteligência de negócios e esforços analíticos.

Como se pode perceber, tomar várias fontes de dados e transformá-las em um todo unificado dentro de uma única estrutura é um desafio técnico em si. À medida que mais empresas criam soluções de integração de dados, elas são encarregadas de criar processos pré-desenvolvidos para mover consistentemente os dados para onde precisam ir. Embora isso proporcione economia de tempo e custos a curto prazo, a implementação pode ser dificultada por inúmeros obstáculos.

Por exemplo, as empresas geralmente sabem o que querem da integração de dados - a solução para um desafio específico. O que eles geralmente não pensam é o caminho que será necessário para chegar lá. Qualquer pessoa que implemente integração de dados deve entender que tipos de dados precisam ser coletados e analisados, de onde vêm esses dados, os sistemas que usarão os dados, que tipos de análise serão conduzidos e com que frequência os dados e relatórios precisarão ser atualizados.

Em muitos esforços de integração, também é necessário considerar dados de sistemas legados. Nesses casos, esses dados, no entanto, muitas vezes perdem marcadores, como horários e datas de atividades, que os sistemas mais modernos geralmente incluem.

Com respeito a dados das demandas de negócios mais recentes, os novos sistemas atuais estão gerando diferentes tipos de dados (como não estruturados ou em tempo real) de todos os tipos de fontes, como vídeos, dispositivos de IoT, sensores e nuvem. Descobrir como adaptar rapidamente sua infraestrutura de integração de dados para atender às demandas da integração de todos esses dados se torna fundamental para a sua empresa vencer, mas é extremamente difícil, pois o volume, a velocidade e o novo formato de dados impõem novos desafios.

Outro desafio técnico potencial reside na existência de dados externos, quando dados recolhidos de fontes externas podem não ser fornecidos com o mesmo nível de detalhe que as fontes internas, o que dificulta a análise com o mesmo rigor. Além disso, contratos em vigor com fornecedores externos podem dificultar o compartilhamento de dados por toda a organização.

Finalmente, quando um sistema de integração estiver em funcionamento, isso não significa que a tarefa está concluída. Cabe à equipe de dados manter os esforços de integração de dados compatíveis com as melhores práticas, bem como as demandas mais recentes da organização e das agências reguladoras.

A maioria dos desafios com respeito à integração de dados, no entanto, é mitigada pela plataforma correta de integração de dados. Existem soluções gratuitas de integração de dados de código aberto que ajudarão a iniciar um negócio.

Há várias maneiras de integrar dados que dependem do tamanho do negócio, da necessidade que se busca atender, e dos recursos disponíveis.

A mais simples delas, a integração manual de dados, é simplesmente o processo pelo qual um usuário individual coleta manualmente os dados necessários de várias fontes, acessando interfaces diretamente, limpando-as conforme necessário e combinando-as em um único depósito. Isso é altamente ineficiente e inconsistente, e faz pouco sentido para todos, exceto para a menor das organizações com recursos mínimos de dados.

A integração de dados via *middleware* é uma abordagem de integração em que um aplicativo de *middleware* atua como um mediador, ajudando a normalizar os dados e trazê-los para o conjunto de dados principais. (Pense em adaptadores para equipamentos eletrônicos antigos com pontos de conexão desatualizados). Os aplicativos legados geralmente não são compatíveis com os outros. O *middleware* entra em ação quando um sistema de integração de dados não consegue acessar os dados de um desses aplicativos por conta própria.

A integração baseada em aplicativos é uma abordagem à integração em que os aplicativos de software localizam, recuperam e integram dados. Durante a integração, o software deve tornar os dados de sistemas diferentes compatíveis entre si para que possam ser transmitidos de uma fonte para outra.

A integração de acesso uniforme é um tipo de integração de dados que se concentra na criação de um *front end* que faz com que os dados pareçam consistentes quando acessados de diferentes origens. Os dados, no entanto, são deixados dentro da fonte original. Usando esse método, sistemas de gerenciamento de banco de dados orientados a objeto podem ser usados para criar a aparência de uniformidade entre bancos de dados diferentes.

A integração de armazenamento comum é a abordagem usada com mais frequência para armazenamento na integração de dados. Uma cópia dos dados da fonte original é mantida no sistema integrado e processada para uma visualização unificada. Isso se opõe ao acesso uniforme, que deixa os dados na origem. A abordagem de armazenamento comum é o princípio subjacente por trás da solução tradicional de *data warehousing.*

As ferramentas de integração de dados têm o potencial de simplificar bastante esse processo. Este trabalho visa descrever uma abordagem genérica para construção de ferramentas de integração de dados baseadas em agentes inteligentes. Com base nas características de tais ferramentas norteamos a proposta. Tais características poderíamos citar a seguir:

- Muitos conectores: Existem muitos sistemas e aplicativos no mundo; Quanto mais conectores pré-construídos sua ferramenta de Integração de Dados, mais tempo sua equipe economizará.

- Código aberto: As arquiteturas de código aberto geralmente fornecem mais flexibilidade, ajudando a evitar o bloqueio de fornecedores.

- Portabilidade: É importante que as empresas migrem cada vez mais para os modelos de nuvem híbrida, para poder criar suas integrações de dados uma vez e executá-las em qualquer lugar.

- Facilidade de uso: As ferramentas de integração de dados devem ser fáceis de aprender e fáceis de usar com uma interface GUI para simplificar a visualização de seus pipelines de dados.

- Modelo de preço transparente: Seu provedor de ferramenta de integração de dados não deve lhe entregar para aumentar o número de conectores ou volumes de dados.

- Compatibilidade de nuvem: Sua ferramenta de integração de dados deve funcionar de maneira nativa em um único ambiente de nuvem híbrida, nuvem múltipla ou nuvem.

Está se tornando cada vez mais proeminente que as organizações necessitam acompanhar as demandas modernas nos negócios e as implicações trazidas por essa nova era de negócios das empresas com respeito às informações, seus sistemas e, consequentemente, dados organizacionais. Entender as necessidades que a integração de dados vem atender, os métodos pelos quais ela é realizada e os obstáculos que surgem na implementação devem fornecer um amplo avanço na descoberta da melhor opção de integração de dados para qualquer empresa ou organização.

Neste sentido, a nossa satisfação se encontra na finalização de uma obra que venha ao encontro dos anseios de uma comunidade acadêmica e de mercado para o esclarecimento deste tema tão complexo mas tão importante para esse nicho de conhecimento.

- O autor

SUMÁRIO

RESUMO

A importância da criação de sistemas de integração de dados está relacionada com o desenvolvimento de sistemas capazes de sobrepor heterogeneidades semânticas entre diferentes fontes de dados, de modo que as necessidades das organizações interessadas no nível de acesso a dados integrados sejam atendidas. Diversas metodologias para sistemas de integração já foram propostas e, em todas elas, observa-se a necessidade de tradução de fontes locais, heterogêneas, para um formato comum, que será integrado. Este livro vem propor o desenvolvimento de um módulo capaz de realizar traduções entre diversas fontes, garantindo que a camada de software responsável pela tradução em um sistema de integração de dados tenha uma autonomia maior (através de um mecanismo de inteligência intrínseco) em relação ao resto do sistema, uma vez que realiza traduções a partir de uma base de conhecimento e decide relacionamentos entre as fontes de dados, metadados e esquemas de implementação.

PALAVRAS-CHAVE: Integração de dados, tradução, agentes, modelo de dados comum.

ABSTRACT

The importance of the creation of data integration systems is related to the development of systems that are capable of overcoming the semantics heterogeneities between different data sources, such that the needs of the organizations interested in the integrated data access level is achieved. Various methodologies for integration systems have already been proposed and, in all of them, it is possible to perceive the need for translation among heterogeneous local sources to a common format, to be integrated. This book proposes the development of a module that is capable of achieving wrapping among several data sources, ensuring that the software layer responsible for the wrapping phase in a data integration system has a major autonomy (throughout an inherent intelligence mechanism) in respect to the remainder of the system, once it carries out translations from a knowledge base and decides relationships between the data sources and implementation schemes.

KEYWORDS: Data integration, wrapping, agents, common data model.

CAPÍTULO 1

INTRODUÇÃO

Neste capítulo, é apresentada uma introdução a este livro, caracterizada por uma contextualização para o tópico abordado, motivações para abordá-lo em projetos acadêmicos e a estruturação do documento como um todo.

Na seção 1.1, é apresentado o contexto deste livro. Na seção 1.2, são apresentadas as motivações que levaram à sua escrita e os seus objetivos e, na seção 1.3, é exposta a sua estruturação.

1.1 Contexto

O acesso às diversas fontes de dados, tanto convencionais como semi-estruturadas, pertencentes a órgãos, empresas públicas e privadas, universidades e centros de pesquisa é, atualmente, necessário para o funcionamento e estruturação de tais organismos. Devido ao advento de um mecanismo de disponibilização de informação como a *World Wide Web* (WWW) e ao fato de a possibilidade dos dados, organizados de maneira estruturada ou semi-estruturada, estarem em constante atualização e serem utilizados por usuários espalhados por diversas partes do mundo, apareceram necessidades de garantir que:

a. O acesso a esses dados ocorra de forma eficiente;

b. As atualizações sobre esses dados ocorram de forma consistente.

Devido às heterogeneidades, tanto semânticas como de localização, e à autonomia destas fontes de dados, observou-se que, no contexto da WWW, acesso eficiente e consistência de atualizações sobre dados só podem ser garantidos através de um acesso integrado às fontes. Uma das formas de acesso integrado a fontes de dados heterogêneas acontece através de sistemas de gerenciamento de bancos de dados federados (SGBDF) [Sheth; Larson, 1990]: uma coleção de sistemas de bancos de dados autônomos e heterogêneos que fornecem um esquema integrado de seus bancos de dados individuais.

Existem diversas metodologias e arquiteturas propostas para que sistemas de integração e distribuídos, se apliquem às reais necessidades das empresas e organizações interessadas pelo nível de acesso de dados que eles possibilitam. Todas estas arquiteturas, entretanto, têm um ponto em comum: a necessidade de tradução de fontes locais, diferentes e heterogêneas, para um formato comum, o chamado Modelo de Dados Comum, MDC [Inmon, 1997].

Este tipo de especificação é necessário porque um mecanismo de captura de relacionamentos entre conceitos semelhantes que são representados de maneiras diferentes nas fontes locais, tanto em seus modelos quanto em dados propriamente ditos, é usado para integrar os dados: a identificação de conceitos similares entre diversos esquemas a serem integrados. Fazer isso não é uma tarefa fácil. Diversas são as metodologias propostas e todas apresentam inúmeras desvantagens [Batini; Lenzerini; Navathe, 1986] [Inmon, 1997] [Samos et al, 1998].

Este trabalho vem propor que a camada de software responsável pela tradução *(wrapping)* [Ciferri; Souza, 2000] entre fontes tenha uma autonomia maior (e desta forma um mecanismo de inteligência intrínseco) em relação às outras camadas de um sistema de integração, sendo capaz de acessar uma base de conhecimento para realizar traduções, decidir que relacionamentos devem ser estabelecidos entre fontes, seus esquemas e metadados do sistema em si, armazenar os resultados de traduções em lugares específicos e promover melhorias na performance do processo mantendo um histórico de traduções anteriores.

Verifica-se, portanto, a necessidade do desenvolvimento de um módulo para sistemas de integração de dados capaz de realizar traduções entre diversas fontes que disponibilize uma interface para implementação de camadas superiores do sistema.

1.2 Motivação e Objetivos

Desenvolver abordagens para sistemas de integração de dados está relacionado com avanços no sentido de desenvolvimento de sistemas extremamente confiáveis, capazes de sobrepor heterogeneidades semânticas das fontes de dados, além de sistemas extensíveis, para que atualizações nos tipos de informação oferecidos nas fontes não impliquem, como em grande parte de sistemas distribuídos, em inutilização de parte ou necessidade de re-implementação do sistema.

Para prover uma visão integrada de múltiplas fontes de dados, é necessária uma camada de software na arquitetura da aplicação. Esta camada deve conter um esquema para a visão de integração que possibilita a execução de consultas e atualizações [Pedersen, 2000]. A literatura conhece esta camada apenas como um leitor de metadados (classes de software que inferem metadados a partir de dados ou modelos) entre fontes [Pedersen, 2000], mas a proposta deste trabalho é prover uma maior autonomia para a mesma, de modo que ela passe a ser vista como uma camada de inferência com poder de tomada de decisão na integração de dados, já que

utiliza um sistema de inteligência, que pode inferir novos fatos a partir de uma base de conhecimento.

O estudo deste tipo de metodologia para mapeamento implica no estudo de diversas outras atividades nos sistemas de integração de dados. Restringindo-se às atividades que poderão ser realizadas pela camada de tradução por meio de agentes inteligentes de tradução, será necessário abordar diversas áreas do conhecimento, cujo estudo contribuirá para a resolução de outros problemas encontrados em sistemas deste tipo (como a análise e integração de esquemas e metadados). Dentre estas áreas, pode-se destacar alguns pontos, tais quais:

- Integração dos esquemas locais para identificação das informações relacionadas;
- Tradução dos esquemas locais para um MDC;
- Geração de mediadores;
- Manutenção do sistema de integração, que pode envolver a manutenção de tradutores e mediadores.

O grande desafio, ao se utilizar uma arquitetura como esta, é realizar um processo totalmente automático de integração de esquemas, em uma abordagem *bottom-up* de distribuição de dados, que esconda as diversidades do sistema do usuário e trabalhe com fontes estruturadas ou semi-estruturadas. É isso que o trabalho propõe fazer por meio do MDC: o aumento do nível de automatização por meio de técnicas de inteligência artificial através da utilização de agentes, por exemplo.

O objetivo deste livro é, portanto, propor um módulo federado fortemente acoplado, permitindo que as fontes de dados mantenham sua autonomia e suas restrições sejam impostas sobre o esquema da federação. Então, cabe como uma visão futura (apresentada na seção 6), basear esta abordagem em um processo de integração semântica, utilizando ontologias [Costa; Campos; Souza, 2002], dicionários de dados, propriedades estruturais, hierarquias e valores derivados dos modelos e metamodelos.

1.3 Estrutura do Trabalho

Além deste capítulo introdutório de contextualização e objetivos, fazem parte deste livro ainda os seguintes capítulos:

- Capítulo 2 – Bancos de Dados Distribuídos, onde são descritos conceitos básicos no universo de bancos de dados e internet, incluindo as aplicabilidades possíveis para um módulo como o proposto por esta abordagem em sistemas de integração de dados;
- Capítulo 3 – Fundamentação Conceitual, onde é feita a proposta de espe-

cificação das unidades básicas do sistema sob a perspectiva de diversas ferramentas já consagradas no meio acadêmico, focalizando conceitos dos componentes da abordagem de mapeamento aqui proposta;

- Capítulo 4 – Metodologia, onde se discutem as abordagens para a codificação da ferramenta, módulos que devem ser utilizados nesta codificação e aplicabilidade das técnicas de implementação dos agentes *threads* nas diversas situações na sociedade de agentes;
- Capítulo 5 – Abordagem de Mapeamento, quando a solução proposta pelo trabalho para o problema do mapeamento de fontes heterogêneas, segundo regras pertencentes às bases de conhecimento dos agentes participantes da sociedade de agentes, é apresentada e resultados arquiteturais e aspectos de implementação são discutidos;
- Capítulo 6 – Conclusões e Trabalhos futuros, onde a abordagem é finalizada e uma análise dos conceitos utilizados e da tecnologia abordada é apresentada. Além disso, propõem-se aplicações da abordagem apresentada em diversas outras arquiteturas e abordagens.

CAPÍTULO 2

BASES DE DADOS HETEROGÊNEAS

Neste capítulo 2, é realizada uma exposição da tecnologia – seus conceitos e a implicação destes no módulo proposto – da qual a abordagem faz parte. É de fundamental importância, para a proposição de um módulo a ser usado em aplicações de integração de dados, que o contexto de bancos de dados distribuídos seja definido.

Na seção 2.1, são introduzidos conceitos de bancos de dados distribuídos e de como tais sistemas são utilizados nas organizações em aplicações cotidianas. Na seção 2.2, de forma geral, é apresentada uma visão para a interface web deste tipo de sistema, além de se discorrer acerca de uma possível visão de comunicação entre diversos pontos e servidores participantes de uma integração. Nas seções seguintes, expõem-se conceitos fundamentais para esta abordagem: integração de dados (seção 2.3), modelo de dados canônico (seção 2.4) e metaconhecimento e inferência, para as traduções necessárias (seção 2.5). Finalmente, na seção 2.6, são apresentados paradigmas para desenvolvimento de sistemas baseados em agentes.

2.1 Sistemas de Bancos de Dados Distribuídos

Uma rede de comunicação com a necessidade de suportar aplicações locais, em cada computador desta rede, além de suportar aplicações globais nas quais estejam envolvidos mais de um computador, acarretando, assim, em uma coleção de dados distribuída por diferentes computadores, em diferentes locais, é o que constitui um sistema de bancos de dados distribuídos (SBDD [Kim, 1995]).

SBDD são extremamente úteis para a tecnologia de integração de dados, devido ao fato de:

- Concederem autonomia para aplicações locais;
- Aumentarem confiabilidade quanto à disponibilidade de fontes de dados;
- Acarretarem melhorias no desempenho dos sistemas;
- Resultarem em flexibilidade das aplicações, o que implica economia.

O advento de sistemas distribuídos ocasionou uma rápida utilização de sistemas

baseados nesta tecnologia por vendedores e empresas diferentes, com suas tecnologias baseadas em diferentes modelos. Entretanto, informação para tomada de decisão deve sempre estar disponível, independente de seu formato. Essa disponibilidade independente de formato caracteriza um sistema de integração de dados operacionais.

Para possibilitar essa disponibilidade, precisa-se de um ambiente que atenda às necessidades informacionais através da integração e consolidação dos dados disponíveis em diferentes conjuntos de fontes de dados. Esse ambiente é o ambiente de integração de dados.

Historicamente, observa-se que as organizações são cada vez mais pressionadas a obter informações de forma mais rápida e confiável, com o objetivo de melhorar o processo de tomada de decisões. Nas últimas décadas, a tecnologia da informação evoluiu consideravelmente, desde os primeiros computadores centrais, ou *mainframes,* até os atuais sistemas distribuídos. Essa visão moderna busca obter vantagens, principalmente em três aspectos:

1. Confiabilidade ou tolerância à falhas;
2. Disponibilidade;
3. Custo acessível.

Um importante componente dos sistemas distribuídos é o SBDD, pois ele é o responsável pelo armazenamento e recuperação das informações, de forma transparente para os clientes. Um SBDD deve ter como características:

1. Autonomia para transações locais;
2. Independência em relação a um site central;
3. Tolerância à falhas (confiabilidade);
4. Independência de localização;
5. Independência de fragmentação (se os dados estiverem fragmentados em vários sites, isso deve ser imperceptível ao cliente);
6. Independência de replicação;
7. Processamento distribuído das consultas;
8. Gerenciamento das transações distribuídas *(two-phase commit, lock,* por exemplo);
9. Independência de *hardware;*
10. Independência de Sistema Operacional;
11. Independência da rede;
12. Independência do Sistema de Gerenciamento de Banco de Dados.

Como se pode facilmente notar, é extremamente complexa a construção de SBDD, considerando-se, por exemplo, o gerenciamento de tal mistura de plataformas, sistemas operacionais, redes e SGBD.

2.2 Interface WEB

Nos SBDD, normalmente há a necessidade de se prover interação com o usuário. O acesso do usuário pode ser feito via qualquer browser, como, por exemplo, o Internet Explorer. A camada de aplicação WEB dos SBDD tem como objetivo prover uma interface que permita aos usuários interagir e acompanhar o desenvolvimento de seu sistema. Para tanto, é necessária uma visão integrada das fontes de dados componentes do sistema, por meio de um repositório central, apresentando, assim, o esquema virtual atual em que se encontra a federação de dados em um determinado instante.

O repositório central é tal que permite o armazenamento de fontes de dados, de códigos-fonte do sistema e de modelos em diversos formatos. Diversas fontes são expostas, podendo-se acessá-las individualmente, obtendo, assim, uma visão atualizada das mesmas, e também realizar integração entre elas. O mesmo repositório permite gerar o esquema atualizado de forma automática, por meio de solicitações ao servidor.

Torna-se possível o acompanhamento da evolução do sistema e o trabalho realizado por outros usuários através do esquema integrado. E a informação pode estar armazenada tanto em códigos-fonte, modelos ou fontes de dados.

Para geração dinâmica de páginas HTML, necessárias para interface com o usuário (para visualização e análise do estado atual do desenvolvimento do SBDF, por exemplo), são normalmente utilizados Servlets ou serviços dinâmicos para geração de conteúdo.

De forma genérica, a comunicação dos pontos WEB de um SBDD estão alocados da seguinte forma: tem-se um serviço no servidor central que, dados os endereços fornecidos das fontes de dados, estabelece conexões com serviços alocados em cada fonte de dados. Observe a Figura 2.1. É a partir do estabelecimento de conexões que os serviços remotos concluem esquemas das fontes de dados locais e depois transformam tais conclusões em um MDC, tornando possível, assim, aos mesmos, a participação na federação. Ao serviço alocado no servidor central, o arquivo no formato MDC resultante deste processo é retornado. Este serviço repassa esta informação para outro serviço, o serviço solicitante da informação. O serviço central ainda é responsável pelo encaminhamento das requisições de consulta para o sistema de gerenciamento da fonte de dados local.

A reunião dos dados em uma fonte de dados integrada é responsabilidade do serviço solicitante. Para tanto, há o estabelecimento de conexões com o repositório do esquema integrado. As fontes de dados resultantes do mapeamento são copiadas em um MDC para o repositório central do SBDD, que está conectado com as fontes de dados originais.

Este mesmo processo ocorre em diversos SBDD conhecidos, incluindo sistemas

de Data Warehouse (DW). Entretanto, esta abordagem propõe um módulo para ser utilizado em sistemas distribuídos que, além de fontes de dados, integrem metadados, modelos, metamodelos, esquemas de representação de dados e metadados e códigos-fonte de linguagens de programação utilizadas para construir os esquemas.

2.3 Integração de Dados

Sistemas de integração de dados, em geral, visam oferecer aos usuários uma interface uniforme de acesso a diferentes fontes de dados. O que ocorre em um mecanismo de consulta [Salgado; Lóscio, 2000] de um sistema integrado é: o usuário determina “o que” ele deseja saber e o sistema determina “onde” tal informação pode ser encontrada. Cabe à interface gráfica, possivelmente sendo parte do próprio sistema de integração, apresentar as respostas para as consultas do usuário.

De forma ilustrativa, dada uma determinada consulta partindo de certa fonte (possivelmente clientes ou usuários), determinar a melhor forma de exposição dos dados resultantes, considerando aspectos tais quais tipos de dados envolvidos na pesquisa, acessibilidade a tais dados e sua localização física, tempo de consulta e exibição de resultados (lag, o retardo médio necessário para garantir a comunicação correta e confiável entre pontos de uma rede) ou mesmo probabilidade de sucesso nas tentativas de acesso aos dados é o objetivo básico dos sistemas de integração de dados.

Considerando isso, esta abordagem é desenvolvida para uma metodologia para construção de uma federação de sistemas de bancos de dados heterogêneos, distribuídos e autônomos, fortemente acoplada, baseadas nas arquiteturas de 5 níveis de Sheth e Larson [Sheth; Larson, 1990] e Samos et al [Samos et al, 1998].

Para se prover uma abordagem satisfatória para a integração de esquemas deve-se identificar e especificar relacionamentos entre esquemas locais e esquemas globais, que são, de fato, a visão do engenheiro de software, usuário de uma ferramenta de integração de esquemas. Diversos trabalhos propõem metodologias para integração de esquemas, como [Navathe; Savasere, 1996] e [Spaccapietra; Parent, 1994].

Estes sistemas precisam identificar conceitos similares entre diversos esquemas a serem integrados. Percebendo esta necessidade, este trabalho discute uma abordagem para a disponibilização de um módulo capaz de tratar a especificação de assertivas de correspondência nas camadas mais baixas da arquitetura, no nível de mapeamento de fontes locais para os modelos de dados utilizados nas operações subsequentes de integração, para ser utilizado em sistemas de integração de esquemas.

2.3.1 Arquitetura Básica

Esta seção apresenta uma arquitetura básica para sistemas de integração de

dados de modo que se torne clara a necessidade de se fazer o mapeamento de fontes locais para um modelo de dados que o sistema de integração seja capaz de entender. Este mapeamento, como será exposto a seguir, submete-se a determinadas regras para que as expectativas geradas junto ao usuário na integração de dados sejam atendidas.

A arquitetura proposta neste documento é baseada na arquitetura de referência para sistemas de gerenciamento de banco de dados fortemente acoplados proposto por Sheth e Larson [Sheth; Larson, 1990] e na arquitetura de Samos et al [Samos et al, 1998].

Aqui é feita uma distinção entre os módulos funcionais do SGBDF e os esquemas da arquitetura, sendo eles: esquema local, esquema componente, esquema exportado, esquema federado, esquema aplicação e esquema externo.

A seguir, o diagrama ilustrdo na figura 2.1 descreve a arquitetura aqui proposta. A escolha desta arquitetura tem por objetivo inserir o processo de modelagem dentro do contexto de SGBDF [Costa; Campos; Souza, 2002]:

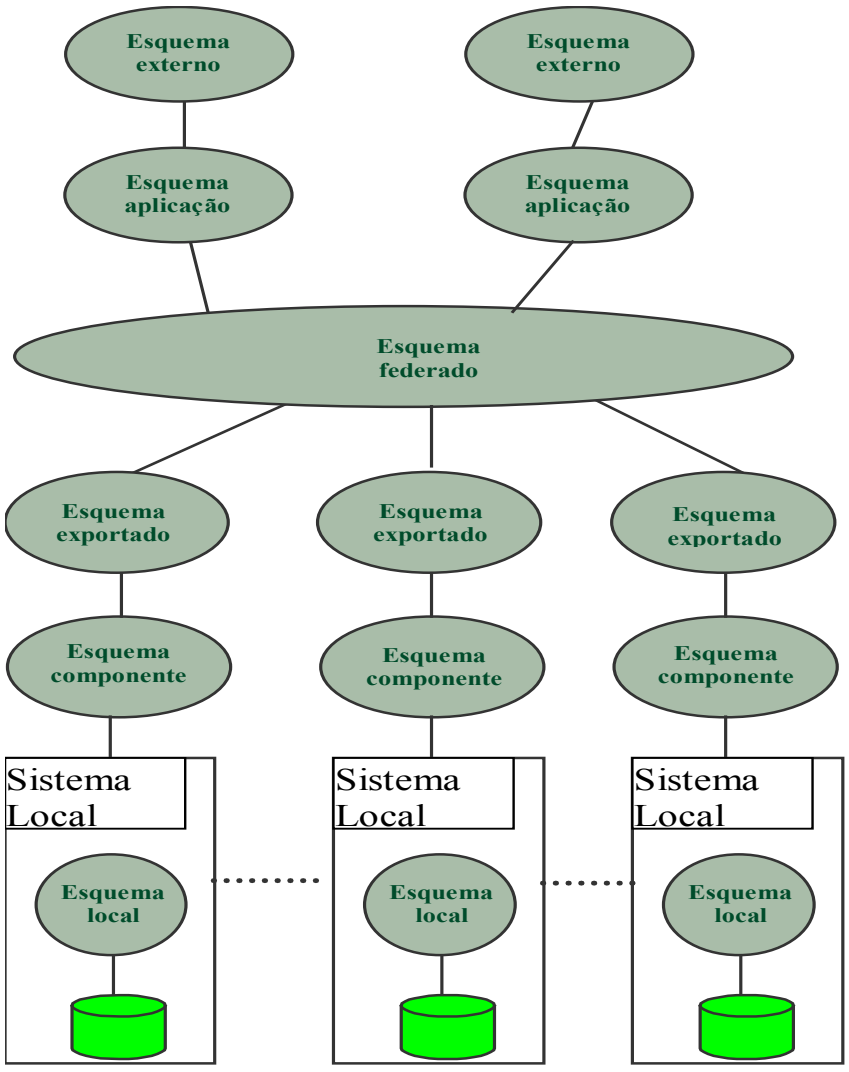

Figura 2.1: Arquitetura do esquema federado

A partir desta arquitetura, e tomando posse dos padrões e ferramentas a serem abordadas na seção 4, pode-se descrever a maneira através da qual a ferramenta de integração de dados (seguindo a arquitetura), que utilizará o módulo no proposto na abordagem, deve tratar o processo de integração.

Para a junção de dados de fontes heterogêneas, o sistema poderia utilizar os seguintes componentes: módulo global, processamento de consultas, módulo de interface, controle de acesso, módulo de comunicação, controle de integridade, gerenciamento de transações e tradutores. Tais componentes são detalhadamente descritos em [Costa; Campos; Souza, 2002]. Uma arquitetura para a visualização destes componentes no sistema é exposta na Figura 2.2 [Costa; Campos; Souza, 2003]*:*

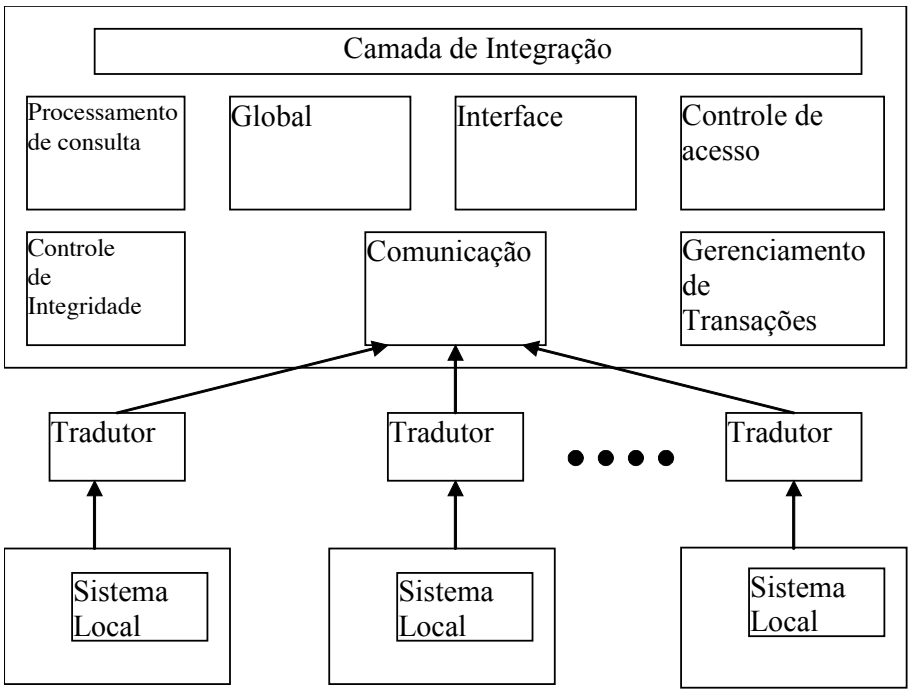

Figura 2.2: Módulos componentes da arquitetura

Esta arquitetura é claramente direcionada para a integração entre diferentes pontos. A seção 2.1 explica como seria a visão Web de um sistema como este.

2.3.2 Integração de Esquemas

Integração de esquemas é uma atividade que deve interligar os esquemas de esquemas pré-existentes em um esquema único, global, unificado. De acordo com Batini [Batini; Lenzerini; Navathe, 1986], a integração de esquemas de bancos de dados deve ocorrer em dois contextos:

a. Integração de visões – para produzir uma descrição global, conceitual, do banco de dados proposto;

b. Integração dos dados – para produzir o esquema global de uma coleção de dados. Este esquema global é uma visão virtual de todos os dados que foram integrados através das fontes participantes da integração.

O objetivo da fase de integração de esquemas (entre diferentes tipos de modelos, que podem expressar desde modelos do tipo Entidade Relacionamento (ER) [Heuser, 1998] até o processo de modelagem de sistemas em si), deve ser a identificação de relacionamentos entre os esquemas que serão integrados e um possível esquema de mediação, uma espécie de fase intermediária dos esquemas, como a Figura 2.3 retrata:

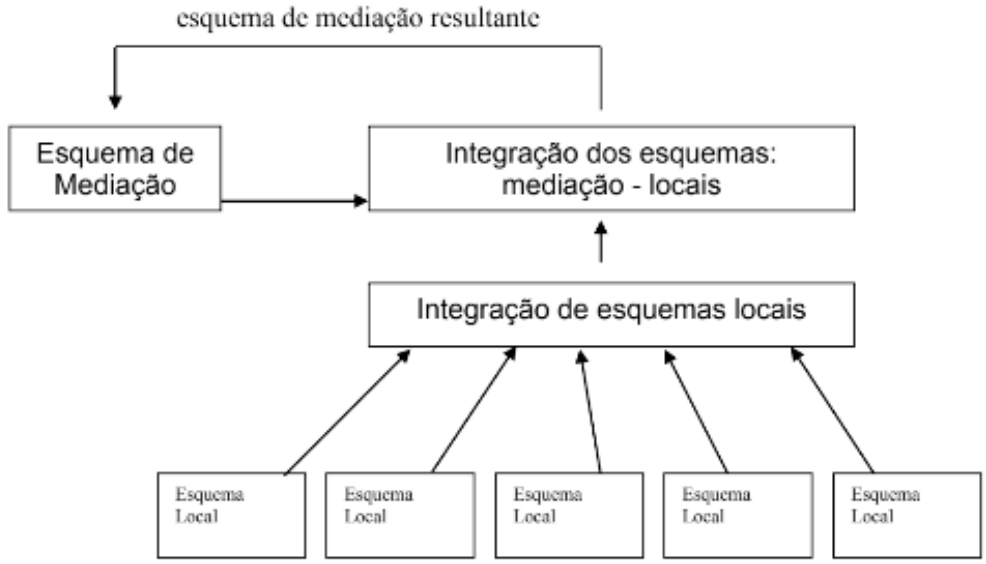

Figura 2.3: Arquitetura geral para integração de esquemas

Figura 2.3: Arquitetura geral para integração de esquemas

O processo de integração de esquemas consiste, basicamente, na especificação de assertivas de correspondência que relacionem os esquemas envolvidos na integração. Diversas ferramentas já foram propostas e desenvolvidas para ajudar no processo da descoberta das assertivas de correspondência, como [Navathe; Savasere, 1996] e [Spaccapietra; Parent, 1994]. O trabalho apresentado neste documento propõe a criação de um motor de inferência por meio do qual haja a definição de regras que trabalhem como estas assertivas de correspondência.

2.4 Modelo de Dados Canônico

Para a transformação dos diferentes modelos das fontes de dados para o esquema componente e integração desses diferentes esquemas componentes em um

esquema federado, deve ser utilizado um modelo de dados canônico [Sheth; Larson, 1990].

Diversas linguagens em toda a literatura são propostas como possíveis MDC [Ciferri; Souza, 2000] [Inmon, 1997]. Para a escolha do MDC a ser utilizado no na abordagem, devem ser levados em consideração diversos fatores, tais quais [Costa; Campos; Souza, 2002]:

- Suporte a modelos conceituais – capacidade do MDC de representar modelos como UML [Booch; Jacobson; Rumbaugh, 2000] e ER [Heuser, 1998];
- Poder de expressividade – o MDC deve ser capaz de representar um grau de expressividade tal a suportar conceitos de orientação a objeto e conceitualização de modelos de bancos de dados;
- Fonte estruturada – o formato dos itens listados no MDC devem ser ajustados ao esquema utilizado no sistema;
- Fonte semi-estruturada – a informação associada ao esquema é contida dentro dos próprios dados, como em HTML [Bryan, 1998], XML [Bray; Paoli; Sperberg-McQueen], SGML [Bryan, 1998];
- Extensibilidade – possibilidade de estruturação extensível do trabalho a diversas arquiteturas diferentes.

Um estudo comparativo de diversos MDC é apresentado na seção 3.3. O MDC escolhido neste trabalho é apresentado na seção 3.4

2.5 Metaconhecimento e Inferência

Especificar assertivas de correspondência em esquemas é uma das fases primordiais para a integração. Este tipo de especificação é necessário porque um mecanismo de captura de relacionamentos entre conceitos semelhantes que são representados de maneiras diferentes é usado na fase posterior na integração. Fazer isso não é uma tarefa fácil. As desvantagens em metodologias propostas são inúmeras.

O procedimento de análise de metaconhecimento e posterior inferência é a principal fase em um módulo mapeador de fontes de dados, uma vez que esta camada tem como sua principal atividade a elaboração de uma interface para a especificação de tais assertivas no processo de mapeamento de fontes, que será utilizada por usuários interessados em realizar integração das fontes disponíveis e que produzirá arquivos no formato MDC escolhido que podem ser lidos por ferramentas de integração que se adaptem a estes módulos.

Todo o processo de interpretação dos dados e das assertivas de correspondência é feito por agentes que interagem entre si. No capítulo 3, a arquitetura de todo um sistema de integração de dados baseada em uma sociedade de agentes é apresentada, muito embora a abordagem se limite a discutir e propor o módulo responsável pelo

mapeamento inteligente de fontes para o MDC.

Uma ferramenta especializada no processo de inferência é o JEOPS [Costa; Campos; Souza, 2002] e o seu uso em conjunto com ferramentas de interpretação de estado de sistema para a construção da federação por parte da sociedade de agentes (como JADE [Costa; Campos; Souza, 2003]) é adequado para uma maior automatização do processo de inferência.

2.6 Sistemas Baseados em Agentes

Desde a década de 80, o interesse por sistemas baseado em agentes e sistemas de sociedade de agentes vem crescendo e fez o tópico se tornar o que hoje é uma das principais atividades que dizem respeito a pesquisa e desenvolvimento em tecnologia de informação, de forma geral [Wooldridge, 1995], [Wooldridge; Jennings, 1995]. Há diversas razões para o crescimento do interesse neste tipo de tecnologia, mas certamente uma das mais importantes é o conceito que um agente é um sistema autônomo, capaz de interagir com outros agentes/sistemas para satisfazer objetivos de desenvolvimento. Da mesma forma que um sistema pode ser entendido como sendo composto essencialmente de diversos objetos passivos que possuem estados e por meio dos quais é possível se realizar operações, pode-se também entender muitos outros como sendo feitos de interação: agentes semi-autônomos que oferecem serviços.

Um sistema baseado em agentes pode ser definido como um sistema no qual agentes têm a responsabilidade de interagir para a realização de atividades que produzam saídas condizentes com os seus objetivos em comum [Wooldridge, 1995]. E um agente define-se como sendo um sistema que segue as seguintes características [Wooldridge; Jennings, 1995]:

- Autonomia: agentes encapsulam algum estado, não acessível a outros agentes, e tomam decisões, baseados neste estado, sobre o que fazer, sem a intervenção direta de usuários ou outros sistemas;
- Reatividade: os agentes estão situados em um ambiente, que pode ser um mundo físico, um usuário (através de uma interface gráfica), uma coleção de outros agentes, a Internet, ou até mesmo muitos destes combinados, e são capazes de entendê-lo e de reagir de uma maneira rápida às mudanças que nele ocorrem;
- Pró-atividade: agentes não agem simplesmente em resposta ao seu ambiente, eles são capazes de exibir um comportamento orientado por um objetivo através da tomada de iniciativa;
- Habilidade social: agentes interagem com outros agentes (e possivelmente usuários) através de uma linguagem de comunicação entre agentes [Wooldridge, 1995], e tipicamente têm a habilidade de se envolver em atividades sociais (como negociação ou resolução cooperativa de problemas) para atingir seus objetivos.

Um dos maiores problemas no desenvolvimento de sistemas de agentes é a obtenção de um equilíbrio racional entre a tendência de um agente reagir às mudanças no ambiente e a de agir no sentido de atingir seus objetivos. É fácil implementar agentes que apenas reajam aos estímulos do ambiente no qual estejam inseridos, assim como os que ajam apenas no sentido de atingir seus objetivos, o que difere quanto a desenvolver um sistema apropriadamente equilibrado entre estes extremos [Wooldridge; Jennings, 1995]. De acordo com Dennet [Dennet, 1987], agentes que alcançam o equilíbrio nestes dois tipos de comportamento podem ser classificados como *sistemas de raciocínio prático*. Desta forma, eles se mostram apropriados para operar nos tipos de ambiente em que o comportamento da tradicional engenharia de software tem se mostrado ineficaz.

CAPÍTULO 3

AGENTES INTELIGENTES EM UM MODELO DE DADOS

Este capítulo é responsável por apresentar a fundamentação conceitual deste livro em suas duas principais vertentes: a sociedade de agentes como forma de realização de traduções entre fontes, e o modelo de dados comum para o qual as traduções serão realizadas. Discorre acerca das especificações necessárias para o desenvolvimento de sistemas que utilizam agentes (seção 3.1) e de como tais especificações foram utilizadas para a proposição da sociedade de agentes utilizada neste módulo (seção 3.2). Apresenta também um estudo comparativo entre diversas abordagens para modelo de dados comum (seção 3.3) e, finalmente, a partir deste estudo, propõe o formato a ser adotado como tal (seção 3.4).

3.1 Especificando a Sociedade de Agentes

Nesta seção, é considerado o problema de se especificar um sistema de agentes. São analisados aspectos que dizem respeito aos requisitos que devem ser levados em consideração para o desenvolvimento de uma ferramenta (ou de um *framework*), além do tipo de propriedades que tais requisitos devem ser capazes de representar.

Tomando por base o conceito de agente como sendo sistemas de pensamento prático, apresentado anteriormente na seção 2.5, a abordagem predominante para a especificação destes sistemas envolve seu tratamento como sistemas intencionais [Wooldridge, 1995] que devem ser entendidos através da atribuição, para tais agentes, de estados mentais como crenças, desejos e intenções [Dennet, 1987]. Seguindo esta idéia, desenvolve-se um número considerável de abordagens para especificar agentes formalmente capazes de representar os seguintes aspectos de uma sociedade de agentes:

- As *crenças* que os agentes possuem – informações sobre o seu ambiente, que inclusive podem ser incompletas ou incorretas;
- Os *objetivos* que os agentes tentarão cumprir;
- As *ações* que os agentes realizam e os efeitos destas ações;
- A *interação pessoal* dos agentes uns com os outros e com o seu ambiente

ao longo do tempo.

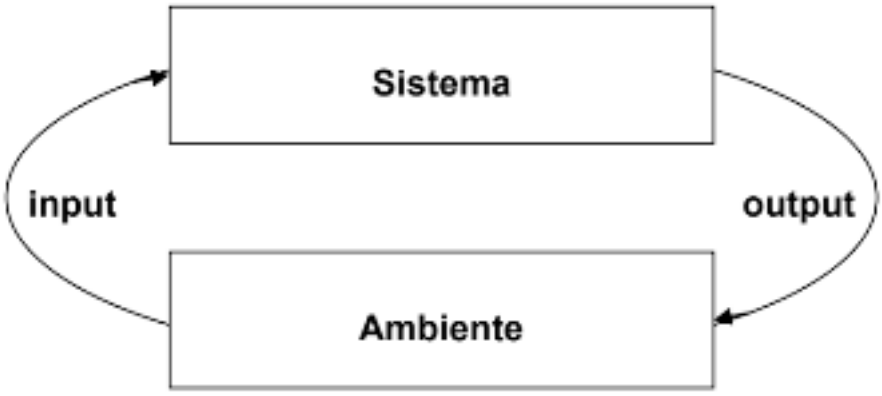

Figura 3.1: Uma visão abstrata de sistemas de software

Para explicar como estes aspectos de interferência e ação particulares para cada agente interagem para efetuar o mapeamento de um sensor de input para um efetuador de output (observe a figura 3.1 [Wooldridge, 1995]) faz-se necessária uma teoria, chamada em [Wooldridge, 1995] de teoria de agente. A abordagem mais adequada para uma teoria de agente formal é, aparentemente, o uso de uma lógica modal temporal (para não fugir demais do tema deste trabalho, ou seja, por questões de objetivo, uma discussão mais detalhada de tais lógicas não é apresentada aqui – para maiores referências e explanações, veja [Wooldridge; Jennings, 1995]). Duas das mais conhecidas destas lógicas são Teoria da Intenção de Cohen-Levesque [Cohen, Levesque, 1990] e o Modelo de crença-desejo-intenção de Rao-Georgeff [Rao; Georgeff, 1995]. O modelo Cohen-Levesque toma como premissa apenas duas primitivas: crenças e objetivos. Outras primitivas (em particular, a noção de intenção) são construídas a partir destas duas. Em contraste, Rao-Georgeff toma intenções por primitivas, em adição às crenças e aos objetivos. A maior dificuldade (e também o maior problema) verificada junto aos teóricos de sistemas de agentes é o desenvolvimento de um modelo formal que seja suficiente para tratar satisfatoriamente as inter-relações entre as diversas primitivas que, juntas, englobam o estado interno de um agente. Comparativelmente, poucas tentativas sérias têm sido feitas para a especificação de um sistema de agentes baseados em lógica (uma tentativa pode ser encontrada em [Fisher; Wooldridge, 1997]).

3.2 Sociedade de Agentes

Uma vez que esta abordagem faz uso de Sociedade de Agentes (SA) como auxiliar na construção e manutenção da federação faz-se necessário uma arquitetura para gerenciamento dos agentes. A figura 3.2 *[*Costa; Campos; Souza, 2003*]* é uma

descrição da arquitetura de agentes (ASA).

Devido ao fato de esta abordagem fazer uso de uma sociedade de agentes para lidar com a construção e manutenção da federação, é necessária uma arquitetura para a administração dos agentes. A estrutura básica da sociedade de agentes é composta por:

- Agente Diretor – agente que controla as operações dos agentes na SA, administrando a o intercâmbio de informação entre eles através do Agente Canal de Comunicação, e recebe solicitações de outros agentes do sistema, repassando-as a agentes que suportem os serviços solicitados. Quando recebe uma resposta, o Agente Diretor a endereça para os solicitantes. Por isso, informações sobre todos os agentes são mantidas pelo Agente Administrador do Sistema;
- Agente Administador de Sistema – responsável pela administração do ciclo de vida dos agentes no Agente Diretor. Suas ações incluem gravar, remover, modificar, pesquisar e controlar o ciclo de vida. Há mais de um Agente Administrador de Sistema na estrutura da sociedade;
- Agente Diretório – provê serviços para outros agentes, sendo, desta forma, um serviço normativo e mandatório. Os agentes podem registrar seus serviços no Agente Diretório ou consultá-lo para achar quais serviços são oferecidos por outros agentes;
- Agente Canal de Comunicação – provê um caminho básico para intercâmbio de informação entre os agentes Administrador de Sistema e Diretor;
- Agente Administrador – lida com acesso ao repositório de metadados. Assim, tem a responsabilidade de inserir, remover e atualizar dados em tal repositório.

Os agentes da SA baseiam-se no conhecimento do ambiente no qual estão inseridos. O conhecimento dos agentes é representado por objetos (entidades) de um dado domínio, com suas propriedades e relações. Tal conhecimento é representado na base de conhecimento e a razão dos agentes nas suas possíveis ações com a ajuda de uma máquina de interface. A base de conhecimento contém sentenças em uma linguagem de representação de conhecimento para a representação dos fatos e regras, enquanto uma máquina de inferência é responsável por deduzir novas situações ou regras.

ASA é constituída de Agente Controlador, Agente Canal de Comunicação (ACC) e os Agentes (definidos em cada um dos ambientes). A notação UML é adotada para a apresentação da arquitetura do ASA.

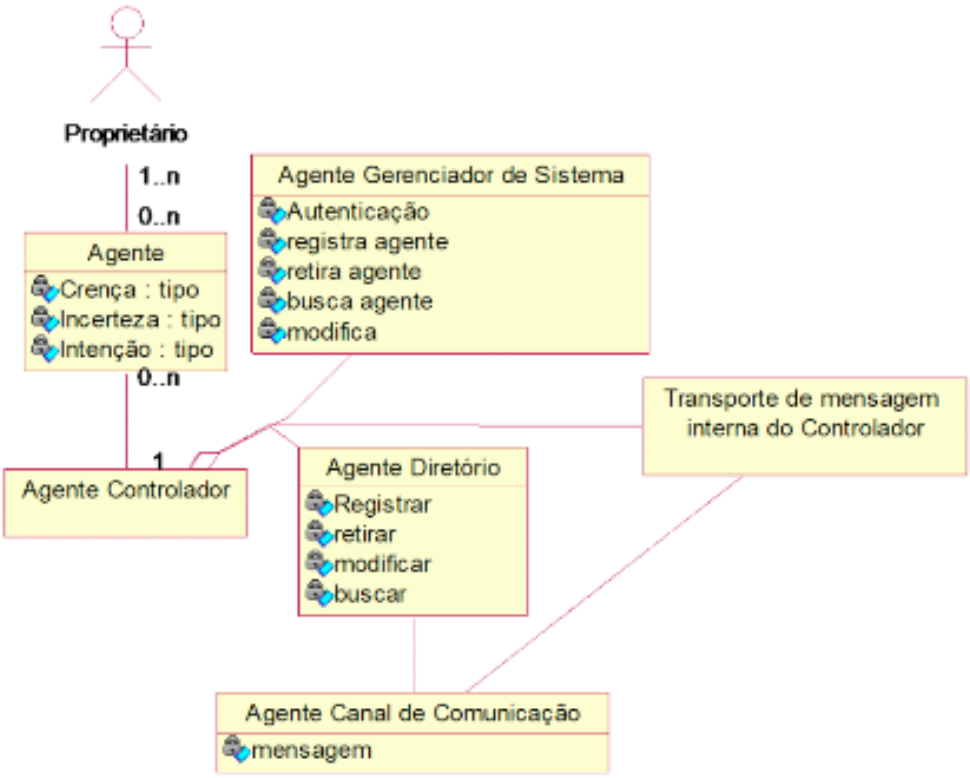

Figura 3.2: Modelo de Referência para o Gerenciamento de Agentes

O **Agente Controlador** (AC) é constituído de outros três: o Agente Gerenciador do Sistema (AGS), o Agente Canal de Comunicação (ACC) e o Agente Diretório (AD). O AC controla o funcionamento dos Agentes da SA, gerenciando a troca de informações entre vários Agentes através do ACC. Um agente deve ser registrado no controlador de forma a interagir com outros agentes. O controlador recebe solicitações dos outros agentes do sistema e as passa para aqueles que fornecem os serviços solicitados. Quando recebe as respostas, o controlador as direciona para os solicitantes. Para tanto, são mantidos registros para cada Agente e sua localização através do AGS. O AC também provê serviços para outros agentes através do AD. Agentes se comunicam através da troca de mensagens em XML.

O **Agente Gerenciador do Sistema** (AGS) é um agente que supervisiona o acesso do AC. Somente um AGS existe no AC. O AGS é responsável por gerenciar o ciclo de vida dos agentes sobre o AC. Suas ações incluem o registro, a remoção, a modificação, busca e controle do ciclo de vida do Agente.

Como já dito, o **Agente Diretório** (AD) provê serviço para outros agentes e o **Agente Canal de Comunicação** provê um caminho básico para a troca de informação entre agentes, AGS e outros AC.

O AC atua em vários contextos, que serão descritos nas seções correspondentes.

3.3 Estudo Comparativo entre MDC

Nesta seção é apresentado um estudo comparando diversos padrões que têm sido normalmente usados como modelo de dados comum em sistemas de integração de dados. O objetivo desta seção é adotar um formato para o qual o módulo proposto mapeará as fontes de dados heterogêneas. Este padrão deve ser capaz de fornecer características suficientes para que o modelo de integração não perca informações que prejudiquem o processo de federação e análise dos dados pertencentes ao sistema distribuído. Na seção 3.3.1 é apresentado o metamodelo MOF [MOF, 2002]. Na seção 3.3.2, a linguagem de modelagem UML é proposta utilização na perspectiva de integração de dados. São apresentados: na seção 3.3.3, o padrão XMI [XMI, 2000] da IBM, na seção 3.3.4, o difundindo padrão CWM, e, finalmente, na seção 3.3.5, a largamente utilizada linguagem XML, como possíveis MDC.

3.3.1 O META-MODELO MOF (Meta Object Facility)

MOF é, na verdade, a definição de uma linguagem e uma estrutura de trabalho para especificação, construção e gerenciamento de modelos para metadados, isto é, não possui uma notação gráfica ou uma linguagem de restrições própria, mas utiliza notação UML e a linguagem *Object Constraint Language* (OCL) [Booch; Jacobson; Rumbaugh, 1998] para estes propósitos.

MOF permite a definição de esquemas e metamodelos. Assim, qualquer modelo baseado em MOF pode ser armazenado em um repositório de metadados: uma vez que os modelos conceituais, relacionais, OO, semi-estruturados forem mapeados para uma linguagem como XMI, eles poderão ser armazenados em um repositório baseado em MOF.

Abaixo é apresentado um quadro resumo das propriedades do MOF de acordo com suas características.

Suporte a Modelos Conceituais	Sim (metamodelos expressos em XMI)
Expressividade	Sim (representação de classes)
Modelos Estruturados	Sim (representação XMI)
Modelos Semi-Estruturados	Sim (representação XMI)
Extensibilidade	Metamodelo para especificação de metadados

Quadro 3.1: Funcionalidades oferecidas pelo padrão MOF

3.3.2 UML (Unified Modeling Language)

UML, um padrão OMG [XMI, 2000], é uma linguagem gráfica para a especificação, visualização, construção e documentação de artefatos de sistemas de software. O padrão UML define uma linguagem de modelagem orientada a objetos muito rica, suportada por um grande número de ferramentas gráficas, uma destas estudada no período da bolsa, o Rational Rose [Booch; Jacobson; Rumbaugh, 1998].

A linguagem UML é extremamente rica, entre outros motivos, pois:

- Inclui idéias de outras linguagens para modelagem (como Booch [Booch; Jacobson; Rumbaugh, 1998]);
- Suporta a modelagem de conceitos atuais no desenvolvimento de software, tais como: concorrência, distribuição, executabilidade, entre outros;
- É extremamente flexível;
- Permite o intercâmbio de modelos e define interfaces de repositório.

É fundamental, para um sistema, a possibilidade de sua visualização em UML, pois qualquer ferramenta que implemente o padrão UML será capaz de permitir a visualização de artefatos de software. Entretanto, UML não é adequada para ser utilizada como MDC, devido ao fato de ser voltada para modelagem OO e ser um modelo para especificação e visualização de sistemas de software, o que inibe o estabelecimento de uma interface para acesso a repositório de metadados e implementação de operadores de integração e derivação.

Abaixo observa-se que, embora satisfaça a pré-requisitos necessários, não é adequada como MDC devido à não satisfação das necessidades descritas acima.

Suporte a Modelos Conceituais	Sim (importa modelos XMI)
Expressividade	Sim (representação de classes)
Modelos Estruturados	Sim (representação XMI)
Modelos Semi-Estruturados	Sim (representação XMI)
Extensibilidade	Sim (seu modelo é uma especificação MOF)

Quadro 3.2: Funcionalidades oferecidas pelo padrão UML

3.3.3 XML (eXtensible Markup Language)

XML é uma linguagem de marcadores para descrição de informações. Uma linguagem de marcadores é um conjunto de convenções de marcadores utilizados para codificação de textos.

XML é um subconjunto de SGML, um padrão internacional para definição de

formatos de representação de textos em meio eletrônico. Uma de suas idéias principais é tornar explícita a separação entre componentes de um documento eletrônico: apresentação, conteúdo e estrutura.

A grande vantagem de XML é que ela é uma linguagem que captura muita semântica, isto é, os marcadores têm significado próprio para o domínio. Além disso, XML é uma linguagem extremamente extensível, o que torna possível aos usuários definir novos marcadores, de acordo com o domínio que está sendo modelado.

Além disso, XML:

- Dá suporte a Unicode;
- É aberta e independente de plataforma;
- É flexível por permitir a apresentação de um mesmo conteúdo em diferentes formatos;
- Permite qualquer nível de aninhamento da estrutura dos documentos;
- É utilizada para representar os metamodelos compatíveis com MOF (incluindo XMI).

Suporte a Modelos Conceituais	Não (não é um modelo baseado em MOF)
Expressividade	Sim (representação de classes)
Modelos Estruturados	Sim (representação própria)
Modelos Semi-Estruturados	Sim (representação própria)
Extensibilidade	Sim (seu modelo é extensível)

Quadro 3.3: Funcionalidades oferecidas pelo padrão XML

3.3.4 XMI (XML Metadata Interchange)

XMI é um padrão para codificação de metadados de ferramentas de desenvolvimento orientadas a objeto que provê uma forma de intercâmbio entre modelos orientados a objetos (OO) e dados usando XML.

XMI, apesar de não ser uma linguagem XML, no nível de codificação é baseado em XML. Segundo Ribeiro [Ribeiro; Florentini, 2000], XMI é uma especificação de como gerar linguagens XML adequadas para modelos de dados e como codificar esses metadados em um documento XML. Desta forma, a especificação do XMI nada mais é que conjunto de regras que normatizam a geração de XML a partir de MOF.

XMI define dois conjuntos de geração de regras para criar documentos XML e *(Document Type Definitions)* DTD XML. O primeiro conjunto de regras especifica como derivar a gramática da linguagem XML correspondente ao metamodelo. O segundo conjunto de regras são os DTD XML que escrevem regras para a construção

do documento XML correspondente ao modelo.

Suporte a Modelos Conceituais	Sim (modelos baseados em MOF ou UML)
Expressividade	Sim (representação de classes)
Modelos Estruturados	Sim (representação XMI)
Modelos Semi-Estruturados	Sim (representação XMI)
Extensibilidade	Sim (seu modelo é uma especificação MOF)

Quadro 3.4: Funcionalidades oferecidas pelo padrão XMI

Observa-se desta forma que, como linguagem de definição de modelos, XMI apresentaria as mesmas limitações providas por UML. Entretanto, XMI torna-se diferencial a partir do fato de ser uma linguagem XML para definição de metadados. Ou seja, ela pode ser utilizada para o estabelecimento de uma interface de acesso a repositório de metadados e, com a utilização de esquemas de implementação adequados, pode ser possível utilizá-la para representar operadores de integração de derivação.

3.3.5 CWM (Common Warehouse Metamodel)

CWM [Crary; Weirich, 1999] é um padrão OMG para intercâmbio de metadados nos ambientes de DW e análise de negócios. Ele provê uma linguagem comum, baseada em um metamodelo genérico (mas semanticamente rico), para DW e análise de negócios, para descrever metadados.

Basicamente, o processo de utilização do CWM pode acontecer, em uma empresa, da seguinte forma:

- Utilização de UML para representação de modelos e metamodelos, bem como para descrever a semântica da análise de objetos e do projeto de modelos;
- Utilização de MOF para definição e manipulação de metamodelos através de interfaces programáveis e CORBA [Vinoski, 1997], para definição da infra-estrutura de objetos distribuídos;
- Utilização de XMI para intercâmbio linear de metadados.

A especificação de CWM consiste de definições de metamodelos em diversos domínios, sendo ela bastante abrangente. Entretanto, o CWM não atende às características de interface para acesso ao repositório de metadados, além de não definir os operadores de integração e derivação, embora dê suporte a todas as outras características.

O que torna o CWM um modelo extremamente útil e poderoso, uma vez que esteja no formato XMI, é fato de representar uma série de outros modelos, como

o Relacional, OO, XML e outros. Como o objetivo desta abordagem é prover uma ferramenta que seja capaz de integrar todo o ciclo de vida do desenvolvimento de software, a conversão desses modelos (e o seu armazenamento no repositório de metadados) para um único modelo torna-se parte fundamental.

Suporte a Modelos Conceituais	Sim (utiliza UML para representação conceitual)
Expressividade	Sim (representação de classes)
Modelos Estruturados	Sim (representação XMI)
Modelos Semi-Estruturados	Sim (representação XMI)
Extensibilidade	Sim (seu modelo é uma especificação MOF)

Quadro 3.5: Funcionalidades oferecidas pelo padrão CWM

3.4 MDC Escolhido

Nesta abordagem, o MDC escolhido é XMI. Dentre os estudados, ele é o único que permite o intercâmbio entre as diversas ferramentas de software devido ao fato de seu esquema ser definido de acordo com o padrão estabelecido pelo OMG. XMI pode mapear MOF, UML, CWM, bem como dar suporte à representação de ambas as fontes estruturadas e semi-estruturadas, o que não acontece com os demais modelos estudados.

Dentre os modelos para portabilidade de dados, XMI é o único que dá suporte a representação de modelos conceituais. O fato dos esquemas de bancos de dados representados em CWM estarem no formato XMI torna clara a sua utilização para derivação de um DW.

Um sistema de bancos de dados federados evolui na integração gradual de um conjunto de sistemas de bancos de dados autônomos não-relacionados. Esse processo de evolução deveria ser subdivido em três fases: pré-integração, desenvolvimento de um sistema de bancos de dados federados (SBDF) e operação desse SBDF [Sheth; Larson, 1990].

O MDC possibilita descrever os esquemas locais divergentes usando uma representação única para as diversas fontes heterogêneas de dados. Nesta arquitetura a transformação do esquema local para o esquema componente é feita através da abordagem *bottom-up*, possibilitando assim, que informações não capturadas através do processo automático de transformação do esquema local para o esquema componente sejam adicionadas ou mesmo modificadas. Desta forma, o MDC facilita a tarefa de negociação e integração realizada quando desenvolvendo um SBDF. Um processo de conversão separado para cada tipo de modelos de dados é requerido [Batini; Lenzerini; Navathe, 1986].

CAPÍTULO 4

AGENTES INTELIGENTES E ESQUEMAS DE DADOS

Este capítulo discorre acerca dos padrões já existentes cuja utilização se mostra fundamental e adequada para a implementação do módulo proposto, e de metodologias que podem ser utilizadas para se manter a abordagem aqui proposta com o máximo nível de reutilização de ferramentas e submódulos. Na seção 4.1, esses módulos são apresentados. Na seção 4.2, discorre-se acerca da aplicabilidade de tais metodologias para o domínio abordado. Na seção 4.3, a metodologia para a implementação da sociedade de agentes é apresentada.

4.1 Módulos para a Abordagem

Nesta seção são apresentados diversos padrões e ferramentas que foram utilizados pontualmente nesta abordagem como módulo para a execução de atividades que, em conjunto, foram capazes de prover aos agentes pertencentes ao módulo de mapeamento um conhecimento abrangente do problema de se mapear fontes heterogêneas de forma eficiente e correta.

4.1.1 XMI: Intercâmbio e Modelagem

XMI tem conseguido larga aceitação como o padrão referência para a representação de informação estruturada em contextos como a WWW. Basear os padrões OMG em XML significa que XMI pode ser usado para intercâmbio de metadados em ferramentas repositórios de metadados não baseados em CORBA.

A especificação XMI [XMI, 2002] suporta que o intercâmbio de qualquer tipo de metadado pode ser expresso usando a especificação MOF, incluindo tanto informações sobre modelos quanto informações sobre metamodelos. A especificação fala que a codificação de metadados consiste tanto de modelos completes quanto de fragmentos de modelos, como as extensões específicas de metadados para a implementação de ferramentas.

XMI tem um suporte opcional para o intercâmbio de metadados, de uma forma

diferenciada, e para o intercâmbio de códigos-fonte, que têm um entendimento incompleto dos metadados. Isto nos permite concluir que, apesar de não ser uma linguagem para portar dados, XMI, por ser baseado em XML no nível de codificação, provê uma forma de relacionamento entre modelos orientados a objetos (OO) e dados usando XML.

Normalizando a geração de documentos XML a partir de MOF, pode-se obter uma leitura de XMI. Esta é uma forma de se observar XMI como capaz de gerar diferentes linguagens XML (MOF não é nada mais do que um exemplo para intercâmbio e portabilidade de metadados) adequadas para modelagem de dados e codificação de metadados.

A especificação de XMI [XMI, 2002] define, basicamente, dois conjuntos de geração de regras:

i) As regras de produção XML para produção de documentos DTD para metadados XMI codificados. Estas DTD servem como especificações sintáticas para documentos XMI, e permitem que ferramentas XML genéricas sejam usadas para compor e validar documentos XMI;

ii) As regras de produção de documentos XML para a codificação de metadados em um formato compatível com XML. As regras de produção podem ser aplicadas ao contrário para decodificar documentos XMI e reconstruir os metadados.

4.1.2 API para XML

A definição de API, *Application Programming Interfaces*, para o paradigma Orientado a Objetos, OO, que é abordado por XML, está intrinsecamente relacionada com a maneira para acesso e manipulação do conteúdo de um documento XML. *Document Object Model,* DOM [Wood et al, 1998] e *The Simple API for XML,* SAX [Brownell, 2002] são as API mais conhecidas na literatura, e as suas propostas têm sido as mais usadas em aplicações desenvolvidas com embasamento técnico-científico, devido ao fato de suas visões condensarem os dois extremos possíveis para o acesso de conteúdo XML, por conseqüência da própria estrutura de um documento XML: DOM propõe a visão de um documento XML através da construção de uma estrutura de dado árvore, enquanto SAX propõe que os eventos é que coordenem a visão de um documento XML.

Abaixo é dada uma explanação acerca das duas API e, após isso, é feita uma introdução ao JDOM [Baumgartner; Flesca; Gottlob, 2001], uma forma de representação Java [Gray, 1996] de um documento XML, que pode ser vista como uma alternativa ao DOM e ao SAX, embora integre as visões propostas pelo DOM e pelo SAX.

4.1.2.1 DOM

O DOM é constituído de um conjunto de interfaces e classes que representam a estrutura de documentos XML e HTML. O documento aqui é visto como uma coleção de objetos (nós) em estrutura de árvore. Assim, ao ler o documento XML, o DOM

propõe que se carregue uma árvore, com os dados contidos neste documento.

É uma API que define uma estrutura lógica de documentos e a maneira que um documento é acessado e manipulado. Devido ao fato de a informação que está contida em documentos XML ser tradicionalmente vista como dados, ao invés de uma composição, ou associações, de documentos, nada mais óbvio que se desenvolver um modelo de objetos que trate, efetivamente, dados na sua maneira natural (administração, consultas, atualizações). O DOM é a API oficialmente recomendada pelo *World Wide Web Consortium*, W3C.

Em Java, tem-se uma série de classes e pacotes relacionados com o pacote org.w3c.dom. O protocolo DOM converte um documento XML em uma coleção de objetos no seu programa. Então se torna possível manipular o modelo de objetos de qualquer forma que faça sentido. Este mecanismo é chamado em Java de protocolo de acesso randômico *("random access" protocol)*, pois é possível acessar qualquer parte dos dados a qualquer hora, além de ser possível modificar, remover e inserir dados.

Ao representar um documento XML no DOM, tem-se a formação de uma estrutura de dados baseada em árvores (o documento XML é visto como uma árvore), onde cada nó contém um dos componentes de uma estrutura XML. Os dois tipos de nós mais comuns são os nós elementos e os nós textos. Os métodos Java para DOM permitem a criação e remoção de nós, atualização de seus conteúdos e diversas outras funcionalidades para lidar com sua hierarquia.

4.1.2.2 SAX

O SAX também é constituído de uma interface que permite a interação com documentos XML. Entretanto, o documento aqui é visto como uma seqüência de eventos, não sendo possível, portanto, acessar randomicamente os dados pertencentes ao documento. O SAX é visto como ideal se o acesso e manipulação dos dados são feitos de maneira seqüencial.

Considerada difícil para programadores, principalmente devido a dois fatores: DOM é um modelo de objetos que lida com a localização de objetos diretamente na memória e a maioria dos programadores não está acostumada com programação baseada em eventos, quando é necessário se manter informado sobre as pesquisas realizadas e continuar a atualizar essa informação enquanto eventos são requisitados.

Em se tratando da tecnologia Java, assim como JDBC é comumente utilizado como interface para implementação de bancos de dados relacionais, normalmente SAX é usado como a interface para implementação de parsers XML (ou propostas que desempenham o papel que está destinado aos parsers ou que ainda se comportam como parsers).

O SAX pode ser visto como um protocolo para "acesso serial" para XML. Seria o mecanismo *fast-to-execute* (já que requer uma quantidade de memória muito menor que qualquer outro mecanismo para manipular documentos XML). Em Java, devido ao fato de SAX ser baseado em eventos, o primeiro passo dado é uma espécie de

validação inicial do documento, o *handler* inicial. Este *handler* é uma espécie de registro (feito com um SAX parser) da localização atual do documento. Então, toda vez que uma nova tag XML ou um erro são encontrados, ou qualquer outro tipo de mensagem é passada, os métodos de parsing *(handler* atual e SAX parser) no documento XML são solicitados.

4.1.2.3 JDOM: ASSOCIANDO DOM E SAX

Há duas principais formas de API XML/SGML *(Standard Generalized Markup Language):*

a) API baseadas em árvores:

É exatamente o que o DOM faz. Este tipo de API interpreta o documento XML como uma árvore, e faz exatamente este mapeamento, ou tradução. A partir disto, permitem a aplicação lidando com o documento XML que navegue através desta árvore. O DOM é, de fato, o representante principal deste tipo de API, embora haja outras API que baseiem sua estrutura em árvore.

API baseadas em árvores são úteis para grande quantidade de aplicações, pois possibilitam consultas rápidas sobre os elementos do documento, além de atualizações. Entretanto, fazem uso de uma exagerada quantidade de recursos do sistema, especialmente se o documento é muito grande.

b) API baseadas em eventos:

O conceito de API baseada em evento, por sua vez, está extremamente relacionado com o conceito de parsing. Um API deste tipo comunica eventos de parsing (como o começo e o fim de elementos) diretamente para a aplicação através de callbacks e não faz uso de estruturas de dados para lidar com os elementos. Para lidar com os diferentes eventos possíveis, a aplicação deve implementar handlers. O SAX é o exemplo principal deste tipo de API.

Estas API possibilitam um acesso mais simples para o documento XML. Devido ao fato de o acesso ser feito num nível menor, estas API não consomem tanto os recursos do sistema. Entretanto, não possibilitam consultas rápidas aos elementos e atualizações eficientes do documento XML.

O JDOM é uma representação, uma abstração Java para um documento XML. Provê uma forma de representar o documento XML para uma leitura, manipulação e escrita fácil e eficiente. É uma espécie de API que discerne as vantagens do DOM e do SAX em detrimento uma da outra e une essas vantagens com a facilidade de programação Java.

Não se trata de um XML parser. JDOM é um document object model (DOM) que usa parsers XML para construir documentos. Por exemplo, a class SAXBuilder do JDOM usa os eventos SAX gerados por um parser XML para construir uma árvore JDOM. É possível ao JDOM a utilização de qualquer parser XML.

Os documentos, no JDOM, podem ser construídos de arquivos XML, árvores DOM, eventos SAX ou qualquer outra fonte. Os documentos também podem ser convertidos em arquivos XML, árvores DOM, eventos SAX ou qualquer outro destino.

Esta habilidade tornar possível lidar com diferentes abordagens possíveis, de acordo com as necessidades de cada sistema.

A grande vantagem do JDOM está associada à possibilidade de utilizar a visão SAX até um determinado ponto do documento XML (que será inútil numa consulta). A partir deste ponto, guarda-se o documento na forma de árvore, até um outro determinado ponto (que possibilitará o acesso rápido aos elementos entre estes dois pontos). Tem-se então uma seleção do que é necessário para a consulta, com acesso rápido e sem comprometer recursos do sistema.

Devido ao fato de ser uma API de implementação Java extremamente poderosa, com a possibilidade de parsing seletivo e acesso rápido a memória, esta abordagem propõe o JDOM para a representação dos documentos que seguem o formato XMI. Uma descrição de como o JDOM e, conseqüentemente, o DOM e o SAX são utilizados no projeto é dada na seção 7.

4.1.3 XMI FRAMEWORK

Assim como o JDOM, XMI Framework [Grose, 2001] é uma API Java que permite ver um documento XMI (no caso do framework, especificamente XMI) como uma coleção de objetos e interfaces Java.

XMI Framework consiste de uma interface hierárquica com classes de implementação correspondentes, que implementam um modelo OO. A hierarquia representa um modelo de objeto que é um subconjunto de MOF e UML. As outras classes no framework são responsáveis pela representação dos modelos em si: representam arquivos XMI, DTD XMI, esquemas XMI, coleções de objetos, modelos e adaptadores (para a conexão entre objetos do usuário e arquivos XMI).

O intuito dos desenvolvedores desta ferramenta era, inicialmente, prover um material de aprendizado da linguagem XMI. Devido ao fato da sua complexidade para programadores e projetistas, normalmente desconhecedores de todo o baixo nível que há por trás de linguagens de modelagem, o propósito da ferramenta foi transferido para permitir o trabalho de classes de engenheiros de software com arquivos XMI, apropriados para a representação de modelos em formato XML, sem ser necessário um conhecimento profundo da linguagem de XMI.

O XMI Application Framework é utilizado aqui para representar modelos integrados e modelos a serem integrados. Todo o processo de construção do MDC para dados é feito por classes que usam os recursos do JDOM. Já para modelos, XMI Framework é utilizado.

4.2 Relações entre Padrões e o Módulo Proposto

Diversos padrões que são utilizados neste módulo foram apresentados na seção anterior. Nesta seção é descrita a forma como tais padrões são aplicados nas diversas

classes do projeto. Será necessário o auxílio da Figura 4.1, uma extensão da Figura 2.3 aplicada à integração com XMI.

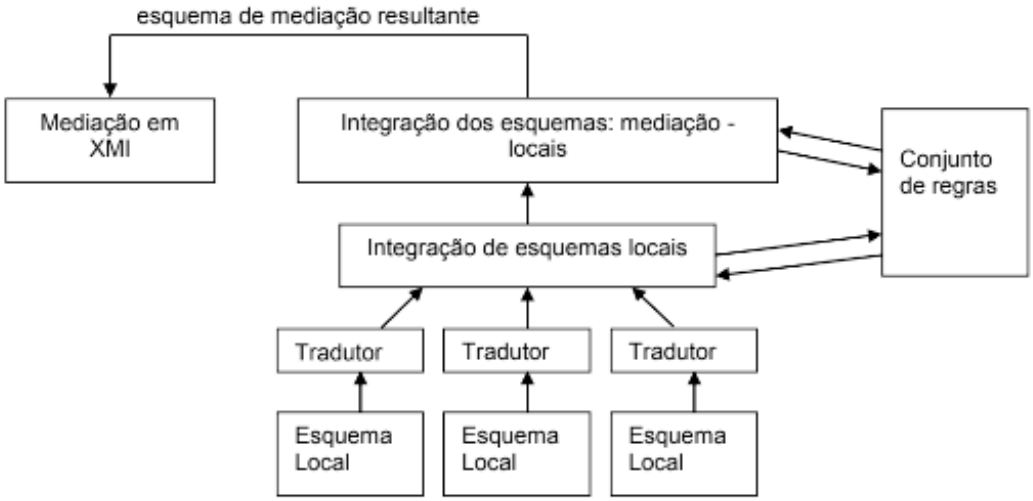

Figura 4.1: Esquema de integração utilizando XMI

Em todos os instantes desta arquitetura os padrões apresentados são utilizados. Analisando o esquema proposto e suas aplicações para a representação de dados, utilizando uma abordagem bottom-up, observa-se que diversas fontes de dados estão em seus esquemas locais, em formatos desconhecidos. O primeiro passo para a integração de esquemas é o mapeamento de tais fontes para o formato do MDC, neste caso, XMI. Portanto, na transição do Esquema Local para o Tradutor, torna-se necessário realizar um mapeamento de uma fonte de dados local para XMI.

Este mapeamento, nesta abordagem, se dá de forma autônoma, para cada SGBD local. Entretanto, para o gerador de documento XMI, pertencente ao este módulo, devem ser geradas, a partir da organização do SGBD, informações que serão reconhecidas na classe **DadoXMI** como sendo elementos XML, atributos XML, conteúdos pertinentes a informação (como PCDATA, CDATA, por exemplo). Os elementos XML são representados na classe **DadoXMI** como JDOM **Element.** Todas as operações para acesso às funcionalidades dos elementos XML têm suporte nesta classe. A partir de toda a informação XML é gerado um documento XML, formatado como um padrão da abordagem (método **criaDoc**() de **DadoXMI**).

A partir dos mapeamentos obtidos nos tradutores, o conjunto de regras (a definição de mecanismos para a elaboração de regras, como motores de inferência, representa uma etapa posterior no processo de continuidade desta pesquisa) é consultado para que, a partir dos mapeamentos dos Esquemas Locais, haja a definição de um esquema integrado, gerado a partir das relevâncias semânticas inerentes ao conjunto de regras, definido, provavelmente, pelas prioridades do usuário que está realizando integração. Depois disso, um novo processo, agora envolvendo o esquema integrado, resultante da interligação dos esquemas locais, e um esquema armazenado

(que pode estar desatualizado ou pode conter informações de pesquisas realizadas anteriormente) acontecerá, o que resultará em saídas que sofrerão o processo descrito acima.

O processo para a integração de modelos é muito parecido. A única diferença é que realizar integração sobre modelos requer mapeamentos de linguagens como UML: estas linguagens são esquemas locais, traduzidos para a linguagem XMI. Para mapeamentos de códigos-fonte Java, modelos UML, MOF e até modelos XMI, é utilizada a ferramenta XMI Toolkit [Costa; Campos; Souza, 2002] da IBM, que faz uso do XMI Framework para a geração de representação XMI da metainformação inerente aos códigos-fonte, modelos e esquemas existentes nos Esquemas Locais.

Portanto, XMI Framework é utilizado para realizar os mapeamentos para a linguagem XMI. Neste módulo, a classe **ModeloXMI** representa os arquivos XMI que são gerados por meio da leitura de arquivos XMI resultantes dos mapeamentos. Esta classe conta com métodos de leitura (como **load**(), que utiliza o poder de implementação do método **XMIFile.load**(), do XMI Framework), escrita (como **write**(), que utiliza o método **XMIFile.write**(), do **XMIFramework**) e atualização de conexões (como os métodos que manipulam **ContentXMI**, outra classe de implementação desta abordagem), que são indispensáveis no processo de integração.

A partir deste passo, a ferramenta encarregada da integração, que utiliza o módulo implementado neste trabalho, deve ser responsável de tomar o sentido de realizar integração dos modelos. A partir daí, faz-se integração em cima de fontes de modelos: por meio da consulta de relevâncias semânticas definidas num conjunto de regras (a definição de mecanismos para definição de regras, como motores de inferência, é etapa posterior no processo de continuidade desta pesquisa) há a geração de um esquema integrado. Este esquema será novamente integrado com um esquema previamente elaborado, que pode já estar desatualizado ou conter informações relevantes de fontes não consultadas no processo corrente, mas já consultadas anteriormente.

Para todas estas etapas de integração são necessárias representações dos modelos que estão sendo manipulados no formato XMI. É a partir do XMI Framework que informações, pertinentes aos dados de modelos e metadados que estão sendo manipulados, são acessadas, integradas, aproveitadas. Todo o processo algorítmico envolve a utilização de classes do XMI Framework.

4.3 Sociedades de Agentes: Implementação

Uma vez dada a especificação (ver Seções 3.1 e 3.2), deve-se implementar os artefatos definidos de acordo com tal especificação. O passo nesta seção considerado é a transição de uma especificação abstrata para um sistema compucional concreto (cuja arquitetura é apresentada no Capítulo 5), no caso, a abordagem em si, proposta por este trabalho.

Há pelo menos duas possibilidades para alcançar a transição aqui considerada:

i) Realizar o desenvolvimento ou a execução direta, de alguma forma, da especificação abstrata;

ii) Traduzir ou compilar, de alguma forma, a espeficação em um formulário computacional físico usando uma técnica automática de transição.

Nas subseções seguintes, cada uma destas possibilidades vai ser analizada.

4.3.1 Execução Direta de Especificações de Agentes

Partindo da suposição de ter-se uma especificação para a sociedade de agentes, expressada em uma linguagem lógica, uma forma de obter a sociedade de agentes concreta a partir desta suposição é tratá-la como uma especificação executável e interpretar essa especificação diretamente, de forma a gerar o comportamento do agente. A interpretação de uma especificação de agentes pode ser vista como um tipo de prova construtiva de satisfatibilidade, de onde se pode mostrar que a especificação é satisfatível pela construção de um modelo (no sentido lógico) para ela. Se puderem ser construídos modelos computacionais para a linguagem de especificação lógica que possuem uma interpretação computacional, então pode-se dizer que o modelo construído executa a especificação.

Para se ter uma visão mais concreta, pode-se considerar uma linguagem de programação concorrente qualquer (ver a linguagem MetateM em [Fisher, 1997]). Nesta linguagem, os agentes são desenhados de forma a possuirem uma especificação temporal e lógica do comportamento que se espera que eles exibam. Esta especificação influi diretamente no comportamento de cada agente. Os modelos para a lógica temporal, dos agentes desenhados nesta linguagem, são especificados por meio de uma seqüência discreta e linear de estados. Uma vez que cada seqüência de estado pode ser analisada através da história que é escrita à medida em que ela aciona os agentes pertencentes à sociedade que se comunicam com o agente implementado por esta linguagem, a lógica temporal na qual a linguagem é baseada tem uma interpretação computacional [Barringer et al, 1989].

É importante salientar que a execução das especificações dos agentes só é possível, inicialmente, em uma linguagem cujos modelos nos quais a lógica temporal é baseada sejam comparativelmente simples, com uma interpretação computacional um tanto óbvia e intuitiva. Entretanto, linguagens para especificação de agentes em geral (como o JEOPS ou os formalismos de Rao e Georgeff em [Rao; Georgeff, 1995]) são baseadas em lógicas consideravelmente mais complexas.

Em particular, estas lógicas são usualmente baseadas em frameworks semânticos denominados de mundos possíveis [Chellas, 1980]. O principal problema é que as semânticas para os frameworks de mundos possíveis não têm uma interpretação computacional direta, de forma que não fica claro o que a “execução” de uma lógica baseada nestas semânticas de fato significaria. Em resposta a isso, pesquisadores

têm tentado desenvolver linguagens para especificação de agentes com uma base semântica simplificada, que possa ser interpretada computacionalmente.

4.3.2 Compilação de Especificações de Agentes

Uma alternativa para a execução direta é a compilação. Neste esquema, a partir de uma especificação abstrata, realiza-se a sua transformação para um modelo computacional concreto através de um processo automático de síntese. As principais vantagens da Compilação sobre a Execução são percebidas na eficiência em tempo de execução. A execução direta de uma especificação de um agente envolve a manipulação de uma representação simbólica em tempo de execução. Esta manipulação geralmente é mensurada por meio da análise de um formulário que serve como modelo para interpretação desta representação simbólica, o que é computacionalmente custoso. As abordagens de compilação tendem a reduzir especificações abstratas simbólicas para modelos computacionais muito mais simples, que não precisam de nenhuma representação simbólica. O trabalho de análise é, desta forma, feita em tempo de compilação: a execução do sistema compilado pode desta forma ser feita com pouca ou nenhuma análise simbólica.

As abordagens para compilação normalmente dependem da relação próxima entre os modelos para lógica temporal/modal (que são tipicamente grafos nomeados) e máquinas de estado finitas (no modelo de autômatos). Um exemplo muito conhecido para esta abordagem no desenvolvimento de agentes é o paradigma de autômatos situados de Rosenschein e Kaelbling [Rosenschein; Kaelbling, 1996], que usa uma lógica epistemática para especificar o componente de percepção de sistemas de sociedade de agentes.

Embora teoricamente atraente, a abordagem geral de síntese automática é limitada a um número considerável de situações. Primeiro, na medida em que a linguagem de especificação de agentes se torna mais expressível, então até a análise em tempo de compilação se torna muito cara. Segundo, os sistemas gerados desta forma não são capazes de aprender (ou seja, não são capazes de se adaptar a situações não previstas ou formular novas situações a partir de sua base de conhecimento em tempo de execução). E, finalmente, como na abordagem de execução direta, os frameworks para especificação de agentes tendem a não possuir nenhuma interpretação computacional concreta, o que torna tal síntese impossível.

CAPÍTULO 5

A ABORDAGEM DE MAPEAMENTO: UM ESTUDO DE CASO

Finalmente, neste capítulo 5 é apresentada a abordagem proposta neste livro e uma especificação para o módulo é proposta. Na seção 5.1, é realizada uma discussão acerca da abrangência desta abordagem em um sistema genérico de integração de dados. Na seção 5.2, fala-se pela primeira vez em Ambiente Comum da Federação, a denominação adotada para o ambiente criado por meio da sociedade de agentes, em nível local, para a realização das traduções diretas. Nas seções subsequentes é discorrido acerca de como o mapeamento deve ser realizado, de como dados em fontes de dados locais (seções 5.3 e 5.4) devem ser tratados para que a abordagem de tradução tenha sucesso na realização de suas atividades, e de como fontes de dados componentes da integração (seções 5.5 e 5.6) devem ser preparadas para integração já nos níveis inferiores do sistema de integração do qual esta abordagem fará parte.

5.1 Abrangência

Esta abordagem tem sido proposta com um objetivo básico: resolver tanto as pendências estruturais quanto semânticas d*e* se integrar diferentes fontes de dados componentes. Com tais propósitos, foi adotado como estratégia o uso de sociedades de agentes inteligentes e ontologias, para descreverem o domínio do problema. Para tornar possível a implementação do módulo que contenha os agentes para realizar mapeamentos e inferência, faz-se necessária a sugestão de uma arquitetura para que o módulo interaja com as camadas mais superiores do virtual sistema, que se preocupam com a transmissão e interligação dos dados que foram mapeados por esta abordagem. Esta arquitetura é descrita a seguir, sendo o sistema formado por três ambientes: o Ambiente Comum da Federação, o Ambiente de Manutenção e o Ambiente Gerador, cujos objetos são, respectivamente, prover acesso às fontes de dados a serem integradas, fazer manutenção nos esquemas tanto da federação quanto dos componentes de fonte de dados, e a integração das fontes de dados escolhidas por analistas/usuários, como mostrado na figura abaixo [Costa; Campos;

Souza, 2003]:

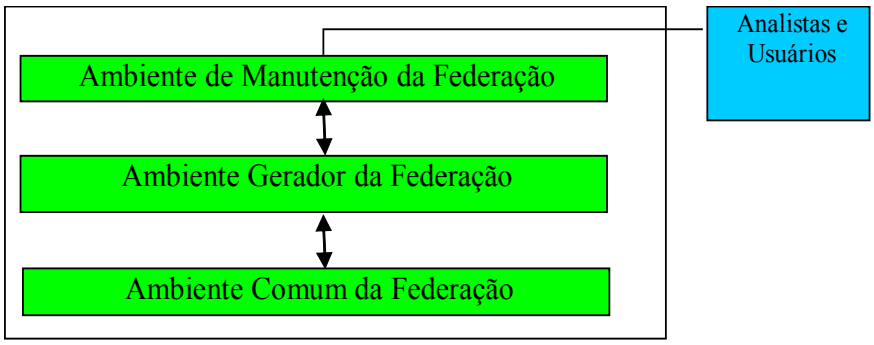

Figura 5.1*:* Ambientes do Sistema de Integração de Dados

Esta abordagem abrange o Ambiente Comum da Federação (ACF). O módulo ACF é apresentado abaixo, e é responsável por interpretar os dados presentes nas fontes de dados e organizá-los em modelos comuns de dados que serão saídas para as partes superiores do sistema. Este é o módulo que apresenta uma sociedade de agentes inteligentes para mapeamento entre fontes a partir de bases de conhecimento e motores de inferência a ser usado em sistemas de integração de dados.

Os agentes das diversas sociedades de agentes aqui propostas são baseados em conhecimento, isto é, conhecem o ambiente no qual estão inseridos. O conhecimento dos agentes é representado por objetos (entidades) de um dado domínio, com suas propriedades e relações. Os agentes sabem, entre outros: o atual estado do ambiente (propriedades importantes), a evolução do ambiente, como identificar o estado desejável do ambiente, como validar o resultado das ações e ainda possuem conhecimento sobre conhecimento (metaconhecimento). Estas informações estão representadas na base de conhecimento e o raciocínio dos agentes sobre suas possíveis ações é realizado com a ajuda de uma máquina de inferência. A base de conhecimento contém sentenças em uma linguagem de representação para fatos e regras, enquanto uma máquina de inferência é responsável por deduzir novos fatos ou hipóteses da base de conhecimento.

5.2 Ambiente Comum da Federação

O ACF tem por objetivo prover acesso às fontes de dados a serem integradas. É composto de uma SA denominada Mapeador das Fontes de Dados Locais (MFDL), a qual provê o acesso, por exemplo, um modelo UML ou um BDR. É no ACF que a abordagem de tradução de fontes de dados a serem integradas acontece.

O MFDL tem um papel fundamental para a abordagem, pois o mesmo é responsável pela transformação do modelo de dados nativo para o MDC e seu armazenamento no repositório de metadados; é responsável pela interligação dos diferentes modelos de

uma mesma fonte de dados, provendo desta forma, os metadados que são integrados nas fases seguintes.

Assim, o ACF interage com o AGF fornecendo informações sobre os esquemas das FDL (ver figura 5.2 abaixo [Costa; Campos; Souza, 2003]).

As seções 5.2, 5.3 e 5.4 abordam, respectivamente, as FDL, o MFDL e as FDC.

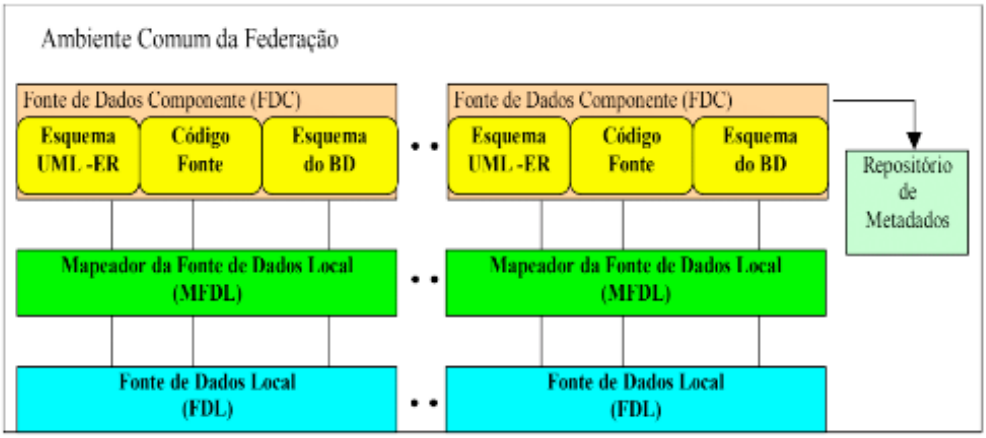

Figura 5.2: Ambiente Comum da Federação

5.3 Fontes de Dados Locais

As fontes de dados locais odem ser autônomas e heterogêneas. No caso desta abordagem, as fontes de dados seriam os esquemas de BD Relacional, OO, XML, UML, ER. Cada uma destas fontes é adicionada ao sistema através da SA MFDL, que funciona como uma ponte entre a fonte de dados e os outros componentes do sistema. Quando uma fonte de dados se junta ao sistema ela publica seu esquema componente, descrevendo as informações que podem estar disponíveis através dessa fonte de dados.

5.4 Mapeador de Fontes Locais

Para cada uma das fontes consideradas no sistema local se tem um agente responsável por sua tradução para um MDC. No caso das fontes UML e ER são feitas conversões para XMI. O esquema de BD é mapeado para CWM no formato XMI, pois o mesmo permite representar o sistema de tipos dos esquemas das FDL, a fim de facilitar o processo de integração destes esquemas. Todos os modelos convertidos para XMI são armazenados no repositório de metadados. O usuário somente define as fontes que deseja integrar, cabendo à SA MFDL executar a atividade de mapeamento das fontes de dados de forma automática.

Os agentes componentes do MFDL são descritos a seguir:

Agente Controlador – controla o funcionamento do MFDL. Seu comportamento é similar ao AC da ASA;

Agente Identificador – é responsável pela identificação das fontes de dados escolhidas pelo Analista/Projetista, isto é, se é um arquivo UML, ou um código fonte Java, ou um BD e também é responsável por indicar sua localização e como fazer para ter acesso a elas. A informação obtida é passada para o AC, que a encaminha para o agente responsável pelo mapeamento;

Agente XMI – manipula informações relacionadas às fontes de dados UML e ER. Este agente tem a capacidade de transformar estas fontes de dados para o formato XMI e armazenar esta informação em um repositório de metadados;

Agente BDR – manipula informações relacionadas a fontes de dados relacionais. Este agente tem a capacidade de transformar os esquemas destas fontes de dados para o formato de troca de CWM e armazenar esta informação em um repositório de metadados;

Agente BDOO – manipula informações relacionadas a fontes de dados OO. Este agente tem a capacidade de transformar os esquemas destas fontes de dados para o formato de troca de CWM e armazenar esta informação em um repositório de metadados;

Agente XML – manipula informações relacionadas a fontes de dados semi-estruturadas (XML). Este agente tem a capacidade de transformar os esquemas destas fontes de dados para o formato de troca de CWM e armazenar esta informação em um repositório de metadados.

A SA MFDL é constituída de cinco bases de conhecimento que auxiliam no processo de mapeamento das fontes de dados. A figura 5.3 abaixo [Costa; Campos; Souza, 2003] mostra os agentes e suas bases de conhecimento que compõem o MFDL, as quais serão detalhadas a seguir.

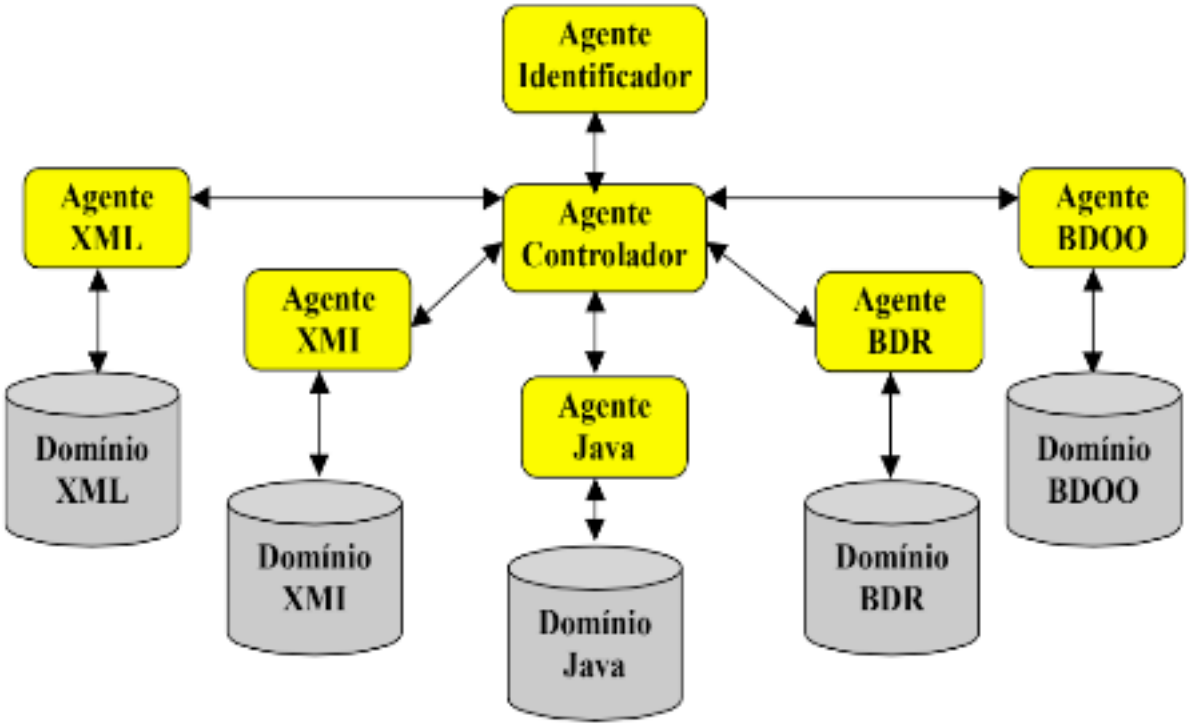

Figura 5.3: Mapeamento das Fontes de Dados Local

Domínio XMI – Todas as classes do metamodelo XMI são armazenadas como instâncias da classe *Class*, bem como todos os atributos e elementos são armazenados respectivamente como instâncias das classes *Attribute* e *Element* de XMI;

Domínio DBR – representa os esquemas das bases de dados relacionais através de um metamodelo, no caso o CWM. Captura as tabelas e seus relacionamentos, restrições de integridade, índices, *triggers* e *procedures*, entre outros;

Domínio DBOO – representa os esquemas das bases de dados OO através de um metamodelo, no caso o CWM. Captura as classes, atributos, tipos de atributos, restrições e referências, entre outros;

Domínio XML – representa o conjunto de definições e declarações, na forma de definição de tipos de elementos, através de um metamodelo, no caso o CWM. Captura atributos, conteúdo, elementos, conteúdo de elementos e tipos de elementos, entre outros.

5.5 Fontes de Dados Componentes

Nesta abordagem, as FDC são os esquemas de BD Relacional, OO, XML, UML e ER, todos no formato XMI. Elas são criadas através do agente MFDL. Assim, todos os modelos conceituais, esquemas de banco de dados de uma determinada fonte são adicionadas ao sistema no formato XMI.

A partir disto, pode-se ter uma visão interligada das fontes de dados do sistema local analisado. Toda ela está em um MDC, no caso XMI. A vantagem dessa abordagem é que se pode ter uma visão da análise e projeto e a visão do projeto da base de dados de forma integrada e interligada. Devido às associações de correspondências entre os objetos das diferentes fontes de dados, a modificação em um dos modelos reflete-se nos demais.

5.6 Integrador de Fontes de Dados Componentes

Dada a informação armazenada no repositório de metadados sobre os modelos, é necessário fazer a integração. Para este fim, tem-se o IFDC. Sua principal tarefa é a interligação dos diferentes modelos das fontes de dados armazenados no repositório de metadados, identificando as associações existentes entre atributos dos diferentes modelos, relacionamentos entre as classes pertencentes a eles e indicação das restrições de integridade existentes, entre outras atividades. Observe a figura 5.4 abaixo [Costa; Campos; Souza, 2003]. Todas as atividades são feitas de forma automática, não requerendo a intervenção do usuário.

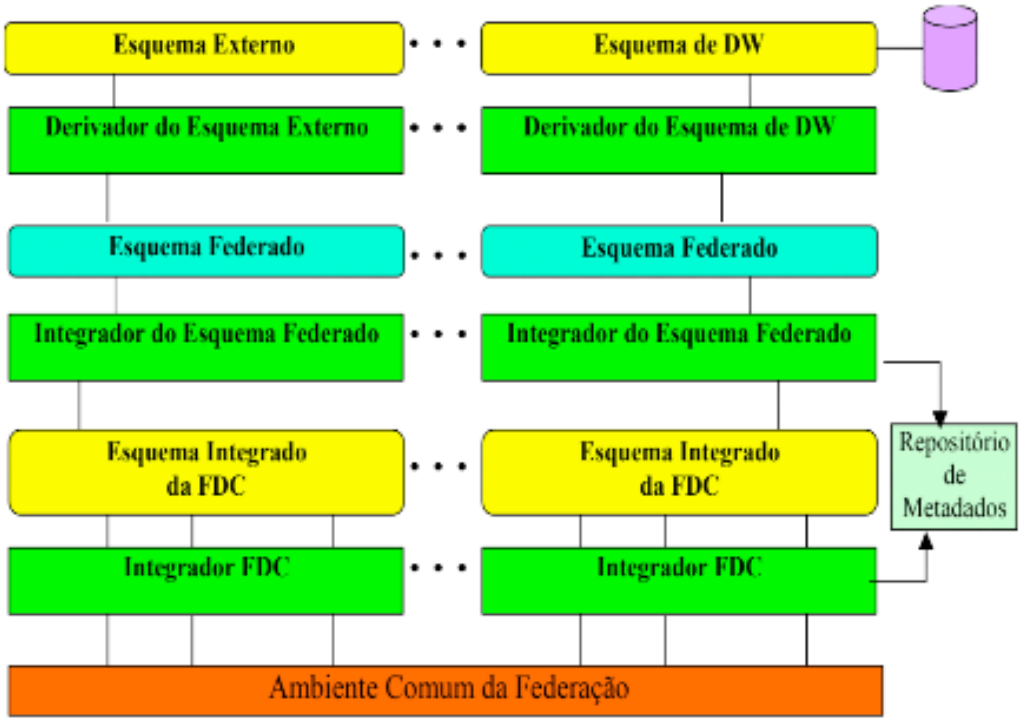

Figura 5.4: Acesso ao ACF

Por isso, esta abordagem propõe o IFDC, que tem como seus principais objetivos a integração automática de fontes de dados escolhidas pelos usuários, após ser estabelecido o acesso a diferentes fontes a serem integradas. Esta camada contém o

problema do conhecimento do domínio sendo integrado e também pode deduzir regras de validação do conhecimento pré-existente, ambos importantes, uma vez que não é possível analisar alguns aspectos de metadados ou ter ontologias para descrever a integração semântica destes aspectos sem eles.

Finalmente, é importante a construção de todos os esquemas da federação. Os componentes do IFDC são: Integrador do Esquema Federado, Interligador de Fontes de Dados Componentes e Derivador do Esquema Externo.

Este ambiente usa uma sociedade de agentes para realizar a tarefa da construção automática da federação, tendo como resultado um modelo federado das fontes de dados sendo integradas. Uma vez que não há intervenção humana durante o processo inteiro, a responsabilidade de confirmar o resultado da integração é dos agentes.

A visão dada pelo sistema para o analista pode ser tanto o esquema federado quando das fontes de dados componentes. Ambas as visões são armazenadas num repositório de dados e podem ser acessadas através de uma interface para o repositório baseado em JMI. Este sistema não opera nas fontes de dados com o objetivo de integrá-las, mas as mantém inalteradas, capturando seu esquema, incluindo todas as restrições das fontes, que são refletidas e mantidas no esquema federado.

A arquitetura do IFDC é baseada na arquitetura de cinco níveis para sistemas de gerenciamento de bancos de dados fortemente acoplados [Sheth; Larson, 1990], como mostrado na figura 2.2. O principal objetivo do Ambiente Comum da Federação é prover acesso às fontes de dados que serão integradas. É composto por uma sociedade de agentes chamada Mapeador de Fontes de Dados Locais, que possibilita o acesso a um modelo como UML ou a bancos de dados. O Mapeador de Fontes de Dados Locais tem um papel importante na abordagem, já que é responsável pela transformação de um modelo de dados nativo para o MDC, pelo seu armazenamento no repositório de metadados, e pela interligação de diferentes modelos de uma mesma fonte de dados, o que torna possível, assim, que os metadados sejam integrados nas fases posteriores. Desta forma, o Ambiente Comum da Federação interage com o Ambiente de Geração da Federação, disponibilizando informação nos esquemas de fontes de dados locais.

A SA utiliza ontologias, informações sobre metadados e XMI. As regras que permitem aos agentes indicarem as associações são baseadas tanto nas informações estruturais quanto na informação semântica. Desta forma, é obtido da FDC o relacionamento entre os modelos conceituais, BD através de XMI, armazenando o resultado em um repositório de metadados baseado em MOF. De forma similar, o processo se repete para as demais FDC. O usuário, no final do processo, uma vez resolvidas as heterogeneidades sintáticas e semânticas, tem como resultado uma visão integrada baseada em UML das FDC dos modelos conceituais e dos esquemas de banco de dados daquela fonte de dados sendo integrada. A validação do resultado gerado nessa fase somente é feita quando terminado todo o processo de integração. A figura 5.5 [Costa; Campos; Souza, 2003] mostra os agentes da SA IFDC.

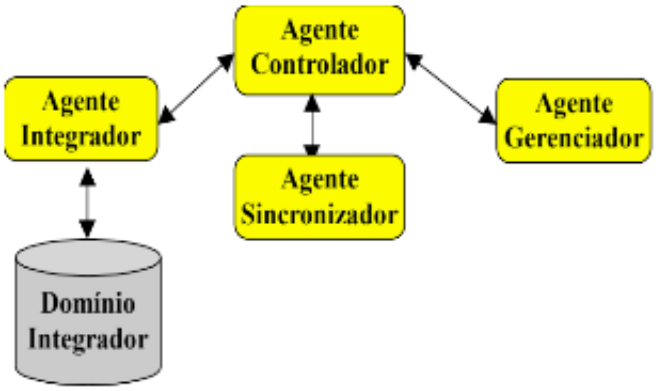

Figura 5.5: Integrador da Fonte de Dados Componentes.

Os componentes da SA responsáveis pela integração das FDC são detalhados a seguir.

Agente Controlador - controla o funcionamento do IFDC. Seu comportamento é similar ao AC da ASA;

Agente Integrador – sua função é interligar as diferentes fontes de dados que descrevem a mesma realidade, armazenadas no repositório de metadados, identificando associações de correspondência entre modelos conceituais, código fonte Java do modelo conceitual, os esquemas de BD e a indicação das restrições de integridade existentes. Este agente utiliza o AG para realizar a persistência da fonte de dados sendo integrada.

Domínio Integrador – representa o conjunto de definições estruturais e semânticas, na forma de definição de tipos de elementos, através de um metamodelo.

Agente Sincronizador - É responsável pela comparação dos modelos UML, Java e esquemas de BD, identificando as diferenças entre eles. Essa comparação permite que seja feita uma sincronização entre o modelo UML e o esquema de BD. A informação obtida é passada para o AC, que encaminha para o agente responsável pelo gerenciamento do repositório de metadados.

Agente Gerenciador – lida com o acesso ao repositório de metadados. Desta forma, tem a responsabilidade de inserir, remover e atualizar dados no repositório de metadados.

CAPÍTULO 6

CONCLUSÕES E TRABALHOS FUTUROS

Neste capítulo, o finalizamos este livro. São resumidas as contribuições (seção 6.1) deste trabalho enquanto abordagem de integração inteligente para sistemas de integração e de como conceitos aqui abordados podem ser utilizados no desenvolvimento de tais sistemas. Na seção 6.2, algumas propostas para trabalhos futuros são expostas, dando especial foco à extensão do módulo proposto no capítulo 5 para se adequar a sistemas que têm necessidade de manter dados replicados em federações ou redes, que é a utilização imediata da abordagem aqui proposta em trabalhos acadêmicos futuros. Na seção 6.3 são apresentadas considerações finais para o livro.

6.1 Contribuições

Neste livro, foi exposto como a Inteligência Artificial contemporânea pode contribuir com a necessidade recorrente das organizações por Integração de dados, por meio de uma proposta de arquitetura para um módulo de software, que foi denominado de Ambiente Comum da Federação. Esse módulo deve levar em consideração as fases de tradução e mapeamento das diferentes fontes de dados em um sistema distribuído com fontes de dados em diversos formatos, para um formato único, que possa ser integrado e apresentado ao usuário por meio de consultas, e utilizado por meio de operações em aplicativos e ferramentas que façam parte do sistema distribuído.

Foi proposta para tal ambiente uma arquitetura inteligente e otimizada, e a utilização de agentes baseados em conhecimento é de fundamental importância para que o processo de automatização e estabelecimento de regras para integração seja bem-sucedido. As atividades envolvidas no processo de mapeamento de tais fontes são extremamente complexas e envolvem integração de esquemas, interligação de relevância, análise textual de diversos tipos (incluindo análise semântica das fontes), e, para discernir o tipo de análise que deve ser feita tomando por base as preferências pessoais de cada usuário, presente na base de conhecimento, o sistema demanda diversos tipos de agente com atividades específicas.

É inevitável que haja uma comunicação efetiva e programada entre esses agentes inteligentes – comunicação essa que, numa sociedade de agentes típica deve ser feita através de troca de mensagens, caracterizando verdadeiramente o conceito de comunidade inteligente, em uma visão horizontal da arquitetura do módulo proposto. Desta forma, o mecanismo tradicional pouco flexível, que faz uso de motores de inferência na fase de tradução nos sistemas federados mais comuns, é substituído por uma forma mais ágil de filtragem de resultados, que pode considerar mecanismos de aprendizagem na melhoria contínua da qualidade dos dados da federação.

Diversas ferramentas foram utilizadas para a construção do estudo de caso da sociedade de agentes, cujo conceito tem sido introduzido e melhorado ao longo da evolução de sistemas inteligentes na medida em que ferramentas, como o JADE, têm sido desenvolvidas e utilizadas em sistemas como esses. Tais ferramentas tipicamente propõem um ambiente *(framework)* para desenvolvimento e comunicação de agentes, o que facilita a definição de uma arquitetura para módulos que realizem esta comunicação para quaisquer fins, e simplifica a implementação de sistemas multi-agentes ou neurais através de um *middleware* de desenvolvimento que suporta as fases de validação e implantação de tais sistemas.

Em uma visão verticalizada, este módulo se encaixa na fase de leitura das fontes de dados após uma consulta ter sido feita, e na posterior fase de tradução *(wrapping)* dos resultados para um modelo de dados comum. Qualquer sistema de integração de dados pode utilizar este módulo e seus modelos – ou estendê-lo para que seja possível comportar mais fontes – para a realização da atividade de tradução.

Sob uma realidade de federação de dados integrados, a arquitetura aqui proposta possibilita uma interface para sistemas que possuam arquitetura federada fortemente acoplada, permitindo que as fontes de dados mantenham sua autonomia e suas restrições sejam impostas sobre o esquema da federação. Então, é inevitável pensar em uma visão futura, quando seria observada a possibilidade de basear esta abordagem em um processo de integração semântica, utilizando ontologias, dicionários de dados, propriedades estruturais, hierarquias e valores de dados derivados dos esquemas de dados.

Um exemplo direto em que a utilização de tal abordagem pode ser estendida, em termos de projetos futuros, seria no possível desenvolvimento de ambientes para replicação inteligente de dados de diversas fontes em formatos e locais heterogêneos, que podem envolver fases de mapeamento e tradução em suas camadas inferiores. Tais módulos podem ser utilizados por organizações para mapear fontes de dados dos mais variados formatos, tais como Oracle, MySQL, PostgreSQL XML, SQL Server, JavaDB, entre outros, para o formato comum na integração de esquemas, XMI MOF.

6.2 Uma Proposta para Replicação

Conforme explicado na seção anterior, um dos possíveis trabalhos futuros a serem realizados diz respeito à implementação de uma ferramenta em um projeto de replicação de dados. Obviamente esta ferramenta tem utilidade adequada para este tipo de técnica, uma vez que a replicação de dados pode ser realizada por um esquema [Sheth; Larson, 1990] em bancos de dados federados e distribuídos com múltiplas fontes.

A replicação de dados é útil para melhorar a disponibilidade dos mesmos. A técnica torna possível armazenar determinados dados em mais de um *site*, durante o processo de projeto de banco de dados distribuídos. O princípio e o objetivo básicos de replicação é tornar independente a disponibilidade dos dados e de determinados sites. Ou seja, devem-se manter os dados disponíveis na rede de bancos de dados distribuídos, independente do fato de seu local de origem (no caso, a fonte original onde os dados foram inicialmente gerados) estar disponível ou não. Deve-se manter os dados permanentemente na rede.

O caso mais extremo de replicação é a replicação de todo o banco de dados em todos os sites do sistema distribuído, o que se convenciona denominar de "caso extremo de um banco de dados totalmente replicado" [Kim, 1995]. Nessa situação, melhora-se bastante a disponibilidade dos dados, uma vez que o sistema pode continuar a operar enquanto pelo menos um site estiver ligado, bem como o desempenho de recuperação para consultas globais, pois o resultado deste tipo de consulta pode ser obtido localmente a partir de qualquer site individual – uma consulta de recuperação pode ser processada, portanto, no site local no qual é submetida, desde que o site inclua um módulo de servidor, o que é requisito em sistemas de bancos de dados distribuídos ou federações de dados.

No entanto, este tipo de replicação apresenta inúmeras desvantagens no que diz respeito às operações de atualização, que ficam consideravelmente desaceleradas, já que uma única atualização lógica deve ser replicada em todas as cópias do banco de dados, para que sejam mantidas consistentes. Se existirem muitas cópias do banco de dados e muitos usuários utilizando tais cópias simultaneamente, um problema potencial de controle de versão pode ocorrer e os aspectos mais desafiadores da gestão da configuração que dizem respeito à desaceleração se tornarão ainda mais evidentes, já que a replicação total torna as técnicas de recuperação e concorrência no banco de dados distribuído mais caras do que seriam se não houvesse replicação.

O outro extremo da replicação total é não haver nenhuma replicação – o banco de dados é formado por dados, em diversos sites, que estão fragmentados unicamente e exatamente em um único site. O banco de dados pode ser visto como um banco de dados com fragmentos disjuntos, uma vez que os dados nos diversos sites "se completam". Esse tipo de abordagem é denominado por Navathe e Savaseri [Navathe;

Savaseri, 1996] de “alocação não-redundante”.

Entre esses dois extremos, existe um amplo espectro de replicação parcial de dados. Na replicação parcial, alguns fragmentos do banco de dados podem ser replicados, enquanto outros não. O número de cópias de cada fragmento pode abranger de um ao número total de sites no SBDD. Um caso típico da replicação parcial de dados ocorre quando diversos participantes do SBDD, que se movimentam sem participar permanentemente da federação, levam consigo partes do BD parcialmente replicados em dispositivos móveis (como smartphones ou laptops) e os sincronizam periodicamente com o sistema distribuído (ou banco de dados do servidor).

A descrição da replicação destes fragmentos é o que se chama de esquema de replicação.

Em um sistema distribuído, cada fragmento (ou cada cópia de fragmento) pode ser designado a um determinado site pertencente a ele. Este processo denomina-se alocação de dados [Kim, 1995]. Para se realizar a escolha destes sites e o grau de replicação nos dados a serem replicados, serão analisadas metas de desempenho, disponibilidade do sistema, tipos de transações submetidas a cada site e freqüência com que estas transações são submetidas. Por exemplo, se determinadas transações que acessam permanentemente determinadas partes do banco de dados forem submetidas, em sua maioria, a um determinado site, o conjunto de fragmentos correspondente a estas transações pode ser alocado apenas neste site.

A utilização da abordagem proposta neste trabalho, em um sistema de replicação de dados, seria extremamente útil na situação em que se realiza mapeamento de fontes que estão replicadas para o formato utilizado na federação por diversas vezes. É o caso de muitas atualizações serem realizadas. Normalmente a atitude mais recomendável seria limitar a replicação, já que encontrar uma solução ótima para alocação de dados distribuídos é um problema complexo de otimização. Esta abordagem tornaria possível utilizar os resultados de um mapeamento e replicá-los, acelerando o processamento das consultas e a atualização dos dados.

6.3 Considerações Finais

Neste capítulo foram apresentadas as principais contribuições trazidas para os sistemas de computação e sistemas distribuídos – em uma abordagem visivelmente direcionada para sistemas de bancos de dados e armazenamento de informações – por este livro, ao apresentar uma abordagem para tradução de fontes de dados para um formato comum por meio de um módulo que tem como ambiente de atuação uma sociedade de agentes independentes entre si, mas interdependentes na realização de tarefas.

A principal contribuição científica deste estudo e da abordagem aqui proposta, conforme já frisado anteriormente, se dá na área de sistemas de integração de dados

distribuídos e heterogêneos. A abordagem traz contribuições também para as áreas de sistemas inteligentes e de integração de esquemas, cuja tecnologia utilizada é invariavelmente análoga a de sistemas de bancos de dados distribuídos e heterogêneos.

A até então conhecida originalidade desta abordagem, na qual foram unidos diversos elementos presentes isoladamente em outras abordagens, metodologias e propostas, e também a contribuição desta pesquisa para se promover um ambiente de desenvolvimento e comunicação extensível e sua representação em sistemas distribuídos são os pontos-chave deste livro.

As vantagens de se utilizar os padrões adotados por esta abordagem foram largamente discutidas. XMI demonstrou ser uma linguagem única para a representação de metamodelos e, em associação com o padrão MOF, baseado em XMI, as vantagens de se utilizar esta linguagem nas diversas etapas da integração de dados em sistemas distribuídos são inúmeras.

O padrão de agentes adotados já foi consolidado em diversos trabalhos como o JEOPS e o JADE. A sociedade de agentes se mostrou como uma arquitetura eficiente para evitar gargalos na aplicação e prover o acesso eficiente aos dados. A possibilidade de aplicação dessa proposta em organizações cujos dados se encontram heterogeneamente distribuídos é de alto potencial, e projetos futuros que abordem o tema de replicação de dados estão diretamente relacionada com o fato de a abordagem fazer uso de agentes para a análise dos dados e visão computacional.

REFERÊNCIAS

[Barringer et al, 1989] Barringer, H. et al. MetateM: A framework for programming in temporal logic. In: REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness (LNCS Volume 430). Springer-Verlag: Berlin, Germany, 1989.

[Batini; Lenzerini; Navathe, 1986] Batini, C.; Lenzerini, M.; Navathe, S. B.. A Comparative Analysis of Methodologies for Database Schema Integration. Database Systems Research and Development Center, Computer and Information Sciences Department, University of Florida, Gainesville, Florida, 1986.

[Baumgartner; Flesca; Gottlob, 2001] Baumgartner, R.; Flesca, S.; Gottlob, G. Visual Web Information Extraction with Lixto, Proc. of VLDB, 2001.

[Booch; Jacobson; Rumbaugh, 1998] Booch, G.; Rumbaugh, J.; Jacobson, I.. UML: guia do usuário. Editora Campus, 1998.

[Bray; Paoli; Sperberg-McQueen] Bray, T.; Paoli; J.; Sperberg-McQueen, C. M.. Extensible Markup Language (XML) 1.0, 1998. World Wide Web Consortium. http://www.w3.org/TR/REC-xml

[Brownell, 2002] Brownell, D.. SAX2, O'reilly, 2002.

[Bryan, 1998] Bryan, M.. *SGML: An Author's Guide to the Standard Generalized Markup Language*. Wokingham/Reading/New York: Addison-Wesley, 1988.

[Chellas, 1980] Chellas, B.. Modal Logic: An Introduction. Cambridge University Press: Cambridge, England, 1980.

[Ciferri; Souza, 2000] Ciferri, C. D. A.; Sousa, F. F.. Distribuindo os dados do Data Warehouse In: Anais do XV Simpósio Brasileiro de Banco de Dados, 2000. Pg. 346-360. PUC – RS: CEFET-PB, João Pessoa.

[Cohen; Levesque, 1990] Cohen, P. R.; Levesque, H. J.. Intention is choice with commitment. Artificial Inteligence, 42:213-261, 1990.

[Costa; Campos; Souza, 2002] Costa, R; Campos, V. V. de S.; Souza, F. da F. de. A.. Integração de Fontes Heterogêneas de Dados utilizando a Web e XMI. In: CSITeA'02. Foz do Iguaçu – Brazil, 2002.

[Costa; Campos; Souza, 2003] Costa, R; Campos, V. V. de S.; Souza, F. da F. de. A.. Bulding Database Federations using Agents. In: IFIP'03. Rio de Janeiro – Brazil, 2003.

[Crary; Weirich, 1999] Crary, K.; Weirich, S., Flexible type analysis (extended version). Technical report, Cornell University, 1999.

[Dennet, 1987] Dennett, D. C.. The Intentional Stance. The MIT Press: Cambridge, MA, 1987.

[Fisher, 1997] Fisher, M.. An alternative approach to concurrent theorem proving. In: J. Geller, H. Kitano, and C. B. Suttner, editors, Parallel Processing in Artificial Intelligence 3. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1997.

[Fisher; Wooldridge, 1997] Fisher, M.; Wooldridge, M.. On the formal specification and verification of multi-agents systems. International Journal of Cooperative Information Systems. 1997.

[Gray, 1996] Gray, R.. Agent Tcl: A flexible and secure mobile-agent system. In: The Fourth Annual Tcl/Tk Workshop Proceedings. The USENIX Association, 1996..

[Grose, 2001] Grose, T.. XMI Application Framework, 2001. Disponível em: http://www.alphaworks.ibm.com/tech/xmiframework

[Heuser, 1998] Heuser, C. A.. Projeto de banco de dados. 1ª. edição. Sagra Luzzatto, 1998. Pg. 11-39.

[Inmon, 1997] Inmon, W. H.. Como construir um Data Warehouse. Rio de Janeiro: Editora Campus, 1997.

[Kim, 1995] Kim, W.. Modern Databases Systems – Object Model, Interoperability, and Beyond. Addison-Wesley, 1995.

[Melo; Silva; Tanaka, 1997] Melo, R. N.; Silva, S. D. Da; Tanaka, A. K.. Bancos de dados em aplicações cliente/servidor. Rio de Janeiro: Infobook, 2ª. Tiragem, 1997.

[MOF, 2002] OMG – Object Management Group. MetaObject Facility (MOF) Specification 1.4. Especificação, 2002.

[Navathe; Savasere, 1996] Navathe, S. B.; Savasere, A.. A Schema Integration Facility using Object-Oriented Model, In: Object Oriented Multidatabase Systems, Cap. 04, Prentice Hall, 1996.

[Pedersen, 2000] Pedersen, T. B. et. al.. Aspects of Data Modeling and Query Processing for Complex Multidimensional Data. Ph.D. Dissertation Faculdade de Engenharia e Ciências, Universidade de Aalborg, Alemanha, 2000.

[Rao; Georgeff, 1995] Rao, A. S.; Georgeff, M.. BDI Agents: from theory to practice. In: Proceedings of the First International Conference on Multi-Agents Systems (ICMAS-95). San Francisco, CA, 1995.

[Ribeiro; Florentini, 2000] Ribeiro, A. C.; Florentini, A. C.. O padrão XMI: uma proposta para sua utilização em Bibliotecas Digitais, 2000, disponível em http://genesis.nce.ufrj.br/dataware/

[Rosenschein; Kaelbling, 1996] Rosenschein, S. J.; Kaelbling, L. P.. A situated view of representation and control. In: P. E. Agre and S. J. Rosenschein, editors, Computational Theories of Interaction and Agency, The MIT Press:
Cambridge, MA, 1996.

[Salgado; Lóscio, 2000] Salgado, A. C.; Lóscio, B. F.. Integração de Dados na Web. In: Anais da XX Jornada de Atualização em Informática, 2000..

[Samos et al, 1998] Samos, J. et. al.. *Database Architecture for Data Warehousing: An Evolutionay Approach. In:* Proceedings 9th International Conference on Database and Expert Systems Applications, DEXA'98, Vienna, Austria, Springer LNCS 1460, 1998, pp. 746-756.

[Schildt, 1996] Schildt, H.. C, completo e total, tradução de Roberto Carlos Mayer. São Paulo: Makron Books, 1996.

[Sheth; Larson, 1990] Sheth, A. P.; Larson, J. A.. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. Computing Surveys, 1990, 22(3):183-236.

[Spaccapietra; Parent, 1994] Spaccapietra, S.; Parent, C.. View Integration: A Step Forward in Solving Structural Conflicts, In: IEEE Trans. On Software Eng., vol 6, No. 2, 1994..

[Wood et al, 1998] Wood, L. et al, Document Object Model (DOM) Level 1 Specification Version 1.0. World Wide Web Consortium, 1998. http://www.w3.org/TR/REC-DOM-Level-1

[Wooldridge, 1995] Wooldridge, M.. Agents and Software Engineering. AI*IA Notizie, 1995.

[Wooldridge; Jennings, 1995] Wooldridge, M.; Jennings, N.R. Intelligent agents: Theory and practice. The knowledge Engineering Review, 1995.

[Vinoski, 1997] Vinoski, S.. CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments. IEEE Communications Magazine, vol. 14, no. 2, February 1997.

[XMI, 2002] OMG – Object Management Group. XML Metadata Interchange (XMI) Specification version 1.2, February 02, 2000. http://www.omg.org.

SOBRE O ORGANIZADOR

RODRIGO ALVES COSTA atualmente (2018) é professor associado do curso de Ciência da Computação da Universidade Estadual da Paraíba (UEPB). Possui doutorado em Ciência da Computação pela UFPE (2016), mestrado pela mesma instituição (2010), MBA em Gerenciamento de Projetos pela Fundação Getúlio Vargas (2007) e graduação em Ciência da Computação pela UFPE (2005). É certificado Project Management Professional (PMP) pelo Project Management Institute (PMI) (2006). Tem vasta atuação profissional no mercado de trabalho, destacando-se em funções como gerente de projetos, analista de sistemas, engenheiro de software e analista de testes e de segurança em empresas como IBM, Siemens, Motorola e C.E.S.A.R. É autor de livros na área de gerenciamento de projetos, marketing digital, segurança da informação e engenharia de software, e atua como consultor e idealizador de qualificações na área de planejamento e governança estratégica em tecnologias da informação em diversas organizações públicas e privadas, incluindo instituições de ensino, sendo um dos fundadores do currículo de Governança de TI da Escola Superior de Redes (ESR), vinculada à Rede Nacional de Pesquisa (RNP), com o livro Gerenciamento de Projetos de TI. Foi bolsista do CNPQ durante o doutorado, mestrado e a graduação (PIBIC), e atualmente está vinculado aos diretórios de pesquisa da entidade, nos grupos de pesquisa em Segurança Computacional, da UFPE, em Gestão, Comportamento e Competências Organizacionais, do IFPB, e no grupo ATLAS - Ação das Tecnologias na Aprendizagem Significativa, da UEPB. Pesquisador na área de Ciência da Computação, com ênfase em segurança computacional e sistemas de informação, e na área de Gerenciamento de Projetos, com ênfase em gerenciamento de projetos de tecnologia da informação, abordagens ágeis de projeto, agilidade organizacional e marketing organizacional por projetos. É membro entusiasta do PMI, da Association for Computing Machinery (ACM) e da Sociedade Brasileira de Computação (SBC).

Agência Brasileira do ISBN
ISBN 978-85-7247-149-7